

Physics Simulation Lab Final Report (2005 - 2020)

Molecular Modeling Basics and Protein Side Chain Placing Code (2005 – 2011)

(Computational Fluid Dynamics (CFD) Basics) (2010 - 2015) اساسيات ديناميكيات الموائع الحسائية (د.م.ح.)

MEAE-CFDNC Computational Fluid Dynamics and Numerical Combustion Code (2019)

IAP Supernova Simulation Code (2019)

IAP-PSC Plasma Simulation Code (Particle-in-Cell Code) (2019)

INT-LMIC (Laser Matter Interaction Code) (2020)

Project Manager:

Dr. Samir Mourad

With contributions of:

Dr. Samir Mourad, M.Sc. Fatima Hamed, M.Sc. Banan Kerdi, M.Sc. Ahlam Houda, M.Sc. Samar Bakoben, M.Sc. Mariam Abdelkarim, M.Sc. Abdurrahman Ibrahim

Last Update: 08.11.2020 16:23

1	MOLECULAR MODELING BASICS	
2	SOME CONCEPTS IN MOLECULAR MODELING	1
5	SURFACES/مساحات السطح	1.1
6	COMPUTER HARDWARE AND SOFTWARE/أجهزة وبرمجيات الكمبيوتر	1.2
7	UNITS OF LENGTH AND ENERGY/وحدات الطول والطاقة	1.3
8	MATHEMATICAL CONCEPTS/المفاهيم الرياضية	1.4
9	REFERENCES/المراجع	1.5
10.....	COMPUTATIONAL QUANTUM MECHANICS / معلوماتية ميكانيكا الكم	2
10	INTRODUCTION /مقدمة	2.1
12.....	Operators /المشغّلون	2.1.1
14.....	Atomic Units /وحدات الذرة	2.1.2
15	ONE-ELECTRON ATOMS	2.2
18	POLYELECTRONIC ATOMS AND MOLECULES/إلكترون متعدد الذرات والجزيئات	2.3
20.....	The Born-Oppenheimer Approximation/مقارنة بورن-أوبنهايمر	2.3.1
20.....	General Polyelectronic Systems and Slater Determinants /نظمة الإلكترون المتعدد العامة و محددات سلاتر	2.3.2
23	MOLECULAR ORBITAL CALCULATIONS /حسابات المدار الجزيئي	2.4
23.....	The Energy of a General Polyelectronic System/ الطاقة للنظام الإلكتروني المتعدد العام	2.4.1
	Calculating the Energy from the Wavefunction: The Hydrogen Molecule /حزبيء الهيدروجين	2.4.2
		27
32.....	The energy of a Closed-shell System/طاقة نظام الطبقة المغلقة	2.4.3
32	THE HARTREE-FOCK EQUATIONS/معادلات هارترى-فوك	2.5
33.....	Hartree-Fock calculations for Atoms and Slater's Rules/حساب الهارترى-فوك للذرات وقواعد سلاتر	2.5.1
36.....	Linear Combination of Atomic Orbitals (LCAO) in Hartree-Fock Theory/التوافق الخطي لمدارات الذرة في نظرية هارترى-فوك	2.5.2
37.....	Closed-shell Systems and the Roothaan-Hall Equations/نظام الطبقة المغلقة ومعادلات روثان-هال	2.5.3
37.....	Solving the Roothaan-Hall Equations / حل معادلات روثان-هال	2.5.4
39.....	A Simple Illustration of the Roothaan-Hall Approach/توضيح بسيط لمنهج روثان-هال	2.5.5
44	BASIS SETS /أسس المجموعات	2.6
45.....	MONTE CARLO SIMULATION METHODS:/ أساليب محاكاة مونتي كارلو	3
45	INTRODUCTION:/المقدمة	3.1
47	CALCULATING PROPERTIES BY INTEGRATION:/خصائص الحساب بالتكامل	3.2
48	SOME THEORETICAL BACKGROUND TO THE METROPOLIS METHOD:/بعض الخلفية النظرية لطريقة متروبوليس	3.3
54	IMPLEMENTATION OF THE METROPOLIS MONTE CARLO METHOD:/تطبيق أسلوب متروبوليس مونتي كارلو	3.4
57.....	Random Number Generators:/العشوائية المولدات الكهربائية للاعداد	3.4.1
62	MONTE CARLO SIMULATION OF MOLECULES:/ محاكاة مونت كارلو للجزيئات	3.5

63.....	الجزيئات الصلبة <i>Rigid Molecules/</i>	3.5.1
67.....	محاكاة مونت كارلو للجزيئات المرنة <i>Monte Carlo Simulations of Flexible Molecules: /</i>	3.5.2
69.....	النماذج المستخدمة في محاكاة مونت كارلو من البوليمار / <i>MODELS USED IN MONTE CARLO SIMULATION OF POLYMERS/</i>	3.6
71.....	نماذج شبكة البوليمار <i>Lattice Models of Polymers</i>	3.6.1
80.....	'Continuous' Polymer Models/	3.6.2
84.....	DICTIONARY ENGLISH-ARABIC FOR MOLECULAR MODELING	4
90.....	PROTEIN SIDECHAIN PLACING	
92.....	INTRODUCTION	5
92.....	TASK TO BE USED FOR PHD THESIS	5.1
92.....	FRÜHERE ARBEITEN IM UMFELD	5.2
92.....	SCWRL 3.0	5.2.1
92.....	Rotamer library	5.2.1.1
92.....	Energy Function	5.2.1.2
92.....	Input and output	5.2.1.3
93.....	MATHEMATICAL METHODS	6
93.....	INTEGER OPTIMIZATION	6.1
93.....	RELAXATION WITH LAGRANGIAN MULTIPLIERS	6.2
94.....	ROTAMERS AND ROTAMER LIBRARY	7
95.....	MOLECULAR DOCKING	8
	FROM ALGORITHMICAL STANDPOINT THE MOLECULAR DOCKING PROBLEM CAN BE CONCERNED THE SAME AS THE SIDECHAIN OPTIMIZATION	8.1
95.....	PROBLEM (SCP)	
	ERZEUGUNG VON SELBST- UND WECHSELENERGIEN VON ROTAMER-ZUSTÄNDEN DER RESIDUEN EINES PROTEINS: MIT	9
97.....	DEAD-END ELIMINATION	
99.....	BIOINFORMATICAL METHODS	10
99.....	BALL	10.1
99.....	<i>BoundingBoxProcessor</i>	10.1.1
99.....	<i>Grid Box Class</i>	10.1.2
101.....	DIE BIBLIOTHEK DOCKING_TOOLS	10.2
102.....	<i>DEE.C und DEE_complete.C</i>	1)
102.....	<i>formats (Ordner) -> dort ist docking.ps (eine Übersichtgraphik)</i>	2)
102.....	<i>structure_generator.C</i>	3)
103.....	<i>amber_energy.C</i>	4)
103.....	<i>docking_grid.C</i>	5)
105.....	<i>greedy_tree.C</i>	6)
107.....	<i>PDB_checker.C</i>	7)
107.....	<i>transform.res</i>	8)
107.....	<i>basicTree.h</i>	9)
107.....	<i>energy.C</i>	10)

108.....	<i>hydrogen_add.C</i>	11)
108.....	<i>util.h</i>	12)
108.....	<i>candidate_generator.C</i>	13)
108.....	<i>energy_flex.C</i>	14)
108.....	<i>protein_mapper.C</i>	15)
108.....	<i>DEE.C</i>	16)
108.....	<i>FDPB.C</i>	17)
108.....	<i>optimizer.C</i>	18)
108.....	<i>selection.h</i>	19)
109.....	SIDE CHAIN OPTIMIZATION WITH LAGRANGIAN MULTIPLIERS	11
109.....	VERSION 1	11.1
124.....	GMEC VERSION 2	11.2
126.....	GMEC VERSION3	11.3
126.....	<i>GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile</i>	11.3.1
134.....	GMEC VERSION4_MITBALL	11.4
134.....	<i>Benutzer Code „docking tools“</i>	11.4.1
134.....	<i>Gesamtprogramm:</i>	11.4.2
134.....	<i>Eingabe: PDB-File</i>	11.4.3
134.....	<i>Berechnung des Energiefiles</i>	11.4.4
135.....	<i>GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile</i>	11.4.5
135.....	<i>Strukturzeugung</i>	11.4.6
136.....	ERGEBNISSE TESTSETS	12
137.....	IMPROVEMENT OF ENERGY FUNCTION	13
137.....	EMPERICAL FORCE FIELD MODELS: MOLECULAR MECHANICS	13.1
137.....	<i>Introduction</i>	13.1.1
137.....	<i>A simple Molecular Mechanics Force Field</i>	13.1.2
141.....	<i>Potential energy functions</i>	13.1.3
142.....	<i>The AMBER force field</i>	13.1.4
144.....	THE CHARMM FORCE FIELD	14
144.....	THE FORCE FIELD USED BY SCWRL 3.0	14.1
145.....	LITERATURVERZEICHNIS	15
145.....	ANHANG A: PROGRAMMCODE FÜR GMEC_LR_SCP	15.1
145.....	UTIL.H	15.2
146.....	DEE_COMPLETE_SCP.C	15.3
170.....	GMEC_LR.H	15.4
181.....	GMEC_LR_SCP.C	15.5
196.....	STRUCTURE_GENERATOR_SCP.C	15.6
209.....	SCP.SH	15.7

210	ANHANG B: ZEITPLAN, ARBEITSPAKETE UND TATSÄCHLICHE ARBEITSSTUNDEN	16
210	ZEITPLAN (ERSTELLT SEPTEMBER 2005)	16.1
211	BESPRECHUNGEN	16.2
218	COMPUTATIONAL FLUID DYNAMICS (CFD) BASICS WITH EXAMPLES (ENGL./ARAB.) (2010 - 2015)	
227.....	علم الفلك	•
227.....	محارق للنفايات: المحاكاة CFD، تكون لمعرفة توزيع درجة الحرارة في المحرقة	•
228.....	محارق صواريخ	•
228.....	المركبات الفضائية	•
229	مدخل الى ديناميكيات الموائع والغازات (FLUID AND GAS DYNAMICS)	17
229	تعريفات اساسية	17.1
229	نظام الوحدات	17.2
229	مضمون القسم الأول من الكتاب	17.3
230	الموائع (FLUIDS)	17.4
230	الكمية المتصلة	17.5
230	الكثافة	17.6
231	الكثافة النسبية	17.7
231	قانون الغاز المثالي (IDEAL GAS)	17.8
231	الجريان المستقر (STEADY FLOW)	17.9
231	اجريان المنتظم (UNIFORM FLOW)	17.10
231	خط الانسياب (STREAMLINE)	17.11
231	أبعاد السريان (DIMENSIONS OF FLOW)	17.12
232	الاجهاد (STRESS)	17.13
232	التدفق الصفائحي (LAMINAR FLOW) التدفق المضطرب (TURBULENT FLOW)	17.14
232	المنظومة وحجم التحكم عنصر مائع لا متناهي الصغر	17.15
233	الضغط المقياسي	17.16
233	القوة الجسمية والقوة السطحية	17.17
234	الاجهاد القصي	17.18
235 (GOVERNING EQUATIONS OF FLUID DYNAMICS)	المعادلات الأساسية في ميكانيك الموائع	18
235	مدخل	18.1
235.....	متجه السريان	18.1.1
236	الاشتقاق الكبير (THE SUBSTANTIAL DERIVATE)	18.2
239	المعنى الفيزيائية من تباعد السرعة $\nabla \cdot \vec{V}$ (DIVERGENCE OF VELOCITY)	18.3
239	حفظ الكتلة (MASS CONSERVATION)	18.4
241	معادلة الاستمرارية (continuity equation)	18.4.1
242	حفظ الطاقة (ENERGY CONSERVATION)	18.5
246	حفظ كمية التحرك (MOMENTUM CONSERVATION)	18.6
246	تلخيص المعادلات الاساسية (GOVERNING EQUATIONS) لديناميك الموائع مع ملاحظات	18.7
	معادلات السريان اللزجي (viscous flow) دون النظر الى تفاعلات الكيميائية (without)	18.7.1
	246 (considering chemical reactions)	

معادلات السريان الا لزجي (inviscid flow) دون النظر الى تفاعلات الكيمائية (without)	18.7.2
250 (considering chemical reactions)	
251..... تعليقات على المعادلات الاساسية	18.7.3
252..... الحالات الجدارية (boundary conditions)	18.7.4
اشكال للمعادلات الاساسية تلائم مع د.م.ح.: ملاحظات	18.8
253	على الشكل التحفظي (CONSERVATION FORM)
19	سرايين لا انضغاطية ولا لزجية (INCOMPRESSIBLE INVISCID FLOWS) : طرق حسابية معتمدة على
263.....	مؤطرات النبع و الدوامة (SOURCE AND VORTEX PANEL METHODS)
263	مدخل 19.1
263	بعض الواجهة الاساسية لسريان لا انضغاطي و لا لزجي 19.2
20	الخصوصيات الرياضية (MATHEMATICAL PROPERTIES) لمعادلات ديناميك الموائع (FLUID DYNAMIC
	267 (EQUATIONS)
267	مدخل 20.1
268	بعض المعادلات التفاضلية الجزئية 20.2
268	تصنيف (CLASSIFICATION) المعادلات التفاضلية الجزئية (PARTIAL DIFFERENTIAL EQ.s) 20.3
السلوك العام للاصناف المختلفة من المعادلات	20.4
275	التفاضلية الجزئية و علاقتها بديناميات الموائع
275.....	المعادلات القطع الزائد (Hyperbolic Equations) 20.4.1
278.....	معادلات القطع مكافئة / Parabolic Equations 20.4.2
279.....	المعادلات القطع الناقص (elliptic equations) 20.4.3
281.....	بعض الملاحظات 20.4.4
281.....	طرح المشاكل بشكل جيد / Well-Posed Problems 20.4.5
281.....	المراجع 20.4.6
21	تفريز لمعادلات التفاضلية الجزئية (DISCRETIZATION OF PDES)
282	مدخل 21.1
283	اشتقاق مقسومات لفرق محدودة ابتدائية (ELEMENTARY FINITE DIFFERENCE QUOTIENTS) 21.2
289.....	ماذا يحدث على الحدود (boundary)؟ 21.3
291	جوانب اساسية لمعادلات الفرق المحدود (FINITE-DIFFERENCE EQUATIONS) 21.3
295.....	تعليق عام 21.3.1
296	أخطاء وتحليل الاستقرار (ERRORS AND AN ANALYSIS OF STABILITY) 21.4
22	تحولات الشبكة (GRID TRANSFORMATIONS)
307	مدخل 22.1
309	GENERAL TRANSFORMATION OF THE EQUATIONS 22.2
314	METRICS AND JACOBIANS 22.3
316	COORDINATE STRETCHING 22.4
319	BOUNDARY-FITTED COORDINATE SYSTEMS 22.5
330	الشبكة التكيفية (ADAPTIVE GRID) 22.6

	طرق الفروق المحدودة الواضحة (EXPLICIT FINITE DIFFERENCE METHODS): بعض التطبيقات	23
	المحددة للسريان اللزجي واللانزجي 337	
337	مدخل (INTRODUCTION)	23.1
338	طريقة لأكس واندروف (THE LAX- WENDROFF METHOD)	23.2
343	MACCORMACK'S METHOD	23.3
346	STABILITY CRITERION مقياس الإستقرار	23.4
	تطبيقات مختارة من تقنيات المعتمدة على الزمن صريح (EXPLICIT TIME-)	23.5
	348 (DEPENDENT TECHNIQUE)	
349	Non-equilibrium Nozzle Flows	23.5.1
351	Flow Field over a Supersonic Blunt Body	23.5.2
353	Internal Combustion Engine Flows	23.5.3
355	Supersonic Viscous Flow over a Rearward-Facing Step With Hydrogen Injection	23.5.4
359	Supersonic Viscous Flow over a Base	23.5.5
360	References	23.5.6
363	الأحجام المحدودة (FINITE VOLUMES)	24
363	نظرة عامة	24.1
368	العناصر المحدودة:	25
368	مدخل الى العناصر المحدودة (FINITE ELEMENTS)	25.1
372	مدخل الي طريقة العناصر المنتهية (FEM) في ديناميكيات الموائع الحسابية (CFD)	25.2
372	شرح طريقة العناصر المنتهية	25.3
374	الصيغة المتحولية (VARIATIONAL FORMULATION)	25.4
374	برهان يظهر وجود حل وحيد	
374	الصيغة المتحولية ل-P2	
375	التقطيع (DISCRETIZATION)	25.5
377	البرمجيات المستخدمة في النمذجة والمحاكاة	26
377	تنسيق الملفات (FORMAT OF FILES)	26.1
380	القيام بالنموذج	26.2
381	تطبيق الشبكة على النموذج	26.3
383	الحلال ELMER	26.4
	استخدام برامج لا تحتاج الى رخصة في ميدان ديناميكيات	27
	384 الموائع الحسابية	
	تحسيب سريان الماء داخل محطة طاقة تعمل على البخار ببرامج	27.1
	384 جاهزة	
	محطة طاقة عن طريق حرق النفايات لتبخير الماء قرب	27.1.1
	384 طرابلس الشام	
	مسألة تكبير حجم حتى تستخدم للتخلص	27.1.2
	386 من نفايات احدى المدن الكبرى وتغزيتها بالكهرباء	
386	حل المسألة	27.1.3

389.....	OpenFOAM داخل Gmsh باستخدام	27.1.3.1
398.....	استخدام برنامج Elmer	27.1.3.2
413.....	مراجع	27.1.4
	انشاء برنامج لتحليل مسألة ما في ميدان ديناميكيات الموائع	27.2
	413	الحسابية (د.م.ح.)
413.....	تحسين السريان في زاوية باستخدام OpenFOAM	27.2.1
426.....	لمحات عن الحرق الحسابي (NUMERICAL COMBUSTION)	28
426	بعض ملاحظات بالنسبة لمحاكاة الحرق	28.1
426.....	(Flame Sheet Model) و (brutto reactions)	28.1.1
427	اساسيات الحرق (BASICS OF COMBUSTION)	28.2
430.....	ملحقات (APPENDICES)	29
430	ملحق أ: مضمون كتاب "ميكانيك الموائع" لمحمد هاشم الصديق	29.1
431	ملحق ب: مضمون كتاب [FERZIGER, PERIC]	29.2
433.....	قاموس انجليزي - عربي (DICTIONARY ENGL.-ARABIC)	30
449.....	MEAE-CFDNC (COMPUTATIONAL FLUID DYNAMICS AND NUMERICAL COMBUSTION) CODE (2019)	
453.....	INTRODUCTION مدخل	31
454.....	BASICS	32
454	GENERAL FORMS	32.1
456	DIFFERENT FORMS OF ENERGY EQUATIONS	32.2
456	VISCOUS TENSOR	32.3
456	CHEMICAL KINETICS	32.4
458	REACTING FLOW CONSERVATION EQUATIONS	32.5
459	BOUNDARY CONDITIONS	32.6
460.....	Reacting Navier-Stokes equations near a boundary	32.6.1
461.....	Comparison between NSCBC implementation for Euler and Navier-Stocks	32.6.2
463.....	Examples of implementation	32.6.3
466	CONSERVATION EQUATION COMPARISON BETWEEN KIVAII AND POINSOT	32.7
467.....	KIVA II مخطط التدفق العام لبرنامج	33
469.....	مخطط تسلسل البرنامج (SEQUENCE DIAGRAM)	34
470.....	CLASS DIAGRAM	35
471.....	CODE GENERATION	36
473.....	PARA VIEW INPUT FILES	37
475	DISPLAYING DATA AS POINTS	37.1
477	DISPLAYING DATA AS STRUCTURED GRID	37.2
481.....	DISCRETIZATION OF PARTIAL DIFFERENTIAL EQUATIONS	38

481	THE CONTINUITY EQUATION (MASS CONSERVATION)	38.1
482	THE MOMENTUM EQUATION FOR FLUID MIXTURE (MOMENTUM CONSERVATION)	38.2
484	THE INTERNAL ENERGY EQUATION (ENERGY CONSERVATION)	38.3
486	CHEMICAL RELATIONS	38.4
486.....	<i>Mass reaction rate</i>	38.4.1
486.....	<i>Rate of progress of reaction j:</i>	38.4.2
487.....	<i>Rate constants:</i>	38.4.3
487.....	<i>Mixture density:</i>	38.4.4
487.....	<i>Mixture viscosity</i>	38.4.5
487.....	<i>Species viscosity</i>	38.4.6
487.....	<i>Pressure:</i>	38.4.7
489.....	APPLICATION EXAMPLE: HYDROGEN OXYGEN COMBUSTION	39
489	HYDROGEN COMBUSTION CHARACTERISTICS	39.1
489.....	<i>Hydrogen characteristics</i>	39.1.1
489.....	<i>Oxygen characteristics</i>	39.1.2
489.....	<i>Water characteristics</i>	39.1.3
489.....	<i>Mixture characteristics</i>	39.1.4
490	PROGRAM CODE	39.2
491	PROGRAM INPUT	39.3
492	PROGRAM RESULTS	39.4
495	RESULTS VIEWED ON PARAVIEW	39.5
495.....	<i>First step: open your .csv files</i>	39.5.1
495.....	<i>Second Step: make sure that the correct properties are enabled.</i>	39.5.2
495.....	<i>Third Step: Add filter for each file and fill with the right parameters:</i>	39.5.3
496.....	<i>Fourth Step: Choose your variable and the desired view</i>	39.5.4
497.....	First time step:	39.5.4.1
498.....	Second time step:	39.5.4.2
499.....	<i>Fifth Step(Optional): You can also add outline shape for your simulation</i>	39.5.5
500.....	ANNEX A: PROGRAM CODE	40
500	A.1. CLASS FUEL:	40.1
502	A.2. CLASS CHEMICAL:	40.2
503	A.3. CLASS MIX:	40.3
504	A.4. CLASS MIXTURE:	40.4
510	A.5. MAIN PROGRAM:	40.5
DXIV	IAP SUPERNOVA SIMULATION (2019)	
517.....	مدخل INTRODUCTION	41
518.....	CHAPTER 1: BASICS	42
522.....	CHAPTER 2: PROGRAM CODE DESCRIPTION	43

524	CHAPTER 3: RESULTS	44
527	ANNEX A: CODE LISTING	45
DXLIII	IAP-PSC (PLASMA SIMULATION CODE) (2019)	
547	BASICS	46
547	LASER-MATTER INTERACTION	46.1
547	<i>Plasma definition</i>	46.1.1
548	<i>Laser-cutting</i>	46.1.2
550	PLASMA WAVES KINETIC THEORY	47
550	GOVERNING EQUATIONS, VLASOV-BOLTZMANN EQUATION	47.1
550	<i>Transport equations, Maxwell Equations</i>	47.1.1
551	The Boltzmann collision operator	47.1.1.1
553	NUMERICAL MODEL	48
553	MESHING	48.1
554	SOME PUBLIC CODES / RESEARCH GROUPS	49
554	THE PLASMA THEORY AND SIMULATION GROUP PTSG UNIV. OF CALIFORNIA, BERKLEY	49.1
555	<i>People of PTSG</i>	49.1.1
555	<i>Projects</i>	49.1.2
555	Current Projects in PTSG	49.1.2.1
555	Recently Completed Projects in PTSG	49.1.2.2
555	<i>PTSG Software</i>	49.1.3
555	<i>General Information</i>	•
555	<i>Acknowledgments</i>	•
556	<i>Description</i>	•
556	<i>Distribution</i>	•
556	<i>Codes that require xgrafix</i>	
556	<i>Codes with own version of xgrafix</i>	
556	<i>Python/Matlab based codes</i>	
557	<i>Code Consulting and Maintenance</i>	
557	<i>Publications Using PTSG Codes, Worldwide</i>	49.1.4
557	<i>Workshops</i>	49.1.5
557	Plasma Device Workshop 2009 (PDW2009)	49.1.5.1
558	COMPARISON BETWEEN ACTUAL IAP-PCS AND XPDP2	50
558	DESCRIPTION OF XPDP2	50.1
558	<i>Abstract of [1]:</i>	50.1.1
558	<i>Abstract of [2]:</i>	50.1.2
559	<i>xpdp2 Code Description</i>	50.1.3
559	ACTUAL IAP-PSC CODE	50.2

561	COMPARING INFORMATION CONTENT OF FILES OF IAP-PSC AND XPDP2	50.3
563.....	IAP-PSC PROGRAM	51
563	DESCRIPTION	51.1
564	THE CODE	51.2
564.....	<i>Class 1: parameters</i>	51.2.1
566.....	<i>Class 2: Lorentz</i>	51.2.2
566.....	<i>Class 3: pos_veloct</i>	51.2.3
567.....	<i>Class 4: Dens_current</i>	51.2.4
567.....	<i>Class 5: Maxwell_equat</i>	51.2.5
569.....	<i>Main program</i>	51.2.6
575	RESULTS	51.3
577.....	LITERATURE	
579.....	INT-LMIC (LASER MATTER INTERACTION CODE) (2020)	
585.....	INTRODUCTION	52
586	PROJECT ENVIRONMENT AND RESEARCH STARTING POINT	52.1
586	OVERVIEW AND TASK OF MASTER THESIS	52.2
587.....	<i>Task of master thesis</i>	52.2.1
587.....	<i>Modeling and Visualization of Laser-gas interaction</i>	52.2.2
588.....	BASICS: PHYSICS OF LASER-PLASMA INTERACTION	53
588	INTRODUCTION	53.1
588.....	<i>Kinetic-Correlated plasmas</i>	53.1.1
589	PLASMA MODELS FOR LASER-PLASMA INTERACTIONS INCLUDING ULTRA-SHORT LASER PULSES AND HIGH ENERGY CIRCUMSTANCES.	53.2
589.....	<i>Static model</i>	I.
589.....	<i>Fluid model</i>	II.
589.....	<i>Kinetic model</i>	III.
589	LASER-GAS INTERACTION AND LASER-PLASMA INTERACTION	53.3
590.....	<i>Single Electron Dynamics and Radiation Friction</i>	53.3.1
590.....	<i>Single Electron Dynamics</i>	53.3.2
590.....	<i>Kinetic and Fluid Equations</i>	53.3.3
591.....	<i>Ponderomotive Force</i>	53.3.4
592.....	<i>Radiation Friction (Reaction)</i>	53.3.5
594	SIMULATION OF INTERACTION LASER-PLASMA (NUMERICAL ASPECT)	53.4
	<i>Summary of numerical method to treat the kinetic model of plasma using Particle-in-Cell (PIC) method with</i>	53.4.1
594.....	<i>Smilei code</i>	
594.....	The Maxwell-Vlasov model	53.4.1.1
595.....	Quasi-particles and the PIC method	53.4.1.2
596.....	Time- and space-centered discretization	53.4.1.3
597.....	Initialization of the simulation and the PIC loop	53.4.1.4
598.....	<i>Reference units</i>	53.4.2

<i>599</i>	<i>..... The PIC loop in detail</i>	53.4.3
600Field interpolation at the particle	53.4.3.1
600 Particle pusher	53.4.3.2
600 Charge conserving current deposition	53.4.3.3
601 Maxwell solvers	53.4.3.4
<i>603</i>	<i>..... Boundary conditions</i>	<i>53.4.4</i>
603 QUANTUM ELECTRODYNAMICS QED AND QUANTUM CHROMODYNAMICS QCD	53.5
<i>603</i>	<i>..... QED MODEL IN PARTICLE-IN-CELL SIMULATIONS</i>	<i>53.5.1</i>
<i>605</i>	<i>..... Quantum Chromodynamics QCD</i>	<i>53.5.2</i>
607 MONTE CARLO TECHNIQUES	53.6
<i>607</i>	<i>..... A simple MC simulation is the determination of n</i>	<i>53.6.1</i>
608 CONTRIBUTION: THE PARTICLE-IN-CELL SIMULATION OF PLASMAS: 1D	54
608 INTRODUCTION	54.1
608 ESPIC	54.2
<i>608</i>	<i>..... Core routines</i>	<i>I.</i>
<i>610</i>	<i>..... Loop over time of simulation</i>	<i>II.</i>
<i>611</i>	<i>..... Important parameters.</i>	<i>III.</i>
611 RESULTS	54.3
<i>611</i>	<i>..... Results without B</i>	<i>54.3.1</i>
<i>615</i>	<i>..... Results with B</i>	<i>54.3.2</i>
617 CONCLUSION	54.4
618 CONTRIBUTION: LASER-MATTER INTERACTION IN IAP-PSC CODE	55
618 IMPROVEMENT IN IAP-PSC CODE WITHOUT LASER-MATTER INTERACTION	55.1
619 MODIFIED CODE	55.2
619 UML DIAGRAMS	55.3
621 SIMULATION IN ONE DIMENSION	55.4
622 SIMULATION IN THREE DIMENSIONS	55.5
622 CONCLUSION	55.6
624 BIBLIOGRAPHY	56
625 LIST OF SYMBOLS	57
626 ANNEX	58
626 IAP_PSC C++ CODE	58.1
653 PARAVIEW INPUT FILES	58.2
<i>655</i>	<i>..... Displaying data as points</i>	<i>58.2.1</i>
<i>657</i>	<i>..... Displaying data as structured grid</i>	<i>58.2.2</i>
<i>660</i>	<i>..... Saving Results</i>	<i>58.2.3</i>
663 FIRST TEST RESULTS	58.3
<i>663</i>	<i>..... IAP_PSC</i>	<i>58.3.1</i>

Molecular Modeling Basics

1 Some concepts in Molecular Modeling

Molecular graphics (MG) is the discipline and philosophy of studying molecules and their properties through graphical representation. IUPAC limits the definition to representations on a "graphical display device".

Computer graphics has had a dramatic impact upon molecular modelling.

It is the interaction between molecular graphics and the underlying theoretical methods that has enhanced the accessibility of molecular modelling methods and assisted the analysis and interpretation of such calculations.

Over the years, two different types of molecular graphics display have been used in molecular modelling. First to be developed were vector devices, which construct pictures using an electron gun to draw lines (or dots) on the screen, in a manner similar to an oscilloscope. Vector devices were the mainstay of molecular modelling for almost two decades but have now been largely superseded by raster devices. These divide the screen into a large number of small "dots", called pixels. Each pixel can be set to any of a large number of colors, and so by setting each pixel to the appropriate color it is possible to generate the desired image.

Molecules are most commonly represented on a computer graphics using 'stick' or 'space filling' representations. Sophisticated variations on these two basic types have been developed, such as the ability to color molecules by atomic number and the inclusion of shading and lighting effects, which give 'solid' models a more realistic appearance.

Computer-generated models do have some advantages when compared with their mechanical counterparts. Of particular importance is the fact

رسومات الجزيئية (MG) هي الانضباط وفلسفة دراسة الجزيئات وخصائصهم من خلال الرسم. اقتصر تعريف IUPAC للـ MG على أنه "جهاز عرض الرسومات".

كان لرسومات الحاسوب أثر كبير على النمذجة الجزيئية. إن التفاعل بين الرسومات والأساليب الجزيئية الكامنة وراء النظرية ، عززت إمكانية الوصول إلى أساليب النمذجة الجزيئية وساعدت في تحليل وتفسير مثل هذه الحسابات.

على مر السنوات، تم استخدام نوعين مختلفين من عرض الرسومات الجزيئية في النمذجة الجزيئية.

الأول، الأجهزة الناقلة (vector devices) ، التي تقوم ببناء الصور باستخدام بندقية إلكترونية لرسم خطوط (أو نقاط) على الشاشة ، بطريقة مشابهة للذبذبات. وكانت هذه الأجهزة عماد النمذجة الجزيئية على مدى عقدين من الزمن تقريباً ولكن الآن حلت محلها الأجهزة النقطية (raster devices) إلى حد كبير. يمكن ضبط كل بيكسل على لون معين من الألوان الكثيرة، وذلك من خلال وضع كل بيكسل على اللون المناسب لتوليد الصورة المطلوبة.

غالباً ما تكون الجزيئات ممثلة على رسومات الحاسوب باستخدام 'stick' أو 'space filling' . وقد تم إضافة بعض التطويرات على هذين النوعين الأساسيين، مثل القدرة على تلوين الجزيئات بواسطة رقم الذرة، وإدراج التظليل وتأثيرات الإضاءة، التي تعطي النماذج الصلبة مظهر أكثر

that a computer model can be very easily interrogated to provide quantitative information, from simple geometrical measures such as the distance between two atoms to more complex quantities such as the energy or surface area. Quantitative information such as this can be very difficult if not impossible to obtain from a mechanical model. Nevertheless, mechanical models may still be preferred in certain types of situation due to the ease with which they can be manipulated and viewed in three dimensions.

A computer screen is inherently two-dimensional, whereas molecules are three-dimensional objects. Nevertheless, some impression of the three-dimensional nature of an object can be represented on a computer screen using techniques such as depth cueing (in which those parts of the object that are further away from the viewer are made less bright) and through the use of perspective. Specialized hardware enables more realistic three-dimensional stereo images to be viewed. In the future 'virtual reality' systems may enable a scientist to interact with a computer-generated molecular model in much the same way that a mechanical model can be manipulated.

Even the most basic computer graphics program provides some standard facilities for the manipulation of models, including the ability to translate, rotate and 'zoom' the model towards and away from the viewer. More sophisticated packages can provide the scientist with quantitative feedback on the effect of altering the structure. For example, as a bond is rotated then the energy of each structure could be calculated and displayed interactively.

For large molecular systems it may not always be desirable to include every single atom in the

واقعية.

إن المقارنة بين النماذج التي يوجد لها الحاسوب مع نظرائهم الميكانيكية لها بعض المزايا. منها خاصة، أولاً حقيقة أن نموذج يمكن أن يقدم الكمبيوتر بكل سهولة معلومات كمية عن القياسات الهندسية البسيطة مثل بعد المسافة بين اثنين من الذرات إلى كميات أكثر تعقيداً مثل مجال الطاقة أو السطح. ولكن الحصول على معلومات كمية كالتالي ذكرت، قد يكون صعب جداً إن لم يكن مستحيلًا ، الحصول عليها من النماذج الميكانيكية. ومع ذلك ، لا يزال استعمال النماذج الميكانيكية مفضلاً في بعض الأوضاع بسبب سهولة التلاعب بها وعرضها الثلاثي الأبعاد.

ثانياً إن شاشة الكمبيوتر بطبيعتها ثنائية الأبعاد ، في حين أن الجزئيات هي كائنات ثلاثية الأبعاد. ومع ذلك ، يمكن لبعض الأفكار ذات طبيعة ثلاثية الأبعاد للكائن أن تُمَثَّل على شاشة الكمبيوتر باستخدام تقنيات مثل عمق cueing (أجزاء الجسم الأكثر بعداً تكون أقل بريقاً) ومن خلال استخدام الرسم المنظوري. تمكن الأجهزة المتخصصة عرض مجسم أكثر واقعية بصور ثلاثية الأبعاد. إن أنظمة "الواقع الافتراضي" قد تمكن العالم (مفرد علماء) في المستقبل، من التفاعل مع النماذج الجزئية التي يوجد لها الحاسوب، بنفس الطريقة التي يمكن التفاعل فيها مع النماذج الميكانيكية.

في عالم النمذجة الجزيئية الحاسوبية ، نجد أن حتى أبسط برامج

computer image; the sheer number of atoms can result in a very confusing and cluttered picture. A clearer picture may be achieved by omitting certain atoms (e.g. hydrogen atoms) or by representing groups of atoms as single 'pseudo-atoms'. The techniques that have been developed for displaying protein structures nicely illustrate the range of computer graphics representation possible. Proteins are polymers constructed from amino acids, and even a small protein may contain several thousand atoms. One way to produce a clearer picture is to dispense with the explicit representation of any atoms and to represent the protein using a 'ribbon'. Proteins are also commonly represented using the cartoon drawings developed by J Richardson.

رسومات الحاسوب يوفر بعض التسهيلات الأساسية للتلاعب في النماذج ، بما في ذلك القدرة على الترجمة ، وتدوير و'تقريب' النموذج نحو وبعيدا عن المشاهد. إن أكثر المجموعات تطوراً ، تُقدّم للعالم (مفرد علماء) ردود الفعل الكمية للبنية على أثر تعيُّرها. على سبيل المثال ، في حال تدوير الرابط ، تُحتسب طاقة كل بنية ويتم عرضها تلقائياً.

في الأنظمة الجزيئية الكبيرة قد لا يكون مرغوب دائماً أن تشمل صورة الكمبيوتر كل الذرات. إذ أن العدد الهائل من الذرات يمكن أن ينتج صورة مشوشة ومربكة جداً. يمكن التوصل إلى صورة أوضح عن طريق حذف ذرات معينة (مثل ذرات الهيدروجين) أو من خلال تمثيل مجموعات من الذرات في شبه ذرة واحدة (ذرة زائفة). تُعرض التقنيات ، التي تم تطويرها لعرض بنية البروتين، مجموعة من تمثيل رسومات الحاسوب الممكنة. البروتينات هي بوليمرات مركّبة من الأحماض الأمينية، وحتى البروتين الصغير قد يحتوي على عدة آلاف من الذرات. الطريقة الوحيدة لإنتاج صورة واضحة هو الاستغناء عن تمثيل مفصل لكل الذرات وتمثيل البروتين باستخدام 'الشريط'. الطريقة الوحيدة لإنتاج صورة واضحة هو الاستغناء عن تمثيل شامل لكل الذرات والقيام بتمثيل البروتين باستخدام 'شريط'. تمثل البروتينات أيضاً باستخدام رسومات الكرتون التي وضعها ج.ريتشاردسون (J Richardson).

1.1 Surfaces/مساحات السطح

Many of the problems that are studied using molecular modelling involve the non-covalent interaction between two or more molecules. The study of such interaction is often facilitated by examining the van der waals, molecular or accessible surfaces of the molecule. The van der waals surface is simply constructed from the overlapping van der waals spheres of the atoms, Fig 8. It corresponds to a CPK or space-filling model. Let us now consider the approach of a small 'probe' molecule, represented as a single van der waals sphere, up to the van der waals surface of a larger molecule.

The finite size of the probe sphere means that there will be regions of 'dead space', crevices that are not accessible to the probe as it rolls about on the larger molecule.

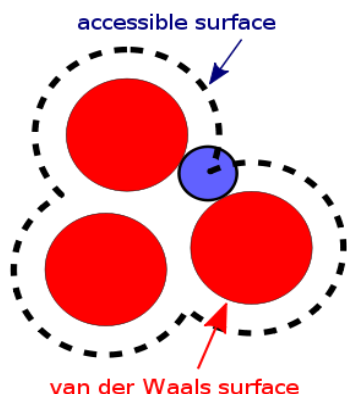


Fig 8: The van der Waals surface is shown in red. The accessible surface is drawn with dashed lines and is created by tracing the center of the probe sphere (in blue) as it rolls along the van der Waals surface. (Source: [http://en.wikipedia.org/wiki/Accessible surface](http://en.wikipedia.org/wiki/Accessible_surface))

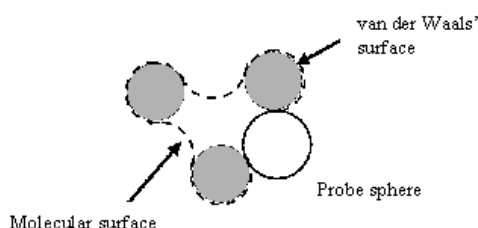


Fig9: (Source: http://www.ccp4.ac.uk/.../newsletter38/03_surfarea.html)

This is illustrated in fig 1.4. The amount of dead space increases with the size of the probe; conversely, a probe of zero size would be able to

إن العديد من المشاكل التي درست باستخدام النمذجة الجزيئية ، تنطوي على التأثير غير التساهمي بين اثنين أو أكثر من الجزيئات. كثيراً ما تسهل دراسة فان دير فال (van der waals) للجزيء والأسطح الجزيئية المتاحة، مثل هذا التفاعل. يتألف سطح فان دير فال (van der waals) ببساطة من تداخل فان دير فال (van der waals) في مجالات الذرات (كما توضح الصورة fig8). وهو يمثل نموذج CPK أو نموذج space-filling. دعونا ننظر الآن إلى اقتراب جزيء صغير 'متوقع' ، ممثلاً بجسم فان دير فال كروي واحد ، إلى سطح جزيء فان دير فال أكبر . الحجم المحدود للجسم الكروي المتوقع يعني أنه ستكون هناك مناطق 'مساحة ميتة'. لا يستطيع الجسم المتوقع أن يصل إلى الشقوق لأنها تلتف حول جزيء أكبر.

يزداد عدد المساحات الميتة مع تزايد عدد الأجسام المتوقعة. وبالعكس إن الجسم المتوقع الذي يساوي حجمه صفر، يمكنه

access all of the crevices. The molecule surface contains two different types of surface element. The contact surface corresponds to those regions where the probe is actually in contact with the van der waals surface of the 'target'. The re-entrant surface regions occur where there are crevices that are too narrow for the probe molecule to penetrate. The molecular surface is usually defined using a water molecule as the probe, represented as a sphere of radius 1.4 \AA .

The accessible surface is also widely used. As originally defined by Lee and Richards this is the surface that is traced by the center of the probe molecule as it rolls on the van der waals surface of the molecule (Fig.1.4). The center of the probe molecule can thus be placed at any point on the accessible surface and not penetrate the van der waals spheres of the atoms in the molecule.

الوصول إلى كل الشقوق. يحتوي سطح الجزيء على نوعين مختلفين من عنصر السطح. يشير السطح المحتك، إلى تلك المناطق حيث أن الجسم المتوقع على احتكاك مع سطح فان دير فال 'الهدف'. تظهر منطقة ال re-entrant surface حيث تتواجد الشقوق الضيقة التي لا تسمح بدخول الجزيء المتوقع. غالباً ما يُحدّد سطح الجزيء باستخدام جزيء من الماء كجسم متوقع مُمثّل في جسم كروي، يبلغ شعاعه 1.4 \AA درجة.

تستخدم ال accessible surface أيضاً بشكل واسع. وهي (بحسب تعريف Lee و Richards الأصلي) السطح الممتد من وسط أو مركز الجزيء المتوقع إلى ما حول سطح فان دير فال للجزيء (Fig.1.4). وبالتالي يمكن وضع مركز الجزيء على أي نقطة في ال accessible surface دون أن يدخل الجسم الكروي للذرات إلى داخل الجزيء.

1.2 Computer Hardware and Software/ أجهزة وبرمجيات الكمبيوتر

The workstations that are commonplace in many laboratories now offer a real alternative to centrally maintained 'supercomputers' for molecular modelling calculations, especially as a workstation or even a personal computer can be dedicated to a single task, whereas the supercomputer has to be shared with many other users. Nevertheless, in the immediate future there will always be some calculations that require the power that only a supercomputer can offer. The speed of any computer system is ultimately

تقدم أماكن العمل الموجودة في العديد من المختبرات بديلاً للحواسيب المركزية العملاقة 'supercomputers' التي تقوم بالعمليات الحسابية للنمذجة الجزيئية، بحيث يكرّس مكان العمل أو حتى جهاز كمبيوتر شخصي لمهمة واحدة، في حين أن الحاسوب العملاق يكون مشترك مع عدة مستخدمين آخرين. ومع ذلك، في المستقبل القريب سيكون هناك دائماً بعض الحسابات التي تتطلب القوة التي لا يمكن أن يقدمها إلا الحاسوب

constrained by the speed at which electrical signals can be transmitted. This means that there will come a time when no further enhancements can be made using machines with 'traditional' single-processor serial architectures, and parallel computers will play an ever more important role.

To perform molecular modelling calculations one also requires appropriate programs (the software). The software used by molecular modelers ranges from simple programs that perform just a single task to highly complex packages that integrate many different methods. There is three items of software have been so widely used: the Gaussian series of programs for performing *ab initio* quantum mechanics, the MOPAC/AMPAC programs for semi-empirical quantum mechanics and the MM2 program for molecular mechanics.

العملات فقط. إن سرعة أي نظام حاسوب مقيدة بالسرعة التي تنتقل فيها الإشارات الكهربائية. وهذا يعني أنه سيأتي وقت لا يمكن إحراز المزيد من التحسينات باستخدام الأجهزة 'التقليدية' ذات معالج واحد لهندسة متسلسلة، والحواسيب المتوازية سوف تلعب دورا أكثر أهمية من أي وقت مضى.

يتطلب أداء العمليات الحسابية للنمذجة الجزيئية أيضا برامج مناسبة (البرنامج). تتراوح البرمجيات المستخدمة في النمذجة الجزيئية بين البرامج البسيطة التي تؤدي مهمة واحدة فقط والبرامج الشديدة التعقيد التي تقوم بدمج العديد من الطرق المختلفة. هناك ثلاثة أنواع من البرامج التي تم استخدامها على نطاق واسع جدا : سلسلة برامج غاوسي Gaussian لتنفيذ *ab initio*¹ ميكانيكا الكم ، وبرامج AMPAC / MOPAC لميكانيكا الكم شبه التجريبية وبرنامج MM2 للميكانيكا الجزيئية.

1.3 Units of Length and Energy/ وحدات الطول والطاقة

Z-matrix is defined using the angstrom as the unit of length (1 Å ≡ 10⁻¹⁰ m ≡ 100pm). The angstrom is a non-SI (International System of units) unit but is a very convenient one to use, as most bond lengths are of the order of 1-2 Å.

يتم تعريف Z-matrix باستخدام انجستروم كوحدة للطول (1 انجستروم ≡ 10⁻¹⁰ م ≡ 100 بيكومتر). انجستروم هي وحدة غير تابعة للنظام الدولي للوحدات ، ولكنها ملائمة جدا

¹ *Ab initio quantum chemistry methods are computational chemistry methods based on quantum chemistry/*

أساليب *Ab initio* هي من طرق المعلوماتية الكيميائية التي تستند إلى كيمياء الكم (بحسب موسوعة ويكيبيديا الإلكترونية)

One other very commonly non-SI unit found in molecular modelling literature is the kilocalorie (1 kcal \equiv 4.1840 kJ). Other systems of units are employed in other types of calculation, such as the atomic units used in quantum mechanics.

للاستخدام، و تتراوح معظم أطوال الروابط بين 1-2 انجستروم. كما أن هناك وحدة أخرى تستخدم في كتب النمذجة الجزيئية، وهي غير تابعة للنظام الدولي للوحدات : السعرات الحرارية kilocalorie (1 سعة حرارية \equiv 4,1840 كيلوجول). وهناك أيضاً أنظمة أخرى من الوحدات تستخدم في أنواع أخرى من الحسابات، مثل الوحدة الذرية التي تستخدم في ميكانيكا الكم.

1.4 Mathematical Concepts/ المفاهيم الرياضية

A full appreciation of all the techniques of molecular modelling would require a mathematical treatment. However, a proper understanding does benefit from some knowledge of mathematical concepts such as vectors, matrices, differential equations, complex numbers, series expansions and lagrangian multipliers and some very elementary statistical concepts.

يجب القيام بالمعالجة الرياضية، من أجل تقدير جميع تقنيات النمذجة الجزيئية. لذلك ، يجب معرفة بعض المفاهيم الرياضية مثل المتجه vector ، المصفوفات matrices، المعادلات التفاضلية differential equations ، والأرقام المعقدة complex numbers ، سلسلة التوسعات ، ومضاعفات لاغرانج وبعض المفاهيم الإحصائية الأولية.

1.5 References / المراجع

1. <http://www.giantmolecule.com/shop/scripts/prodView.asp?idproduct=6>
2. <http://www1.imperial.ac.uk/medicine/people/r.dickinson/>
3. <http://www.answers.com/topic/molecular-graphics>
4. <http://commons.wikimedia.org/wiki/File:L-proline-zwitterion-from-xtal-3D-balls-B.png>
5. http://en.wikipedia.org/wiki/Accessible_surface
6. http://www.ccp4.ac.uk/.../newsletter38/03_surfarea.html

2 Computational Quantum Mechanics / معلوماتية ميكانيكا الكم

2.1 Introduction / مقدمة

There are number of quantum theories for treating molecular systems. The one which has been widely used is molecular orbital theory. However, alternative approaches have been developed, some of which we shall also describe, albeit briefly. We will be primarily concerned with the ab initio and semi-empirical approaches to quantum mechanics but will also mention techniques such as Huckel theory, valence bond theory and Density functional.

هناك عدد من نظريات الكم لمعالجة الأنظمة الجزيئية. وتعتبر نظرية المدار الجزيئي ، النظرية الأكثر استعمالاً. كما تم وضع بعض النهج الأخرى. نذكر أولاً مناهج ab initio وال semi-empirical لميكانيكا الكم. كما نذكر أيضاً بعض التقنيات مثل نظرية Huckel ، نظرية تكافؤ السندات valence bond و نظرية الكثافة الوظيفية Density functional.

The starting point for any discussion of quantum mechanics is the Schrödinger equation. The full , time-dependent form of this equation is:

إن معادلة شرودنجر Schrödinger هي نقطة الإنطلاق لأية مناقشة في ميكانيكا الكم. النموذج الكامل للمعادلة المتعلقة بالزمن هو

eq.2,1

$$\left(-\frac{\hbar^2}{2m} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) + V \right) \Psi(\mathbf{r}, t) = i\hbar \frac{\partial}{\partial t} \Psi(\mathbf{r}, t)$$

Eq. (2,1) refers to a single particle (e.g. an electron) of mass m which is moving through space (given by a position vector $\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$) and time (t) under the influence of an external field V (which might be the electrostatic potential due to the nuclei of a molecule). h is Planck's constant divided by 2π

يشير Eq. (2,1) إلى جسيم (مثل الإلكترون) لكتلة m ، يتحرك عبر الفضاء (يُحدّد بواسطة متجه $\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$) والوقت (t) تحت تأثير الحقل الخارجي V (التي قد يكون إمكانية الكهرباء المرتبطة بنوى الجزيء). h هو قيمة Planck الثابتة مقسومة على

and i is the square root of -1 . Ψ is the wavefunction which characterizes the particle's motion; it is from the wavefunction that we can derive various properties of the particle. When the external potential V is independent of time then the wavefunction can be written as the product of a spatial part and time part: $\Psi(r, t) = \psi(r) T(t)$. We shall only consider situations where the potential is independent of time, which enables the time-dependent Schrödinger equation to be written in the more familiar, time-independent form:

$2\pi i$ هو الجذر التربيعي لـ -1 . Ψ هو الدالة الموجية الذي يميز حركة الجسيمات. الذي هو التالي من الدالة الموجية التي تمكننا من استنتاج الخصائص المختلفة للجسيمات. عندما تكون الكتلة الخارجية V غير مرتبطة بالوقت، يُمكن كتابة الدالة الموجية كنتيجة لجزء مكاني وزماني: $\Psi(r, t) = \psi(r) T(t)$. يجب أن تأخذ الحالات بعين الاعتبار، عندما تكون الكتلة غير مرتبطة بالوقت، مما يسمح لمعادلة شرودنغر المرتبطة بالوقت، بأن تكتب على هذا النحو الغير مرتبط بالوقت:

eq.2,2

$$E\psi(r) = -\frac{\hbar^2}{2m}\nabla^2\psi(r) + V(r)\psi(r).$$

E is the energy of the particle and we have used the abbreviation ∇^2 (pronounced 'del squared'):

E هي طاقة الجسيم. وقد تم استعمال هذا الاختصار ∇^2 (المسمى 'del squared')

eq.2,3

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

It is usual to abbreviate the left-hand side of eq. (1,1) to $\hat{H}\Psi$, where \hat{H} is the Hamiltonian operator:

عادةً ما تُختصر الجهة اليسرى من المعادلة رقم (1,1) إلى $\hat{H}\Psi$

بحيث أن \hat{H} هي Hamiltonian operator:

eq.2,4

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V$$

This reduces the Schrödinger equation to $\hat{H}\Psi = E\Psi$. To solve the Schrödinger equation it is necessary to find values of E and functions Ψ . The Schrödinger equation falls into the category

مما يختصر معادلة شرودنغر إلى $\hat{H}\Psi = E\Psi$ لحل هذه المعادلة، يجب إيجاد قيمة الـ E والـ Ψ . تقع معادلة شرودنغر داخل فئة

of equations known as partial differential eigenvalue equations in which an operator acts on a function (the eigenfunction) and returns the function multiplied by a scalar (the eigenvalue). A simple example of an eigenvalue equation is:

المعادلات المعروفة بالتفاضل الجزئي لمعادلات القيمة الذاتية ، حيث يقوم المحدد بالتأثير على وظيفة (eigenfunction) ويُردها مضروبة بـ scalar (القيمة الذاتية). مثال بسيط على معادلة :
القيمة الذاتية

Eq.2,5

$$\frac{d}{dx}(y) = ry$$

The operator here is d/dx . One eigenfunction of this equation is $y=e^{ax}$ with the eigenvalue r being equal to a . Eq.1,5 is a first-order differential equation. The Schrödinger equation is a second-order differential equation as it involves the second derivative of Ψ . A simple example of an equation of this type is

المشغل هنا هو d/dx . وظيفة الـ Eigen لهذه المعادلة هي :
 $y=e^{ax}$ زائد r (القيمة الذاتية) تساوي a . تنتمي المعادلة 1,5 إلى الترتيب التفاضلي الأول. وتنتمي معادلة شرودنغر إلى الترتيب التفاضلي الثاني، وتشمل المشتق الثاني لـ Ψ . مثال بسيط لمعادلة من هذا النوع:

Eq.2,6

$$\frac{d^2y}{dx^2} = ry$$

The solutions of eq.2,6 have the form $y = A \cos kx + B \sin kx$, where A, B and k are constants. In the Schrödinger equation Ψ is the eigenfunction and E the eigenvalue.

يتخذ حلّ المعادلة 2, 6 كل $y = A \cos kx + B \sin kx$ ، حيث أن A, B, k ثابتون. في معادلة شرودنغر، Ψ هي وظيفة الـ Eigen و E هي قيمتها.

2.1.1 Operators / المشغلون

The most commonly used operator is that for the energy, which is the Hamiltonian operator itself, \hat{H} . The energy can be determined by calculating

إن مشغل هاميلتون للطاقة هو المشغل الأكثر شيوعاً. يمكن

the following integral:

احتساب الطاقة من خلال احتساب هذا التكامل:

Eq.2,7

$$E = \frac{\int_{-\infty}^{+\infty} \Psi^* \hat{H} \Psi dT}{\int_{-\infty}^{+\infty} \Psi^* \Psi dT} \Rightarrow \int \Psi^* \hat{H} \Psi dT = \int \Psi^* E \Psi dT$$

(Ψ^*): the wavefunction may be a complex number.

(Ψ^*): الدالة الموجية قد تكون عدد مركب.

E: scalar and so can be taken outside the integral.

If the wavefunction is normalized then the denominator in eq.2,7 will equal 1.

E: يمكن أن تخرج من التكامل. إذا كانت الدالة الموجية طبيعية

فإن المخرج في المعادلة eq.2,7 يساوي 1.

The Hamiltonian operator is composed of two parts that reflect the contributions of: kinetic and potential energies to the total energy. The kinetic energy operator is:

يتألف مشغل هاميلتون من جزئين، بحيث تعكس إسهامات:

الطاقة الحركية و طاقة الوضع على إجمالي الطاقة. مشغل الطاقة

الحركية هو:

Eq.2,8

$$-\frac{h^2}{2m} \nabla^2$$

And the operator for the potential energy simply involves multiplication by the appropriate expression for the potential energy. For an electron in an isolated atom or molecule the potential energy operator comprises the electrostatic interactions between the electron and nucleus and the interactions between the electron and the other electrons. For a single electron and a single nucleus with Z protons the potential energy operator is thus:

ويشمل مشغل طاقة الوضع ضرب العبارة الجبرية المناسبة

لإمكانات الطاقة. بالنسبة للإلكترون في ذرة أو جزيء معزول،

يشمل مشغل طاقة الوضع التفاعلات الكهروستاتيكية بين

الإلكترون والنواة و التآثرات بين الإلكترون والإلكترونات الأخرى.

بالنسبة للإلكترون واحد ونواة واحدة مع زد من البروتونات، فإن

مشغل الطاقة المحتملة هو على النحو التالي :

Eq.2,9

$$V = -\frac{Ze^2}{4\pi\epsilon_0 r}$$

Operator for linear momentum along the x direction : مشغل زخم الحركة الخطي أو كمية الحركة الخطية في موازاة الاتجاه x

:X

Eq.2,10

$$\frac{h}{i} \frac{\partial}{\partial x}$$

The expectation value of this quantity can thus be obtained by evaluating the following integral: ويمكن الحصول على قيمة التوقع لهذه الكمية من خلال تقييم

المتكامل التالي :

Eq.2,11

$$p_x = \frac{\int \Psi^* \frac{h}{i} \frac{\partial}{\partial x} \Psi dT}{\int \Psi^* \Psi dT}$$

2.1.2 Atomic Units / وحدات الذرة

The atomic units of length, mass and energy are as follow:

الوحدات الذرية للكتلة والطول والطاقة هي على النحو التالي :

- 1 unit of charge equals the absolute charge on an electron, $|e| = 1.60219 \times 10^{-19} \text{ C}$. شحنة واحدة تساوي القيمة المطلقة لشحنة إلكترون. $|e| = 1.60219 \times 10^{-19} \text{ C}$
- 1 mass unit equals the mass of the electron, $m_e = 9.10593 \times 10^{-31} \text{ kg}$. وحدة الكتلة (كتلة واحدة) تساوي كتلة الإلكترون: $m_e = 9.10593 \times 10^{-31} \text{ kg}$
- 1 unit of length (1Bohr) is given by $a_0 = \frac{h^2}{4\pi^2 m_e e^2} = 5.29177 \times 10^{-11} \text{ m}$. تُعطى وحدة الطول (1 نموذج بور أو بوهر) بواسطة $a_0 = \frac{h^2}{4\pi^2 m_e e^2} = 5.29177 \times 10^{-11} \text{ m}$.

It is the radius of the first orbit in Bohr's treatment of the hydrogen atom. It also turns out to be the most probable distance of 1s electron from the nucleus in the hydrogen atom.

إنه شعاع المدار الأول في نموذج بور لذرة الهيدروجين. ويتحول أيضاً إلى أن يكون المسافة الأكثر ترجيحاً من 1s إلكترون من النواة في ذرة الهيدروجين.

- 1 unit of energy (1 Hartree) is given by $E_a = e^2/4\pi\epsilon_0 a_0 = 4.35981 \times 10^{-18} J$

It corresponds to the interaction between two electronic charges separated by the Bohr radius. The total energy of the 1s electron in the hydrogen atom equals -0.5 Hartree.

- تُعطي وحدة الطاقة (1 هارترى) بواسطة $E_a = e^2/4\pi\epsilon_0 a_0 = 4.35981 \times 10^{-18} J$

كما إنه يتوافق مع التأثير بين شحنتين إلكترونيتين يفصلهما شعاع بوهر . يساوي مجموع الطاقة لـ 1s إلكترون في ذرة الهيدروجين -0.5 هارترى.

2.2 One-electron Atoms

In an atom that contains a single electron, the potential energy depends upon the distance between the electron and the nucleus as given by the Coulomb equation.

في الذرة التي تحتوي على إلكترون واحد، تتركز الطاقة الكامنة على المسافة بين الإلكترون والنواة بحسب معادلة كولومب. ومن الأكثر ملاءمة، تحويل معادلة شرودنجر للإحداثيات القطبية r, θ, ϕ و ϕ (دالة موجية) حيث :

It is more convenient to transform the Schrodinger equation to polar coordinates r, θ and ϕ , (wavefunction) where:

r : the distance from the nucleus

r : المسافة من نواة

θ : the angle to the z axis

θ : زاوية للمحور z

ϕ : the angle from the x axis in the xy plane

ϕ : زاوية من المحور x في الطائرة xy

Eq.2,12

$$\Psi_{nlm} = R_{nl}(r)Y_{lm}(\theta, \phi)$$

$Y(\theta, \phi)$: angular function called a *spherical harmonic*

$Y(\theta, \phi)$: وظيفة زاوية تسمى تناسق كروي

R(r) : radial function

R(r): وظيفة شعاعية

n: principal quantum number: 0, 1, 2,...

n: عدد الكم الرئيسي: 0, 1, 2, ...

l: azimuthal quantum number : 0, 1, ..., (n-1)

l: عدد الكم السمتي: 0, 1, ..., (n-1)

m: magnetic quantum number : -l, -(l-1), ...0...(l-1), l

m: عدد الكم المغناطيسي: -l, -(l-1), ...0...(l-1), l

Eq.2,13

$$R_{nl}(r) = - \left[\left(\frac{2Z}{na_0} \right)^3 \frac{(n-l-1)!}{2n[(n+l)!]^3} \right]^{1/2} \exp\left(-\frac{\rho}{2}\right) \rho^l L_{n+l}^{2l+1}(\rho)$$

$\rho = 2Zr/na_0$, where na_0 is the Bohr radius.

$\rho = 2Zr/na_0$, حيث na_0 هي شعاع بوهر.

$L_{n+l}^{2l+1}(\rho)$ is a special type of function called a

$L_{n+l}^{2l+1}(\rho)$ هي نوع مميز من الوظائف تسمى Laguerre

Laguerre Polynomial

Polynomial

Eq.2,14

$$Y_{lm}(\theta, \phi) = \Theta_{lm}(\theta)\Phi_m(\phi)$$

With:

$$\Phi_m(\phi) = \frac{1}{\sqrt{2\pi}} \exp(im\phi)$$

$$\Theta_{lm}(\theta) = \left[\frac{(2l+1)}{2} \frac{(l-|m|)!}{(l+|m|)!} \right]^{1/2} P_l^{|m|}(\cos\theta)$$

$\Phi_m(\phi)$: The solutions to the Schrödinger equation for a particle on a ring.

$\Phi_m(\phi)$: الحلول لمعادلة شرودنجر لجسيم.

$P_l^{|m|}(\cos\theta)$: Series of function called the associated Legendre polynomials.

$P_l^{|m|}(\cos\theta)$: سلسلة وظائف تدعى (the associated Legendre polynomials).

(Legendre polynomials.

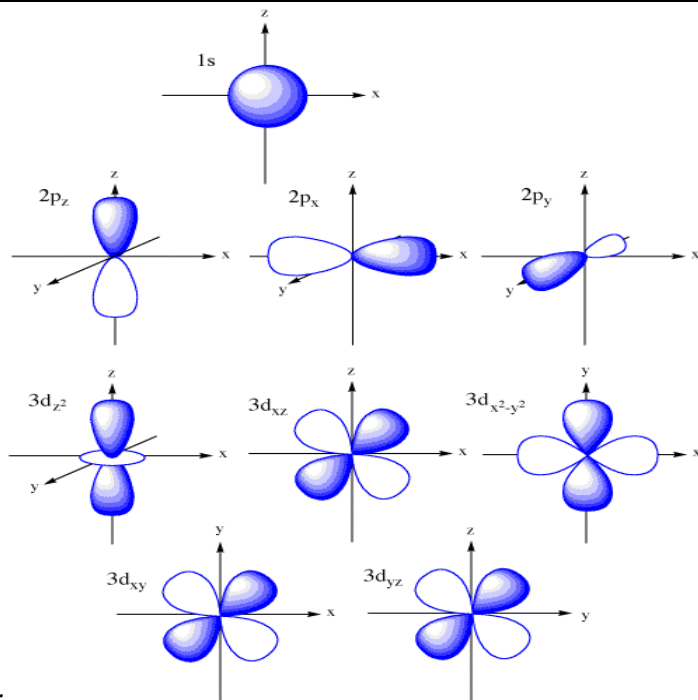


Fig 2.1:

The common graphical representations of s, p and d orbitals/

التمثيل الرسومي المشترك لمدار s,p,d

Src: <http://butane.chem.uiuc.edu/pshapley/GenChem2/Intro/orbit.gif>

The energy of each solution is a function of the principal quantum number only; thus orbitals with the same value of n but different l and m are degenerate. The orbitals are often represented as shown in fig 2.1. These graphical representations are not necessarily the same as the solutions given above. For example, the 'correct' solutions for the 2p orbitals comprise one real and two complex functions:

إن طاقة كل حل هي وظيفة العدد الكمي الرئيسي فقط، وبالتالي إن المدارات لها نفس قيمة n أما قيمة l,m فتكون مختلفة. وغالبا ما تتمثل المدارات كما هو مبين في الشكل رقم 2،1. هذه الأشكال البيانية ليس بالضرورة لها نفس الحلول المذكورة أعلاه. على سبيل المثال ، الحلول 'الصحيحة' لمدارات 2p تتكون من واحد حقيقي وظيفتين معقدتين :

$$2p(+1) = \sqrt{3/4\pi} R(r) \sin \theta e^{i\phi}$$

$$2p(0) = \sqrt{3/4\pi} R(r) \cos \theta$$

$$2p(-1) = \sqrt{3/4\pi} R(r) \sin \theta e^{-i\phi}$$

$R(r)$: The radial part of wavefunction

$R(r)$: الجزء الشعاعي من الدالة الموجية.

$\sqrt{3/4\pi}$: A normalization factor for the angular part.

$\sqrt{3/4\pi}$: عامل تنسيق أحادي للجزء الزاوي.

$2p(0)$: function corresponds to the $2p_z$ orbital that is pictured in Fig 2.1.

$2p(0)$: وظيفة تتوافق مع مدار $2p_z$ المصور في Fig 2.1.

The linear combinations below are the $2p_x$ and $2p_y$ orbitals shown in Fig 2.1.

التوافقيات الخطية أدناه تعود لمدار $2p_x$ ومدار $2p_y$ الموجودين

في Fig 2.1.

$$2p_x = 1/2[2p(+1) + 2p(-1)] = \sqrt{3/4\pi} R(r) \sin \theta \cos \phi$$

$$2p_y = -1/2[2p(+1) - 2p(-1)] = \sqrt{3/4\pi} R(r) \sin \theta \sin \phi$$

These linear combinations still have the same energy as the original complex wavefunctions.

هذه التوافقيات الخطية ما زال لديها نفس طاقة الدالة الموجية

المركبة الأصلية.

2.3 Polyelectronic Atoms and Molecules / إلكترونات متعددة الذرات والجزيئات

Solving the Schrödinger equation for atoms with more than one electron is complicated by a number of factors. The first complication is that the Schrödinger equation for such systems cannot be solved exactly (solutions can only be approximations to the real true solutions).

إن عملية حل معادلة شرودنجر لذرات ذات أكثر من إلكترون واحد، هي عملية معقدة وذلك بسبب عدد من العوامل. المشكلة الأولى هي أنه لا يمكن إيجاد حل دقيق لمعادلة شرودنجر لمثل هذه الأنظمة. (يمكن إيجاد حلول تقريبية فقط للحلول الحقيقية الصحيحة). المشكلة الثانية مع الأنواع المتعددة للإلكترون هو أنه يجب علينا حساب غزل الإلكترون.

A second complication with multi-electron species is that we must account for electron spin.

Spin is characterized by the quantum number s , which for an electron can only take the value $1/2$. The spin angular momentum is quantized such that its projection on the z axis is either $+\hbar$ or $-\hbar$. These two states are characterized by the quantum number m_s , which can have values of $+1/2$ or $-1/2$, and are often referred to as 'up spin' and 'down

يتميز الغزل أو السبين بعدد الكم s ، التي يمكن للإلكترون أن يأخذ قيمة تساوي $1/2$.

يُعد غزل الزخم الزاوي مثل إسقاطه على محور Z هو أيضاً $+\hbar$ أو

spin' respectively. The spin part defines the electron spin and is labeled α or β . These spin functions have value of 0 or 1 depending on the quantum number m_s of the electron. Each spatial orbital can accommodate two electrons, with paired spins. In order to predict the electronic structure of a Polyelectronic atom or a molecule, the *Aufbau principle* is employed, in which electrons are assigned to the orbitals, two electrons per orbital. For most of the situations that we shall be interested in the number of electrons, N , will be an even number that occupy the $N/2$ lowest-energy orbitals.

Electrons are indistinguishable. If we exchange any pair of electrons, then the distribution of electron density remains the same. According to the Born interpretation, the electron density is equal to the square of the wavefunction. It therefore follows that the wavefunction must either remain unchanged when two electrons are exchanged, or else it must change sign. In fact, for electrons the wavefunction is required to change sign: this is the *antisymmetry principle*.

$-\hbar$. تتميز هاتان الحالتان بعدد الكم m_s ، التي يمكن أن يأخذ قيمة $+1/2$ أو $-1/2$. وغالبا ما يشار إليها باسم "مع عقارب الساعة" أو "عكس عقارب الساعة" يحدد جزء السبين (الجزء الغزلي) إلكترون الغزل (السبين) ويسمى α أو β . تساوي وظائف السبين هذه قيمة صفر أو واحد بحسب عدد كم الإلكترون m_s .

كل مدار يمكن أن يستوعب إلكترونين، مع غزليين (2 غزل/سبين). من أجل توقع البنية الإلكترونية للذرة أو الجزيء المتعدد الإلكترونات، يتم عمل على أساس قاعدة اوف باو، التي تركز على نسب الإلكترونات إلى المدارات. وبالنسبة لمعظم الحالات التي نهتم من خلالها بعدد الإلكترونات، N ، سوف يشغل مدار الطاقة الأدنى $N/2$ ، عدد مزدوج.

إن الإلكترونات غير متميزة. إذا قمنا بتبديل أي زوج من الإلكترونات، فإن توزيع الكثافة يبقى نفسه. وفقاً لتفسير برون، إن كثافة الإلكترون تساوي مكعب الدالة الموجية. لذلك إن الدالة الموجية يجب أن لا تتغير أيضاً عندما يتم تبديل اثنين من الإلكترونات، وإلا فإنه يجب تغيير العلامة. في الواقع إن الدالة الموجية مطلوبة بالنسبة للإلكترونات من أجل تغيير العلامة، وهذا ما يُعرف بمبدأ عدم التناظر.

Eq.2,15

$$\alpha\left(\frac{1}{2}\right) = 1, \alpha\left(-\frac{1}{2}\right) = 0, \beta\left(+\frac{1}{2}\right) = 0, \beta\left(-\frac{1}{2}\right) = 1$$

2.3.1 The Born-Oppenheimer Approximation/ مقارنة بورن-أوبنهايمر

The electronic wavefunction depends only on the positions of the nuclei and not on their momenta. Under the Born-Oppenheimer approximation the total wavefunction for the molecule can be written in the following form:

Eq.2,16

$$\Psi_{tot}(nuclei, electrons) = \Psi(electrons)\Psi(nuclei)$$

The total energy equals to the sum of the nuclear energy and the electronic energy. The electronic energy comprises the kinetic and potential energy of the electrons moving in the electrostatic field of the nuclei, together with electron-electron repulsion:

Eq.2,17

$$E_{tot} = E(electrons) + E(nuclei)$$

تعتمد الدالة الموجية الالكترونية فقط على مواقع النوى وليس على عزمها. وبموجب تقريب بورن-أوبنهايمر، يمكن كتابة الدالة الموجية الإجمالية للجزيء على الشكل التالي :

يساوي إجمالي الطاقة مجموع الطاقة النووية والطاقة الالكترونية. تضم الطاقة الالكترونية، الطاقة الحركية والطاقة المحتملة من الإلكترونات المتحركة في الحقل الكهربائي للنوى، جنبا إلى جنب مع تباعد الإلكترون-الإلكترون.

2.3.2 General Polyelectronic Systems and Slater Determinants / أنظمة الإلكترون المتعدد العامة و محددات سلاتر

A determinant is the most convenient way to write down the permitted functional forms of a Polyelectronic wavefunction that satisfies the antisymmetry principle. In general, if we have N electrons in spin orbitals X_1, X_2, \dots, X_N then an acceptable form of the wavefunction is:

إن المحدد هو الطريقة الأكثر ملائمة لكتابة الأشكال الوظيفية المتاحة للدالة الموجية المتعددة الإلكترونات التي تُطبق مبدأ عدم التناظر. بشكل عام، إذا كان لدينا N إلكترونات في المدارات الغزلية X_1, X_2, \dots, X_N ، فإن شكل الدالة الموجية الملائم هو:

Eq.2,18

$$\psi = \frac{1}{\sqrt{N!}} \begin{vmatrix} X1(1) & X2(1) & \dots & XN(1) \\ X1(2) & X2(2) & \dots & XN(2) \\ \vdots & \vdots & & \vdots \\ X1(N) & X2(N) & \dots & XN(N) \end{vmatrix}$$

X1(1): indicates a function that depends on the space and spin coordinates of the electron labeled '1'.

$\frac{1}{\sqrt{N!}}$: ensures that the wavefunction is normalized.

This functional form of the wavefunction is called a Slater Determinant and is the simplest form of an orbital wavefunction that satisfies the antisymmetric principle.

(If any two rows of determinant is identical, then the determinant vanishes)

When the Slater determinant is expanded, a total of N! terms results. This is because N! different permutations of N electrons.

For example, for the three-electron system the determinant is

$$\psi = \frac{1}{\sqrt{12}} \begin{vmatrix} X1(1) & X2(1) & X3(1) \\ X1(2) & X2(2) & X3(2) \\ X1(3) & X2(3) & X3(3) \end{vmatrix}$$

Expansion of the determinant gives the following expression:

$$X1(1)X2(2)X3(3) - X1(1)X3(2)X2(3) + X2(1)X3(2)X1(3) - X2(1)X1(2)X3(3) + X3(1)X1(2)X2(3) - X3(1)X2(2)X1(3)$$

X1(1): تدل على وظيفة متعلقة بالفضاء وإحداثيات الغزل للإلكترون "1".

$\frac{1}{\sqrt{N!}}$: يضمن إن الدالة الموجية منسبةً أحاديًا.

هذا الشكل الوظيفي للدالة الموجية يسمى مُحدد سلاتر وهو الشكل الأبسط لمدار الدالة الموجية التي يُنفذ شروط مبدأ عدم التناظر.

(إذا كان هناك تطابق بين صفين من المحدد ، يؤدي ذلك إلى اختفاء المحدد)

ينتج عن توسع مُحدد السلاتر، مجموعة من N! مصطلح . وذلك بسبب الـ N! تبديل مختلف لـ N إلكترون. مثال: إن المحدد لنظام ذو ثلاثة إلكترونات هو:

ينتج عن امتداد المحدد، العبارة الجبرية التالية:

This expansion contains six terms ($\equiv 3!$). The six possible permutations of three electrons are: 123,132,213,231,312,321. Some of these permutations involve single exchanges of electrons; others involve the exchange of two electrons. For example, the permutation 132 can be generated from the initial permutation by exchanging electrons 2 and 3 (If we do so we will obtain the wavefunction with a changed sign $-\Psi$). By contrast, the permutation 312 requires that electrons 1 and 3 are exchanged and then electrons 1 and 2 are exchanged. (This gives rise to an unchanged wavefunction).

In general an odd permutation involves an odd number of electron exchanges and leads to a wavefunction with a changed sign; an even permutation involves an even number of electron exchanges and returns the wavefunction

هذا الامتداد يحتوي على ستة حدود ($\equiv 3!$). إن التباديل الستة الممكنة للإلكترونات الثلاثة هي: 123,132,213,231,312,321. تنطوي بعض هذه التباديل على تبادلات مفردة من الإلكترونات، في حين ينطوي البعض الآخر على تبادل اثنين من الإلكترونات. مثلاً، يمكن أن نحصل على التبدلة 132 من خلال التبدلة الأولية عبر تبديل الإلكترون 2 والإلكترون 3 (إذا قمنا بذلك، سنحصل على الدالة الموجية مع تغيير بالعلامة $-\Psi$). وبالعكس، تتطلب التبدلة 312 تبديل الإلكترونات 1 و 3 ومن ثم تبديل الإلكترونات 1 و 2 (هذا ما يسبب دالة موجية غير متغيرة).

بشكل عام، تنطوي التبدلة المفردة على تبادل عدد مفرد من الإلكترونات مما يؤدي إلى تغيير علامة الدالة الموجية؛ تنطوي التبدلة المزدوجة على تبادل عدد مزدوج من الإلكترونات ويعيد الدالة الموجية دون تغيير.

The Slater determinant can be reduced to a shorthand notation. In one system of the various notation systems, the terms along the diagonal of the matrix are written as a single-row determinant

يمكن تقليص محدد السلاتر إلى مجموعة مختزلة. من إحدى طرق الإختزال المختلفة، تتم كتابة الحدود الموجودة على طول قطري المصفوفة كصف محدد مفرد.

Eq.2,19

$$\begin{vmatrix} X1(1) & X2(1) & X3(1) \\ X1(2) & X2(2) & X3(2) \\ X1(3) & X2(3) & X3(3) \end{vmatrix} \equiv |X1 \ X2 \ X3|$$

The normalization factor is assumed. It is often convenient to indicate the spin of each electron in the determinant; this is done by writing a bar

إن عامل التنسيب الأحادي ضروري. غالباً ما يكون مناسب للإشارة إلى غزل كل إلكترون في المحدد؛ ويتم ذلك عن طريق كتابة

when the spin part is β (spin down); a function without a bar indicates a spin (spin up). Thus, the following are all commonly used ways to write the Slater determinantal wave function for the Be atom (which has the electronic configuration $1s^2 2s^2$)

شريط أفقي فوق الوظيفة، عندما يكون الجزء الغزلي β (غزل إلى الأسفل)؛ أما عندما يكون الجزء الغزلي α (غزل إلى الأعلى) فإن الوظيفة تكون بدون شريط أفقي فوقها. فيما يلي جميع الطرق المستخدمة لكتابة محدد سلاتر للدالة الموجية لذرة البريليوم (توزيعها الإلكتروني هو $1s^2 2s^2$)

Eq.2,20

$$\psi = \frac{1}{\sqrt{24}} \begin{vmatrix} \phi_{1s}(1) & \bar{\phi}_{1s}(1) & \phi_{2s}(1) & \bar{\phi}_{2s}(1) \\ \phi_{1s}(2) & \bar{\phi}_{1s}(2) & \phi_{2s}(2) & \bar{\phi}_{2s}(2) \\ \phi_{1s}(3) & \bar{\phi}_{1s}(3) & \phi_{2s}(3) & \bar{\phi}_{2s}(3) \\ \phi_{1s}(4) & \bar{\phi}_{1s}(4) & \phi_{2s}(4) & \bar{\phi}_{2s}(4) \end{vmatrix}$$

$$\equiv |\phi_{1s}\bar{\phi}_{1s}\phi_{2s}\bar{\phi}_{2s}|$$

$$\equiv |1s^{-1}s \ 2s^{-2}s|$$

An important property of determinants is that a multiple of any column can be added to another column without altering the value of the determinant. This means that the spin orbitals are not unique; other linear combinations give the same energy.

إحدى الصفات المهمة للمُحدّدات هي أن مُركّب أي عامود يمكن أن يُضاف إلى عامود آخر بدون تبديل قيمة المُحدّد. هذا يعني أن غزل المدارات ليست فريدة، ويمكن للتوافق الخطية الأخرى أن تعطي الطاقة ذاتها.

2.4 Molecular Orbital Calculations / حسابات المدار الجزيئي

2.4.1 The Energy of a General Polyelectronic System/ الطاقة للنظام الإلكتروني المتعدد العام

For N n-electron system, the Hamiltonian takes the following general form:

من أجل نظام N n-إلكترون ، تتخذ الهاميلتون هذا الشكل العام:

$$\hat{H} = \left(-\frac{1}{2} \sum_{i=1}^N \nabla_i^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \dots + \frac{1}{r_{12}} + \frac{1}{r_{13}} + \dots \right)$$

A, B, C, etc: indicates the nuclei.

1, 2, 3, ...: indicates the electrons.

The Slater determinant for a system of N electrons in N spin orbitals can be written:

A, B, C... إلخ: يدل على النوى.

1, 2, 3...: يدل على الإلكترونات.

يمكن كتابة المحدد سلاتر لنظام من N إلكترونات و N مدار غزلي حسب الشكل التالي:

$$\begin{vmatrix} X_1(1) & X_2(1) & \dots & X_N(1) \\ X_1(2) & X_2(2) & \dots & X_N(2) \\ \vdots & \vdots & \dots & \vdots \\ X_1(N) & X_2(N) & \dots & X_N(N) \end{vmatrix}$$

Each term in the determinant can thus be written $X_i(1)X_j(2)X_k(3)\dots X_u(N-1)X_v(N)$ where i,j,k,\dots,u,v is a series of N integers.

As usual, the energy can be calculated from

يمكن كتابة كل حد في المحدد كـ $(1)X_j(2)X_k(3)\dots X_u(N-1)X_v(N)$ حيث i,j,k,\dots,u,v هم تسلسلات لـ N تكامل. كالعادة، يمكن احتساب الطاقة من:

$$E = \frac{\int \Psi \hat{H} \Psi}{\int \Psi \Psi}$$

$$\int \Psi \hat{H} \Psi = \int \dots \int d_{T_1} d_{T_2} \dots d_{T_N} \left\{ [X_i(1)X_j(2)X_k(3) \dots] \times \left(-\frac{1}{2} \sum_i \nabla_i^2 - (1/r_{1A}) - (1/r_{1B}) \dots + (1/r_{12}) + (1/r_{13}) + \dots \right) \times [X_i(1)X_j(2)X_k(3) \dots] \right\}$$

$$\int \Psi \Psi = \int \dots \int d_{T_1} d_{T_2} \dots d_{T_N} \{ [X_i(1)X_j(2)X_k(3) \dots] [X_i(1)X_j(2)X_k(3) \dots] \}$$

If the spin orbitals form an orthonormal set then only products of identical terms from the determinant will be non-zero when integrated over all the space.

في حال إتخذت المدارات الغزلية شكل مجموعة متعامدة ومستظمة، فإن الحدود (جمع حدّ term) المماثلة الناتجة فقط من المحدد لا

(If the spin orbitals are normalized, integral will equal 1)

(If the term involves different electrons, it will equal zero, due to the orthogonality of spin orbitals).

The numerator in the energy expression can be broken down into a series of one-electron and two-electron integrals. Each of these individual integrals has the general form:

تساوي صفر عندما تتكامل.

(إذا كانت المدارات الغزلية منسبة آحادياً، يساوي التكامل واحد) (في حال إحتواء الحدّ على إلكترونات مختلفة، فإنه يساوي صفر، بسبب تعامد مدارات الغزل).

يمكن تقسيم البسط في العبارة الجبرية إلى سلسلة من تكاملات الإلكترون الواحد وتكاملات الاثنین من الإلكترون. كل تكامل منفرد من هذه التكاملات تأخذ هذا الشكل العام:

$$\int \dots \int d_{T_1} d_{T_2} \dots [\text{term1}] \text{operator} [\text{term2}]$$

[term1] and [term2] each represent one of the $N!$ terms in the Slater determinant. To simplify this integral, we first recognize that all spin orbitals involving an electron that does not appear in the operator can be taken outside the integral. For example, if the operator is $1/r_{1A}$, then all spin orbitals other than those that depend on the coordinates of electron 1 can be separated from the integral. The orthogonality of the spin orbitals means that the integral will be zero unless all indices involving these other electrons are the same in [term1] and [term2].

For integrals that involve two-electron operators (i.e. $1/r_{ij}$), only those terms that do not involve the coordinates of the two electrons can be taken outside the integral.

يُمثّل الـ [term1] و [term2] كل حدّ من محدد السلاتر. من أجل تبسيط هذا التكامل، يجب أن ندرك أولاً أن كل مدار غزلي ينطوي على إلكترون لا يظهر في المشغّل، يمكن أن يخرج من التكامل. على سبيل المثال، إذا كان $1/r_{1A}$ هو المشغّل، فإن كل مدارات الغزل ما عدا اللواتي يعتمدن على إحداثيات الإلكترون 1، يمكن فصلهم من التكامل. إن تعامدية المدارات الغزلية تعني أن التكامل يساوي صفر إلا إذا كانت كل المؤشرات تتضمن هذه الإلكترونات الأخرى هي نفسها في [term1] و [term2].

في حالة التكاملات التي تتضمن مشغّل اثنين من الإلكترونات مثل $(1/r_{ij})$ ، فقط هذه الحدود (terms) التي لا تتضمن إحداثيات الاثنین من الإلكترونات، تستطيع أن تخرج من التكامل.

It is more convenient to write the energy expression in a concise form that recognizes the three types of interaction that contribute to the total electronic energy of the system.

First, there is the kinetic and potential energy of each electron moving in the field of the nuclei. The energy associated with the contribution for the molecular orbital X_i is often written H_{ii}^{core} and M nuclei. For N electrons in N molecular orbitals this contribution to the total energy is (the actual electron may not be 'electron 1'):

من الأفضل كتابة عبارة الطاقة الجبرية بشكل موجز يتضمن أنواع التأثير الثلاثة التي تسهم في إجمالي الطاقة الإلكترونية للنظام.

أولاً، يوجد هناك الطاقة الحركية والطاقة الوضع لكل إلكترون يتحرك داخل النوى. غالباً ما تُكتب الطاقة المرتبطة بإسهام مدار الجزيء X_i هكذا H_{ii}^{core} و M نوى. من أجل N إلكترون في N مدارات جزيء، هذا الإسهام على إجمالي الطاقة هي (الإلكترون الفعلي ليس بالضرورة 'electron 1'):

$$E_{total}^{core} = \sum_{i=1}^N \int d_{T1} X_i(1) \left(-\frac{1}{2} \nabla_i^2 - \sum_{A=1}^M \frac{Z_A}{r_{iA}} \right) X_i(1) = \sum_{i=1}^N H_{ii}^{core}$$

The second contribution to the energy arises from the electrostatic repulsion between pairs of electrons. This interaction depends on the electron-electron distance (J_{ij}). The total Coulomb contribution to the electronic energy of the system is obtained as a double summation over all electrons, taking care to count each interaction just once:

ينشأ الإسهام الثاني للطاقة من التبعاد الكهروستاتيكي بين أزواج من الإلكترونات. يعتمد هذا التبعاد على المسافة بين الإلكترون-إلكترون (J_{ij}). يتم الحصول على إجمالي إسهام كولومب لطاقة النظام الإلكترونية باعتباره جمع مزدوج على كل الإلكترونات، مع الحرص على عد كل تأثير مرة واحدة:

$$E_i^{Coulomb} = \sum_{j \neq i}^N d_{T1} d_{T2} X_i(1) X_j(2) \frac{1}{r_{12}} X_j(2) X_i(1) \\ = \sum_{j \neq i}^N d_{T1} d_{T2} X_i(1) X_i(1) \frac{1}{r_{12}} X_j(2) X_j(2)$$

The third contribution to the energy is the exchange 'interaction'.

If two electrons occupied the same region of space and had parallel spins then they could be considered to have the same set of quantum number. Electrons with the same spin thus tend to 'avoid' each other,

الإسهام الثالث للطاقة هو التبادل "التأثير".

إذا احتل اثنين من الإلكترونات نفس المنطقة في الفضاء وكان غزلهم موازياً، يكون لديهم نفس مجموعة أرقام الكم. تميل الإلكترونات ذات السبين (الغزل) المتطابقة إلى "تجنب" بعضها

and they experience a lower Coulombic repulsion, giving a lower energy. The total exchange energy is calculated by the following equation:

البعض، وتشهد عملية التبادل الكولومبي الأدنى، مما يعطي طاقة أدنى. يُحتسب إجمالي الطاقة من خلال المعادلة التالية:

$$E_{total}^{exchange} = \sum_{i=1}^N \sum_{j'=i+1}^N \iint d_{T1} d_{T2} X_i(1)X_j(2) \left(\frac{1}{r_{12}}\right) X_i(2)X_j(1) = \sum_{j=1}^N \sum_{j'=i+1}^N K_{ij}$$

K_{ij} : Energy due to the exchange.

The prime on the counter j' indicates that the summation is only over electrons with the same spin as electron i .

K_{ij} : طاقة متعلقة بالتبادل.

إن العلامة فوق العداد j' تدل على أن الجمع هو فقط على الإلكترونات ذات سبين (غزل) متطابقة مع سبين الإلكترون i .

2.4.2 Calculating the Energy from the Wavefunction: The Hydrogen Molecule / احتساب الطاقة من الدالة الموجية:

جزيء الهيدروجين

In the most popular kind of quantum mechanical calculations performed on molecules each molecular spin orbital is expressed as a linear combination of atomic orbitals (the LCAO approach)². Thus each molecular orbital can be written as a summation of the following form:

في النوع الأكثر شعبية من العمليات الحسابية لميكانيكية الكم التي تجرى على الجزيئات، يُرمز إلى كل غزل مدار جزيء بتوفيق خطي لمدارات ذرية (طريقة الاندماج الخطي للمدارات الذرية والمدارات الجزيئية). وهكذا يُمكن أن يُكتب كل مدار جزيئي كمجموع الشكل التالي:

Eq.2,21

$$\psi_i = \sum_{\mu=1}^k c_{\mu i} \phi_{\mu}$$

² LCAO is a quantum superposition of atomic orbitals and a technique for calculating molecular orbitals in quantum chemistry. (Ref:Wikipedia)/ LCAO هو تراكم الكم من المدارات الذرية وتقنية لحساب المدارات الجزيئية في كيمياء الكم

where ψ_i is a molecular orbital represented as the sum of k atomic orbitals ϕ_μ , each multiplied by a corresponding coefficient $c_{\mu i}$, and μ represents which atomic orbital is combined in the term.³ There are two electrons with opposite spins in the lowest energy spatial orbital (labeled $1\sigma_g$), which is formed from a linear combination of two hydrogen-atom $1s$ orbitals:

حيث ψ_i هو المدار الجزيئي مُمثلاً كـ مجموع k من المدارات الذرية ϕ_μ ، كل واحد مضروب بمعامل المناسبة $c_{\mu i}$ ، و μ تمثل من حيث يتم الجمع مع المدار الذري في المدى. هناك نوعان من الإلكترونات مع سبينات مضادة أو معكوسة في الطاقة الأدنى للمدار المكاني (المسمى $1\sigma_g$)، والذي يتكون من توفيق خطي لاثنتين من مدارات $1s$ لذرة الهيدروجين :

Eq.2,22

$$1\sigma_g = A(1s_A + 1s_B)$$

To calculate the energy of the ground state of the hydrogen molecule for a fixed internuclear distance we first write the wavefunction as a 2×2 determinant:

من أجل احتساب طاقة الحالة القاعية لجزيء الهيدروجين للمسافة الداخلية الثابتة للنوى. علينا أن نكتب أولاً الدالة الموجية كمحدد 2×2 .

Eq.2,23

$$\Psi = \begin{vmatrix} X1(1) & X2(1) \\ X1(2) & X2(2) \end{vmatrix} = X1(1)X2(2) - X1(2)X2(1)$$

(See paragraph 2.1.1 operators) In atomic units the Hamiltonian is thus:

(راجع المقطع 2.1.1 المشغل). الهاملتون في الوحدات الذرية هي:

Eq.2,24a

$$\hat{H} = -\frac{1}{2} \nabla^2_1 - \frac{1}{2} \nabla^2_2 - \frac{Z_A}{r_{1A}} - \frac{Z_B}{r_{1B}} - \frac{Z_A}{r_{2A}} - \frac{Z_B}{r_{2B}} + \frac{1}{r_{12}}$$

Eq.2,24b

$$= \hat{H}_1 + \hat{H}_2 + (1/r_{12})$$

³ Ref: http://en.wikipedia.org/wiki/Linear_combination_of_atomic_orbitals_molecular_orbital_method المصدر:

<p>1 and 2: indicate the electrons. A and B: indicate the nuclei. Z_A and Z_B: nuclear charges =1. The energy of this hydrogen molecule:</p>	<p>A, B: يدل على النوى. 1, 2: يدل على الإلكترونات. Z_B و Z_A شحنة النوى تساوي 1. طاقة جزيء الهيدروجين:</p>
--	--

Eq.2,25

$$E = \frac{\int_{-\infty}^{+\infty} \Psi^* \hat{H} \Psi dT}{\int_{-\infty}^{+\infty} \Psi^* \Psi dT}$$

<p>The normalization constant for the wavefunction of the two electrons hydrogen molecule is $1/\sqrt{2}$ and so the denominator in Eq.2, 25 is equal to 2. Substitution of hydrogen molecule wavefunction into Eq.2, 25</p>	<p>التنسيب الأحادي الثابت للدالة الموجية لإلكتروني الهيدروجين هو $1/\sqrt{2}$ و المقام في المعادلة 2, 25 تساوي 2. تبدل الدالة الموجية لجزيء الهيدروجين في المعادلة 2, 25.</p>
---	--

Eq.2,26

$$E = \frac{1}{2} \iint dT_1 dT_2 \{ [X_1(1)X_2(2) - X_2(1)X_1(2)] [\hat{H}_1 + \hat{H}_2 + (1/r_{12})] [X_1(1)X_2(2) - X_2(1)X_1(2)] \}$$

Eq.2,27

$$\begin{aligned}
 E = & \iint dT_1 dT_2 X_1(1)X_2(2) (\hat{H}_1) X_1(1)X_2(2) \\
 & - \iint dT_1 dT_2 X_1(1)X_2(2) (\hat{H}_1) X_2(1)X_1(2) + \dots \\
 & + \iint dT_1 dT_2 X_1(1)X_2(2) (\hat{H}_2) X_1(1)X_2(2) \\
 & - \iint dT_1 dT_2 X_1(1)X_2(2) (\hat{H}_2) X_2(1)X_1(2) + \dots \\
 & + \iint dT_1 dT_2 X_1(1)X_2(2) \left(\frac{1}{r_{12}}\right) X_2(1)X_1(2) \\
 & - \iint dT_1 dT_2 X_1(1)X_2(2) \left(\frac{1}{r_{12}}\right) X_2(1)X_1(2) + \dots
 \end{aligned}$$

Each of these individual terms can be simplified if we recognize that terms dependent upon electrons other than those in the operator can be separated out. For example, the first term in the expansion, Eq.2,25,is:

يمكن اختزال كل حدّ منفرد، إذا لاحظنا أن الحدود (terms) معتمدة على الإلكترونات بعكس الإلكترونات الموجودة في المحدد والتي يمكن تقسيمها. مثال على ذلك، الحدّ الأول من المعادلة Eq.2,25 :

Eq.2,28

$$\iint dT_1 dT_2 X_1(1) X_2(2) (\hat{H}_1) X_1(1) X_2(2)$$

The operator \hat{H} is a function of the coordinates of electron 1 only, so terms involving electron 2 can be separated as follows:

إن المشغل \hat{H} هو وظيفة لإحداثيات الإلكترون 1 فقط، إذاً يمكننا فصل المصطلحات المتعلقة بالإلكترون 2 كالتالي:

Eq.2,29

$$\iint dT_1 dT_2 X_1(1) X_2(2) (\hat{H}_1) X_1(1) X_2(2) = \int dT_2 X_2(2) X_2(2) \int dT_1 X_1(1) \left(-\frac{1}{2} \nabla_1^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) X_1(1)$$

If the molecular orbitals are normalized, the integral $\int dT_2 X_2(2) X_2(2) = 1$.

في حال كانت مدارات الجزيء منسبة آحادياً، فإن التكامل $\int dT_2 X_2(2) X_2(2)$ يساوي 1.

Eq.2,30

$$\int dT_1 X_1(1) \left(-\frac{1}{2} \nabla_1^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) X_1(1) = \int d_v 1\sigma_g(1) \left(-\frac{1}{2} \nabla_1^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) 1\sigma_g(1) \int d_\sigma \alpha(1) \alpha(1)$$

d_v indicates integration over spatial coordinates.
 d_σ indicates integration over the spin coordinates.
 The integral over the spin coordinates =1.
 Now we can substitute the atomic orbital combination for $1\sigma_g$:

يشير d_v على مدى تكامل الإحداثيات المكانية.
 يشير d_σ على مدى تكامل الإحداثيات الغزبية. إن التكامل عبر الإحداثيات الغزبية يساوي 1.

يمكننا الآن استبدال $1\sigma_g$ بقيمتها الحقيقية:

Eq.2,31

$$\int d\nu 1\sigma_g(1) \left(-\frac{1}{2} \nabla^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) 1\sigma_g(1)$$

$$= A^2 \int d\nu_1 \{1s_A(1) + 1s_B(1)\} \left(-\frac{1}{2} \nabla^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) \{1s_A(1) + 1s_B(1)\}$$

The integral in Eq.2,31 can in turn be factorized to give a sum of integrals, each of which involves a pair of atomic orbitals:

يمكن تجزئة التكامل Eq.2,3 إلى مجموعة تكاملات، يتضمن كل واحد منها زوج من المدارات الذرية:

Eq.2,32

$$\int d\nu_1 \{1s_A(1) + 1s_B(1)\} \left(-\frac{1}{2} \nabla^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) \{1s_A(1) + 1s_B(1)\}$$

$$= \int d\nu_1 1s_A(1) \left(-\frac{1}{2} \nabla^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) 1s_A(1) + \int d\nu_1 1s_A(1) \left(-\frac{1}{2} \nabla^2 - \frac{1}{r_{1A}} - \frac{1}{r_{1B}} \right) 1s_B(1)$$

$$+ \dots$$

If we apply the same procedure to the second term in Eq.2,27 :

إذا قمنا بتطبيق نفس الإجراءات على الحد في المعادلة Eq.2,27:

Eq.2,33

$$\iint dT1 dT2 X1(1) X2(2) (\hat{H}_1) X2(1) X1(2) = \int dT1 X1(1) (\hat{H}) X2(1) \int dT2 X2(2) X1(2)$$

Eq.2,34

$$\int dT2 X2(2) X1(2) = 0$$

Eq.2,34 equals zero because the molecular orbitals are orthogonal.

تساوي المعادلة Eq.2,34 صفر لأن مدارات الجزيء متعامدة.

2.4.3 The energy of a Closed-shell System/ طاقة نظام الطبقة المغلقة

In a closed-shell system containing N electrons in $N/2$ orbitals, there are two spin orbitals associated with each spatial orbital $\psi_i: \psi_{i\alpha}$ and $\psi_{i\beta}$. The electronic energy of such a system can be calculated in a manner analogous to that for the hydrogen molecule. First, there is the energy of each electron moving in the field of the bare nuclei. For an electron in a molecular orbital X_i , this contributes energy H_{ii}^{core} . If there are two electrons in the orbital then the energy is $2H_{ii}^{core}$ and for $N/2$ orbitals. The total contribution to the energy will be:

في نظام طبقة مغلقة يحتوي N إلكترونات في $N/2$ مدار، يوجد هناك اثنين من مدارات الغزل مرتبطة بكل واحد من المدارات المكانية $\psi_i: \psi_{i\alpha}$ و $\psi_{i\beta}$. يمكن احتساب الطاقة الإلكترونية بطريقة مماثلة لاحتساب طاقة جزيء الهيدروجين. أولاً، هناك طاقة كل إلكترون يتحرك في مجال النواة المجردة. من أجل إلكترون في مدار جزيء X_i ، تكون الطاقة H_{ii}^{core} . إذا كان هناك اثنين من الإلكترونات في المدار، تكون الطاقة $2H_{ii}^{core}$ لـ $N/2$ مدار. ويكون إجمالي إسهام الطاقة:

$$\sum_{i=1}^{N/2} 2H_{ii}^{core}$$

The Coulomb interaction between each pair of electrons in the same orbital must be included; there is no exchange interaction because the electrons have paired spins. The total energy is thus given as:

يجب أخذ التأثير الكولومبي بين كل زوج من الإلكترونات في نفس المدار بعين الاعتبار. ولكن لا يوجد تبادل تأثير لأن الإلكترونات لديها سبينات (غزل) مزدوجة. يكون إجمالي الطاقة إذاً:

$$J_{ii} = K_{ii}$$

$$E = 2 \sum_{i=1}^{N/2} 2H_{ii}^{core} + \sum_{i=1}^{N/2} \sum_{j=1}^{N/2} (2J_{ij} - K_{ij})$$

2.5 The Hartree-Fock Equations/ معادلات هارتري-فوك

In most electronic structure calculations we are usually trying to calculate the molecular orbitals. But for many-body problems there is no 'correct' solution; so the variation theorem provides us with a mechanism to decide whether one proposed wavefunction is 'better' than another. (The best wavefunction is the one with the *lowest energy*). The Hartree-Fock equations are obtained by imposing this condition on the expression for the energy.

The Fock operator (f_i) takes the form:

في معظم حسابات البنية الإلكترونية، نحاول عادةً احتساب مدارات الجزيء. ولكن بالنسبة للعديد من مسائل الأجسام، لا يوجد هناك أي حل "صحيح"، لذا تقدّم لنا نظرية التغير آلية لتساعدنا على تقرير ما إذا كانت الدالة الموجية المقترحة هي "أفضل" من الأخرى. (إن الدالة الموجية الأفضل هي الدالة التي تمتلك الطاقة الأدنى). يُمكن الحصول على معادلات هارترى-فوك من خلال إدخال هذا الشرط في العبارة الجبرية للطاقة.

يأخذ مُحدد فوك (f_i) الشكل التالي:

$$f_i(1) = H^{core}(1) + \sum_{j=1}^N \{J_j(1) - K_j(1)\}$$

The Fock operator for a closed-shell system, has the following form:

يأخذ مُحدد فوك (f_i) لنظام الطبقة المطبقة، الشكل التالي:

$$f_i(1) = H^{core}(1) + \sum_{j=1}^{N/2} \{2J_j(1) - K_j(1)\}$$

The Hartree-Fock equations then take on the standard eigenvalue form:

تأخذ معادلات هارترى-فوك بشكل القيمة الذاتية الأساسية.

$$f_i X_i = \epsilon_i X_i$$

2.5.1 Hartree-Fock calculations for Atoms and Slater's Rules / احتساب الهارترى-فوك للذرات وقواعد سلاتر

The Hartree-Fock equations are usually solved in different ways for atoms and molecules. For atoms, the equations can be solved numerically if it is assumed that the electron distribution is spherically symmetrical. However, these numerical solutions

تُحل معادلات هارترى-فوك عادةً للذرات بطرق مختلفة عن الجزيئات. بالنسبة للذرات، يمكن حل المعادلات رقمياً في حالة أن الإلكترونات موزعة بشكل كروي متناظر. ولكن هذه الحلول

are not particularly useful. Fortunately, analytical approximations to these solutions can be used with considerable success. These approximate analytical functions thus have the form:

الرقمية ليست دائماً مفيدة. لحسن الحظ، يُمكن استخدام التقريب التحليلي لهذه الحلول بشكل ناجح. هذه الوظائف التقريبية التحليلية تأخذ الشكل التالي:

$$\psi = R_{nl}(r)Y_{lm}(\theta, \phi)$$

Y is a spherical harmonic and R is a radial function. Slater suggested a simpler analytical form for the radial functions:

Y هي توافق كروي و R هي وظيفة شعاعية. اقترح سلاتر شكل تحليلي أبسط للوظائف الشعاعية:

$$R_{nl}(r) = (2\zeta)^{n+1/2} [(2n)!]^{-1/2} r^{n-1} e^{-\zeta r}$$

These functions are universally known as Slater type orbitals (STOs). The first three Slater functions are as follows:

تُعرف هذه الوظائف عالمياً كنوع مدارات سلاتر (STOs). تتخذ أول ثلاث وظائف سلاتر الشكل التالي:

$$R_{1s}(r) = 2\zeta^{3/2} e^{-\zeta r}$$

$$R_{2s}(r) = R_{2p}(r) = \left(\frac{4\zeta^5}{3}\right)^{1/2} r e^{-\zeta r}$$

$$R_{3s}(r) = R_{3p}(r) = R_{3d}(r) = \left(\frac{8\zeta^7}{45}\right)^{1/2} r^2 e^{-\zeta r}$$

To obtain the whole orbital we must multiply R(r) by the appropriate angular part. Slater provided a series of empirical rules for choosing the orbital exponents ζ , which are given by:

يجب ضرب R(r) بالجزء الزاوي المناسب، من أجل الحصول على المدار الكامل. اشترط سلاتر سلسلة من القواعد التجريبية لاختيار الأس، الذي يُمكن الحصول عليه من:

$$\zeta = \frac{Z - \sigma}{n^*}$$

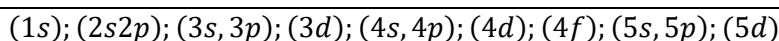
Z is the atomic number and σ is a shielding constant. n^* is an effective principal quantum number, which takes the same value as the true

Z هو عدد ذري و σ هي عدد shielding الثابت. n^* هو عدد كم رئيسي فعال، بحيث يأخذ نفس قيمة عدد الكم

principal quantum number for $n=1, 2, 3$, but for $n=4, 5, 6$ has the values 3.7, 4.0, 4.2, respectively. The shielding constant is obtained as follows:

First, divide the orbitals into the following groups:

الرئيسي الفعلي لـ $n=1,2,3$ ، أما في حالة $n=4,5,6$ يأخذ القيم التالية بالتدرج 3.7, 4.0, 4.2. يُمكن الحصول على عدد shielding الثابت من خلال:
أولاً، تقسيم المدارات إلى المجموعات التالية:



For a given orbital, σ is obtained by adding together the following contributions:

- Zero from an orbital further from the nucleus than those in the group;
- 0.35 from each other electron in the same group, but if the other orbital is the 1s then the contribution is 0.3;
- 1.0 for each electron in a group with the quantum number 1 fewer than the current orbital.;
- For each electron with a principal quantum number 1 fewer than the current orbital: 1.0 if the current orbital is d or f; 0.85 if the current orbital is s or p.

The shielding constant for the valence electrons of silicon is obtained using Slater's rules as follows.

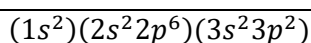
The electronic configuration of Si is :

في حالة مدار محدد، يُمكن الحصول على σ من خلال جمع الإسهامات التالية:

- صفر من المدار الأبعد عن النوى من هؤلاء الموجودين في المجموعة.
- 0.35 من كل إلكترون في نفس المجموعة، ماعدا في حالة، إذا كان المدار الآخر 1s يكون الإسهام 0.3.
- 1.0 لكل إلكترون في المجموعة ذو عدد كم يساوي 1 أقل من المدار الحالي.

(d) لكل إلكترون ذو عدد كم رئيسي يساوي 1 أقل من المدار الحالي: 1.0 في حالة أن المدار الحالي d أو f، 0.85 إذا كان المدار الحالي s أو p.

يُمكن الحصول على عدد shielding الثابت للإلكترونات المتكافئة للسيليكون باستخدام قواعد السلاتر على النحو التالي. التوزيع الإلكتروني للسيليكون Si هو:



We therefore count 3×0.35 under rule (b), 2.0 under rule (c) and 8×0.85 under rule (d), giving a total of 9.85. When subtracted from the atomic number (14) this gives 4.15 for the value of $Z-\sigma$.

بناءً على ذلك نحصي 3×0.35 بحسب القاعدة b، 2.0 بحسب القاعدة c، و 8×0.85 بحسب القاعدة d، مما ينتج مجموع يساوي 9.85. في حال حسم هذا المجموع من 14، يتم الحصول على

2.5.2 Linear Combination of Atomic Orbitals (LCAO) in Hartree-Fock Theory/ التوافق الخطي لمدارات الذرة في نظرية

هارتري-فوك

The most popular strategy, to find solution of the Hartree-Fock for the molecules, is to write each spin orbital as a linear combination of single electron orbitals:

الإستراتيجية الأكثر شعبية، لإيجاد حل لمعادلة هارتري-فوك للجزيئات، هي كتابة كل غزل مداري كتوافق خطية لمدارات الإلكترون المفرد.

$$\psi_i = \sum_{v=1}^k c_{vi} \phi_v$$

The one-electron orbitals ϕ_v are commonly called basis functions and often correspond to the atomic orbitals.

K: number of basis functions.

At the Hartree-Fock limit the energy of the system can be reduced no further by the addition of any more basis functions; however, it may be possible to lower the energy below the Hartree-Fock limit by using a functional form of the wavefunction that is more extensive than the single Slater determinant.

For a given basis set and a given functional form of the wavefunction (i.e. a Slater determinant) the best set of coefficients c_{vi} is that for which the energy is minimum, at which point

تُعرف مدارات الإلكترون الواحد ϕ_v بالوظائف الأساسية وغالباً ما تدل على المدارات الذرية.

K: عدد الوظائف الأساسية.

عند حدّ الهارتري-فوك، يُمكن تخفيض طاقة النظام من خلال إضافة أي وظيفة من الوظائف الأساسية، يمكن تخفيض الطاقة تحت حدّ الهارتري-فوك باستخدام الشكل الوظيفي للدالة الموجية التي تعتبر أكثر شمولاً من مُحدّد سلاتر المفرد.

إن أفضل مجموعة معامل c_{vi} لمجموعة أساسية مُحددة و شكل وظيفي محدد للدالة الموجية (أي مُحدد سلاتر)، هي حيث تكون الطاقة بجدها الأدنى في هذه النقطة

$$\frac{\partial E}{\partial c_{vi}} = 0$$

for the coefficients C_{vi} . The objective is thus to determine the set of coefficients that gives the lowest energy for the system.

معامل C_{vi} . إن الهدف إذاً هو تحديد مجموعة المعامل التي تعطي أقل طاقة للنظام.

2.5.3 Closed-shell Systems and the Roothaan-Hall Equations/ نظام الطبقة المغلقة ومعادلات روثن-هال

We shall initially consider a closed-shell system with N electrons in $N/2$ orbitals. The derivation of the Hartree-Fock equations for such a system was first proposed by Roothaan [Roothaan 1951] and (independently) by Hall [Hall 1951]. Unlike the integro-differential form of the Hartree-Fock equations, Roothaan and Hall recast the equations in matrix form, which can be solved using standard techniques and can be applied to systems of any geometry.

The standard form for the expression for the Fock matrix in the Roothaan-Hall equations:

سوف نعتبر ،بشكل أولي، نظام الطبقة المغلقة مع N إلكترون في $N/2$ مدار. تم إقتراح إستنتاج معادلات الهارتري-فوك لمثل هذا النظام، من قبل [Roothaan 1951] Roothaan و(بشكل مستقل) [Hall 1951] Hall. بخلاف شكل integro-differential لمعادلات الهارتري-فوك، أعاد روثن وهال صياغة المعادلات إلى شكل مصفوفة، بحيث يُمكن حلها باستخدام تقنيات أساسية يُمكن استخدامها على أي نظام جيومترى. الشكل الأساسي للعبارة الجبرية لمصفوفة فوك في معادلات روثن-هال:

$$F_{\mu\nu} = H_{\mu\nu}^{core} + \sum_{\lambda=1}^K \sum_{\sigma=1}^K P_{\lambda\sigma} \left[(\mu\nu|\lambda\sigma) - \frac{1}{2}(\mu\lambda|\nu\sigma) \right]$$

2.5.4 Solving the Roothaan-Hall Equations / حل معادلات روثن-هال

The Fock matrix is a $K \times K$ square matrix is symmetric if real basis functions are used.

The Roothaan-Hall equations can be conveniently written as a matrix equation:

تكون مصفوفة فوك $K \times K$ مربع مصفوفة متناظرة، في حال كانت الوظائف الأساسية مستعملة. يمكن كتابة معادلات روثن-هال على نحو ملائم كمعادلة مصفوفة:

FC=SCE

The elements of the $K \times K$ matrix C are the coefficients C_{vi} :

عناصر $K \times K$ مصفوفة C

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,K} \\ C_{2,1} & C_{2,2} & \dots & C_{2,K} \\ \vdots & \vdots & & \vdots \\ C_{K,1} & C_{K,2} & \dots & C_{K,K} \end{pmatrix}$$

E is a diagonal matrix whose elements are the orbital energies:

E هي قُطر مصفوفة بحيث أن عناصرها هي طاقات المدار:

$$E = \begin{pmatrix} \varepsilon_1 & 0 & \dots & 0 \\ 0 & \varepsilon_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \varepsilon_K \end{pmatrix}$$

A common scheme for solving the Roothaan-Hall equations is as follows:

المخطط الشائع لحل معادلات الروثان-هال هو كالتالي:

1. Calculate the integrals to form the Fock matrix, F .
2. Calculate the overlap matrix, S .
3. Diagonalise S .
4. Form $S^{-1/2}$.
5. Guess, or otherwise calculate, an initial density matrix, P .
6. Form the Fock matrix using the integrals and the density matrix P .
7. Form $F' = S^{-1/2} \cdot F \cdot S^{-1/2}$.
8. Solve the secular equation $|F' - EI| = 0$ to give the eigenvalue E and the eigenvectors C' by diagonalising F' .
9. Calculate the molecular orbital coefficients, C from $C = S^{-1/2} \cdot C'$.
10. Calculate a new density matrix, P , from the matrix C .
11. Check for convergence. If the calculation has converged, stop. Otherwise repeat from step 6 using the new density matrix, P .

1. احتساب المعامل إلى شكل مصفوفة فوك، F .

2. احتساب تداخل المصفوفة، S .

3. تشخيص S .

4. تشكيل $S^{-1/2}$.

5. تخمين، أو بطريقة أخرى احتساب، كثافة المصفوفة

الأساسية، P .

6. تشكيل مصفوفة فوك باستخدام المعامل وكثافة المصفوفة

P .

7. تشكيل $F' = S^{-1/2} \cdot F \cdot S^{-1/2}$.

8. حل المعادلة $|F' - EI| = 0$ من أجل الحصول على القيمة

الذاتية E والمتجهات الذاتية C' عبر تشخيص F' .

9. احتساب معامل المدار الجزئي، C من $C = S^{-1/2} \cdot C'$.

10. احتساب كثافة جديدة للمصفوفة، P ، من المصفوفة C .

11. التحقق من وجود تقارب. في حال أن الحساب قد تقارب،

This procedure requires an initial guess of the density matrix, P .

The result of a Hartree-Fock calculation is a set

of K molecular orbital, where K is the number of basis functions in the calculation. The N electrons are then fed into these orbitals in accordance with the Aufbau principle, two electrons per orbital, starting with the lowest energy orbitals. The remaining orbitals do not contain any electrons; these are known as the virtual orbitals.

يجب الوقوف. وإلا يجب تكرار الخطوات ابتداءً من الخطوة 6 مع استخدام الكثافة الجديدة للمصفوفة P . يتطلب هذا الإجراء تخمين أولي لكثافة المصفوفة P . إن نتيجة العملية الحسابية هارتري-هول هي مجموعة من k مدار جزئي، بحيث k هو عدد الوظائف الأساسية في العملية الحسابية. تقوم الـ N إلكترونات بملء المدارات وفقاً لقاعدة أوف باو، اثنين من الإلكترونات بالمدار الواحد، ابتداءً من المدارات ذات الطاقة الأدنى. تُعرف المدارات المتبقية والتي لا تحتوي على أي إلكترونات بالمدارات الافتراضية.

2.5.5 A Simple Illustration of the Roothaan-Hall Approach/ توضيح بسيط لمنهج روثان-هول

Example: HeH⁺.

Objective: how the Roothaan-Hall method can be used to derive the wavefunction, for a fixed internuclear distance of 1 Å.

There are two basis functions, $1s_A$ (centered on the helium atom) and $1s_B$ (on the hydrogen).

Each wavefunction is expressed as a linear combination of the two $1s$ atomic orbitals centered on the nuclei A and B:

مثال: HeH⁺.

الهدف: معرفة كيفية استخدام طريقة روثان-هال من أجل الحصول على الدالة الموجية، لمسافة داخلية للنوى تساوي 1 Å.

هناك اثنين من الوظائف الأساسية، $1s_A$ (مركزة على ذرة الهيليوم) و $1s_B$ (على الهيدروجين).

تُعرف كل دالة موجية كتوافق خطية للمدارات الذرية $1s$ المركزة في النوى A و B:

$$\psi_1 = c_{1A}1s_A + c_{1B}1s_B$$

$$\psi_2 = c_{2A}1s_A + c_{2B}1s_B$$

Solving the Roothaan-Hall:

-1 and 2- Calculate the integrals (here there is 2

حل الروثان-هول: -1 و -2 احتساب المعامل (هنا يوجد اثنين

electron integrals) to form the Fock matrix, F, and calculate the overlap matrix, S:

The diagonal elements of the overlap matrix, S, are equal to 1.0 as each basis function is normalised; if the off-diagonal elements have smaller, but non-zero, values that are equal to the overlap between $1s_A$ and $1s_B$ for the internuclear distance chosen. The matrix S is:

من معامل الإلكترون) من أجل تشكيل مصفوفة فوك F واحتساب المصفوفة المتشابكة S:

إن قطر عناصر المصفوفة المتشابكة S ، يساوي واحد ، ككل وظيفة أساسية منسبة آحادياً. في حال أن العناصر خارج القطر تملك قيمة غير لاغية أصغر بحيث تساوي التشابك بين $1s_A$ و $1s_B$ لمسافة معينة داخل النوى. المصفوفة S هي:

$$S = \begin{pmatrix} 1.0 & 0.392 \\ 0.392 & 1.0 \end{pmatrix}$$

The core contributions $H_{\mu\nu}^{core}$ can be calculated as the sum of three 2×2 matrices comprising the kinetic energy (T) and nuclear attraction terms for the two nuclei A and B (V_A and V_B). The elements of these three matrices are obtained by evaluating the following integrals:

يمكن احتساب الإسهامات الأساسية كمجموع ثلاثة مصفوفات (2×2) تضم الطاقة الحركية (T) ومصطلحات الجذب النووي لاثنتين من النواة A و B (V_B و V_A). يمكن الحصول على عناصر المصفوفات الثلاثة من خلال تقييم المعامل التالية:

$$T_{\mu\nu} = \int dv_1 \phi_\mu(1) \left(-\frac{1}{2} \nabla^2 \right) \phi_\nu(1)$$

$$V_{A,\mu\nu} = \int dv_1 \phi_\mu(1) \left(-\frac{Z_A}{r_{1A}} \right) \phi_\nu(1)$$

$$V_{B,\mu\nu} = \int dv_1 \phi_\mu(1) \left(-\frac{Z_B}{r_{1B}} \right) \phi_\nu(1)$$

The matrices are:

المصفوفات هي:

$$T = \begin{pmatrix} 1.412 & 0.081 \\ 0.081 & 0.760 \end{pmatrix} \quad V_A = \begin{pmatrix} -3.344 & -0.758 \\ -0.758 & -1.026 \end{pmatrix} \quad V_B = \begin{pmatrix} -0.525 & -0.308 \\ -0.308 & -1.227 \end{pmatrix}$$

H^{core} is the sum of these three:

H^{core} هي جمع هذه الثلاثة:

$$H^{core} = \begin{pmatrix} -2.457 & -0.985 \\ -0.985 & -1.493 \end{pmatrix}$$

As far as the two-electron integrals are concerned, with two basis functions there are a total of 16 possible two-electron integrals. There are however only six unique two-electron integrals, as the indices can be permuted as follows:

بما أن تكاملات الإلكترونين مأخوذة بالاعتبار، مع اثنين من المعادلات الأساسية، فإن هناك مجموع 16 احتمال التكامل للإلكترونين. ولكن هناك فقط ستة معامل فريدة للإلكترونين، كما يمكن تبديل المؤشرات على الشكل التالي:

$$(i) (1s_A 1s_A | 1s_A 1s_A) = 1.056$$

$$(ii) (1s_A 1s_A | 1s_A 1s_B) = (1s_A 1s_A | 1s_B 1s_A) = (1s_A 1s_B | 1s_A 1s_A) = (1s_B 1s_A | 1s_A 1s_A) = 0,303$$

$$(iii) (1s_A 1s_B | 1s_A 1s_B) = (1s_A 1s_B | 1s_B 1s_A) = (1s_B 1s_A | 1s_A 1s_B) = (1s_B 1s_A | 1s_B 1s_A) = 0.112$$

$$(iv) (1s_A 1s_A | 1s_B 1s_B) = (1s_B 1s_B | 1s_A 1s_A) = 0.496$$

$$(v) (1s_A 1s_B | 1s_B 1s_B) = (1s_B 1s_A | 1s_B 1s_B) = (1s_B 1s_B | 1s_A 1s_B) = (1s_B 1s_B | 1s_B 1s_A) = 0.244$$

$$(vi) (1s_B 1s_B | 1s_B 1s_B) = 0.775$$

To reiterate, these integrals are calculated as follows:

للتأكيد، تحسب التكاملات على الشكل التالي:

$$(\mu\nu|\lambda\sigma) = \iint dv_1 dv_2 \phi_\mu(1)\phi_\nu(1) \frac{1}{r_{12}} \phi_\lambda(2)\phi_\sigma(2)$$

Having calculated the integrals, we are now ready to start the SCF calculation. To formulate the Fock matrix it is necessary to have an initial guess of the density matrix, P. The simplest approach is to use the null matrix in which all elements are zero. In this initial step the Fock matrix F is therefore equal to H^{core} .

The Fock matrix must be transformed to F' by pre- and post- multiplying by $S^{-1/2}$:

بعد حساب التكامل ، نحن الآن على استعداد للبدء في حساب الSCF. من أجل صياغة مصفوفة فوك إنه من الضروري أن يكون هناك تخمين الأولي لكثافة المصفوفة P. إن أبسط نهج هو استخدام المصفوفة الفارغة بحيث تساوي جميع عناصرها صفر. في هذه الخطوة الأولية تساوي مصفوفة فوك F ، H^{core} .

يجب تحويل مصفوفة فوك إلى F' من قبل وبعد الضرب بـ $S^{-1/2}$:

$$S^{-1/2} = \begin{pmatrix} -1.065 & -0.217 \\ -0.217 & 1.065 \end{pmatrix}$$

F' for the first iteration is thus:

الـ F' لأول تكرار:

$$F' = \begin{pmatrix} -2.401 & -0.249 \\ -0.249 & -1.353 \end{pmatrix}$$

Diagonalisation of F' gives its eigenvalues and eigenvectors, which are:

إن تشخيص F' يعطي القيمة الذاتية و المتجه الذاتي :

$$E = \begin{pmatrix} -2.458 & 0.0 \\ 0.0 & -1.292 \end{pmatrix} C' = \begin{pmatrix} 0.975 & -0.220 \\ 0.220 & 0.975 \end{pmatrix}$$

The coefficients C are obtained from $C=S^{-1/2} C'$ and are thus:

يمكن الحصول على المعامل C من خلال $C=S^{-1/2} C'$:

$$C = \begin{pmatrix} 0.991 & -0.446 \\ 0.022 & 1.087 \end{pmatrix}$$

To formulate P the density matrix P we need to identify the occupied orbital(s). With a two-electron system both electrons occupy the orbital with the lowest energy. At this stage the lowest-energy orbital is:

من أجل تشكيل P، نحن بحاجة لتحديد المدارات المشغولة. مع نظام الاثنين-إلكترون، تحتل كلتا الإلكترونين المدار مع الطاقة الأدنى. في هذه المرحلة الطاقة الأدنى للمدار هي:

$$\psi = 0.991 1s_A + 0.022 1s_B$$

The orbital is composed of the s orbital on the helium nucleus; in the absence of any electron-electron repulsion the electrons tend to congregate near the nucleus with the larger charge. The density matrix corresponding to this initial wavefunction is:

يتألف المدار في نواة الهيليوم من s مدار، في حال غياب تنافر الإلكترون-إلكترون، تميل الإلكترونات إلى التجمع بالقرب من النواة مع أكبر شحنة. إن كثافة المصفوفة المتعلقة بالدالة الموجية الأولية هي:

$$P = \begin{pmatrix} 1.964 & 0.044 \\ 0.044 & 0.001 \end{pmatrix}$$

The new Fock matrix is formed using P and the two-electron integrals together with H^{core} .

The complete Fock matrix is:

تتألف مصفوفة فوك الجديدة باستخدام P وتكامل الاثنين-إلكترون مع H^{core} .
إن مصفوفة فوك الكاملة هي:

$$F = \begin{pmatrix} -1.406 & -0.690 \\ -0.690 & -0.618 \end{pmatrix}$$

The energy that corresponds to this Fock matrix is -3.870 Hartree. In the next iteration, the various matrices are as follows:

تساوي الطاقة التي تتعلق بمصفوفة فوك -3.870 هارتري. في التكرار التالي، المصفوفات المنوعة هي على الشكل التالي:

$$F' = \begin{pmatrix} -1.305 & -0.347 \\ -0.347 & -0.448 \end{pmatrix} \quad E = \begin{pmatrix} -1.427 & 0.0 \\ 0.0 & -3.25 \end{pmatrix}$$

$$C' = \begin{pmatrix} 0.943 & -0.334 \\ 0.334 & 0.943 \end{pmatrix} \quad C = \begin{pmatrix} 0.931 & -0.560 \\ 0.150 & 1.076 \end{pmatrix}$$

$$P = \begin{pmatrix} 1.735 & 0.280 \\ 0.280 & 0.045 \end{pmatrix} \quad F = \begin{pmatrix} -1.436 & -0.738 \\ -0.738 & -0.644 \end{pmatrix}$$

$$\text{Energy} = -3.909 \text{ Hartree}$$

The calculation proceeds as illustrated in the table below, which shows the variation in the coefficients of the atomic orbitals in the lowest-energy wavefunction and the energy for the first four SCF iterations. The energy is converged to six decimal places after six iterations and the charge density matrix after nine iterations.

The final wavefunction still contains a large proportion of the 1s orbital on the helium atom, but less than was obtained without the two-electron integrals.

تستمر العملية الحسابية بحسب الشكل المبين في الجدول ادناه، والذي يبين تفاوت معامل المدارات الذرية في الطاقة الدنيا للدالة الموجية والطاقة لأول أربعة تكرار SCF. تُقارب الطاقة ستة أماكن عشرية بعد ستة تكرار وشحنة كثافة المصفوفة بعد تسعة تكرار.
إن الدالة الموجية النهائية لا تزال تحتوي على نسبة كبيرة من مدار س 1s لذرة الهيليوم، ولكن أقل من الذي تم الحصول عليه بدون تكامل الاثنين-إلكترون.

Iteration	C(1s _A)	C(1s _B)	Energy
1	0.991	0.022	-3.870
2	0.931	0.150	-3.909
3	0.915	0.181	-3.911
4	0.912	0.187	-3.911

Table: variation in basis set coefficients and electronic energy for the HeH⁺ molecule.

جدول: تفاوت في تعيين أسس المعامل والطاقة الإلكترونية للجزيء الـ HeH⁺.

2.6 Basis Sets / أسس المجموعات

A basis set in chemistry is a set of functions used to create the molecular orbitals, which are expanded as a linear combination of such functions with the weights or coefficients to be determined. Usually these functions are atomic orbitals *Type equation here.*, in that they are centered on atoms. Otherwise, the functions are centered on bonds or lone pairs. Pairs of functions centered in the two lobes of a p orbital have also been used.

إن المجموعات الأساسية في الكيمياء هي مجموعة من الوظائف المستعملة من أجل إنشاء مدارات الجزيء، الموسَّع على شكل توافق خطية لمثل هذه الوظائف مع الأوزان والمعامل التي يجب الحصول عليها. تكون هذه الوظائف عادةً مدارات ذرية، بحيث تكون مرتكزة في الذرات. وإلا، فإن الوظائف تكون مرتكزة على الروابط أو على الأزواج الوحيدة. إن الأزواج المرتكزة على الاثنين من فصوص مدارات p ، يتم أيضاً إستعمالها.

3 Monte Carlo Simulation Methods: أساليب محاكاة مونت كارلو

3.1 Introduction: المقدمة

The Monte Carlo simulation method occupies a special place in the history of molecular modeling, as it was the technique used to perform the first computer simulation of a molecular system. A Monte Carlo simulation generates configurations of a system by making random changes to the positions of the species present, together with their orientations and conformations where appropriate. Many computer algorithms are said to use a 'Monte Carlo' method, meaning that some kind of random sampling is employed. In molecular simulations 'Monte Carlo' is almost always used to refer to methods that use a technique called importance sampling. Importance sampling methods are able to generate states of low energy, as this enables properties to be calculated accurately. We can calculate the potential energy of each configuration of the system, together with the values of other properties, from the positions of the atoms. The Monte Carlo method thus samples from 3N-dimensional space of the positions of the particles. There is no momentum contribution in a Monte Carlo simulation, in contrast to a molecular dynamics simulation. How then can Monte Carlo simulation be used to calculate thermodynamic quantities, given that phase space is 6N-dimensional?

تحتل طريقة محاكاة مونت كارلو مكانا خاصا في تاريخ النمذجة الجزيئية، كما كانت التقنية المستخدمة لتنفيذ المحاكاة الحاسوبية الأولى من نظام الجزيئية. يولد محاكاة مونت كارلو تكوينات نظام عن طريق إجراء تغييرات عشوائية لمواقف الأنواع الموجودة، جنبا إلى جنب مع توجهاتها والتشكل عند الاقتضاء. ويقال إن خوارزميات الحاسوب عديدة لاستخدام أسلوب 'مونت كارلو'، مما يعني أنه يعمل نوع من عينات عشوائية. في المحاكاة الجزيئية يستخدم 'مونت كارلو' تقريبا دائما للاستناد على الأساليب التقنية التي تستخدم أهمية أخذ العينات.

أهمية طرق العينات انها قادرة على توليد الطاقة من الحالات المنخفضة الطاقة، وهذا يسمح للخصائص أن تكون محسوبة بدقة. ويمكننا حساب الطاقة الكامنة مع كل تكوين نظام، جنبا إلى جنب مع قيم الخصائص الأخرى، من مواقف الذرات. طريقة مونت كارلو عينات من الفضاء 3N الأبعاد للمواقف الجسيمات. لا يوجد زخم مساهمة في محاكاة مونت كارلو، وعلى النقيض من محاكاة ديناميات الجزيئية. ثم كيف يمكن أن تستخدم محاكاة مونت كارلو لحساب الكميات الحرارية، ونظرا لأن مساحة المرحلة 6N

To resolve this difficulty, let identical particles of mass m can be written:

الأبعاد؟

حل هذه الصعوبة ، يمكن كتابة الجزيئات المتطابقة بوزن m بالشكل الآتي:

$$Q_{NVT} = \frac{1}{N!} \frac{1}{h^{3N}} \iint dp^N dr^N \exp\left[-\frac{\hat{H}(p^N, r^N)}{k_{BT}}\right]$$

The factor $N!$ Disappears when the particles are no longer indistinguishable. $\hat{H}(p^N, r^N)$ Is the Hamiltonian that corresponds to the energy of the system? The value of the Hamiltonian depends upon the $3N$ positions and $3N$ momenta of the particles in the system

تختفي جزيئات العامل $N!$ عندما لم تعد متعذر تميزها. هل (P^N, r^N) لها ملتونيان يتوافق مع نظام طاقة؟ تعتمد قيمة هاملتونيان على $3N$ مواقع و على $3N$ زخم الجزيئات في النظام

The canonical function of an ideal gas:

الوظيفة الكنسي للغاز المثالي هي:

$$Q_{NVT} = \frac{V^N}{N!} \left(\frac{2\pi k_B T m}{h^2}\right)^{3N/2}$$

This is often written in terms of the *de Broglie thermal wavelength*, λ :

هكذا يُكتب في كثير من الأحيان بمصطلح *de Broglie thermal wavelength*:

$$Q_{NVT} = \frac{V^N}{N! \lambda^{3N}}$$

$$\text{Where } \lambda = \sqrt{h^2 / 2\pi k_B T m}$$

Any deviations from ideal gas behavior are

يعود أي انحراف في سلوك الغاز المثالي إلى

due to interactions within the system as a consequence of these interactions. So we have this partition function :

التفاعلات داخل النظام كنتيجة لهذه التفاعلات. لذلك لدينا هذه الوظيفة التقسيمية :

$$Q_{NVT} = Q_{NVT}^{ideal} + Q_{NVT}^{excess}$$

$$\text{Where } Q_{NVT}^{excess} = \frac{1}{V^N} \int dr^N \exp \left[-\frac{V(r^N)}{k_B T} \right]$$

3.2 Calculating Properties by Integration: خصائص الحساب بالتكامل

To calculate the partition function for a system of N atoms using this simple Monte Carlo integration method would involve the following steps:

1. Obtain a configuration of the system by randomly generating 3N Cartesian coordinates, which are assigned to the particles.
2. Calculate the potential energy of the configuration, $V(r^N)$.
3. From the potential energy, calculate the Boltzmann factor, $\exp(-V(r^N)/k_B T)$.
4. Add the Boltzmann factor to the accumulated sum of Boltzmann factors and the potential energy contribution to its accumulated sum and return to step1.
5. After a number, N_{trial} of iterations, the mean value of the potential energy would be calculating using:

لحساب دالة قسم النظام من ذرات N باستخدام أسلوب التكامل البسيط لمونتي كارلو يستلزم تطبيق الخطوات التالية:

- 1- الحصول على التوزيع الإلكتروني النظام عن طريق توليد عشوائي ل 3N من الإحداثيات الديكارتية، التي يتم تعيينها للذرات.
- 2- حساب الطاقة المحتملة للتوزيع الإلكتروني $V(r^N)$.
- 3- حساب عامل بولتزمان من الطاقة الكامنة، مثلاً $\exp(-V(r^N)/k_B T)$.
- 4- إضافة عامل بولتزمان إلى المبلغ المتراكم لعوامل بولتزمان ومساهمة الطاقة الكامنة إلى مبلغ المتراكم والعودة إلى الخطوة الأولى.
- 5- بعد عدد، N حالة من التكرار، فإن متوسط قيمة الطاقة الكامنة يكون حسابها باستخدام:

$$\langle V(r^N) \rangle = \frac{\sum_{i=1}^{N_{trial}} V_i(r^N) \exp[-V_i(r^N)/k_B T]}{\sum_{i=1}^{N_{trial}} \exp[-V_i(r^N)/k_B T]}$$

Unfortunately, this is not a feasible approach for calculating thermodynamic properties due to the large number of configurations that have extremely small Boltzmann factors caused by high-energy overlaps between the particles.

لسوء الحظ، هذا ليس نهجاً عملياً لحساب الخصائص الحرارية بسبب وجود عدد كبير من التكوينات التي تعتبر من العوامل الصغيرة للغاية لبولتزمان الناتجة عن تداخل الطاقة العالية بين الجسيمات.

3.3 Some Theoretical Background to the Metropolis Method: / بعض الخلفية النظرية لطريقة متروبوليس

The Metropolis algorithm generates a Markov chain of states. A Markov chain satisfies the following two conditions:

1. The outcome of each trial depends only upon the preceding trial and not upon any previous trials.
2. Each trial belongs to a finite set of possible outcomes.

يولد خوارزمية متروبوليس سلسلة ماركوف للحالات. تستوفي سلسلة ماركوف الشرطين التاليين:

1. تعتمد نتيجة كل تجربة فقط على التجربة السابقة وليست على أي تجربة سابقة.
2. كل تجربة تنتمي إلى مجموعة محدودة من النتائج المحتملة.

Condition (1) provides a clear distinction between the molecular dynamics and Monte Carlo methods, for in a molecular dynamics simulation all of the states are connected in time.

Suppose the system is in state m . we denote the probability of moving to state n as Π_{mn} the various can be considered to constitute an $N \times N$ matrix Π (the transition matrix), where N is the number of possible

يبين الشرط الأول الفرق الواضح بين الديناميات الجزيئية وأساليب مونت كارلو، في محاكاة الديناميات الجزيئية جميع الحالات ترتبط في الوقت المناسب. لنفترض أن النظام في الحالة m نحن ندل على احتمال انتقاله إلى الحالة N حيث يمكن اعتبارها مثل Π_{mn} المختلفة لتشكل $N \times N$ مصفوفة

states. Each row of the transition matrix sums to 1 (i.e. the sum of the probabilities Π_{mn} for a given m equals 1). The probability that the system is in a particular state is represented by a probability vector \mathbf{p} :

$$\mathbf{P}=(p_1, p_2, \dots, p_m, p_n, \dots, p_N)$$

Thus p_1 is the probability that the system is in state 1 and p_m the probability that the system is in state m . If $\mathbf{p}(1)$ represents the initial (randomly chosen) configuration, then the probability of the second state is given by:

$$P(2)=\mathbf{p}(1)\Pi$$

Π (المصفوفة الانتقالية) ، حيث N هو عدد ممكن من الحالات. جمع كل صف من المصفوفة الانتقالية

يساوي 1 (أي مجموع الاحتمالات Π_{mn} لمعطي m يساوي 1). احتمال أن يكون النظام في

حالة معينة يمثلته احتمال المتجه \mathbf{p} :

$$\mathbf{P}=(p_1, p_2, \dots, p_m, p_n, \dots, p_N)$$

وبالتالي \mathbf{p}_1 هو احتمال أن يكون النظام في

الحالة 1 و \mathbf{p}_m احتمال أن يكون النظام في

الحالة m . إذا $\mathbf{p}(1)$ يمثل التوزيع الإلكتروني الأولي

(اختيار عشوائي) ، إذاً الاختيار الثاني يعطى

بالشكل التالي:

$$P(2)=\mathbf{p}(1)\Pi$$

The probability of the third state is:

$$p_{(3)} = p_{(2)} \pi = p_{(1)} \pi \pi$$

The equilibrium distribution of the system can be determined by considering the result of applying the transition matrix an infinite number of times. This limiting distribution of the Markov chain is given by

$$p_{(limit)} = \lim_{n \rightarrow \infty} p_{(1)} \pi^N$$

One feature of the limiting distribution is that it is independent of the initial guess $p(1)$. The limiting or equilibrium distribution for a molecular or atomic system is one in which the probabilities of each state are proportional to the Boltzmann factor. We can illustrate the use of the probability distribution and the transition matrix by considering a two-level system in which the energy levels are such that the ratio of the Boltzmann factors is 2:1.

The expected limiting distribution matrix enables the limiting distribution to be achieved:

$$\Pi = \begin{pmatrix} 0.5 & 0.5 \\ 1 & 0 \end{pmatrix}$$

We can illustrate the use of this transition matrix as follows. Suppose the initial probability vector is (1,0) and so the system starts with a 100% probability of being in state 1 and no probability of being in state 2. Then the second state is given by:

$$P(2) = (1 \ 0) \begin{pmatrix} 0.5 & 0.5 \\ 1 & 0 \end{pmatrix} = (0.5 \ 0.5)$$

The third state is $p(3) = (0.75, 0.75)$. Successive applications of the transition matrix give the limiting distribution (2/3, 1/3).

When the limiting distribution is reached then applications of the transition matrix must return the same distribution back:

احتمال الحالة الثالثة هو:

$$p_{(3)} = p_{(2)} \pi = p_{(1)} \pi \pi$$

ويمكن ان نحدد توزيع التوازن في النظام باعتبار ان نتيجة تطبيق المصفوفة الانتقالية لعدد لا حصر له من المرات. و المعادلة التالية تقدم التوزيع المحدود من سلسلة ماركوف:

$$p_{(limit)} = \lim_{n \rightarrow \infty} p_{(1)} \pi^N$$

واحدة من ميزات التوزيع المحدود هو أنه مستقل عن التخمين الأولي $P(1)$. التوزيع المحدود أو المتوازن لنظام الجزيئية أو الذرية هي التي تكون فيها الاحتمالات لكل حالة متناسبة مع عامل بولتزمان. يمكننا توضيح استخدام التوزيع للاحتمالية و للمصفوفة الانتقالية من خلال اعتبار النظام من مستويين حيث مستويات الطاقة لنسبة عوامل بولتزمان هي 2:1.

ان توقع التوزيع المحدود للمصفوفة يُمكن من انجاز التوزيع المحدود الآتي:

$$\Pi = \begin{pmatrix} 0.5 & 0.5 \\ 1 & 0 \end{pmatrix}$$

يمكننا توضيح استخدام المصفوفة الانتقالية على النحو التالي. لنفترض أن ناقل الاحتمال الأولي هو (1,0) وإذا بدء تشغيل النظام مع احتمال 100% بوجوده في حالة (1) ولا يوجد أي احتمال لوجوده في الحالة (2). بعد ذلك، وتعطى الحالة الثانية عن طريق:

$$p_{limit} = p_{limit} \pi$$

Thus, if an ensemble can be prepared that is at equilibrium, then one Metropolis Monte Carlo step should return an ensemble that is still at equilibrium. A consequence of this is that the elements of the probability vector for the limiting distribution must satisfy:

$$\sum_m p_m \pi_{mn} = p_n$$

This can be seen to hold for our simple two-level example:

$$(2/3 \ 1/3) \begin{pmatrix} 1/2 & 1/2 \\ 1 & 0 \end{pmatrix} = (2/3 \ 1/3)$$

We will henceforth use the symbol (p) to refer to the limiting distribution.

Closely related to the transition matrix is the stochastic matrix, Whose elements are labeled α_{mn} . This matrix gives the probability of choosing the two states m and n between which the move is to be made. It is often known as the underlying matrix of the Markov chain. If the probability of accepting a trial move from m to n is p_{mn} then the probability of making a transition from m to n (π_{mn}) is given by multiplying the probability of choosing states m and n (α_{mn}) by the probability of accepting the trial move (p_{mn}):

$$\pi_{mn} = \alpha_{mn} p_{mn}$$

It is often assumed that the stochastic matrix α is symmetrical (i.e. the probability of choosing the states m and n is the same whether the move is made from m to n or from n to m). If the probability of state n is greater than that of state m in the limiting distribution (i.e. if the Boltzmann factor of n is greater than that of m because the energy of n is lower than the energy of m) then in the Metropolis recipe, the transition matrix element π_{mn} for progressing from m to n equals the probability of selecting the two states in the first place (i.e. π_{mn}

$$P(2) = (1 \ 0) \begin{pmatrix} 0.5 & 0.5 \\ 1 & 0 \end{pmatrix} = (0.5 \ 0.5)$$

الحالة الثالثة هي $p(3) = (0,75, 0,75)$. تعطي التطبيقات المتعاقبة للمصفوفة الانتقالية التوزيع المحدود $(2/3, 1/3)$.

عند الوصول إلى الحد من التوزيع المحدود , يجب إعادة نفس توزيع طلبات المصفوفة الانتقالية مرة أخرى:

$$p_{limit} = p_{limit} \pi$$

كذلك ، إذا كان من الممكن تحضير المجموعة التي هي في التوازن ، ثم خطوة متروبوليس مونتي كارلو التي ينبغي أن تعيد مجموعة هي أيضاً في حالة توازن. ونتيجة لذلك هو أن عناصر ناقل الاحتمال للتوزيع المحدود يجب أن تلي :

$$\sum_m p_m \pi_{mn} = p_n$$

ويمكن ملاحظة ذلك على سبيل المثال على مستويين بسيطين:

$$(2/3 \ 1/3) \begin{pmatrix} 1/2 & 1/2 \\ 1 & 0 \end{pmatrix} = (2/3 \ 1/3)$$

من الآن و صاعداً سنستعمل الرمز p لنشير الى التوزيع المحدود.

ترتبط المصفوفة الانتقالية ارتباطاً وثيقاً بالمصفوفة العشوائية ، حيث عناصره تسمى α_{mn} . هذه المصفوفة تعطي احتمال اختيار حالتين m أو n حيث بينها يجب ان تكون الحركة موجودة.

ومن المعروف في كثير من الأحيان على أنها المصفوفة

$=\alpha_{mn} (p_n \geq p_m)$). If the Boltzmann weight of the state n is less than that of state m , then probability of permitting the transition is given by multiplying the stochastic matrix element α_{mn} by the ratio of the probabilities of the state n to the previous state m .

This can be written:

$$\pi_{mn} = \alpha_{mn} (p_n \geq p_m)$$

$$\pi_{mn} = \alpha_{mn} (p_n/p_m) \quad (p_n < p_m)$$

These two conditions apply if the initial and final states m and n are different. If m and n are the same state, then the transition matrix element is calculated from the fact that the rows of the stochastic matrix sum to 1:

$$\pi_{mn} = 1 - \sum_{m \neq n} \pi_{mn}$$

Let us now try to reconcile the metropolis algorithm as outlined in section with the more formal approach that we have just developed. We recall that in the Metropolis method a new configuration n is accepted if its energy is lower than the original state m .

If the energy is higher, however, then we would like to choose the move with a probability according to Equation (8.24). This is achieved by comparing the Boltzmann factor

$$\exp(-\Delta\xi(r^N)/k_B T) \quad (\Delta\xi(r^N) = [\xi(r^N)_n - \xi(r^N)_m])$$

To a random number between 0 and 1. If the Boltzmann factor is greater than the random number then the new state is accepted. If it is smaller than the new state (m) then the new state is rejected. Thus if the energy of the new state (n) is very close to 1, and so the move is likely to be accepted. If the energy difference will be very close to 1, and so the move is likely to be accepted. If the energy difference is very large, however, then the Boltzmann factor will be close to zero and the move is

الكامنة من سلسلة ماركوف.

إذا كانت احتمالية قبول نقل التجربة من m إلى n هو p_{mn} إذا احتمال الانتقال من m إلى n هو (π_{mn}) نحصل عليه عن طريق ضرب احتمال اختيار الحالة m و n (α_{mn}) باحتمال قبول نقل الحالة (p_{mn}) :

$$\pi_{mn} = \alpha_{mn} p_{mn}$$

غالباً ما يفترض أن مصفوفة الاستوكاستك «matrice stochastique» α هي متناظرة (أي احتمال اختيار الحالة m و n هو نفسه إذا كان الانتقال يجري من m إلى n أو من n إلى m). إذا كان احتمال n أعلى من الحالة m في توزيع الحد (أي إذا كان عامل بولتزمان n أكبر من m لأن طاقة n أقل من طاقة m) حيث في وصفة متروبوليس (Metropolis)، ليتقدم عنصر المصفوفة الانتقالية π_{mn} من m إلى n يجب ان يساوي احتمال اختيار الحالتين معاً في المكان الاول

(أي $\pi_{mn} = \alpha_{mn} (p_n \geq p_m)$). إذا كان

وزن الحالة n في بولتزمان أقل من الحالة m ، حيث يمكن حساب الاحتمال الذي يسمح بالانتقال بضرب عناصر لمصفوفة الاستوكاستك (matrice stochastique) α_{mn} (بالمليون) بنسبة احتمالات الحالة n على الحالة السابقة m . يمكن كتابة هذا:

$$\pi_{mn} = \alpha_{mn} (p_n \geq p_m)$$

unlikely to be accepted.

The metropolis method is derived by imposing the condition of microscopic reversibility: at equilibrium the transition between two states occurs at the same rate. The rate of transition from a state m to state n equals the product of the population (p_m) and the appropriate element of the transition matrix (π_{mn}). Thus, at equilibrium we can write:

$$\pi_{mn}p_m = \pi_{nm}p_n$$

The Ratio of the transition matrix elements thus equals the ratio of the Boltzmann factors of the two states:

$$\pi_{mn} = \alpha_{mn} (p_n/p_m) \quad (p_n < p_m)$$

يمكن تطبيق هذين الشرطين إذا الحالة الأولية والنهائية ل m و n مختلفتين. إذا m و n هي نفس الحالة ، إذاً يتم احتساب عنصر المصفوفة الانتقالية من كون أن مجموع صفوف المصفوفة الاستوكاستك يساوي 1 :

$$\pi_{mn} = 1 - \sum_{m \neq n} \pi_{mn}$$

دعونا الآن نحاول التوفيق بين قاعدة Metropolis على النحو المبين في المقطع الاعلى مع تقريبه أكثر من المنهج الذي سنضعه للتو. ونشير إلى أن في أسلوب متروبوليس يتم قبول التكوين الجديد n إذا كانت طاقتها أقل من الحالة الأصلية m . ومع ذلك ، إذا كانت الطاقة هي أعلى من ذلك ، ثم نود اختيار هذا الانتقال مع وجود احتمال وفقاً لمعادلة (8.24). تتحقق هذه النتيجة من خلال مقارنة عوامل بولتزمان

$$\exp(-\Delta\xi(r^N)/k_B T) (\Delta\xi(r^N) = [\xi(r^N)_n - \xi(r^N)_m])$$

لرقم عشوائي بين 0 و 1. إذا كان عامل بولتزمان أكبر من الرقم العشوائي إذا الحالة الجديدة هي مقبولة. إذا كان أصغر من حالة الجديدة (m) ، إذا يتم رفض الحالة الجديدة. ثم طاقة الحالة الجديدة (n) قريبة جداً من 1 ، إذا حركة الانتقال يحتمل أن تكون مقبولة. إذا كان فرق الطاقة قريبة جداً من 1 ، إذا حركة الانتقال يحتمل أن تكون مقبولة. إذا كان فرق الطاقة كبير جداً ، إذا ، عندها

ستكون عامل بولتزمان قريبة من الصفر، و من غير المحتمل أن يكون الانتقال مقبولاً. يتم اشتقاق أسلوب metropolis من خلال فرض شرط قابلية قلب الاتجاهات المجهرية : التوازن في الانتقال بين حالتين يحدث بنفس النسبة. معدل الانتقال من الحالة m إلى الحالة n يساوي الناتج من السكان (p_m) والعنصر المناسب للمصفوفة الانتقالية (π_{mn}) . وهكذا، في توازن نستطيع كتابة :

$$\pi_{mn}p_m = \pi_{nm}p_n$$

بالتالي نسبة عناصر المصفوفة الانتقالية تساوي معدل عوامل بولتزمان في الحالتين:

$$\frac{\pi_{mn}}{\pi_{nm}} = \exp[-(\xi(r^N)_n - \xi(r^N)_m)/k_B T]$$

3.4 تطبيق أسلوب /Implementation of the Metropolis Monte Carlo Method: متروبوليس مونت كارلو

A Monte Carlo Program to simulation an atomic fluid is quite simple to construct. At each iteration of the simulation a new configuration is generated. This is usually done by making a random change to the Cartesian coordinates of a single randomly chosen particle using a random number generator. If the random number generator

وهناك برنامج لمحاكاة مونت كارلو للسائل الذري هو بسيط جداً لبناءه. في كل تكرار للمحاكاة يتم إنشاء توزيع الكتروني جديد. عادة ما يتم ذلك عن طريق إجراء تغيير عشوائي في الإحداثيات الديكارتية لواحدة من

produces numbers (ξ) in the range 0 to 1, moves in both positive and negative directions are possible if the coordinates are changed as follows:

$$X_{new}=X_{old}+(2\xi-1)\delta r_{max}$$

$$y_{new}=y_{old}+(2\xi-1)\delta r_{max}$$

$$Z_{new}=Z_{old}+(2\xi-1)\delta r_{max}$$

A unique random number is generated for each of the three directions X, Y and Z. δr_{max} is the maximum possible displacement in any direction. The energy of the new configuration is then calculated; This need not require a complete recalculation of the energy of the entire consequence, the neighbor list used by a Monte Carlo simulation must contain all the neighbors of each atom, because it is necessary to identify all the atoms which interact with the moving atom (recall that in molecular dynamics the neighbor list for each atom contains only neighbors with a higher index). Proper account should be taken of periodic boundary conditions and the minimum image convention when generating new configurations and calculating is higher in energy than its predecessor then the Boltzmann factor, $\exp(-\Delta\xi(r^N)/k_B T)$, is compared to a random number between 0 and 1. If the Boltzmann factor is greater than the random number then the new configuration is accepted; If not then it is rejected and the initial configuration is retained for the next move. This acceptance condition can be written in the following concise fashion:

$$\text{Rand}(0,1) \leq \exp(-\Delta\xi(r^N)/k_B T)$$

The size of the move at each iteration is governed by the maximum displacement, δr_{max} .

الجسيمات المختارة عشوائياً باستخدام مولد رقم عشوائي. إذا كان مولد العدد العشوائي ينتج أرقام (ξ) في نطاق 0 إلى 1، يمكن ان يتحرك في كلا الاتجاهين إيجابي وسلبي اذا تم تغيير الإحداثيات على النحو التالي:

$$X_{new}=X_{old}+(2\xi-1)\delta r_{max}$$

$$y_{new}=y_{old}+(2\xi-1)\delta r_{max}$$

$$Z_{new}=Z_{old}+(2\xi-1)\delta r_{max}$$

يتم إنشاء رقم عشوائي وحيد لكل من الاتجاهات الثلاثة X و Y و Z. δr_{max} هو النزوح إلى أقصى حد ممكن في أي من الاتجاهات. ومن ثم يتم حساب الطاقة من التوزيع الإلكتروني الجديد، وهذا لا يتطلب إعادة الحساب بكامله من الطاقة لمجموعة النتائج، القائمة القريبة المستخدمة في محاكاة مونت كارلو يجب أن تحتوي على جميع الجاورين لكل ذرة، لأنه ضروري لتحديد جميع الذرات التي تتفاعل مع الذرة المتحركة (نشير إلى أن قائمة الجاورين لكل ذرة في الديناميات الجزيئية لا تحتوي إلا على جيران ذات مؤشر مرتفع). ينبغي أن تؤخذ في الاعتبار الظروف المناسبة للحدود الدورية واتفاقية الصورة ذات الحد الأدنى عند إنشاء توزيع إلكتروني جديد وحساب أعلى في الطاقة من سابقتها، ثم عامل بولتزمان، $\exp(-\Delta\xi(r^N)/k_B T)$ ، وبالمقارنة مع عدد عشوائي بين 0 و 1. إذا كان عامل بولتزمان أكبر من الرقم العشوائي إذا يتم قبول التكوين الجديد، وإذا تعذر ذلك فيتم رفضه ويتم الاحتفاظ بالتوزيع الإلكتروني الأولي للمرحلة المقبلة. يمكن كتابة شرط القبول بطريقة موجزة

This is an adjustable parameter whose value is usually chosen so that approximately 50% of the trial moves are accepted. If the maximum displacement is too small then many moves will be accepted but the states will be very similar and the phase space will only be explored very slowly. Too large a value δr_{max} and many trial moves will be rejected because they lead to unfavorable overlaps. The maximum displacement can be adjusted automatically while the program is running to achieve the desired acceptance ratio by keeping a running score of the proportion of moves that are accepted. Every so often the maximum displacement is then scaled by a few percent: if too many moves have been accepted then the maximum displacement is increased; too few and δr_{max} is reduced.

As an alternative to the random selection of particles it is possible to move the atoms sequentially (this requires one fewer call to the random number generator per iteration). Alternatively, several atoms can be moved at once; If an appropriate value for the maximum displacement is chosen then this may enable phase space to be covered more efficiently.

As with a molecular dynamics simulation, a Monte Carlo simulation comprises an equilibration phase followed by a production phase. During equilibration, appropriate thermodynamic and structural quantities such as the total energy (and the partitioning of the energy among the various components), mean square displacement and order parameters (as appropriate) are monitored until they achieve stable values, whereupon the production phase can commence. In a Monte Carlo simulation from the canonical ensemble, the volume will change and should therefore also be monitored to ensure that a stable system

كالتالي :

$$\text{Rand}(0,1) \leq \exp(-\Delta\xi(r^N)/k_B T)$$

ويخضع حجم التحرك في كل تكرار إلى النزوح في حده الأقصى δr_{max} .

هذا هو عامل متغير قابل للتعديل وعادة ما يتم اختيار قيمتها بحيث يتم قبول حوالي 50٪ من تحركات التجربة الأولى. إذا كان الحد الأقصى من الانتقال صغير جداً، سيتم قبول تحركات كثيرة ولكن الحالات سوف تكون متشابهة جداً و سيكون استكشاف مرحلة التباعد بطيئة جداً. أيضاً سيتم رفض القيمة الكبيرة δr_{max} و عدة تجارب لأنها تؤدي إلى تداخل غير مرغوب به. ويمكن تعديل الانتقال الاعلى تلقائياً أثناء تشغيل البرنامج لتحقيق نسبة القبول المطلوبة عن طريق الاحتفاظ بدرجة تشغيل مقبولة من نسبة التحركات. هكذا غالباً يتم تحجيم كل اعلى انتقال إلى نسبة قليلة بالمئة : إذا الكثير من التحركات قد قبلت بالتالي ألاتنتقال ألعلى يتم زيادته؛ ايضاً خفض عدد قليل و δr_{max} .

كبديل لعملية اختيار الجزئيات عشوائياً , من الممكن تحريك الذرات بالتسلسل (وهذا يتطلب عدد أقل من استدعاء مولّد العدد العشوائي بالتكرار). بدلا من ذلك، يمكن نقل عدة ذرات في آن واحد؛ إذا تم اختيار قيمة مناسبة للانتقال بحده الأقصى , اذاً هذا قد يمكن تغطية مرحلة التباعد بشكل أفضل..

كما هو الحال مع محاكاة الديناميات الجزيئية ، تضم

density is achieved.

محاكاة مونت كارلو مرحلة التوازن تليها مرحلة الإنتاج. خلال التوازن، يطابق الكميات الحرارية والهيكليّة مثل مجموع الطاقة (وتقسيم الطاقة بين مختلف المكونات)، يعني ذلك مساحة الانتقال وطلب العوامل المتغيرة (حسب مقتضى الحال) يتم مراقبتها حتى تحقيق قيمة مستقرة، وعندها يمكن أن تبدأ مرحلة الإنتاج. في محاكاة مونت كارلو للمجموعة الكنسية، الحجم يتغير و لذا ينبغي بالمقابل رصدها لضمان تحقيق كثافة إستقرار النظام.

3.4.1 Random Number Generators: / العشوائية للاعداد الكهربية للمولدات

The random number generator at the heart of every Monte Carlo simulation program accessed a very large number of times, not only to generate new configuration but also to decide whether a given move should be accepted or not. Random number generators are also used in other modeling applications; for example, in a molecular dynamics simulation the initial velocities are normally assigned using a random number generator. The number produced by a random number generator are not, in fact, truly random; the same sequence of numbers should always be generated when the program in run with the same initial conditions (if not, then a serious error in the hardware or software must be suspected!). The sequences of numbers are thus often referred to as 'pseudo-random' numbers are they possess the statistical proprieties of 'true' sequences of random numbers. Most random number generators

وصول مولد العدد العشوائي عدد كبير جدا من المرات إلى قلب كل برنامج من محاكاة مونت كارلو، ليس فقط لتوليد التكوين الجديد ولكن أيضا ليقرر ما إذا كان ينبغي قبول خطوة معينة أو لا. وتستخدم أيضا المولدات الكهربية للاعداد العشوائية في تطبيقات النمذجة الأخرى، على سبيل المثال، عادة يتم تعيين السرعات الأولية في محاكاة الديناميات الجزيئية باستخدام مولد الأعداد العشوائية. العدد المنتج عبر مولد الأرقام العشوائية ليس في الواقع عشوائي حقا، عند تشغيل البرنامج في نفس الظروف الأولية ينبغي دائما أن يتولد نفس تسلسل الأرقام (إن لم يتم ذلك، يجب الاشتباه بخطأ خطير في الأجهزة أو البرامج!). إذا غالبا، تسلسل الأرقام تسمى بأرقام "شبه عشوائية" الذين يملكون الخصائص

are designed to generate different sequences of numbers if a different seeds. One simple strategy is to use the time and/or date as the seed; this is information that can often be obtained automatically by the program from the computer's operating system.

The numbers produced by a random number generator should satisfy certain statistical properties. This requirement usually supersedes the need for a computationally very fast algorithm as other parts of a Monte Carlo simulation take much more time (such as calculating the change in energy). One useful and simple test of random number generator is to break sequence of random numbers into blocks of k numbers, which are taken to be coordinates in a k -dimensional space. A good random number should give a random distribution of points. Many of the common generators do not satisfy this test because the points lie on a plane or because they show clear correlations [Sharp and bays 1992].

The *linear congruential* method is widely used for generating random numbers. Each number in the sequence is generated by taking the previous number, multiplying by a constant (the multiplier, a), adding s second constant (the increment, b), and taking the remainders when dividing by third constant (the modulus, m). The first value is the seed, supplied by the user. Thus

$$\xi[1]=\text{seed}$$

$$\xi[i]=\text{MOD}\{(\xi[i-1]\times a+b),m\}$$

The MOD function returns the remainder when the first argument is divided by the second (for example, MOD (14.5) equals 4). If the constants are chosen carefully, the linear

الإحصائية "الصحيحة" لمتواليات الأرقام العشوائية. يتم تصميم معظم مولدات الأرقام العشوائية لتوليد سلاسل مختلفة من الأرقام إذا كانت الذريات (الاصول) مختلفة. استراتيجية واحدة بسيطة هي باستخدام الوقت و / أو التاريخ حسب الاصول، وهذه هي المعلومات التي كثيرا ما يمكن الحصول عليها تلقائيا من قبل برنامج نظام تشغيل الكمبيوتر.

وينبغي للأرقام التي تنتجها مولدات الأرقام العشوائية تلبية خصائص إحصائية معينة. هذا الشرط عادة يحل محل الحاجة إلى حسابي خوارزمية سريعة جدا وأجزاء أخرى من محاكاة مونت كارلو تستغرق وقتا أكثر من ذلك بكثير (مثل حساب التغير في الطاقة). اختبار واحد مفيد وبسيط من مولد الأعداد العشوائية لكسر تسلسل الأرقام العشوائية إلى كتل من أرقام k ، التي تتخذ لتنسيقها في الفضاء ب k أبعاد. وينبغي أن يكون هناك عدد عشوائي لا بأس به يعطي توزيع عشوائي للنقاط. العديد من المولدات المنتشرة لا تلي هذا الاختبار لأن النقاط موجودة على سطح أو لأنها تظهر إرتباطات واضحة [Sharp and bays 1992].

يستخدم على نطاق واسع أسلوب *linear congruential* لتوليد أرقام عشوائية. يتم إنشاء كل عدد في المتتالية من خلال اتخاذ عدد سابق، بضرب بعدد ثابت (عامل الضرب، a)، إضافة الثابت الثاني (الزيادة، b)، وأخذ الباقي عند قسمة ثابت ثالث

congruential method generates all possible integers between 0 and $m-1$, and the period (i.e. the number of iterations before the sequence starts to repeat itself) will be equal to the modulus.

Fig 8.3:

(وحدة القياس ، m) . القيمة الأولى هي الذرات ،
والمزودة من قبل المستخدم. وبالتالي

$$\xi[1]=\text{seed}$$

$$\xi[i]=\text{MOD}\{(\xi[i-1]\times a+b),m\}$$

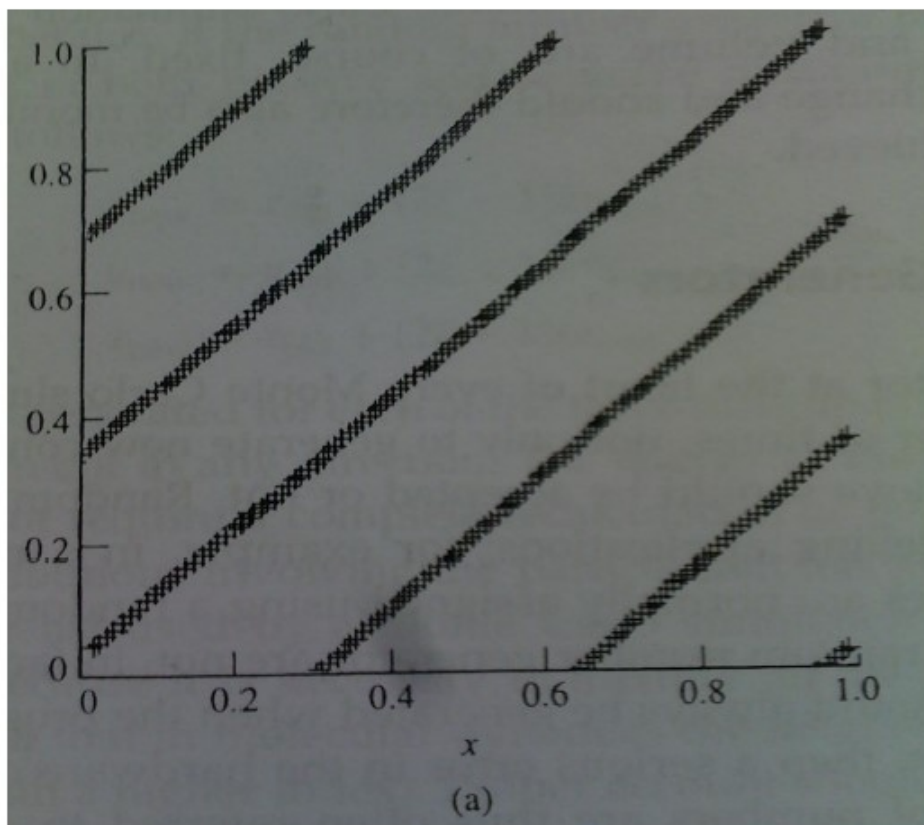
ترجع الوظيفة MOD الباقي عندما يكون الوسيط
الاول يقسم على الثاني (على سبيل المثال ، MOD
(14.5) يساوي 4). إذا تم اختيار الثوابت بعناية ، و
يولد أسلوب ال linear congruential جميع
الأعداد الصحيحة الممكنة بين 0 و $m - 1$ ، والدورة
(أي عدد التكرارات قبل ان يبدأ التسلسل بتكرار نفسه)
سوف تكون مساوية لوحدة القياس (modulus).

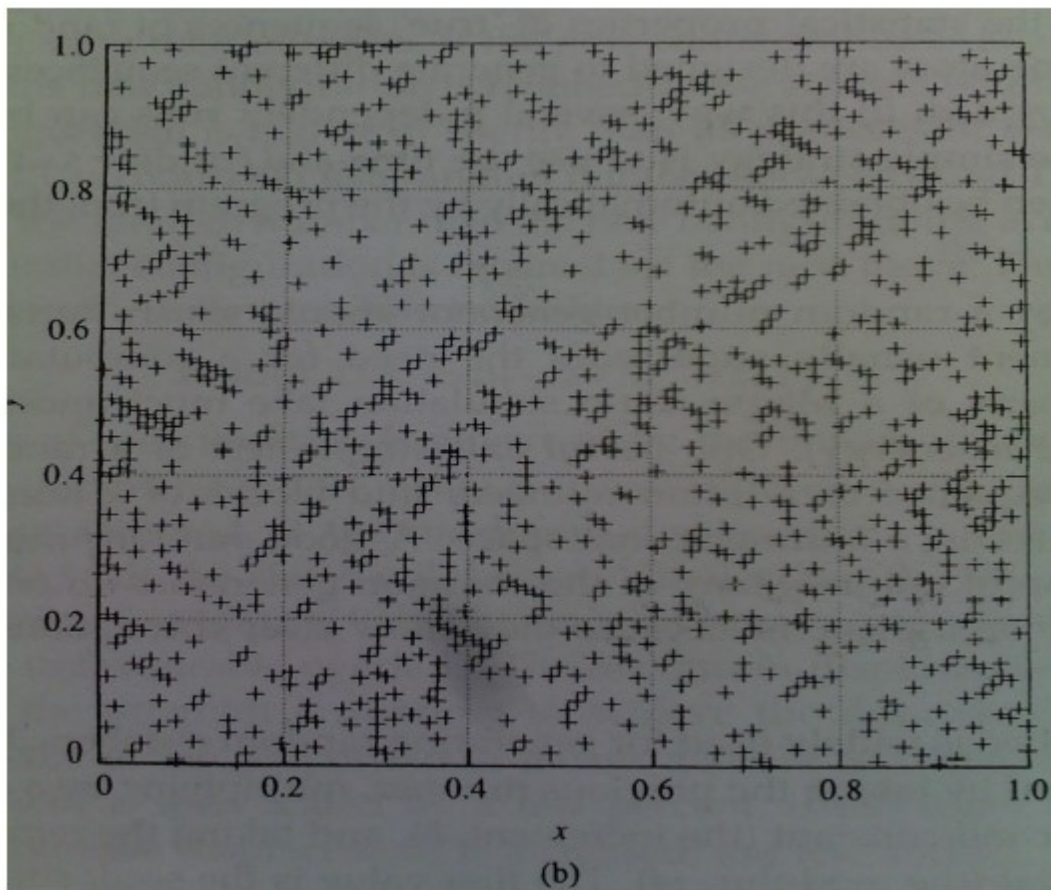
Two 'random' distributions obtained by plotting pairs of values from a linear congruential random generator. The distribution (a) was obtained using

: اثنان من التوزيعات العشوائية التي نحصل عليها
برسم أزواج القيم من مولد خطي عشوائي متطابق.
و توزيع (a) تم الحصول عليه باستخدام m

$m=32769$, $a=10924$, $b=11830$. The distribution (b) was obtained using $m=6075$, $a=106$, $b=1283$. Data from [Sharp and Bays 1992].

و $b=11830$ ، $10924 = a$ ، 32769
 التوزيع (b) تم الحصول عليه باستخدام $m = 6075$ ، $a = 106$ ، $b = 1283$ البيانات من
 [Sharp and Bays 1992]





The period cannot of course be greater than m . The linear congruential method generates integral values, which can be converted to real numbers between 0 and 1 by dividing by m . The modulus is often chosen to be the largest prime number that can be represented in a given number of bits (usually chosen to be the number of bits per word; $2^{31}-1$ is thus a common choice on a 32-bit machine).

Although popular, by virtue of the ease with which it can be programmed, the linear congruential method does not satisfy all of the requirements that are now regarded as important in a random number generator. For example, the points obtained from a linear congruential generator lie on $(k-1)$ -dimensional planes rather than uniformly filling up the space. Indeed, if the constants a , b and m are chosen inappropriately then the linear congruential method can give truly

ليس من المؤكد ان تكون الفترة اكبر من m . الاسلوب linear congruential يولد قيم صحيحة لا تتجزأ، والتي يمكن تحويلها إلى أرقام حقيقية بين 0 و 1 بقسمته على m . العامل الأكثر إختياراً ليكون أول أكبر رقم الذي يمكن ان يمثل في عدد معين من الذرات (عادة يختار ليكون عدد الذرات في الكلمة ، 2^31-1 وبالتالي اختيار مشترك على جهاز 32-bit).

على الرغم انه واسع الانتشار ، بحكم سهولة مع التي يمكن برمجتها، فإن أسلوب linear congruential لا يلبى جميع الاحتياجات التي تعد الآن مهمة في توليد الأرقام العشوائية. على سبيل المثال

terrible results, as shown in figure 8.3. One random number generator that is claimed to perform well in all of the standard tests is that of G Marsaglia, which is described in Appendix 8.1.

النقاط التي تم الحصول عليها من مولد خطي منسجم (linear congruential generator) موجودة على $(k-1)$ منأبعاد المشروع بدلا من ملء المساحة بشكل موحد. في الواقع، إذا تم إختيار الثوابت a و b و m ، بشكل غير مناسب، فإن أسلوب linear congruential يمكن أن تعطي نتائج رهيبة حقا، كما هو مبين في الصورة 8.3.

3.5 محاكاة مونت كارلو للجزيئات / Monte Carlo Simulation of molecules:

The Monte Carlo method is most easily implemented for atomic systems because it is only necessary to consider the translational degrees of freedom. The algorithm is easy to implement and accurate results can be obtained from relatively short simulations of a few tens of thousands of steps. There can be practical problems in applying the method to molecular systems, and especially to molecules which have a significant degree of conformational flexibility. This is because, in such systems, it is necessary to permit the internal degrees of freedom to vary. Unfortunately, such changes often lead to high-energy overlaps either within the molecule or between the molecule and its neighbors and thus a high rejection rate.

تطبيق أسلوب مونت كارلو هو أكثر سهولة للأنظمة الذرية لأنه ضروري فقط للنظر في درجة حرية الحركة. تنفيذ الخوارزمية سهل ويمكن الحصول على نتائج دقيقة من خلال محاكاة نسبيًا قصيرة تتألف من بضع عشرات الآلاف من الخطوات. يمكن أن يكون هناك مشاكل عملية في تطبيق الأسلوب على الأنظمة الجزيئية ، وخصوصا على الجزيئات التي لديها درجة عالية من المرونة متعلق بتكوين جزئي. هذا لأنه ، في مثل هذه الأنظمة، فمن الضروري السماح للدرجات الداخلية المتحررة ان تختلف. لسوء الحظ ، مثل هذه التغييرات كثيرا ما تؤدي إلى تداخل بطاقة عالية، سواء داخل الجزيئية أو بين جزيئات وجيرانها ، وبالتالي ترتفع نسبة الرفض.

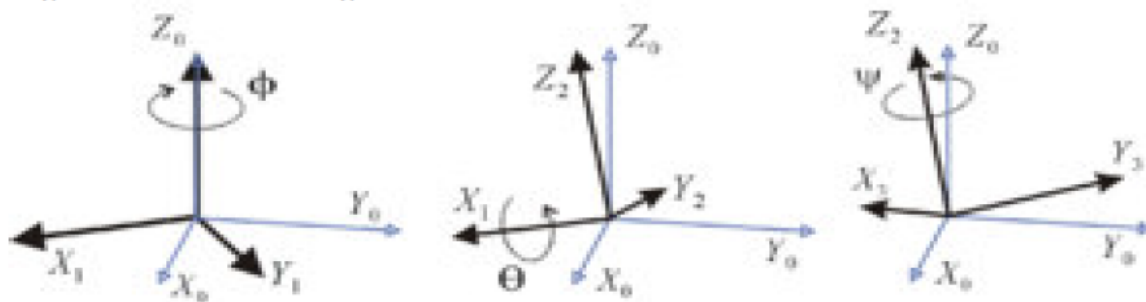
3.5.1 Rigid Molecules/الجزينات الصلبة

For rigid, non-spherical molecules, the orientations of the molecules must be varied as well as their positions in space. It is usual to translate and rotate one molecule during each Monte Carlo step. There are various ways to generate a new orientation of a molecule. The simplest approach is to choose one of the three Cartesian axes (x, y or z) and to rotate about the chosen axis by a randomly chosen angle ξw , chosen to lie within the maximum angle variation, ξw_{max} [Baker and Watts 1969]. The rotation is achieved by applying routine trigonometric relationships. For example, if the vector (x_i, y_j, z_k) describes the orientation of a molecule then the new vector (x'_i, y'_j, z'_k) that corresponds to rotation by ξw about the x axis calculated as follows:

لتجميد الجزئيات الغير كروية، يجب أن تختلف توجهات الجزئيات فضلا عن موقعها في الفضاء. ومن المعتاد نقل وتدوير جزئية خلال كل خطوة من خطوات مونت كارلو. هناك طرق مختلفة لإنشاء توجه جديد للجزئية. النهج الأبسط هو اختيار واحد من المحاور الثلاثة الديكارتية (x, y ou z) وتدويرها حول محور مختار من قبل زاوية ξw اختيرت عشوائيا، اختارت أن تقع في أقصى اختلاف للزاوية ξw [Baker et Watts 1969]. يتحقق الدوران من خلال تطبيق منتظم للعلاقات المثلثية. وتدوير حول محور اختاره زاوية اختيارها عشوائيا، اختارت أن تقع في أقصى الزاوية الاختلاف، [بيكر واتس 1969]. ويتحقق من خلال تطبيق التناوب العلاقات المثلثية روتينية. على سبيل المثال، إذا كان الناقل (x_i, y_j, z_k) يصف اتجاه الجزئية، ثم الناقل الجديد (x'_i, y'_j, z'_k) الذي يتوافق مع دوران باتجاه ξw حول المحور x الذي يُحسب كما يلي

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \delta w & \sin \delta w \\ 0 & -\sin \delta w & \cos \delta w \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

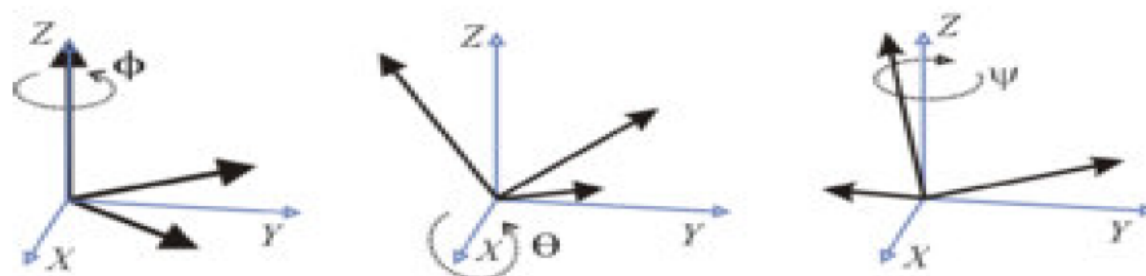
Fig. 8.4: The Euler angles.⁴



A rotation represented by Euler angles with $(\phi, \theta, \psi) = (-60^\circ, 30^\circ, 45^\circ)$ using the 3-1-3 (Z-X-Z) co-moving axes rotations

الدوران يمثله زوايا أويلر مع
 $((\phi, \theta, \psi) = (-60^\circ, 30^\circ, 45^\circ)$
 باستخدام 3-1-3 (Z-X-Z) دوران للمحاور
 المشاركة بالحركة

⁴http://en.wikipedia.org/wiki/Euler_angles



The same rotation alternatively expressed by $(\phi, \theta, \psi) = (45^\circ, 30^\circ, -60^\circ)$ using the 3-1-3 (Z-X-Z) fixed axes rotations

الدوران نفسه بدلا من
 $(\phi, \theta, \psi) = (45^\circ, 30^\circ, -60^\circ)$
 باستخدام 3-1-3 (Z-X-Z) تثبيت محاور الدوران

The Euler angles are often used to describe the orientations of a molecule. There are three Euler angles; ϕ, θ and ψ . ϕ is a rotation about the Cartesian z axis; this has the effect of moving the x and y axes. θ is a rotation about the new x axis. Finally, ψ is a rotation about the new z axis (Figure 8.4). If the Euler angles are randomly changed by small amounts $\delta\phi$, $\delta\theta$ and $\delta\psi$ then a vector V_{old} is moved according to the following matrix equation:

وغالبا ما تستخدم زوايا أويلر لوصف توجهات الجزيئة. هناك ثلاث زوايا أويلر؛ ϕ, θ and ψ . ϕ هو الدوران حول المحور الديكارتي Z، وهذا له تأثير في تحريك المحاور X و Y. θ هو الدوران حول المحور الجديد X. وأخيرا، ψ هو الدوران حول المحور الجديد Z (الصورة 8.4). إذا تغيرت زوايا أويلر بشكل عشوائي بقيمة

$V_{new} = AV_{old}$ Where the matrix A is	صغيرة $\delta\phi$ ، $\delta\psi$ و $\delta\theta$ ثم يتم نقل الناقل V_{old} وفقا لمعادلة المصفوفة التالية : $V_{new} = AV_{old}$ عندما تكون المصفوفة A :
---	---

$$\begin{pmatrix} \cos \delta\phi \cos \delta\psi - \sin \delta\phi \cos \delta\theta \sin \delta\psi & \sin \delta\phi \cos \delta\psi - \cos \delta\phi \cos \delta\theta \sin \delta\psi & \sin \delta\theta \sin \delta\psi \\ -\cos \delta\phi \sin \delta\psi - \sin \delta\phi \cos \delta\theta \cos \delta\psi & -\sin \delta\phi \sin \delta\psi - \cos \delta\phi \cos \delta\theta \cos \delta\psi & \sin \delta\theta \cos \delta\psi \\ \sin \delta\phi \cos \delta\theta & -\cos \delta\phi \sin \delta\theta & \cos \delta\theta \end{pmatrix}$$

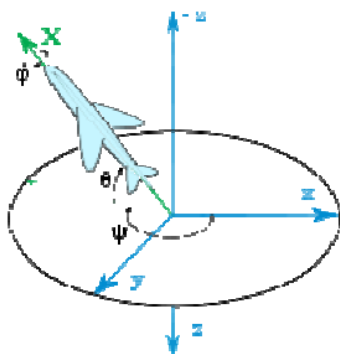
It is important to note that simply sampling displacements of the three Euler angles does not lead to uniform distribution; it is necessary to sample from $\cos \theta$ rather than θ (figure 8.5).	ومن المهم الإشارة إلى أن الفعل البسيط بأخذ عينات لتحرك زوايا أويلر الثلاثة لا يؤدي إلى توزيع موحد ، بل من الضروري أخذ عينات من $\cos \theta$ بدلا من θ (الصورة 8.5).
---	---

Fig. 8.5:⁵

الصورة 8.5 :

To achieve a uniform distribution of points over the surface of a sphere it is necessary to sample from $\cos \theta$ Rather than θ . If the sampling is uniform in θ then the number of points per unit area increases with θ , leading to an uneven distribution over the sphere.

لتحقيق توزيع موحد للنقاط على سطح الكرة ، من الضروري تعيينه من $\cos \theta$ بدلا من θ . إذا كان أخذ العينات موحد في θ في ذلك الحين عدد النقاط في وحدة المساحة يتزايد مع θ ، مما أدى إلى التوزيع غير المتكافئ على الكرة.



The preferred approach is to sample directly in $\cos \theta$ as follows:

النهج المفضل هو عينة مباشرة في $\cos \theta$ على النحو التالي :

$$\begin{aligned}\phi_{new} &= \phi_{old} + 2(\xi - 1)\delta\phi_{max} \\ \cos\phi_{new} &= \cos\phi_{old} + 2(\xi - 1)\delta(\cos\theta)_{max} \\ \psi_{new} &= \psi_{old} + 2(\xi - 1)\delta\psi_{max}\end{aligned}$$

The alternative is to sample in ϕ and to modify the acceptance or rejection criteria as follows:

البديل هو عينة في ϕ وتعديل معايير القبول أو الرفض على النحو التالي :

$$\begin{aligned}q_0 &= \cos\frac{1}{2}\phi \cos\frac{1}{2}(\phi + \psi) \\ q_0 &= \sin\frac{1}{2}\phi \cos\frac{1}{2}(\phi + \psi) \\ q_0 &= \sin\frac{1}{2}\phi \sin\frac{1}{2}(\phi + \psi) \\ q_0 &= \cos\frac{1}{2}\phi \sin\frac{1}{2}(\phi + \psi)\end{aligned}$$

The Euler angle rotation matrix can then be written

يمكن بعد ذلك كتابة تناوب مصفوفة زاوية يولر :

$$\mathbf{A} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}$$

To generate a new orientation, it is necessary to rotate the quaternion vector to a new (random) orientation. As it is a four-dimensional vector, the orientation must be performed in four-dimensional space. This can be achieved as follows [Vesely 1982]:

لإنشاء التوجه الجديد ، من الضروري تدوير الناقل الرباعي باتجاه (عشوائي) جديد. كما هو ناقل رباعي الأبعاد ، يجب إجراء التوجه في الفضاء الرباعي الأبعاد. ويمكن تحقيق ذلك على النحو التالي

<p>1. Generate pairs of random numbers (ξ_1, ξ_2) between -1 and 1 until $S_1 = \xi_1^2 + \xi_2^2 < 1$</p> <p>2. Do the same for pairs ξ_3 and ξ_4 until $S_1 = \xi_3^2 + \xi_4^2 < 1$</p> <p>3. Form the random unit four-dimensional vector $(\xi_1, \xi_2, \xi_3 \sqrt{(1 - S_1/S_2)}, \xi_4 \sqrt{(1 - S_1/S_2)})$.</p> <p>To achieve an appropriate acceptance rate the angle between the two vectors that describe the new and old orientations should be less than some value; this corresponds to sampling randomly and uniformly from a region on the surface of a sphere.</p> <p>The introduction of an orientation component as well as translational moves is made. Trial and error is often the most effective way to find best combination of parameters.</p>	<p style="text-align: right;">[Vesely1982]</p> <p>1- توليد أزواج من الأرقام العشوائية (ξ_1, ξ_2) بين -1 و 1 حتى $S_1 = \xi_1^2 + \xi_2^2 < 1$</p> <p>2- تفعل الشيء نفسه بالنسبة للأزواج ξ_3 و ξ_4 حتى $S_1 = \xi_3^2 + \xi_4^2 < 1$</p> <p>3- تشكيل وحدة ناقل عشوائي رباعي الأبعاد $(\xi_1, \xi_2, \xi_3 \sqrt{(1 - S_1/S_2)}, \xi_4 \sqrt{(1 - S_1/S_2)})$.</p> <p>لتحقيق معدل القبول المناسب للزاوية بين الناقلين اللذين يصفان التوجهات الجديدة والقديمة يجب أن تكون أقل من بعض القيم ، وهذا يتوافق مع أخذ العينات عشوائيا وبشكل موحد من منطقة على سطح الكرة. يتم إدخال عنصر التوجيه وكذلك يتحقق التحرك بالانزلاق. التجربة والخطأ وغالبا ما يكون أنجع وسيلة للعثور على أفضل مجموعة من العوامل المتغيرة في التجربة.</p>
---	--

3.5.2 Monte Carlo Simulations of Flexible Molecules: محاكاة مونت كارلو للجزيئات المرنة/

<p>Monte Carlo Simulations of flexible molecules are often difficult to perform successfully unless the system is small, or some of the internal degrees of freedom are frozen out, or special models or methods are employed. The simplest way to generate a new configuration of a flexible molecule is to perform random changes to the Cartesian coordinates of</p>	<p>محاكاة مونت كارلو للجزيئات المرنة غالبا ما تكون صعبة الأداء بنجاح إلا إذا كان النظام صغير ، أو بعض من درجات الحرية الداخلية هي مجمدة خارجه ، أو تستخدم النماذج أو الطرق الخاصة. إن أبسط طريقة لإنشاء التكوين الجديد من جزيء مرن هو إجراء</p>
---	---

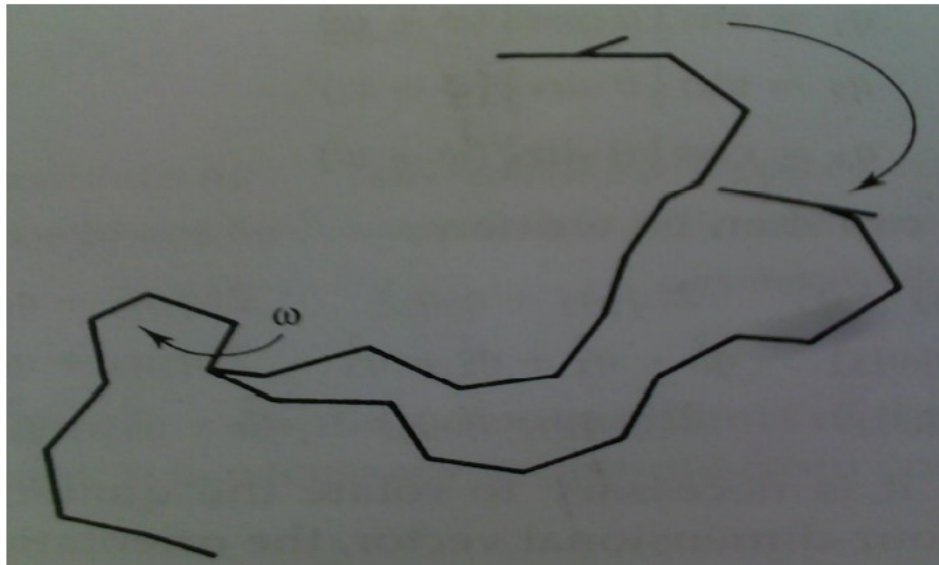
individual atoms, in addition to translations and rotations of the entire molecule. Unfortunately, it is often found that very small atomic displacements are required to achieve an acceptable acceptance ratio, which means that the phase space is covered very slowly. For example, even small movements away from an equilibrium bond length will cause a large increase in the energy. One obvious tactic is to freeze out some of the internal degrees of freedom, usually the 'hard' degrees of freedom such as the bond lengths and the bond angles. Such algorithms have been extensively used to investigate small molecules such as butane. However, for large molecules, even relatively small bond rotations may cause large movements of atoms down the chain. This invariably leads to high-energy configurations as illustrated in figure 8.6. The rigid bond and rigid angle approximation must be used with care, for freezing out some of the internal degrees of freedom can affect the distributions of other internal degrees of freedom.

تغييرات عشوائية في الإحداثيات الديكارتية من ذرات فردية ، بالإضافة إلى تحريك وتناوب الجزيء بأكمله. لسوء الحظ ، كثيرا ما وجدت أن هناك حاجة تحريك ذرية صغيرة جدا للحصول على نسبة قبول واقعية و مقبولة ، مما يعني أن تغطي مساحة المرحلة ببطء شديد. على سبيل المثال ،

حتى الحركات الصغيرة التي هي خارج عن طول رابط التوازن تسبب زيادة كبيرة في الطاقة. أسلوب واحد واضح هو لتجميد بعض درجات من الحرية الداخلية، وعادة درجات الحرية 'القاسية' مثل طول الرابط و زواياه. وقد تم استخدام هذه الخوارزميات على نطاق واسع لدراسة الجزيئات الصغيرة مثل البوتان. مع ذلك بالنسبة للجزيئات الكبيرة ، تدوير الرابط، حتى و لو كان نسبياً قليل يسبب تحركات كبيرة للذرات في السلسلة. وهذا يؤدي حتما إلى تكوينات ذات طاقة عالية كما هو موضح في الصورة 8.6. يجب استخدام القيمة التقريبية للرابط القاسي و للزاوية القاسية بحذر، لتجميد بعض درجات من الحرية الداخلية يمكن أن تؤثر على توزيع درجات الحرية الداخلية الأخرى.

Figure 8.6

A Bond rotation in the middle of a molecule may lead to a large movement at the end.



الشكل 8.6

يمكن ان يؤدي دوران الرابط في منتصف الجزيء إلى حركة كبيرة في نهاية المطاف

3.6 Models Used in Monte Carlo Simulation of Polymers/ النماذج المستخدمة في محاكاة مونت كارلو من البوليمار /

A polymer is a macromolecule that is constructed by chemically linking together a sequence of molecular fragments. In simple synthetic polymers such as polyethylene or polystyrene all of the molecular fragments comprise the same basic unit (or monomer). Other polymers contain mixtures of monomers- Proteins, for example, are polypeptide chains in which each unit one of the twenty amino acids. Cross-linking between different chains gives rise to yet further variations in the constitution and structure of polymer. All of these features may affect the overall properties of the molecule, sometimes in a dramatic way. Moreover, one may be interested in the properties of the polymer under different conditions, such as in solution, in a polymer melt or in the

بوليمر هو جزيء كبير أنشأ من خلال ربط كيميائي في نفس الوقت سلسلة من قطع جزيئية. في البوليمار البسيط الاصطناعي مثل البولي اثيلين (polyethylene) أو البوليستيرين (polystyrene) جميع القطع الجزيئية تتكون من نفس الوحدة الأساسية (أو مونومر monomer). البوليمرات الأخرى تحتوي على خليط من البروتينات ، على سبيل المثال ، هي سلاسل أحادي-البيتيد (polypeptide) التي في كل وحدة منها هناك الأحماض الأمينية العشرين. التزاوج بين السلاسل المختلفة يسبب أيضاً تغييرات في بنية وهيكل البوليمر. ويمكن لجميع هذه الملامح ان تؤثر على الخصائص العامة للجزيء ، وأحيانا بطريقة

crystalline state. Molecular modeling can help to develop theories for understanding the properties of polymers and can also be used to predict their properties.

A wide range of time and length scales are needed to completely describe a polymer's behavior. The timescale ranges from approximately 10^{-14} S (i.e. the period of a bond vibration) through to seconds, hours or even longer for collective phenomena.

The size scale ranges from the $1-2\text{\AA}$ of chemical bonds to the diameter of a coiled polymer, which can be several hundreds of \AA angstroms. Many kinds of model have been used to represent and simulate polymeric systems and predict their properties. Some of these models are based upon very simple ideas about the nature of the intra-and intermolecular interactions within the system but have nevertheless proved to be extremely useful. One famous example is Flory's rotational isomeric state model [Flory 1969]. Increasing computer performance now makes it possible to use techniques such as molecular dynamics and Monte Carlo simulations to study polymer systems.

Most simulations on polymers are performed using empirical energy models (through with faster computers and new methods it is becoming possible to apply quantum mechanics to larger and larger system). Moreover, there are various ways in which the configurationally and conformational degrees of freedom may be restricted so as to produce a computationally more efficient model. The

دراماتيكية. وعلاوة على ذلك ، قد تكون مهمة في خصائص البوليمر في ظل ظروف مختلفة ، كما هو الحال في السائل ، في بوليمر ذائب او في حالة البلورية. يمكن ان تساعد النمذجة الجزيئية على تطوير نظريات لفهم خصائص البوليمرات ويمكن أن تستخدم أيضا للتنبؤ بخصائصهم.

وهناك حاجة إلى نطاق واسع من الوقت و قياس طويل لوصف سلوك البوليمار بدقة. ويتراوح مقياس الوقت من حوالي 10^{-14} (اي فترة اهتزاز الرابط) من خلال الثواني ، ساعات أو حتى فترة أطول لظواهر مشتركة. ويتراوح حجم النطاق $1-2\text{\AA}$ من الروابط الكيميائية إلى قطر بوليمر ملفوف ، والتي يمكن عدة مئات من ال \AA ngstroms . وقد استخدمت أنواع كثيرة من النماذج لتمثيل ومحاكاة النظم البوليمرية و لتنبؤ خصائصها. وتستند بعض هذه النماذج على أفكار بسيطة جدا حول طبيعة التفاعلات البينية والجزيئات داخل النظام ولكن مع ذلك ثبت إمكانية ان يكون مفيدا للغاية. ومن الأمثلة الشهيرة في نموذج فلوري لدوران الحالة ايزوميريا [فلوري 1969] (Flory)

modèle de l'État de rotation isomères [Flory 1969]. زيادة أداء الكمبيوتر تجعل من

الممكن الآن استخدام تقنيات مثل الديناميات الجزيئية ومحاكاة مونتج كارلو لدراسة نظم البوليمار. يتم تنفيذ معظم عمليات المحاكاة على البوليمار باستخدام نماذج تجريبية للطاقة (من خلال أجهزة الكمبيوتر الأسرع مع أساليب جديدة، اصبح من الممكن تطبيق ميكانيكا الكم على نظام أكبر وأكبر). وعلاوة على ذلك، هناك طرق مختلفة التي في

simplest models use a lattice representation in which the polymer is constructed from connected interaction centers, which are required to occupy the vertices of a lattice. AT the next level of complexity are the bead models, where the polymer is composed of a sequence of connected 'beads'. Each bead represents an 'effective monomer' and interacts with the other beads to which it is bonded and also with other nearby beads. The ultimate level of detail is achieved with the atomistic models, in which each non-hydrogen atom is explicitly represented (and sometimes all of the hydrogen as well). Our aim here to is give a flavor of the way in which Monte Carlo methods can be used to investigate polymeric systems. We divide the discussion into lattice and continuum models but recognize that is a spectrum of models from the simplest to the most complex.

صورتها و في التوزيع الالكتروني درجات التحرر يمكن أن تقتصر على طريقة إنتاج نموذج حسابي فعال أكثر. تستخدم النماذج الشبكية البسيطة التمثيل التي يكون فيها البوليمر مؤلف من مراكز تفاعل متصلة، والتي هي المطلوبة لتكون في رؤوس الشبكة. في المستوى التالي من التعقيد هي نماذج الحبيبات (bead models) ، حيث يتألف البوليمر من تسلسل متصل من 'حبيبات'. كل حبة يمثل 'مونومر فعالة كل حبيبة تمثل "مونومر فعال يتفاعل مع غيرها من الحبيبات التي ترتبط بها وأيضا مع حبيبات أخرى مجاورة. ويتم تحقيق المستوى النهائي بالتفصيل مع النماذج الذرية، و فيها كل ذرة هيدروجين غير ممثلة بشكل واضح (وأحيانا مجموعة من الهيدروجين أيضاً). هدفنا هنا هو إعطاء نكهة للطريقة التي يمكن فيها استخدام اسلوب مونتج كارلو لدراسة نظم البوليمار. نقسم المناقشة الى اسلوبين الشبكية والتواصل، ولإدراك بأنه مجموعة من النماذج من الأيسر إلى الأكثر تعقيدا.

3.6.1 Lattice Models of Polymers \ نماذج شبكة البوليمار

Lattice Models have provided many insights into the behavior of polymers despite the obvious approximations involved. The simplicity of a lattice model means that many states can be generated and examined very rapidly. Both two-dimensional and three-dimensional lattices

قدّمت نماذج الشبكات رؤى في سلوك البوليمار رغم تقريبات كثيرة و واضحة مشاركة به. بساطة النموذج الشبكي يعني أن العديد من الحالات يمكن أن تتولد ويتم فحصها بشكل سريع جدا. وتستخدم

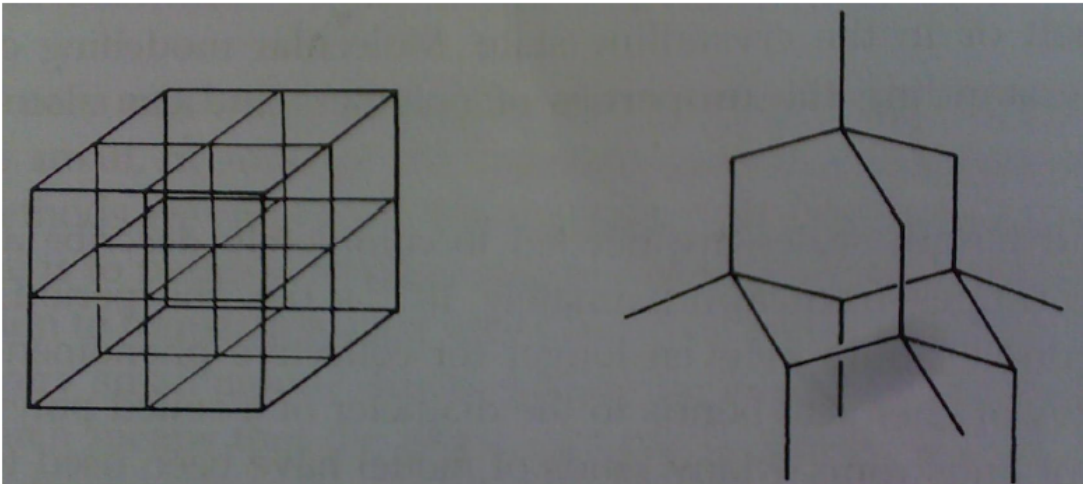
are used. The simplest models use cubic or tetrahedral lattices in models are usually very simple, in part to reflect the simplicity of the representation but also to permit the rapid calculation of the energy.

More complex models have been developed in which the lattice representation is closer to the 'true' geometry of the molecule. For example, in figure 8.8 we show the bond fluctuation model of polyethylene, in which the 'bond' between successive moments on the lattice

الشبكات الثنائية الأبعاد والثلاثية الأبعاد. النموذج الأبسط هو استخدام شبكات السطوح المكعبة أو رباعية وعادة ما تكون نماذج بسيطة جدا ، ويعكس في جزء منه إلى بساطة التمثيل ولكن أيضا للسماح بحساب سريع للطاقة. تم تطوير نماذج أكثر تعقيدا حيث فيها التمثيل الشبكي أقرب إلى الهندسة الحقيقية "الصحيحة" للجزيئة. على سبيل المثال ، في الصورة 8.8 نعرض نموذج تقلب روابط من البولي إيثيلين (polyethylene) ، والذي فيه 'روابط' بين لحظات متتالية على الشبكة.

Figure 8.7

Cubic and tetrahedral (diamond) lattices, which are commonly used for lattice simulations of polymers



الشكل 8.7

مكعب ، ورباعي السطوح (الماس) الأسوار، والتي تستخدم عادة لمحاكاة شبكيات للبوليمرات

Figure 8.8

The bond fluctuation model. In this example three bonds in the polymer are incorporated into a single 'effective bond' between 'effective monomers'. (Figure adapted Baschnagel J, K Binder, and W paul, M Laso, U suter, I Batoulis, W jilge and t burger 1991. On the construction of coarse-Grained models for linear Flexible Polymer-Chains-Distribution-Functions of Groups of consecutive Monomers. Journal of chemical Physics 95:6014-6025.)

الشكل 8.8

نموذج رابطة التقلب. في هذا المثال يتم دمج ثلاثة روابط في البوليمر إلى "رابط فعال" واحد بين "مونومرات فعالة".

(Figure adapted Baschnagel J, K Binder, and W paul, M Laso, U suter, I Batoulis, W jilge and t burger 1991.

. وفي بناء الحبيبات الكبيرة لنماذج خطية مرنة ل بوليمرات-سلاسل-توزيع -وظائف, لمجموعات المونومرات على التوالي. مجلة الفيزياء الكيميائية 95:6014-6025.)

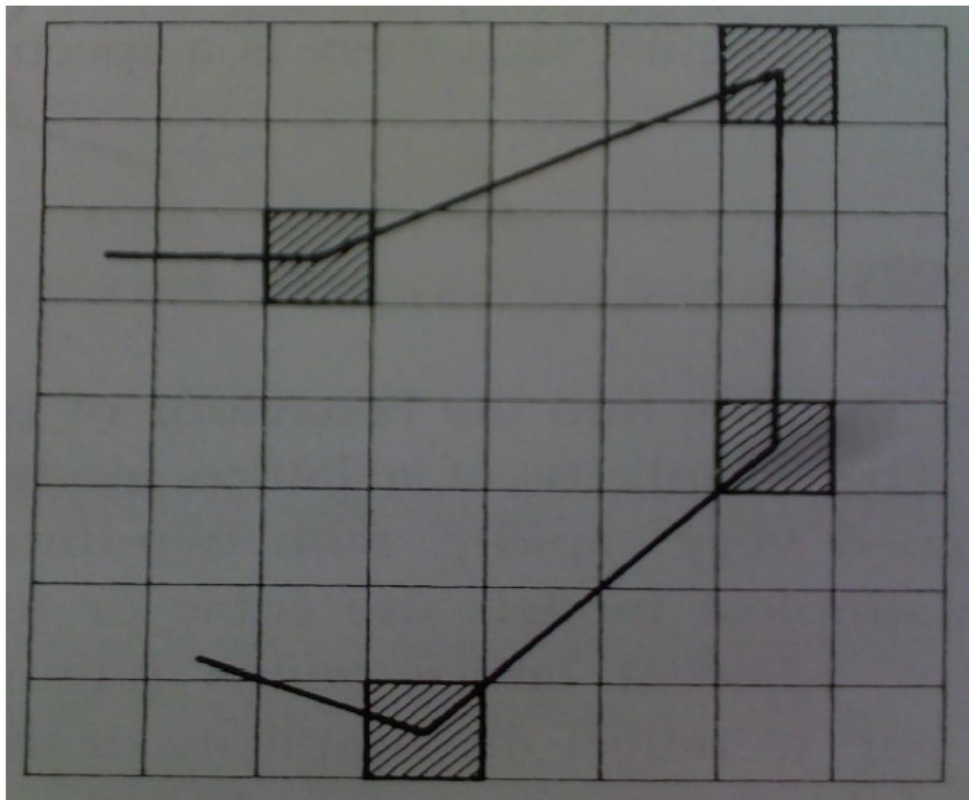
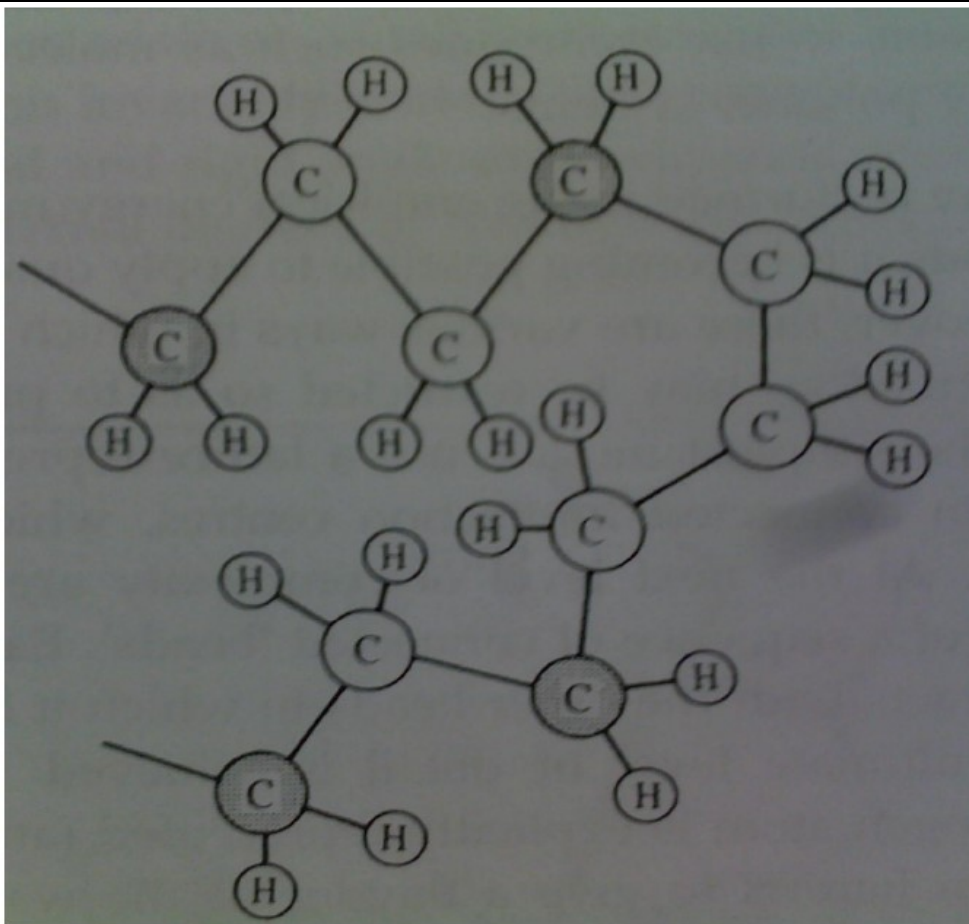
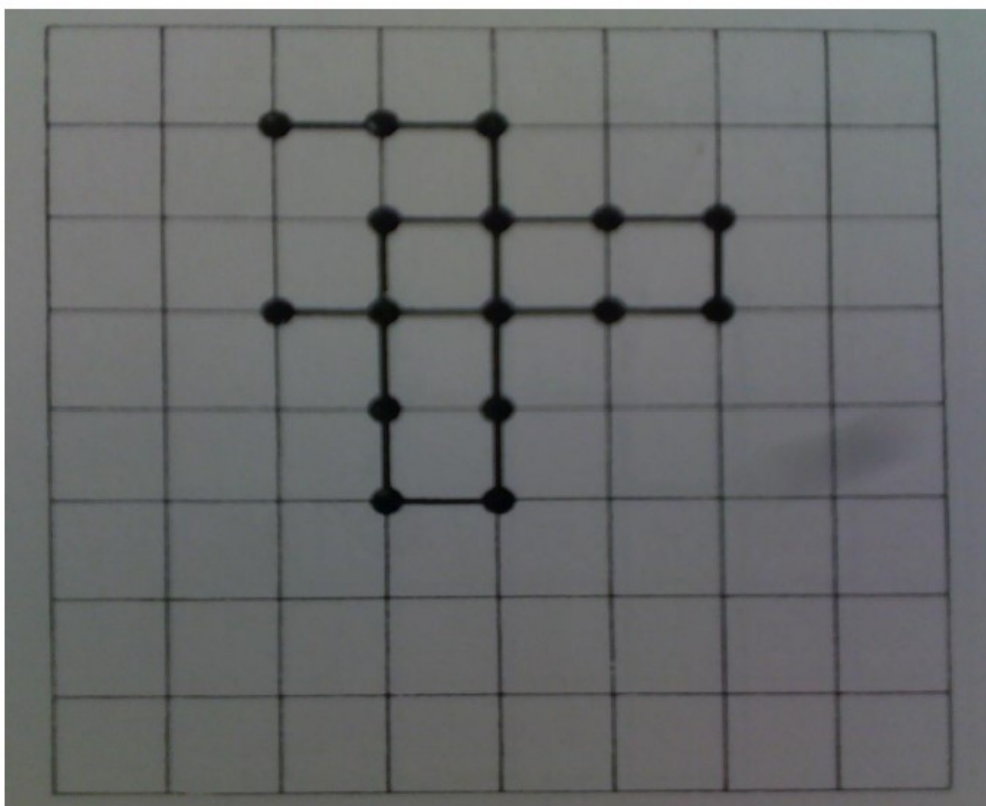


Figure 8.9

In a random walk on a square lattice the chain can cross itself.

في المشي العشوائي على شبكة مربع من السلسلة
يمكن ان يعبر عن نفسه.



Represent three bonds in the actual molecule [Baschnagel et al. 1991]. In this model each monomer is positioned at the center within the lattice and five different distances are possible for the monomer-monomer bond lengths.

Lattices can be used to study a wide variety of polymeric systems, from single polymer chains to dense mixtures. The simplest type of simulation in a 'random walk', in which to chain is randomly grown in the lattice until it contains the desired number of bonds (Figure 8.9), In this model the chain is free to cross itself (i.e. excluded volume effects are ignored). Various properties can be calculated from such simulations, by averaging the results over a large number of trials. For example

تمثل ثلاثة روابط في الجزيئة الفعلية
[Baschnagel et al. 1991] . في هذا
النموذج وضع كل مونومر في مركز داخل الشبكة و
هناك خمس مسافات مختلفة ممكنة للمسافة بين رابط
مونومار - مونومار.
ويمكن استخدام الشبكات لدراسة طائفة واسعة من
أنظمة البوليمارية ، من سلاسل
بوليمر (polymer) واحدة لمخاليط كثيفة. أبسط
نوع من المحاكاة في "المشي العشوائي" ، حيث
السلسلة مزروعة بشكل عشوائي في الشبكة حتى
يحتوي على العدد المرغوب فيه من الروابط (الصورة

measure of the size of a polymer in the mean square end-to-end distance, (R_n^2) is related to the number of bonds (n) and the length of each bond (l) by:

(8.9) ، وفي هذا النموذج للسلسلة انما حرة في تزواج نفسها (أي يتم تجاهل آثار الحجم و تستبعد). ويمكن حساب خصائص مختلفة من المحاكاة هذه، عن طريق حساب متوسط النتائج من خلال عدد كبير من المحاكيات. مثلاً لقياس حجم البوليمر في مربع وسطي لمسافة من الطرف إلى الطرف، (R_n^2) هي متعلقة بعدد من السندات (n) وطول كل سند (l) من خلال :

$$(R_n^2) = nl^2$$

The radius of gyration is another commonly calculated property; this is the root mean square distance of each atom (or monomer) from the center of mass. For the random walk model the radius of gyration (s^2) is given in the asymptotic limit by:

نصف قطر الدوران هو خاصية أخرى تحسب عادة ، وهذا هو جذر لمربع وسطي المسافة من كل ذرة (أو مونومر) من مركز الكتلة اي من الوزن الاجمالي. لنموذج المشي العشوائي يُعطى نصف قطر الدوران (s^2) في حد المقارب (asymptotic limit) من خلال :

$$(s^2) = (R_n^2)/6$$

The ability of the chain to cross itself in the random walk may seem to be a serious limitation, but it is found to be valid under some circumstances. When excluded volume effects are not important (also known as 'theta' conditions) then a subscript '0' is often added to properties

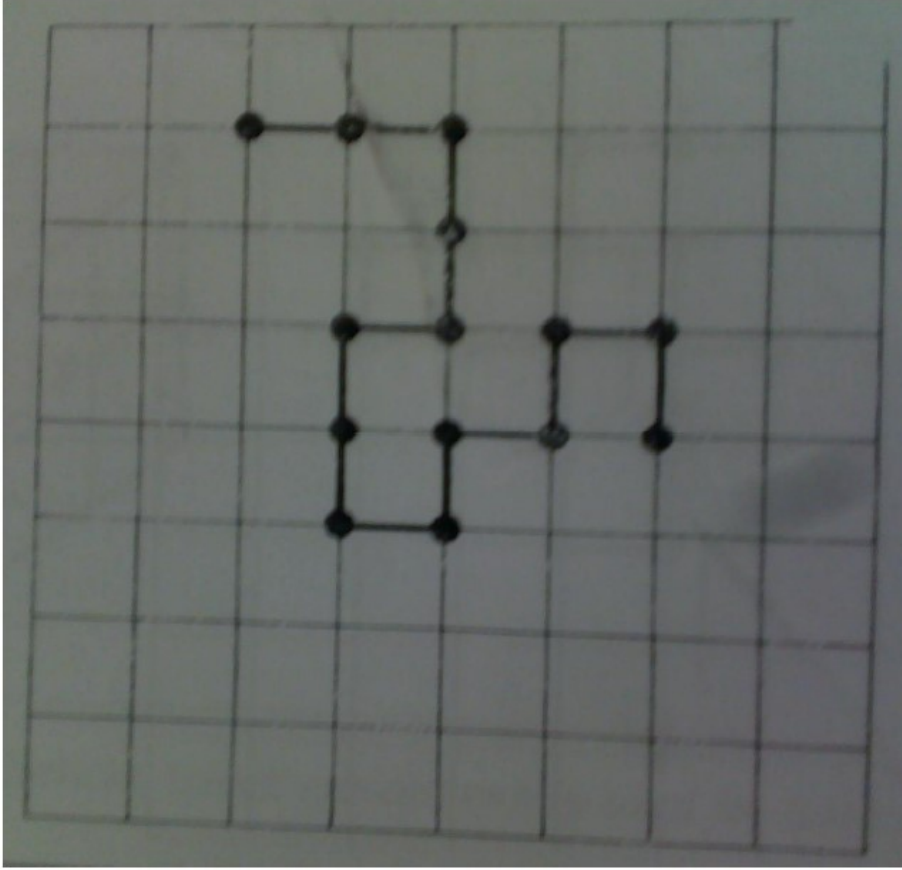
يبدو وجود قيود خطيرة لقدرة السلسلة لتزواج نفسها في المشي العشوائي ، ولكن تكون صالحة في بعض الظروف. عندما تكون آثار الحجم المستبعدة ليست مهمة (المعروف أيضا بظروف 'ثيتا' 'theta')

such as the mean square end-to-end distance, $\langle R_n^2 \rangle_0$. Excluded volume effects can be taken into account by generating a 'self-avoiding walk' of the chain in the lattice (Figure 8.10). In this model only one monomer can occupy each lattice site. Self-avoiding walks have been used to exhaustively enumerate all possible conformations for a chain of a given length one the lattice. If all states are known then the partition function can be determined and thermodynamic quantities calculated. The 'energy' of each state may be calculated using an appropriate interaction model. For example, the energy may be proportional to the number of adjacent pairs of occupied lattice sites. A variation on this is to use polymers

conditions)) إذاً الاشتراك '0' غالباً ما يضاف إلى الخصائص مثل مربع وسطي لمسافة من الطرف إلى الطرف $\langle R_n^2 \rangle_0$. الآثار التي يمكن اتخاذها في الاعتبار حجم مستثنى عن طريق توليد الذاتية تجنب المشي 'من سلسلة شعيرية في (الشكل 8.10). في هذا النموذج واحد فقط يمكن مونومر تشغل كل موقع شبكي. تجنب المشي الذاتي قد استخدمت في تعداد شاملة جميع التشكلات الممكنة لسلسلة من طول معين واحد شعيرية. إذا كنت تعرف جميع الدول ومن ثم يمكن تحديد وظيفة التقسيم وتحسب الكميات الحرارية. قد يحسب 'الطاقة' كل دولة باستخدام نموذج التفاعل المناسب. على سبيل المثال ، قد الطاقة أن يكون متناسبا مع عدد من أزواج المتاخمة لمواقع شعيرية المحتملة. د الاختلاف على هذا هو استخدام البوليمرات

Figure 8.10
Self-avoiding walk: only one monomer can occupy each lattice site

تجنب السير الذاتي : مونومر واحد فقط يمكن أن يشغل كل موقع شبكي



Consisting of two types of monomer (A and B), which have up to three different energy values: A-A, B-B and A-B. Again, the energy is determined by counting the number of occupied adjacent lattice sites. The relationship between the mean square end-to-end distance and the length of the chain (n) has been investigated intensively; with the self-avoiding walk the result obtained is different from the random

تتكون من نوعين من المونومر (A و B)، والتي قد تصل إلى ثلاث قيم مختلفة للطاقة A-A, B-B و A-B. مرة أخرى ، يتم تحديد الطاقة عن طريق حساب عدد المواقع المحتلة في الشبكة المجاورة. وقد تم بشكل مكثف دراسة العلاقة بين المسافة في مربع وسطي من الطرف إلى الطرف طول السلسلة (n) ، مع تجنب المشي الذاتي النتيجة التي حصلت عليها تختلف عن المشي العشوائي ، مع (R_n^2) كونه نسبي إلى $n^{1.18}$ في حد المقارب.

walk, with $\langle R_n^2 \rangle$ being proportional to $n^{1.18}$ in the asymptotic limit.

Having grown a polymer onto the lattice, we now have to consider the generation of alternative configurations. Motion of the entire polymer chain or large-scale conformational changes is often difficult, especially for densely packed polymers. In variants of the verdier-Stockmayer algorithm [Verdier and Stockmayer 1962] new configurations are generated using combinations of 'crankshaft'; 'kink jump' and 'end rotation' moves (figure 8.11). Another widely used algorithm in Monte Carlo simulation of polymers (not just in lattice models) is the 'slithering snake' model. Motion of the entire polymer chain is very difficult, especially for densely packed polymers, and one way in which the polymer can move is by wriggling around obstacles, a process known as reptation. To implement a slithering snake algorithm, one end of the polymer chain is randomly chosen as the 'head' and an attempt is made to grow a new bead at one of the available adjacent lattice positions. Each of the remaining beads is then advanced to that of its predecessor in the chain illustrated in figure 8.12. The procedure is then repeated. Even if it is impossible to move the chosen 'head' the configuration must still be included when ensemble averages are calculated.

بعد أن نمت بوليمر على الشبكة ، الآن لناخذ بعين الاعتبار في توليد تكوينات (توزيع الكتروني) بديلة. من الصعب في كثير من الأحيان تغيير في سلسلة البوليمر لأنه متعلق بتكوين جزئي أو تغييرات واسعة النطاق ، وخاصة للبوليمرات الثقيلة. في المتغيرات من خوارزمية فريدير - [Stockmayer 1962] Stockmayer يتم إنشاء تكوينات جديدة باستخدام مزيج من حركات 'العمود المرفقي' ؛ 'القفز الشبكي' و 'نهاية التناوب' الصورة (8.11). آخر الخوارزمية المستخدمة على نطاق واسع في محاكاة مونت كارلو للبوليمرات (وليس فقط في نماذج شبكية) هو نموذج "انزلاق الثعبان". حركة سلسلة كامل من البوليمر هو أمر صعب جدا ، وخاصة للبوليمرات الثقيلة جداً ، وإحدى الطرق التي يمكن فيها ان تتحرك البوليمر هي ان تتلوى حول العقبات ، هذه العملية معروفة بسمعة جيدة. لتنفيذ خوارزمية انزلاق الثعبان ، يتم اختيارها عشوائيا واحدة من نهاية سلسلة البوليمر لتسمى باسم 'رأس' ، وبذلت محاولة لرعاية خرزة جديدة في واحد من مواقع متوفرة في الشبكة المجاورة. ثم كل من الخرز المتبقية يتم تقديمها من سابقتها في السلسلة الموضحة في الصورة 8.12. ثم يتم تكرار هذا الإجراء. حتى لو كان من المستحيل نقل 'رئيس' التوزيع الالكتروني الذي تم اختياره يجب دائماً ان يكون ضمن حساب معدلات المجموعة.

Figure 8.11

The 'crankshaft', 'kink jump' and 'end

"ناقل الحركة" ، 'شبكة القفز' و "نهاية الدوران"

rotation' moves used in Monte Carlo simulations of polymers " هي التحركات المستخدمة في محاكاة مونت كارلو للبوليمرات

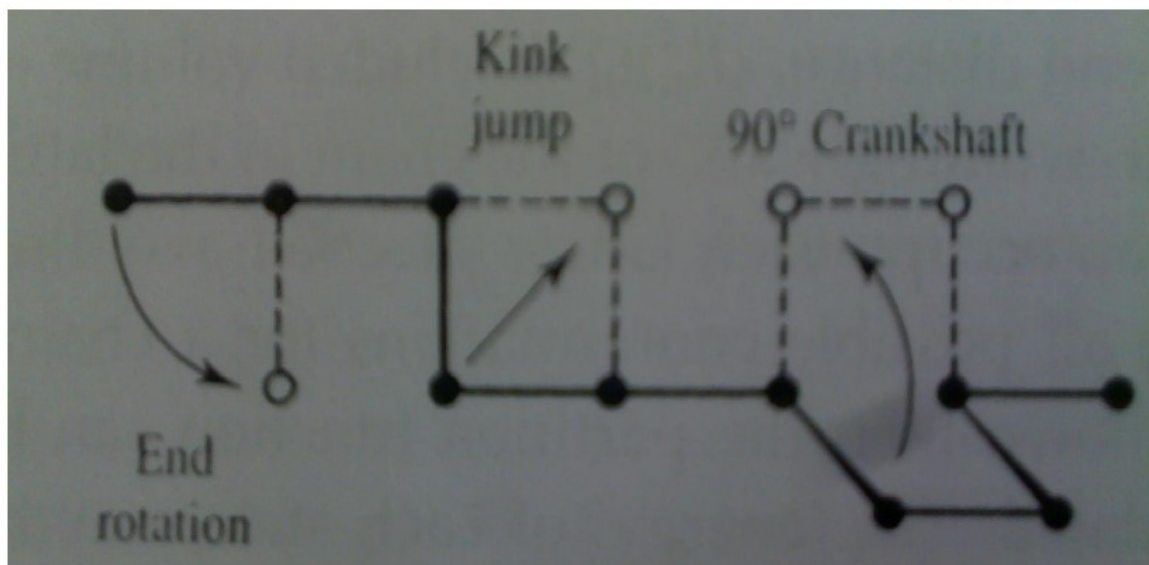
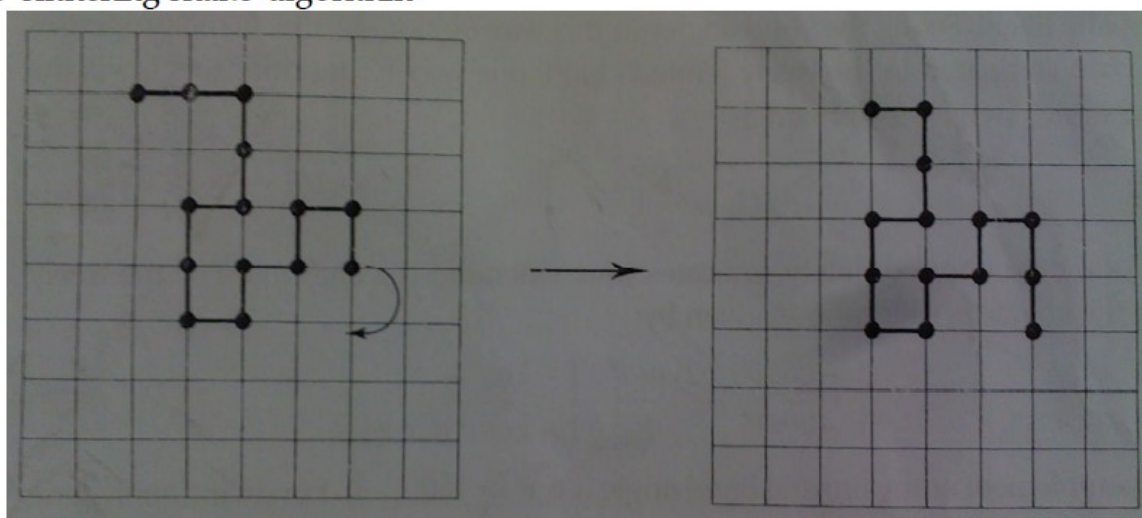


Figure 8.12 The 'slithering snake' algorithm

الخوارزمية 'انزلاق الأفعى'



3.6.2 'Continuous' Polymer Models/

The simplest of the continuous polymer models consists of a string of connected beads (Figure 8.13). The beads are freely jointed and interact with the other beads via a spherically symmetric potential such as the Lennard-Jones potential. The beads should not be thought of as being identical to the

النماذج الأبسط من البوليمر المستمر يتكون من سلسلة متصلة من الحُرز (الشكل 8.13). الحُرز هي المتصلة بحرية وتتفاعل مع الحبات الأخرى عبر احتمالية متماثلة كروية مثل احتمالية لينارد جونز. لا ينبغي أن تكون

monomers in the polymer; though they are often referred to as such ('effective monomers' is a more appropriate term). Similarly, the links between the beads should not be thought of as bonds. The links may be modeled as rods of a fixed and invariant length or may be permitted to vary using a harmonic potential function.

In Monte Carlo studies with this freely jointed chain model the beads can sample from a continuum of positions. The pivot algorithm is one way that new configurations can be generated. Here, a segment of the polymer is randomly selected and rotated by a random amount, as illustrated in figure 8.13. For isolated polymer chains the pivot algorithm can give a good sampling of the configurationally/conformational space. However, for polymers in solution or in the melt, the proportion of accepted moves is often very small due to high-energy steric interactions.

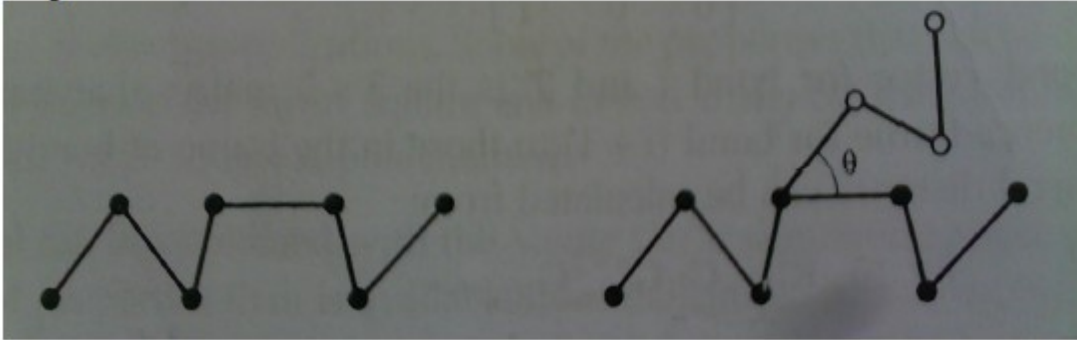
تؤخذ الحيات بأنها مطابقة للمونومرات في البوليمر، على الرغم من أنها غالباً ما يشار إلى هذا النحو ('مونومرات فعالة' هو المصطلح الأكثر ملاءمة). وبالمثل، لا ينبغي أن تعتبر ادوات الربط بين الحيات روابط. قد تكون على غرار الروابط والقضبان من طول ثابت وغير متغير أو يمكن أن يسمح بتغييره باستخدام وظيفة الامكانية المنسجمة.

في دراسات مونت كارلو مع نموذج السلاسل المتصلة بحرية يمكن أخذ عينة من كمية متصلة من المواضع. خوارزمية المحور هي طريقة تمكن من توليد أشكال جديدة. هنا جزء من بوليمار تم اختياره بشكل عشوائي و دار على المحور بمعدل عشوائي، كما هو موضح في الشكل 8.13. لسلاسل البوليمار المعزولة من خوارزمية المحور تستطيع اعطاء أخذ عينات أفضل من الفضاء شكل/تكوين . على كل حال، بالنسبة للبوليمار في الذائب او في اللين، النسبة من الحركات المقبولة هي كثيراً ما تكون ضعيفة جداً ناشئاً عن تفاعلات طاقة عالية

Figure 8.13

The bead model for polymer simulations. The beads may be connected by stiff rods or by harmonic springs

نموذج الحزرة لمحاكاة البوليمر. قد تكون الحزرة متصلة بواسطة قضبان قاسية أو زنبركات متناسقة



The most unrealistic feature of the freely jointed chain model is the assumption that bond angles can vary continuously. In the freely rotating chain model the bond angles are held fixed but free rotation is possible about the bonds, such that any torsion angle value between 0° and 360° is equally likely. Fixing the bond angles in this way obviously affects the properties of the chain when compared to the freely jointed chain; one way quantify this is via the characteristic ratio C_n , which is defined as:

الميزة الأكثر واقعية لنموذج السلسلة المتصلة بحرية هو الافتراض بأن زوايا الروابط يمكن أن تختلف باستمرار. في نموذج سلسلة التي تدور بحرية وتعقد زوايا الروابط بشكل ثابت ولكن من الممكن الدوران الحر حول الروابط، بحيث من الممكن أيضاً أن تكون أية قيمة لزوايا الالتواء بين 0° و 360° . تحديد زوايا الروابط بهذه الطريقة تؤثر على خصائص السلسلة بالنسبة للسلسلة المترابطة بشكل تلقائي؛ طريقة واحدة للتحديد هي عبر خصائص النسبة C_n ، الذي يعرف بأنه :

$$C_n = \frac{\langle R_n^2 \rangle_0}{nl^2}$$

The characteristic ratio approximately indicates how extended the chain is. For the freely rotating chain the characteristic ratio is given by:

تقريباً تدل الخصائص النسبية كم تطول السلسلة. مميزات نسبة تدوير السلسلة بشكل تلقائي تحدد من خلال:

$$C_n = \frac{1 + \cos \theta'}{1 - \cos \theta'} - \frac{2 \cos \theta'}{n} + \frac{1 + \cos^n \theta'}{(1 - \cos \theta')^2}$$

Where θ' is the supplement of the normal bond angle (i.e. $\theta'=180^\circ-\theta$). For an infinitely long chain the characteristic ratio becomes:

حيث تكون θ' الزاوية المكملة للروابط الطبيعية مثلاً:
 $\theta-180^\circ=\theta'$ (لسلسلة بطول لانهائي تصبح
 خصائص النسبة:

$$C_\infty = \frac{1 + \cos \theta'}{1 - \cos \theta'}$$

بسم الله الرحمن الرحيم

4 Dictionary English-Arabic for Molecular Modeling

<u>English</u>	<u>عربي</u>
----------------	-------------

Atom	ذرة
Absolute	قيمة مطلقة
Angular momentum	زخم زاوي / كمية الحركة الزاوية
Antisymmetry	عدم التناظر

Bohr	نموذج بور
Bond	رابط

Charge	شحنة
Covalent bond	رابط تساهمي
Computational chemistry	المعلوماتية الكيميائية
Coordinate Systems	إحداثيات النظام
Cartesian coordinates	الإحداثيات الديكارتية
computer simulation	المحاكاة الحاسوبية
Cross	تزاوج
Computer-generated models	النماذج التي يوجدها الحاسوب
Configuration (<i>electronic configuration</i>)	توزيع إلكتروني
Combination	توافق

Coefficients	معامل
Charge	شحنة
Counter	عدّاد

Double bond	رابط مزدوج
Determinant	المحدّد
Denominator	المقام
Deviation	انحراف
Dimensional	الابعاد

Energy surface	طاقة السطح
Expression	عبارة جبرية
Expansion	امتداد
Electrostatics	كهروستاتيكا
Exponents	الأس
Eigenvalue	القيمة الذاتية
Eigenvector	المتجه الذاتي

Factor	عامل
Factorisation	تحليل
Function	دالة

Ground State	حالة قاعية أو حالة أرضية
--------------	--------------------------

Internal coordinates	الإحداثيات الداخلية
Indistinguishable	غير متميزة
Integral	تكامل
Index	مؤشر
Interaction	تأثير
Iteration	تكرير

Kinetic Energy	الطاقة الحركية
----------------	----------------

momentum	زخم الحركة أو كمية الحركة
Mechanical models	باستخدام نماذج ميكانيكية
Molecular Graphics	رسومات الجزيئية
Molecular modelling	النمذجة الجزيئية
molecular system	نظام الجزيئية
Model	نموذج

Non-linear	غير خطي
Non-covalent bond	رابط غير تساهمي
Normalization	تنسيب آحادي
Nuclei	النوى
Numerator	البسط

Orthogonal	متعامدة
Orthonormal	متعامد ومستنظم

Particle	جسيم
Potential Energy Surfaces	أسطح الطاقة الكامنة
Pseudo-atoms	شبه ذرة واحدة (ذرة زائفة)
Polymer	مركب كيميائي
Probe molecule	جزيء متوقع
Processor	معالج
Potential energy	طاقة الوضع
Polar coordinates	النظام الإحداثي القطبي
Polynomial	كثيرة الحدود
Projection	إسقاط
Polyelectronic	متعددة الالكترونات
Permutations	التباديل
Phase space	مرحلة التباعد

Quantum mechanics	ميكانيكا الكم
-------------------	---------------

Radius	شعاع
Raster devices	الأجهزة النقطية
Real number	عدد حقيقي

Repulsion	تباعد
Random sampling	عينات عشوائية

Sampling	العَيِّنَات
Structure	بُنْيَة
Simulation	المحاكاة
Sinusoidal	الجيبية
Single bond	رابط مفرد
Spin	السبين أو الغزل أو الطئشة
Square	مكعب
Simplification	تبسيط
Substitution	تبدال
Symmetry	تناظر

Torsion angle	زاوية الإلتواء
Theoretical chemistry	الكيمياء النظرية
Term	حدّ
Thermodynamic	الحرارية

Vector devices	الأجهزة الناقلة
Virtual reality	الواقع الافتراضي
Vector	المتّجه

valence

تكافؤ

Wavefunction

دالة موجية

Protein Sidechain placing



Verein für Gentechnik, Ökologie und Gesundheit (VGÖG) e.V.

<http://www.zgoeg.de>

Optimierung einer Glucose-1-Phosphatase aus *Pantoea agglomerans* und einer Phytase aus *Klebsiella terrigena*

2nd Report (2. Zwischenbericht)

January 2005 – Dezember 2005

- Protein Sidechain Optimization with Lagrangian relaxation
(Proteinseitenkettenoptimierung mit LR)

Stand: 2. Januar 2006

In cooperation with
BFEL, Karlsruhe and the Institute of Simulation of Biological Systems, University of Tübingen

Acknowledge

Zunächst danke ich Gott, dem Herrn der Welten, der der Garant des Erfolgs ist. Möge Er diese Arbeit annehmen und uns im Diesseits rechtleiten und so im Jenseits ins Paradies eintreten lassen und uns vor der Strafe des Feuers bewahren.

Desweiteren danke ich meinen lieben Professor Dr. Oliver Kohlbacher für seine sehr gute Betreuung. Ich habe sehr viel von ihm gelernt.

5 Introduction

5.1 Task to be used for PhD thesis

Es soll ein neuer Algorithmus entwickelt werden, der die Seitenketten-Prediktion von Proteinen vornimmt und der auf ganzzahliger Optimierung mit Lagrangian Multipliers beruht.

5.2 Frühere Arbeiten im Umfeld

5.2.1 SCWRL 3.0

Siehe [Canutescu, Shelenkov & Dunbrack 2003].

5.2.1.1 Rotamer library

SCWRL 3.0 uses a new version of the backbone-dependent rotamer library.

A number of improvements have been made in the Bayesian statistical analysis in the determination of probabilities and average dihedral angles and variances for each rotamer at each value of ϕ and ψ . This new rotamer library is available at <http://dunbrack.fccc.edu/bbdep.html>.

5.2.1.2 Energy Function

The energy function consists of a log-probability term from the backbone-dependent rotamer library and steric terms between the side chains and the backbone. The library term has the form

$$E_{lib}(\chi_i) = -K \log p(\chi_i | R, \phi, \psi) / p(\chi_i = 1 | R, \phi, \psi)$$

where R is the residue type, and K is a constant, currently set to 3.0 based on optimization of the energy function for a 180-protein test set.

5.2.1.3 Input and output

It takes a PDB-formatted file that contains backbone coordinates and outputs a file, also in PDB format, containing backbone and predicted sidechain coordinates.

6 Mathematical methods

6.1 Integer Optimization

Ein Optimierungsproblem hat eine Energiefunktion, welche minimiert werden soll. Es gibt Constraints, die eingehalten werden müssen.

6.2 Relaxation with Lagrangian Multipliers

7 Rotamers and rotamer library

Siehe [Bower et. al. 1997].

SCWRL 3.0 uses a new version of the backbone-dependent rotamer library.

A number of improvements have been made in the Bayesian statistical analysis in the determination of probabilities and average dihedral angles and variances for each rotamer at each value of ϕ and ψ . This new rotamer library is available at <http://dunbrack.fccc.edu/bbdep.html>.

8 Molecular Docking

The following is from [Leach], pp. 661-667:

In molecular docking, we attempt to predict the structure (or structures) of the inter-molecular complex formed between two or more molecules. Docking is widely used to suggest the binding modes of protein inhibitors.

The „docking problem“ is thus concerned with the generation and evaluation of plausible structures of intermolecular complexes.

8.1 From algorithmical standpoint the molecular docking problem can be concerned the same as the sidechain optimization problem (SCP)

The sidechains of the binding sites of the docked molecules are concerned as free and optimized.

9 Erzeugung von Selbst- und Wechselenergien von Rotamer-Zuständen der Residuen eines Proteins: mit Dead-end elimination

Das folgende ist [Looger&Hellinga2001] entnommen.

DEE theorems are powerful tools for the combinatorial optimization of protein side-chain placement in protein design and homology modeling. In order to reach their full potential , the theorems must be extended to handle very hard problems.

The DEE algorithms rely on the pairwise decomposition of an energy function that describes the interaction between the rotamers in the protein.

Each DEE algorithm is a filter that identifies and eliminates rotamers that provably cannot be members of the GMEC.

10 Bioinformatical methods

10.1 BALL

You can download the newest version of BALL from

Im folgenden werden einige BALL-Klassen beschrieben, die in den docking tools benutzt werden:

10.1.1 BoundingBoxProcessor

/usr/local/BALL/include/BALL/STRUCTURE/geometricProperties.h

Geometric property processors

The applicators, processors, and collectors described in this chapter are used to extract geometric properties out of a given molecular object or to extract parts of these objects according to their geometric properties. Using the **BoundingBoxProcessor**, the bounding box of a given molecular object can be calculated. The bounding box is represented by the lowest and highest coordinates occurring in the molecular object, i.e. the bounding box is the smallest rectangular box (with sides parallel to the coordinate axes) that encloses all atoms in the molecular object. The **GeometricCenterProcessor** calculates the geometric center of all atoms contained in the molecular object it is applied to. With the aid of the **FragmentDistanceCollector** it is possible to collect all molecular fragments that are within a given distance from a certain fragment. This is useful to extract the relevant molecular environment (e.g. to examine a binding site).

Bounding box creating processor

This class iterates over all atoms of a given molecular object and determines the lowest and the highest coordinates occurring. It returns two coordinates (`getLower`, `getUpper`) describing the smallest cuboid (whose sides are parallel to the planes defined by the coordinate axes) enclosing all atoms of the molecular object. This processor is useful to determine the extent of a molecular object if you want to define a **ThashGrid** or alike objects. The coordinates returned by `getLower` and `getUpper` are only valid, if the processor has been applied to a molecular object containing atoms.

10.1.2 Grid Box Class

/usr/local/BALL/include/BALL/DATATYPE/hashGrid.h

These boxes represent the buckets of a three-dimensional hash grid. Every such box contains a linear list of the objects that are contained in this box. This list is accessible through a `DataIterator`.

class HashGridBox3

Protein Sidechain placing

Constructor using two vectors and a single spacing. This constructor creates a hash grid at `origin` with spacing `spacing`. The vector `size` has to be relative to `origin` and defines the opposite corner of the grid, thereby setting the size of the grid.

10.2 Die Bibliothek docking_tools

Diese Programme benutzen BALL (siehe [Kohlbacher&Lenhof 2000]) und stammen von Prof. Kohlbacher ([Kohlbacher2005]).

Folgende Files werden von der DEE-Optimierung übernommen:

DEE_complete.C

formats (Ordner) -> dort ist docking.ps (eine Übersichtsgaphik)

structure_generator.C

amber_energy.C

docking_grid.C

greedy_tree.C

~~PDB_checker.C~~

transform.res

basicTree.h

energy.C

hydrogen_add.C

util.h

~~candidate_generator.C~~

energy_flex.C

protein_mapper.C

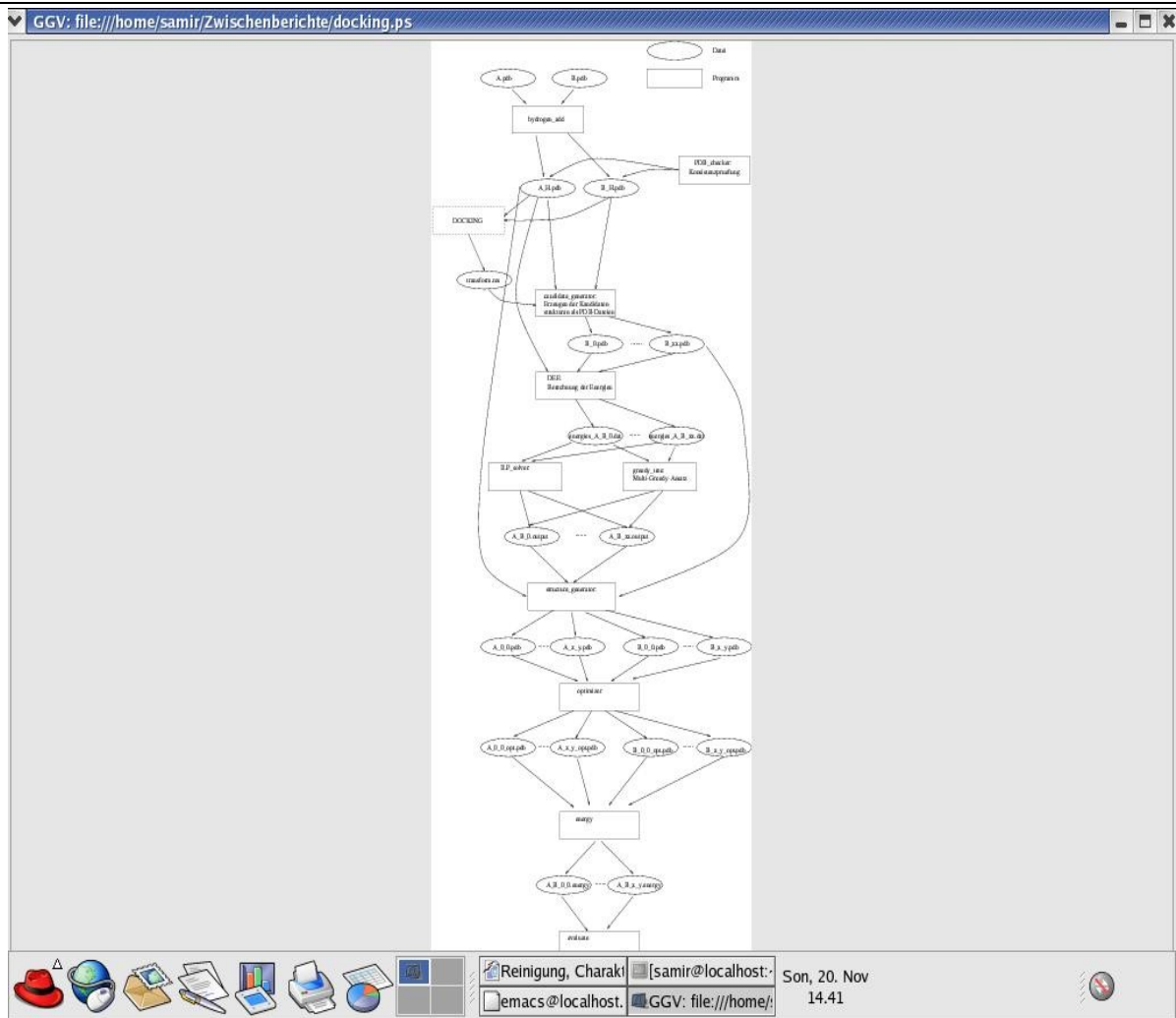
DEE.C

FDPB.C

optimizer.C

selection.h

Docking-Tools



1) **DEE.C und DEE_complete.C**

Berechnung der Eigenenergien (x_{vv}) und der Wechselenergien (x_{uv}) von am Bindungsvorgang beteiligten Sidechains.

Input: pdb-Datei_A pdb-Datei_B

Output: energies_A_B.dat

2) **formats (Ordner) -> dort ist docking.ps (eine Übersichtsgraphik)**

Hier wird die grobe Struktur der Ausgabedateien beschrieben

3) **structure_generator.C**

Erzeugt eine pdb-Datei

Input: Ausgabedatei der Optimierung (noch zu untersuchen)

Output: PDB-Datei

4) `amber_energy.C`

Input:

<pdb file> [<pdb file>]

Function:

calculates the total AMBER energy given for a set of PDB files

1. create structures for the PDB files and the movable residues
2. parse the arguments
3. move the contents of all PDB files to a common system
4. creating fragment DB
5. assign charges
6. checking residues
7. create force field

5) `docking_grid.C`

Function:

- creates a grid (deutsch: Gitter) large enough to contain any docking complex of A and B
- calculates the potential caused by A on this grid.

Input:

<pdbfileA> <pdbfileB> <grid_file> [<PB-optionfile2>]

Output:

<gridfile> (Gitternetz-Datei)

The format of <gridfile>:

```
<< "%s %s      : name A, name B" << endl
<< "(%f %f %f)  : grid origin" << endl
<< "(%f %f %f)  : grid dimension" << endl
<< "%d %d %d    : number of grid points x, y ,z" << endl
<< "n lines with %f: grid values ordered by x, y, and z" << endl
<< endl;
```

Algorithm:

```
// read any options for the FDPB calculation
```

```

// create a fragment database (used to normalize the atom names)
// read the PDB file A
// read the PDB file B
Log.info() << "normalizing names..." << endl;
Log.info() << "assigning charges on A" << endl;
if (charge_processor.getNumberOfErrors())
{
    Log.error() << "Problems assigning charges: " <<
charge_processor.getNumberOfErrors() << " unknown atoms." << endl;
    Log.info() << "assigning radii on A" << endl;
    if (radius_processor.getNumberOfErrors())
    {
        Log.error() << "Problems assigning radii: " << radius_processor.getNumberOfErrors()
<< " unknown atoms." << endl;
        return 1;
    };
    Log.info() << "extent = " << extent << endl;
    Log.info() << "grid lower = " << lower << " grid upper = " << upper << endl;

    FDPB FDPB_object;

    options[FDPB::Option::SOLVENT_DC] = 78.0;
    options[FDPB::Option::SOLUTE_DC] = 2.0;
    options[FDPB::Option::BOUNDARY] = FDPB::Boundary::DIPOLE;
    options[FDPB::Option::DIELECTRIC_SMOOTHING] =
FDPB::DielectricSmoothing::HARMONIC;
    options[FDPB::Option::CHARGE_DISTRIBUTION] =
FDPB::ChargeDistribution::TRILINEAR;
    options[FDPB::Option::BORDER] = 0.0;
    options.setVector(FDPB::Option::BOUNDING_BOX_LOWER, lower);
    options.setVector(FDPB::Option::BOUNDING_BOX_UPPER, upper);
    options.setDefault(FDPB::Option::SPACING, extent / 128);

```

```
Log.info() << "setting up PB..." << endl;
if (!FDPB_object.setup(systemA, options))
{
    Log.error() << "Fehler in FDPB::setup()" << FDPB_object.getErrorCode() << endl;
    return 2;
}

Log.info() << "solving equations..." << flush;
if (!FDPB_object.solve())
{
    Log.error() << "Fehler in FDPB::solve()" << FDPB_object.getErrorCode() << endl;
    return 1;
}

Log.info() << "writing grid_file..." << flush;
```

6) greedy_tree.C

Function:

1. calculate the optimal rotamers of the binding-site side-chains by a multi-greedy strategy.
2. The optimal rotamers are written to <output file>
3. if max_candidates is not given, a default of 1 is assumed.
4. if max_leaves is not given, a default of 20000 is assumed.

Input:

<energies file> <name of output file> [<max_candidates> [<max_leaves>]]

Output:

<output file>

1. open energies file
2. open output file
3. ignore all comment lines
4. read the PDB filenames
5. read the contact distance

```
6. read the template energy
7. create data structures for the residue names and energies (L98)
8. read the number of residues in the binding site
9. read the number of residues in A
10.create a two-dimensional field to hold the interaction energies:
    vector<vector<float>>      E_rest(number_of_residues);

11.read the residues and their energies
12.for (i = 0; i < number_of_residues_of_A; i++)
13.    read the line
14.    save the residue path
15.    check the residue index
16.    read the number of rotamers
17.    resize E_rest
18.    read the rotamer energies E_rest
19.read the number of residues in B
20....(wie A)
21.read the pairwise energies
22.create a vector for the pairwise energies
23.allocate the correct vector sizes
    vector<vector<vector<vector<float>>>>      E_pw(number_of_residues);

    Size s, t;
    // allocate the correct vector sizes
    for (i = 0; i < number_of_residues; i++)
    {
        // allocate the array
        E_pw[i].resize(E_rest[i].size());

        for (s = 0; s < E_rest[i].size(); s++)
        {
            // allocate the array
            E_pw[i][s].resize(number_of_residues);

            for (j = 0; j < number_of_residues; j++)
            {
                // allocate the array
                E_pw[i][s][j].resize(E_rest[j].size());
            }
        }
    }
```


}

24.read the energies

for

1. read a line from the erngies file
2. verify the indices
3. assign pairwise energy

25.Build the tree with all rotamer conformations whose total energy is less than energy_bound

26.Sort the candidates

27.output the energy values and the conformations of the best candidates

7) **PDB_checker.C**

8) **transform.res**

9) **basicTree.h**

10) **energy.C**

Input:

<pdb file 1> <pdb file 2>

Function:

1. setup logging to print the current time in front of each line
2. check arguments
3. read the proteins A + B and insert them into the system AB
4. setup force field
5. assign charges, types, and radii
6. setup FDPB
7. normalize the names and build the bonds according to the fragment database
8. create an AMBER force field
9. perform the first setup (with assignment of charges, type names, and types) amber.setup(AB)
- 10.do not assign anything afterwards
- 11.assign PARSE charge and radius set
- 12.read the FDPB options
- 13.calculate initial energy contributions of A and B
- 14.calculate the solvent excluded surfaces of A and B
- 15.calculate the complex SES
- 16.calculate the change in the solvation free energy
- 17.calculate the electrostatic interaction energy
- 18.calculate the electrostatic interaction energy
- 19.dump the options for documentation purposes
- 20.calculate the changes in solvation energy
- 21.calculate the average interaction energy
- 22.calculate the total binding free energy

Protein A;

Protein B;

System AB;

FragmentDB frag_db;

RotamerLibrary rot_lib("/KM/comp-bio/BALL-data/rotamers/bbind99.Aug.lib", frag_db);

FDPB fdpb;

AmberFF amber;

AssignChargeProcessor PARSE_charges("/KM/comp-bio/BALL-data/charges/PARSE.crg");

ClearChargeProcessor clear_charges;

AssignRadiusProcessor PARSE_radii("/KM/comp-bio/BALL-data/radii/PARSE.siz");

ClearRadiusProcessor clear_radii;

11) hydrogen_add.C

12) util.h

13) candidate_generator.C

14) energy_flex.C

15) protein_mapper.C

16) DEE.C

17) FDPB.C

Electrostatic Contribution to the Free Energy of Solvation

-> Finite difference Poisson-Boltzmann (FDPB) method

1. read the PDB file into system S (read atoms)
2. normalize the names
3. build the bonds according to the fragment database
4. assign PARSE charge and radius set
5. read the FDPB options
6. perform FDPB calculation
7. FDPB setup CPU time: " << T.getCPUTime() << endl;
8. T.reset();
9. fdpb.solve() "FDPB solve CPU time: " << T.getCPUTime() << endl;
10. dump the options for documentation purposes
11. total energy: " << fdpb.getEnergy() << " kJ/mol" << endl;

18) optimizer.C

19) selection.h

11 Side chain optimization with Lagrangian Multipliers

11.1 Version 1

Entwurf fuer ein Paper, voerlaeufiger Titel: Sidechain placing in Homology Modeling via Lagrangian Relaxation, (zu verwenden fuer die Promotion)

Samir Mourad

July 21, 2005

The following 16 pages are from the latex file 210705GMEC.tex



210705GMEC-LR.dvi

Sidechain placing in Homology Modeling via Lagrangian Relaxation

Samir Mourad¹ and Oliver Kohlbacher²

February 2004/Mrz-Mai 2005

Abstract

We illustrate a new approach to the sidechain placing problem. The approach is based on formulating the problem as an integer linear program and then relaxing in a Lagrangian way a suitable set of constraints.

Key Words: molecular modeling, discrete optimization, Lagrangian Relaxation

1 Introduction

1.1 Sidechain placing in homology modeling

Conformations occurring in proteins can be adequately described by a rather small set of so-called rotamers for each amino acid. These rotamer libraries can be used to reduce the sidechain placement problem to a combinatorial optimization problem: search for the set of rotamers with the minimum energy, i.e., the global minimum energy conformation (GMEC). As the number of rotamer combinations is very high (...), efficient methods are required to identify the GMEC or suboptimal solutions sufficiently close to the GMEC.

1.2 Sidechain conformation optimization

In [3] a combinatorial approach for sidechain conformation optimization in Protein Docking area is introduced. There are introduced two methods. One uses an integer linear program and branch-and-cut algorithm. In [8] the constraints of the integer program of [3] are improved.

In this paper a Lagrangian Relaxation (LR) approach is introduced for sidechain conformation optimization to be used in sidechain placing for homology modeling of proteins. The theory of Lagrangian Optimization is a well established branch in of Combinatorial Optimization and has been used successfully in a large number of applications, in different domains [2]. Recently [5] described an LR approach for Structural Alignment of Large-Size Proteins this was the first time that a similar approach was used for an alignment problem in Computational Molecular Biology. Nowadays, LR is the most successful tool to tackle very large problems. These algorithms are capable of finding near-optimal solutions to instances with millions of variables and thousands of constraints within minutes on a PC.

¹Universitaet Tuebingen, Wilhelm Schickard Institute for Computer Science, Dept. for Simulation of biological Systems, Sand 14, D-72076 Tuebingen and VGOEG, Haid-und-Neu-Str.7, D-76139 Karlsruhe, email: mourad@zgoeg.de

²Universitaet Tuebingen, Wilhelm-Schickard-Institute for Computer Science, Dept. for Simulation of biological Systems, Room C318, Sand 14, D-72076 Tuebingen

1.3 Lagrangian Relaxation

The LR approach is particularly well suited for those cases in which the formulation of a problem consists of two sets of constraints: a set of nice constraints and a set of bad constraints, whose removal makes the resulting problem, called the Lagrangian relaxed problem, easily solvable.

1. The strategy then consists in removing the bad constraints from the formulation and putting them into the objective function, each weighted by some coefficient (Lagrangian Multiplier). The weight for a constraint represents a penalty which is incurred by a solution which does not satisfy that constraint. To any choice of weights corresponds a (relatively easy) problem whose solution yields a bound to the original problem.

2. The core question of LR is then to determine the optimal weights, i.e., the Lagrangian multipliers yielding the best bound. In most cases, the determination of these multipliers is equivalent to solving a suitable LP, which would be too time consuming in practice. On the other hand near-optimal multipliers can be found by a simple iterative procedure called subgradient optimization, in which, at each iteration, the Lagrangian relaxed problem is solved and the multipliers are updated based on the corresponding solution.

3. Besides yielding an upper bound on the optimal solution of the original problem, the Lagrangian multipliers (and the associated costs/profits in the objective function) can be used to drive simple heuristic procedures (in most cases of greedy nature). These procedures typically produce substantially different solutions for different Lagrangian multipliers.

4. Accordingly, if the Lagrangian multipliers are embedded within an iterative procedure to define near-optimal multipliers, namely they are called at each iteration with the current multipliers, the best solution found over all iterations tends to be near-optimal.

1.3.1 Design of a general LR/MIP algorithm

The following introduction to LR is from [6].

Lagrangean Relaxation is a Price Directive decomposition technique, which in the first instance simplifies and reduces the problem in question by relaxing groups of constraints. Lagrangean relaxation has been successfully used in processing many different instances of combinatorial optimisation problems, such as the Travelling salesman Problem. Many combinatorial optimisation problems consist of an easy problem that is complicated by the addition of extra constraints. Applying LR in these problems involves identifying these complicating constraints, and then relaxing them by attaching penalties to the complicating constraints and then absorbing them into the objective function. These penalties are known as the Lagrange multipliers. Due to the relaxation of the complicating constraints, the relaxed problem becomes much easier to solve. The next aim is to find tight upper and lower bounds to the problem by iteratively processing sequence of modified sub-problems. LR involves addressing two important issues; one is a strategic issue and the other a tactical issue. The strategic issue concerns the classification and relaxation of the constraints. The strategic question is of the form What constraints are to be relaxed? The tactical issue deals with the selection of a good technique for updating the Lagrange multipliers. The tactical questions are of the form, How the reduced problem can be solved? or How can we calculate an efficient bound?.

1.3.2 Relaxation of constraints

Before defining the general MIP problem, lets identify the following index sets:

$$\begin{aligned}
 B &= \{1, \dots, |B|\} && \text{Index set for binary variables,} \\
 I &= \{|B| + 1, \dots, |B| + |I|\} && \text{Index set for integer variables,} \\
 C &= \{|B| + |I| + 1, \dots, |B| + |I| + |C|\} && \text{Index set for continues variables,} \\
 N &= B \cup I \cup C && \text{Index set for all variables.}
 \end{aligned}$$

Hence, the general MIP problem can be written as:

P_0 :

$$\begin{aligned}
 \min \sum_{j \in N} c_j x_j \\
 \text{s.t. } \sum_{j \in N} a_{kj} x_j (\geq) d_k, \quad k=1, \dots, m \\
 \sum_{j \in N} b_{lj} x_j (\geq) g_l, \quad l=1, \dots, n \\
 x_j \in R^+ \quad \text{iff } j \in C \\
 x_j \in \{0, 1\} \quad \text{iff } j \in B \\
 x_j \in Z^+ \quad \text{iff } j \in I
 \end{aligned}$$

In the following of this subsection 1.3.2 ...

This initial problem P_0 is known as the master problem. Since this master problem is difficult to solve, we relax a set of constraints, $CO \in [1, m]$, by attaching Lagrange multipliers $\lambda_k \geq 0$. Then, this relaxed group of constraints are appended to the objective function and forms the following Lagrange Lower Bound Problem (LLBP):

$P_{L(\lambda)}$:

$$\begin{aligned}
 \min \sum_{j \in N} x_j (c_j - \sum_{k=1}^m \lambda_k a_{kj}) + \sum_{k=1}^m \lambda_k d_k \\
 \text{s.t. } \sum_{j \in N} b_{lj} x_j (\geq) g_l, \quad l=1, \dots, n \\
 x_j \in R^+ \quad \text{iff } j \in C \\
 x_j \in \{0, 1\} \quad \text{iff } j \in B \\
 x_j \in Z^+ \quad \text{iff } j \in I
 \end{aligned}$$

The Lagrangian multipliers, λ_k , penalise the violation of the corresponding relaxed constraints introduced in the objective function. The selection of which set of constraints to be relaxed is a *strategic issue*.

After decomposing the master problem, we are interested in choosing the appropriate numerical

values for the Lagrange multipliers (tactical issue) for the problem $P_{L(\lambda)}$. In particular, we are interested in finding the values of λ that gives the maximum lower bound.³ The Lagrange lower bound is also known as the Lagrange dual program.

$$\max_{\lambda_k} \left\{ \begin{array}{l} \min \sum_{j \in N} x_j (c_j - \sum_{k=1}^m \lambda_k a_{kj}) + \sum_{k=1}^m \lambda_k d_k \\ s.t. \sum_{j \in N} b_{lj} x_j (\geq) g_l, \quad l = 1, \dots, n \\ x_j \in R^+ \quad \text{iff} \quad j \in C \\ x_j \in \{0, 1\} \quad \text{iff} \quad j \in B \\ x_j \in Z^+ \quad \text{iff} \quad j \in I \end{array} \right\} \quad (P_{Dual})$$

The best value for λ_k is calculated by applying iterative updating techniques to the above system (P_{dual}). There are two well-known techniques that have been widely used: Subgradient Optimization and Multiplier Adjustment.

The estimation of good solution to NP-hard problems by using a non-exact method, like LR, does not depend only on the calculation of good lower bound. It is equally important to calculate good solutions that are feasible and provide upper bounds to the master problem. We thus reduce the duality gap and provide tight bound for the optimal solution. The duality gap is defined as the relative difference between the lower bound and the upper bound. In ideal instances, the Lagrange lower bound is equal to the upper bound. The upper bounds are usually calculated by using a Lagrange heuristic (LH). An instant of a LH algorithm is to take the LLBP solution vector and to attempt to convert it to a feasible solution vector to the master problem.

1.3.3 Determination of the Lagrangian multipliers

There have been two main techniques that have been successfully applied for finding Lagrange multipliers in a wide variety of problem instances. There are the *subgradient optimization* and *multiplier adjustment*. Subgradient optimisation is an iterative procedure that, starting from an initial set of Lagrange Multipliers, attempts to improve the lower bound of the LLBP in a systematic way. Multiplier adjustment is also an iterative procedure, but modifies only one component of the multiplier in an iteration.

The literature suggests that *subgradient optimization* is the preferable method for general discrete optimisation problems. Subgradient is straight forward to implement and can be applied without modifications for different problem instances.

Algorithmic Framework of Subgradient Optimisation

Define C_j as the cost coefficient vector of the LLBP ($P_{L(\lambda)}$). Hence,

$$C_j = c_j - \sum_{k=1}^m \lambda_k a_{kj}$$

³If the P_0 problem is max ..., then we are seeking for the minimum upper bound.

where $j = 1, \dots, n$ (number of coefficients (variables)) and $k = 1, \dots, m$ (number of constraints). The main steps that have to be followed to apply subgradient optimisation are set out below:

STEP1: *initialisation*

- Set π which is a user-defined parameter, equal to 2. ($0 \leq \pi \leq 2$)
- Set the lower bounds to $-\infty$ and the upper bounds UB, Z_{UB} to $+\infty$.
- Set N_LR = 0 number of Lagrange operations.
- Initialise the Lagrange multipliers λ .

STEP2: *calculate lower bound with subgradient method*

- Solve the LLBP($P_{L(\lambda)}$) for the current set of λ_k to obtain the solution vector X_j and the lower bound Z_{LB} . ($Z_{LB}^t = \{x_j\}$)
- If the $Z_{LB} > LB$, set $LB = Z_{LB}$.

STEP3: *calculate upper bound*

- Apply a Lagrange Heuristic to find a feasible upper bound Z_{UB} . If $Z_{UB} < UB$, set $UB = Z_{UB}$.

STEP4: *Update the multipliers*

1. Calculate the Subgradients G_k^t for current solution vector X_j .

$$G_k^t = d_k - \sum_{j \in N} a_{kj} x_j, \quad k = 1, \dots, m$$

If all $G_i \leq 0$ for each ' \geq ' constraint, then Z_{LB} is feasible.

2. Define a scalar step size T.

$$T = \frac{\pi(Z_{UB} - Z_{LB})^2}{\sum_{i=1}^m (G_k^t)}$$

3. Update the Lagrange Multipliers set

$$\lambda_k^{t+1} = \max(0, \lambda_k^t + TG_k^t), \quad k = 1, \dots, m$$

STEP5: *Stopping criteria*

1. $\pi < 0.005$
2. $(UB-LB) = 0.0$

$$3. \sum_{i=1}^m (G_k^t)^2 = 0$$

If stopping rules are not satisfied then go to STEP 2.

The user-defined parameter, controls the step size T. In the case wherein the lower bound did not improve for 30 consecutive iterations, we half this parameter. Generally speaking, the smaller the value of this parameter, the smaller is the oscillation of the resulted lower bound (ZLB). In fact, when the value of the parameter is small, we are trying to improve the lower bound by searching on the "neighbourhood" of the LB.

There are three termination conditions of the algorithm. The algorithm terminates when the user-defined parameter becomes very small (i.e. 0.005), or when the dual gap (UB-LB) is equal to zero, or when the sum of squares of all the subgradients is equal to zero ($\sum_{i=1}^m (G_k^t)^2 = 0$). The last termination implies that all the constraints are perfectly satisfied and therefore all the Slack variables of the model are equal to zero.

2 Sidechain placing in Homology Modeling via Lagrangian Relaxation - ILP formulation from Kingsford et.al.2005

2.1 ILP formulation

If all pairwise energies between rotamers in positions i and j are non-positive, then we can remove all variables x_{uv} with $u \in V_i$ and $E_{uv} = 0$, and modify the equality constraints

$$\sum_{u \in V_j} x_{uv} = x_{vv} \quad \text{for } j = 1, \dots, p \text{ and } v \in V/V_j$$

For each V_j let $N^+(V_j)$ the set union of the V_i for which there exists some $v \in V_i$ and $u \in V_j$ with $E_{uv} > 0$. Let D' be the set of pairs $\{u,v\}$ with $u \in V_j$ such that either $v \in N^+(V_j)$, or $v \notin N^+(V_j)$ but $E_{uv} < 0$. There will be edge variables x_{uv} only for pairs in D' .

Our modified ILP is as follows:

$$\text{Minimize } E' = \sum_{u \in V} E_{uu}x_{uu} + \sum_{\{u,v\} \in D'} E_{uv}x_{uv}$$

subject to

$$\sum_{u \in V_j} x_{uu} = 1 \quad \text{for } j = 1, \dots, p$$

$$\sum_{u \in V_j} x_{uv} = x_{vv} \quad \text{for } j = 1, \dots, p \text{ and } v \in N^+(V_j)$$

$$\sum_{u \in V_j: E_{uv} < 0} x_{uv} \leq x_{vv} \quad \text{for } j = 1, \dots, p \text{ and } v \notin N^+(V_j)$$

An inequality constraint is not included if the sum on the left-hand side is empty.

2.2 Lagrangian Relaxation of Kingsford-Formulation

$$\left. \begin{array}{l} \min_{x_{uu}, x_{uv}} \left\{ \sum_{u \in V} E_{uu} x_{uu} \right. \\ + \sum_{\{u,v\} \in D'} E_{uv} x_{uv} \\ + \sum_{j=1, \dots, p} \sum_{v \in N^+(V_j)} \lambda_{jv} \left(\sum_{u \in V_j} x_{uv} - x_{vv} \right) \\ + \sum_{j=1, \dots, p} \sum_{v \notin N^+(V_j)} \mu_{jv} \left(\sum_{u \in V_j: E_{uv} < 0} x_{uv} - x_{vv} \right) \left. \right\} \\ \text{subject to:} \\ \sum_{u \in V_j} x_{uu} = 1 \quad \text{for } j = 1, \dots, p \\ x_{uu}, x_{uv} \in \{0, 1\} \end{array} \right\} (PDual)$$

2.2.1 Dimension of variables and constraints

State variables x_{uu} and x_{uv} :

There are $|V| = n_1 + \dots + n_p$ variables x_{uu}
and $|V|^2/2$ variables x_{uv} .

Lagrangian multipliers λ_{jv} and μ_{jv} :

There are $p * |V|$ multiplier variables λ_{jv}
and $p * |V|$ multiplier variables μ_{jv} .

Constraints:

There are p constraints
 $\sum_{u \in V_j} x_{uu} = 1 \quad \text{for } j = 1, \dots, p.$

For a normal protein with rotamers from a library like the Dunbrack-Library (see [7]) $p \approx 400$ and $n_i \approx 80$ for $i=1..p$. Thus we have $\approx 5 * 10^8$ state variables,
 $\approx 1,2 * 10^7$ Lagrangian multipliers, and ≈ 400 constraints

2.2.2 Implementation of the Lagrangian Relaxation

We can rewrite the following term in the energy function

$$\sum_{\{u,v\} \in D'} E_{uv} x_{uv}$$

as

$$\sum_{j=1, \dots, p: u \in V_j, v \in N^+(V_j)} E_{uv} x_{uv} + \sum_{j=1, \dots, p: v \notin N^+(V_j), u \in V_j: E_{uv} < 0} E_{uv} x_{uv}$$

and

$$\sum_{u \in V} E_{uu} x_{uu} \text{ as } \sum_{v \in V} E_{vv} x_{vv}.$$

The algorithm consists of the following steps:

1. initialisation of \mathbf{x} , λ and μ .
2. for fix λ and μ the energy function is minimized over x_{uu} and x_{uv} .

That means each x_{uu} and x_{uv} are set (within the uncomplicating constraints) either to 0 or to 1 such that the relaxed energy function is minimized with fixed λ and μ .

$$z_{LR}(\lambda, \mu \geq 0) = \min_{x_{vv}, x_{uv}} \left\{ \begin{array}{l} \sum_{v \in V} E_{vv} x_{vv} - \sum_{j=1, \dots, p: v \in N^+(V_j)} \lambda_{jv} x_{vv} - \sum_{j=1, \dots, p: v \notin N^+(V_j)} \mu_{jv} x_{vv} \\ + \sum_{j=1, \dots, p: v \in N^+(V_j)} \sum_{u \in V_j} E_{uv} x_{uv} \\ + \sum_{j=1, \dots, p: v \notin N^+(V_j), u \in V_j: E_{uv} < 0} E_{uv} x_{uv} \\ + \sum_{j=1, \dots, p: v \in N^+(V_j)} \sum_{u \in V_j} \lambda_{jv} x_{uv} \\ + \sum_{j=1, \dots, p: v \notin N^+(V_j)} \sum_{u \in V_j: E_{uv} < 0} \mu_{jv} x_{uv} \end{array} \right\}$$

subject to:

$$E_{v_{11}} x_{v_{11}} \dots \dots \dots \lambda_{2v_{11}} x_{v_{11}} \text{ oder } \mu_{2v_{11}} x_{v_{11}}$$

$$\sum_{u \in V_j} x_{uu} = 1 \text{ for } j = 1, \dots, p$$

$$x_{uu}, x_{uv} \in \{0, 1\}$$

First z_{LR} is optimized with respect to x_{vv} . There are only concerned the terms in the first line. We can divide the $v \in V$ into p classes $V_j, j = 1 \dots p$. Because we are minimizing we suppose $E_{vv} < 0$.

There are $|V| = n_1 + \dots + n_p$ elements in $\sum_{v \in V} E_{vv} x_{vv}$
and $(p-1) * |V|$ elements in
 $-\sum_{j=1, \dots, p: v \in N^+(V_j)} \lambda_{jv} x_{vv} - \sum_{j=1, \dots, p: v \notin N^+(V_j)} \mu_{jv} x_{vv}$

$$\begin{array}{c}
 \hline
 j = 1 : \\
 E_{v_{11}v_{11}} x_{v_{11}v_{11}} \dots \lambda_{2v_{11}} x_{v_{11}v_{11}} \text{ oder } \mu_{2v_{11}} x_{v_{11}v_{11}} \\
 \cdot \\
 \cdot \\
 \lambda_{pv_{11}} x_{v_{11}v_{11}} \text{ oder } \mu_{pv_{11}} x_{v_{11}v_{11}} \\
 \cdot \\
 \cdot \\
 E_{v_{1n_1}v_{1n_1}} x_{v_{1n_1}v_{1n_1}} \dots \lambda_{2v_{1n_1}} x_{v_{1n_1}v_{1n_1}} \text{ oder } \mu_{2v_{1n_1}} x_{v_{1n_1}v_{1n_1}} \\
 \cdot \\
 \cdot \\
 \lambda_{pv_{1n_1}} x_{v_{1n_1}v_{1n_1}} \text{ oder } \mu_{pv_{1n_1}} x_{v_{1n_1}v_{1n_1}} \\
 \hline
 \cdot \\
 \cdot \\
 \hline
 j = k : \\
 E_{v_{k1}v_{k1}} x_{v_{k1}v_{k1}} \dots \lambda_{1v_{k1}} x_{v_{k1}v_{k1}} \text{ oder } \mu_{1v_{k1}} x_{v_{k1}v_{k1}} \\
 \cdot \\
 (\text{not } \lambda_{kv_{k1}} x_{v_{k1}v_{k1}} \text{ oder } \mu_{kv_{k1}} x_{v_{k1}v_{k1}}) \\
 \cdot \\
 \lambda_{pv_{k1}} x_{v_{k1}v_{k1}} \text{ oder } \mu_{pv_{k1}} x_{v_{k1}v_{k1}} \\
 \cdot \\
 \cdot \\
 E_{v_{kn_k}v_{kn_k}} x_{v_{kn_k}v_{kn_k}} \dots \lambda_{1v_{kn_k}} x_{v_{kn_k}v_{kn_k}} \text{ oder } \mu_{1v_{kn_k}} x_{v_{kn_k}v_{kn_k}} \\
 \cdot \\
 (\text{not } \lambda_{kv_{kn_k}} x_{v_{kn_k}v_{kn_k}} \text{ oder } \mu_{kv_{kn_k}} x_{v_{kn_k}v_{kn_k}}) \\
 \cdot \\
 \lambda_{pv_{kn_k}} x_{v_{kn_k}v_{kn_k}} \text{ oder } \mu_{pv_{kn_k}} x_{v_{kn_k}v_{kn_k}} \\
 \hline
 \cdot \\
 \cdot \\
 \hline
 j = p : \\
 E_{v_{p1}v_{p1}} x_{v_{p1}v_{p1}} \dots \lambda_{1v_{p1}} x_{v_{p1}v_{p1}} \text{ oder } \mu_{1v_{p1}} x_{v_{p1}v_{p1}} \\
 \cdot \\
 \lambda_{(p-1)v_{p1}} x_{v_{p1}v_{p1}} \text{ oder } \mu_{(p-1)v_{p1}} x_{v_{p1}v_{p1}} \\
 \cdot \\
 \cdot \\
 E_{v_{pn_p}v_{pn_p}} x_{v_{pn_p}v_{pn_p}} \dots \lambda_{1v_{pn_p}} x_{v_{pn_p}v_{pn_p}} \text{ oder } \mu_{1v_{pn_p}} x_{v_{pn_p}v_{pn_p}} \\
 \cdot \\
 \lambda_{(p-1)v_{pn_p}} x_{v_{pn_p}v_{pn_p}} \text{ oder } \mu_{(p-1)v_{pn_p}} x_{v_{pn_p}v_{pn_p}} \\
 \hline
 \end{array}$$

Determination of the x_{vv} :

In each class V_j only one x_{vv} is 1, the rest is 0.

```

/* j: residue, n[j]: number of rotamers in residue j */
j, k, l = 0;
j = 1..p :
  for k = 1..n[j] :
    relaxedRotamerIntEnergy[j][k] = 0;
    relaxedRotamerIntEnergy[j][k] = relaxedRotamerIntEnergy[j][k] + Evjk vjk
  for l = 1..p, l ≠ j
    relaxedRotamerIntEnergy[j][k] = relaxedRotamerIntEnergy[j][k] - max(λlvjk, μlvjk)

/* now for all rotamers of all residues the relaxed internal energies are computed.
Now for each residue j the rotamer with the minimal internal energy has to be chosen. */

```

```

for j = 1..p :
  /* Initialisation */
  RotWithMinRelaxIntEnergy[j] = 1;
  MinRelaxIntEnergy[j] = relaxedRotamerIntEnergy[j][1];
  for k = 1..n[j] :
    if relaxedRotamerIntEnergy[j][k] < MinRelaxIntEnergy[j]
      RotWithMinRelaxIntEnergy[j] = k;
      MinRelaxIntEnergy[j] = relaxedRotamerIntEnergy[j][k];
    endif

```

/* now for all variables x_{vv} are set. */

```

for j = 1..p :
  for k = 1..n[j] :
    /* Initialisation */
    xvv[j][k] = 0;

  for j = 1..p :
    for k = 1..n[j] :
      if RotWithMinRelaxIntEnergy[j] == k
        xvv[j][k] = 1;
      endif

```

Determination of the x_{uv} :

$$x_{uv} = \begin{cases} 1, & \text{if } x_{uu} = 1 \text{ and } x_{vv} = 1 \text{ (from determination above)} \\ 0, & \text{otherwise} \end{cases}$$

3. The Lagrangian multipliers are updated

To update the Lagrangian multipliers λ and μ we have to identify the variables in STEP4 of the subgradient algorithm in 1.3.3.

STEP4: *Update the multipliers*

- (a) Calculate the Subgradients $G_{k'}^t$ for current solution vector $X_{j'}$.

$$G_{k'}^t = d_{k'} - \sum_{j' \in N} a_{k'j'} x_{j'}, \quad k' = 1, \dots, m$$

- (b) Define a scalar step size T.

$$T = \frac{\pi(Z_{UB} - Z_{LB})^m}{\sum_{i=1}^m (G_{k'}^t)^2}$$

- (c) Update the Lagrange Multipliers set

$$\lambda_{k'}^{t+1} = \max(0, \lambda_{k'}^t + TG_{k'}^t), \quad k' = 1, \dots, m$$

The variable sets $d_{k'}$ and $a_{k'j'}$ in (a) are from the complicating constraints (see 1.3.2).

We can write our complicated constraints

$$\sum_{u \in V_j} x_{uv} = x_{vv} \quad \text{for } j = 1, \dots, p \text{ and } v \in N^+(V_j)$$

$$\sum_{u \in V_j: E_{uv} < 0} x_{uv} \leq x_{vv} \quad \text{for } j = 1, \dots, p \text{ and } v \notin N^+(V_j)$$

as:

```

for j = 1..p :
  for k = 1..n[j] :
    for l = 1..p, l ≠ j :
      if v ∈ N+(Vj)
        ∑r=1n[l] xuv[l][r][j][k] - xvv[j][k] = 0
      elseif v ∉ N+(Vj) and u ∈ Vj : Euv < 0
        ∑r=1n[l] xuv[l][r][j][k] - xvv[j][k] ≤ 0
    
```

variable substitution:

elements of V:

$j = 1..p$

$k = 1..n[j]$

for each element of V there is now an edge defined to all other elements of V except to those which are in the same class V_j :

$l = 1..p, l \neq j$

$r = 1..n[l]$

$x_{uv}[l][r][j][k] \rightarrow x[l][r][j][k]$

$x_{vv}[j][k] \rightarrow x[p+1][0][j][k]$

Some explanations concerning the variables:

The following variables are defined through the following indices:

$x_{uv} : [l][r][j][k], (j = 1..p; k = 1..n[j]; l = 1..p, l \neq j; r = 1..n[l])$
 $x_{vv} : [j][k], (j = 1..p; k = 1..n[j])$
 m complicating constraints: $[j][k][l], (j = 1..p; k = 1..n[j]; l = 1..p, l \neq j)$
 $a_{j'k'} : [l_1][r_1][j_1][k_1]$ (for j' : index of state variables) and
 $[j_2][k_2][l_2]$ (for: k' : index for the m complicating constraints)

Computation of $a_{j'k'}$:

```

for j1 = 1..p :
  for k1 = 1..n[j1] :
    for l1 = 1..p, l1 ≠ j1 :
      if v ∈ N+(Vj)
        for r1 = 1..n[l1] :
          for j2 = 1..p :
            for k2 = 1..n[j2] :
              for l2 = 1..p, l2 ≠ j2 :
                a[l1][r1][j1][k1][l2][j2][k2] = 1 /* factors of xuv */
                a[p + 1][0][j1][k1][l2][j2][k2] = -1 /* factors of xvv */
              elseif v ∉ N+(Vj) and u ∈ Vj : Euv < 0
                for r1 = 1..n[l1] :
                  for j2 = 1..p :
                    for k2 = 1..n[j2] :
                      for l2 = 1..p, l2 ≠ j2 :
                        a[l1][r1][j1][k1][l2][j2][k2] = 1 /* factors of xuv */
                        a[p + 1][0][j1][k1][l2][j2][k2] = -1 /* factors of xvv */
              else
                for r1 = 1..n[l1] :
                  for j2 = 1..p :
                    for k2 = 1..n[j2] :
                      for l2 = 1..p, l2 ≠ j2 :
                        a[l1][r1][j1][k1][l2][j2][k2] = 0
                        a[p + 1][0][j1][k1][l2][j2][k2] = 0
  
```

We identify $d_{k'} = 0$.

We have to compute the vector $G_{k'}^t$, $k'=1, \dots, m$. m is the number of complicating constraints, which are integrated into the energy function. Thus m is also the number of Lagrangian multipliers.

Computation of the vector $G_{k'}^t, k'=1, \dots, m$:

Calculate the Subgradients $G_{k'}^t$ for current solution vector $X_{j'}$.

$$G_{k'}^t = d_{k'} - \sum_{j' \in N} a_{k'j'} x_{j'}, \quad k' = 1, \dots, m$$

Because $d_{k'} = 0$:

$$G_{k'}^t = - \sum_{j' \in N} a_{k'j'} x_{j'}, \quad k' = 1, \dots, m$$

Because $k' = 1, \dots, m$ is equivalent to *for* $j_2 = 1..p$:

for $k_2 = 1..n[j_2]$:

for $l_2 = 1..p, l_2 \neq j_2$

we write $G_t[l_2][j_2][k_2]$ instead of $G_{k'}^t$.

/ initialization */*

for $j_2 = 1..p$:

for $k_2 = 1..n[j_2]$:

for $l_2 = 1..p, l_2 \neq j_2$:

$G_t[l_2][j_2][k_2] = 0$

/ - $\sum_{j' \in N} a_{k'j'} x_{uv}$ */*

for $j_1 = 1..p$:

for $k_1 = 1..n[j_1]$:

for $l_1 = 1..p, l_1 \neq j_1$:

if $v \in N^+(V_j)$

for $r_1 = 1..n[l_1]$:

for $j_2 = 1..p$:

for $k_2 = 1..n[j_2]$:

for $l_2 = 1..p, l_2 \neq j_2$:

$G_t[l_2][j_2][k_2] = G_t[l_2][j_2][k_2] - a[l_1][r_1][j_1][k_1][l_2][j_2][k_2] * x[l_1][r_1][j_1][k_1]$

elseif $v \notin N^+(V_j)$ and $u \in V_j : E_{uv} < 0$

for $r_1 = 1..n[l_1]$:

for $j_2 = 1..p$:

for $k_2 = 1..n[j_2]$:

for $l_2 = 1..p, l_2 \neq j_2$:

$G_t[l_2][j_2][k_2] = G_t[l_2][j_2][k_2] - a[l_1][r_1][j_1][k_1][l_2][j_2][k_2] * x[l_1][r_1][j_1][k_1]$

/ - $\sum_{j' \in N} a_{k'j'} x_{vv}$ */*

for $j_1 = 1..p$:

for $k_1 = 1..n[j_1]$:

for $j_2 = 1..p$:

for $k_2 = 1..n[j_2]$:

for $l_2 = 1..p, l_2 \neq j_2$:

$G_t[l_2][j_2][k_2] = G_t[l_2][j_2][k_2] - a[p+1][0][j_1][k_1][l_2][j_2][k_2] * x[p+1][0][j_1][k_1]$

15

4) Go to 1. (p.16)

hier ist folgender Fehler: die x_{uv} werden auf 1 gesetzt. Darin besteht die Relaxation. Dafür werden die entsprechenden Constraints als Strafterme in die Minimierungsfunktion eingefügt.

11.2 GMEC Version 2

Der Algorithmus funktioniert folgendermassen:

Das Programm hat folgende Teile:

Aufruf:

```
./gmec sidechainanzahl sequencefile datafile_energies
```

1. Eingabe durch Kommandozeile:

- Anzahl der Sidechains

2. Eingabe durch Datafile:

- Anzahl der Rotamere pro Sidechain
- Selbst-Energien der Rotamere
- wechselseitigen Energien der Rotamere

Folgende Vektoren müssen in entsprechende Datenstrukturen gepackt werden:

lambda,

- x_{vv} , // kommt in Klasse Sidechain<aminosaeure>

```
map<bool,> x_vv
```

```
for
```

```
1..#anzahl_sidechains
```

```
1..#anzahl_der_rotamere_in_sidchain_i
```

```
lese E_vv aus datafile
```

```
setze  $x_{vv} = 0$  // bei Initialisation
```

- x_{uv} , // Wechselwirkung zwischen verschiedenen Rotameren -> eigene Klasse
- complicating Constraints: es gibt soviele, wie es Rotamerzustände x_{vv} insgesamt gibt

complicating constraints:

$$-x_{vv}[i][j] + x_{uv}[..][..][i][j] + \dots + x_{uv}[..][..][i][j] = 0$$

linke_Seite_complicating_constraints[i][j]:

$$-x_{vv}[i][j] + x_{uv}[..][..][i][j] + \dots + x_{uv}[..][..][i][j]$$

beim Update der lambdas wird zur Berechnung des Gradienten d. von einem Schritt zum anderen pro Constraint höchstens eine Variablen geändert (das einzige x_{vv} , wenn es den Wert wechselt von 0 auf 1 oder umgekehrt), d.h. linke_seite_compl_constr[i][j]($x_{vv}[i][j]$) = fester_Wert[i][j] + $x_{vv}[i][j]$

- $G = -(\text{linke_seite_compl_constr}[i][j](x_{vv}[i][j]))$

3. Berechnung fester Werte, die durch die Struktur des Programms vorgegeben sind.

- a: Vorfaktoren der Zustandsvariablen in den complicated constraints.
#a = #Zustandsvariablen(alle x_vv + alle x_uv) * #complicated_constraints

4. Aufbau folgender Datenstrukturen:

- **class Sidechain** ist eine Templateklasse Sidechain<Aminosaeure>
pro Sidechain i: vector<short> x_vv(anzahl_rotamere_des_sidechain_i)
pro Rotamerzustand eines Sidechains: Interaktionen zu allen anderen
- **vector<Sidechain> sidechains_of_protein;** // im Konstruktor Protein(short sidechainanzahl)
wird durch eine Schleife (int i; i<sidechainanzahl+1;i++) der Vektor mit den Datenelementen Sidechain aufgebaut.
Der Sidechain i ist dann **sidechains_of_protein [i]**. Die Klasse Sidechain ist also eine generische Klasse Sidechain<i>. Diese generische Klasse hat folgende Datenelemente:
class Sidechain<i> {
int _anzahl_rotamere= anzahl_rotamere[i]; //anzahl_rotamere[i] wird aus datafile
//eingelesen
- **class InteractionsBetweenSidechains(int anzahlSidechains, Protein protein)**
- **class ComplicatedConstraints**

Folgendes muss für den Algorithmus berechnet werden:

- Aufbau der Datenstrukturen (durch die Konstruktoren und die Eingabedaten (Kommandozeile und Datafile), (Initialisierung der Einzelwerte mit 0)
- Initialisierung von lambda (je eins pro complicating constraint) Anmerkung: in version2 noch nicht unterscheiden zwischen lambda und mu
- **Iteratives** Lösen des inneren Optimierungsproblems (pro Iterationsschritt ist lambda fest):
 - die relaxierte Energiefunktion, die die complicated constraints mit Lambda-Vorfaktoren enthält, wird nach x_vv minimiert (unter dem constraint, dass in einem V_j gilt: $\sum x_{vv} = 1$), d.h. es wird pro Sidechain ein x_vv auf 1 und die restlichen auf 0 gesetzt.
Wie die x_vv gesetzt werden ist von den aktuellen lambdas abhängig.
 - alle x_uv werden aber auf 1 gesetzt (dies ist gerade die Relaxation)
 - Berechnung des Gradienten G für ein update der lambdas: G(x_vv, x_uv, a)
 - update des skalaren Vorfaktors von G für das update der lambdas
 - Update der lambdas
 - Abfragen des Abbruchkriteriums: falls nicht erfüllt, gehe in eine weitere Iteration

Folgende Klassen gibt es:

- ..
- InteractionsBetweenSidechains: diese Klasse erzeugt und und initialisiert die Variablen x_uv und die Wechselenergien E_uv; ein
for (j=1;j < p+1;j++){
for (k=1;k<n[j]+1;k++){
x[p+1][0][j][k] = 0;

```
for (l=1;l<p+1;l++){
  for (r=1;r<n[l]+1;r++){
    E_uv[j][k][l][r]=0;
    x_uv[j][k][l][r]=0;
    x[j][k][l][r]=0;
  }
}
```

11.3 GMEC Version3

Optimierung mit Lagrangian Multipliers compilierbar

11.3.1 GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile

Eingabe:

- Anzahl der Rotamere pro Sidechain
- Selbst-Energien der Rotamere
- wechselseitigen Energien der Rotamere

Datenstrukturen:

- **für die belegten Rotamere x_{vv} und die Selbstenergien E_{vv} :**
class Protein, class Sidechain, class Rotamer
- **für die Wechselbeziehungen x_{uv} und Wechselenergien E_{uv} :**
class InteractionsBetweenSidechains
- **für die ComplicatingConstraints inklusive Lagrangian Multipliers**
class ComplicatingConstraints
- **für den Optimierungsalgorithmus**
main.cc, class InnerOptimization

Folgende Vektoren müssen in entsprechende Datenstrukturen gepackt werden:

lambda,

- x_{vv} , // kommt in Klasse Sidechain<aminosaeure>
map<bool,> x_vv
for
1..#anzahl_sidechains

Protein Sidechain placing

```
1..#anzahl_der_rotamere_in_sidchain_i
    lese E_vv aus datafile
    setze x_vv =0 // bei Initialisation
```

- x_uv, // Wechselwirkung zwischen verschiedenen Rotameren -> eigene Klasse
- complicating Constraints: es gibt sovielen, wie es Rotamerzustände x_vv insgesamt gibt

complicating constraints:

```
-x_vv[i][j] + x_uv[...][i][j] + .... + x_uv[...][i][j] = 0
```

linke_Seite_complicating_constraints[i][j]:

```
-x_vv[i][j] + x_uv[...][i][j] + .... + x_uv[...][i][j]
```

beim Update der lambdas wird zur Berechnung des Gradienten d. von einem Schritt zum anderen pro Constraint höchstens eine Variable geändert (das einzige x_vv, wenn es den Wert wechselt von 0 auf 1 oder umgekehrt), d.h. linke_seite_compl_constr[i][j](x_vv[i][j]) = fester_Wert[i][j] + x_vv[i][j]

- $G = -(\text{linke_seite_compl_constr}[i][j](x_{vv}[i][j]))$
5. Berechnung fester Werte, die durch die Struktur des Programms vorgegeben sind.
- a: Vorfaktoren der Zustandsvariablen in den complicated constraints.
#a = #Zustandsvariablen(alle x_vv + alle x_uv) * #complicated_constraints

6. Aufbau folgender Datenstrukturen:

- **class Sidechain** ist eine Templateklasse Sidechain<Aminosaeure>
pro Sidechain i: vector<short> x_vv(anzahl_rotamere_des_sidechain_i)
pro Rotamerzustand eines Sidechains: Interaktionen zu allen anderen
- **vector<Sidechain> sidechains_of_protein;** // im Konstruktor Protein(short sidechainanzahl) wird durch eine Schleife (int i; i<sidechainanzahl+1;i++) der Vektor mit den Datenelementen Sidechain aufgebaut.
Der Sidechain i ist dann **sidechains_of_protein [i]**. Die Klasse Sidechain ist also eine generische Klasse Sidechain<i>. Diese generische Klasse hat folgende Datenelemente:
class Sidechain<i> {
int _anzahl_rotamere= anzahl_rotamere[i]; //anzahl_rotamere[i] wird aus datafile
//eingelesen
- **class InteractionsBetweenSidechains(int anzahlSidechains, Protein protein)**
- **class ComplicatedConstraints**

Folgendes muss für den Algorithmus berechnet werden:

- Aufbau der Datenstrukturen (durch die Konstruktoren und die Eingabedaten (Kommandozeile und Datafile), (Initialisierung der Einzelwerte mit 0)
- Initialisierung von lambda (je eins pro complicating constraint) Anmerkung: in version2 noch nicht unterscheiden zwischen lambda und mu
- **Iteratives** Lösen des inneren Optimierungsproblems (pro Iterationsschritt ist lambda fest):
 - die relaxierte Energiefunktion, die die complicated constraints mit Lambda-Vorfaktoren enthält, wird nach x_vv minimiert (unter dem constraint, dass in einem V_j gilt: $\sum x_{vv} = 1$), d.h. es wird pro Sidechain ein x_vv auf 1 und die restlichen auf 0 gesetzt.
Wie die x_vv gesetzt werden ist von den aktuellen lambdas abhängig.
 - alle x_uv werden aber auf 1 gesetzt (dies ist gerade die Relaxation)
 - Berechnung des Gradienten G für ein update der lambdas: G(x_vv, x_uv, a)
 - update des skalaren Vorfaktors von G für das update der lambdas
 - Update der lambdas
 - Abfragen des Abbruchkriteriums: falls nicht erfüllt, gehe in eine weitere Iteration

Folgende Klassen gibt es:

- ..
- InteractionsBetweenSidechains: diese Klasse erzeugt und initialisiert die Variablen x_uv und die Wechselenergien E_uv; ein
for (j=1;j < p+1;j++){
for (k=1;k<n[j]+1;k++){
x[p+1][0][j][k] = 0;

Protein Sidechain placing

```
for (l=1;l<p+1;l++){
  for (r=1;r<n[l]+1;r++){
    E_uv[j][k][l][r]=0;
    x_uv[j][k][l][r]=0;
    x[j][k][l][r]=0;
  }
}
```

main.cc:

```
int main(int argc, char* argv[])
{
```

```
//Programmaufruf: ./gmec_LR num_of_sidechains sequencefile datafile_energies lagrange_initfile
```

```
//number of side chains is coming from command line.
```

```
//atoi wandelt char* in int
```

```
int num_of_sidechains = atoi(argv[1]);
```

```
char* sequencefile = argv[2];
```

```
char* datafile_energies = argv[3]; //zunächst kingsford-Format, dann Dunbrackbib
```

```
char* lagrange_initfile = argv[4]; //
```

```
cout <<"Anzahl der Seitenketten: " << num_of_sidechains <<"\n";
```

```
cout <<"Sequenzfile: " << sequencefile <<"\n";
```

```
cout <<"Eigenenergien und Interaktionsenergien: " << datafile_energies <<"\n\n";
```

```
//Aufbau der Struktur des Proteins mit variablen Positionen der Seitenketten
```

```
Protein protein(num_of_sidechains,sequencefile,datafile_energies);
```

```
//jetzt sind die x_vv[i][j] mit 0 initialisiert und die E_vv[i][j] eingelesen
```

```
//Aufbau der Struktur für die Wechselwirkungen
```

Protein Sidechain placing

```
InteractionsBetweenSidechains interactions_between_sidechains(protein);
//jetzt sind die x_uv[i][j][k][l] mit 1 initialisiert und die E_uv[i][j][k][l] mit 0 initialisiert

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

//noch zu tun: hier müssen bereits die E_uv eingelesen werden, um im nächsten Schritt überhaupt die
//Complicating Constraints richtig aufzubauen
// ...
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

//Aufbau der complicating constraints
ComplicatingConstraints complicating_constraints(protein, interactions_between_sidechains);
//jetzt sind die Gleichungen und die Ungleichungen innerhalb der compl. constraints initialisiert.
//Erzeugung der lambdas und die mus und Intitialisierung mit 0.1

/////Here begins the algorithm

////////////////////////////////////

//STEP1: Initialization
////////////////////////////////////

// Initialisation of user-defined parameter pi (for stopping criteria)
int stopping_criteria_pi;
stopping_criteria_pi = 2;

// Initialization of N_LR (number of Lagrange operations)
unsigned int N_LR;

//subgradients (for STEP 4)
vector <float> gradient_eq;
vector <float> gradient_uneq;
```

```

//scalar step size T
float T;
T=2;

// Initialization of Lagrangian Multipliers
// FileIO fileIO(lagrange_initfile);
// fileIO.lagrangianMultipliersInit(complicating_constraints);
//hier muss die Struktur geändert werden: die Initialisierung kann gleichzeitig mit der Erzeugung
vorgenommen werden
//da die Anzahl der eq-Compl.Constr. und der uneq-Compl.Constr. durch die E_uv mitdefiniert ist
//...

#if LAGR_INIT==1
for (int i=0; i<complicating_constraints.compl_eq_constraints.size();i++)
    cout << "lambda[" << i << "]" << complicating_constraints.compl_eq_constraints[i].lambda << "\n";
for (int i=0; i<complicating_constraints.compl_uneq_constraints.size();i++)
    cout << "mu[" << i << "]" << complicating_constraints.compl_uneq_constraints[i].mu << "\n";
#endif

while (stopping_criteria_pi>2){

////////////////////
//STEP2: calculate lower bound with subragient method Z_LB ={x_vv, x_uv}
////////////////////

// Inner optimization. Bei festem Lambdas und Mus werden die x_vv upgedated.
InnerOptimization inner_optimization(protein, complicating_constraints);

```



```
////////////////////////////////////
//STEP3: calculate upper bound with heuristic method
////////////////////////////////////
//this step is ignored

////////////////////////////////////
//STEP4: Update the lagrangian multipliers
////////////////////////////////////

//// 1. Calculate the Subgradients G_t[k] for the current solution vector (x_vv, x_uv)
///// If all G_i <= 0 for each >= constraint, then Z_LB (the actual solution vector) is feasible
if (N_LR != 0){
    for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)
        gradient_eq.push_back(0.0);
    for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)
        gradient_uneq.push_back(0.0);

    for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)
        gradient_eq[i] = -(complicating_constraints.compl_eq_constraints[i].sum_x_uv_relaxed
            - complicating_constraints.compl_eq_constraints[i].x_vv);
    for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)
        gradient_eq[i] = -(complicating_constraints.compl_uneq_constraints[i].sum_x_uv_relaxed
            - complicating_constraints.compl_uneq_constraints[i].x_vv);
}
else {
    for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)
        gradient_eq[i] = -(complicating_constraints.compl_eq_constraints[i].sum_x_uv_relaxed
            - complicating_constraints.compl_eq_constraints[i].x_vv);
    for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)
        gradient_uneq[i] = -(complicating_constraints.compl_uneq_constraints[i].sum_x_uv_relaxed
```

```

        - complicating_constraints.compl_uneq_constraints[i].x_vv);
    }

    /// 2. Define a scalar step size
    T = 1/sqrt(T); //alle zwei Durchläufe wird die Schrittweite halbiert

    /// 3. Update der Lagrangian Multipliers

    for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)
        complicating_constraints.compl_eq_constraints[i].lambda =
            (complicating_constraints.compl_eq_constraints[i].lambda
             + T * gradient_eq[i]) < 0.0 ? 0.0 :
            (complicating_constraints.compl_eq_constraints[i].lambda
             + T * gradient_eq[i]);
    for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)
        complicating_constraints.compl_uneq_constraints[i].mu =
            (complicating_constraints.compl_uneq_constraints[i].mu
             + T * gradient_uneq[i]) < 0.0 ? 0.0 :
            (complicating_constraints.compl_uneq_constraints[i].mu
             + T * gradient_uneq[i]);

    N_LR++;
} //while

#ifdef TEST_INTERACTIONS==1
    cout <<"interactions_between_sidechains.interaction_src_sidechains.size()"
         << interactions_between_sidechains.interaction_src_sidechains.size() <<"\n";
#endif

```

```
return 0;
```

```
}//end main
```

11.4 GMEC Version4_mitBALL

11.4.1 Benutzer Code „docking tools“

See chapter „Methoden“.

11.4.2 Gesamtprogramm:

1. Eingabe: PDB-File
2. Berechnung des Energiefiles (durch DEE_complete_SCP.C)
3. GMEC-LR (Lösen des Optimierungsproblems) gmec_lr_SCP.C, gmec.h
4. Strukturzeugung structure_generator_SCP.C

die docking tools werden um gmec_lr_SCP.C und um gmec_lr.h ergänzt

die Dateien *_SCP optimieren nur eine pdb-Datei und nicht wie bei den docking tools die binding sites zweier Proteine.

11.4.3 Eingabe: PDB-File

11.4.4 Berechnung des Energiefiles

Mit DEE_complete_oneInputProtein.C

Aus der PDB-Datei wird mithilfe der Dead-End-Elimination-Methode, der eine bestimmte Energiefunktion zugrunde liegt, das Energiefile erzeugt.

1. Einlesen des pdb-Files in das BALL-Object Protein

Protein A;

```
PDBFile f;
```

```
f.open(argv[1]);
```

```
f >> A;
```

```
f.close();
```

```
Log.info() << "read " << A.countAtoms() << " atoms in A" << endl;
```

2. Building a bounding box around the protein (returning highest and lowest coordinates). Describing the smallest cuboid (whose sides are parallel to the planes defined by the coordinate axes) enclosing all atoms of the molecular object.

BoundingBoxProcessor box;

Protein Sidechain placing

```
A.apply(box);
```

3. Building a list of all residues in A:

```
list<Residue*> initial_residues_A;
```

```
calculateInitialLists(initial_residues_A, A);
```

4. Instatiating FragmentDB: The resulting data can then be used by add_hydrogens and build_bonds

```
FragmentDB frag_db("/usr/local/BALL/data/fragments/Fragments.db");
```

5. Instatiating the rotamer library:

```
RotamerLibrary rot_lib("rotamers/bbind99.Aug.lib", frag_db);
```

6. remove residues without rotamers from the residues vector

7. create vector of the corresponding rotamer sets

11.4.5 GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile

TODO

11.4.6 Strukturzeugung

structure_generator_SCP.C:

12 Ergebnisse Testsets

Die Laufzeiten der einzelnen Programme (energiefile erzeugung, ...) werden für die Proteine separat gemessen.

13 Improvement of Energy function

13.1 Empirical Force Field Models: Molecular Mechanics

13.1.1 Introduction

- Quantum mechanics view is too complex
- molecular mechanics cannot provide properties that depend upon the electronic distribution in a molecule
- several assumptions: the first assumption is the Born-Oppenheimer approximation:

The screenshot shows a web browser window with the address bar containing the URL <http://hyperphysics.phy-astr.gsu.edu/HBASE/molecule/bornop.html>. The search bar contains the text 'Born-Oppenheimer approximation'. The page content is as follows:

The Born-Oppenheimer Approximation

The Born-Oppenheimer Approximation is the assumption that the electronic motion and the nuclear motion in molecules can be separated. It leads to a molecular wave function in terms of electron positions \vec{r}_i and nuclear positions \vec{R}_j .

$$\Psi_{\text{molecule}}(\vec{r}_i, \vec{R}_j) = \Psi_{\text{electrons}}(\vec{r}_i, \vec{R}_j) \Psi_{\text{nuclei}}(\vec{R}_j)$$

This involves the following assumptions

- The electronic wavefunction depends upon the nuclear positions \vec{R}_j but not upon their velocities, i.e., the nuclear motion is so much slower than electron motion that they can be considered to be fixed.
- The nuclear motion (e.g., rotation, vibration) sees a smeared out potential from the speedy electrons.

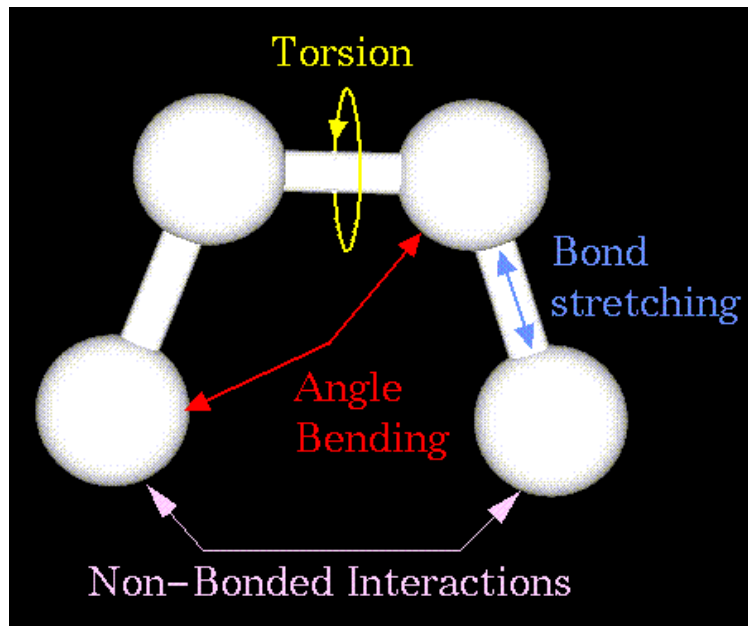
- Molecular Mechanics: simple model of interactions within a system with contributions from processes as
 - stretching of bonds
 - opening/closing of angles
 - rotations about single bonds
- parameters developed from data on small molecules can be used to study much larger molecules such as polymers

13.1.2 A simple Molecular Mechanics Force Field

Protein Sidechain placing

- Energetic penalties are associated with the deviation of bonds and angles away from their „equilibrium“ values.

-

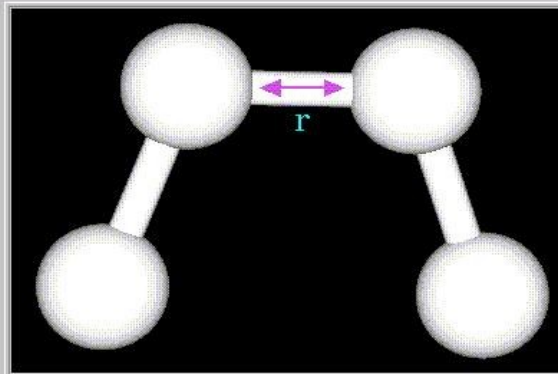


Energy =

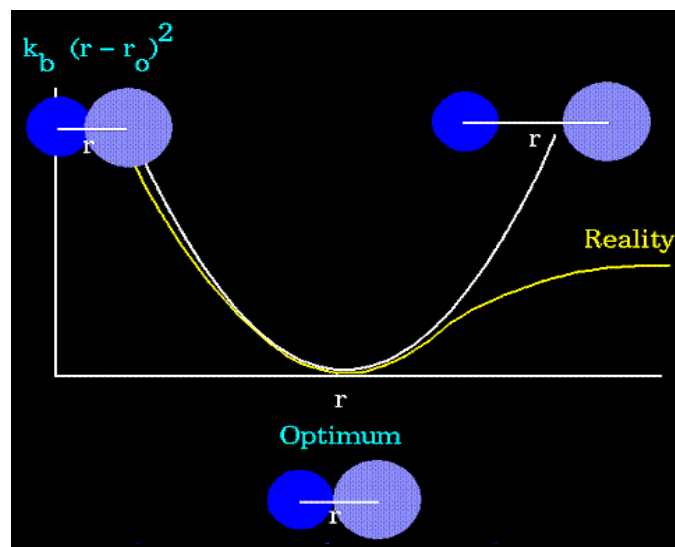
**Stretching Energy +
Bending Energy +
Torsion Energy +
Non-Bonded Interaction Energy**

• **Stretching Energy**

$$E = \sum_{\text{bonds}} k_b (r - r_o)^2$$



The stretching energy equation is based on Hooke's law. The "kb" parameter controls the stiffness of the bond spring, while "ro" defines its equilibrium length. Unique "kb" and "ro" parameters are assigned to each pair of bonded atoms based on their types (e.g. C-C, C-H, O-C, etc.). This equation estimates the energy associated with vibration about the equilibrium bond length. This is the equation of a parabola, as can be seen in the following plot:



- **Torsion Energy**

The torsion energy is modeled by a simple periodic function, as can be seen in the following plot:

The torsion energy in molecular mechanics is primarily used to correct the remaining energy terms rather than to represent a physical process. The torsional energy represents the amount of energy that must be added to or subtracted from the Stretching Energy + Bending Energy + Non-Bonded Interaction Energy terms to make the total energy agree with experiment or rigorous quantum mechanical calculation for a model dihedral angle (ethane, for example might be used a a model for any H-C-C-H bond).

The "A" parameter controls the amplitude of the curve, the n parameter controls its periodicity, and "phi" shifts the entire curve along the rotation angle axis (tau). The parameters are determined from curve fitting. Unique parameters for torsional rotation are assigned to each bonded quartet of atoms based on their types (e.g. C-C-C-C, C-O-C-N, H-C-C-H, etc.). Torsion potentials with three combinations of "A", "n", and "phi" are shown in the following plot:

Notice that "n" reflects the type symmetry in the dihedral angle. A CH₃-CH₃ bond, for example, ought to repeat its energy every 120 degrees. The *cis* conformation of a dihedral angle is assumed to be the zero torsional angle by convention. The parameter phi can be used to synchronize the torsional potential to the initial rotameric state of the molecule whose energy is being computed.

- **Non-Bonded Energy**

The non-bonded energy represents the pair-wise sum of the energies of all possible interacting non-bonded atoms i and j :

The non-bonded energy accounts for repulsion, van der Waals attraction, and electrostatic interactions. van der Waals attraction occurs at short range, and rapidly dies off as the interacting atoms move apart by a few Angstroms. Repulsion occurs when the distance between interacting atoms becomes even slightly less than the sum of their contact radii. Repulsion is modeled by an equation that is designed to rapidly blow up at close distances ($1/r^{12}$ dependency). The energy term that describes attraction/repulsion provides for a smooth transition between these two regimes. These effects are often modeled using a 6-12 equation, as shown in the following plot:

The "A" and "B" parameters control the depth and position (interatomic distance) of the potential energy well for a given pair of non-bonded interacting atoms (e.g. C:C, O:C, O:H, etc.). In effect, "A" determines the degree of "stickiness" of the van der Waals attraction and "B" determines the degree of "hardness" of the atoms (e.g. marshmallow-like, billiard ball-like, etc.).

The "A" parameter can be obtained from atomic polarizability measurements, or it can be calculated quantum mechanically. The "B" parameter is typically derived from crystallographic data so as to reproduce observed average contact distances between different kinds of atoms in crystals of various molecules.

The electrostatic contribution is modeled using a Coulombic potential. The electrostatic energy is a function of the charge on the non-bonded atoms, their interatomic distance, and a molecular dielectric expression that accounts for the attenuation of electrostatic interaction by the environment (e.g. solvent or the molecule itself). Often, the molecular dielectric is set to a constant value between 1.0 and 5.0. A linearly varying distance-dependent dielectric (i.e. $1/r$) is sometimes used to account for the increase in environmental bulk as the separation distance between interacting atoms increases.

Partial atomic charges can be calculated for small molecules using an *ab initio* or semiempirical quantum technique (usually MOPAC or AMPAC). Some programs assign charges using rules or templates, especially for macromolecules. In some force-fields, the torsional potential is calibrated to a particular charge calculation method (rarely made known to the user). Use of a different method can invalidate the force-field consistency.

13.1.3 Potential energy functions

We have briefly reviewed the variety of interactions which are important in protein interactions and seen suitable simple mathematical forms for their representation. These are drawn together to form a potential energy function:

This function can be used to calculate a value for the potential energy (**PEF(R)**) for any conformation of a given protein - defined by the (normally Cartesian) coordinate vector **R**. A number of important points can be made:

- We called the function a "potential" energy function as it does not contain contributions made to the total energy made by the motions of the atoms involved. It is possible to calculate these using molecular dynamics methods.

- The function aims to give reasonable values for the difference in "microstate" energies between two different conformations. The absolute value for the energy given does not mean anything (certainly NOT the free energy of formation). Only differences have meaning. The above equation also does not allow the examination of any process which involves the change in chemical bonding, e.g., one cannot simulate chemical reactions in an enzyme active site with it.
- This kind of function is normally of little use in estimating whether a protein adopts a particular fold - much more useful is the approach set out by Sippl which uses an empirically based approach to identify mis-folds. (See Sippl, M.J. (1990) Calculation of conformational ensembles from potentials of mean force - an approach to the knowledge-based prediction of local structures in globular-proteins, J. Mol. Biol. **213**:859-883).
- To be able to calculate the potential energy of a protein using the above equation involves a large number of parameters (equilibrium bond lengths b_{eq} , bond stretching constants K_b ...). The process of finding these is arduous and in there are only around four potential energy functions in common usage for proteins (CHARMm, AMBER, GROMOS and ECEPP).
- Although results obtained with current potential energy functions are only approximate they have one great advantage - they are computationally cheap. This allows the introduction of realistic representation of environment - such as having large numbers of explicitly modelled water molecules surrounding a protein. It also allows the calculation of the potential energy for many different conformations of the same molecule. This facilitates the use of techniques such as molecular dynamics which allows the thermal motions of a system to be explored. This can be contrasted with quantum chemical methods which even for small systems are so expensive that only a limited number of calculations can be made but produce very accurate energies.

13.1.4 The AMBER force field

From wikipedia.org:

AMBER (an acronym for **Assisted Model Building and Energy Refinement**) is a force field for molecular dynamics originally developed by Peter Kollman's group in the University of California, San Francisco. **AMBER** is also the name for the molecular dynamics simulation package associated with this force field, now coordinated by David A. Case at Scripps Research Institute. A notable use of AMBER is in the distributed computing project Folding@home where it was recently (as of October 15, 2004) in the simulation of protein folding.

The force field takes the form of

Some explanation for the terms in the force field expression:

- First term (summing over bonds): in molecular geometry some forces are computed from chemical bonds between atomic pairs.

- Second term (summing over angles): Forces between atoms in the same molecule may arise even though the atoms are not themselves bonded side-by-side. The calculation of such forces involve the angles of the bonds that join the atoms.
- Third term (summing over torsions): Atoms and groups of atoms may exert forces on each other.
- Fourth term (electrostatic forces): The value r_{ij} is a vector. Also note that the argument r_N on the left-hand side is a vector, which gives the position of each point in the volume containing the molecules. The vector r_{ij} is a displacement vector between two vectors r_i and r_j . Some forces caused by charges arising from ionization and dipoles are fairly weak (due to cancellation by other charges) and are thus reduced by a power of 6 or even 12 on the displacement vector r_{ij} . Charges that are not cancelled cause a force proportional to .

14 The CHARMM force field

CHARMM (**C**hemistry at **H**ARvard **M**acromolecular **M**echanics) is a force field for molecular dynamics as well as the name for the molecular dynamics simulation package associated with this force field. The CHARMM Development Project involves a network of developers in the United States and elsewhere working with Martin Karplus and his group at Harvard to develop and maintain the CHARMM program. Licenses for this software are available, for a free, to people and groups working in academia.

14.1 The force field used by SCWRL 3.0

The energy function consists of a log-probability term from the backbone-dependent rotamer library and steric terms between the side chains and the backbone and between side chains. The library

term has the form

$$E_{lib}(r_i) = -K \log \frac{p(r_i|R, \phi, \psi)}{p(r_i = 1|R, \phi, \psi)} \quad (10)$$

where R is the residue type, and K is a constant, currently set to 3.0 based on optimization of the energy function for a 180-protein test set. The argument of the log is normalized to the highest probability rotamer (defined as $r_i = 1$), so that the energy of this rotamer is zero and all others are positive. The steric energy function between side-chain atoms and the backbone and between side-chain atoms of different residues is essentially the same as in earlier versions of SCWRL (Bower et al. 1997). This function is a very simple linear repulsive energy term, such that

$$\begin{aligned} E(r) &= 0 & r > R_{ij} \\ &= 10 & r < 0.8254R_{ij} \\ &= 57.273 \left(1 - \frac{r}{R_{ij}}\right) & 0.8254R_{ij} \leq r \leq R_{ij} \end{aligned} \quad (11)$$

where r is the interatomic distance, and R_{ij} is the sum of the hard-sphere radii for atoms i and j . C β is treated as a backbone atom. The radii used for atoms are as follows: carbon, 1.6 Å; oxygen, 1.3 Å; nitrogen, 1.3 Å; and sulfur, 1.7 Å. The form of the function is designed to represent the repulsive part of the van der Waals energy in a linear form. It is capped at a value of 10.0 kcal/mole to alleviate the fixed rotamer approximation.

15 Literaturverzeichnis

- 1) [Bower et. al. 1997] Bower, Cohen, Dunbrack, jr., Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: a new homology modeling tool. *J. Mol. Biol.* **1997**, 267, 1268-1282
- 2) [Breitenbach et. al. 1976] Breitenbach M., chromatogr. Synth. Biol. Polym. 2, 271-276 (1976)
- 3) [Brooks, Bruccoleri et.al.1983] Brooks,B.R. et.al., CHARMM: A program for macromolecular energy, minimization, and dynamics calculation. *J. Comput. Chem.* **1983**, 4, 187-217.
- 4) [Bruccoleri et.al.1997] Li,H., Tejero,R., Monleon,D., Bassolino-Klimas,D., Abate-Shen,C., Bruccoleri,R.E., Montelione,G.T., Homology modeling using simulated annealing of restrained molecular dynamics and conformational search calculations with CONGEN: application in predicting the three-dimensional structure of murine homeodomain Msx.-1. *Protein Sci.* **1997**, 6, 956-970
- 5) [Canutescu, Shelenkov & Dunbrack 2003] A. A. Canutescu, A. A. Shelenkov, and R. L. Dunbrack, Jr. A graph theory algorithm for protein side-chain prediction. *Protein Science* **12**, 2001-2014 (2003).
- 6) [Dunbrack] Roland L. Dunbrack, Jr.: Homology modeling in Biology and Medicine (Kap.5 aus Legauer, Th. Bioinformatics – From Genomes to Drugs, Vol.I, Wiley-Ch)
- 7) [Kohlbacher2003] docking_tools, Programmibibliothek zur Durchführung von Docking mehrerer Proteine, bekommen von Prof. Kohlbacher im November 2005.
- 8) [Kohlbacher&Lenhof 2000] O. Kohlbacher, H.-P. Lenhof: BALL – Rapid Software Prototyping in Computational Molecular Biology, *Bioinformatics* 2000, 16(9):815-824
Further information see <http://voyager.bioinf.uni-sb.de/OK/BALL/>
- 9) [Leach] Andrew R. Leach, Molecular Modeling – Principles and Applications, 2nd edition, 2001, Prentice Hall
- 10) [Lippman] C++ Primer
- 11) [Looger&Hellinga2001] Loren L. Looger and Homme W. Hellinga, Generalized Dead-end Elimination Algorithms Make Large-scale Protein Side-chain Structure Prediction Tractable: Implications for Protein Design and Structural Genomics, *J. Mol. Biol.* (2001) **307**, 429-445
- 12) [Meyers1998] Scott Meyers, Effectiv C++ Programmieren – 50 Wege zur Verbesserung Ihrer Programme und Entwürfe, 3.Auflage, Addison-Wesley, 1998
- 13) [Nelson et. al. 2001] Nelson, Cox. Lehninger Biochemie. 3. Auflage, Springer-Verlag Berlin Heidelberg New York 2001
- 14) [Rauhut 2001] Reinhard Rauhut. Bioinfomatik, Wiley VCH Weinheim, New York, Chichester, Brisbane, Singapore, Toronto 2001

15.1 ANHANG A: Programmcode für GMEC_LR_SCP

15.2 util.h

```
String getResiduePathName(const Residue& res)
{
    // start with the residue name and ID (e.g. ARG78)
    String name = res.getName();
        name = name.trim();
```

```
        String tmp = res.getID();
        name += tmp.trim();

// now iterate over all parent nodes in the composite tree
const Composite* parent = res.getParent();
while (parent != 0)
{
    // if the parent has an non-empty name
    const AtomContainer* base_frag = dynamic_cast<const AtomContainer*>(parent);
    if (base_frag != 0)
    {
        tmp = base_frag->getName();
        tmp = tmp.trim();

        // add the name in front of the current name
        name = tmp + ":" + name;
    }
    parent = parent->getParent();
}

return name;
}
```

15.3 DEE_complete_SCP.C

```
// DEE_complete_SCP.C
// Version: 09.12.2005
// author: Samir Mourad

#include <algorithm>
#include <BALL/COMMON/logStream.h>
#include <BALL/MOLMEC/AMBER/amber.h>
```


Protein Sidechain placing

```
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/STRUCTURE/fragmentDB.h>
#include <BALL/STRUCTURE/residueChecker.h>
#include <BALL/STRUCTURE/rotamerLibrary.h>
#include <BALL/COMMON/limits.h>
#include <BALL/DATATYPE/hashGrid.h>
#include <BALL/STRUCTURE/geometricProperties.h>

using namespace BALL;
using namespace std;

#include "util.h"

inline void calculateInitialLists
    (list<Residue*>& contacts_A,
     Protein& A)
{
    /*
     * BoundingBoxProcessor box;
     * A.apply(box);
     *
     * HashGrid3<Atom*>  grid(box.getLower() - Vector3(distance),
     *
     * box.getUpper() - box.getLower() + Vector3(distance * 2.0),
     *
     * distance);
     *
     * AtomIterator atom_it = A.beginAtom();
     * for (; +atom_it; ++atom_it)
     * {
     *     grid.insert(atom_it->getPosition(), &(*atom_it));
     * }
     *
     * // now iterate over all residues of B and
```

```
// store the residue in a list if any of its atoms has a contact with A
// residues in A are selected and collected afterwards
contacts_B.clear();

*/

ResidueIterator res_it;

/*
HashGridBox3<Atom*>::BoxIterator box_it;
HashGridBox3<Atom*>::DataIterator data_it;
HashGridBox3<Atom*>* closest_box;
Vector3 atom_position;
float distance_2 = distance * distance;

for (res_it = B.beginResidue(); +res_it; ++res_it)
{
    bool contact = false;
    for (atom_it = res_it->beginAtom(); +atom_it; ++atom_it)
    {
        atom_position = atom_it->getPosition();
        closest_box = grid.getBox(atom_it->getPosition());

        if (closest_box != 0)
            for (box_it = closest_box->beginBox(); +box_it; ++box_it)
            {
                for (data_it = box_it->beginData(); +data_it; ++data_it)
                {
                    if (atom_position.getSquareDistance((*data_it)->getPosition())
<= distance_2)
                    {
                        contact = true;
                        Fragment* frag = (*data_it)->getFragment();
                        if (frag != 0)
                        {
                            frag->select();
                        }
                    }
                }
            }
        }
    }
}
```

```
        }
    }
}

if (contact)
{
    contacts_B.push_back(&(*res_it));
}

*/
// collect selected residues in A
contacts_A.clear();
for (res_it = A.beginResidue(); + res_it; ++res_it)
{
    //    if (res_it->isSelected())
    //{
        contacts_A.push_back(&(*res_it));
    //    }
}

Log.info() << "binding site contains " << contacts_A.size() << " residues of A" << endl;
}

int main(int argc, char** argv)
{
    Log.setPrefix(cout, "[%T] ");
    Log.setPrefix(cerr, "[%T:ERROR] ");
    /*
if (argc < 4)
```

```

    {
        Log.error() << argv[0] << " <pbd file 1> <pdb file 2> <contact distance> [<energy
bound> [<# candidates> [<dist_dep>]]]" << endl;
        Log.error() << " contact distance: distance between any two atoms of the binding site
in Angstrom" << endl;
        Log.error() << " energy bound: in kJ/mol" << endl;
        Log.error() << " dist_dep: use distance dependent electrostatics (true/false)" << endl;
        Log.error() << " output is written to <pdb file 1>_<pdb file 2>_energies_complete.dat"
<< endl;

        return 1;
    }

    */
    Protein A;
    PDBFile f;
    f.open(argv[1]);
    f >> A;
    f.close();
    Log.info() << "read " << A.countAtoms() << " atoms in A" << endl;
    /*
    f.open(argv[2]);
    f >> B;
    f.close();
    Log.info() << "read " << B.countAtoms() << " atoms in B" << endl;
    */
    //      const float distance = atof(argv[3]);
    // const float distance = 100.0;
    //cout << "contact distance = " << distance << " A" << endl;
    float energy_bound = 100.0;      // kJ/mol
    float max_energy = 1000000.0;    // kJ/mol
    Size max_candidates = 4;
    /*
    if (argc > 4)
    {

```

```
        energy_bound = atof(argv[4]);
        cout << "energy bound = " << energy_bound << " kJ/mol" << endl;
    }

    if (argc > 5)
    {
        max_candidates = atoi(argv[5]);
        cout << "max # of candidates = " << max_candidates << endl;
    }
    /*
    if (argc > 6)
    {
        if (!strcmp(argv[6], "true"))
        {
            use_distance_dependent_electrostatics = true;
            Log.info() << "using distance dependent electrostatics" << endl;
        }
    }
    */
    list<Residue*> initial_residues_A;
    //list<Residue*> initial_residues_B;

    calculateInitialLists(initial_residues_A, A);

    // create fragment database and rotamer library
    // (Dunbrack's backbone-independent rotamer library 08/1999)
    Log.info() << "creating fragment and rotamer databases..." << endl;
    FragmentDB frag_db("/usr/local/BALL/data/fragments/Fragments.db");
    RotamerLibrary rot_lib("rotamers/bbind99.Aug.lib", frag_db);

    // remove residues without rotamers from the binding site vectors
```

```

// and create vectors of the corresponding rotamer sets
Log.info() << "setup rotamer sets..." << endl;
vector<Residue*> residues;
vector<ResidueRotamerSet> rotamer_sets;
list<Residue*>::const_iterator list_it;
vector<String> residue_names;
for (list_it = initial_residues_A.begin(); list_it != initial_residues_A.end(); ++list_it)
{
    const Fragment* frag = frag_db.getReferenceFragment(**list_it);
    if (frag != 0)
    {
        if (!frag->hasProperty(Residue::PROPERTY__HAS_SSBOND))
        {
            const ResidueRotamerSet* rotamer_set_pointer =
rot_lib.getRotamerSet(frag->getName());
            if (rotamer_set_pointer != 0)
            {
                ResidueRotamerSet rs(*rotamer_set_pointer);
                if (rs.getNumberOfRotamers() > 0)
                {
                    rotamer_sets.push_back(rs);
                    residues.push_back(*list_it);

                    residue_names.push_back(getResiduePathName(**list_it));

                    Log.info() << "found rotamers for side chain " <<
residue_names.back()
                    << " (index " <<
residues.size() - 1 << ")" << endl;
                }
            }
        }
    }
}

```

```

Log.info() << "Cannot find a rotamer set for residue " <<
(*list_it)->getName() << " of A" << endl;
    }
}
else
{
    Log.info() << "Residue " << (*list_it)->getName() << " ignored - has
sulphur bridge!" << endl;
}
}
else
{
    Log.warn() << "Could not find a reference fragment for " << (*list_it)-
>getName() << endl;
}
}

// remember the number of residues from A
Size number_of_residues_of_A = residues.size();
/*
for (list_it = initial_residues_B.begin(); list_it != initial_residues_B.end(); ++list_it)
{
    const Fragment* frag = frag_db.getReferenceFragment(**list_it);
    if (frag != 0)
    {
        if (!frag->hasProperty(Residue::PROPERTY__HAS_SSBOND))
        {
            const ResidueRotamerSet* rotamer_set_pointer =
rot_lib.getRotamerSet(frag->getName());
            if (rotamer_set_pointer != 0)
            {
                ResidueRotamerSet rs(*rotamer_set_pointer);
                if (rs.getNumberOfRotamers() > 0)
                {

```

```
rotamer_sets.push_back(rs);
residues.push_back(*list_it);

residue_names.push_back(getResiduePathName(**list_it));

Log.info() << "found rotamers for side chain " <<
residue_names.back()
<< " (index " <<
residues.size() - 1 << ")" << endl;
    }
}
else
{
    Log.info() << "Cannot find a rotamer set for residue " <<
(*list_it)->getName() << " of B" << endl;
}
}
else
{
    Log.info() << "Residue " << (*list_it)->getName() << " ignored - has
sulphur bridge!" << endl;
}
}
else
{
    Log.warn() << "Could not find a reference fragment for " << (*list_it)-
>getName() << endl;
}
}

*/

Size number_of_residues = residues.size();

Log.info() << "calculate initial rotamers..." << endl;
```



```
vector<Rotamer> initial_rotamers;
Size i;
for (i = 0; i < residues.size(); i++)
{
    // calculate the initial rotamer of the residues
    Rotamer r = rotamer_sets[i].getRotamer(*residues[i]);
    initial_rotamers.push_back(r);

    // add it to the residue`s rotamer set
    rotamer_sets[i].addRotamer(r);
}

Log.info() << "setup force field..." << endl;

// insert the two proteins into a common system
System AB;
AB.insert(A);
//AB.insert(B);

// check the system for consistency
Log.info() << "checking system integrity..."<< endl;
//     ResidueChecker residue_checker(frag_db);
//     AB.apply(residue_checker);

// normalize the names and assign build the bonds according to
// the fragment database
Log.info() << "normalizing names..." << endl;
AB.apply(frag_db.normalize_names);
AB.apply(frag_db.build_bonds);

// unmark all residues in the binding site
/*
AB.select();
```

```
*/
Size j;
Size t;
/*
for (i = 0; i < number_of_residues; ++i)
{
    residues[i]->deselect();
}
*/

Log.info() << "setting up force field..." << endl;

// create an AMBER force field
AmberFF FF;
FF.options[AmberFF::Option::FILENAME] = "Amber/amber94.ini";
FF.options[AmberFF::Option::ASSIGN_CHARGES] = "true";
FF.options[AmberFF::Option::ASSIGN_TYPENAMES] = "true";
FF.options[AmberFF::Option::ASSIGN_TYPES] = "true";
FF.options[AmberFF::Option::OVERWRITE_CHARGES] = "true";
FF.options[AmberFF::Option::OVERWRITE_TYPENAMES] = "true";
FF.options.setBool(AmberFF::Option::DISTANCE_DEPENDENT_DIELECTRIC,
use_distance_dependent_electrostatics);

// perform the first setup (with assignment of charges, type names, and types)
FF.setup(AB);

// do not assign anything afterwards
FF.options[AmberFF::Option::ASSIGN_CHARGES] = "false";
FF.options[AmberFF::Option::ASSIGN_TYPENAMES] = "false";
FF.options[AmberFF::Option::ASSIGN_TYPES] = "false";

// calculate self-energy of the rest system
Log.info() << "calculating rigid energy contributions..." << endl;
float    energy_rest = FF.updateEnergy();
```

```
Log.info() << "total energy of complex without binding site residues: " << energy_rest
<< " kJ/mol" << endl;
```

```
Log.info() << "calculating single rotamer energy contributions..." << endl;
```

```
Log.info() << "number of residues: " << number_of_residues << endl;
```

```
Size s;
```

```
AB.deselect();
```

```
vector<vector<float>> internal_energy_is(number_of_residues);
```

```
vector<vector<bool>> is_valid(number_of_residues);
```

```
// calculate the internal energy of each rotamer
```

```
for (i = 0; i < number_of_residues; i++)
```

```
{
```

```
residues[i]->select();
```

```
FF.setup(AB);
```

```
ResidueRotamerSet rotamer_set(rotamer_sets[i]);
```

```
internal_energy_is[i].resize(rotamer_set.getNumberOfRotamers());
```

```
is_valid[i].resize(rotamer_set.getNumberOfRotamers());
```

```
for (s = 0; s < rotamer_set.getNumberOfRotamers(); s++)
```

```
{
```

```
rotamer_set.setRotamer(*residues[i], rotamer_set[s]);
```

```
internal_energy_is[i][s] = FF.updateEnergy();
```

```
Log.info() << " E_int(" << i << ", " << s << ") = " << internal_energy_is[i][s] << endl;
```

```
if (internal_energy_is[i][s] > max_energy)
```

```
{
```

```
is_valid[i][s] = false;
```

```
} else {
```

```
is_valid[i][s] = true;
```

```
}
```

```
}
```

```
        residues[i]->deselect();
    }

    // calculate interaction energies of each rotamer of each residue
    // with the rest system

    // create a two-dimensional array to hold the interaction energies
    // and resize it to the required number of residues and rotamers
    vector<vector<float>> energy_rest_i_s(number_of_residues);
    vector<vector<float>> min_i_s(number_of_residues);
    vector<vector<float>> max_i_s(number_of_residues);
    for (i = 0; i < number_of_residues; ++i)
    {
        energy_rest_i_s[i].resize(rotamer_sets[i].getNumberOfRotamers());
        min_i_s[i].resize(rotamer_sets[i].getNumberOfRotamers());
        max_i_s[i].resize(rotamer_sets[i].getNumberOfRotamers());
    }

    Log.info() << "starting the calculation of the rotamer energy contributions " << endl;

    // calculate all interaction energies rest/rotamer i,s
    Residue* current_residue;
    Size total_number_of_rotamers = 0;
    for (i = 0; i < number_of_residues; ++i)
    {
        AB.select();

        // unmark all residues in the binding site
        for (j = 0; j < number_of_residues; ++j)
        {
            residues[j]->deselect();
        }

        // select the current residue only
```

```
current_residue = residues[i];
current_residue->select();
Log.info() << "starting setup" << endl;
FF.setup(AB);
Log.info() << "setup done" << endl;

AB.deselect();
current_residue->select();

// assign the rotamers and determine their
// energies with the template
ResidueRotamerSet rs(rotamer_sets[i]);
Size valid_rotamers = 0;
for (s = 0; s < rs.getNumberOfRotamers(); s++)
{
    rs.setRotamer(*current_residue, rs[s]);

    float unopt_energy = FF.updateEnergy();
    energy_rest_i_s[i][s] = FF.getEnergy();
    if ((unopt_energy < max_energy) && is_valid[i][s])
    {
        if ((energy_rest_i_s[i][s] - internal_energy_is[i][s]) > energy_bound)
        {
            is_valid[i][s] = false;
        }
        else
        {
            valid_rotamers++;
        }
    }
    else
    {
        is_valid[i][s] = false;
    }
}
```

```
    }

    Log.info() << "E_rest[" << i << "]" << s << "] = " << energy_rest_i_s[i][s] << "
kJ/mol";

    Log.info() << " ES: " << FF.getESEnergy() << " VdW: " << FF.getVdWEnergy();
    if (!is_valid[i][s])
    {
        Log.info() << "[invalid]";
    }
    Log.info() << " - angle dev.: ";
    if (rs.getNumberOfTorsions() >= 1)
    {
        Log.info() << rs[s].chi1 - initial_rotamers[i].chi1;
    }
    if (rs.getNumberOfTorsions() >= 2)
    {
        Log.info() << " " << rs[s].chi2 - initial_rotamers[i].chi2;
    }
    if (rs.getNumberOfTorsions() >= 3)
    {
        Log.info() << " " << rs[s].chi3 - initial_rotamers[i].chi3;
    }
    if (rs.getNumberOfTorsions() >= 4)
    {
        Log.info() << " " << rs[s].chi4 - initial_rotamers[i].chi4;
    }
    Log.info() << endl;

}

Log.info() << "residue " << i << " has " << valid_rotamers << "/" << rs.getNumberOfRotamers() << "
rotamers" << endl;;

total_number_of_rotamers += valid_rotamers;
```

```

    if (valid_rotamers == 0)
    {
        // we didn't find any valid rotamer (e.g. backbone clash)
        // so we reactivate the best half of the candidates
        Size half_number = (Size)((rs.getNumberOfRotamers() + 1) / 2.0);
        Log.info() << "adding " << half_number << " minimum rotamers: ";
        for (Size counter = 0; counter < half_number; counter++)
        {
            Index min_index = INVALID_INDEX;
            float min = Limits<float>::max();
            for (Size j = 0; j < rs.getNumberOfRotamers(); ++j)
            {
                if ((energy_rest_i_s[i][j] - internal_energy_is[i][j]) < min)
                    && !is_valid[i][j])
                {
                    min = energy_rest_i_s[i][j] - internal_energy_is[i][j];
                    min_index = j;
                }
            }
            if (min_index != INVALID_INDEX)
            {
                is_valid[i][min_index] = true;
                Log.info() << min_index << " ";
            }
        }
        Log.info() << endl;
    }
}

Log.info() << "total number of rotamers: " << total_number_of_rotamers << endl;

// create data structure for pair interaction energies
vector<vector<vector<vector<float>>>> energy_is_jt;

```

```
energy_is_jt.resize(number_of_residues);
for (i = 0; i < number_of_residues; i++)
{
    energy_is_jt[i].resize(rotamer_sets[i].getNumberOfRotamers());
    for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
    {
        energy_is_jt[i][s].resize(number_of_residues);
        for (j = 0; j < number_of_residues; j++)
        {
            energy_is_jt[i][s][j].resize(rotamer_sets[j].getNumberOfRotamers());
        }
    }
}

// calculate all pair interaction energies
AB.deselect();
for (i = 0; i < number_of_residues - 1; i++)
{
    Log.info() << "pairwise interactions of residue " << i + 1
        << "/" << number_of_residues << endl;
    residues[i]->select();
    for (j = i + 1; j < number_of_residues; j++)
    {
        residues[j]->select();
        FF.setup(AB);
        for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
        {
            if (is_valid[i][s])
            {
                rotamer_sets[i].setRotamer(*residues[i], rotamer_sets[i][s]);
                for (t = 0; t < rotamer_sets[j].getNumberOfRotamers(); t++)
                {
```



```

        if (is_valid[j][t])
        {
            rotamer_sets[j].setRotamer(*residues[j],
rotamer_sets[j][t]);

            float energy = FF.updateEnergy();

            energy -= internal_energy_is[i][s] +
internal_energy_is[j][t];

            energy_is_jt[i][s][j][t] = energy;
            energy_is_jt[j][t][i][s] = energy;
            //Log.info() << " E[" << i << "][" << s << "][" << j <<
"[" << t << "] = " << energy << " kJ/mol"
            // << "(VdW =
" << FF.getVdWEnergy() << " kJ/mol / ES = " << FF.getESEnergy() << " kJ/mol)" << endl;
        }
        else
        {
            energy_is_jt[i][s][j][t] = max_energy;
            energy_is_jt[j][t][i][s] = max_energy;
        }
    }
}
else
{
    // set all rotamer energies to max_energy
    for (t = 0; t < rotamer_sets[j].getNumberOfRotamers(); t++)
    {
        energy_is_jt[i][s][j][t] = max_energy;
        energy_is_jt[j][t][i][s] = max_energy;
    }
}
}

```

```
        }
        residues[j]->deselect();
    }
    residues[i]->deselect();
}

long double number_of_combinations = 1;

Size count;
for (i = 0; i < number_of_residues; i++)
{
    count = 0;
    for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
    {
        if (is_valid[i][s])
        {
            count++;
        }
    }
    cout << "residue " << i << " has " << count << " rotamers" << endl;
    number_of_combinations *= count;
}
Log.info() << "number of possible combinations: " << number_of_combinations << endl;

Log.info() << "starting DEE..." << endl;;

Size removed_in_this_iteration;
do
{
    removed_in_this_iteration = 0;

    for (i = 0; i < number_of_residues; ++i)
```

```

    {
        for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
        {
            float sum_min = energy_rest_i_s[i][s];
            float sum_max = sum_min;

            for (j = 0; j < number_of_residues; j++)
            {
                if (j != i)
                {
                    float min = Limits<float>::max();
                    float max = Limits<float>::min();
                    for (t = 0; t < rotamer_sets[j].getNumberOfRotamers();
t++)
                    {
                        float tmp = energy_is_jt[i][s][j][t];
                        if ((tmp < max_energy) && is_valid[i][s] &&
is_valid[j][t])
                        {
                            if (tmp < min) min = tmp;
                            if (tmp > max) max = tmp;
                        }
                    }
                    // remove rotamers of i in conflict with all other
rotamers of the j-th sidechain

                    //if (min == Limits<float>::max())
                    //{
                    //    is_valid[i][s] = false;
                    //}
                    sum_min += min;
                    sum_max += max;
                }
            }
            min_i_s[i][s] = sum_min;

```

```
        max_i_s[i][s] = sum_max;
    }
}

for (i = 0; i < number_of_residues; ++i)
{
    for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
    {
        if (is_valid[i][s])
        {
            float tmp = min_i_s[i][s];
            for (t = 0; t < rotamer_sets[i].getNumberOfRotamers(); t++)
            {
                if ((t != s) && is_valid[i][t] && (max_i_s[i][t] < tmp))
                {
                    is_valid[i][s] = false;
                    removed_in_this_iteration++;
                    break;
                }
            }
        }
    }
}

Log.info() << "DEE removed " << removed_in_this_iteration << " rotamers" << endl;
} while (removed_in_this_iteration > 0);

Log.info() << endl;

number_of_combinations = 1;
for (i = 0; i < number_of_residues; i++)
{
```

```

        count = 0;
        for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
        {
            if (is_valid[i][s])
            {
                count++;
            }
        }
        cout << "residue " << i << " has " << count << " rotamers" << endl;
        number_of_combinations *= count;
    }
    Log.info() << "number of possible combinations: " << number_of_combinations << endl;

    const float max_LP_energy = 1e15;

    String  outfilename  =  String(argv[1])  /*  +  "_"  +  String(argv[2])*/  +
    "_energies_complete.dat";

    Log.info() << "ALLAH! " << endl;

    ofstream outfile(outfilename.c_str());
    Log.info() << "writing LP input to " << outfilename << endl;

    // write the parameters
    outfile << ";  energy_bound = " << energy_bound << endl;
    outfile << ";  distance_dependent_ES = " << use_distance_dependent_electrostatics <<
endl;

    outfile << ";" << endl;

```

```
// write the filenames of A and B
outfile << argv[1] << endl;
//outfile << argv[2] << endl;

// write the contact distance in Angstrom
//      outfile << distance << endl;

// write the template energy
outfile << energy_rest << endl;

// write number of residues
outfile << number_of_residues_of_A << endl;

// write number of residues of A
//outfile << number_of_residues_of_A << endl;

// write E rest and general information for each residue of A
for (i = 0; i < number_of_residues_of_A; ++i)
{
    String stringA(residue_names[i]);
String stringB("");
String stringC("");
for(String::iterator it= stringA.begin();it != stringA.end();it++)
    stringA.substitute(stringB, stringC);

    //outfile << residue_names[i] << " " << i << " ";
outfile << stringA << " " << i << " ";
    outfile << rotamer_sets[i].getNumberOfRotamers();
    for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); ++s)
    {
        float tmp = energy_rest_i_s[i][s];
        if ((tmp >= max_LP_energy) || !is_valid[i][s])
```

```
        {
            tmp = max_LP_energy;
        }
        outfile << " " << tmp;
    }
    outfile << endl;
}

// write the number of residues of B
//     outfile << number_of_residues - number_of_residues_of_A << endl;

// write E rest and general information for each residue of B
/*
for (i = number_of_residues_of_A; i < number_of_residues; ++i)
{
    outfile << residue_names[i] << " " << i << " ";
    outfile << rotamer_sets[i].getNumberOfRotamers();
    for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); ++s)
    {
        float tmp = energy_rest_i_s[i][s];
        if ((tmp >= max_LP_energy) || !is_valid[i][s])
        {
            tmp = max_LP_energy;
        }
        outfile << " " << tmp;
    }
    outfile << endl;
}
*/
// write pairwise interaction energies
for (i = 0; i < (number_of_residues - 1); i++)
{
    for (j = i + 1; j < number_of_residues; j++)
```

```
        {
            for (s = 0; s < rotamer_sets[i].getNumberOfRotamers(); s++)
            {
                for (t = 0; t < rotamer_sets[j].getNumberOfRotamers(); t++)
                {
                    float tmp = energy_is_jt[i][s][j][t];
                    if (tmp >= max_LP_energy || !is_valid[i][s] || !is_valid[j][t])
                    {
                        tmp = max_LP_energy;
                    }
                    outfile << i << " " << j << " " << s << " " << t << " " << tmp << endl;
                }
            }
        }
    }
}

outfile.close();

Log.info() << "done." << endl;

return 0;
}
```

15.4 gmec_lr.h

```
// gmec_lr.h
// Version: 30.11.2005
// author: Samir Mourad
#ifndef _GMEC_LR_
#define _GMEC_LR_
```



```
#include <vector>
#include <utility>
#include <fstream>
```

```
using namespace std;
```

```
const int MAX_POSSIBLE_SIDECHAINS_IN_PROTEIN = 420;
const int MAX_POSSIBLE_ROTAMERS_IN_SIDECHAIN = 81;
```

```
class ComplEqConstr
{
public:
    ComplEqConstr(){}
    ComplEqConstr(pair<unsigned int,unsigned int>& x_vv_index_sc_rot_input , bool x_vv_input,
        int index_constr_dest_sc_input,
        int sum_x_uv_relaxed_input);
    //Zuweisungsoperator
    ComplEqConstr& operator=(const ComplEqConstr& rhs);
    //Copy-Konstruktor
    ComplEqConstr (const ComplEqConstr& src)
    {
    }
}
```

```
pair <unsigned int,unsigned int> x_vv_index_sc_rot; //Index des x_vv, welches an diesem Complicating
Constraint beteiligt ist
```

```
                //(mehrere haben den gleichen x_vv-Index!!)

bool x_vv;
int index_constr_dest_sc;
int sum_x_uv_relaxed;
float lambda; //Lagrangian multiplier
};

class ComplUneqConstr
{
public:
    ComplUneqConstr(){}
    ComplUneqConstr(pair<unsigned int,unsigned int>& x_vv_index_sc_rot_input , bool x_vv_input,
                    int index_constr_dest_sc_input,
                    int sum_x_uv_relaxed_input);
    //Zuweisungsoperator
    ComplUneqConstr& operator=(const ComplUneqConstr& rhs);

    //Copy-Konstruktor
    ComplUneqConstr (const ComplUneqConstr& src)
    {
    }

    pair <unsigned int,unsigned int> x_vv_index_sc_rot; //Index des x_vv, welches an diesem Complicating
    Constraint beteiligt ist
                //(mehrere haben den gleichen x_vv-Index!!)

bool x_vv;
int index_constr_dest_sc;
int sum_x_uv_relaxed;
float mu; //Lagrangian multiplier
};
```

```
class ComplicatingConstraints
{

private:

public:

    ComplicatingConstraints(vector<vector<bool>>& x_rest,
                            vector<vector<vector<vector<bool>>>& x_pw,
                            vector<vector<float>>& E_rest,
                            vector<vector<vector<vector<float>>>& E_pw,
                            vector<vector<bool>>& N_plus_V);

    //Zuweisungsoperator
    ComplicatingConstraints& operator=(const ComplicatingConstraints& rhs);

    //Copy-Konstruktor
    ComplicatingConstraints (const ComplicatingConstraints& src)
    {
    }

    // die compl. Constraints werden so aufgeschrieben, dass alle Terme links sind
    // und eine Null auf der rechten Seite der Gleichung ist.
    // da die x_uv ständig auf 1 gesetzt bleiben, ist für ein Compl. Constraint nur entscheidend,
    // a) wieviel x_uv in dem Compl.Constr. sind und b) der Index und der Wert des einen x_vv
    vector <ComplEqConstr> compl_eq_constraints;
    vector <ComplUneqConstr> compl_uneq_constraints;
    vector <vector <vector <ComplEqConstr>>> field_compl_eq_constraints;
    vector <vector <vector <ComplUneqConstr>>> field_compl_uneq_constraints;

    void initLagrangianMultipliers();
```

Protein Sidechain placing

```
// hier muss noch eine Funktion zum Einlesen der Energien  
// aus dem vorangegangenen Programmteil (Berechnen der Energien aus der PDB-Datei)  
// void set_Rot_is_element_of_N_plus_V (vector<vector<float>> E_rest);
```

```
};
```

```
class InnerOptimization
```

```
{
```

```
private:
```

```
public:
```

```
InnerOptimization(){}  
void makeInnerOptimization(vector<vector<bool>>& x_rest,
```

```
vector<vector<float>>& E_rest,
```

```
vector<vector<bool>>& N_plus_V,
```

```
ComplicatingConstraints& complicating_constraints);
```

```
};
```

```
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////
```

Protein Sidechain placing

```
ComplEqConstr::ComplEqConstr(pair<unsigned int,unsigned int>& x_vv_index_sc_rot_input , bool  
x_vv_input,
```

```
    int index_constr_dest_sc_input,
```

```
    int sum_x_uv_relaxed_input)
```

```
{  
    x_vv_index_sc_rot = x_vv_index_sc_rot_input;  
    x_vv = x_vv_input;  
    index_constr_dest_sc_input = index_constr_dest_sc;  
    sum_x_uv_relaxed = sum_x_uv_relaxed_input;
```

```
}
```

```
//Zuweisungsoperator
```

```
ComplEqConstr& ComplEqConstr::operator=(const ComplEqConstr& rhs)
```

```
{  
}
```

```
ComplUneqConstr::ComplUneqConstr(pair<unsigned int,unsigned int>& x_vv_index_sc_rot_input , bool  
x_vv_input,
```

```
    int index_constr_dest_sc_input,
```

```
    int sum_x_uv_relaxed_input)
```

```
{  
    x_vv_index_sc_rot = x_vv_index_sc_rot_input;  
    x_vv = x_vv_input;  
    index_constr_dest_sc_input = index_constr_dest_sc;  
    sum_x_uv_relaxed = sum_x_uv_relaxed_input;
```

```
}
```

```
//Zuweisungsoperator
```

```
ComplUneqConstr& ComplUneqConstr::operator=(const ComplUneqConstr& rhs)
```

```
{
```

```
}

void ComplicatingConstraints::initLagrangianMultipliers()
{
    //Initialization of all lambdas with 0.1 (it could be also zero)
    for (unsigned int i=0; i<compl_eq_constraints.size();i++)
        compl_eq_constraints[i].lambda = 0.1;
    //Initialization of all mu with 0.1 (it could be also zero)
    for (unsigned int i=0; i<compl_uneq_constraints.size();i++)
        compl_uneq_constraints[i].mu = 0.1;

    #if LAGRANGIAN_INIT_TEST==1
        cout << "Initialization of all lambdas and mus with zero \n";
    #endif

}

ComplicatingConstraints::ComplicatingConstraints(vector<vector<bool> >& x_rest,
        vector<vector<vector<vector<bool> > >& x_pw,
        vector<vector<float> >& E_rest,
        vector<vector<vector<vector<float> > >& E_pw,
        vector<vector<bool> >& N_plus_V)
{

    pair <unsigned int,unsigned int> xv_index_sc_rot; //Hilfsvariable
```

```

unsigned int count_Euv_greater_zero = 0;
field_compl_eq_constraints.resize(E_rest.size());
field_compl_uneq_constraints.resize(E_rest.size());
for (unsigned int i=0; i < E_rest.size(); i++){
    //allocate array
    field_compl_eq_constraints[i].resize(E_rest[i].size());
    field_compl_uneq_constraints[i].resize(E_rest[i].size());
    for (unsigned int s=0; s < E_rest[i].size(); s++){
        //allocate array
        field_compl_eq_constraints[i][s].resize(E_rest.size());
        field_compl_uneq_constraints[i][s].resize(E_rest.size());
        for (unsigned int j=0; j < E_rest.size(); j++){
            if ( (N_plus_V[j][i]) ){ // v element aus N+(V_j)
                xv_index_sc_rot.first=i;
                xv_index_sc_rot.second=s;

                ComplEqConstr compl_eq_constraint(xv_index_sc_rot, x_rest[i][s],j,E_pw[i][s][j].size());

                //compl_eq_constraints.push_back(compl_eq_constraint);

                field_compl_eq_constraints[i][s][j] = compl_eq_constraint;
            }
            else { // v nicht element aus N+(V_j)
                xv_index_sc_rot.first=i;
                xv_index_sc_rot.second=s;

                for (unsigned int t=0; t < E_rest[j].size(); t++){

                    if (E_pw[j][t][i][s] > 0)
                        count_Euv_greater_zero++;
                }
            }
        }
    }
}

```

```

    ComplUneqConstr                                compl_uneq_constraint(xvv_index_sc_rot,
x_rest[i][s],j,count_Euv_greater_zero);

    //compl_uneq_constraints.push_back(compl_uneq_constraint); //Benutzung des Copy-Konstruktors

    field_compl_uneq_constraints[i][s][j]          =    compl_uneq_constraint;    //Benutzung    des
Zuweisungsoperators
        }
    }
}

}

}

//Zuweisungsoperator
ComplicatingConstraints& ComplicatingConstraints::operator=(const ComplicatingConstraints& rhs)
{
}

//Definition von Funktionen von InnerOptimization

void InnerOptimization::makeInnerOptimization(vector<vector<bool> >& x_rest,
        vector<vector<float> >& E_rest,
        // vector < vector<bool> >& N_plus_V,
        ComplicatingConstraints& complicating_constraints)
{
    cout << "inner optimization. Bei festem Lambdas und Mus werden die x_vv upgedated.\n";

    // Bestimmung der x_vv: for i=1..sidechains_of_protein.size()-1: nur eines der x_ij ist 1, der Rest null.

        // E_vv[i][s]-(lambda[i][s][0]+...+lambda[i][j][]) +

```


Protein Sidechain placing

```
//      mu[i][j][0]+...+ mu[i][j][]

//für jedes i gibt es j=0..sidechains_of_protein[i].rotamers_of_sidechain.size()-1
Summen,

//es ist das j zu nehmen, wodurch sich die minimale Summe ergibt
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//!!!!!!!!!!!!!!!!!!!! Diese Implementierung ist nicht optimal, da der gesamte lambda und mu-Raum pro k-Schleife
durchgegangen wird.
// !!!!!!!!!!!!!!! Dies ist nicht nötig, hier muss noch der Code optimiert werden
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//float
sum[MAX_POSSIBLE_SIDECHAINS_IN_PROTEIN][MAX_POSSIBLE_ROTAMERS_IN_SIDECHAIN];

float
sum[MAX_POSSIBLE_SIDECHAINS_IN_PROTEIN][MAX_POSSIBLE_ROTAMERS_IN_SIDECHAIN];

//sum[i][j][k] "hart" als Array und nicht als Vektor kodiert (zur Vereinfachung)
for (unsigned int i=0; i < x_rest.size(); i++){
for (unsigned int s=0; s < x_rest[i].size(); s++){

sum[i][s]= E_rest[i][s];

for (unsigned int j=0; j < E_rest.size(); j++){
sum[i][s]= sum[i][s] - complicating_constraints.field_compl_eq_constraints[i][s][j].lambda;
- complicating_constraints.field_compl_uneq_constraints[i][s][j].mu;
}

// schauen, ob es unter vector eine Funktion gibt, die die Summe der Vektorelemente
angibt,
// dann kann man auch die Elemente
// k<i auf null setzen
```

```
    }
  }

    // Bestimmung des Index für die minimale Summe
float optimal_sum_index[MAX_POSSIBLE_SIDECHAINS_IN_PROTEIN];
    for (unsigned int i=0; i < x_rest.size(); i++){
    optimal_sum_index[i]=0;
        for (unsigned int s=1; s<x_rest[i].size(); s++){
            optimal_sum_index[i]=(sum[i][s] < sum[i][0]) ? s: optimal_sum_index[i];
        }
    }

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//!!!!!!!!!!!!!!!!!!!! Diese Implementierung ist nicht optimal.
// !!!!!!!!!!!!!!! hier muss noch der Code optimiert werden.
// beim neuen x_vv nur die veränderten x_vv-Werte durchgegangen werden müssen, vorher müsste ein
Flag gesetzt werden
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

        // Setzen der neuen x_vv-Werte
    for (unsigned int i=0; i < x_rest.size(); i++){
        for (unsigned int j=0; j<x_rest[i].size(); j++){
            x_rest[i][j] = 0;
            if (j==optimal_sum_index[i])
                x_rest[i][j] = 1;
        }
    }

}

//end    InnerOptimization::makeInnerOptimization(Protein    protein,    ComplicatingConstraints
complicating_constraints)
```

```
#endif //_GMEC_LR_
```

15.5 gmec_lr_SCP.C

```
// gmec_lr_SCP.C
//Version: 9.12.05
//Author: Samir Mourad
#include <BALL/common.h>
#include <BALL/DATATYPE/string.h>
#include <BALL/MATHS/vector3.h>
#include <BALL/COMMON/limits.h>
//#include "basicTree.h"

#include "gmec_lr.h"

#include <algorithm>
#include <fstream>

using namespace BALL;
using namespace std;
/*
struct EnergyData
{
    float energy;
    Size residue_index;
    Size rotamer_index;
```

```

};

*/

/*
bool less_than(BasicTree<EnergyData>* e1, BasicTree<EnergyData>* e2)
{
    return(e1->getData().energy < e2->getData().energy);
}

*/

Size max_candidates = 1;

int main(int argc, char** argv)
{
    Log.setPrefix(cout, "[%T] ");
    Log.setPrefix(cerr, "[%T:ERROR] ");
    if ((argc != 3) && (argc != 4) && (argc != 5))
    {
        Log.error() << argv[0] << " <energies file> <output file> [<max_candidates>
[<max_leaves>]]" << endl;
        Log.error() << " calculate the optimal rotamers of the binding-site side-chains with
lagrangian multipliers" << endl;
        Log.error() << " The optimal rotamers are written to <output file>" << endl;
        Log.error() << " if max_candidates is not given, a default of 1 is assumed." << endl;
        Log.error() << " if max_leaves is not given, a default of 20000 is assumed." << endl;
        return 1;
    }

    ifstream energies(argv[1]);
    if (!energies)
    {

```

```
    Log.error() << "could not open energies file: " << argv[1] << endl;
    return 1;
}

ofstream output(argv[2], ios::out);
if (!output)
{
    Log.error() << "could not open energies file: " << argv[1] << endl;
    return 1;
}

if (argc == 4)
{
    max_candidates = atoi(argv[3]);
}

Size max_leaves = 20000;
if (argc == 5)
{
    max_leaves = atoi(argv[4]);
}

Log.info() << "max_leaves = " << max_leaves << endl;

String line;

line.getline(energies);

// ignore all comment lines
while (line[0] == ';')
{
    line.getline(energies);
}
```

```
// read the PDB filenames
String filename_A =line;
line.getline(energies);

//String filename_B = line;
//line.getline(energies);

Log.info() << "A will be read from: " << filename_A << endl;
//Log.info() << "B will be read from: " << filename_B << endl;

// read the contact distance
//      float contact_distance;
//      contact_distance = line.toFloat();
//      Log.info() << "contact distance: " << contact_distance << " A" << endl;

// read the template energy
float template_energy;
//line.getline(energies);
template_energy = line.toFloat();
Log.info() << "template energy: " << template_energy << " kJ/mol" << endl;

// create data structures for the residue names and energies

vector<String> residue_names;

// read the number of residues in the binding site
//line.getline(energies);
//Size number_of_residues = line.toUnsignedInt();
//Log.info() << "number of residues: " << number_of_residues << endl;

// read the number of residues in A
line.getline(energies);
Size number_of_residues_of_A = line.toUnsignedInt();
```

```
Log.info() << "number of residues in A: " << number_of_residues_of_A << endl;

// create a twodimensional field to hold the interaction energies
vector<vector<float>> E_rest(number_of_residues_of_A);

// create a twodimensional field to hold vars of the interaction energies
vector<vector<bool>> x_rest(number_of_residues_of_A);

// read the residues and their energies
Size i, j;
for (i = 0; i < number_of_residues_of_A; i++)
{
    // read the line
    line.getline(energies);

    // save the residue path
    residue_names.push_back(line.getField(0));
    Log.info() << "reading residue " << residue_names.back() << endl;

    // check the residue index
    if (line.getField(1).toUnsignedInt() != i)
    {
        Log.error() << "read wrong residue index: " << line.getField(1) << " while
expecting " << i << endl;
        return 2;
    }

    // read the number of rotamers
    Size number_of_rotamers = line.getField(2).toUnsignedInt();

    // resize E_rest
    E_rest[i].resize(number_of_rotamers);
    // resize x_rest
    x_rest[i].resize(number_of_rotamers);
```

```
// read the rotamer energies E_rest and initialize x_rest (these are the x_vv)
for (j = 0; j < number_of_rotamers; j++)
{
    E_rest[i][j] = line.getField(3 + j).toFloat();
x_rest[i][j] = 0;

}
}
Log.info() << "done." << endl;
/*
// read the number of residues in B
line.getline(energies);
Size number_of_residues_of_B = line.toUnsignedInt();
Log.info() << "number of residues in B: " << number_of_residues_of_B << endl;

for (i = number_of_residues_of_A; i < number_of_residues; i++)
{
    // read the line
    line.getline(energies);

    // save the residue path
    residue_names.push_back(line.getField(0));

    // check the residue index
    if (line.getField(1).toUnsignedInt() != i)
    {
        Log.error() << "read wrong residue index: " << line.getField(1) << " while
expecting " << i << endl;
        return 2;
    }
}
```



```
// read the number of rotamers
Size number_of_rotamers = line.getField(2).toUnsignedInt();

// resize E_rest
E_rest[i].resize(number_of_rotamers);
// resize x_rest
x_rest[i].resize(number_of_rotamers);

// read the rotamer energies E_rest and initialize x_rest (these are the x_vv)
for (j = 0; j < number_of_rotamers; j++)
{
    E_rest[i][j] = line.getField(3 + j).toFloat();
x_rest[i][j] = 0;
}
}
*/

// read the pairwise energies

// create a vector for the pairwise energies
vector<vector<vector<vector<float>>>> E_pw(number_of_residues_of_A);
vector<vector<vector<vector<bool>>>> x_pw(number_of_residues_of_A);
Size s, t;
// allocate the correct vector sizes
for (i = 0; i < number_of_residues_of_A; i++)
{
    // allocate the array
    E_pw[i].resize(E_rest[i].size());
    x_pw[i].resize(E_rest[i].size());
    for (s = 0; s < E_rest[i].size(); s++)
    {
```

```
// allocate the array
E_pw[i][s].resize(number_of_residues_of_A);
x_pw[i][s].resize(number_of_residues_of_A);

for (j = 0; j < number_of_residues_of_A; j++)
{
    // allocate the array
    E_pw[i][s][j].resize(E_rest[j].size());
    x_pw[i][s][j].resize(E_rest[j].size());
}
}

Size count = 0;
// read the rotamer energies E_pw and initialize x_pw (these are the x_uv)
for (i = 0; i < (number_of_residues_of_A - 1); i++)
{
    for (j = i + 1; j < number_of_residues_of_A; j++)
    {
        for (s = 0; s < E_rest[i].size(); s++)
        {
            for (t = 0; t < E_rest[j].size(); t++)
            {
                // read a line from the emrgies file
                line.getline(energies);

                // verify the indices
                if ((line.getField(0).toUnsignedInt() != i)
                    || (line.getField(1).toUnsignedInt() != j)
                    || (line.getField(2).toUnsignedInt() != s)
                    || (line.getField(3).toUnsignedInt() != t))
                {
```

```

        Log.error() << "wrong indices: " << line << endl;
    }

    count++;

    // assign pairwise energy
    E_pw[i][s][j][t] = line.getField(4).toFloat();
    x_pw[i][s][j][t] = 0;
    E_pw[j][t][i][s] = E_pw[i][s][j][t];
    x_pw[j][t][i][s] = x_pw[i][s][j][t];
    }
}
}

Log.info() << "read " << count << " pairwise energies." << endl;

// Bestimmung der N+(V_j):
vector < vector<bool> > N_plus_V;
N_plus_V.resize(number_of_residues_of_A);
for (i=0; i < E_rest.size(); i++)
    N_plus_V[i].resize( E_rest[i].size() );

for (j=0; j < E_rest.size(); j++){
    for (i=0; i < E_rest.size(); i++){
        for (t=0; t < E_rest[j].size(); t++){
            for (s=0; s < E_rest[i].size(); s++){
                if (E_pw[j][t][i][s]>0)
                    N_plus_V[j][i]=1;
                else
                    N_plus_V[j][i]=0;
            }
        }
    }
}

```

```

    }
    //cout << " N_plus_V["<<j<< "]"<<i<<"]= "<< N_plus_V[j][i] << "\n";
    }
}
cout << " N+(V_j) bestimmt\n";
// Log.info() << " N+(V_j) bestimmt\n";
//evtl. als Funktion implementieren, wo beim ersten E_pw>0 der Ausstieg erfolgt

////vorerst

//          char*          lagrange_initfile          =
"/home/samir/src/docking_and_SCP_tools_work/lagrange_initfile.txt"; //

// Ende vorerst
N_plus_V[1][0]=1;
//Aufbau der complicating constraints
ComplicatingConstraints complicating_constraints(x_rest,
        x_pw,
        E_rest,
        E_pw ,
        N_plus_V);

//////Here begins the algorithm

////////////////////////////////////
//STEP1: Initialization
////////////////////////////////////

```

Protein Sidechain placing

```
// Initialisation of user-defined parameter pi (for stopping criteria)
int stopping_criteria_pi;
stopping_criteria_pi = 2;

// Initialization of N_LR (number of Lagrange operations)
unsigned int N_LR=0;

//subgradients (for STEP 4)
vector <float> gradient_eq;
vector <float> gradient_uneq;

//scalar step size T
float T;
T=2;

InnerOptimization inner_optimization;

while (stopping_criteria_pi > 0.0005){

////////////////////////////////////
//STEP2: calculate lower bound with subragient method Z_LB ={x_vv, x_uv}
////////////////////////////////////

// Inner optimization. Bei festem Lambdas und Mus werden die x_vv upgedated.
inner_optimization.makeInnerOptimization(x_rest, E_rest,
//N_plus_V,
complicating_constraints);

////////////////////////////////////
//STEP3: calculate upper bound with heuristic method
////////////////////////////////////
```

//this step is ignored

////////////////////////////////////

//STEP4: Update the lagrangian multipliers

////////////////////////////////////

//// 1. Calculate the Subgradients $G_t[k]$ for the current solution vector (x_{vv}, x_{uv})

///// If all $G_i \leq 0$ for each \geq constraint, then Z_{LB} (the actual solution vector) is feasible

if ($N_{LR} == 0$){

for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)

gradient_eq.push_back(0.0);

for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)

gradient_uneq.push_back(0.0);

for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)

gradient_eq[i] = -(complicating_constraints.compl_eq_constraints[i].sum_x_uv_relaxed
- complicating_constraints.compl_eq_constraints[i].x_vv);

for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)

gradient_eq[i] = -(complicating_constraints.compl_uneq_constraints[i].sum_x_uv_relaxed
- complicating_constraints.compl_uneq_constraints[i].x_vv);

}

else {

for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)

gradient_eq[i] = -(complicating_constraints.compl_eq_constraints[i].sum_x_uv_relaxed
- complicating_constraints.compl_eq_constraints[i].x_vv);

for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)

gradient_uneq[i] = -(complicating_constraints.compl_uneq_constraints[i].sum_x_uv_relaxed
- complicating_constraints.compl_uneq_constraints[i].x_vv);

}

//// 2. Define a scalar step size

$T = 1/\sqrt{T}$; //alle zwei Durchläufe wird die Schrittweite halbiert

```

//// 3. Update der Lagrangian Multipliers

for (int i=0; complicating_constraints.compl_eq_constraints.size(); i++)
    complicating_constraints.compl_eq_constraints[i].lambda =
        (complicating_constraints.compl_eq_constraints[i].lambda
         + T * gradient_eq[i]) < 0.0 ? 0.0 :
(complicating_constraints.compl_eq_constraints[i].lambda
 + T * gradient_eq[i]);

for (int i=0; complicating_constraints.compl_uneq_constraints.size(); i++)
    complicating_constraints.compl_uneq_constraints[i].mu =
        (complicating_constraints.compl_uneq_constraints[i].mu
         + T * gradient_uneq[i]) < 0.0 ? 0.0
:(complicating_constraints.compl_uneq_constraints[i].mu
 + T * gradient_uneq[i]);

N_LR++;
//vorerst:
stopping_criteria_pi = stopping_criteria_pi/3;
};//while

Log.info() << "N_LR= " << N_LR << endl;

//////////Ausgabe in Datei, die in structure_generator gegeben wird

    Log.info() << "writing solutions to " << argv[2] << endl;
    output << "# created by " << argv[0] << " from " << argv[1] << endl;
    // output << contact_distance << endl;

    int number_of_solutions =1;
    output << number_of_solutions << endl;

```

```
output << argv[1] << endl;
// Output the energy_value and the system
//vorest wird die template energy anstatt energy value benutzt
template_energy=4e+15;
    output << template_energy << " " << number_of_residues_of_A << endl;

    for (unsigned int j=0; j< residue_names.size(); j++)
    {
        output << residue_names[j];
        output << " ";
        output << j;
    output << " ";
        for (unsigned int t=0; t < E_rest[j].size(); t++){
            if (x_rest[j][t]==1){
                output << t << " " << endl;
            }
        }
    }
    Log.info() << " Anzahl der Rotamere: " << x_rest[j].size() << "\n";
        // hier noch optimieren: raus aus der for t=0..-Schleife
    }

    Log.info() << "Anzahl der Residuen: " << E_rest.size() << " "<< "\n";
```



```
        output.close();  
    cout << "nach output.close()\n";  
  
        return 0;  
}
```

15.6 structure_generator_SCP.C

```
// structure_generator_SCP.C
//Version: 9.12.05
//Author: Samir Mourad

#include <algorithm>
#include <BALL/MOLMEC/AMBER/amber.h>
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/STRUCTURE/fragmentDB.h>
#include <BALL/STRUCTURE/residueChecker.h>
#include <BALL/STRUCTURE/rotamerLibrary.h>
#include <BALL/COMMON/limits.h>
#include <BALL/DATATYPE/hashGrid.h>
#include <BALL/STRUCTURE/geometricProperties.h>

using namespace BALL;
using namespace std;

#include "util.h"

inline void calculateInitialLists(list<Residue*>& contacts_A, System& A)
{
    /*
        BoundingBoxProcessor box;
        A.apply(box);

        HashGrid3<Atom*>  grid(box.getLower() - Vector3(distance),
box.getUpper() - box.getLower() + Vector3(distance * 2.0),
                                                                    distance);

        AtomIterator atom_it = A.beginAtom();
```

```
        for (; +atom_it; ++atom_it)
    {
        grid.insert(atom_it->getPosition(), &(*atom_it));
    }

    // now iterate over all residues of B and
    // store the residue in a list if any of its atoms has a contact with A
    // residues in A are selected and collected afterwards
    contacts_B.clear();

*/

    ResidueIterator res_it;
    /*
    HashGridBox3<Atom*>::BoxIterator box_it;
    HashGridBox3<Atom*>::DataIterator data_it;
    HashGridBox3<Atom*>* closest_box;
    Vector3 atom_position;
    float distance_2 = distance * distance;

    for (res_it = B.beginResidue(); +res_it; ++res_it)
    {
        bool contact = false;
        for (atom_it = res_it->beginAtom(); +atom_it; ++atom_it)
        {
            atom_position = atom_it->getPosition();
            closest_box = grid.getBox(atom_it->getPosition());

            if (closest_box != 0)
                for (box_it = closest_box->beginBox(); +box_it; ++box_it)
                {
                    for (data_it = box_it->beginData(); +data_it; ++data_it)
                    {
```

```
if (atom_position.getSquareDistance((*data_it)->getPosition())
<= distance_2)
    {
        contact = true;
        Fragment* frag = (*data_it)->getFragment();
        if (frag != 0)
            {
                frag->select();
            }
    }
}

if (contact)
{
    contacts_B.push_back(&(*res_it));
}
}

*/
// collect selected residues in A
contacts_A.clear();
for (res_it = A.beginResidue(); + res_it; ++res_it)
{
    // if (res_it->isSelected())
    // {
        contacts_A.push_back(&(*res_it));
    // }
}

Log.info() << "binding site contains " << contacts_A.size() << " residues of A" << endl;
}
```

```
int main(int argc, char** argv)
{
    Log.setPrefix(cout, "[%T] ");
    Log.setPrefix(cerr, "[%T:ERROR] ");
    /*
    if (argc != 4 && argc != 5)
    {
        Log.error() << argv[0] << " <rotamer output file> <PDB outfile A> <PDB outfile B>
[<number of solution>]" << endl;
        Log.error() << " read an output file from a rotamer placement tool" << endl;
        Log.error() << " and create two PDB files from it" << endl;
        Log.error() << " if <number of solution> is not given, the first solution" << endl;
        Log.error() << " from the output file is assumed." << endl;
        return 1;
    }
    */

    if (argc != 3 && argc != 4)
    {
        Log.error() << argv[0] << " <rotamer output file> <PDB outfile A> [<number of
solution>]" << endl;
        Log.error() << " read an output file from a rotamer placement tool" << endl;
        Log.error() << " and create one PDB files from it" << endl;
        Log.error() << " if <number of solution> is not given, the first solution" << endl;
        Log.error() << " from the output file is assumed." << endl;
        return 1;
    }

    ifstream rotamer_file(argv[1]);
```

```
    if (rotamer_file.bad())
    {
        Log.error() << "Cannot open rotamer output file " << argv[1] << endl;
        return 2;
    }

    Log.info() << "reading rotamer file " << argv[1] << endl;

    String line;
    line.getline(rotamer_file); //erste Zeile aus rotamer file (.rot-File, Ausgangsfile von
gmec_lr_SCP)

    //Log.info() << "Zeile vor read the number of solutions" << line << endl;

    // read the number of solutions in this file
    line.getline(rotamer_file);
        Size number_of_solutions =1;
    //number_of_solutions = line.toUnsignedInt();
        //Log.info() << "file contains " << number_of_solutions << " solutions." << endl;

    // read the filename for the energies.dat file
    line.getline(rotamer_file);
    String energy_filename = line;
    Log.info() << "energies file: " << energy_filename << endl;

    // read all solutions
    vector<vector<String>> rotamer_solutions(number_of_solutions);
    Size i, j;
    for (i = 0; i < number_of_solutions; i++)
    {
        Log.info() << "reading solution " << i << endl;
```

```
        line.getline(rotamer_file);
        Size residues = line.getField(1).toUnsignedInt();
        for (j = 0; j < residues; j++)
        {
            line.getline(rotamer_file);
            rotamer_solutions[i].push_back(line);
        }
    }

    rotamer_file.close();
    Log.info() << "reading solutions done." << endl;

    // choose the solution to set
    Size solution_index = 0;
    if (argc == 3)
    {
        solution_index = atoi(argv[2]) - 1;
        if (solution_index >= number_of_solutions)
        {
            Log.error() << "invalid solution index " << argv[2] << " (valid range: 1 - " <<
number_of_solutions << endl;
            return 6;
        }
    }
    Log.info() << "selection solution " << solution_index << endl;

    // now read the interesting parts of the energies.dat file
    ifstream energies_file(energy_filename.c_str());
    if (energies_file.bad())
    {
        Log.error() << "cannot open energies.dat file: " << energy_filename << endl;
        return 3;
    }
}
```

```
    }

    // step over comments
    line.getline(energies_file);
    while (line[0] == ';')
    {
        line.getline(energies_file);
    }

    // read the filenames of A and B
    String filename_A = line;
    //line.getline(energies_file);
    //String filename_B = line;
String filename_B;

    Log.info() << "reading A from " << filename_A << endl;
    //Log.info() << "reading B from " << filename_B << endl;

    // read the contact distance (in Angstrom)
    //line.getline(energies_file);
float contact_distance;

    //float contact_distance = line.toFloat();

line.getline(energies_file);
    float energy_value = line.toFloat();

    // that's everything we need to know
    energies_file.close();

    // read A and B
    PDBFile pdb_file;
    pdb_file.open(filename_A);
```



```
Log.info() << "pdb_file.open(filename_A); " << endl;
    if (pdb_file.bad())
    {
        Log.error() << "cannot open file for protein A: " << filename_A << endl;
        return 4;
    }
System A;
pdb_file >> A;
pdb_file.clear();
pdb_file.close();
Log.info() << "read " << A.countAtoms() << " atoms from " << filename_A << endl;

/*
pdb_file.open(filename_B);
if (pdb_file.bad())
{
    Log.error() << "cannot open file for protein B: " << filename_B << endl;
    return 5;
}

System B;
pdb_file >> B;
pdb_file.clear();
pdb_file.close();
Log.info() << "read " << B.countAtoms() << " atoms from " << filename_B << endl;
*/

// calculate the binding site residues
list<Residue*> initial_residues_A;
// list<Residue*> initial_residues_B;
calculateInitialLists(initial_residues_A, A);
```

```
// create fragment database and rotamer library
// (Dunbrack's backbone-independent rotamer library 08/1999)
Log.info() << "creating fragment and rotamer databases..." << endl;
FragmentDB frag_db("/usr/local/BALL/data/fragments/Fragments.db");
RotamerLibrary rot_lib("rotamers/bbind99.Aug.lib", frag_db);

// remove residues without rotamers from the binding site vectors
// and create vectors of the corresponding rotamer sets
Log.info() << "setup rotamer sets..." << endl;
vector<Residue*> residues;
vector<ResidueRotamerSet> rotamer_sets;
list<Residue*>::const_iterator list_it;
vector<String> residue_names;
for (list_it = initial_residues_A.begin(); list_it != initial_residues_A.end(); ++list_it)
{
    const Fragment* frag = frag_db.getReferenceFragment(**list_it);
    if (frag != 0)
    {
        if (!frag->hasProperty(Residue::PROPERTY__HAS_SSBOND))
        {
            const ResidueRotamerSet* rotamer_set_pointer = rot_lib.getRotamerSet(frag->getName());
            if (rotamer_set_pointer != 0)
            {
                ResidueRotamerSet rs(*rotamer_set_pointer);
                if (rs.getNumberOfRotamers() > 0)
                {
                    rotamer_sets.push_back(rs);
                    residues.push_back(*list_it);
                    residue_names.push_back(getResiduePathName(**list_it));

                    Log.info() << "found rotamers for side chain " << residue_names.back()
                        << " (index " << residues.size() - 1 << ")" << endl;
                }
            }
        }
    }
}
```

```
        }
        else
        {
            Log.info() << "Cannot find a rotamer set for residue " << (*list_it)->getName() << " of A" << endl;
        }
    }
    else
    {
        Log.info() << "Residue " << (*list_it)->getName() << " ignored - has sulphur bridge!" << endl;
    }
}
else
{
    Log.warn() << "Could not find a reference fragment for " << (*list_it)->getName() << endl;
}
}

// remember the number of residues from A
Size number_of_residues_of_A = residues.size();
/*
for (list_it = initial_residues_B.begin(); list_it != initial_residues_B.end(); ++list_it)
{
    const Fragment* frag = frag_db.getReferenceFragment(**list_it);
    if (frag != 0)
    {
        if (!frag->hasProperty(Residue::PROPERTY__HAS_SSBOND))
        {
            const ResidueRotamerSet* rotamer_set_pointer = rot_lib.getRotamerSet(frag->getName());
            if (rotamer_set_pointer != 0)
            {
                ResidueRotamerSet rs(*rotamer_set_pointer);
                if (rs.getNumberOfRotamers() > 0)
```

```
{
    rotamer_sets.push_back(rs);
    residues.push_back(*list_it);
    residue_names.push_back(getResiduePathName(**list_it));

    Log.info() << "found rotamers for side chain " << residue_names.back()
        << " (index " << residues.size() - 1 << ")" << endl;
}
else
{
    Log.info() << "Cannot find a rotamer set for residue " << (*list_it)->getName() << " of B" << endl;
}
}
else
{
    Log.info() << "Residue " << (*list_it)->getName() << " ignored - has sulphur bridge!" << endl;
}
}
else
{
    Log.warn() << "Could not find a reference fragment for " << (*list_it)->getName() << endl;
}
}

*/

Size number_of_residues = residues.size();

Log.info() << "calculate initial rotamers..." << endl;
vector<Rotamer> initial_rotamers;
for (i = 0; i < residues.size(); i++)
{
```

Protein Sidechain placing

```
// calculate the initial rotamer of the residues
Rotamer r = rotamer_sets[i].getRotamer(*residues[i]);
initial_rotamers.push_back(r);

// add it to the residue`s rotamer set
rotamer_sets[i].addRotamer(r);
    }

    // parse the solution
    if (number_of_residues != rotamer_solutions[solution_index].size())
    {
        Log.error() << "wrong number of residues in rotamer solution: " <<
rotamer_solutions[solution_index].size() << " instead of " << number_of_residues << endl;
        return 8;
    }

    vector<Size> rotamer_indices;

    for (i = 0; i < rotamer_solutions[solution_index].size(); i++)
    {

        rotamer_indices.push_back(rotamer_solutions[solution_index][i].getField(2).toUnsign
edInt());
    }

    // now set the rotamers
    if (rotamer_indices.size() != number_of_residues)
    {
        Log.error() << "wrong number of side chains!" << endl;
        exit(1);
    }

    Log.info() << "assigning rotamers" << endl;
```

```
for (i = 0; i < number_of_residues; i++)
{
    rotamer_sets[i].setRotamer(*residues[i], rotamer_sets[i][rotamer_indices[i]]);
    Log.info() << residue_names[i] << ": " << i << "/" << rotamer_indices[i] << endl;
}
Log.info() << "done." << endl;

Log.info() << "A contains " << A.countAtoms() << " atoms" << endl;
//      Log.info() << "B contains " << B.countAtoms() << " atoms" << endl;

// writing structures

Log.info() << "writing " << argv[2] << endl;
pdb_file.open(argv[2], ios::out);
if (pdb_file.bad())
{
    Log.error() << "cannot open file for protein A: " << argv[2] << endl;
    return 10;
}
pdb_file << A;
pdb_file.clear();
pdb_file.close();

/*
Log.info() << "writing " << argv[3] << endl;
PDBFile pdb_file2;
pdb_file2.open(argv[3], ios::out);
if (pdb_file2.bad())
{
    Log.error() << "cannot open file for protein B: " << argv[3] << endl;
    return 11;
}
pdb_file2 << B;
```

```
        pdb_file2.close();
        */
        Log.info() << "done." << endl;

        return 0;
    }
}
```

15.7 SCP.sh

#SCP.sh, Version: 09.12.05, Author: Samir Mourad

#1. der Name des Proteins im Energie file (z.B. 1igd.pdb_1igd.pdb_energies_complete.dat)

muss zusammengeschrieben sein (momentan muss die Datei editiert und der Name per Hand geändert werden)

#2. in DEE_complete.C muss der Pfad der Fragmente angepasst werden

(z.B. /usr/local/BALL/data/fragments/Fragments.db)

#Programmanwendung:

#SCP:

./DEE_complete_SCP 1igd_5peptid.pdb # -> erzeugt <energiefile>

./gmec_lr_SCP 1igd_5peptid.pdb_energies_complete.dat 1igd_5peptid.rot # -> erzeugt <rotamer_output>

./structure_generator_SCP 1igd_5peptid.rot 1igd_5peptid_complete.pdb # -> erzeugt <protein_complete.pdb>

16 ANHANG B: Zeitplan, Arbeitspakete und tatsächliche Arbeitsstunden

16.1 Zeitplan (Erstellt September 2005)

Geplant

Tatsächlich

Implementierung von GMEC-LR

Einarbeitung in effektives C++- Mo, 19.9.05-
Programmieren:

1. Effective C++/C++ Primer lesen
zunächst das GMEC-LR Programm
nochmals lesen, um zu sehen, welche
Themen zunächst optimiert werden
sollen.

- memory handling
- classes
- initialization

Variablen- und Funktionsnamen im
bisheriger Code GMEC-LR ändern.

Algorithmus ändern: kurz in Paper
beschreiben in kapitel „Implementation“

Implementierung unter Beachtung effektiver
Programmregeln

Test am vorhandenen Datenset

Geplant

Tatsächlich

Einarbeitung, wie große Testsets gemacht werden

Im Paper das entsprechende Kapitel
schreiben.

Geplant

Tatsächlich

Einbettung des Programms nach BALL

	Geplant	Tatsächlich
Einarbeitung in Immunoinformatics		

	Geplant	Tatsächlich
Vollständige Implementierung von GMEC-LR	28.10.-18.11.(3 Wochen)	

Einarbeitung in BALL und ILP-Code von 1 Woche: ab 28.10.

Oliver:

der Code muss 1. mit der aktuellen BALL-Version zum Laufen gebracht werden und 2. der Optimierungsteil gegen GMEC-LR ausgetauscht werden.

Eingabe: PDB-File 3 Tage

1. Berechnung des Energiefiles

2. GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile 3 Tage: parallel zu Einarbeitung in BALL...

3. **Strukturerzeugung** 1 Woche

16.2 Besprechungen

Bespr. Vom 10.8.05

Arbeitspunkte	Bemerkungen	Zu erledigen bis	Erledigt ?
<p>Code: dynam. allozieren->für jede Seitenkette</p> <p>Datenstrukturen->Klassen auf die Constraints</p> <p>Kleines Beispiel: z.B. 2 Seitenketten -> Programm verifizieren (per Hand mögl. nachrechnen)</p> <p>Variablenname calc_lower</p> <p>Methodenname: calcLower...</p> <p>Literatur:</p> <p>Effective C++, Primer</p> <p>Iteratoren -</p>			

e

Bespr. Vom 26.10.05, 10.45-11 in Tübingen, Teilnehmer: Samir Mourad, Oliver Kohlbacher

Arbeitspunkte	Bemerkungen	Zu erledigen bis	Erledigt ?
Klasse Energiewechselwirkungen	für Zu überlegen, ob als Element eines Rotamers;		
Deadline Veröffentlichung GMEC-LR in einem Paper	für von	Ende 2005, d.h. in zwei Monaten	
Gesamtprogramm: Eingabe: PDB-File	Oliver schickt Code vom ILP- Paper, der Code muss 1. mit der aktuellen BALL-Version zum Laufen gebracht werden		
5. Berechnung Energiefiles	des zum Laufen gebracht werden		
6. GMEC-LR (Lösen des Optimierungsproblems), Rotamere kommen aus Energiefile	und 2. der Optimierungsteil gegen GMEC-LR ausgetauscht werden.		
7. Strukturzeugung Nächster Termin: 30.11.2005, 16-17 Tübingen	Mi, in	29.11.05	

Bespr. vom

Arbeitspunkte	Bemerkungen	Zu erledigen bis	Erledigt ?

Bespr. vom Arbeitspunkte	Bemerkungen	Zu erledigen bis	Erledigt ?
------------------------------------	--------------------	---------------------------------	-----------------------

Bespr. vom Arbeitspunkte	Bemerkungen	Zu erledigen bis	Erledigt ?
------------------------------------	--------------------	---------------------------------	-----------------------

Protein Sidechain placing

Arbeitsstunden:

Soll: ca. 4000 Stunden bis zum Ende der Promotion inscha Allah (Stand 21.9.2005)

tatsächliche Arbeitsstunden:

Monat

Geleistete Arbeitsstunden

Bearbeitete Arbeitspakete

19.09.05-30.09.2005

01.10.05-31.10.2005

Arbeitsstunden Samir Mourad/VGÖG für Promotion

Sept. 2005

	Kommen	Gehen	Pause	gearbeitete Stunden	Arbeitsgebiet
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
				0	
19.09.05				4	Einarbeitung C++
20.09.05				4	Einarbeitung C++
21.09.05	9	17	2	6	Einarbeitung C++
22.09.05	9,5	12,5		3	Einarbeitung C++
23.09.05				0	
24.09.05				0	
25.09.05				0	
26.09.05	8	12		4	Einarbeitung C++ Dokum. 2.Zwischenb.
27.09.05				4	Einarbeitung C++ Dokum. 2.Zwischenb.
28.09.05	11	19	3	5	Einarbeitung C++ Dokum. 2.Zwischenb.
29.09.05	7,5	19	3	8,5	
30.09.05	7,5	19	3	8,5	
				0	
				0	
				Gesamtarbeitsstunden:	
				47	

Zu tun:

<i>Datum</i>	<i>Arbeitspunkt</i>	<i>Zu erledigen bis</i>
	Effektiv C++ lesen:	03.10.05
22.09.05	• Speicherverwaltung	
	• Klassen	
01.10.05		
22.09.05	Algorithmus ändern: nach x_uv optimieren, weiterhangeln (siehe handschriftl. Notiz vom 10.8.05)	03.10.05
22.09.05	Variablenamen, Funktionsnamen im Programm ändern	03.10.05
22.09.05	Code ändern: 1. Speicherbereitstellung für die Vars 2. Klassen für Datenelemente erstellen, für die es bisher keine Klassen gibt.	03.10.05
22.09.05	Code an kleinem Beispiel (z.B. 2 Seitenketten, je 3 Rotamere) ausprobieren und per Hand nachrechnen, ob er das richtige macht.	03.10.05
22.09.05	Code ausprobieren am grossen Beispiel von Kingsford	03.10.05
22.09.05	Code in BALL einbetten	03.10.05
22.09.05	Einarbeitung, wie man große Datensätze untersucht (-> Paper von Kingsford)	03.10.05
22.09.05	Einarbeitung in Immunoinformatics (Buch von Lund et. al. lesen)	
22.09.05	In Paper: Constraints beschreiben	
04.10.05	New und delete nicht verwenden, anstattdessen STL-Containerklassen (z.B. vector) in den bisherigen Arrays ist zuviel Luft, anstattdessen 1. für num_of_sidechains aus Kommandozeile holen und dann die Struktur in einen Vektor (ein Element ist ein Sidechain) packen. 2. Bei den anderen arrays die Anzahl der Rotamere im pro sidechains genau festlegen (nicht über eine max. Anzahl gehen).	
06.10.05		

Weitere mögliche Themen für die Promotion:

1. docking: docking tools und BALL benutzen
2. Immunoinformatics
evtl. 1. und 2. kombinieren
3. Wirkstoffdesign

Computational Fluid Dynamics (CFD) Basics with Examples (engl./arab.) (2010 - 2015)

يتمحور هذا الكتاب حول اربع مواضيع رئيسية موزعة على 12 جزء .

المحور الأول و الذي لا بد منه هو مدخل إلى ديناميكيات الموائع ويشرح المعادلات الاساسية التي تحكم الموائع.

المحور الثاني يتحدث عن المعنى الفيزيائي والرياضي لتقنية الحسابات التي تستخدم في ميدان حساب معالم الموائع عن طريق الحاسوب.

المحور الثالث هو تعريف بطرق الحساب مثل طريقة العناصر المنتهية.

و أخيرا ، يأتي المحور الرابع الذي يمثل الجانب التطبيقي لديناميكيات الموائع الحاسوبية (د.م.ح.) و الذي يشرح بإسهاب أهم البرمجيات المجانية و المفتوحة المصدر المستخدمة في هذا المجال مع تطبيق عملي على سلوك الموائع في محطة طاقة كهربائية عن طريق حرق نفايات.

Mourad, Hamed
ELKerdi, Houda

Computational
Fluid
Dynamics

Computational Fluid Dynamics (CFD)

ديناميكيات الموائع الحاسوبية (د.م.ح.)

Samir Mourad
Fatima Hamed
Banan ELKerdi
Ahlam Houda

سمير مراد
فاطمة حامد
بنان الكردي
احلام هدى



September 2015



978-3-940871-17-6



AECENAR

Association for Economical and Technological Cooperation
in the Euro-Asian and North-African Region

www.aecenar.com

ديناميكيات الموائع الحسائية (د.م.ح.)

Computational Fluid Dynamics (CFD)

Samir Mourad سمير مراد

Fatima Hamed فاطمة حامد

Banan Kerdi بنان الكردي

Ahlam Houda احلام هدى

الاصدار الاول:

ذو الحجة 1436 الموافق 15 September 2015

ISBN 978-3-940871-17-6

تمهيد: بعض الميادين التي تُستخدم فيها ديناميكيات الموائع الحسابية (CFD) 227

1 مدخل الى ديناميكيات الموائع والغازات (FLUID AND GAS DYNAMICS) 229

- 1.1 تعريفات اساسية 229
- 1.2 نظام الوحدات 229
- 1.3 مضمون القسم الأول من الكتاب 229
- 1.4 الموائع (FLUIDS) 230
- 1.5 الكمية المتصلة 230
- 1.6 الكثافة 230
- 1.7 الكثافة النسبية 231
- 1.8 قانون الغاز المثالي (IDEAL GAS) 231
- 1.9 الجريان المستقر (STEADY FLOW) 231
- 1.10 اجريان المنتظم (UNIFORM FLOW) 231
- 1.11 خط الانسياب (STREAMLINE) 231
- 1.12 أبعاد السريان (DIMENSIONS OF FLOW) 231
- 1.13 الاجهاد (STRESS) 232
- 1.14 التدفق الصفائحي (LAMINAR FLOW) التدفق المضطرب (TURBULENT FLOW) 232
- 1.15 المنظومة وحجم التحكم عنصر مائع لا متناهي الصغر 232
- 1.16 الضغط المقياسي 233
- 1.17 القوة الجسمية والقوة السطحية 233
- 1.18 الاجهاد القصي 234

2 المعادلات الأساسية في ميكانيك الموائع (GOVERNING EQUATIONS OF FLUID DYNAMICS) 235

- 2.1 مدخل 235
- 2.1.1 متجه السريان 235
- 2.2 الاشتقاق الكبير (THE SUBSTANTIAL DERIVATE) 236
- 2.3 المعنى الفيزيائية من تباعد السرعة (DIVERGENCE OF VELOCITY) $\nabla \cdot \vec{V}$ 239
- 2.4 حفظ الكتلة (MASS CONSERVATION) 239
- 2.4.1 $continuity\ equation$ معادلة الاستمرارية () 241
- 2.5 حفظ الطاقة (ENERGY CONSERVATION) 242
- 2.6 حفظ كمية التحرك (MOMENTUM CONSERVATION) 246
- 2.7 تلخيص المعادلات الاساسية (GOVERNING EQUATIONS) لديناميك الموائع مع ملاحظات 246
- 2.7.1 $without\ considering\ chemical\ reactions$ دون النظر الى تفاعلات الكيميائية () $viscous\ flow$ معادلات السريان اللزجي () 246
- 2.7.2 $without\ considering\ chemical\ reactions$ (دون النظر الى تفاعلات الكيميائية) $inviscous\ flow$ معادلات السريان الالزجي () 250
- 2.7.3 تعليقات على المعادلات الاساسية 251
- 2.7.4 $boundary\ conditions$ للحالات الجدارية () 252

253	(CONSERVATION FORM) اشكال للمعادلات الاساسية ثلاث مع د.م.ح.: ملاحظات على الشكل التحفظي	2.8
3	سرايين لا انضغاطية ولا لزجية (INCOMPRESSIBLE INVISCID FLOWS) : طرق حسابية معتمدة على مؤشرات النبع و الدوامة	
	263 (SOURCE AND VORTEX PANEL METHODS)	
3.1	مدخل 263	
3.2	بعض الواجهة الاساسية لسريان لا انضغاطي و لا لزجي 263	
4	الخصائص الرياضية (MATHEMATICAL PROPERTIES) لمعادلات ديناميك الموائع (FLUID DYNAMIC EQUATIONS) 267	
4.1	مدخل 267	
4.2	بعض المعادلات التفاضلية الجزئية 268	
4.3	تصنيف (CLASSIFICATION) المعادلات التفاضلية الجزئية (PARTIAL DIFFERENTIAL EQ.S) 268	
4.4	السلوك العام للانواع المختلفة من المعادلات التفاضلية الجزئية و علاقتها بديناميات الموائع 275	
4.4.1	Hyperbolic Equations (المعادلات القطع الزائد) 275	
4.4.2	Parabolic Equations /معادلات القطع مكافئة 278	
4.4.3	elliptic equations (المعادلات القطع الناقص) 279	
4.4.4	بعض الملاحظات 281	
4.4.5	Well-Posed Problems طرح المشاكل بشكل جيد / 281	
4.4.6	المراجع 281	
5	تفريز لمعادلات التفاضلية الجزئية (DISCRETIZATION OF PDES) 282	
5.1	مدخل 282	
5.2	اشتقاق مقسومات لفرق محدودة ابتدائية (ELEMENTARY FINITE DIFFERENCE QUOTIENTS) 283	
5.3	جوانب اساسية لمعادلات الفرق المحدود (FINITE-DIFFERENCE EQUATIONS) 291	
5.3.1	تعليق عام 295	
5.4	أخطاء وتحليل الاستقرار (ERRORS AND AN ANALYSIS OF STABILITY) 296	
6	تحويلات الشبكة (GRID TRANSFORMATIONS) 307	
6.1	مدخل 307	
6.2	GENERAL TRANSFORMATION OF THE EQUATIONS 309	
6.3	METRICS AND JACOBIANS 314	
6.4	COORDINATE STRETCHING 316	
6.5	BOUNDARY-FITTED COORDINATE SYSTEMS 319	
6.6	ADAPTIVE GRID 330	
	335 REFERENCES	
7	طرق الفروق المحدودة الواضحة (EXPLICIT FINITE DIFFERENCE METHODS): بعض التطبيقات المحددة لسريان اللزج واللانزج	
	337	
7.1	مدخل (INTRODUCTION) 337	
7.2	طريقة لأكس واندروف (THE LAX- WENDROFF METHOD) 338	
7.3	MACCORMACK'S METHOD 343	
7.4	STABILITY CRITERION مقياس الاستقرار 346	

	7.5	تطبيقات مختارة من تقنيات المعتمدة على الزمن صريح (EXPLICIT TIME-DEPENDENT TECHNIQUE)	348
7.5.1		Non-equilibrium Nozzle Flows	349
7.5.2		Flow Field over a Supersonic Blunt Body	351
7.5.3		Internal Combustion Engine Flows	353
7.5.4		Supersonic Viscous Flow over a Rearward-Facing Step With Hydrogen Injection	355
7.5.5		Supersonic Viscous Flow over a Base	359
7.5.6		References	360
	8	الأحجام المحدودة (FINITE VOLUMES)	363
	8.1	نظرة عامة	363
	9	العناصر المحدودة:	368
	9.1	مدخل الى العناصر المحدودة (FINITE ELEMENTS)	368
	9.2	مدخل الي طريقة العناصر المنتهية (FEM) في ديناميكيات الموائع الحسابية (CFD)	372
	9.3	شرح طريقة العناصر المنتهية	372
	9.4	الصيغة المتحولية (VARIATIONAL FORMULATION)	374
	9.5	التقطيع (DISCRETIZATION)	375
	10	البرمجيات المستخدمة في النمذجة والمحاكاة	377
	10.1	تنسيق الملفات (FORMAT OF FILES)	377
	10.2	القيام بالنموذج	380
	10.3	تطبيق الشبكة على النموذج	381
	10.4	الحلال ELMER	383
	11	استخدام برامج لا تحتاج الى رخصة في ميدان ديناميكيات الموائع الحسابية	384
	11.1	تحسين سريان الماء داخل محطة طاقة تعمل على البخار ببرامج جاهزة	384
11.1.1		محطة طاقة عن طريق حرق النفايات لتبخير الماء قرب طرابلس الشام	384
11.1.2		مسألة تكبير حجم حتى تستخدم للتخلص من نفايات احدى المدن الكبرى وتغزيتها بالكهرباء	386
11.1.3		حل المسألة	386
11.1.4		مراجع	413
	11.2	انشاء برنامج لتحليل مسألة ما في ميدان ديناميكيات الموائع الحسابية (د.م.ح.)	413
11.2.1		OpenFOAM تحسين السريان في زاوية باستخدام	413
	12	لمحات عن الحرق الحسابي (NUMERICAL COMBUSTION)	426
	12.1	بعض ملاحظات بالنسبة لمحاكاة الحرق	426
12.1.1		(Flame Sheet Model) و (brutto reactions)	426
	12.2	اساسيات الحرق (BASICS OF COMBUSTION)	427
		مراجع	429
		ملحقات (APPENDICES)	430
		ملحق أ: مضمون كتاب "ميكانيك الموائع" لمحمد هاشم الصديق	430

431 ملحق ب: مضمون كتاب [FERZIGER, PERIC]

433 قاموس انجليزي - عربي



عن أبي هريرة رضي الله عنه قال قال رسول الله صلى الله عليه وسلم
إذا مات ابن آدم انقطع عمله إلا من ثلاث صدقة جارية أو علم ينتفع به أو ولد صالح يدعو له.

- رواه مسلم-1631

اللَّهُمَّ انفعني بما علمتني وعلمي ما ينفعني وارزقني علما تنفعني به.

المقدمة

الحمد لله رب العالمين والصلاة والسلام على اشرف المرسلين.

بإشراف المهندس سمير مراد وترجمة وكتابة مجموعة من العاملين (فاطمة حامد، بنان الكردي واحلام هدى) في مركز AECENAR للابحاث التطبيقية في راسنحاش - لبنان جاء هذا الكتاب المتميز.

هو كتاب متميز في موضوعه حيث أن حركة الموائع تدخل في صميم حياتنا اليومية، بدءاً من حركة السوائل في عروقنا وصولاً إلى حركة الطائرة أثناء سفرنا. كما استطاعت المجموعة العاملة على هذا الكتاب أن تحيط بموضوع ديناميكيات الموائع الحسابية من كل جوانبها إن من ناحية الدراسة النظرية التي اعتمدت على ترجمة وتلخيص مصادر عربية و أجنبية مشهود لها بالتعمق والشمولية في هذا الميدان أما من ناحية الدراسة التطبيقية عبر النمذجة و المحاكاة باستخدام برمجيات مجانية ومفتوحة المصدر على أنظمة تشغيل معقدة نسبياً. استطاع المؤلفون أن يقربوا مواضيع هذا الكتاب للقارئ- وهي مواضيع يتهيبها ويجهلها الكثير من الناس- تقريباً لم يفقدها العمق، وأن يوضح غوامضها بأسلوب مشرق إشراقاً لا يفترّط في دقة العلم وصرامته.

نسأل الله تعالى ان يجعل هذا العمل في ميزان حسنات كل من شارك فيه وان يجعله علم ينتفع به بعد موتنا.

م. سمير مراد

مدير مركز AECENAR

تمهيد: بعض الميادين التي تُستخدم فيها ديناميكيات الموائع الحاسوبية (CFD)

لا بد أن تُستخدم المحاكاة في ميادين صعبة ومكلفة التطبيق عمليا. إليكم بعض مجالات تطبيقاتها:

- علم الفلك
- محارق للنفايات: المحاكاة CFD، تكون لمعرفة توزيع درجة الحرارة في المحرقة

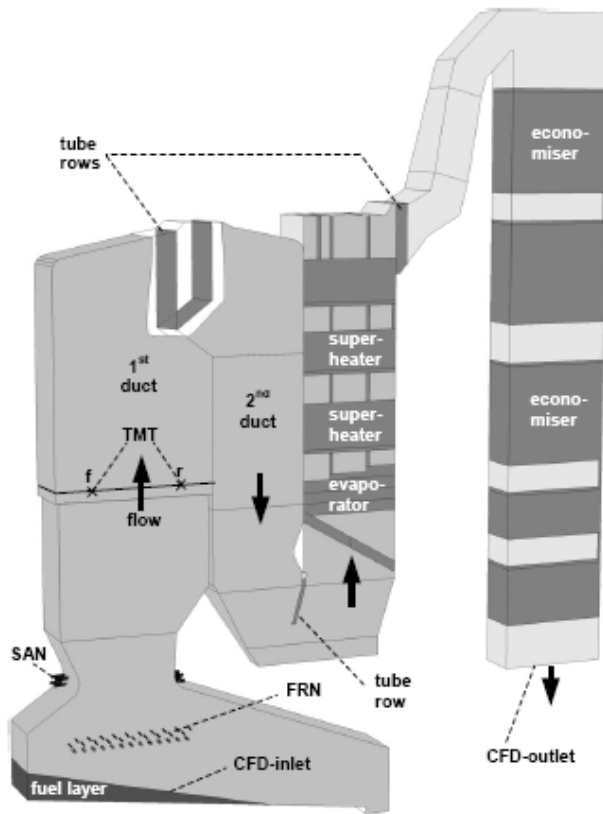
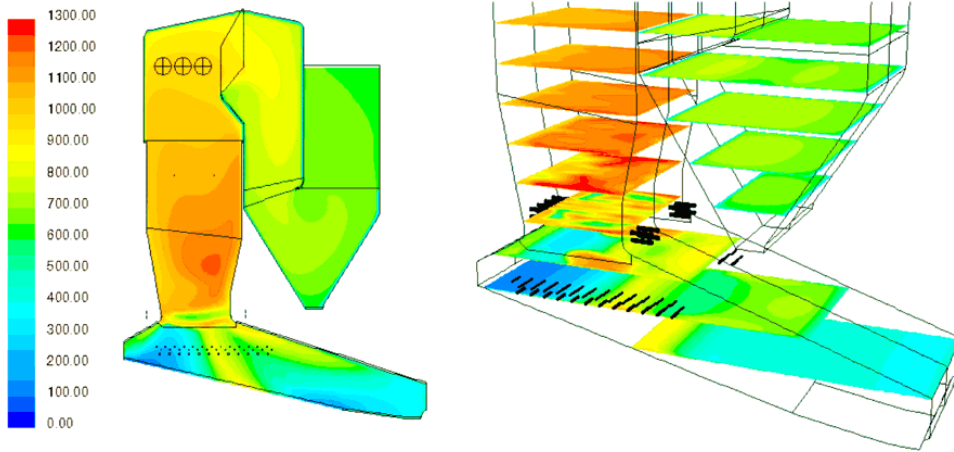


Figure 1: CFD model of the biomass furnace and boiler
Explanations: modeled tube bundles and rows are pictured dark gray; SAN...secondary air nozzles, FRN...flue gas recirculation nozzles, TMT... suction pyrometer temperature measurement traverses

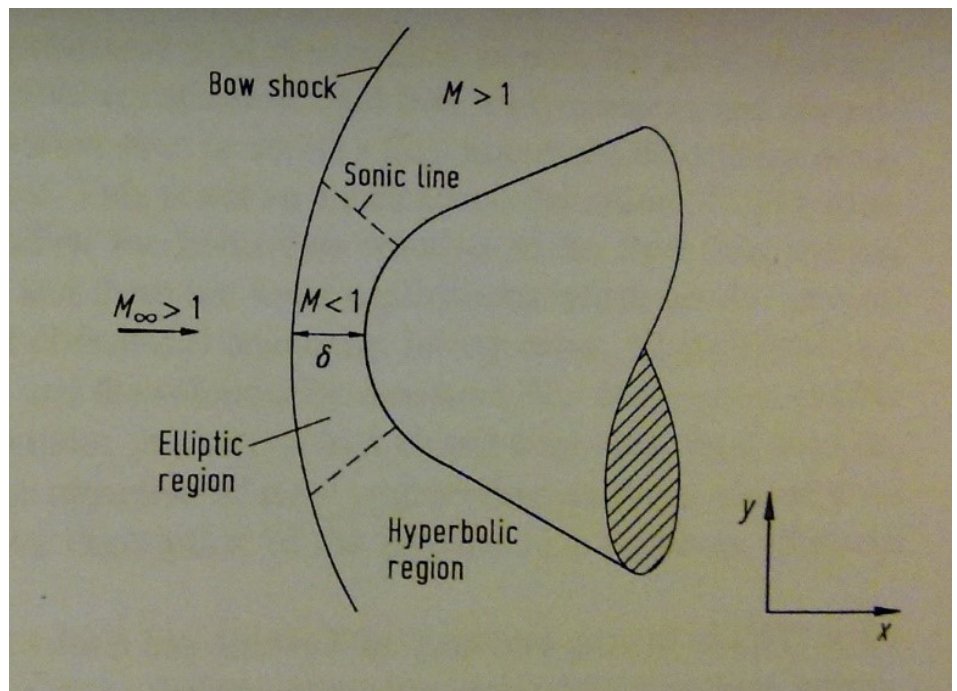
From: Scharler et. al. 2004, Advanced CFD analysis of large fixed bed biomass boilers ..., 2nd World Conf...., Rome, 2004



Isosflächen der Rauchgastemperatur [°C] in der Symmetrieebene der Feuerung (links) und in horizontalen Schnittebenen (rechts). Aus: <http://www.bios-bioenergy.at/de/cfd-simulationen.html>

• محارق صواريخ

• المركبات الفضائية



17 fluid and gas dynamics مدخل الى ديناميكيات الموائع والغازات)

ديناميك الغازات (gas dynamics) هو علم سريان الغازات أو أي مزيج من الغازات. أهم تطبيق لهذا العلم هو دراسة حركة الهواء لاستعماله في دراسات آروديناميك الطيران (plane aerodynamics) و آروديناميك لمحركات الطيران.

17.1 تعريفات اساسية

ميكانيكا الموائع (Fluid Mechanics) هو تخصص فرعي من ميكانيكا المواد المتصلة (Mechanics Continuum) وهو معني أساساً بالموائع، التي هي السوائل والغازات، ويدرس هذا التخصص السلوك الفيزيائي الظاهر الكلي لهذه المواد، ويمكن تقسيمه من ناحية إلى إستاتيكا الموائع - أي في حالة عدم الحركة، أو ديناميك الموائع أي في حالة الحركة، ويندرج تحتها تخصصات أخرى معينة، فهناك الديناميكيات الهوائية (أيروديناميك) والديناميكيات المائية (هيدروديناميك).

يسعى هذا التخصص إلى تحديد القيم الفيزيائية الخاصة بالموائع، مثل السرعة، الضغط، الكثافة، درجة الحرارة، اللزوجة ومعدل التدفق، وقد ظهرت تطبيقات حسابية حديثة لإيجاد حلول للمسائل المتصلة بميكانيكا الموائع، ويسمى التخصص المعني بذلك ديناميكيات الموائع الحسابية (بالإنجليزية: Computational FluidDynamics) (CFD).

17.2 نظام الوحدات

النظام المستخدم هنا هو النظام العالمي للوحدات (SI).

القائمة أدناه تبين وحداته الأساسية:

الطول	الكتلة	الزمن	درجة الحرارة	القوة	الطاقة	القدرة	الضغط
m	kg	s	K	N	J	W	Pa
متر	كيلو غرام	ثانية	كلفن	نيوتن	جول	وات	باسكال

17.3 مضمون القسم الأول من الكتاب

في الجزء الاول من هذا الكتاب ستناول ان شاء الله التالي:

(a) تلخيص لميكانيكا الموائع (بالإنجليزية: Fluid Mechanics)

(b) مدخل ملخص للتحليل العددي (بالإنجليزية: Numerics / Numerical Computation)

(c) اساليب ديناميكيات الموائع الحاسوبية (بالإنجليزية: Computational Fluid Dynamics)

يوجد مرجع باللغة العربية بالغ الأهمية في اختصاص ميكانيكا الموائع و هو كتاب ميكانيك الموائع من محمد هاشم صديق⁴.

17.4 الموائع (fluids)

الموائع جمع لكلمة مائع (fluid) تشكل مجموعة من أطوار المادة، وهي أية مادة قابلة للانسياب تحت تأثير إجهاد القص وتأخذ شكل الإناء الحاوي لها. تتضمن الموائع كلاً من السوائل، الغازات، البلازما وأحياناً الأصلاب اللدنة plastic solids. تصنف الموائع عادة إلى:

- **موائع قابلة للانضغاط (compressible fluids)** وهي الموائع التي تتغير كثافتها بتغير الضغط الواقع عليها مثل الغازات. و يُسمى أيضاً السريان الانضغاطي.
- **موائع غير قابلة للانضغاط (incompressible fluids)** وهي الموائع التي لا تتغير كثافتها بتغير الضغط الواقع عليها مثل السوائل. و يُسمى أيضاً السريان اللانضغاطي.
- **موائع نيوتنية: المائع النيوتني** هو مائع تكون فيه علاقة الإجهاد (stress) - الانفعال (تشوه المواد نتيجة الإجهاد) علاقة خطية أي على شكل مستقيم يمر من مبدأ الإحداثيات، ويعرف اسم ثابت التناسب باللزوجة. سمي هذا المائع على اسم العالم اسحق نيوتن.
- **موائع غير نيوتنية: مائع لا نيوتوني** هو مائع لا يمكن وصف جريانه باستخدام ثابت اللزوجة. تعتبر أغلب محاليل البوليمرات و**البوليمرات الذائبة** من الموائع اللانويوتونية والكثير من السوائل الشائعة مثل الكتشب، ذائب النشا، الدم و**الشامبو**.

17.5 الكمية المتصلة

يمكن اعتبار المائع كمية متصلة إذا كانت أصغر مسافة في التحليل أكبر من متوسط المسار الحر للجزيئات.

$$L \gg \lambda$$

17.6 الكثافة

باعتبار أن الحجم V_0 هو مكعب أصغر مسافة ترد عي التحليل وتستوفي شرط الكمية المتصلة فإن الكثافة ρ تعرف كما يلي: $\rho =$

$$\lim_{\Delta V \rightarrow V_0} \frac{\Delta m}{\Delta V}$$

⁴ [Siddiq]

حيث m الكتلة بالكيلوغرام و V الحجم بالتر المكعب و وحدة الكثافة هي $\frac{kg}{m^3}$.

17.7 الكثافة النسبية

هي كثافة المادة منسوبة الى الكثافة المعيارية للماء، و هي $1000kg / m^3$

$$S = \rho / \rho_0$$

17.8 قانون الغاز المثالي (ideal gas)

$$p = R\rho T$$

حيث يرتبط الضغط المطلق للغاز p بالدرجة المطلقة للحرارة والكثافة ρ .

R ثابت الغاز و قيمته للهواء $287 J/(K kg)$.

17.9 الجريان المستقر (steady flow)

هو الجريان الذي لا تتغير صفاته مع الزمن عند أي موضع محدد.

17.10 اجريان المنتظم (uniform flow)

17.10

يوصف الجريان بأنه منتظم عند مقطع إذا كانت قيمة كل من خواصه ثابتة في كل نقاط المقطع.

17.11 خط الانسياب (streamline)

17.11

يُعرف خط الانسياب بأنه الخط الذي تشكل المماسات له في كل أجزائه اتجاهات السرعة في وقت محدد.

17.12 أبعاد السريان (dimensions of flow)

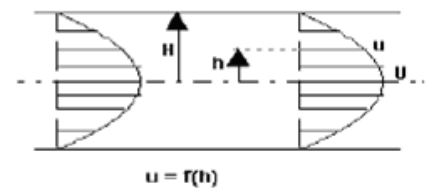
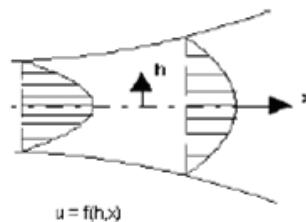
17.12

يوصف السريان بأنه أحادي، ثنائي او ثلاثي البعد بناءً على العدد الأدنى من الإحداثيات المكانية التي يمكن ان يوصف بها.

الشكل 1.2 يعطي مثالاً لسريان احادي البعد وآخر ثنائي البعد.

سريان ثنائي البعد

سريان احادي البعد



الشكل 1.2

الإجهاد (stress)

17.13

$$\sigma = \lim_{\Delta A \rightarrow 0} \frac{\Delta F}{\Delta A} \quad \text{الإجهاد هو القوة السطحية العاملة على وحدة مساحة}$$

و للإجهاد مُركبين أحدهما عمودي والآخر مماسة $\underline{\sigma} = \underline{\sigma}_n + \underline{\sigma}_t$. ويفضّل في ميكانيكا الموائع استخدام تعبير الضغط p في

$$\underline{\sigma}_n = -p\underline{n} \quad \text{الاتجاه المتعامد حيث}$$

$$\underline{\sigma}_t = \underline{\tau} \quad \text{و يستخدم تعبير الإجهاد القصي } \tau \text{ في الاتجاه المماس حيث}$$

$$\underline{\sigma} = -p\underline{n} + \underline{\tau} \quad \text{وبذلك}$$

التدفق الصفائحي (laminar flow) التدفق المضطرب

17.14

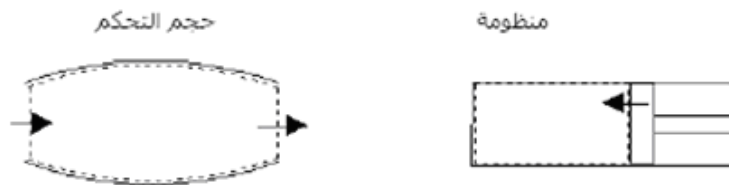
(turbulent flow)

يتصف التدفق الصفائحي بثبات الانسيابية بحيث يمكن اعتبار طبقاته تنزلق فوق بعضها البعض في شكل صفائح او رقائق، بينما يتصف التدفق المضطرب بالعنف و الاضطراب.

و يمكن إثبات أن التحول من الحالة الصفائحية إلى الحالة المضطربة عند معدل سريان ثابت يحدث بزيادة السرعة او زيادة قطر الجسم الذي يجري فيه المائع (diameter) او إنقاص اللزوجة. ويجمع المتغيرات الثلاثة مقدار لأبعدي يعرف بعدد رينولدز (Reynolds number) Re يحكم التحول المذكور. ويحدث هذا التحول للسريان في الانابيب في الفسحة $4000 \geq Re \geq 2000$. ويسمى عدد رينولدز الذي يحدث عنده التحول عدد رينولدز الحساس Re_c .

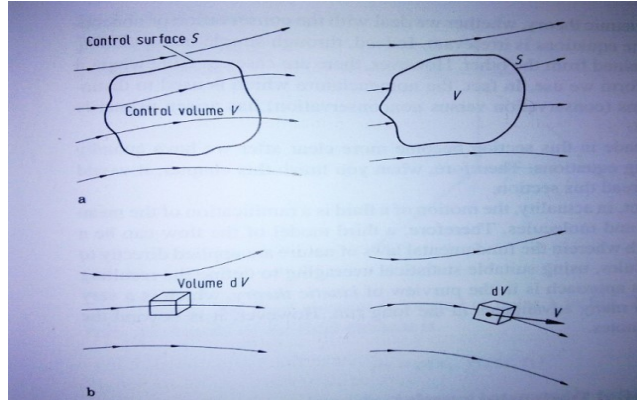
المنظومة وحجم التحكم عنصر مانع لا متناهي الصغر

17.15



الشكل 1.3

المنظومة (system) معنية بكمية محددة من المادة، يحدها عن بقية المائع جدار تخيلي او حقيقي و يمكن ان يتغير موقعه وشكله مع الوقت. حجم التحكم (control volume) بالمنطقة محدد وثابت في المكان، ويمكن ان تتغير المادة داخل حجم التحكم مع الزمن. هذا الحجم مرسوم في الشكل (1.3.1 a) على اليسار ولكن ايضاً يمكن ان ننظر الى حجم التحكم كما هو في الشكل (1.3.1 b) على اليمين و هو حجم التحكم الذي يتحرك مع السريان.



الشكل (1.3.1 a and b)
([Wendt 2009], Fig. 2.1)

الشكل (1.3.1 a), الجهة اليسرى: حجم التحكم المحدود V ؛ مساحة التحكم المحدودة S ثابتة في المكان.

معادلات الموائع التي نحصل عليها مباشرة بتطبيق قواعد الفيزياء الأساسية الى حجم التحكم المحدود الذي يكون في شكل تكاملي. هذه الاشكال التكاملية من المعادلة الاساسية تستطيع ان تُعالج بطريقة غير مباشرة للحصول على المعادلات التفاضلية الجزئية. المعادلات التي تم الحصول عليها، سواء في شكل تكاملي أو تفاضلي جزئي، تسمى الشكل التحفظي (*conservation form*) للمعادلات الأساسية. المعادلات التي تم الحصول عليها عبر حجم التحكم المحدود تتحرك مع المائع (الشكل 1.3.1 الجانب الأيمن)، سواء في شكل تكاملي أو تفاضلي جزئي، ويطلق عليه الشكل الغير التحفظي (*non-conservation form*) من المعادلات الأساسية. إذا أخذنا في الاعتبار عنصر مائع متناهي الصغر، فهو ثابت في المساحة (الشكل 1.3.1 b، الجانب الأيسر)، يمكن أن نشق مباشرة المعادلات التفاضلية الجزئية. هذا هو أيضاً الشكل التحفظي.

إذا أخذنا في الاعتبار عنصر مائع لامتناهي الصغر، يتحرك في المساحة (الشكل 1.3.1 b، الجانب الأيمن)، يمكن أن نشق بشكل مباشر المعادلات التفاضلية الجزئية. هذا هو أيضاً النموذج الغير تحفظي. من الناحية النظرية الأيرودينامية العامة، سواء نحن نتعامل مع أشكال تحفظية أو غير تحفظية المعادلات هي سواء. ومع ذلك، هناك حالات في ال CFD حيث نهتم بأي شكل نستخدم.

الضغط المقياسي

17.16

الضغط المقياسي = الضغط المطلق - الضغط الجوي

القوة الجسمية والقوة السطحية

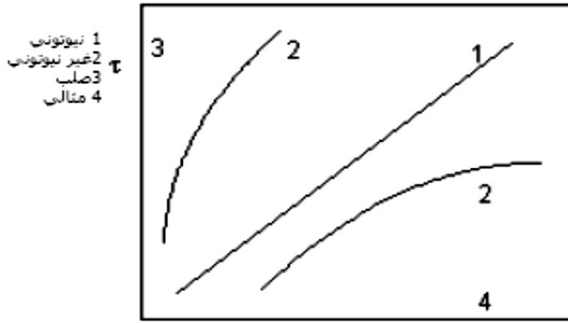
17.17

القوة الجسمية هي التي تنشأ عن كتلة الجسم مثل قوة الجاذبية والقوة السطحية و هي تلك التي تعمل على سطح المادة وتنحصر في الضغط والقص.

تنسب الى نيوتن العلاقة النظرية بين الاجهاد القصي τ وممال السرعة في الاتجاه المتعامد $\frac{\partial u}{\partial y}$ للسريان الصفائحي و هي:

تُعرف اللزوجة الكينماتية ν كما يلي: $\nu = \frac{\mu}{\rho}$ ووحدتها m^2/s .

$$\tau = \mu \frac{\partial u}{\partial y} \dots\dots\dots(1.3)$$



1 نيوتوني
2 غير نيوتوني
3 صلب
4 مثالي

الشكل (1.4) du/dy

وقد أجريت تجارب للتحقق من المعادلة معملياً و عُلم أنها صحيحة لمعظم الموائع المستخدمة في التطبيقات الهندسية مثل الماء والهواء و الوقود النقطي. و سُمي ثابت المعادلة μ باللزوجة أو اللزوجة المطلقة أو اللزوجة الحركية، ووحدتها Pa.s. وتعرف الموائع التي تستجيب لهذه العلاقة عند درجة حرارة ثابتة بالموائع **النيوتونية** - الشكل (1.4).

تُسمى فصيلة الموائع التي لا

تُعطي علاقة خطية بين القص وممال السرعة موائع **لانبيوتونية**. أمثلة لها البوية و النفط الشمعي.

تؤثر درجة الحرارة في قيمة اللزوجة حيث تنقص مع ازدياد الحرارة للسوائل وتزيد مع ازدياد الحرارة للغازات .

18 المعادلات الأساسية في ميكانيك الموائع (Governing Equations of Fluid Dynamics)

التالي منبني على [صديق]، فصل 2 و [Anderson 1991].

18.1 مدخل

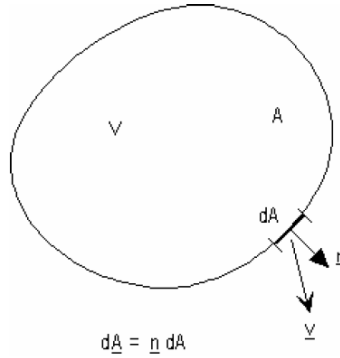
الاساس في CFD هو المعادلات الأساسية في ميكانيك الموائع و هي معادلات الحفظ الثلاث:

حفظ الكتلة (mass conservation) وحفظ الطاقة (energy conservation) وحفظ كمية الحركة (momentum conservation)

(. و قدم لذلك بتعريف متجه السريان الذي يشكل عنصراً مشتركاً في كل معادلات الحفظ.

18.1.1 متجه السريان

الشكل 2.1



الحجم التحكمي الموضح في الشكل (2.1) حجمه V و مساحته A . بالتركيز على المساحة التفاضلية dA فان الكتلة الخارجة عبرها

هي dm في الوقت dt ليصبح معدل السريان \dot{m} . سرعة السريان في الموضع هي المتجه \underline{v} بزاوية α مع المتجه أحادي الطول \underline{n}

المتعامد على المساحة dA حيث $d\underline{A} = \underline{n} dA$

$$dm = \rho dV = \rho \underline{v} \cdot d\underline{A}$$

\dot{m} = معدل سريان الكتلة عبر كل السطح A هو:

$$(2.1) \dots \dot{m} = \iint_A \rho \underline{v} \cdot d\underline{A}$$

نعرف متجه سريان الكتلة كما يلي:

$$\rho \underline{v} = \text{متجه سريان الكتلة} = (\text{متجه السرعة}) (\text{الكتلة في وحدة حجمية})$$

وبالمثل:

$$\rho\left(e + \frac{v^2}{2} + gz\right)\underline{v} = (\text{متجه السرعة})(\text{الطاقة في وحدة حجمية})$$

وبالمثل:

متجه سريان كمية التحرك = (متجه السرعة)(كمية التحرك في وحدة حجمية) = $\rho u \underline{v}, \rho v \underline{v}, \rho w \underline{v}$ في الاتجاهات x, y, z على التوالي.

و بذلك فان معدل سريان الطاقة عبر السطح A =

$$(2.2) \dots \iint_A \rho\left(e + \frac{v^2}{2} + gz\right)\underline{v} \cdot d\underline{A}$$

و معدل سريان كمية التحرك عبر السطح A =

$$(2.3) \dots \iint_A \rho \underline{v} (\underline{v} \cdot d\underline{A})$$

18.2 الاشتقاق الكبير (The Substantial Derivate)

As a model for the flow, we will adopt the picture shown at the right of Fig. 1.3.1 (b).

Namely that of an **infinitesimally small fluid element moving with the flow**. The motion of the fluid element is shown in detail in Fig. 2.2.1.

Here, the fluid element is moving through Cartesian space. The unit vectors along the x, y, z axis are $\vec{i}, \vec{j}, \vec{k}$.

The vector velocity field in this Cartesian space is given by

$$\vec{V} = u\vec{i} + v\vec{j} + w\vec{k}$$

Where the components of velocity are given respectively by

$$\begin{aligned} u &= u(x, y, z, t) \\ v &= v(x, y, z, t) \\ w &= w(x, y, z, t) \end{aligned}$$

Note that we are considering in general an *unsteady flow*, where $u, v,$ and w are functions of both (space and time, t). In

كنموذج للسريان، سوف نعتمد على الصورة المعروضة على يمين الشكل 1.3.1 (b). ألا وهو عنصر من الموائع المتناهي الصغر تتحرك مع السريان.

حركة عنصر السريان معروضة بالتفصيل في الشكل 2.2.1.

هنا، العنصر المائع يتحرك عبر الفضاء الديكارتي Cartesian space.

وحدة المتجهات على طول المحور x, y, z ، تكون $\vec{i}, \vec{j}, \vec{k}$.

يتم إعطاء مجال متجهات السرعة في هذا المجال من قبل ديكارت

Cartesian space عبر:

$$\vec{V} = u\vec{i} + v\vec{j} + w\vec{k}$$

حيث يتم إعطاء مكونات السرعة على التوالي

$$\begin{aligned} u &= u(x, y, z, t) \\ v &= v(x, y, z, t) \\ w &= w(x, y, z, t) \end{aligned}$$

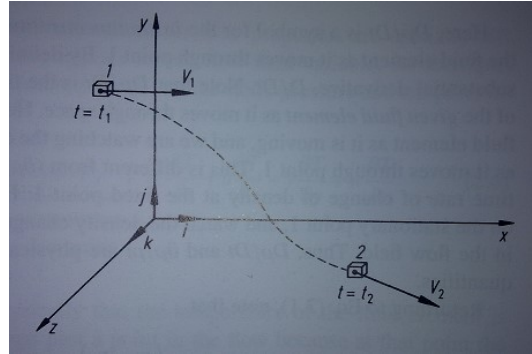
علما أننا نأخذ بعين الاعتبار بالعموم سريان غير رتيب، حيث u, v و

w هي وظائف المكان والزمان t . على حدٍ سواء، بالإضافة إلى ذلك

addition the scalar density field is given by $\rho = \rho(x, y, z, t)$.

هو إعطاء مقدار الكثافة العددية من قبل

$$\rho = \rho(x, y, z, t)$$



الشكل (2.2.1)

([Wendt 2009], Fig. 2.2)

At the time t_1 the fluid element is located at point 1 in Fig. 2.2.1. At this point and time, the density of the fluid element is

$$\rho_1 = \rho(x_1, y_1, z_1, t_1)$$

في الوقت t_1 حيث يكون العنصر المائع موجود في النقطة 1 على الشكل 2.2.1. عند هذه النقطة والوقت ، وكثافة العنصر المائع

$$\rho_1 = \rho(x_1, y_1, z_1, t_1)$$

At a later time t_2 the fluid element has moved to the point 2 where the density is

$$\rho_2 = \rho(x_2, y_2, z_2, t_2)$$

في وقت لاحق t_2 انتقل العنصر المائع إلى نقطة 2 حيث الكثافة هي

$$\rho_2 = \rho(x_2, y_2, z_2, t_2)$$

Since $\rho = \rho(x, y, z, t)$, we can expand this function in a Taylor's series about point 1 as follows:

بما ان $\rho = \rho(x, y, z, t)$ ، يمكننا توسيع نطاق هذه المهمة في سلسلة تايلور Taylor's series حول النقطة 1 على النحو التالي:

$$\rho_2 = \rho_1 + \left(\frac{\partial \rho}{\partial x}\right)_1 (x_2 - x_1) + \left(\frac{\partial \rho}{\partial y}\right)_1 (y_2 - y_1) + \left(\frac{\partial \rho}{\partial z}\right)_1 (z_2 - z_1) + \left(\frac{\partial \rho}{\partial t}\right)_1 (t_2 - t_1) + (\text{higher order terms})$$

With ignoring the higher order terms we obtain

مع تجاهل مصطلحات الترتيبية الأعلى لكي نحصل على

$$\frac{\rho_2 - \rho_1}{t_2 - t_1} = \left(\frac{\partial \rho}{\partial x}\right)_1 \left(\frac{x_2 - x_1}{t_2 - t_1}\right) + \left(\frac{\partial \rho}{\partial y}\right)_1 \left(\frac{y_2 - y_1}{t_2 - t_1}\right) + \left(\frac{\partial \rho}{\partial z}\right)_1 \left(\frac{z_2 - z_1}{t_2 - t_1}\right) + \left(\frac{\partial \rho}{\partial t}\right)_1 \quad (2.1.1)$$

Eq. (2.1.1) is physically the average time-rate-of-change in density of the fluid element as it moves from point 1 to point 2. In the limit, as t_2 approaches t_1 , this term becomes

المعادلة (2.1.1) فيزيائياً هي متوسط الوقت لمعدل التغير في كثافة العنصر المائع وهي تنتقل من النقطة 1 إلى النقطة 2. في الحد، t_2 مثل t_1 ، يصبح هذا المصطلح

$$\lim_{t_2 \rightarrow t_1} \left(\frac{\rho_2 - \rho_1}{t_2 - t_1}\right) \equiv \frac{D\rho}{Dt}$$

$\frac{D\rho}{Dt}$ Is a symbol for the *instantaneous* time rate of change of density. هذا هو رمز لحظية معدل الوقت لتغيير الكثافة. وفقاً للتعريف ، هذا ما يسمى رمز الاشتقاق الكبير ، D/Dt .

By definition, this symbol is called the substantial derivate, D/Dt .

هو معدل الوقت لتغيير كثافة عنصر مائع معين. وثبتت أعيننا مع

$\frac{D\rho}{Dt}$ is the time rate of change of density of the *given fluid element*. Our eyes are locked with the fluid element, not with the point in the space. So $\frac{D\rho}{Dt}$ is different

العنصر المائع، وليس مع نقطة في الفضاء.

physically and numerically from $\left(\frac{\partial\rho}{\partial t}\right)_1$

كذلك $\frac{D\rho}{Dt}$ تختلف فيزيائياً وعددياً من $\left(\frac{\partial\rho}{\partial t}\right)_1$ التي هي فيزيائياً

المعدل الزمني لتغير الكثافة في نقطة ثابتة 1.

which is physically the time rate of change of density at the fixed point 1.

بالعودة الى المعادلة (2.1.1) ، نلاحظ أنَّ

Returning to Eq. (2.1.1), note that

$$\lim_{t_2 \rightarrow t_1} \left(\frac{x_2 - x_1}{t_2 - t_1} \right) \equiv u$$

$$\lim_{t_2 \rightarrow t_1} \left(\frac{y_2 - y_1}{t_2 - t_1} \right) \equiv v$$

$$\lim_{t_2 \rightarrow t_1} \left(\frac{z_2 - z_1}{t_2 - t_1} \right) \equiv w$$

Thus, taking the limit of Eq.(2.1.1) as $t_2 \rightarrow t_1$, we obtain

وهكذا، بأخذ الحد للمعادلة (2.1.1) عندما $t_2 \rightarrow t_1$ ، نحصل

$$\frac{D\rho}{Dt} \equiv \frac{\partial\rho}{\partial t} + u \frac{\partial\rho}{\partial x} + v \frac{\partial\rho}{\partial y} + w \frac{\partial\rho}{\partial z} \quad (2.1.2)$$

From (2.1.2) we obtain an expression for the substantial derivate in Cartesian coordinates

من (2.1.2) نحصل على التعبير عن الاشتقاق الكبير في الإحداثيات

الديكارتية

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + w \frac{\partial}{\partial z} \quad (2.1.3)$$

In cartesian coordinates the vector operator ∇ is defined as

في الإحداثيات الديكارتية يتم تعريف عامل المتجه

▽

$$\nabla \equiv \vec{i} \frac{\partial}{\partial x} + \vec{j} \frac{\partial}{\partial y} + \vec{k} \frac{\partial}{\partial z} \quad (2.1.4)$$

Hence Eq.(2.1.3) can be written as

وبالتالي يمكن أن تكون المعادلة (2.1.3) مكتوبة

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + (\vec{V} \cdot \nabla) \quad (2.1.5)$$

Eq.(2.1.5) represents a definition of the substantial derivative operator in vector notation; thus it is valid for any coordinate system.

المعادلة (2.1.5) تمثل تعريف عامل الاشتقاق الكبير في تدوين المتجهات، وبالتالي يصح لأي نظام احداثيات.

$\frac{\partial}{\partial t}$ is called the *local derivative* which is physically the time rate of change at a fixed point; $\vec{V} \cdot \nabla$ is called the *consecutive derivative*, which is physically the time rate of change due to the movement of the fluid element from one location to another in the flow field where the flow properties are spatially different. The substantial derivative applies to any flow-field variable, for example, Dp/Dt , DT/Dt , ..., where p and T are static pressure and temperature respectively.

$\frac{\partial}{\partial t}$ تسمى المشتقات المحلية التي هي فعليا المعدل الزمني للتغيير في نقطة ثابتة، ويسمى الاشتقاق المتتالي، وهو فعليا معدل الوقت للتغيير بسبب حركة العنصر السائل من مكان إلى آخر في حقل السريان حيث خصائص السريان هي مختلفة مكانياً. الاشتقاق الكبير ينطبق على أي متغير في ميدان التدفق، على سبيل المثال، Dp/Dt ، DT/Dt ، حيث p و T هي الضغط ودرجة الحرارة على التوالي.

The substantial derivative is essentially the same as the total differential from calculus. Therefore, the substantial derivative is nothing more than a total derivative with respect to time.

الاشتقاق الكبير هو اساساً نفس مجموع التفاضل من حساب التفاضل و التكامل. لذلك، الاشتقاق الكبير ليس أكثر من مجرد مجموع المشتقات مع احترام الوقت.

18.3 المعنى الفيزيائية من تباعد السرعة (divergence of velocity) $\nabla \cdot \vec{V}$

تباعد السرعة (divergence of velocity) $\nabla \cdot \vec{V}$

$$\nabla \cdot \vec{V} = \frac{1}{\delta V} \frac{D(\delta V)}{Dt} \dots \dots \dots (2.4)$$

$\nabla \cdot \vec{V}$ is physically the time rate of change of the volume of a moving fluid element, per unit volume.

$\nabla \cdot \vec{V}$ هو التغيير الزمني لحجم التحكمي (control volume) من عضو مائع (fluid element) جارٍ (moving) و ذلك حسب الحجم التحكمي (per control volume)

18.4 حفظ الكتلة (mass conservation)

صيغة قانون حفظ الكتلة مطبقاً على سريان المائع:

"معدل تراكم الكتلة داخل الحجم التحكمي مضافاً إليه خالص معدل سريان الكتلة إلى خارج الحجم التحكمي يساوي صفر.

$$\iiint_V \rho dV = \text{الكتلة الكلية داخل الحجم التحكمي}$$

معدل ازدياد الكتلة داخل الحجم التحكمي (control volume):

$$\frac{\partial}{\partial t} \iiint_V \rho dV = \iiint_V \frac{\partial \rho}{\partial t} dV$$

لأن حدود التكامل لا تعتمد على الوقت.

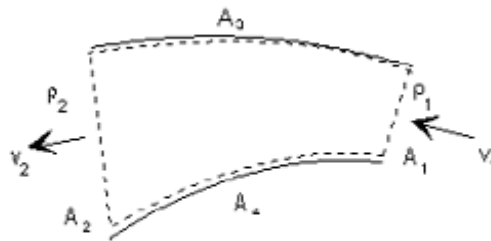
من المعادلة (2.1) خالص سريان الكتلة إلى خارج الحجم التحكمي

$$= \iint_A \rho \underline{v} \cdot d\underline{A}$$

$$\iiint_V \frac{\partial \rho}{\partial t} dV + \iint_A \rho \underline{v} \cdot d\underline{A} = 0 \quad (2.4)$$

المادلة (2.4) هي معادلة حفظ الكتلة في الصورة التكاملية (integral form).

تطبيق على سريان احادي البعد (الشكل 2.2)



الشكل 2.2

الحد الاول في المعادلة (2.4)

يساوي صفر نسبةً لرتابة السريان.

السطحان (3) و (4) لا يعتبرهما

كتلة. ولذلك يصير فيهما تكامل

الحد الثاني و معادلة الكتلة صفراً.

تختزل الكتلة بذلك الى الصورة:

$$\iint_{A_1} \rho v_1 \cdot dA_1 + \iint_{A_2} \rho v_2 \cdot dA_2 = 0$$

وبلا حظة ان المتجه \underline{A} يتجه إلى خارج الحجم التحكمي

$$- \iint_{A_1} \rho v_1 \cdot dA_1 + \iint_{A_2} \rho v_2 \cdot dA_2 = 0$$

$$-\rho_1 v_1 A_1 + \rho_2 v_2 A_2 = 0$$

$$\rho v A = \text{ثابت} \dots \dots \dots (2.5)$$

18.4.1 معادلة الاستمرارية (continuity equation)

يطلق هذا الاسم عامةً على معادلة حفظ الكتلة في صورتها التفاضلية. بدءاً من المعادلة (2.4) يمكن تحويل الحد الثاني من صورة التكامل السطحي الى صورة التكامل الحجمي باستخدام نظرية التباعد (divergence theorem). للحصول على المعادلات الأساسية لحركة الموائع، يجب دائماً اتباع الطريقة التالية :

- اختيار المبادئ الفيزيائية الأساسية المناسبة من الفيزياء
- تطبيق هذه المبادئ الفيزيائية لنموذج سريان مناسب.
- من هذا التطبيق، استخراج المعادلات الرياضية التي تتضمن المبادئ الفيزيائية.

لذا، في حالتنا الفيزيائية المبدأ هو : "الكتلة هي So, in our case the physical principle is: "Mass is Conserved". المحفوظة" ("Mass is Conserved").

$$\iiint_V \frac{\partial \rho}{\partial t} dV + \iiint_V (\nabla \cdot \rho \underline{v}) dV = 0$$

$$\iiint_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \underline{v} \right) dV = 0$$

تبعاً لقوانين التكامل تكون قيمة المكامل صفرًا إذا كانت قيمة التكامل صفرًا و كانت حدود التكامل اختياريةً.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \underline{v} = 0 \dots \dots \dots (2.6a)$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) + \frac{\partial}{\partial z}(\rho w) = 0 \dots \dots \dots (2.6b)$$

حيث w, v, u هي مركبات السرعة في الاتجاهات x, y, z . و في حال ان السريان لا انضغاطي (incompressible flow)

$$\dots \dots \dots \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 (2.7)$$

Divergence Theoreme:

18.5 حفظ الطاقة (energy conservation)

1 إذا كانت $f = f(x, y, z)$ فان مجال f هو المتجه:

$$\nabla f = \frac{\partial f}{\partial x} \underline{i} + \frac{\partial f}{\partial y} \underline{j} + \frac{\partial f}{\partial z} \underline{k} \dots\dots\dots(1)$$

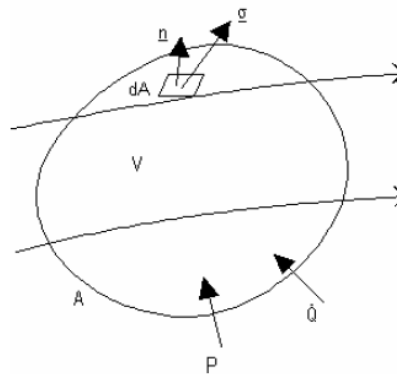
2 إذا كانت ϕ متجه ذا مركبات مطلقة ϕ_x و ϕ_y و ϕ_z في الاتجاهات x و y و z ، على التوالي ، فان التباعد لـ ϕ

$$\nabla \cdot \phi = \frac{\partial \phi_x}{\partial x} + \frac{\partial \phi_y}{\partial y} + \frac{\partial \phi_z}{\partial z} \dots\dots\dots(2)$$

3 تربط نظرية التباعد التكامل الحجمي و التكامل السطحي بالعلاقة

$$\iiint_V (\nabla \cdot \phi) dV = \iint_A \phi \cdot d\underline{A} \dots\dots\dots(3)$$

الشكل 2.5



تستمد معادلة حفظ الطاقة من القانون الاول للحركية الحرارية مطبقاً على حجم تحطمي:

"معدل تراكم الطاقة داخل الحجم التحكمي مضافاً اليه خالص معدل سريان الطاقة الى خارج الحجم التحطمي بانتقال الكتلة يعادل القدرة المبذولة على المائع داخل الحجم التحكمي مضافاً اليها خالص معدل سريان الحرارة إلى داخل الحجم التحكمي".

$$\frac{\partial}{\partial t} \iiint_V \rho \left(e + \frac{v^2}{2} + gz \right) dV + \iint_A \rho \left(e + \frac{v^2}{2} + gz \right) \underline{v} \cdot d\underline{A} = \iint_A (\underline{\sigma} \cdot \underline{v}) dA + P + \dot{Q}$$

الحدان الاوليان في جانب المعادلة الأيمن يعبران عن القدرة المبذولة على المائع داخل الحجم التحكمي، و \dot{Q} معدل سريان الحرارة إلى داخل الحجم التحكمي. بتجاهل اللزج (viscosity) يصبح الإجهاد (stress) σ :

$$\underline{\sigma} = -p \underline{n}$$

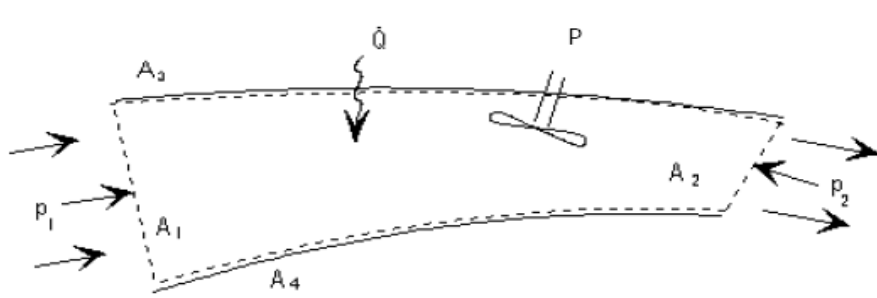
<-

$$\iiint_V \frac{\partial}{\partial t} [\rho(e + \frac{v^2}{2} + gz)] dV + \iint_A \rho(e + \frac{v^2}{2} + gz) \underline{v} \cdot d\underline{A} = - \iint_A p \underline{v} \cdot d\underline{A} + P + \dot{Q}$$

$$\iiint_V \frac{\partial}{\partial t} [\rho(e + \frac{v^2}{2} + gz)] dV + \iint_A \rho(e + \frac{p}{\rho} + \frac{v^2}{2} + gz) \underline{v} \cdot d\underline{A} = P + \dot{Q} \dots \dots \dots (2.8)$$

تطبيق على سريان رتيب أحادي البعد:

رتابة السريان تعني أن الحد الأول في المعادلة (2.8) يساوي صفر، و لا انتقال للكتلة عبر الأسطح (3) و (4).
وبذلك تختزل المعادلة إلى الصورة



الشكل 2.5

$$-\rho_1(e_1 + \frac{p_1}{\rho_1} + \frac{v_1^2}{2} + gz_1)v_1 A_1 + \rho_2(e_2 + \frac{p_2}{\rho_2} + \frac{v_2^2}{2} + gz_2)v_2 A_2 = P + \dot{Q}$$

بالاستعانة بمعادلة حفظ الكتلة للسريان الرتيب أحادي البعد (2.5)

$$\rho_1 v_1 A_1 = \rho_2 v_2 A_2 = \dot{m}$$

$$\dot{m}(e_1 + \frac{p_1}{\rho_1} + \frac{v_1^2}{2} + gz_1) + P + \dot{Q} = \dot{m}(e_2 + \frac{p_2}{\rho_2} + \frac{v_2^2}{2} + gz_2)$$

$$\frac{e_1}{g} + \frac{p_1}{\rho_1 g} + \frac{v_1^2}{2g} + z_1 + \frac{P}{\dot{m}g} + \frac{\dot{Q}}{\dot{m}g} = \frac{e_2}{g} + \frac{p_2}{\rho_2 g} + \frac{v_2^2}{2g} + z_2 \dots \dots \dots (2.9)$$

$\dot{Q} = 0$ في كثير من التطبيقات الهندسية يمكن تجاهل انتقال الحرارة
 $T_1 = T_2, e_1 = e_2$ و تجاهل التغير في درجة الحرارة
 $\rho_1 = \rho_2 = \rho$ ويمكن اعتبار السريان لا انضغاطي

فتصبح المعادلة (2.9)

$$\frac{P_1}{\rho_1 g} + \frac{v_1^2}{2g} + z_1 + \frac{P}{m g} = \frac{P_2}{\rho_2 g} + \frac{v_2^2}{2g} + z_2 \dots\dots\dots(2.10)$$

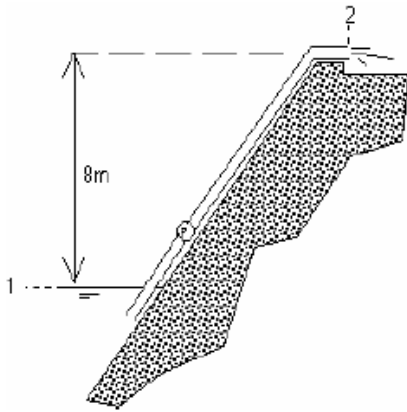
في حال أن القدرة P موجبة فإنها تمثل مضخة و إذا كانت سالبة فتمثل عنفة.
 في حال عدم وجود مضخة أو عنفة بين المقطعين (1) و (2) تصبح المعادلة (2.10)

$$\frac{P_1}{\rho g} + \frac{v_1^2}{2g} + z_1 = \frac{P_2}{\rho g} + \frac{v_2^2}{2g} + z_2 = \text{السمت الكلي} \dots\dots\dots(2.11)$$

أي: السمت الكلي = سمت الرفع + سمت السرعة + سمت الضغط

مثال

يُعرف الآتي عن وحدة ضخ ترفع الماء من النيل إلى أعلى الجرف:



الشكل (2.6)

- الرفع: 8m
- معدل السريان الحجمي 15 l/s
- قطر الأنبوب صعيد المضخة: 154mm
- قطر الأنبوب سافل المضخة: 102mm
- كثافة الماء: 1000kg/m³

المطلوب حساب:

- (أ) السرعة صعيد وسافل المضخة
- (ب) القدرة الخارجة من المضخة إذا اعتبرنا السريان لا لزج.

(أ) معادلة حفظ الكتلة (2.5) للسريان اللانضغاطي تُعطي

$$\mathbf{v}_u \cdot A_u = \mathbf{v}_d \cdot A_d = \dot{V} = 0.015 \text{ m}^3/\text{s}$$

$$v_u = \frac{0.015}{\frac{\pi}{4}(0.154)^2} = 0.81 \text{ m/s}$$

$$v_d = \frac{0.015}{\frac{\pi}{4}(0.102)^2} = 1.84 \text{ m/s}$$

حيث اللاحقة u تعني صعيد المضخة و اللاحقة d تعني سافل المضخة.

(ب) معادلة الطاقة لهذه الحالة (2.10)

$$\frac{p_1}{\rho g} + \frac{v_1^2}{2g} + z_1 + \frac{P}{\dot{m}g} = \frac{p_2}{\rho g} + \frac{v_2^2}{2g} + z_2$$

$$P = \dot{m}g \left[\frac{p_2 - p_1}{\rho g} + \frac{v_2^2 - v_1^2}{2g} + (z_2 - z_1) \right]$$

المقطعان (1) و (2) مفتوحان للجو و يعني ذلك

$$p_1 = p_2 = p_a$$

$$p_2 - p_1 = 0$$

$$z_2 - z_1 = 8 \text{ كما أن}$$

السطح (1) سطح النيل: سرعة نقصانه صفر!

$$v_1 = 0, v_2 = v_d$$

معدل سريان الكتلة \dot{m}

$$\dot{m} = \rho \dot{V} = 1000(0.015) = 15.0 \text{ kg/s}$$

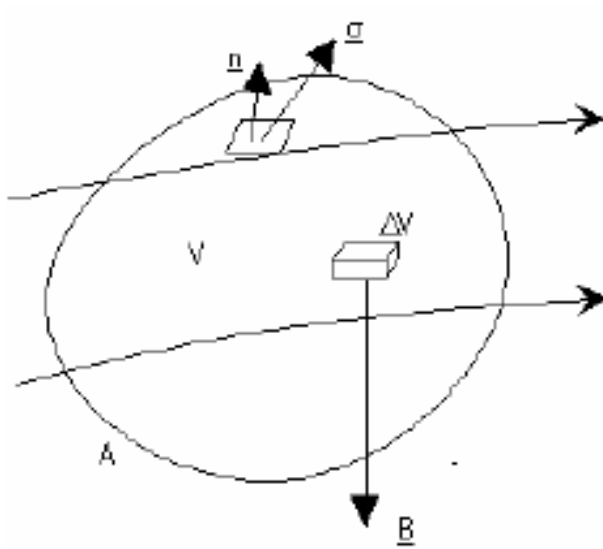
وتصبح المعادلة

$$P = (15.0)(9.81) \left[\frac{(1.84)^2}{2(9.81)} + 8 \right] = 1203 \text{ W}$$

القدرة الخارجة = **1.2 kW**

18.6 حفظ كمية التحرك (momentum conservation)

الشكل 2.6



يستمد هذا القانون من قانون نيوتن الثاني (Second Newtonian Law) للحركة مطابقاً على حجم التحكمي: "معدل تراكم كمية التحرك داخل الحجم التحكمي مضافاً إليه خالص معدل سريان كمية التحرك إلى خارج الحجم التحكمي بإنقال الكتلة يعادل مجموع القوى المؤثرة على المائع".

$$\frac{\partial}{\partial t} \iiint_V (\rho \underline{v}) dV + \iint_A \rho \underline{v} (\underline{v} \cdot d\underline{A}) = \iiint_V \underline{B} dV + \iint_A \underline{\sigma} dA$$

$$\iiint_V \frac{\partial}{\partial t} (\rho \underline{v}) dV + \iint_A \rho \underline{v} (\underline{v} \cdot d\underline{A}) = \iiint_V \underline{B} dV + \iint_A \underline{\sigma} dA \dots\dots\dots(2.12)$$

نسترجع هنا أن الإجهاد $\underline{\sigma}$ يساوي مجموع المتجهين $\rho \underline{n}$ - و $\underline{\tau}$. كما أن \underline{B} هي القوة الجسمية على وحدة حجمية و تتمثل في الأحوال الأعم في قوة الجاذبية على وحدة حجمية أي $\underline{B} = -\rho g \underline{k}$.

18.7 تلخيص المعادلات الأساسية (governing equations) لديناميك الموائع مع ملاحظات

18.7.1 معادلات السريان اللزجي (viscous flow) دون النظر الى تفاعلات الكيميائية (without considering chemical reactions)

Viscous flow: a flow which includes the dissipative, transport phenomena of viscosity and thermal conduction. The additional transport phenomenon of mass diffusion is not included because we are limiting our considerations to a homogenous, non-chemically reacting gas. Combustion for example is a flow with a chemical reaction. If

السريان اللزجي هو الذي يتضمن ظواهر التبدد والنقل ، اللزوجة والتوصيل الحراري إضافة لم يتم تضمين ظاهرة النقل لنشر الكتلة لأننا قمنا بتحديد اعتباراتنا إلى تفاعلات غاز متجانسة و غير كيميائياً. الاحتراق على سبيل المثال هو سريان مع تفاعل كيميائي. إذا كان لا بد من شمل النشر، لن يكون هناك

diffusion were to be included, there would be additional continuity equations – the species continuity equations involving mass transport of chemical species i due to a concentration gradient in the species.

Moreover the energy equation would have an additional term to account for energy transport due to the diffusion of species.

With the above restrictions in mind, the governing equations for an unsteady, three-dimensional, compressible, viscous flow are:

معادلات استمرارية إضافية -- أنواع معادلات الاستمرارية التي تنطوي على نقل الكتلة للأنواع الكيميائية i بسبب تدرج التركيز للأنواع.

وعلاوة على ذلك فإن معادلة الطاقة لديها إضافة مدة على حساب نقل الطاقة بسبب انتشار الأنواع.

مع الاخذ في الاعتبار القيود المذكورة أعلاه ، والمعادلات الأساسية لغير ثابت، ثلاثي الأبعاد انضغاطي، ، والسريان اللزج هي :

Continuity equations

(Non-conservation form – [Wendt 2009], Eq.2.18)

معادلات الاستمرارية

(بالشكل الغير محافظي)

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \vec{V} = 0$$

(Conservation form – [Wendt 2009], Eq. 2.27)

الشكل التحفظي

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$$

Equation [Wendt 2009], (2.18) is the continuity equation in non-conservation form. Note that:

المعادلة [Wendt 2009], (2.18) هي معادلة الاستمرارية في الشكل الغير تحفظي. ملاحظة ما يلي :

1) By applying the model of an *infinitesimal fluid element*, we have obtained Eq. [Wendt 2009], (2.18) *directly* in partial differential form.

1) من خلال تطبيق نموذج لعنصر مائع متناهي الصغر، لنحصل على المعادلة. [Wendt 2009], (2.18) مباشرة على شكل تفاضلي جزئي.

2) By choosing the model to be *moving with the flow*, we have obtained the **non-conservation** form of the continuity equation, namely Eq. [Wendt 2009], (2.18).

2) عن طريق اختيار النموذج الذي يتحرك مع السريان، لقد حصلنا على الشكل الغير تحفظي لمعادلة الاستمرارية ، وهي

Equation [Wendt 2009], (2.27) is the continuity equation in **conservation**

المعادلة. [Wendt 2009], (2.18).

form. Note that:

- 1) By applying the model of a *finite control volume*, we have obtained Eq. [Wendt 2009], (2.23) *directly* in integral form. Only after some manipulation of the integral form the partial differential form, namely Eq. [Wendt 2009], (2.27), is obtained.
- 2) By choosing the model to be *fixed in space*, we have obtained the conservation form of the continuity equation, namely Eqs. [Wendt 2009], (2.13) and (2.27).

المعادلة [Wendt 2009]، (2.27) هي معادلة الاستمرارية في

الشكل التحفظي ملاحظة ما يلي :

(1) من خلال تطبيق نموذج لمراقبة الحجم المحدود، حصلنا على

المعادلة. [Wendt 2009]، (2.23) مباشرة في شكل

متكامل. فقط بعد مرور بعض معالجات للشكل التفاضلي

الجزئي. اي [Wendt 2009]، (2.27). التي حصلنا عليها

(2) عن طريق اختيار نموذج للتثبيت في الفضاء، لنحصل على

شكل التحفظي لمعادلة الاستمرارية

Momentum equations

(Non-conservation form – [Wendt 2009],
Eqs. 2.36a-c)

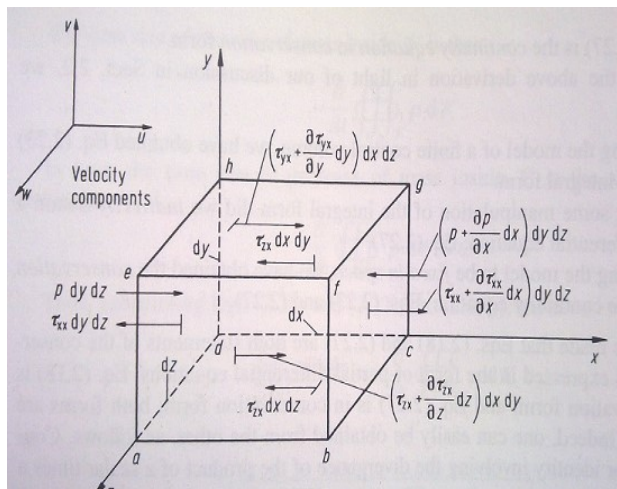
معادلات كمية التحرك

$$\text{x-component: } \rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$$

$$\text{y-component: } \rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$$

$$\text{z-component: } \rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$$

[Wendt 2009], Fig.2.5:
Infinitesimally small,
moving fluid element.
Only the forces in the x
direction are shown.



[Wendt 2009]، الشكل 2.5: تم

الصغير. لا تظهر إلا للقوات في الاتجاه

Total force in the x-direction: F_x

F_x هي القوة الاجمالية في اتجاه x

[Wendt 2009], S.28 Def. of body forces and surface forces:

هناك نوعين من القوة في هذا الاطار:

1) *Body forces*, which act directly on the volumetric mass of the fluid element.

(1) قوات جسمية التي تتفاعل مباشرةً على الكتلة الحجمية للعضو

Examples:

gravitational, electric and magnetic forces. Def.: body force on the fluid element acting in the x-direction = $\rho f_x (dx dy dz)$.

مائعي (fluid element). و امثلة هي: القوة الجاذبية والكهربائية

والمغناطيسية.

تعريف: القوة الجسمية على العضو المائع تتمثل في الاتجاه x =

$$\rho f_x (dx dy dz)$$

2) *Surface forces*, which act directly on the surface of the fluid element.

(2) قوات سطحية التي تتفاعل مباشرةً على سطح العضو المائعي. وهو

They are due to only two sources: (a) pressure distribution acting on the surface, imposed by the outside fluid surrounding the fluid element, and (b) the shear and normal stress distributions acting on the surface, also imposed by the outside fluid "tugging" or "pushing" on the surface by means of friction.

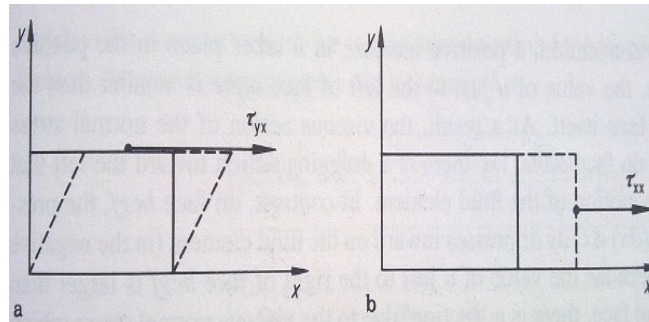
ناشئ من مصدرين اثنين فقط : (a) توزيع الضغط التي تعمل

على السطح، التي يفرضها خارج المائع في المناطق المحيطة بالعضو

المائع، و (b) هي توزيعات الضغط الطبيعي و القص التي تعمل

على السطح، كما فرضت من قبل خارج المائع "التجاذبات" أو

"الدفء" على السطح عن طريق الاحتكاك.



[Wendt 2009], Fig.2.6: Illustration of shear and normal stresses

[Wendt 2009], الشكل 2.6: رسم

توضيحي للقص و للضغوطات الطبيعية

(Conservation form – [Wendt 2009], Eqs. 2.42a-c)

الشكل التحفظي – [Wendt 2009], Eqs.

2.42a-c))

$$\text{x-component: } \frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} - \rho f_x$$

$$\text{y-component: } \frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} - \rho f_y$$

$$\text{z-component: } \frac{\partial(\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} - \rho f_z$$

Energy equation

معادلة الطاقة

(Non-conservation form – [Wendt 2009], Eq. 2.52)

الشكل الغير تحفظي

$$\begin{aligned} \rho \frac{D}{Dt} \left(e + \frac{V^2}{2} \right) &= \rho q + \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) \\ &- \frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} + \frac{\partial(u\tau_{xx})}{\partial x} \\ &+ \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} \\ &+ \frac{\partial(v\tau_{zy})}{\partial z} + \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} + \rho \vec{f} \cdot \vec{V} \end{aligned}$$

(Conservation form – [Wendt 2009], Eq. 2.64)

الشكل التحفظي

$$\begin{aligned} \frac{\partial}{\partial t} \left[\rho \left(e + \frac{V^2}{2} \right) \right] + \nabla \cdot \left[\rho \left(e + \frac{V^2}{2} \vec{V} \right) \right] \\ = \rho q + \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) \\ + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) - \frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} + \frac{\partial(u\tau_{xx})}{\partial x} \\ + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \frac{\partial(v\tau_{xy})}{\partial x} + \frac{\partial(v\tau_{yy})}{\partial y} \\ + \frac{\partial(v\tau_{zy})}{\partial z} + \frac{\partial(w\tau_{xz})}{\partial x} + \frac{\partial(w\tau_{yz})}{\partial y} + \frac{\partial(w\tau_{zz})}{\partial z} + \rho \vec{f} \cdot \vec{V} \end{aligned}$$

without) معادلات السريان الا لزجي (inviscid flow) دون النظر الى تفاعلات الكيميائية 18.7.2

(considering chemical reactions)

Here are the viscous terms of the above equations dropped. هنا شروط اللزوجة لمعادلات الإسقاط أعلاه.

Surveying the above governing equations, several comments and observations can be made:

- 1) They are coupled system of non-linear partial differential equations, and hence are very difficult to solve analytically. To date, there is no general closed-form solution to these equations.
- 2) For the momentum and energy equations, the difference between the non-conservation and conservation forms of the equation is just the left-hand side.
- 3) Note that the conservation form of the equations contain terms on the left-hand side which include the divergence of some quantity, such as $\nabla \cdot (\rho \cdot \vec{V})$, $\nabla \cdot (\rho u \vec{V})$, etc. For this reason, the conservation form of the governing equations is sometimes called the *divergence form*.
- 4) The normal and stress terms in these equations are functions of the velocity gradients, as given by [Wendt 2009], Eqs. (2.43a-f).
- 5) The system contains five equations in terms of six unknown flow-field variables, ρ, p, u, v, w, e . In aerodynamics, it is generally reasonable to assume the gas is a perfect gas (which assumes that intermolecular forces are negligible). For a perfect gas, the equation of state is $p = \rho RT$, where R is the specific gas constant. This provides a sixth

إذا تأملنا المعادلات الأساسية، نستطيع ان نقول التالي:

- (1) هي مجموعة مزوجة من المعادلات التفاضلية الجزئية الغير خطية وبالتالي من الصعب جدا حلها تحليلياً، حتى الآن ، لا يوجد اي حل تحليلي لهذه المعادلات.
- (2) لمعادلات كمية التحرك والطاقة ، الفرق بين الأشكال الغير تحفظية و التحفظية على المعادلة هو مجرد الجانب الأيمن.
- (3) لاحظ أن شكل التحفظي للمعادلات تحتوي شروط على الجانب الأيمن، التي تشمل بعض الاختلاف في الكمية ، مثل $\nabla \cdot (\rho \cdot \vec{V})$, $\nabla \cdot (\rho u \vec{V})$ ، وما إلى ذلك. لهذا السبب ، يسمى في بعض الأحيان الشكل التحفظي للمعادلات الأساسية بشكل التباعد.
- (4) الشروط العادية و الضغط، في هذه المعادلات هي دالات من تدرجات السرعة ، كما معطى حسب [Wendt 2009], Eqs. (2.43a-f).
- (5) تحتوي المنظومة على خمسة معادلات في المصطلحات لستة متغيرات غير معروفة لحقل سريان ρ, p, u, v, w, e . في الديناميكا الجوية ، من المعقول أن نفترض عموماً الغاز هو غاز المثالي (الذي يفترض أن القوات بين الجزيئات تكاد لا تذكر). بالنسبة للغاز مثالي ، المعادلة للحالة هي $p = \rho RT$ حيث R هو الثابت المحدد للغاز. هذا يعطي المعادلة السادسة ، لكنه

equation, but it also introduces a seventh unknown, namely temperature, T . A seventh equation to close the entire system must be a thermodynamic relation between state variables. For example, $e = e(T,p)$ For a calorically perfect gas (constant specific heats), this relation would be $e = c_v T$ where c_v is the specific heat at constant volume.

يقدم أيضا مجهول سابع ، وهي درجة الحرارة ، T . المعادلة السابعة لإغلاق النظام بأكمله يجب أن تكون علاقة حرارية بين متغيرات الحالة. على سبيل المثال ، $e = e(T,p)$ بالنسبة لغاز مثالي بالوحدات الحرارية (تسخين ثابت محدد) ، فسوف تكون هذه العلاقة $e = c_v T$ حيث c_v هي الحرارة النوعية لحجم ثابت.

6) Historically, the momentum equations for a viscous flow are called the **Navier-Stokes equations**. However, in modern CFD literature, "a Navier-Stokes solution" simply means a solution of a *viscous flow problem* using *full governing equations (including continuity as well as energy and momentum)*.

(6) تاريخيا ، وتسمى معادلات كمية التحرك للتدفق اللزج بمعادلات نافير ستوكس (Navier-Stokes). ومع ذلك ، في الأدب ال CFD الحديث " ، وهو حل نافير ستوكس " يعني ببساطة إيجاد حل لمشكلة التدفق اللزج باستعمال المعادلات الأساسية (بما في ذلك الاستمرارية فضلا عن الطاقة وكمية التحرك).

18.7.4 الحالات الجدارية (boundary conditions)

The boundary conditions, and sometimes the initial conditions, dictate the particular solutions to be obtained from the governing equations. (This makes the difference for example between the flow over a Boing 757 or past a wind mill, although the equations are the same). For a viscous fluid, the boundary condition on a surface assumes no relative velocity between the surface and the gas immediately at the surface. This is called the *no-slip* condition. If the surface is stationary, then at the surface (for a viscous flow) $u = v = w = 0$

الحالات الجدارية ، وأحيانا الحالات الأولية، تملئ حلولاً معينة التي يمكن الحصول عليها من المعادلات الأساسية. (وهذا ما يجعل الفرق مثلا بين السريان على ال Boing 757 او طاحونة الرياح السابقة ، على الرغم من ان المعادلات هي نفسها). للمائع اللزج، الحالة الجدارية على السطح لا تتحمل السرعة النسبية بين السطح والغاز مباشرة على السطح. وهذا ما يسمى حالة عدم الانزلاق (*no-slip*). إذا كان السطح هو ثابت إذا $u = v = w = 0$ على السطح (للسريان اللزج)

For an inviscid fluid, the flow slips over the surface (there is no friction to promote its 'sticking' to the surface); hence, at the surface,

للسائل الغير لزج، السريان ينزلق على السطح (لا يوجد احتكاك من أجل تعزيز "الاصق" على السطح)، وبالتالي على السطح،

the flow must be tangent to the surface. $\vec{V} \cdot \vec{n} = 0$ at the surface (for a inviscid flow), where \vec{n} is a unit vector perpendicular (that means orthogonal) to the surface. The boundary conditions elsewhere in the flow depend on the type of problem being considered, and usually pertain to inflow and outflow boundaries at a finite distance from the surfaces, or an 'infinity' boundary condition infinitely far from surface.

The boundary conditions discussed above are physically boundary conditions in nature.

In CFD we have an additional concern, namely the proper numerical implementation of the boundary conditions.

السريان يجب أن يكون مماس الى السطح. $\vec{V} \cdot \vec{n} = 0$ على السطح (للسريان الالزجي) حيث \vec{n} هو وحدة متجه عمودي (وهذا يعني متعامد) على السطح. الحالات الجدارية في أماكن أخرى من السريان يعتمد على نوع المشكلة التي يجري النظر فيها، وتتعلق عادة بحدود السريان الداخل و الخارج على مسافة محدودة من السطوح ، أو حالة الحدود "اللانهاية" التي بشكل مطلق بعيدة من السطح. الحالات الجدارية التي نوقشت أعلاه هي فعليا الحالات الجدارية الفيزيائية في الطبيعة. في CFD لدينا قلق إضافي، لمعرفة التنفيذ العددية السليم للحالات الجدارية.

18.8 اشكال للمعادلات الاساسية تلائم مع د.م.ح.: ملاحظات على الشكل التحفظي (conservation form)

نستطيع ان نكتب مجموعة المعادلات الاساسية بالشكل التحفظي (conservation form) بالشكل العام التالي:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} = J \quad [\text{Wendt}], \text{ Eq. 2.65}$$

حيث

$$U = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho(e + V^2/2) \end{Bmatrix} \quad H = \begin{Bmatrix} \rho w \\ \rho u w - \tau_{zx} \\ \rho v w - \tau_{zy} \\ \rho w^2 + p - \tau_{zz} \\ \rho(e + V^2/2)w + p w - k \frac{\partial T}{\partial z} - u \tau_{zx} - v \tau_{zy} - w \tau_{zz} \end{Bmatrix}$$

$$F = \begin{Bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho v u - \tau_{xy} \\ \rho w u - \tau_{xz} \\ \rho(e + V^2/2)u + p u - k \frac{\partial T}{\partial x} - u \tau_{xx} - v \tau_{xy} - w \tau_{xz} \end{Bmatrix} \quad J = \begin{Bmatrix} 0 \\ \rho f_x \\ \rho f_y \\ \rho f_z \\ \rho(u f_x + v f_y + w f_z) + p q \end{Bmatrix}$$

$$G = \begin{Bmatrix} \rho v \\ \rho u v - \tau_{yx} \\ \rho v^2 + p - \tau_{yy} \\ \rho w v - \tau_{yz} \\ \rho(e + V^2/2)v + p v - k \frac{\partial T}{\partial y} - u \tau_{yx} - v \tau_{yy} - w \tau_{yz} \end{Bmatrix}$$

In [Wendt], Eq. 2.65, the column vectors F , G , and H are called the flux terms (or flux vectors), and J represents a 'source term' (which is zero if body forces are negligible). For an unsteady problem, U is called the solution vector because the elements in $U(\rho, \rho u, \rho v, \text{etc.})$ are the dependent variables which are usually solved numerically in steps of time. Please note that, in this formalism, it is the elements of U that are obtained computationally, i.e. numbers are obtained for the products $\rho, \rho u, \rho v, \rho w$ and $\rho(e + V^2/2)$. Of course, once numbers are known for these dependent variables (which includes ρ by itself), obtaining the primitive variables is simple:

في المعادلة [Wendt], Eq. 2.65، الموجهات العمودية F و G و H تسمى الموجهات السريانية، و J يمثل "مصطلح مصدر" (والذي هو صفر إذا كانت قوى الجسم تكاد لا تذكر). لمشكلة غير رتيبة، تسمى U متجه الحل لان العناصر في $U(\rho, \rho u, \rho v, \dots)$ هي التي تعتمد على متغيرات يتم حلها عادة عددياً في خطوات الزمن. يرجى ملاحظة أنه في هذه الشكليات، فإن عناصر U هي التي يتم الحصول عليها حسابياً، مثلاً الأرقام التي يتم الحصول عليها للمنتجات $\rho, \rho u, \rho v, \rho w$ و $\rho(e + V^2/2)$ بطبيعة الحال، عندما تعرف الأرقام لأول مرة لهذه المتغيرات التابعة (التي تضم ρ في حد ذاته)، الحصول على المتغيرات البدائية هي بسيطة :

$$\begin{aligned}\rho &= \rho \\ u &= \frac{\rho u}{\rho} \\ v &= \frac{\rho v}{\rho} \\ w &= \frac{\rho w}{\rho} \\ e &= \frac{\rho(e + V^2/2)}{\rho} - \frac{u^2 + v^2 + w^2}{2}\end{aligned}$$

لسريان لا لزج المعادلة [Wendt et. al. 2009] Eq.(2.65)، تبقى كما هي، إلا ان الموجهات العامودية اصبحت ابسط.

إذا تأملنا الشكل التحفظي للمعادلات اللا لزجية في باب 2.7.2 نجد ان

$$U = \left\{ \begin{array}{l} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho(e + V^2/2) \end{array} \right\}$$

$$F = \left\{ \begin{array}{l} \rho u \\ \rho u^2 + p \\ \rho v u \\ \rho w u \\ \rho u(e + V^2/2)u + p u \end{array} \right\}$$

$$G = \left\{ \begin{array}{l} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho wv \\ \rho v(e + V^2/2) + pv \end{array} \right\}$$

$$H = \left\{ \begin{array}{l} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho w(e + V^2/2) + pw \end{array} \right\}$$

$$J = \left\{ \begin{array}{l} 0 \\ \rho f_x \\ \rho f_y \\ \rho f_z \\ \rho(uf_x + vf_y + wf_z) + p \dot{q} \end{array} \right\}$$

For the numerical solution of an unsteady inviscid flow, once again the solution vector is U , and the dependent variables for which numbers are directly obtained are products $\rho, \rho u, \rho v, \rho w$ and $\rho(e + V^2/2)$. For a steady inviscid flow, $\partial U / \partial t = 0$.

Frequently, the numerical solution to such problems takes the form of 'marching' techniques; for example, if the solution is being obtained by marching in the x-direction, then [Wendt et. al. 2009], Eq.(2.65) can be written as

للحل العددي للسريان اللازجي الغير رتيب، مرة أخرى متجه الحل هو U ، والمتغيرات التابعة لاية ارقام التي يتم الحصول عليها مباشرة من المنتجات $\rho, \rho u, \rho v, \rho w$ و $\rho(e + V^2/2)$ للسريان اللازجي الرتيب $\partial U / \partial t = 0$.

في كثير من الأحيان، فإن الحل العددي لهذه المشاكل تأخذ شكل تقنيات "سيرية" ('marching')، على سبيل المثال، إذا كان يتم الحصول على حل عن طريق السير في اتجاه x ، ثم [Wendt et. al. 2009], Eq.(2.65)

يمكن كتابتها على النحو التالي

$$\frac{\partial F}{\partial x} = J - \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} \quad [\text{Wendt}], \text{ Eq. 2.66}$$

Here, F becomes the 'solution vector', and the dependent variables for which numbers are obtained are $\rho, \rho u, \rho v, \rho w$ and $\rho(e + V^2/2)$.

From these dependent variables, it is still possible to obtain the primitive variables, although the algebra is more complex than in the previously discussed case.

Notice that the governing equations when written in the form of [Wendt et. al. 2009], Eq.(2.65), have no flow variables outside the

هنا F تصبح "متجه المحلول" و المتغيرات التابعة لاية ارقام يمكن الحصول عليها تكون $\rho, \rho u, \rho v, \rho w$ و $\rho(e + V^2/2)$.

من هذه المتغيرات التابعة يمكن دائماً الحصول على المتغيرات الاولية (primitive variables) على الرغم من أن الجبر هو أكثر تعقيداً مما

كانت عليه في الحالة التي نوقشت سابقاً. نلاحظ أن المعادلات

الاساسية عند كتابتها في الشكل من [Wendt et. al. 2009]،

single x, y, z , and t derivatives. Indeed, the terms in [Wendt et. al. 2009], Eq. (2.65) have everything buried inside these derivatives. The flow equations in the form of [Wendt et. al. 2009], Eq.(2.65) are said to be in strong conservation form. In contrast, examine the forms [Wendt et. al. 2009], Eq.(2.42a,b and c) and [Wendt et. al. 2009], Eq.(2.64). These equations have a number of x, y and z derivatives explicitly appearing on the right – hand side. These are the *weak conservation* form of the equations.

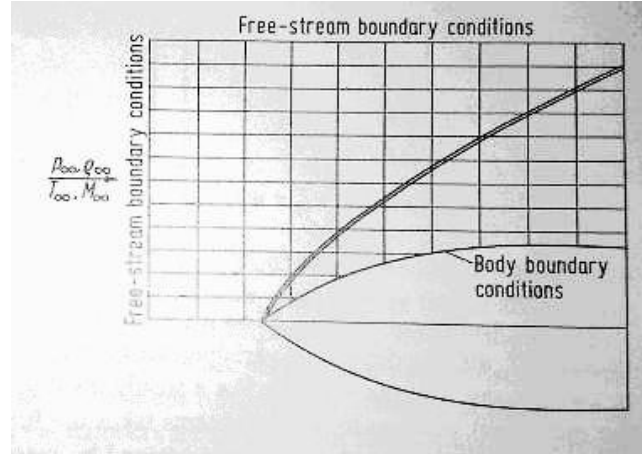
The form of the governing equations giving by Eq. (2.65) is popular in CFD; let us explain why. In flow fields involving shock waves, there are sharp, discontinuous changes in the primitive flow-field variables p, ρ, u, T , etc., across the shocks. Many computations of flows with shocks are designed to have the shock waves appear naturally within the computational space as a direct result of the overall flow field solution, i.e. as a direct result of the general algorithm, without any special treatment to take care of the shocks themselves. Such approaches are called shock capturing methods. This is in contrast to the alternate approach, where shock waves are explicitly introduced into the flow-field solution, the exact Rankine-Hugoniot relations for changes across a shock are used to relate the flow immediately ahead of and behind the shock, and the governing flow equations are used to calculate the remainder of the flow field. This approach is called the shock-fitting method. These two different approaches are illustrated in Figs. 2.8 and 2.9. In Fig.2.8, the computational domain for calculating the supersonic flow over the body

المعادلة (2.65) ، ليس لديهم متغيرات السريان خارج المفرد X, Y, Z ، والمشتقات t . في الواقع ، الشروط في [Wendt et. al. 2009], Eq.(2.65) لديها كل شيء متخفي داخل هذه المشتقات. معادلات السريان في الشكل [Wendt et. Al 2009], Eq. (2.65) تكون معروفة باسم الشكل التحفظي القوي في المقابل ، دراسة أشكال [Wendt et. al. 2009], Eq. (2.42a,b and c) [Wendt et. al. 2009], Eq.(2.64). هذه المعادلات لديها عدد من المشتقات x, y, z التي تظهر بوضوح على الجانب الأيمن. هذه هي الاشكال التحفظية الضعيفة في المعادلة.

شكل المعادلات الاساسية معطى عبر المعادلة (2.65) هي معروفة جداً في CFD؛ دعونا نوضح السبب. في مجالات السريان تشمل موجات الصدمة، هناك تكون حادة، التغيرات المتقطعة في متغيرات مجال السريان الاولي (primitive flow-field variables): p, ρ, u, T, \dots ، عبر الصدمات. صممت العديد من حسابات السريان مع الصدمات هي مصممة لتظهر موجات الصدمة بشكل طبيعي في غضون الحسابة كنتيجة مباشرة من محلول حقل السريان العام، أي كنتيجة مباشرة للخوارزمية العامة، دون أي معالجة خاصة لاخذ الحذر من الصدمات نفسها. ويسمى هذا النهج أساليب التقاط الصدمة. هذا هو النقيض للنهج البديل ، حيث يتم إدخال بوضوح موجات الصدمة في محلول مجال السريان، يتم استخدام العلاقات الدقيقة Rankine-Hugoniot للتغيرات عبر الصدمة لربط السريان مباشرة امام و وراء الصدمة ، و معادلات السريان الاساسية تُستخدم لحساب ما تبقى من مجال السريان. وهذا ما يسمى نهج أسلوب الصدمة المناسب (shock-fitting method). ويتضح هذين النهجين المختلفين في الشكل 2.8 و 2.9. في الشكل 2.8، المجال الحسابي

extends both upstream and downstream of the nose. The shock wave is allowed to form within the computational domain as a consequence of the general flow-field algorithm,

[Wendt et.al.2009],
Fig.2.8: Mesh for the shock-capturing approach



[Wendt et.al.2009]

الشكل 2.8 : شبكة لنهج التقاط الصدمة

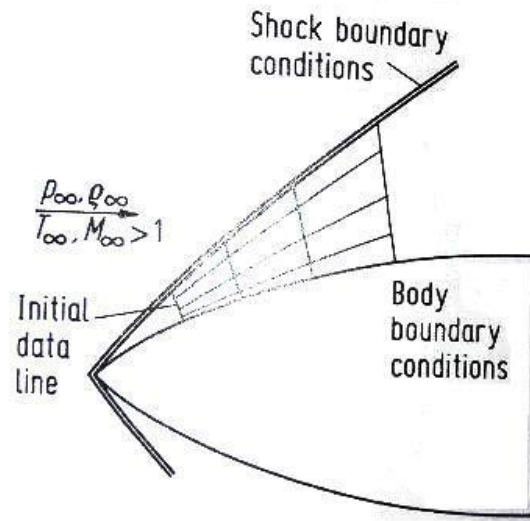
without any special shock relations being introduced. In this manner, the shock wave is captured within the domain by means of the computational solution of the governing partial differential equations. Therefore, Fig. 2.8 is an example of the shock-capturing method. In contrast, Fig. 2.9 illustrates the same flow problem, except that now the computational domain is the flow between the shock and the body. The shock wave is introduced directly into the solution as an explicit discontinuity, and the standard oblique shock relations (the Rankine-Hugoniot relations) are used the free stream supersonic flow ahead of the shock to the flow computed by the partial differential equations downstream of the shock. Therefore, Fig. 2.9 is an example of the shock-fitting method. There are advantages and disadvantages of both methods. For example, the shock-capturing method is ideal for complex flow problems involving shock waves for which we do not know either the location or number of shocks. Here, the shocks simply form within the computational domain as

دون إدخال أية علاقات لصدمة خاصة. في هذه الطريقة ، يتم التقاط موجة الصدمة داخل المجال عن طريق الحل الحسابي للمعادلات التفاضلية الجزئية الأساسية. ولذلك ، الشكل 2.8 مثال على أسلوب التقاط الصدمة. في المقابل ، الشكل 2.9 يوضح مشكلة السريان نفسها ، إلا أن المجال الحسابي الآن هو السريان بين الصدمة والجسم. ادخال موجة الصدمة مباشرة في المحلول بمثابة انقطاع واضح ، وتستخدم معيار العلاقات المقياسية للصدمة المائلة (العلاقات Rankine-Hugoniot) سريان الانسياب الحر الفوق الصوتي قبل الصدمة لحساب السريان بواسطة المعادلات التفاضلية الجزئية باتجاه الصدمة . ولذلك ، الشكل 2.9 مثال على أسلوب الصدمة الملائمة. هناك مزايا وعيوب لكل من هذه الأساليب. على سبيل المثال ، الأسلوب التقاط الصدمة الأسلوب الأفضل لمشاكل السريان المعقدة التي تنطوي على موجات الصدمة التي لا نعرف مكان أو عدد الصدمات. هنا ، تتشكل الصدمات ببساطة داخل المجال الحسابي كما يكون في الطبيعة. وعلاوة على

nature would have it. Moreover, this takes place without requiring any special treatment of the shock within the algorithm, and hence simplifies the computer programming. However, a disadvantage of this approach is that the shocks are generally smeared over a number of grid points in the computational mesh, and hence the numerically obtained shock thickness bears no relation what-so-ever to the actual physical shock thickness, and the precise location of the shock discontinuity is uncertain within a few mesh sizes. In contrast, the advantage of the shock-fitting method is

[Wendt et.al.2009], Fig.2.9: Mesh for the shock-fitting approach

ذلك ، وهذا يحدث من دون الحاجة إلى أي علاج خاص لحالة الصدمة داخل الخوارزمية ، و بالتالي يبسط برمجية الكمبيوتر. ومع ذلك ، فإن العائق في هذا النهج هو أن الصدمات عموماً تلتطخ على عدد من النقاط الشبكة في الشبكة الحاسوبية ، وبالتالي الحصول عددياً على سمك الصدمة لا علاقة له على الإطلاق بسمك الصدمة الفيزيائي الفعلي ، و الموقع الدقيق في تقطع الصدمة غير مؤكد ضمن بعض أحجام شبكة. في المقابل ، الفائدة من أسلوب الصدمة المناسبة (shock-fitting) هو



[Wendt

et.al.2009] :

. الشكل 2.9

: شبكة لنهج

الصدمة

المناسبة

that the shock is always treated as a discontinuity, and its location is well-defined numerically. However, for a given problem you have to know in advance approximately where to put the shock waves, and how many there are. For complex flows, this can be a distinct disadvantage. Therefore, there are pros and cons associated with both shock-capturing and shock-fitting methods, and both have been employed extensively in CFD. In fact, a combination of these two methods is used to predict the formation and approximate location of shocks, and then these shocks are fit with explicitly in those parts of a flow field where you know

أن تعامل الصدمة دائماً على أنها متقطعة ، وموقعها واضح المعالم من الناحية العددية. ومع ذلك ، لمشكلة معينة يجب أن تعرف سابقاً و لو حتى تقريبياً أين توضع موجات الصدمة، و عددها. لتدفقات معقدة ، يمكن ان يكون هذا عائقاً واضح. لذلك ، هناك إيجابيات وسلبيات على حد سواء مرتبطة بكلا الاسلوبين: التقاط الصدمة (shock-capturing) و الصدمة المناسبة (shock-fitting) ، واستخدام الاسلوبين على نطاق واسع في CFD. في الواقع ، يتم استخدام مزيج من هاتين الطريقتين للتنبؤ بتشكيل

in advance they occur, and to employ a shock-capturing method for the remainder of the flow field in order to generate shocks that you cannot predict in advance.

Again, what does all of this discussion have to do with the conservation form of the governing equations as given by Eq. (2.65)? Simply this. For the shock-capturing method, experience has shown that the conservation form of the governing equations should be used. When the conservation form is used, the computed flow-field results are generally smooth and stable. However, when the non-conservation form is used for a shock-capturing solution, the computed flow-field results usually exhibit unsatisfactory spatial oscillations (wiggles) upstream and downstream of the shock wave, the shocks may appear in the wrong location and the solution may even become unstable. In contrast, for the shock-fitting method, satisfactory results are usually obtained for either form of the equations-conservation or non-conservation.

والموقع التقريبي للصدمات ، ومن ثم يتم احتواء هذه الصدمات بوضوح مع في أجزاء من حقل السريان حيث نعرف سابقاً أنها تحدث ، واستخدام طريقة التقاط الصدمة لما تبقى من حقل السريان من أجل توليد الصدمات التي لا يمكن التنبؤ بها مسبقاً. مرة أخرى ، ماذا يعني كل هذا النقاش يجب أن نفعل مع الشكل التحفظي للمعادلات الأساسية تعطي حسب المعادلة. (2.65)؟ هذا ببساطة. لأسلوب التقاط الصدمة ، وقد أثبتت التجربة أنه يجب استخدام النموذج التحفظي للمعادلات الأساسية. عندما يستخدم الشكل التحفظي عموماً تكون النتائج الحسابية على نحو سلس ومستقر. ومع ذلك ، عندما يتم استخدام شكل غير تحفظي لمحاول التقاط الصدمة ، النتائج الحسابية لحقل السريان تظهر عادة المكانية التذبذبات غير مرضية (ملتوية) بعكس أو باتجاه موجة الصدمة ، قد تظهر الصدمات في الموقع الخطأ والمحلل قد يصبح أيضاً غير مستقر. في المقابل ، لأسلوب الصدمة المناسبة ، وعادة ما يتم الحصول على نتائج مرضية لأي شكل من أشكال المعادلات التحفظية أو غير التحفظية.

Why is the use of the conservation form of the equations so important for the shock-capturing method? The answer can be seen by considering the flow across a normal shock wave, as illustrated in Fig. 2.10. Consider the density distribution across the shock, as sketched in Fig. 2.10(a). Clearly, there is a discontinuous increase in ρ across the shock. If the non-conservation form of the governing equations were used to calculate this flow, where the primary dependent variables are the primitive variables such as p and ρ , then the equations would see a large discontinuity in the dependent variable p . This in turn would compound the numerical errors associated with the calculation of p . On the other hand, recall the continuity equation for a normal shock wave (see Refs.[1,3]):

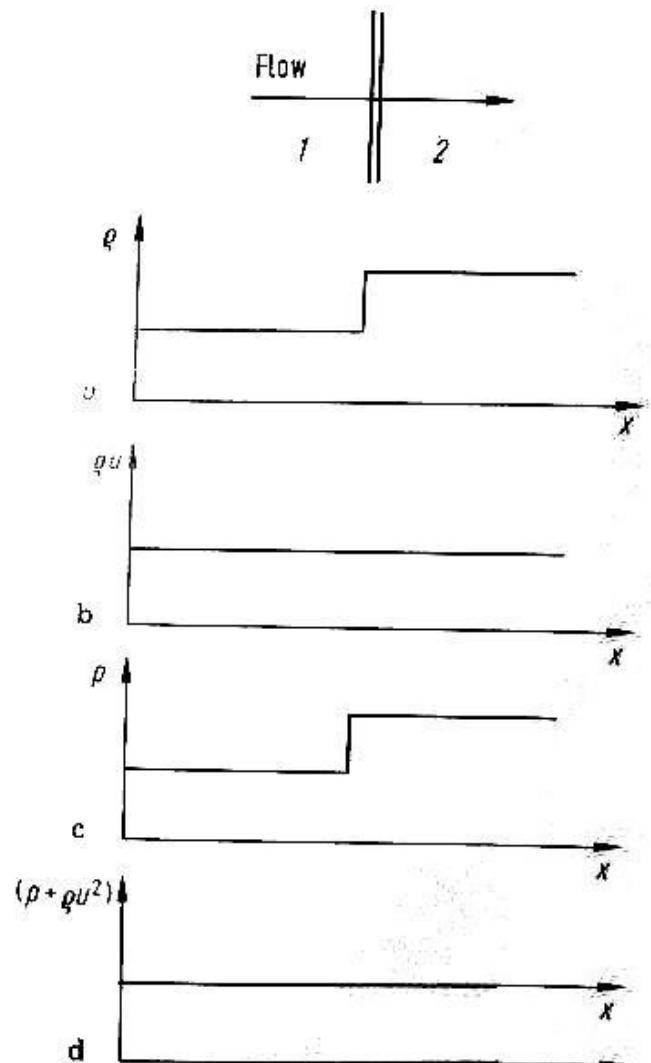
$$\rho_1 u_1 = \rho_2 u_2 \quad (2.67)$$

From Eq. (2.67), the *mass flux*, ρu , is constant across the shock wave, as illustrated in Fig. 2.10(b). The conservation form of the governing equations uses the product ρu as a dependent variable, and hence the conservation form of the equations see no discontinuity in this dependent variable across the shock wave. In turn, the numerical accuracy and stability of the solution should be greatly enhanced. To reinforce this discussion, consider the momentum equation across a normal shock wave [1,3]:

$$(2.68) \rho_1 + \rho_1 u_1^2 = \rho_2 + \rho_2 u_2^2$$

As show in Fig. 2.10(c), the pressure itself is discontinuous across the shock ; however, from Eq. (2.68) the flux variable $(\rho + \rho u^2)$ is constant across the shock.

[Wendt et. al. 2009], Fig.2.10: Variation of flow properties through a normal shock wave



This is illustrated in Fig. 2.10(d). Examining the inviscid flow equations in the conservation form given by Eq. (2.65), we clearly see that the quantity $(\rho + \rho u^2)$ is one of the dependent variables. Therefore, the conservation form of the equations would see no discontinuity in this dependent variables across the shock. Although this example of the flow across a normal shock wave is somewhat simplistic, it serves to explain why the use of the conservation form of the governing equations are so important for calculations using the shock-capturing method. Because the conservation form uses flux variables as the dependent

variables, and because the changes in these flux variables are either zero or small across a shock wave, the numerical quality of a shock-capturing method will be enhanced by the use of the conservation form in contrast to the non-conservation form, which uses the primitive variables as dependent variables.

In summary, the previous discussion is one of the primary reasons why CFD makes a distinction between the two forms of the governing equations—conservation and non-conservation. And this is why we have gone to great lengths in this chapter to derive these different forms, and why we should be aware of the differences between the two forms.

(: طرق حسابية معتمدة على مؤطرات Incompressible Inviscid Flows) سرايين لا انضغاطية ولا لزجية 19 (Source and Vortex Panel Methods) الدوامة و النبع و الدوامة)

19.1 مدخل

في هذا الفصل سننظر ان شاء الله الى التحليل العددي (numerical analysis) لسرايين (flows) لا انضغاطية (incompressible) و لا لزجية (inviscid). مبدئياً يمكن ان يستخدم طريقة الفرق المحدود (finite-difference method) - التي ستناقش في ما بعد ان شاء الله - لحل هذا النوع من السرايين. ولكن يوجد طرق اخرى تأدي عدة الى حلول اكثر مناسبة لسرايين لا انضغاطية (incompressible) و لا لزجية (inviscid).

هذا الفصل يناقش احد هذه الطرق - المسماة طرق حسابية معتمدة على مؤطرات النبع و الدوامة (Source and Vortex Panel Methods). هذه الطرق اصبحت هي الطرق المقياسية والمعتمد عليها عادة في الشركات التي تصنع الطائرات و هذا منذ العقد 1960

طرق المؤطرات هي طرق حسابية عددية (numerical methods) تحتاج الى قوة حسابية ضخمة و لذلك كومبيوترات سريعة.

19.2 بعض الواجهة الاساسية لسريان لا انضغاطي و لا لزجي

السريان الغير انضغاطي (incompressible flow) هو سريان بكثافة (density) ثابتة ($\rho = const.$).

تصور عضو مائع (fluid element) بكثافة ثابتة ($m = const.$) يجري في سريان غير انضغاطي (incompressible flow) في موازاة خط انسياب (streamline). لأن الكثافة ثابتة فبالتالي الحجم (volume) لهذا العضو مائعي هو ايضا ثابت ($V = const.$). و لأن $\nabla \vec{V}$ (هي السرعة) يشكل التغيير لحجمي لعضو مائعي على مدار الزمان نستطيع ان نكتب:

$$\nabla \vec{V} = 0 \quad (3.1)$$

و هو الgradient و هو الgrad و هو علامة ملخصة لNABLA-Operator هنا ال ∇

و إلى هذا فاذا العضو مائعي (fluid element) ايضاً لا يدور لما يتحرك في موازاة الخط الانسياب (streamline) فبالتالي هذا السريان (flow) يسم لا دوراني (irrotational). لهذا النوع من السرايين، يمكن ان يعبر عن السرعة (velocity) كپوتنيزيال (potential) - يُعلم ب ϕ ⁵.

$$\vec{V} = \nabla \phi \quad (3.2)$$

⁵ لمزيد من الشرح انظر ملحق أ و (Anderson 1991).

$$\text{grad}\phi = \nabla\phi = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \phi = \begin{pmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \\ \frac{\partial\phi}{\partial z} \end{pmatrix}$$

إذا جمعنا الآن معادلة (3.1) و (3.2) نصل الى:

$$\nabla \cdot \nabla\phi = 0$$

أو،

$$\nabla^2\phi = 0$$

(3.3) تسمى معادلة Laplace (Laplace's equation)، احد المعادلات المشهورة والمدروسة جيداً في مجال الفيزياء الرياضية (mathematical physics).

من معادلة (3.3) نرى ان سرايين (flows) لا انضغاطية (incompressible) و لا لزجية (inviscid) تُحَكَّم بمعادلة Laplace (Laplace's equation).

و معادلة Laplace (Laplace's equation) هي خطية (linear).

و لذلك كل عدد من حلول خصوصية لمعادلة (3.3) يمكن ان تزداد (added) مع بعض ليستنتج حل آخر.

و هذا يُري فلسفة اساسية لحل من سريان غير انضغاطي (incompressible flow) و هو ان:

تركيب معقد لسريان غير انضغاطي و لا دوراني (incompressible, irrotational flow) يمكن ان يجمع (synthesized) من

سرايين اساسية (elementary flows)

بالتالي سننظر إن شاء الله الى بعض السرايين اساسية (elementary flows) التي تلائم (satisfy) مع معادلة Laplace (Laplace's equation).

Uniform flow

السريان المتماثل

$$\phi = V_{\infty} x$$

Source flow

السريان المصدر

$$\phi = \frac{\Lambda}{2\pi} \ln r$$

Vortex flow

السريان الدوامة

$$\phi = -\frac{\Gamma}{2\pi} \theta$$

In [Wendt et. al. 2009] there are two methods described which use these elementary flows:

- Non-lifting Flows Over Arbitrary Two-Dimensional Bodies: The Source Panel Method
- Lifting Flows Over Arbitrary Two-Dimensional Bodies: The Vortex Panel Method

Also the application “The Aerodynamics of Drooped Leading-Edge Wings Below and Above Stall” is described.

20 (Fluid Dynamic Equations) لمعادلات ديناميك الموائع (Mathematical Properties) الخصائص الرياضية ()

كثير من المضمون مأخوذ من

Chapter 4 (Mathematical Properties of Fluid Dynamic Equations) [Wendt et. al. 2009],

20.1 مدخل

المعادلات الأساسية من ديناميك الموائع التي استخلصت في الباب الثاني من الكتاب هي إما في الشكل التفاضلي (differential form) أو الشكل التكاملية (integral form).

أمثلة:

$$\frac{\partial}{\partial t} \iiint_V \rho \, dV + \iint_S \rho \vec{V} \cdot \vec{dS} = 0$$

الشكل التكاملية لمعادلة الاستمرارية:

الشكل تفاضلي الجزئي (Partial differential form) لمعادلات كمية التحرك (Momentum equations):

$$\text{x-component: } \rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$$

$$\text{y-component: } \rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$$

$$\text{z-component: } \rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$$

المعادلات الأساسية في شكل من الأشكال التفاضلية الجزئية مثل المعادلات a-c 2.36 فوق هي الشكل الأكثر شيوعاً و استخداماً في ديناميات الموائع الحاسوبية (CFD). لذلك قبل ان ندرس الطرق العددية (numerical methods) من اجل حل هذه المعادلات فمن المفيد معالجة بعض الخصائص الرياضية للمعادلات التفاضلية الجزئية نفسها. و ينبغي لأي حل عددي صحيح للمعادلات ان يحمل خاصية طاعة الخصائص الرياضية العامة للمعادلات الأساسية.

ادرس المعادلات الاساسية لحركت الموائع مثلما استنتج من الفصل الثاني (Chap. 2). لاحظ انه في جميع الحالات المشتقات (derivates) الاعلى ترتيباً تحدث بطريقة خطية (linear). أي لا توجد منتجات (products) او اسّية (exponentials) للمشتقات (derivates) الاعلى ترتيب - تظهر من تلقاء نفسها, مضروبة بالمعامل (coefficients) التي هي لنفسها دالات (functions) للمتغيرات التابعة (dependent variables)؛ يسمى مثل هذا النظام للمعادلات بالنظام الشبه خطي (quasilinear system). على سبيل المثال لسرايين اللالزجية (inviscid flows) نجد عندما ندرس المعادلات الموجودة في القسم 2.7.2 ان المشتقات ذات الترتيب الاعلى (highest order derivates) هي ذات الدرجة الاولى (first order) وكلها تظهر خطياً (linearly).

ولسرايين اللزجية (viscid flows) نجد عندما ندرس المعادلات الموجودة في القسم 2.7.1 ان المشتقات ذات الترتيب الاعلى هي ذات الدرجة الثانية (second order) وكلها تظهر خطياً (linearly). لهذا السبب في المقطع التالي سندرس بعض الخصائص لنظام (system) شبه خطي للمعادلات التفاضلية الجزئية (quasilinear partial differential equations). في هذه العملية سوف نقوم بوضع تصنيف لثلاثة انواع من المعادلات التفاضلية الجزئية- و كل من الثلاثة تلاقت في ميكانيكا الموائع (fluid dynamics).

20.2 بعض المعادلات التفاضلية الجزئية

التالي مؤخوذ من كتاب [1]:

1- معادلة التوصيل الحراري في البعد الواحد :

$$u_t = u_{xx}$$

2- معادلة التوصيل الحراري في البعدين :

$$u_t = u_{xx} + u_{yy}$$

3- معادلة لابلاس بالإحداثيات القطبية :

$$u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta} = 0$$

4- معادلة الموجة في الأبعاد الثلاثة :

$$u_{tt} = u_{xx} + u_{yy} + u_{zz}$$

5- معادلة الإرسال البرقي :

$$u_{tt} = u_{xx} + \alpha u_t + \beta u$$

20.3 تصنيف (Classification) المعادلات التفاضلية الجزئية (Partial Differential Eq.s)

للتبسيط لنعتبر نظام (system) بسيط نسبياً لمعادلات الشبه خطية. فهي لن تكون معادلات السريان لكنها تشبهها في بعض النواحي. فان هذا القسم هو مثال مبسط. لنعتبر نظام المعادلات الشبه خطي الواردة ادناه:

$$(4.1a) \quad a_1 \frac{\partial u}{\partial x} + b_1 \frac{\partial u}{\partial y} + c_1 \frac{\partial v}{\partial x} + d_1 \frac{\partial v}{\partial y} = f_1$$

$$(4.1b) \quad a_2 \frac{\partial u}{\partial x} + b_2 \frac{\partial u}{\partial y} + c_2 \frac{\partial v}{\partial x} + d_2 \frac{\partial v}{\partial y} = f_2$$

حيث u و v هي المتغيرات التابعة، الدالات ل x و y .

و المعامل (coefficients) $a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, f_1, f_2$ تستطيع ان تكون دالات ل x, y, u و v .

لنعتبر اي نقطة في مستو xy . دعونا نبحث عن خطوط (او اتجاهات) من خلال هذه النقطة (ان وجدت) حيث المشتقات ل u و v تكون غير محددة (indeterminant) على طول هذه الخطوط (او اتجاهات). و عبرها ربما تكون متقطعة (discontinuous).

هذه الخطوط تسمى **الخطوط الخصائية (characteristic lines)**.

Quasilineare partielle Differentialgleichungen 2.Ordnung mit zwei unabhängigen Variablen können in drei Typen unterteilt werden: hyperbolisch, parabolisch und elliptisch. Diese Einteilung basiert auf Eigenschaften von Charakteristiken-Linien, entlang welcher sich die Informationen über die Lösung ausbreiten. Jede derartige Gleichung hat zwei Sätze von Charakteristiken . Die verschiedenen Eigenschaften der Gleichungen können verschiedenen Strömungstypen zugeordnet werden. [3], p.20

للتثور على هذه **الخطوط الخصائية** نفترض ان u و v مستمرة (continuous)؛ و بالتالي:

$$(4.2a) \quad du = \frac{\partial u}{\partial x} dx + \frac{\partial u}{\partial y} dy : u = u(x,y)$$

$$(4.2b) \quad dv = \frac{\partial v}{\partial x} dx + \frac{\partial v}{\partial y} dy : v = v(x,y)$$

المعادلات (4.1) و (4.2) تشكل نظاماً من اربعة معادلات خطية (linear) مع اربعة مجهولات (matrix form) على النحو يمكن كتابة هذه المعادلات بشكل مصفوفة

التالي:

$$(4.3) \quad \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ dx & dy & 0 & 0 \\ 0 & 0 & dx & dy \end{bmatrix} \begin{bmatrix} \partial u / \partial x \\ \partial u / \partial y \\ \partial v / \partial x \\ \partial v / \partial y \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ du \\ dv \end{bmatrix}$$

دعونا نرمز ب $[A]$ مصفوفة المعامل (coefficient matrix):

$$[A] = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ dx & dy & 0 & 0 \\ 0 & 0 & dx & dy \end{bmatrix}$$

علاوة على ذلك ترك $|A|$ تكون المحددة (determinant) ل $[A]$ من قاعدة كرامر (Cramer's rule)، اذا كانت $|A| \neq 0$ ، عندها نستطيع الحصول على حلول وحيدة (unique) ل $\partial u / \partial x, \partial u / \partial y, \partial v / \partial x$ و $\partial v / \partial y$.
و من ناحية اخرى، اذا كانت $|A| = 0$ ، عندها تكون $\partial u / \partial x, \partial u / \partial y, \partial v / \partial x$ و $\partial v / \partial y$ في الحالة الافضل، غير محددة (indeterminant). نحن نبحث عن اتجاهات محددة (particular) في المستوى (plane) التي على طولها المشتقات ل u و v هي غير محددة. لذلك دعونا نجعل $|A| = 0$ ، و ننظروا ماذا سسيجري.

$$\begin{vmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ dx & dy & 0 & 0 \\ 0 & 0 & dx & dy \end{vmatrix} = 0$$

لذلك

$$(4.4) \quad (a_1 c_2 - a_2 c_1)(dy)^2 - (a_1 d_2 - a_2 d_1 + b_1 c_2 - b_2 c_1)(dx)(dy) + (b_1 d_2 - b_2 d_1)(dx)^2 = 0$$

اقسم المعادلة (4.4) على $(dx)^2$:

$$(4.5) \quad (a_1 c_2 - a_2 c_1) \left(\frac{dy}{dx} \right)^2 - (a_1 d_2 - a_2 d_1 + b_1 c_2 - b_2 c_1) \frac{dy}{dx} + (b_1 d_2 - b_2 d_1) = 0$$

المعادلة (4.5) هي معادلة من الدرجة الثانية (quadratic equation) في dy/dx .

لأية نقطة في المستوى xy ، حل المعادلة (4.5) ستعطي الانحدارات (slopes) على طول الخطوط تلك المشتقات (derivatives) لـ u و v هي غير محددة. هذه الخطوط في الفضاء xy على طولها تسمى مميزات الخطوط (characteristic lines) لنظام المعادلات الذي قدمت بـ (4.1a) و (4.1b)

في المعادلة (4.5) دع

$$\begin{aligned} a &= (a_1c_2 - a_2c_1) \\ b &= -(a_1d_2 - a_2d_1 + b_1c_2 - b_2c_1) \\ c &= (b_1d_2 - b_2d_1) \end{aligned}$$

و من ثم يمكن كتابة المعادلة (4.5) كما يلي:

$$(4.6) \quad a \left(\frac{dy}{dx} \right)^2 + b \left(\frac{dy}{dx} \right) + c = 0$$

لهذا السبب من الصيغة التربيعية (quadratic formula):

$$(4.7) \quad \frac{dy}{dx} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

المعادلة (4.7) تعطي اتجاه الخطوط المميزة (characteristic lines) خلال النقطة معينة (given) في مستوى xy . هذه الخطوط لها طبيعة مختلفة، تعتمد على قيمة المميز (discriminant) في المعادلة (4.7). ندل على المتميز بـ D .

$$(4.8) \quad D = b^2 - 4ac$$

قد تكون الخطوط المميزة (characteristic lines) حقيقية (real) و مختلفة، او حقيقية و متساوية، او تخيلية (imaginary)، اعتماداً على قيمة D . خصوصاً:

إذا كانت $D > 0$:

يوجد خطان حقيقيان و متخالفين خلال كل نقطة في المستوى xy . عندما يكون في هذه الحالة، فان نظام المعادلات

المقدم من (4.1 a, b) يسمى قطعي زائدي (hyperbolic)

إذا كانت $D = 0$:

يوجد خاصة (characteristic) حقيقية واحدة. عندما يكون في هذه الحالة، فان نظام المعادلات المقدم من (4.1 a, b) يسمى تقسي مكافئي (parabolic)

اذا كانت $D < 0$:

الخطوط المميزة هي خيالية. يكون في هذه الحالة، فان نظام المعادلات المقدم من (4.1 a, b) يسمى الاهليجية / بيضاوي الشكل (elliptic).

تصنيف المعادلات التفاضلية الجزئية الشبه خطية بأنها الاهليجية (elliptic)، قطعية مكافئة (parabolic) او قطعية زائدة (hyperbolic) هو تصنيف عام في هذا النوع من المعادلات. هذه الفئات الثلاثة من المعادلات لديها سلوك مختلف تماماً. أصل الكلمات: الاهليجي (elliptic)، قطعي مكافئ (parabolic) و قطعي زائد (hyperbolic) هو ببساطة تشابه مباشر مع الحالة للاقسام المخروطية (conic sections).

المخروط (cone):



شكل ثلاثي الأبعاد له قاعدة دائرية ورأس واحد. ويصل بالرأس سطح منحن.

المعادلات العامة للاقسام المخروطية من الهندسة التحليلية (analytic geometry) هي:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

حيث اذا

$b^2 - 4ac > 0$ ، القسم المخروطي هو قطع زائد (hyperbola)

$b^2 - 4ac = 0$ ، القسم المخروطي هو قطع مكافئ (parabola)

$b^2 - 4ac < 0$ ، القسم المخروطي هو قطع ناقص (ellipse)

التالي مؤخوذ من كتاب [1]:

$$u_t = u_{xx} \quad (1)$$

(1) هي معادلة تفاضلية جزئية تصف التوصيل الحراري (heat transfer) في البعد الواحد.

كل معادلة تفاضلية جزئية خطية مثل (1) تمثل أحد الأنماط الآتية :

- أ- القطع المكافئ .
- ب- القطع الزائد .
- ج- القطع الناقص .

فمعادلات القطع المكافئ تصف سريان الحرارة وعمليات الانتشار وتحقق

الخاصية :

$$b^2 - 4ac = 0$$

ومعادلات القطع الزائد تصف حركات الاهتزاز وحركات الموجة وتحقق

الخاصية :

$$b^2 - 4ac > 0$$

ومعادلات القطع الناقص تصف ظواهر الحالة المستقرة وتحقق الخاصية :

$$b^2 - 4ac < 0$$

أمثلة

أ- $u_t = u_{xx}$ معادلة قطع مكافئ لأن : $B^2 - 4AC = 0$

ب- $u_{tt} = u_{xx}$ معادلة قطع زائد لأن : $B^2 - 4AC = 4$

ج- $u_{\eta\eta} = 0$ معادلة قطع زائد لأن : $B^2 - 4AC = 1$

د- $u_{xx} + u_{yy} = 0$ معادلة قطع ناقص لأن $B^2 - 4AC = -4$

هـ - $y_{xx} + u_{yy} = 0$ $B^2 - 4AC = -4y$ $\left\{ \begin{array}{l} \text{عندما } y > 0 \text{ يكون قطع ناقص} \\ \text{عندما } y = 0 \text{ يكون قطع مكافئ} \\ \text{عندما } y < 0 \text{ يكون قطع زائد} \end{array} \right.$

(في حالة المعاملات المتغيرة يتغير الوضع من نقطة إلى أخرى).

ملاحظات

1- بصورة عامة يكون $B^2 - 4AC$ دالة بدلالة المتغيرات المستقلة وعليه تتغير المعادلة

من نمط لآخر تبعاً لمجال المتغيرات (ولو أن ذلك غير مألوف).

2- إن المعادلة الخطية العامة (1) قد صيغت بدلالة المتغيرات المستقلة x, y . في معظم

المسائل يمثل أحد المتغيرين الزمن وعندئذ يمكن كتابة المعادلة بدلالة x, t .

3- يمكن توضيح مخطط التصنيف العام كما في شكل (1-2).

خطية				غير خطية			الخطية
1	2	3	4	5	...	m	الرتبة
معاملات ثابتة				معاملات متغيرة			معاملات (المعادلات الخطية)
متجانسة				غير متجانسة			التجانس (المعادلات الخطية)
1	2	3	4	5	...	n	عدد المتغيرات
قطع زائد			قطع مكافئ		قطع ناقص		الأنماط الأساسية

نلاحظ بالنسبة للمعادلات التفاضلية الجزئية القطع الزائد (hyperbolic PDEs)، ان يكون هناك الميزتين (characteristics) حقيقية (real) و مختلفة (distinct)، تتيح تطوير طريقة الحل تصل الى حل جاهز لهذه المعادلات. اذا عدنا الى المعادلة (4.3) حاولنا حلها ل $\partial u / \partial y$ باستخدام طريقة كرامر (Cramer's rule)، نصل الى:

$$\partial u / \partial y = \frac{|N|}{|A|} = \frac{0}{0}$$

حيث محددة العداد (numerator determinant) هي:

$$(4.9) \quad |N| = \begin{vmatrix} a_1 & f_1 & c_1 & d_1 \\ a_2 & f_2 & c_2 & d_2 \\ dx & du & 0 & 0 \\ 0 & dv & dx & dy \end{vmatrix}$$

السبب لماذا $|N|$ يجب ان تكون صفر هو ان $\partial u / \partial y$ غير محدد، بالشكل 0/0. بما ان $|A|$ هي مسبقاً وصلت الى صفر، اذاً $|N|$ يجب ان تكون صفر للسماح بان تكون $\partial u / \partial y$ غير محددة.

ان توسيع (expansion) المعادلة (4.9) ستؤدي الى معادلات التي تنطوي على متغيرات مجال السريان (flow field variables) التي هي معادلات تفاضلية عادية (ordinary differential equations)، و في بعض الحالات هي معادلات جبرية (algebraic equations). هذه المعادلات التي تتم الحصول عليها من (4.9) تسمى بمعادلات التوافق (compatibility equations) و هي تستمر فقط على الخطوط المميزة (characteristic lines). هذا هو جوهر من

حل المعادلة التفاضلية القطع الزائد الاصلية (original hyperbolic PDE):

فقط ضع معادلات ابسط - معادلات تفاضلية عادية (ordinary differential equations) (و هي معادلات التوافق (compatibility equations)) - على طول الخطوط المميزة في المستوى xy . هذه الطريقة تسمى طريقة الخصائص (method of characteristics). هذه الطريقة تطورت بدرجة عالية لحل السريان اللا لزجي الفوق صوتي (inviscid supersonic flows). لهذا النوع من السريان المعادلات الاساسية تكون من نوع المعادلات التفاضلية القطع الزائد. طريقة الخصائص هي اسلوب كلاسيكي من اجل حل السريان اللا لزجي الفوق صوتي.

20.4 السلوك العام للاصناف المختلفة من المعادلات التفاضلية الجزئية و علاقتها بديناميات الموائع

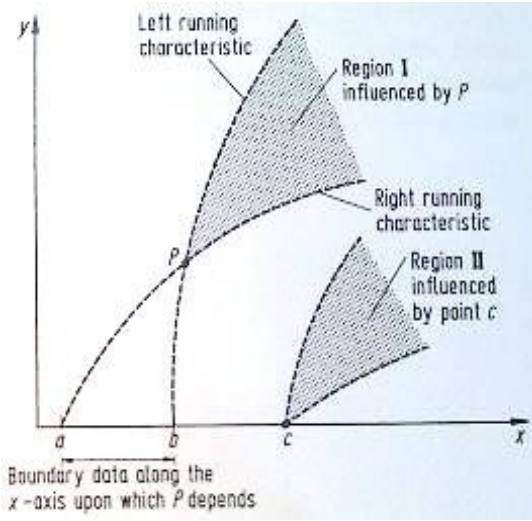
في هذا القسم، نناقش ببساطة ودون براهين رياضية، بعض من سلوك المعادلات تفاضلية القطع الزائد (hyperbolic)، القطع المكافئ (parabolic) والقطع الناقص (elliptic)، و سنعلق هذا السلوك بحل مشاكل من ميدان ديناميات الموائع.

20.4.1 المعادلات القطع الزائد (Hyperbolic Equations)

للمعادلات القطع الزائد المعلومات في نقطة معينة P تؤثر فقط على تلك المناطق بين الخصائص التي تتقدم (advancing characteristics). على سبيل المثال، دراسة الرسمة 4.1، التي رسمت لمشكلة ثنائية الابعاد (two-dimensional) مع اثنين من المتغيرات المستقلة الفضائية (independent space variables).

النقطة P تقع في مكان معين (x,y) . لتأمل الخصائص التي تجري الى اليمين و الى الشمال (right running and left running characteristic) كما يبين الرسم 4.1

الشكل 4.1:



مجال (domain) و حدود لحل المعادلات القطع الزائد (hyperbolic equations). سريان ثابت (steady) ثنائي الابعاد (Two-dimensional).
الرسم مأخوذ من [2].

المعلومات عند النقطة P لا تؤثر (influences) الا على المنطقة المظلمة - المنطقة المصنفة ب I بين الخصائص الاثنتين التي تتقدم (two advancing characteristics) خلال نقطة P. و هذا له تأثير مباشر على شروط الحدود (boundary conditions) للمعادلات القطع الزائد. لنفترض ان المحور x (x-axis) هو شرط حدودي (boundary condition) للمشكلة، يعني المتغيرات التابعة u و v معروفة على طول المحور x. هنالك الحل ممكن الحصول عليه عبر "السير الى الامام" ('marching forward') في المسافة y، بدءاً من حدود معينة. و مع ذلك، فان الحل ل u و v في النقطة P تعتمد فقط على جزء من الحدود بين a و b ، كما نبين في الرسم 4.1.

المعلومة عند النقطة c التي هي خارج الفاصل (interval) ab، هي تنتشر على طول الخصائص الى c، و تأثير فقط على المنطقة II. النقطة P هي خارج المنطقة II، و بالتالي لا تلمس معلومات من النقطة c. لهذا السبب النقطة P تعتمد فقط على الجزء من الحدود الذي يتم حصره بين الخصائص الاثنتين التي تذهب من خلال النقطة P و تعترض الحدود لتحديد الفاصل ab.

في ديناميكية الموائع، الانواع التالية من السريان هي محددة من المعادلات التفاضلية الجزئية القطع الزائد (hyperbolic PDEs)، و بالتالي يعرض السلوك المذكور آنفاً:

السريان الثابت اللالزجي الفوق الصوتي (Steady, inviscid supersonic flow).

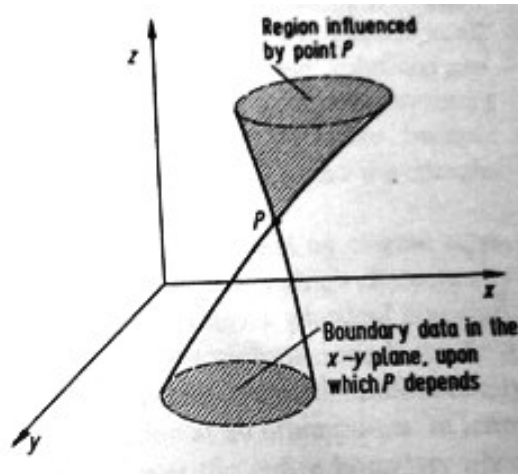
اذا كان السريان في ثنائي الابعاد (two-dimensional) فبالتالي السلوك هو مثل المعروض في الشكل 4.1.

اذا كان السريان ثلاثي الابعاد، هناك مساحات مميزة في المستوى xyz، كما رسمت في الشكل 4.2.

لنعتبر النقطة P في مكان محدد في المستوى (x,y,z) . المعلومات عند P تؤثر على الحجم المظلل في المساحة المميزة التي تتوسع. بالإضافة الى ذلك، اذا كان المستوى xy هو سطح جداري (boundary surface)، عندها فقط ذلك الجزء من الجدار المحصورة من قبل السطح المميز المتراجع، الذي يؤثر على P .

في الشكل 4.2، نُحل المتغيرات التابعة من خلال البدء بالمعطيات (data) في المستوى xy ، و بالتالي بـ "السير" في الاتجاه z .

لمشكلة سريان فوق الصوتي لا لزج (inviscid supersonic flow problem)، الاتجاه العام للسريان يكون ايضاً الاتجاه z .



الشكل 4.2:

المجال و الحدود (Domain and boundaries) لحل المعادلات القطع الزائد.

سريان ثابت ثلاثي الابعاد

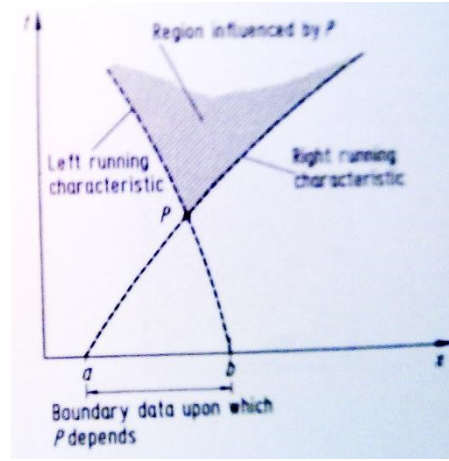
(Three-dimensional steady flow)

سريان متغير انضغاطي لا لزج (Unsteady inviscid compressible flow).

لتغيير سريان لا لزج من بعد واحد او ثنائي الابعاد، المعادلات الاساسية هي من نواف القطع الزائد، لا يهم ما اذا كان السريان هو محلياً (locally) تحت سرعة الصوت (subsonic) او فوق صوتي (supersonic). هنا الوقت هو اتجاه سير الحساب (marching direction).

للسريان اللا لزج من بعد واحد، لننظر الى النقطة P من المستوى (x,t) المبين في الشكل 4.3.

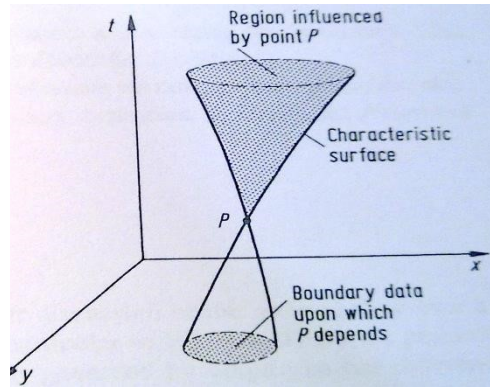
مرة اخرى، المنطقة المتأثرة بالنقطة P هي المنطقة المظللة الواقعة بين اثنين من الخصائص التي تتقدم من خلال P ، و الفاصل ab هو الجزء الوحيد من الحدود على طول المحور x الذي يعتمد عليه الحل في النقطة P .



الشكل 4.3:

المجال (Domain) و الحدود من اجل حل المعادلات القطع الزائد. سريان متغير من بعد واحد (One-dimensional unsteady flow).

للسريان اللا لزجي الثنائي الابعاد (two-dimensional)، نعتبر النقطة P في المستوى (x,y,t) كما هو مبين في الشكل 4.4. بدءاً بالبيانات الاولية المعروفة في المستوى x,y ، الحل "يسير" ('marches') الى الامام في الوقت (time).



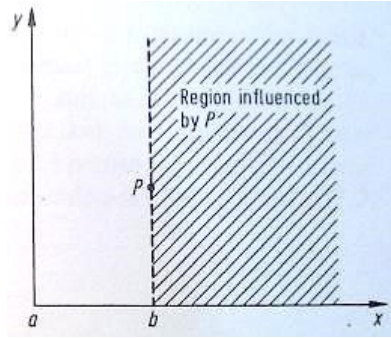
الشكل 4.4:

المجال و الحدود لحل المعادلات القطع الزائد. سريان غير ثابت ثنائي الابعاد (Two-dimensional unsteady flow)

20.4.2 معادلات القطع مكافئة / Parabolic Equations

للمعادلات القطع المكافئة، المعلومات عند النقطة P في المستوى xy تؤثر على كل المنطقة من المستوى الى جهة واحدة من P. هذا هو مرسوم في الشكل 4.5، حيث تم رسم خط مميز واحد من خلال النقطة P.

لنفترض ان المحور x و المحور y تشكل حدود. الحل عند P يتأثر بشروط الحدود على المحور y بكامله، فضلاً عن الجزء في المحور x من a الى b . حلول المعادلات القطع المكافئ هي ايضاً حلول "مسيرة" ('marching')؛ بدءاً بالشروط الحدودية (boundary conditions) على طول كل من المحاور x و y ، يتم الحصول على حل لمجال السريان عبر "مسيرة" في الاتجاه العام x .



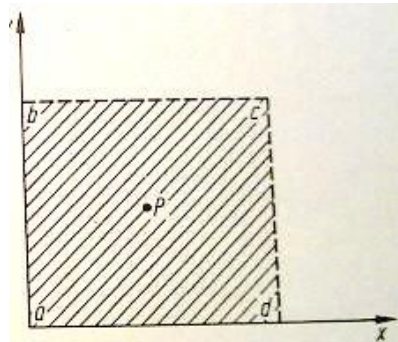
الشكل 4.5:

المجال و الحدود من لحل المعادلات القطع المكافئ (parabolic equations) في بعدين (in two dimensions).

في ديناميكيا الموائع، هناك اشكال مخفضة (reduced forms) من معادلات ناوير- ستوكس (Navier-Stokes) التي تمثل سلوك من نوع القطع المكافئ. اذا تم تجاهل شروط الاجهاد اللزجي (viscous stress) التي تنطوي على المشتقات بالنسبة الى x في هذه المعادلات، نحن نحصل على المعادلات ناوير- ستوكس (Navier-Stokes) القطعي المكافئ ('parabolized' Navier-Stokes equations)، التي تسمح حل بسير الى الوراء في اتجاه x ، بدءاً من بعض المعطيات المنصوص عليها على طول المحاور x و y . المزيد من الخفض لمعادلات ناوير- ستوكس (Navier-Stokes) لأعداد رينولز (Reynolds numbers) العالية تؤدي الى معادلات الطبقة الجدارية (boundary layer equations) التي هي معروفة جيداً. هذه الطبقة الجدارية (boundary layer equations) تُظهر السلوك القطع المكافئ في الشكل 4.5.

20.4.3 المعادلات القطع الناقص (elliptic equations)

للمعادلات القطع الناقص، المعلومات عند النقطة P في المستوى xy تؤثر على كل المناطق الاخرى للمجال (domain). رسمت هذه الصورة في الشكل 4.6، الذي يري مجال مستطيل الشكل (rectangular).

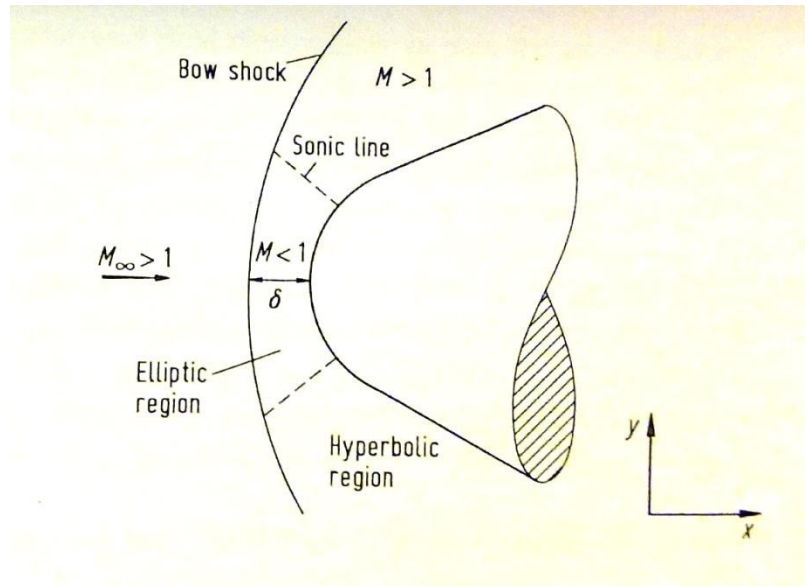


الشكل 4.6:

المجال و الحدود لحل معادلات القطع الناقص بعدين (two dimensions).

هنا المجال هو مغلق تماماً، تحيط بها الحدود المغلقة $abcd$. للمعادلات القطع الناقص، لان النقطة P تؤثر على كل النقاط في المجال، و ايضاً الحل عند النقطة P يتأثر بكامل الحدود (boundary) المغلق $abcd$. لذا، يجب إتمام الحل عند النقطة P في آن واحد مع إتمام الحل في جميع نقاط المجال. هذا يكون في تباين شديد مع "سير" ('marching') الحلول المناسبة للمعادلات القطع الزائد و القطع المكافئ.

في ديناميكا الموائع السريان الثابت (steady)، الذي هو ما دون سرعة الصوت (subsonic)، الا لزجي (inviscid) هو يوافق لمعادلات القطع الناقص. هذا ايضاً يتضمن السريان اللا انضغاطي (incompressible) (الذي يتضمن نظرياً عدد ماخ (Mach number) يساوي صفر). اذاً، لهذه الانواع من السرايين، يجب تطبيق الشروط الجدارية (boundary conditions) الفيزيائية تحيط كاملاً بالسريان، و حل ميدان السريان (flow-field) في كل النقاط في السريان يجب ان تُحصل عليه في نفس الوقت (simultaneously)، لأن الحل عند نقطة معينة يؤثر على حل كل النقاط الاخرى. من حيث الشكل 4.6، يجب ان تطبق الشروط الجدارية على الجدار $abcd$ بأكمله. هذه الشروط الجدارية (boundary conditions) يمكنها ان تأخذ الاشكال التالية: تحديد المتغيرات التابعة (u dependent variables) و v على طول الجدار. هذا النوع من الشروط الجدارية تسمى شرط ديريشلت (Dirichlet condition). و تحديد (specification) المشتقات (derivatives) للمتغيرات التابعة u و v مثل $\partial u / \partial y$ على طول الجدار. هذا النوع من الشروط الجدارية يسمى شرط نيومان (Neumann condition).



في هذه المرحلة سيكون مهم للطالب، حل الشكل المغلق لبعض المعادلات التفاضلية الجزئية (PDE) الخطية من الانواع القطع الزائد (*hyperbolic*)، والقطع المكافئ (*parabolic*) والقطع الناقص (*elliptic*). لهذا انظر كتب لمادة الرياضيات.

20.4.5 طرح المشاكل بشكل جيد / Well-Posed Problems

في الحل للمعادلات التفاضلية الجزئية هو من السهل في بعض الاحيان التوصل الى حل باستعمال شروط اولية (initial conditions) و جدارية (boundary) غير صحيحة او غير كافية. مثلاً "سوء طرح" المشكلة تؤدي عادة الى نتائج زائفة (مزورة). لذلك نحن نعرف مشكلة مطروحة بشكل جيد كما يلي: اذا كان الحل لمعادلة تفاضلية جزئية موجودة و فريدة (*unique*)، و اذا كان الحل يعتمد باستمرار على الشروط الجدارية الاولية، بالتالي المشكلة تكون مطروحة بشكل جيد.

20.4.6 المراجع

[1] رس فارلو، المعادلات التفاضلية الجزئية (ترجمة: د. هها عواد الكبيسي)، منشورات جامعة عمر المختار، البيضاء، 2005

[2] [Wendt et. al. 2009], Chapter 4 (Mathematical Properties of Fluid Dynamic Equations)

[3] Ferziger, Peric, "Numerische Strömungsmechanik", Springer-Verlag Berlin Heidelberg 2008

21 Discretization of PDEs (تفريز لمعادلات التفاضلية الجزئية)

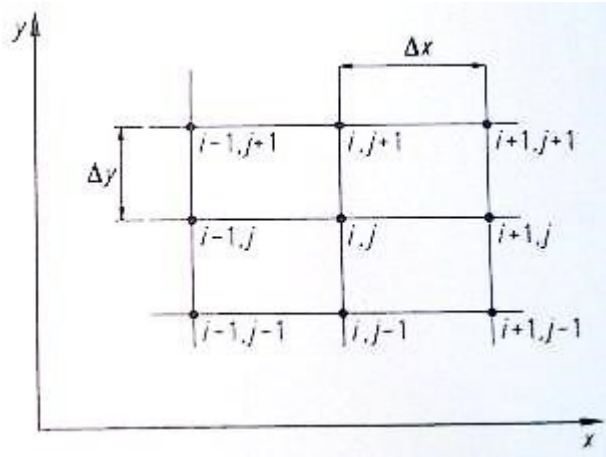
معظم المضمون مأخوذ من

Chapter 5 (Discretization of Partial Differential Equations) [Wendt et. al. 2009],

21.1 مدخل

حلول تحليلية (Analytical solutions) لمعادلات التفاضلية الجزئية (PDEs) تعطي تعبيرات مقفولة الشكل التي تعطي التغيرات للمتغيرات التابعة (dependent variables) على المجال (domain) بشكل مستمر (continuously). مقارنة مع ذلك، الحلول العددية (numerical solutions) تستطيع أن تُجيب على نقاط منفصلة (discrete points) في المجال فقط، و تسمى نقاط الشبكة (grid points).

الشكل 5.1: نقاط الشبكة المنفصلة



على سبيل المثال، انظر في الشكل 5.1، مما يري جزء من شبكة منفصلة في المستوى xy . لنفترض أن تباعد نقاط الشبكة في اتجاه x هو موحد (uniform)، والتي تقدمها Δx ، وهذا التباعد في اتجاه y هو أيضا موحد (uniform)، والتي تقدمها Δy ، كما هو مبين في الشكل 5.1. بشكل عام، Δx و Δy يكونان مختلفين. ومع ذلك، فإن الغالبية العظمى من التطبيقات CFD تنطوي على حلول عددية على الشبكة بتباعد موحد (uniform spacing) في كل اتجاه، لأن هذا يبسط إلى حد كبير برمجة الحل، ويوفر مساحة التخزين الحاسوبي ويعطي نتائج عادة دقيقة أكثر.

هذا التباعد الموحد لا يجب أن يحدث في الفضاء xy الفيزيائي (physical xy space)؛ كما هو الحال في كثير من الأحيان في CFD، وتجري الحسابات العددية (numerical calculations) في الفضاء الحسابي (computational space) المتحول التي لديها تباعد موحد (uniform spacing) في المتغيرات المستقلة المتحولة (transformed independent variables)، ولكن الذي يتوافق مع التباعد غير الموحد (non-uniform spacing) في المستوى

الفيزيائي (physical plane). في أي حال، في هذا الفصل إننا نفترض التباعد الموحد في كل اتجاه النظام الإحداثي (coordinate system)، ولكن ليس بالضرورة متساوية التباعد (equal spacing) لكلا الاتجاهين، أي سنتخذ Δx و Δy من الثوابت (constants)، ولكن هذا ليس من الضروري أن تكونا Δx و Δy على قدم المساواة. عودة إلى الشكل 5.1، يتم تحديد نقاط الشبكة وفقاً لمؤشر i (index) الذي يمتد في اتجاه x ، ومؤشر j (index) الذي يمتد في اتجاه y . وبالتالي، إذا كان (i, j) هو مؤشر (index) لنقطة P في الشكل 5.1، ثم النقطة على يمين P تعرّف بأنها $(i+1, j)$ ، وهذه النقطة الأعلى منها مباشرة هي $(i, j+1)$ الخ.

تستخدم طريقة الفروق المحدودة (finite differences) على نطاق واسع في CFD، وبالتالي سيتم تخصيص معظم هذا الفصل على المسائل المتعلقة بالفروق المحدودة (finite differences).

فلسفة الفروق المحدودة (finite differences) هو استبدال المشتقات الجزئية (partial derivatives) التي تظهر في المعادلات الأساسية لميكانيكا الموائع (governing equations of fluid dynamics). مع فرق للمقسومات الجبرية (algebraic difference quotients)، ينتج نظام من المعادلات الجبرية (system of algebraic equations) التي يمكن حلها لمتغيرات حقل السريان (flow-field) في النقاط المعينة من الشبكة المنفصلة في السريان (كما هو موضح في الشكل 5.1). دعونا نتقل الآن للحصول على بعض من فرق لمقسومات الجبرية (algebraic difference quotients) الأكثر شيوعاً التي تستخدم لتفريز (discretize) المعادلات التفاضلية الجزئية (PDE).

21.2 اشتقاق مقسومات لفرق محدودة ابتدائية (Elementary Finite Difference Quotients)

يقوم تمثيل الفروق المحدودة (Finite difference) للمشتقات (derivatives) على أساس توسعات سلسلة تايلر (Taylor's series expansions). على سبيل المثال، إذا $u_{i,j}$ يدل على مكون x (component) للسرعة (velocity) في نقطة (i, j) ، إذاً السرعة (velocity) $u_{i+1,j}$ في النقطة $(i+1, j)$ يمكن أن تعبر عنها الأطراف الرياضية من توسعات سلسلة تايلر (Taylor's series expansions) حول النقطة (i, j) ، على النحو التالي:

$$u_{i+1,j} = u_{i,j} + \left(\frac{\partial u}{\partial x}\right)_{i,j} \Delta x + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j} \frac{(\Delta x)^3}{6} + \dots \quad (5.1)$$

المعادلة (5.1) هي رياضياً

تعبير

دقيق عن $u_{i+1,j}$ إذا :

(أ) عدد من الأطراف الرياضية (terms) هي لانهائية (infinite)، و السلسلة تتلاقى (converges)،

(ب) و / أو $\Delta x \rightarrow 0$.

لحسابات العددية (numerical computations)، فإنه من غير العملي إدخال عدد لا حصر له من الأطراف (terms) في المعادلة (5.1). لذلك، المعادلة (5.1) تكون مَقْطُوعَةً (truncated). على سبيل المثال، إذا يتم تجاهل الاطراف الرياضية قيمة الأسية (order of magnitude) $(\Delta x)^3$ و الترتيب الأعلى (higher order)، المعادلة (5.1) تختصر إلى:

$$u_{i+1,j} \approx u_{i,j} + \left(\frac{\partial u}{\partial x} \right)_{i,j} \Delta x + \left(\frac{\partial^2 u}{\partial x^2} \right)_{i,j} \frac{(\Delta x)^2}{2} \dots \dots \dots (5.2)$$

نقول إن المعادلة (5.2) هي في المرتبة الثانية من الدقة (second-order accuracy)، وذلك لأن المصطلح الرياضي للترتيب (terms of order) $(\Delta x)^3$ و الأعلى قد أهملنا. إذا قمنا بإهمال الطرف الرياضي للترتيب $(\Delta x)^2$ (order) و الأعلى، نحصل من المعادلة (5.1)،

حيث المعادلة (5.3) $u_{i+1,j} \approx u_{i,j} + \left(\frac{\partial u}{\partial x} \right)_{i,j} \Delta x$ (5.3) هو من الدرجة الأولى من الدقة.

و (5.3)، إهمال الأطراف الرياضية ذات الترتيب الأعلى تمثل خطأ الاقتران (truncation error) في تمثيل السلسلة المحدودة (finite series). على سبيل المثال، خطأ الاقتران (truncation error) للمعادلة (5.2) هو:

$$\sum_{n=3}^{\infty} \left(\frac{\partial^n u}{\partial x^n} \right)_{i,j} \frac{(\Delta x)^n}{n!}$$

ويمكن تقليل خطأ الاقتران (truncation error) عبر:

(أ) نقل المزيد من الأطراف الرياضية (terms) في سلسلة تايلر (Taylor's series)، أي المعادلة (5.1). هذا يؤدي إلى ارتفاع مستوى

الدقة (accuracy) في تمثيل $u_{i+1,j}$

(ب) تخفيض حجم Δx .

دعونا نعود إلى المعادلة (5.1)، و نحلها لـ $(\partial u / \partial x)_{i,j}$

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} - \underbrace{\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \frac{\Delta x}{2} - \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j} \frac{\Delta x^2}{6} - \dots}_{\text{Truncation error}}$$

او

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O(\Delta x) \quad (5.4)$$

في المعادلة (5.4)، رمز $O(\Delta x)$ هو التدوين الرياضي الشكلي (formal mathematical notation) الذي يمثل حدود رياضية (terms) ذات الترتيب (of-order-of) بالنسبة لـ Δx . المعادلة (5.4) هي عبارة فروقية بالاتجاه الامامي للمشتق $(\partial u / \partial x)$ في النقطة الشبكية (i, j) ذات درجة اولى

(first order forward difference expression for the derivative $(\partial u / \partial x)$ at grid point (i, j)).

المعادلة (5.4) هو تدوين أكثر دقة من المعادلة (5.3)، الذي ينطوي على تدوين "المساواة تقريبا (approximately equal)" ؛ في المعادلة (5.4) ترتيب حجم خطأ الاقتران (truncation error) عُرضت بشكل صريح من قبل تدوين O .

دعونا الآن نكتب توسيع سلسلة تايلر (Taylor's series expansion) لـ $u_{i-1,j}$ ، وُسِّعَت على $u_{i,j}$.

$$u_{i-1,j} = u_{i,j} + \left(\frac{\partial u}{\partial x}\right)_{i,j} (-\Delta x) + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} \frac{(-\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3}\right)_{i,j} \frac{(-\Delta x)^3}{6} + \dots$$

or,

$$u_{i-1,j} = u_{i,j} - \left(\frac{\partial u}{\partial x} \right)_{i,j} \Delta x + \left(\frac{\partial^2 u}{\partial x^2} \right)_{i,j} \frac{(\Delta x)^2}{2} - \left(\frac{\partial^3 u}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^3}{6} + \dots \quad (5.5)$$

التحليل لـ $(\partial u / \partial x)_{i,j}$ ، يوصلنا الى

$$\boxed{\left(\frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta x} + O(\Delta x)} \quad (5.6)$$

المعادلة (5.6) عبارة فروقية بالاتجاه الامامي للمشتق $(\partial u / \partial x)$ في النقطة الشبكية (i, j) ذات درجة اولى

. (first order *rearward* difference expression for the derivative $(\partial u / \partial x)$ at grid point (i, j) .)

دعونا الآن نطرح (subtract) المعادلة (5.5) من (5.1) .

$$u_{i+1,j} - u_{i-1,j} = 2 \left(\frac{\partial u}{\partial x} \right)_{i,j} \Delta x + \left(\frac{\partial^3 u}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^3}{3} + \dots \quad (5.7)$$

نحل المعادلة (5.7) لـ $(\partial u / \partial x)_{i,j}$ ، و نحصل على

$$\boxed{\left(\frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + O(\Delta x)^2} \quad (5.8)$$

المعادلة (5.8) عبارة فروقية مركزية للمشتق $(\partial u / \partial x)$ في النقطة الشبكية (i, j) ذات درجة ثانية (*second order central*)

. (*difference* for the derivative $(\partial u / \partial x)$ at grid point (i, j) .)

للحصول على العبارة الجبرية للاختلاف المحدود للمشتق الجزئي الثاني $(\partial^2 u / \partial x^2)_{i,j}$ تذكر أولاً أن ترتيب مصطلح

الحجم (order-of magnitude) في المعادلة (5.8) يأتي من المعادلة (5.7) ، و ان المعادلة (5.8) يمكن كتابتها بالشكل

$$\left(\frac{\partial u}{\partial x} \right)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - \left(\frac{\partial^3 u}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^2}{6} + \dots \quad (5.9)$$

التالي:

بإستبدال المعادلة (5.9) في (5.1) ، نحصل على

$$\begin{aligned}
 u_{i+1,j} = u_{i,j} &+ \left[\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - \left(\frac{\partial^3 u}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^2}{6} + \dots \right] \Delta x \\
 &+ \left(\frac{\partial^2 u}{\partial x^2} \right)_{i,j} \frac{(\Delta x)^2}{2} + \left(\frac{\partial^3 u}{\partial x^3} \right)_{i,j} \frac{(\Delta x)^3}{6} \\
 &+ \left(\frac{\partial^4 u}{\partial x^4} \right)_{i,j} \frac{(\Delta x)^4}{24} + \dots
 \end{aligned} \tag{5.10}$$

حل المعادلة (5.10) بالنسبة ل

$(\partial^2 u / \partial x^2)_{i,j}$

$$\boxed{\left(\frac{\partial^2 u}{\partial x^2} \right)_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + O(\Delta x)^2} \tag{5.11}$$

، نحصل على

المعادلة (5.11) هي عبارة للفرق الثاني المركزي من درجة ثانية

(*second-order central second difference*) للمشتق (derivative) $(\partial^2 u / \partial x^2)$ في نقطة الشبكة (i, j) . تعابير الفروق

(Difference expressions) للمشتقات من y تُحصل عليها بنفس الطريقة تماماً و النتائج مماثلة تماماً للمعادلات

السابقة للمشتقات x وهم:

$$\begin{aligned}
 \left(\frac{\partial u}{\partial y} \right)_{i,j} &= \frac{u_{i,j+1} - u_{i,j}}{\Delta y} + O(\Delta y) && \text{Forward difference} \\
 \left(\frac{\partial u}{\partial y} \right)_{i,j} &= \frac{u_{i,j} - u_{i,j-1}}{\Delta y} + O(\Delta y) && \text{Rearward difference} \\
 \left(\frac{\partial u}{\partial y} \right)_{i,j} &= \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} + O(\Delta y)^2 && \text{Central difference} \\
 \left(\frac{\partial^2 u}{\partial y^2} \right)_{i,j} &= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} + O(\Delta y)^2 && \text{Central second difference}
 \end{aligned}$$

ومن المثير للاهتمام ان نلاحظ ان الفرق المركزي الثاني (central second difference) المعطى على سبيل المثال عن

طريق المعادلة (5.11) يمكن تفسيره كفرق أمامي (forward difference) للمشتقات الأولى (first derivatives) ، مع

وجود الفرق للوراء (rearward differences) المستخدمة في المشتقات الأولى (first derivatives). إذا اسقاطنا

للتسهيل الرمز O ، لدينا:

$$\begin{aligned}\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} &= \left[\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)\right]_{i,j} \approx \frac{\left(\frac{\partial u}{\partial x}\right)_{i+1,j} - \left(\frac{\partial u}{\partial x}\right)_{i,j}}{\Delta x} \\ \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} &\approx \left[\left(\frac{u_{i+1,j} - u_{i,j}}{\Delta x}\right) - \left(\frac{u_{i,j} - u_{i-1,j}}{\Delta x}\right)\right] \frac{1}{\Delta x} \\ \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} &\approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2}\end{aligned}\quad (5.12)$$

معادلة (5.12) هي نفس حاصل الفرق (difference quotient) مثل المعادلة (5.11). ويمكن استخدام نفس الفلسفة للتوليد بسرعة حاصل الفرق المحدود (finite difference quotient) للمشتقات المختلطة (mixed derivative) $(\partial^2 u / \partial x \partial y)$ في نقطة (i, j) على الشبكة. على سبيل المثال،

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial y} \right) \quad (5.13)$$

في المعادلة (5.13)، اكتب المشتق لـ x كفرق مركزي للمشتقات لـ y ، ومن ثم ضع المشتقات لـ y أيضا في شكل الفرق المركزية (Central differences).

$$\begin{aligned}\frac{\partial^2 u}{\partial x \partial y} &= \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial y} \right) = \frac{\left(\frac{\partial u}{\partial y}\right)_{i+1,j} - \left(\frac{\partial u}{\partial y}\right)_{i-1,j}}{2\Delta x} \\ \frac{\partial^2 u}{\partial x \partial y} &\approx \left[\left(\frac{u_{i+1,j+1} - u_{i+1,j-1}}{2\Delta y} \right) - \left(\frac{u_{i-1,j+1} - u_{i-1,j-1}}{2\Delta y} \right) \right] \frac{1}{2\Delta x} \\ \frac{\partial^2 u}{\partial x \partial y} &\approx \frac{1}{4\Delta x \Delta y} (u_{i+1,j+1} + u_{i-1,j-1} - u_{i+1,j-1} - u_{i-1,j+1})\end{aligned}$$

or

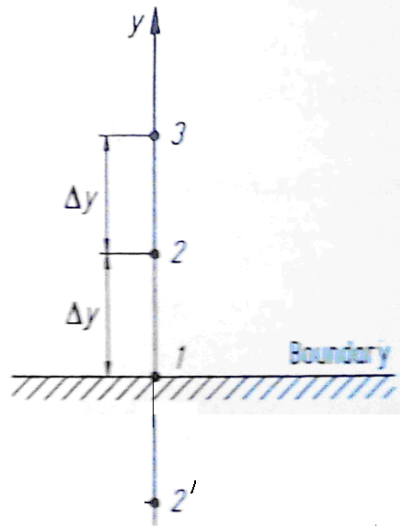
$$\boxed{\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{1}{4\Delta x \Delta y} (u_{i+1,j+1} + u_{i-1,j-1} - u_{i+1,j-1} - u_{i-1,j+1}) + O[(\Delta x)^2, (\Delta y)^2]} \quad (5.14)$$

ويمكن الحصول على العديد من الفروق التقريبية الأخرى للمشتقات (derivatives) أعلاه، فضلا عن المشتقات ذات الترتيب الأعلى (higher-order derivatives) من ذلك. الفلسفة هي نفسها. لجدول مفصل للعديد من أشكال حواصل الفرق (difference quotients)، انظر مثلاً الصفحات 44 و 45 من

Anderson, D.A., Tannehill, John C. and Pletcher, Richard H., Computational Fluid Mechanics and Heat Transfer, McGraw-Hill, New York, 1984.

ماذا يحدث على الحدود (boundary)؟

ماذا يحدث على الحدود (boundary)؟ اي نوع من الفرق (differencing) يكون بالإمكان اذا كان ليس لدينا الا اتجاه واحد لنمشي فيه اي الاتجاه الذي يتباعد عن الحدود (boundary)؟



الشكل 5:2: نقاط للشبكة عند جدار

على سبيل المثال، اعتبر الشكل 5.2، والذي يُوضح جزء من الحدود. مع النقطة 1 من الشبكة (grid) تكون على الحدود. و النقاط 2 و 3 في مسافة (distance) Δy و $2\Delta y$ فوق الحدود. الآن نريد ان نضع تقريب لـ $\frac{\partial u}{\partial y}$ بالفرق المحددة على الحدود.

فمن السهل وضع الفرق الأمامي (forward difference) كما

$$\left(\frac{\partial u}{\partial y}\right)_1 = \frac{u_2 - u_1}{\Delta y} + O(\Delta y) \quad (5.15)$$

التي هي من الدرجة الأولى للدقة (first-order accuracy). لكن، كيف يمكننا الحصول على النتيجة التي هي من الدرجة الثانية للدقة (second-order accuracy)؟

لا نستطيع ان نضع فرق مركزي (central difference) كما هو في المعادلة (5.8) لأنه يتطلب نقطة اخرى وراء الجدار كما هو موضح في النقطة 2' في الشكل 5.2. النقطة 2' هي خارج نطاق الحساب (computation)، وليس لدينا عموماً أي معلومات عن u في هذه النقطة.

في الأيام الأولى من CFD ، هناك العديد من الحلول قد طرحت. مثلاً افترض أن $u_2 = u_2$. ويسمى هذا الشرط جدار الإنعكاسي (reflection boundary). في معظم الحالات، لا معنى مادي (physical sense) لهذا، وبمجرد غير دقيق، إن لم يكن أكثر من ذلك. لذلك نحن نطرح هذا السؤال مرة أخرى، كيف يمكننا العثور على فرق محدود (finite difference) من الدرجة الثانية في الدقة (second-order accurate) على الحدود (boundary)؟ الجواب بسيط، وأنه يوضح طريقة أخرى لاحتساب حواصل الفرق المحدود.

نفترض ان الحدود (boundary) u يمكن ان يعبر عنه متعدد الحدود (polynomial)

$$u = a + by + cy^2 \quad (5.16)$$

اذا طبقنا هذه المعادلة على نقاط الشبكة في الشكل 5.2، نصل الى

$$u_1 = a$$

$$u_2 = a + b\Delta y + c(\Delta y)^2$$

$$u_3 = a + b(2\Delta y) + c(2\Delta y)^2$$

وحل (solving) هذا النظام (system) بالنسبة لـ b :

$$b = \frac{-3u_1 + 4u_2 - u_3}{2\Delta y} \quad (5.17)$$

نعود الى المعادلة (5.16)، وبالمفاضلة (differentiating) نصل الى:

$$\frac{\partial u}{\partial y} = b + 2cy \quad (5.18) \quad \text{و بتقييم (evaluation) المعادلة (5.18) على}$$

الحدود (عند نقطة 1) حيث $y = 0$:

$$\left(\frac{\partial u}{\partial y}\right)_1 = b \quad (5.19)$$

بعد الجمع بين المعادلات (5.18) و (5.19)، نحصل على:

$$\left(\frac{\partial u}{\partial y}\right)_1 = \frac{-3u_1 + 4u_2 - u_3}{2\Delta y} \quad (5.20)$$

لإظهار ترتيب الدقة للمعادلة (5.20) سننظر في توسيع سلسلة تايلر (Taylor's series expansion) حول النقطة 1.

$$u(y) = u_1 + \left(\frac{\partial u}{\partial y}\right)_1 y + \left(\frac{\partial^2 u}{\partial y^2}\right)_1 \frac{y^2}{2} + \left(\frac{\partial^3 u}{\partial y^3}\right)_1 \frac{y^3}{6} + \dots \quad (5.21)$$

قارن المعادلات (5.21) و (5.16). التعبير المتعدد الحدود (polynomial expression) الذي افترضناه في المعادلة (5.16) هو يساوي استخدام أول ثلاث مصطلحات في سلسلة تايلر (Taylor's series). وبالتالي، المعادلة (5.16) هي من $O(\Delta y)^3$ في تشكيل المشتق (derivative) في المعادلة (5.20)، نحن قسمناه بـ Δy ، الأمر الذي يجعل المعادلة (5.20) من نوع $O(\Delta y)^2$ وبالتالي يمكن أن نكتب المعادلة (5.20) كما يلي:

$$\left(\frac{\partial u}{\partial y}\right)_1 = \frac{-3u_1 + 4u_2 - u_3}{2\Delta y} + O(\Delta y)^2 \quad (5.22)$$

وهذا هو حاصل الفرق ذات الدرجة الثانية من دقة (second-order-accurate difference quotient) على الحدود الذي كنا نبحث عنه. كلا المعادلتين (5.15) و (5.22) تسمى الفُرُق من جانب واحد (one-sided differences)، لأنها تعبر عن المشتق (derivative) لدالة (function) في نقطة عن طريق مصطلح رياضي بمتغيرات تابعة التي تعتمد على جانب واحد فقط من هذه النقطة. يمكن تشكيل العديد من فُرُق من جانب واحد (one-sided differences)، بأعلى درجات من الدقة (accuracy)، وذلك باستخدام نقاط إضافية إلى جانب واحد (one side) من الحدود.

21.3 جوانب أساسية لمعادلات الفرق المحدود (Finite-Difference Equations)

الجوهر من حلول عن طريق الفرق المحدودة (finite-difference) في ال CFD هو استخدام المقسومات الفرقية (difference quotients) التي استخرجت في فصل 5.2 بدل المشتقات الجزئية في المعادلات الأساسية لميكانيك الموائع (governing flow equations). النتيجة هي منظومة من معادلات فرقية جبرية (system of algebraic difference equations) للمتغيرات التابعة (dependent variables) في كل نقطة من الشبكة (grid).

في هذا الباب، سندرس بعض الجوانب الأساسية لمعادلة فرقية (a difference equation).

اعتبر المعادلة النموذجية التالية، والتي يفترض فيها أن u هو المتغير التابع (dependent variable) و يكون دالة (function) من x و t .

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad (5.23)$$

نختار هذه المعادلة البسيطة تسهياً للعمل، في هذه المرحلة من مناقشاتنا ليس هناك ميزة يمكن الحصول عليها عن طريق التعامل مع معادلات السريان (flow equations) الأكثر تعقيداً. المعادلة (5.23) هي من نوع القطع المكافئ (parabolic).

إذا قمنا باستبدال مشتق الوقت (time derivative) في المعادلة (5.23) بفارق إلى الأمام (forward difference)، ومشتق المكاني (spatial derivative) مع اختلاف مركزي، نصل إلى النتيجة التالية:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \quad (5.24)$$

سؤال: ما هو خطأ الاقتطاع (truncation error) لمعادلة الفرق محدود كاملة (finite-difference equation)؟

الجمع بين المعادلات (5.23) و (5.24)، وكتابة بشكل واضح أخطاء الاقتطاع (truncation errors) المرتبطة بمحاصل الفرق (difference quotients) من المعادلات (5.4) و (5.10) أصبح لدينا:

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{(u_{i+1}^n - 2u_i^n + u_{i-1}^n)}{(\Delta x)^2} + \left[-\left(\frac{\partial^2 u}{\partial t^2}\right)_i \frac{\Delta t}{2} + \left(\frac{\partial^4 u}{\partial x^4}\right)_i \frac{(\Delta x)^2}{12} + \dots \right] \quad (5.25)$$

على جانب اليسار للمعادلة (5.25) هناك المعادلة التفاضلية الجزئية الأصلية (original partial differential equation) ، وعلى الجانب الأيمن هناك المصطلحين الأول والثاني لمصطلح الفرق المحدودة (finite difference expression) لهذه المعادلة.

و المصطلحات الواردة في أقواس مربعة [] هي خطأ الاقتطاع (truncation error) للمعادلة الكاملة. خطأ الاقتطاع (truncation error) لهذا البيان (representation) هو $O[\Delta t, (\Delta x)^2]$.

هل معادلة الفرق المحدود (finite-difference equation) تساوي المعادلة التفاضلية الأصلية (original differential equation) إذا عدد نقاط الشبكة يذهب إلى ما لا نهاية، أي لو مثلاً $\Delta x \rightarrow 0$ و $\Delta t \rightarrow 0$ ؟

فحص المعادلة (5.25)، نلاحظ أن خطأ الاقتطاع (truncation error) يذهب إلى الصفر، وبالتالي معادلة الاختلاف (difference equation) تقترب حقا من المعادلة التفاضلية الأصلية.

عندما يكون هذا هو الحال، يقال إن بيان الفرق المحدودة (finite-difference representation) للمعادلة التفاضلية الجزئية (partial differential equation) متناسق (consistent).

حل المعادلة (5.24) يأخذ شكل حل 'السير' ('marching') في خطوات من الزمن. (ولنتذكر من المقطع 4.3.2 أن حلول السير (marching solutions) هي سمة من سمات لمعادلات القطع المكافئ (parabolic equations)).

فترض أننا نعرف المتغير التابع (dependent variable) لكل x في بعض لحظة من الزمن، لظروف الأولية (initial conditions) المعطية. و بفحص المعادلة (5.24)، نرى أنها تحتوي على متغير واحد فقط غير معروف (unknown)، وهو u_i^{n+1} . وبهذه الطريقة، يمكن الحصول على المتغير التابع (dependent variable) في الوقت $(t + \Delta t)$ مباشرة من النتائج المعروفة (known results) في الوقت t ، يعني ذلك انه يتم الحصول عليها مباشرة من القيم المعروفة (known values) u_{j-1}^n و u_{j+1}^n و u_j^n . هذا هو مثال على حل للفرق المحدودة بالشكل الواضح المباشر (explicit finite-difference solution).

بالمقابل كمثال مضاد، نعود إلى المعادلة التفاضلية الجزئية الأصلية (original partial differential equation) التي قدمتها المعادلة (5.23). هذه المرة، نكتب الاختلافات المكانية (spatial differences) على الجانب الأيمن و بمصطلحات المعدل (average properties) بين n و $(n+1)$ ، وهذا هو

وهذا الشكل من الاختلاف (differencing) المبين في المعادلة (5.26) يسمى الشكل الكرانك- نيكلسون (Crank-form) (Nicolson).

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2} \left[\frac{u_{i+1}^{n+1} + u_{i+1}^n - 2u_i^{n+1} - 2u_i^n + u_{i-1}^{n+1} + u_{i-1}^n}{(\Delta x)^2} \right] \quad (5.26)$$

افحص المعادلة (5.26). غير المعروف u_i^{n+1} لا يُعبَّر عنه فقط عبر كميات معروفة (terms of the known quantities) في نقطة الزمان n - و هي u_i^{n+1}, u_i^n ، و u_i^{n-1} - ولكن أيضا عبر كميات غير معروفة و هي في نقطة الزمان $n+1$ - و هي u_i^{n+1} و u_i^{n+1-i} .

وبالتالي، اذا حاولنا ان نطبق المعادلة (5.26) عند نقطة معينة i في الشبكة (grid) لا يمكن في هذه النقطة بحد ذاتها الحصول على الحل ل u_i^{n+1} . بدلا من ذلك، المعادلة (5.26) يجب أن تكون مكتوبة في جميع نقاط الشبكة، مما يؤدي الى نظام من المعادلات الجبرية (system of algebraic equations) حيث يكون المجهول (u_i^{n+1} unknown) لجميع i و يمكن حلها سوياً في وقت واحد. هذا مثال على حل ضمني للفرق المحدود (implicit finite-difference solution). لأنها تعالج مع حل لنظم كبيرة (large systems) من المعادلات الجبرية الخطية في وقت واحد (simultaneous linear algebraic) وتشارك عادة الطرق الضمنية (implicit methods) في التلاعب بالمصفوفات الكبيرة (large matrices). وفيما يلي موجز من الايجابيات (advantages) والسلبات (disadvantages) الرئيسية بالنسبة لهذين المنهجين.

(1) النهج الصريح (Explicit approach)

(أ) ايجابية (advantage): بسيط نسبياً لإنشاء (set up) برنامج (program).

(ب) السلبية (disadvantage): على صعيد المثال اعلاه، ل Δx معين، يجب أن يكون Δt أقل من بعض الحدود (limit) التي تفرضها قيود الاستقرار (stability constraints). في كثير من الحالات، يجب أن تكون Δt ضئيلة للغاية للحفاظ على الاستقرار (stability)، وهذا يمكن أن يؤدي إلى تشغيل الكمبيوتر (computer) لوقت طويل لإجراء حسابات (calculations) على مدى فترة معينة من الزمان t .

(2) النهج الضمني (Implicit approach)

(أ) ميزة. يمكن الحفاظ على الاستقرار (stability) للقيم الأكبر بكثير من Δt ، وبالتالي باستخدام خطوات وقت أقل بكثير لجعل العمليات الحسابية (calculations) على مدى فترة معينة من t . هذه النتائج تأخذ وقتاً أقل في الكمبيوتر (computer).
(ب) العيب. أكثر تعقيداً لإنشاء برنامج (program).

(ج) العيب: بما ان التلاعب بالمصفوفة الضخمة (massive matrix) هي بشكل عام ضرورية في كل خطوة من الوقت، وقت الكمبيوتر في كل خطوة وقت هو أكبر بكثير مما كانت عليه في النهج الصريح (explicit approach).

(د) العيب: بما انه يمكن اتخاذ Δt كبيرة ، وخطأ اقتطاع أكبر (truncation error)، واستخدام طرق ضمنية (implicit methods) لتتبع العابرين المحددين (اختلافات الوقت للمتغيرات المستقلة (independent variable)) قد لا تكون دقيقة كالنهج الصريح (explicit approach).

ومع ذلك للتوصل الى حل مشروط بالوقت حيث فيه حالة الاستقرار (steady state) هي النتيجة المرجوة بالنسبة لناحية الوقت غير الدقيق (inaccuracy) هي ليست مهمة.

خلال الفترة من عام 1969 إلى حوالي عام 1979، فإن الغالبية العظمى من الحلول CFD العملية التي تنطوي على حلول 'السير' ('marching') (كما هو الحال في المثال أعلاه) حيث الطرق الواضحة (explicit methods) هي المستخدمة.

ومع ذلك، فإن العديد من تطبيقات ال CFD الأكثر تطوراً تلك التي تتطلب نقاط شبكة (grid points) قريبة جداً من بعضها في بعض مناطق التدفق (regions of the flow) الذي يتطلب وقت تشغيل أكبر للكمبيوتر نظراً إلى خطوات السير الصغيرة (small marching steps) المطلوبة.

وقد جعلت هذه الميزة (advantage) المذكورة أعلاه الطرق الضمنية (implicit methods) جذابة للغاية ألا وهي القدرة على استخدام خطوات سير كبيرة حتى بالنسبة لشبكة دقيقة جداً. لهذا السبب كانت الطرق الضمنية (implicit methods) في الثمانينيات محوراً رئيسياً من تطبيقات ال CFD.

21.3.1 تعليق عام

فمن الواضح أن حلول الفرق المحدودة ، تبدو فلسفياً واضحة باستبدال المشتقات الجزئية (partial derivatives) في المعادلات الأساسية (governing equations) بمحاصل الفرق الجبرية (algebraic difference quotients)، و تقليص الفرق للحصول على حلول لهذه المعادلات الجبرية (algebraic equations) في كل نقطة من نقاط الشبكة. ومع ذلك، هذه الفكرة مضللة. لأي تطبيق معين، ليس هناك ما يضمن أن مثل هذه الحسابات (calculations) ستكون دقيقة (accurate)، أو حتى مستقرة (stable)، في ظل جميع الشروط. وعلاوة على ذلك، فإن شروط الحدود (boundary conditions) لمشكلة معينة إملأ حل، وبالتالي فإن العلاج المناسب لشروط الحدود (boundary conditions) في إطار تقنية محدودة، ولا سيما الفرق المحدود (finite-difference) أمر في غاية الأهمية

21.4 أخطاء وتحليل الاستقرار (Errors and an Analysis of Stability)

At the end of the last section, we stated that no guarantee exists for the accuracy and stability of a system of finite-difference, equations under all conditions.

في نهاية المقطع الأخير، ذكرنا أنه لا وجود لضمان دقة واستقرار نظام الفرق المحدود، للمعادلات في كل الشروط.

However for linear equations there is a formal way of examining the accuracy and stability and these ideas at least provide guidance for the understanding of the behaviour of the more complex non-linear system that is our governing flow equations.

ولكن للمعادلات الخطية هناك وسيلة رسمية لفحص الدقة والاستقرار وهذه الأفكار على الأقل تقدم توجيه لبيان فهم سلوك النظام غير الخطي والأكثر تعقيدا وهذه هي المعادلات الأساسية للسريان.

In this section we introduce some of these ideas, applied to simple linear equations.

في هذا القسم نقدم بعض هذه الأفكار التي تطبق على المعادلات الخطية البسيطة.

The material in this section is patterned somewhat after section 3–6 of the excellent book on CFD by Dale Anderson, John Tannehill and Richard Pletcher (Ref. [1]) which should be consulted for more details. Consider a partial differential equation, such as for example Eq. (5.23). The numerical solution of this equation is influenced by two sources of error:

ونمط هذه المواد في هذا الباب إلى حد ما بعد القسم 3-6 من كتاب CFD الجديد الممتاز من قبل دايل أندرسون (Dale Anderson)، جون تانهيل (John Tannehill) وريتشارد بلاتشار (Richard Pletcher) (المرجع [1]) التي ينبغي التشاور معها لأي تفصيل دقيق. نعتبر المعادلة التفاضلية الجزئية، مثل على سبيل المثال المعادلة (5.23). ويتأثر الحل العددي لهذه المعادلة من قبل اثنين من مصادر الخطأ:

1. Discretization error. The difference between the exact analytical solution of the partial differential equation (for example, Eq. (5.23)) and the exact (round-off free) solution of the corresponding difference equation (for example, Eq. (5.24)).

From our previous discussion, the discretization error is simply the truncation error for the difference equation plus any errors introduced by the numerical treatment of the boundary conditions.

1. خطأ التفريز (Discretization error). الفرق بين الحل التحليلي (analytical solution) الدقيق للمعادلة التفاضلية الجزئية (partial differential equation) (على سبيل المثال المعادلة (5.23)) والحل الدقيق (دون تقريب (round-off free)) الذي يتوافق مع معادلة الفرق (difference equation) (على سبيل المثال المعادلة (5.24)).

في مناقشتنا السابقة خطأ التفريز (discretization error) هو ببساطة خطأ اقتطاع (truncation error) معادلة الفرق (difference equation) بالإضافة الى الاخطاء التي تدخل في المعالجة الرقمية (numerical treatment) لشروط الحدود (boundary conditions).

2. Round-off error. The numerical error introduced after a repetitive number of calculations in which the computer is constantly rounding the numbers to some significant figure.

2. خطأ التقريب (Round-off error). يدخل الخطأ العددي (numerical error) بعد عدد من العمليات الحسابية (calculations) المتكررة في جهاز الكمبيوتر الذي يقوم بتقريب (rounding) الأرقام باستمرار إلى بعض الأعداد المعبرة (significant figure).

إذا تركنا

A = الحل التحليلي (analytical solution) للمعادلة التفاضلية الجزئية (partial differential equation)

D = الحل الدقيق (exact solution) لمعادلة الفرق (difference equation)

N = الحل العددي (numerical solution) من جهاز كمبيوتر حقيقي مع دقة متناهية (finite accuracy)

إذاً

خطأ التفريز (Discretization error) $A - D =$

التقريب (Round-off) $N - D = \varepsilon =$ (5.27)

من المعادلة (5.27)، يمكن أن نكتب

$$N = D + \varepsilon \quad (5.28)$$

حيث مرة أخرى ε هو خطأ التقريب (round-off error)، لبقية مناقشتنا في هذا القسم، و سوف نسميه ببساطة "خطأ"

للإيجاز. الحل العددي (numerical solution) N يجب ان تكفي معادلة الفرق (difference equation). وبالتالي من المعادلة

(5.24)،

$$\frac{D_i^{n+1} + \varepsilon_i^{n+1} - D_i^n - \varepsilon_i^n}{\Delta t} = \frac{D_{i+1}^n + \varepsilon_{i+1}^n - 2D_i^n - 2\varepsilon_i^n + D_{i-1}^n + \varepsilon_{i-1}^n}{(\Delta x)^2} \quad (5.29)$$

By definition, D is the exact solution of the difference equation, hence it exactly satisfies:

بحكم التعريف، D هو الحل الدقيق (exact solution) لمعادلة الفرق (difference equation)، وبالتالي انه يفني تماما:

$$\frac{D_i^{n+1} - D_i^n}{\Delta t} = \frac{D_{i+1}^n - 2D_i^n + D_{i-1}^n}{(\Delta x)^2} \quad (5.30)$$

Subtracting Eq. (5.30) from (5.29),

طرح المعادلة (5.30) من (5.29)،

$$\frac{\varepsilon_i^{n+1} - \varepsilon_i^n}{\Delta t} = \frac{\varepsilon_{i+1}^n - 2\varepsilon_i^n + \varepsilon_{i-1}^n}{(\Delta x)^2} \quad (5.31)$$

From Eq. (5.31), we see that the error ε also satisfies the difference equation.

من المعادلة (5.31)، نرى ان الخطأ (error) ε يكفي ايضاً معادلة الفرق (difference equation).

تحليل الاستقرار - Stability Analysis

We now consider aspects of the stability of the difference equation, Eq. (5.24). **If errors ε_i are already present at some stage of the solution of this equation (as they always are in any real computer solution), then the solution will be stable if the ε_i 's shrink, or at best stay the same, as the solution progresses from step n to $n+1$; on the other hand, if the ε_i 's grow larger during the progression of the solution from steps n to $n+1$, then the solution is unstable.**

That is, for a solution to be stable,

نحن نعتبر الآن جوانب الاستقرار (aspects of the stability) في معادلة الفرق (difference equation)، المعادلة (5.24). إذا كانت الأخطاء ε_i موجودة في بعض مراحل الحل لهذه المعادلة (كما هم دائماً في أي حل حقيقي للكمبيوتر)، ثم فإن الحل يكون مستقراً (**stable**) إذا كانت الأخطاء ε_i تتقلص، أو في أحسن الأحوال تبقى نفسها، حيث الحل يتقدم من الخطوة n إلى $n+1$ ، ومن ناحية أخرى، إذا كانت ε_i تكبر مع تقدم الحل من المرحلة n إلى $n+1$ فإن الحل يكون غير مستقر (**unstable**).

بطريقة أخرى للتوصل إلى حل يكون مستقر (stable)،

$$|\varepsilon_i^{n+1} / \varepsilon_i^n| \leq 1 \quad (5.32)$$

For Eq. (5.24), let us examine under what conditions Eq. (5.32) holds. Assume that the distribution of errors along the x-axis is given by a Fourier series in x , and that the time-wise variation is exponential in t , i.e.

للمعادلة (5.24)، دعونا نبحث تحت أي شروط تستمر المعادلة (5.32). لنفرض ان توزيع الأخطاء (distribution of errors) على طول محور x (x-axis) تكون معطاة من قبل سلسلة فورييه (Fourier series) في x ، وهذا من ناحية الوقت الاختلاف هو الأسّي (exponential) في t ، أي

$$\varepsilon(x, t) = e^{at} \sum_m e^{ik_m x} \quad (5.33)$$

Where k_m is the wave number and where the exponential factor a is a complex number. Since the difference equation is linear, when Eq. (5.33) is substituted into Eq. (5.31) the behaviour of each term of the series is the same as the series itself. Hence, let us deal with just one term of the series, and write

حيث k_m هو عدد الموجات (wave number) وحيث العامل الأسّي (exponential factor) a هو عدد مركب (complex number). بما ان معادلة الفرق (difference equation) هي خطية (linear)، عندما يتم استبدال المعادلة (5.33) في المعادلة (5.31) سلوك كل مصطلح (term) من هذه السلسلة (series) هو نفس السلسلة (series) ذاتها. ومن ثم، دعونا نتعامل مع مصطلح واحد فقط من هذه السلسلة (series)، وكتابة

$$\varepsilon_m(x, t) = e^{at} e^{ik_m x} \quad (5.34)$$

Substitute Eq. (5.34) into Eq. (5.31),

استبدال المعادلة (5.34) في المعادلة (5.31).

$$\frac{e^{a(t+\Delta t)} e^{ik_m x} - e^{at} e^{ik_m x}}{\Delta t} = \frac{e^{at} e^{ik_m(x+\Delta x)} - 2e^{at} e^{ik_m x} + e^{at} e^{ik_m(x-\Delta x)}}{(\Delta x)^2} \quad (5.35)$$

Divide Eq. (5.35) by $e^{at} e^{ik_m x}$.

تقسيم (Divide) المعادلة (5.35) من قبل $e^{at} e^{ik_m x}$.

$$\frac{e^{a\Delta t} - 1}{\Delta t} = \frac{e^{ik_m \Delta x} - 2 + e^{-ik_m \Delta x}}{(\Delta x)^2}$$

or,

$$e^{a\Delta t} = 1 + \frac{\Delta t}{(\Delta x)^2} (e^{ik_m \Delta x} + e^{-ik_m \Delta x} - 2) \quad (5.36)$$

تذكير معادلة (identity) تلك

$$\cos(k_m \Delta x) = \frac{e^{ik_m \Delta x} + e^{-ik_m \Delta x}}{2}$$

الشكل

ويمكن كتابة المعادلة (5.36) على

$$e^{a\Delta t} = 1 + \frac{2\Delta t}{(\Delta x)^2} [\cos(k_m \Delta x) - 1] \quad (5.37)$$

التالي:

تذكر معادلة تريجونوميترك اخرى (trigonometric identity)

$$\sin^2[(k_m \Delta x)/2] = \frac{1 - \cos(k_m \Delta x)}{2}$$

المعادلة (5.37) تصبح في نهاية المطاف

$$e^{a\Delta t} = 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \quad (5.38)$$

من المعادلة (5.34)

$$\frac{\varepsilon_i^{n+1}}{\varepsilon_i^n} = \frac{e^{a(t+\Delta t)} e^{ik_m x}}{e^{at} e^{ik_m x}} = e^{a\Delta t} \quad (5.39)$$

الجمع (Combining) بين المعادلات (5.39) و (5.38) يصبح لدينا

$$\left| \frac{\varepsilon_i^{n+1}}{\varepsilon_i^n} \right| = |e^{a\Delta t}| = \left| 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \right| \leq 1 \quad (5.40)$$

Equation (5.40) must be satisfied to have a stable solution, as dictated by Eq. (5.32). In Eq. (5.40) the factor

يجب أن توفى المعادلة (5.40) كل شروط ليكون لدينا حل مستقر وفقاً لما تمليه المعادلة (5.32). في المعادلة (5.40) العامل

$$\left| 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \right| \equiv G$$

is called the amplification factor, and is denoted by G. Evaluating the inequality in Eq. (5.40), namely $G \leq 1$, we have two possible situations which must hold simultaneously:

وهو يسمى عامل التضخيم (amplification factor) ويرمز اليه عبر الرمز G. تقييم التفاوت (inequality) في المعادلة (5.40)،

أي $G \leq 1$ ، لدينا اثنين من الحالات المحتملة التي يجب ان تحصل و تستمر في نفس الوقت:

$$(1) \quad 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \leq 1$$

Thus

$$\frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \geq 0$$

Since $\Delta t/(\Delta x)^2$ is always positive, this condition always holds.

بما ان $\Delta t/(\Delta x)^2$ هي دائماً ايجابي هذا الشرط يستمر دائماً

$$(2) \quad 1 - \frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \geq -1$$

Thus

$$\frac{4\Delta t}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] - 1 \leq 1$$

For the above condition to hold,

لاستمرار الشروط اعلاه

$$\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2} \quad (5.41)$$

Equation (5.41) gives the stability requirement for the solution of the difference equation, Eq. (5.24), to be stable.

المعادلة (5.41) تعطي متطلبات الاستقرار (stability requirement) لحل معادلة الفرق (difference equation), المعادلة (5.24)، ممكن ان تكون مستقرة (stable).

Clearly, for a given Δx , the allowed value of Δt must be small enough to satisfy Eq. (5.41).

بوضوح، لاجل Δx محددة، يجب أن تكون قيمة Δt صغيرة بما يكفي لتلبية حاجة المعادلة (5.41).

Here is a stunning example of the limitation placed on the marching variable by stability considerations for explicit finite difference models.

هنا هو مثال مذهل لوضع القيود على متغير السير (marching variable) التي تفرضها اعتبارات الاستقرار (stability) لنماذج الفرق المحدود الواضحة (explicit finite difference models).

As long as $\Delta t/(\Delta x)^2 \leq 1/2$, the error will not grow for subsequent marching steps in t , and the numerical solution will proceed in a stable manner.

طالما $\Delta t/(\Delta x)^2 \leq 1/2$ ، الخطأ لن ينمو لخطوات السير (marching steps) اللاحقة في t ، والحل العددي (numerical solution) سيحدث في حالة مستقرة (stable manner).

On the other hand, if $\Delta t/(\Delta x)^2 > 1/2$, then the error will progressively become larger, and will eventually cause the numerical marching solution to 'blow up' on the computer.

من ناحية أخرى، إذا $\Delta t/(\Delta x)^2 > 1/2$ ، إذاً الخطأ سوف يصبح تدريجياً أكبر، ويسبب في نهاية المطاف حل عددي للسير (numerical marching solution) لتفجير ('blow up') جهاز الكمبيوتر.

The above analysis is an example of a general method called the von Neuman stability method, which is used frequently to study the stability properties of linear difference equations.

إن التحليل (analysis) الوارد أعلاه هو مثال على طريقة عامة تسمى طريقة استقرار فون نيومان (von Neuman stability method)، التي كثيراً ما تستخدم لدراسة خصائص الاستقرار (stability properties) لمعادلات الفرق الخطية (linear difference equations).

مثال - Stability analysis of a hyperbolic equation

آخر: تحليل الاستقرار للمعادلة القطعية

Let us quickly examine the stability characteristics of another simple equation, this time a hyperbolic equation. Consider the first order wave equation:

دعونا بسرعة نقوم بدراسة خصائص الاستقرار (stability characteristics) لمعادلة بسيطة أخرى وهذه المرة لمعادلة قطعية (hyperbolic equation). لنعبر معادلة الدرجة الأولى للموجة (first order wave equation):

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0 \quad (5.42)$$

دعونا نستبدل المشتق المكاني (spatial derivative) مع الفرق المركزي (central difference) (انظر المعادلة (5.8)).

$$\frac{\partial u}{\partial x} = \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} \quad (5.43)$$

دعونا نستبدل مشتق الوقت (time derivative) مع الفرق ذات الدرجة الأولى (first order difference) حيث يتم تمثيل قيمة المعدل (average value) بين نقاط الشبكة (grid points) $(i+1)$ و $(i-1)$ ، أي

$$u(t) = \frac{1}{2}(u_{i+1}^n + u_{i-1}^n)$$

Then

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - \frac{1}{2}(u_{i+1}^n + u_{i-1}^n)}{\Delta t} \quad (5.44)$$

استبدال المعادلات (5.43) و (5.44) في (5.42)، يصبح لدينا

$$u_i^{n+1} = \frac{u_{i+1}^n + u_{i-1}^n}{2} - c \frac{\Delta t}{\Delta x} \left(\frac{u_{i+1}^n - u_{i-1}^n}{2} \right) \quad (5.45)$$

الجمع بين المعادلات (5.18) و (5.19)، نحصل على التفريق (differencing) المستخدم في المعادلة المذكورة أعلاه، حيث المعادلة

(5.44) مستعملة لتمثيل مشتق الوقت (time derivative)، التي تسمى طريقة لاكس (Lax)، وبعد بيتر لاكس (Peter Lax)

عالم الرياضيات الذي كان اول من طرحها. لو افترضنا الآن شكل الخطأ

$\varepsilon_m(x, t) = e^{ate^{ikmt}}$ (error) المعمول بما سابقاً، واستبدال هذا الشكل في المعادلة (5.45)، عامل التضخيم أصبح \sin

$$G = \cos(km\Delta x) - iC \sin(km\Delta x) \quad (5.46)$$

where $C = c.\Delta t/\Delta x$. The stability requirement is $|e^{at}| \leq 1$, which when applied to Eq. (5.46) yields

حيث $C = c.\Delta t/\Delta x$. الشرط المطلوب للاستقرار (stability requirement) هو $|e^{at}| \leq 1$ ، عندما تطبق على المعادلة (5.46) نحصل على:

$$C = c \frac{\Delta t}{\Delta x} \leq 1 \quad (5.47)$$

في المعادلة (5.47)، تسمى C عدد كوران (*Courant number*). هذه المعادلة تقول إن $(\Delta t \leq \Delta x/c)$ من أجل أن يكون الحل العددي (numerical solution) في المعادلة (5.45) مستقراً (stable). وعلاوة على ذلك، المعادلة (5.47) تسمى شرط كوران – فريدرخس – ليفي (*Courant–Friedrichs–Lewy condition*)، عموماً يكتب كشرط CFL. من المهم الإشارة إلى معيار الاستقرار (stability) العام للمعادلات القطعية (hyperbolic equations). دعونا ندرس الأهمية الفيزيائية (physical) لشرط ال CFL. لنعتبر معادلة الموجة (wave equation) ذات الدرجة الثانية

$$\frac{\partial^2 u}{\partial t^2} = c \frac{\partial^2 u}{\partial x^2} \quad (5.48)$$

الخطوط الرئيسية (characteristic lines) لهذه المعادلة (انظر القسم 4.2) تكون مقدمة عبر

$$x = ct \quad (\text{right running})$$

and

$$x = -ct \quad (\text{left running})$$

ورسمت في الشكل 5.3(a) و (b). في كلا الجزئين (a) و (b) من الشكل 5.3، سمح للنقطة b أن تكون تقاطع (intersection) لخصائص الاندفاع يميناً (right-running) خلال نقطة الشبكة $(i-1)$ ، و خصائص الاندفاع يساراً (left-running) خلال نقطة الشبكة $(i+1)$.

للمعادلة (5.48)، شرط ال CFL المعطى في المعادلة (5.47) يعطي معيار الاستقرار (stability criterion). لنفترض $\Delta t_{C=1}$ يدل على قيمة ال Δt المقدمة بواسطة المعادلة (5.47) حيث $C = 1$. ثم $\Delta t_{C=1} = \Delta x/c$ وبالتالي نقطة التقاطع (stability criterion) b على مسافة $\Delta t_{C=1}$ فوق المحور x (x-axis)، كما رسمت في الرسم 5.3(a) و (b).

لنفترض الآن أن $C < 1$ ، وهي الحالة (case) المرسوم في الرسم 5.3(a). ثم من المعادلة (5.47)، $\Delta t_{C<1} < \Delta t_{C=1}$ ، كما هو مبين في

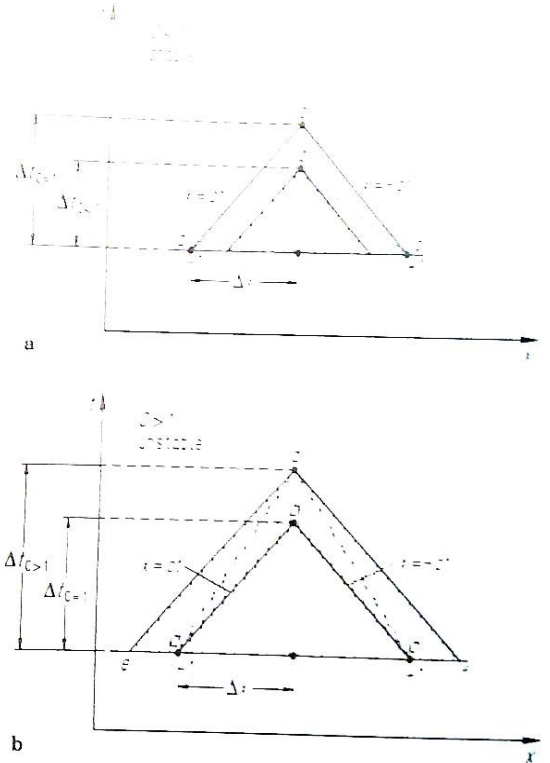
لنفترض النقطة d تتوافق مع نقطة في الشبكة عند النقطة i ، الموجودة في الوقت $(t+\Delta t_{C<1})$. بما ان الخصائص (properties) عند النقطة d تحسب عددياً (calculated numerically) من معادلة الفرق (equation) باستخدام نقاط الشبكة (grid) $(i-1)$ و $(i+1)$ ، النطاق العددي (numerical domain) للنقطة d يكون المثلث adc (triangle) الذي يظهر في الشكل 5.3(a).

المجال التحليلي (analytical domain) للنقطة d هو المثلث المظلل (shaded triangle) في الشكل 5.3(a)، المعرف عنه بالخصائص (characteristics) عند النقطة d . ونلاحظ أن في الشكل 5.3(a) المجال العددي (numerical domain) للنقطة d يشمل المجال التحليلي (analytical domain). في المقابل، لنفترض الحالة المبينة في الشكل 5.3(b). هنا، $C > 1$ ، إذاً، من المعادلة (5.47)، $\Delta t_{C>1} > \Delta t_{C=1}$ ، كما هو مبين في الشكل 5.3(b). لنفترض النقطة d

الشكل 5.3: توضيح للأهمية (physical)

(significance) الفيزيائية

لشروط CFL



في الشكل 5.3(b) التي تتناسب مع نقطة الشبكة i ، الموجودة في الوقت $(t+\Delta t_{C>1})$. بما ان الخصائص في النقطة d تحسب عددياً (calculated numerically) من معادلة الفرق (difference equation) باستخدام نقاط شبكة (grid points) $(i-1)$ و $(i+1)$ ، النطاق العددي (numerical domain) للنقطة d هو المثلث adc (triangle) الذي يظهر في الشكل 5.3(b). المجال التحليلي (analytical domain) للنقطة d هو المثلث المظلل (shaded triangle) في الشكل 5.3(b). والمعرف عنه من خلال الخصائص (characteristics) عند النقطة d . نلاحظ أن في الشكل 5.3(b) المجال العددي (numerical domain) لا يشمل كل المجال

التحليلي (analytical domain)، وهذا هو الشرط (condition) الذي يؤدي إلى سلوك غير مستقر (unstable behaviour). ولذلك، يمكن أن نقدم التفسير الفيزيائي (physical interpretation) التالي لشرط ال CFL (CFL condition) :

من أجل الاستقرار (stability)، المجال الحسابي (computational domain) يجب أن يشمل كل المجال التحليلي (analytical domain). الاعتبارات المذكورة أعلاه تدرس مع الاستقرار (stability). مسألة الدقة (accuracy)، والتي تختلف تماماً في بعض الأحيان، يمكن أيضاً أن تدرس من وجهة نظر الشكل 5.3. نعتبر الحالة المستقرة (stable case) كما هو مبين في الشكل 5.3(a). نلاحظ ان المجال التحليلي (analytic domain) للتبعية (dependence) للنقطة d هو المثلث المظلل (shaded triangle) في الشكل 5.3(a). من مناقشاتنا في الفصل 4 (Chap. 4)، والخصائص في نقطة d نظرياً يعتمد فقط على النقاط داخل المثلث المظلل (shaded triangle). ومع ذلك، نلاحظ ان نقاط الشبكة العددية (numerical grid points) $(i-1)$ و $(i+1)$ تكون خارج مجال التبعية (of dependence) وبالتالي نظرياً يجب ان لا يؤثر على الخصائص (properties) عند النقطة d. من ناحية أخرى، الحساب العددي (numerical calculation) للخصائص (properties) في نقطة d تأخذ معلومات من نقاط الشبكة (grid points) $(i-1)$ و $(i+1)$. وهذه الحالة تكون قد تفاقمت عندما يتم اختيار $\Delta t_{C1} < \Delta t_{C1}$ صغيرة جداً، $\Delta t_{C1} \ll \Delta t_{C1}$. في هذه الحالة، على الرغم من ان العمليات الحسابية (calculations) في حالة مستقرة (stable)، قد تكون النتائج (results) غير دقيقة (inaccurate) تماماً بسبب البعد (mismatch) الواسع بين المجال التبعية للنقطة (domain of dependence) d، و بين موقع البيانات العددية الفعلية (actual numerical data) المستخدمة لحساب الخصائص (properties) عند d. في ضوء المناقشة الواردة أعلاه، نخلص إلى أن العدد الحالي (Courant number) يجب أن يكون مساوي أو أقل من وحدة (unity) من أجل الاستقرار (stability)، $C \leq 1$ ، المرغوب فيه بنفس الوقت هو أن يكون C أقرب إلى وحدة (unity) كاحتمال من أجل الدقة (accuracy).

References

Anderson, D.A., Tannehill, John C. and Pletcher, Richard H., Computational Fluid Mechanics and Heat Transfer, McGraw-Hill, New York, 1984.

http://en.wikipedia.org/wiki/Computational_fluid_dynamics

22) Grid transformations تحولات الشبكة (

22.1 مدخل

إذا كانت كل تطبيقات CFD تتعامل مع المشاكل الفيزيائية المنتظمة، باستخدام الشبكة المستطيلة (uniform, rectangular grid)، لن يكون هناك أي سبب لتغيير معادلات التحكم المستمدة من الجزء 2 و يمكننا ببساطة تطبيق هذه المعادلات في فضاء (x,y,z,t) ، والفروق المحدودة. هذه المعادلات وفقا لحواصل الفرق المستمدة من الجزء 5، والحساب بعيدا، باستخدام قيم موحدة لل Δx ، Δy ، Δz و Δt ، ومع ذلك، بعض المشاكل الحقيقية يمكن استيعابها أكثر من أي وقت مضى، لنفترض أننا نريد حساب تدفق الهواء من الجنيح، كما هو واضح في Fig 6.1، حيث وضعنا الجنيح في شبكة مستطيلة.

بعض المشاكل مع هذا النوع من الشبكات:

(1) تسقط بعض نقاط الشبكة داخل الجنيح، أي أنهم تماما خارج التدفق. إذا ما هي قيمة خصائص التدفق التي يمكن ان ننسبها إلى هذه النقاط؟

(2) هناك عدد قليل، و إن وجد من نقاط الشبكة التي تقع على سطح الجنيح. هذا ليس جيدا. وذلك لأن سطح الجنيح هو شرط حيوي لحدود تحديد التدفق، وبالتالي سطح الجنيح يجب أن يظهر بوضوح وبقوة بالحل العددي.

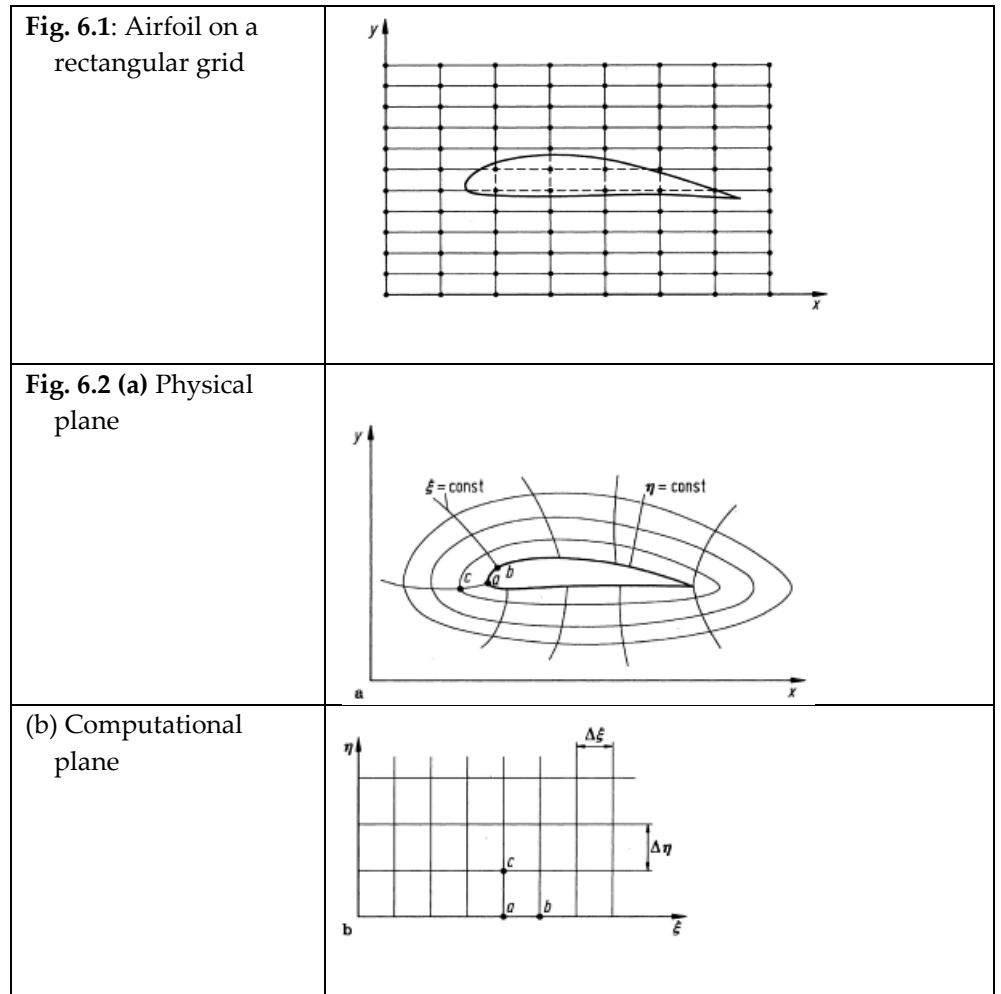
كنتيجة. يمكننا أن نستنتج أن الشبكة المستطيلة في Fig. 6.1 غير مناسبة لإيجاد حل لمجال التدفق. النقيض من ذلك، الشبكة التي تظهر خصائصها في Fig. 6.2(a). هنا نرى شبكة غير منتظمة و منحنية التي تقوم بالالتفاف كليا حول الجنيح. تنسيق جديد للخطوط ξ و η = ثابت. وهذا ما يسمى نظام أبعاد الحدود-المركبة، وسيتم مناقشتها بالتفصيل لاحقا في هذا الفصل. والنقطة المهمة هي أن نقاط الشبكة تسقط بشكل طبيعي على سطح

الجنيح، كما هو مبين في Fig. 6.2(a). و ما هو بنفس القدر من الأهمية هو أنه، في الحيز الفيزيائي المبين في Fig. 6.2(a)، وحواصل الفرق التقليدية التي يصعب استخدامها. ما يجب القيام به هو تحويل الشبكة المنحنية في الفضاء الفيزيائي إلى شبكة مستطيلة من حيث ξ و η يظهران. في Fig. 6.2(b) والتي توضح شبكة رباعية الأبعاد من

حيث ξ و η . الشبكة المستطيلة كما هو مبين في Fig. 6.2(b) وتسمى أيضا التخطيط الحاسوبي. هنا يأتي التطابق في النقط بين هذه الشبكة وشبكة الخطوط المنحنية في Fig. 6.2(a) وتسمى أيضا التخطيط الفيزيائي. على سبيل

المثال، النقاط a و b و c في التخطيط الفيزيائي (Fig. 6.2a) تتوافق مع نقاط a و b و c في التخطيط الحاسوبي، والذي يتضمن $\Delta\xi$ متجانسة و $\Delta\eta$ متجانسة. ثم يتم نقل المعلومات المحسوبة إلى التخطيط الفيزيائي. بالإضافة إلى ذلك، عندما يتم حل المعادلات التي تحكم البعد الحاسوبي، لا بد من التعبير ب ξ و η باعتبارها المتغيرات بدلا من x و y ، أي يجب أن تتحول المرتبطة ب (x, y) إلى (ξ, η) والمتغيرات المستقلة الجديدة. والغرض من هذا الفصل هو وصف لأول مرة التحول العام للمعادلات التي تتحكم بالتدفق بين التخطيط الفيزيائي و التخطيط الحاسوبي.

بعد ذلك، سيتم مناقشة عدة شبكات محددة. هذه المواد هي مثال على أبحاث متطورة في مجال CFD تسمى انشاء شبكة (grid generation).



General Transformation of the Equations 22.2

للبساطة، سوف ننظر تدفق متقلب ثنائي الأبعاد (two-dimensional unsteady flow)، مع المتغيرات المستقلة x, y, t ؛ نتائج لتدفق متقلب ثلاثي الأبعاد، مع المتغيرات المستقلة (x, y, z, t) ، هي مشابهة، و ببساطة تنطوي على مزيد من المصطلحات (terms).

سنقوم بتحويل المتغيرات في الحيز الفيزيائي (x, y, z) إلى الحيز (ξ, η, τ) ، حيث

$$\xi = \xi(x, y, t) \quad (6.1a)$$

$$\eta = \eta(x, y, t) \quad (6.1b)$$

$$\tau = \tau(t) \quad (6.1c)$$

في التحول المذكور أعلاه، تتغير τ حسب t فقط، وكثيرا ما تعطى على شكل $t = \tau$. هذا يبدو تافها (trivial) إلى حد ما؛ ومع ذلك، Eq. (6.1c) يجب أن تتم من خلال التحول (transformation) بطريقة رسمية (formal)، وإلا ستختفي بعض الجمل (terms) الضرورية. من قاعدة السلسلة من حساب التفاضل (chain rule of differential calculus)، لدينا

$$\left(\frac{\partial}{\partial x}\right)_{y,t} = \left(\frac{\partial}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial x}\right)_{y,t} + \left(\frac{\partial}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial x}\right)_{y,t} + \left(\frac{\partial}{\partial \tau}\right)_{\xi,\eta} \left(\frac{\partial \tau}{\partial x}\right)_{y,t} \quad 0$$

الاضافيات السفلية (subscripts) في التعبير أعلاه للتأكيد على ما يجري عقد المتغيرات المستمر (constant) في التفريق جزئي (partial differentiation). في التعبيرات اللاحقة، سيتم إسقاط الاضافيات السفلية (subscripts). ومع ذلك، فمن المفيد دائما ابقائهم في عقلك. وهكذا، وسوف نكتب التعبير أعلاه كما

$$\frac{\partial}{\partial x} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial x}\right) \quad (6.2)$$

Similarly,

$$\frac{\partial}{\partial y} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial y}\right) \quad (6.3)$$

Also,

$$\begin{aligned} \left(\frac{\partial}{\partial t}\right)_{x,y} &= \left(\frac{\partial}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial t}\right)_{x,y} + \left(\frac{\partial}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial t}\right)_{x,y} \\ &\quad + \left(\frac{\partial}{\partial \tau}\right)_{\xi,\eta} \left(\frac{\partial \tau}{\partial t}\right)_{x,y} \end{aligned} \quad (6.4)$$

or,

$$\frac{\partial}{\partial t} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial t}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial t}\right) + \left(\frac{\partial}{\partial \tau}\right)\frac{d\tau}{dt} \quad (6.5)$$

معادلات (6.2)، (6.3) و (6.5) تسمح للمشتقات بالنسبة لـ x, y و t إلى أن تتحول إلى مشتقات بالنسبة لـ ξ, η و τ . معاملات المشتقات (coefficients of the derivatives) بالنسبة لـ ξ, η و τ وتسمى المقاييس (metrics)، على سبيل المثال $\partial \xi / \partial x$ ، $\partial \eta / \partial x$ و $\partial \eta / \partial y$ هي جمل مقياسية (metric terms) والتي يمكن الحصول عليها من التحول العام للمعادلات Eqs. (6.1a, b and c). إذا اعطت المعادلات Eqs. (6.1a, b and c) كاشكال تحليلي مغلق (closed form analytic expressions)، ثم يمكن أيضا الحصول على المقاييس (metrics) في شكل مغلق. ومع ذلك، فإن التحول (transformation) على أساس المعادلات Eqs. (6.1a, b, and c) هي في كثير من الأحيان وجود علاقة عددية (numerical) بجته، وفي هذه الحالة المقاييس يمكن تقييمها من قبل حواصل الفروق المحدودة (finite-difference quotients) - عادة الاختلافات المركزية. إذا درسنا معادلات التحكم المستمدة في الفقرة الثانية، نلاحظ أن معادلات السريان اللزجي (viscous flow) تشمل المشتقات ثانية (second derivatives). ولذلك، فإننا بحاجة إلى التحول لهذه المشتقات. يمكن الحصول عليها على النحو التالي. تبداً من المعادلة (6.2) ونجعلها

$$A = \frac{\partial}{\partial x} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial x}\right)$$

Then,

$$\begin{aligned} \frac{\partial^2}{\partial x^2} &= \frac{\partial A}{\partial x} = \frac{\partial}{\partial x} \left[\left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial x}\right) \right] \\ &= \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial^2 \xi}{\partial x^2}\right) + \underbrace{\left(\frac{\partial \xi}{\partial x}\right)\left(\frac{\partial^2}{\partial x \partial \xi}\right)}_B + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial^2 \eta}{\partial x^2}\right) + \underbrace{\left(\frac{\partial \eta}{\partial x}\right)\left(\frac{\partial^2}{\partial \eta \partial x}\right)}_C \end{aligned} \quad (6.6)$$

المشتقات المختلطة المرموز لها بواسطة B و C في المعادلة (6.6c) Eq. ويمكن الحصول عليها من قاعدة السلسلة من حساب التفاضل (chain rule of differential calculus) على النحو التالي:

$$B = \frac{\partial^2}{\partial x \partial \xi} = \frac{\partial}{\partial x} \left(\frac{\partial}{\partial \xi} \right)$$

معتمدين على قاعدة السلسلة، المعادلة (6.2) Eq. لدينا:

$$B = \left(\frac{\partial^2}{\partial \xi^2} \right) \left(\frac{\partial \xi}{\partial x} \right) + \left(\frac{\partial^2}{\partial \eta \partial \xi} \right) \left(\frac{\partial \eta}{\partial x} \right) \quad (6.7)$$

Similarly:

$$C = \frac{\partial^2}{\partial x \partial \eta} = \frac{\partial}{\partial x} \left(\frac{\partial}{\partial \eta} \right) = \left(\frac{\partial^2}{\partial \xi \partial \eta} \right) \left(\frac{\partial \xi}{\partial x} \right) + \left(\frac{\partial^2}{\partial \eta^2} \right) \left(\frac{\partial \eta}{\partial x} \right) \quad (6.8)$$

وباستبدال B و C من المعادلات (6.7) Eqs. و (6.8) ووضعها في المعادلة (6.6) Eq. وإعادة ترتيب تسلسل الشروط، يصبح لدينا:

$$\frac{\partial^2}{\partial x^2} = \left(\frac{\partial}{\partial \xi} \right) \left(\frac{\partial^2 \xi}{\partial x^2} \right) + \left(\frac{\partial}{\partial \eta} \right) \left(\frac{\partial^2 \eta}{\partial x^2} \right) + \left(\frac{\partial^2}{\partial \xi^2} \right) \left(\frac{\partial \xi}{\partial x} \right)^2 + \left(\frac{\partial^2}{\partial \eta} \right) \left(\frac{\partial \eta}{\partial x} \right)^2 + 2 \left(\frac{\partial^2}{\partial \eta \partial \xi} \right) \left(\frac{\partial \eta}{\partial x} \right) \left(\frac{\partial \xi}{\partial x} \right) \quad (6.9)$$

المعادلة (6.9) تعطي المشتقات الجزئية الثانية فيما يتعلق ب x من حيث المشتقات الأولى والثانية، والمختلط فيما يتعلق ب ξ و η ، مضروباً بمقاييس مختلفة. دعونا الآن نستمر في الحصول على الجزئية الثانية فيما يتعلق ب y. من المعادلة (6.3) Eq. دع

$$D \equiv \frac{\partial}{\partial y} = \left(\frac{\partial}{\partial \xi} \right) \left(\frac{\partial \xi}{\partial y} \right) + \left(\frac{\partial}{\partial \eta} \right) \left(\frac{\partial \eta}{\partial y} \right)$$

Then,

$$\begin{aligned} \frac{\partial^2}{\partial y^2} &= \frac{\partial D}{\partial y} = \frac{\partial}{\partial y} \left[\left(\frac{\partial}{\partial \xi} \right) \left(\frac{\partial \xi}{\partial y} \right) + \left(\frac{\partial}{\partial \eta} \right) \left(\frac{\partial \eta}{\partial y} \right) \right] \\ &= \left(\frac{\partial}{\partial \xi} \right) \left(\frac{\partial^2 \xi}{\partial y^2} \right) + \left(\frac{\partial \xi}{\partial y} \right) \left(\frac{\partial^2}{\partial \xi \partial y} \right) + \left(\frac{\partial}{\partial \eta} \right) \left(\frac{\partial^2 \eta}{\partial y^2} \right) + \left(\frac{\partial \eta}{\partial y} \right) \left(\frac{\partial^2}{\partial \eta \partial y} \right) \end{aligned} \quad (6.10)$$

Using Eq. (6.3),

$$E = \frac{\partial}{\partial y} \left(\frac{\partial}{\partial \xi} \right) = \left(\frac{\partial^2}{\partial \xi^2} \right) \left(\frac{\partial \xi}{\partial y} \right) + \left(\frac{\partial^2}{\partial \eta \partial \xi} \right) \left(\frac{\partial \eta}{\partial y} \right) \quad (6.11)$$

and

$$F = \frac{\partial}{\partial y} \left(\frac{\partial}{\partial \eta} \right) = \left(\frac{\partial^2}{\partial \eta \partial \xi} \right) \left(\frac{\partial \xi}{\partial y} \right) + \left(\frac{\partial^2}{\partial \eta^2} \right) \left(\frac{\partial \eta}{\partial y} \right) \quad (6.12)$$

باستبدال Eqs. (6.11) و (6.12) في (6.10). نحصل على التالي

$$\frac{\partial^2}{\partial y^2} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial^2 \xi}{\partial y^2}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial^2 \eta}{\partial y^2}\right) + \left(\frac{\partial^2}{\partial \xi^2}\right)\left(\frac{\partial \xi}{\partial y}\right)^2 + \left(\frac{\partial^2}{\partial \eta^2}\right)\left(\frac{\partial \eta}{\partial y}\right)^2 + 2\left(\frac{\partial^2}{\partial \eta \partial \xi}\right)\left(\frac{\partial \eta}{\partial y}\right)\left(\frac{\partial \xi}{\partial y}\right) \quad (6.13)$$

المعادلة (6.13) تعطي المشتقات الجزئية الثانية فيما يتعلق ب y من حيث الأولى، والمشتقات الثانية، والمختلطة فيما يتعلق ب ξ و η ، مضروبة بمقاييس مختلفة. نواصل الآن العمل للحصول على الجزئية الثانية فيما يتعلق ب x و y .

$$\frac{\partial^2}{\partial x \partial y} = \frac{\partial}{\partial x} \left(\frac{\partial}{\partial y}\right) = \frac{\partial D}{\partial x} = \frac{\partial}{\partial x} \left[\left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial y}\right) \right] = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial^2 \xi}{\partial x \partial y}\right) + \left(\frac{\partial \xi}{\partial y}\right)\left(\frac{\partial^2}{\partial \xi \partial x}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial^2 \eta}{\partial x \partial y}\right) + \left(\frac{\partial \eta}{\partial y}\right)\left(\frac{\partial^2}{\partial \eta \partial x}\right) \quad (6.14)$$

نستبدل (6.7) و (6.8) لـ B و C على التوالي في المعادلة (6.14)، ونعيد ترتيب المعادلة.

$$\frac{\partial^2}{\partial x \partial y} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial^2 \xi}{\partial x \partial y}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial^2 \eta}{\partial x \partial y}\right) + \left(\frac{\partial^2}{\partial \xi^2}\right)\left(\frac{\partial \xi}{\partial x}\right)\left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial^2}{\partial \eta^2}\right)\left(\frac{\partial \eta}{\partial x}\right)\left(\frac{\partial \eta}{\partial y}\right) + \left(\frac{\partial^2}{\partial \eta \partial \xi}\right)\left[\left(\frac{\partial \eta}{\partial x}\right)\left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial \xi}{\partial x}\right)\left(\frac{\partial \eta}{\partial y}\right)\right] \quad (6.15)$$

المعادلة (6.15) تعطي المشتقات الجزئية الثانية بالنسبة ل x و y من حيث الأولى، والمشتقات الثانية، والمختلطة فيما يتعلق ب ξ و η ، مضروبة بمقاييس مختلفة.

جميع المعادلات الواردة أعلاه تمثل كل ما هو ضروري لتحويل المعادلات التي تحكم التدفق تم الحصول عليها في الفقرة الثانية (Ch. 2) مع (x, y, t) كمغيرات مستقلة ل ξ ، η ، و T كمغيرات مستقلة جديدة. بوضوح، عندما يتم هذا التحويل، والمعادلات التي تتغير حسب ξ ، η ، و T تصبح طويلة نوعا ما. دعونا ننظر في مثال بسيط، وهو التدفق غير اللزج، غير الدوراني، الثابت، و غير القابل للانضغاط، حيث معادلة لابلاس هي المعادلة التي تحكم.

$$\text{Laplace's Equation : } \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (6.16)$$

تحويل المعادلة (6.16) من (x, y) إلى (ξ, η) ، حيث $\xi(x, y) = \xi$ و $\eta(x, y) = \eta$ ، بالإعتماد على Eqs. (6.9) و (6.13):

$$\begin{aligned} & \left(\frac{\partial^2 \phi}{\partial \xi^2}\right) \left(\frac{\partial \xi}{\partial x}\right)^2 + 2 \left(\frac{\partial^2 \phi}{\partial \xi \partial \eta}\right) \left(\frac{\partial \eta}{\partial x}\right) \left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial^2 \phi}{\partial \eta^2}\right) \left(\frac{\partial \eta}{\partial x}\right)^2 \\ & + \left(\frac{\partial \phi}{\partial \xi}\right) \left(\frac{\partial^2 \xi}{\partial x^2}\right) + \left(\frac{\partial \phi}{\partial \eta}\right) \left(\frac{\partial^2 \eta}{\partial x^2}\right) + \left(\frac{\partial^2 \phi}{\partial \xi^2}\right) \left(\frac{\partial \xi}{\partial y}\right)^2 \\ & + 2 \left(\frac{\partial^2 \phi}{\partial \eta \partial \xi}\right) \left(\frac{\partial \eta}{\partial y}\right) \left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial^2 \phi}{\partial \eta^2}\right) \left(\frac{\partial \eta}{\partial y}\right)^2 \\ & + \left(\frac{\partial \phi}{\partial \xi}\right) \left(\frac{\partial^2 \xi}{\partial y^2}\right) + \left(\frac{\partial \phi}{\partial \eta}\right) \left(\frac{\partial^2 \eta}{\partial y^2}\right) = 0 \end{aligned}$$

Rearranging terms, we obtain

$$\begin{aligned} & \frac{\partial^2 \phi}{\partial \xi^2} \left[\left(\frac{\partial \xi}{\partial x}\right)^2 + \left(\frac{\partial \xi}{\partial y}\right)^2 \right] + \frac{\partial^2 \phi}{\partial \eta^2} \left[\left(\frac{\partial \eta}{\partial x}\right)^2 + \left(\frac{\partial \eta}{\partial y}\right)^2 \right] \\ & + 2 \frac{\partial^2 \phi}{\partial \xi \partial \eta} \left[\left(\frac{\partial \eta}{\partial x}\right) \left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial \eta}{\partial y}\right) \left(\frac{\partial \xi}{\partial y}\right) \right] \\ & + \frac{\partial \phi}{\partial \xi} \left[\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} \right] + \frac{\partial \phi}{\partial \eta} \left[\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} \right] = 0 \end{aligned} \quad (6.17)$$

ادرس المعادلات (6.16) و (6.17) ؛ معادلة لابلاس (Laplace) السابقة في الفضاء الفيزيائي (x, y) ، و الأخيرة هي معادلة لابلاس في الفضاء الحاسوبي (ξ, η) . تحتوي المعادلة بوضوح العديد من الشروط. ومرة أخرى نؤكد أن (6.1), (6.13), (6.9), (6.5), (6.3), (6.2) و (6.15) تستخدم لتحويل المعادلات التي تحكم التدفق من التخطيط الفيزيائي (x, y) إلى التخطيط الحاسوبي (ξ, η) ، وأن الهدف من التحول في معظم التطبيقات CFD هو تحويل شبكة غير موحدة في الحيز الفيزيائي (مثل كما هو مبين في Fig. 6.2a) إلى شبكة موحدة في الحيز الحاسوبي (مثل ما هو مبين في Fig. 6.2b). معادلات التحكم التفاضلية الجزئية المتحولة تكون محدودة-الفرق في التخطيط الحاسوبي، حيث توجد $\Delta \xi$ متجانسة و $\Delta \eta$ متجانسة، كما هو مبين في Fig. 6.2b. يتم احتساب متغيرات ميدان التدفق في جميع نقاط الشبكة في التخطيط الحاسوبي، مثل نقاط a, b و c في Fig. 6.2(b). هذه هي نفس متغيرات مجال التدفق التي توجد في التخطيط الفيزيائي في نقاط المقابلة a, b و c في Fig. 6.2(a). وبالنظر إلى التحول الذي يحقق كل هذا في الشكل العام من قبل Eqs. 6.1a, b, and c. وبطبيعة الحال، لتنفيذ حل لمشكلة معينة، التحويلات تُعطى بشكل عام من قبل المعادلات (6.1a, b, and c) Eqs. لذلك يجب تحديدها صراحة. سيتم إعطاء أمثلة لبعض التحويلات محددة في الأقسام اللاحقة ان شاء الله.

في Eqs. (6.2), (6.3), (6.4), (6.5), (6.6), (6.7), (6.8), (6.9), (6.10), (6.11), (6.12), (6.13), (6.14), (6.15) والشروط التي تحوي هندسة الشبكات، مثل $\partial\xi/\partial x$ ، $\partial\eta/\partial y$ ، $\partial\xi/\partial x$ ، $\partial\eta/\partial y$ ، وما إلى ذلك، تُسمى: المقاييس. إذا كان التحول، مُعطى (6.1a, b and c)، من الناحية التحليلية، يمكن الحصول على قيم تحليلية لهذه المقاييس. ومع ذلك، في العديد من التطبيقات CFD، (6.1a, b and c)، تُعطى التحولات بشكل عددي، وبالتالي تُحسب المقاييس كما الفروق المحدودة.

أيضا، في العديد من التطبيقات، يُعبر عن التحولات بسهولة أكثر كمعكوس Eqs. (6.1a, b)، وهذا قد يتيح لدينا التحول العكسي.

$$x = x(\xi, \eta, \tau) \quad (6.18a)$$

$$y = y(\xi, \eta, \tau) \quad (6.18b)$$

$$t = t(\tau) \quad (6.18c)$$

في (6.18a, b and c)، ξ ، η و τ هي المتغيرات المستقلة. ومع ذلك، في التحولات المشتقة التي قدمتها المعادلات Eqs. (6.2), (6.3), (6.4), (6.5), (6.6), (6.7), (6.8), (6.9), (6.10), (6.11), (6.12), (6.13), (6.14), (6.15) وشروط المقاييس $\partial\xi/\partial x$ ، $\partial\eta/\partial y$ ، ما هي إلا مشتقات جزئية من حيث x ، y و t باعتبارها المتغيرات المستقلة. ولذلك، من أجل حساب شروط القياس في هذه المعادلات من التحول العكسي في (6.18a, b and c)، نحن في حاجة لربط $\partial\xi/\partial x$ ، $\partial\eta/\partial y$ ، وما إلى ذلك لعكس أشكال $\partial\xi/\partial x$ ، $\partial\eta/\partial y$ ، الخ. هذه الأشكال معكوس المقاييس هي القيم التي يمكن الحصول عليها مباشرة من التحول العكسي عبر (6.18a, b and c). دعونا نمضي قدما لإيجاد مثل هذه العلاقات.

النظر في المتغير التابع (المتصل) في المعادلات التي تحكم التدفق، مثل عنصر x من سرعة، u . حيث $u(x, y) = u$ من المعادلة 6.18 a b و $x(\eta, \xi) = x$ و $y(\eta, \xi) = y$. التفاضل الكامل ل u هو:

$$\frac{\partial u}{\partial \xi} = \frac{\partial u}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial u}{\partial y} \frac{\partial y}{\partial \xi} \quad (6.20)$$

المعادلاتان (6.20) و (6.21) يمكن أن ننظر إليهما باعتبارهما معادلتين لمجهولين اثنين $\partial u/\partial y$ و $\partial u/\partial x$. حل نظام المعادلات (6.20) و (6.21) لل $\partial u/\partial x$ باستخدام قاعدة كرامر Cramer، لدينا

$$(6.22) \quad \frac{\partial u}{\partial x} = \frac{\begin{vmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial u}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix}}{\begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix}}$$

في المعادلة (6.22)، يتم التعرف على المحددات كمصفوفه جاكوبي *Jacobian determinant* ، و التي نرمز لها بالتالي:

$$J \equiv \frac{\partial(x,y)}{\partial(\xi,\eta)} \equiv \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix}$$

وبالتالي، المعادلة (6.22) يمكن أن تكتب

$$\boxed{\frac{\partial u}{\partial x} = \frac{1}{J} \left[\left(\frac{\partial u}{\partial \xi} \right) \left(\frac{\partial y}{\partial \eta} \right) - \left(\frac{\partial u}{\partial \eta} \right) \left(\frac{\partial y}{\partial \xi} \right) \right]} \quad (6.23)$$

الآن دعونا نعود إلى Eqs. (6.20) and (6.21) ، وحل ل $\partial u/\partial y$.

$$\frac{\partial u}{\partial y} = \frac{\begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial u}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial u}{\partial \eta} \end{vmatrix}}{\begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix}}$$

or,

$$\boxed{\frac{\partial u}{\partial y} = \frac{1}{J} \left[\left(\frac{\partial u}{\partial \eta} \right) \left(\frac{\partial x}{\partial \xi} \right) - \left(\frac{\partial u}{\partial \xi} \right) \left(\frac{\partial x}{\partial \eta} \right) \right]} \quad (6.24)$$

انظر الى Eqs. (6.23) and (6.24) التي تعبر عن المشتقات من متغيرات مجال التدفق في البعد الفيزيائي من حيث

المشتقات إلى متغيرات مجال التدفق في البعد الحاسوبي.

المعادلات (6.23) و (6.24) تنجز نفس التحولات المشتقة كما قدمتها Eqs. (6.2) و (6.3). لكن، Eqs. (6.2) و (6.3) حيث الشروط المترية هي $\partial \xi / \partial x$ ، $\partial \eta / \partial y$ ، الخ، و المعادلات الجديدة (6.23) و (6.24) تحوي المقاييس المعكوسة، $\partial x / \partial \xi$ ، $\partial y / \partial \eta$ ، الخ ونلاحظ أيضا أن Eqs. (6.23) و (6.24) تشمل مصفوفه جاكوبي Jacobian من التحول. لذلك، كلما كان لدينا تحول يعطى في شكل Eqs. (6.18a, b and c)، و التي يمكنك من خلالها الحصول بسهولة على المقاييس في شكل $\partial \xi / \partial x$ ، $\partial x / \partial \eta$ ، الخ، والتدفق الذي يحكم التحول يمكن التعبير عن معادلاته من حيث هذه المقاييس العكسية ومصفوفه جاكوبي Jacobian، J ولكن مجموعة مماثلة و طويلة أكثر من النتائج يمكن الحصول عليها في تحول ثلاثي الأبعاد من (x, y, z) إلى (ξ ، η ، ζ). استشارة المرجع. [1] لمزيد من التفاصيل. مناقشة اعلاه قد اقتضت عمدا إلى بعدين من أجل إظهار المبادئ الأساسية دون التبعر النظر مع التفاصيل.

Coordinate Stretching 22.4

في الأقسام الثلاثة المتبقية من هذا الفصل، سوف ندرس ثلاثة أنواع من تحولات الشبكة. الأكثر بساطة مطروحة هنا. وهنا نطرح تمتد الشبكة في واحدة أو أكثر بالنسبة لاجتاهات. على سبيل المثال، وبالإعتماد على التخطيط الفيزيائي والحسابي المبين في Fig. 6.3(a, b). لنفترض أننا نتعامل مع تدفق لزج على سطح مستو، حيث السرعة تتغير بشكل ملحوظ بالقرب من السطح كما هو موضح في ملف تعريف سرعة رسمت في التخطيط الفيزيائي (Fig. 6.3a). لحساب تفاصيل هذا التدفق قرب السطح، نعتمد على شبكة متباعدة في الاتجاه y ينبغي أن تستخدم الشبكة، كما رسمت في التخطيط الفيزيائي. ومع ذلك، بعيدا عن السطح، يمكن للشبكة أن تكون أكثر قُربا. لذلك، يجب أن تكون الشبكة المناسبة واحدة في أي تنسيق خطوط و تصبح تدريجيا متباعدة كلما اقتربنا من السطح. من ناحية أخرى، نحن نرغب في التعامل مع شبكة موحدة في التخطيط الحسابي، كما هو مبين في الشكل Fig. 6.3(b). في الحقيقة نرى أن الشبكة في الحيز الفيزيائي قد "امتدت"، كما لو أنها شبكة موحدة وضعت على قطعة من المطاط، وقد امتدت صعودا في الاتجاه y . تحول تحليلي بسيط قادر على أن ينفذ هذا التمدد في الشبكة.

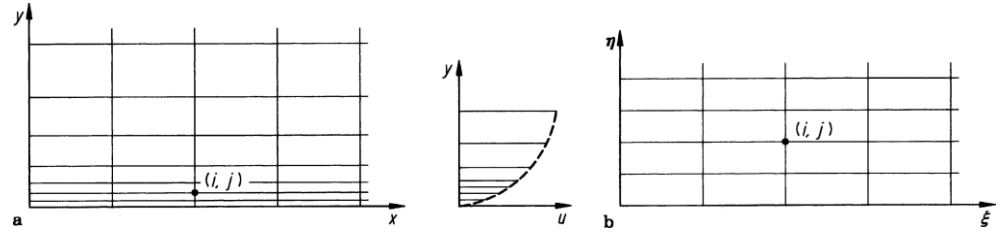


Fig. 6.3 Example of grid stretching. (a) Physical plane. (b) Computational plane

$$\xi = x \quad (6.25a)$$

$$\eta = \ln(y+1) \quad (6.25b)$$

التحول العكسي هو

$$x = \xi \quad (6.26a)$$

$$y = e\eta - 1 \quad (6.26b)$$

المقاييس المعكوسة يتم الحصول عليها على النحو التالي:

في المعادلة (6.22)، يتم التعرف على محددات القاسم كمحدد مصفوفه جاكوبي Jacobian determinant،

$$J = e\eta \quad \text{المرموز لها بواسطة:}$$

وبالتالي، المعادلة (6.22) يمكن أن تكتب:

$$\frac{\partial x}{\partial \xi} = 1; \quad \frac{\partial x}{\partial \eta} = 0; \quad \frac{\partial y}{\partial \xi} = 0; \quad \frac{\partial y}{\partial \eta} = e \quad (6.27)$$

دعونا ننظر في معادلة الاستمرارية، التي مهدت لهذه المعادلة (6.27). لتدفق ثنائي الأبعاد:

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \quad (6.28)$$

المعادلة (6.27) هي معادلة الاستمرارية مكتوبة من حيث التخطيط الفيزيائي. هذه المعادلة يمكن أن تتحول من قبل

النتائج العامة التي قدمها (6.23) and (6.24)، إلى

$$\frac{1}{J} \left[\frac{\partial(\rho u)}{\partial \xi} \left(\frac{\partial y}{\partial \eta} \right) - \frac{\partial(\rho u)}{\partial \eta} \left(\frac{\partial y}{\partial \xi} \right) \right] + \frac{1}{J} \left[\frac{\partial(\rho v)}{\partial \eta} \left(\frac{\partial x}{\partial \xi} \right) - \frac{\partial(\rho v)}{\partial \xi} \left(\frac{\partial x}{\partial \eta} \right) \right] = 0 \quad (6.29)$$

باستبدال المعادلة (6.29) في المقاييس المعكوسة من المعادلة (6.27):

$$e^\eta \frac{\partial(\rho u)}{\partial \xi} + \frac{\partial(\rho v)}{\partial \eta} = 0 \quad (6.30)$$

المعادلة (6.30) هي معادلة الاستمرارية في التخطيط الحاسوبي. كما يمكن الحصول عليها من التحويل المباشر الذي قدمته Eqs. (6.25a and b). هنا، والمقاييس هي:

$$\frac{\partial \xi}{\partial x} = 1; \quad \frac{\partial \xi}{\partial y} = 0; \quad \frac{\partial \eta}{\partial x} = 0; \quad \frac{\partial \eta}{\partial y} = \frac{1}{y+1} \quad (6.31)$$

باستخدام التحولات التي قدمتها Eqs. (6.2) و (6.3)، تصبح المعادلة (6.28) كالتالي:

$$\frac{\partial(\rho u)}{\partial \xi} \left(\frac{\partial \xi}{\partial x} \right) + \frac{\partial(\rho u)}{\partial \eta} \left(\frac{\partial \eta}{\partial x} \right) + \frac{\partial(\rho v)}{\partial \xi} \left(\frac{\partial \xi}{\partial y} \right) + \frac{\partial(\rho v)}{\partial \eta} \left(\frac{\partial \eta}{\partial y} \right) = 0 \quad (6.32)$$

باستبدال المعادلة (6.32) في المقاييس في المعادلة (6.31)، يصبح لدينا:

$$\frac{\partial(\rho u)}{\partial \xi} + \frac{1}{y+1} \frac{\partial(\rho v)}{\partial \eta} = 0 \quad (6.33)$$

من المعادلة (6.26b)، $y+1 = e^\eta$. تصبح المعادلة (6.33):

$$\frac{\partial(\rho u)}{\partial \xi} + \frac{1}{e^\eta} \frac{\partial(\rho v)}{\partial \eta} = 0$$

or

$$e^\eta \frac{\partial(\rho u)}{\partial \xi} + \frac{\partial(\rho v)}{\partial \eta} = 0 \quad (6.34)$$

المعادلة (6.34) مطابقة للمعادلة (6.30). كل ما قمنا به هنا هو شرح كيفية الحصول على المعادلات تحولت إما تحول مباشر أو تحول عكسي. النتائج هي نفسها. مثال على شبكة أكثر تعقيدا تتمدد، في كل من الاتجاهات X و y ، كما ورد في المرجعين [2، 3]. هنا، ندرس التدفق اللزج الأسرع من سرعة الصوت على قاعدة حادة. وتوضح التخطيطات الفيزيائية والحسابية في Fig. 6.4. أن التحكم بالسائل المتمدد ينجز من خلال تحويلات تستخدم من قبل هولست Holst [4].

$$x = \frac{\xi_0}{A} [\sinh((\xi - x_0)\beta_x) + A]$$

where

$$A = \sinh(\beta_x x_0)$$

and

$$x_0 = \frac{1}{2\beta_x} \ln \left[\frac{1 + (e^{\beta_x} - 1)\xi_0}{1 + (e^{-\beta_x} - 1)\xi_0} \right]$$

حيث ξ_0 هو الموقع في التخطيط الحسابي حيث الحد الأقصى للتشابك، و β_x هو الثابت الذي يسيطر على درجة من التشابك في ξ_0 .

مع قيم أكبر من β_x توفر شبكة دقيقة في المنطقة المتشابكة. ويتم إنجاز عرضية تمتد بقسمة التخطيط الفيزيائي إلى قسمين: (1) الحيز المباشر وراء هذه الخطوة، و (2) في الحيز التالي (سواء أمام وخلف) للخطوة. ويستند هذا التحول على تلك المستخدمة من قبل روبرتس Roberts [5]، وتعطى من خلال:

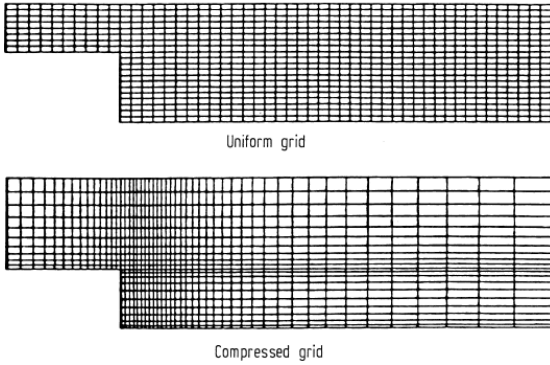


Fig. 6.4 Comparison of uniform and compressed grid

$$y = \frac{(\beta_y + 1) - (\beta_y - 1)e^{-c(\eta - 1 - \alpha)/(1 - \alpha)}}{(2\alpha + 1)(1 + e^{-c(\eta - 1 - \alpha)/(1 - \alpha)})}$$

where

$$c = \log \left(\frac{\beta_y + 1}{\beta_y - 1} \right)$$

α و β_y الثوابت القابلة للاستخدام، وتختلف عن القسمين اللذين تم تحديدهما أعلاه. التحولات الجبرية الواردة أعلاه

نتجت عن شبكة تمدد كما هو مبين في Fig. 6.4.

Boundary-Fitted Coordinate Systems 22.5

الآن نعتبر أن التدفق يجري خلال قناة تتباعد كما هو مبين في (Fig. 6.5 a). منحنى de هو الجدار العلوي من القناة، وخط fg هو خط المنتصف لهذا التدفق. الشبكة المستطيلة البسيطة في التخطيط الفيزيائي ليست مناسبة، للأسباب التي ذكرناها في المقطع 6.1. (Sect. 6.1) بدلا من ذلك، سنستخدم الشبكة المنحنية في (Fig. 6.5 a) التي تسمح لكل من الحدود العليا و fg المنتصف أن تكون خطوط منسقة، بما يناسب بالضبط هذه الحدود. في المقابل، فإن شبكة الخطوط المنحنية في (Fig. 6.5(a) يجب أن تتحول إلى شبكة مستطيلة في التخطيط الحاسوبي، (Fig. 6.5(b)). ويمكن تحقيق ذلك على النحو التالي. السماح $(y_s = f(x))$ ليكون تنسيق من المساحة de العلوي في (Fig. 6.5(a)). ثم التحول التالي سوف يؤدي إلى شبكة مستطيلة في البعد (ξ, η) :

$$\xi = x$$

$$\eta = y/y_s, y_s = f(x)$$

ما سبق هو مثال بسيط من الحدود المركبة على نظام الإحداثيات. ويرد مثال أكثر تطورا في Fig. 6.6، و وضع القضية موضع في Fig. 6.2.

لننظر في أمر الجنيح الوارد في (Fig. 6.6a). نظام منحنى الأضلاع يلف حول الجنيح، حيث تنسيق الخط $\eta = \eta_1 = \eta_2$ ثابت على سطح الجنيح. هذه هي الحدود الداخلية للشبكة، المعروفة ب Γ_1 . و الحدود الخارجية من الشبكة تعرف ب Γ_2 في Fig. 6.6a، وتُعطى بواسطة $\eta = \eta_2 = \eta_1$ ثابت على سطح الجنيح.

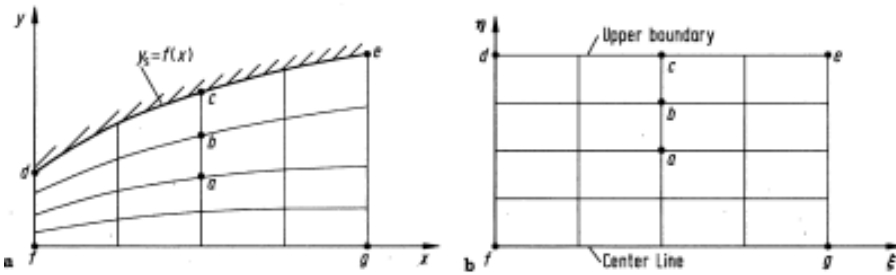
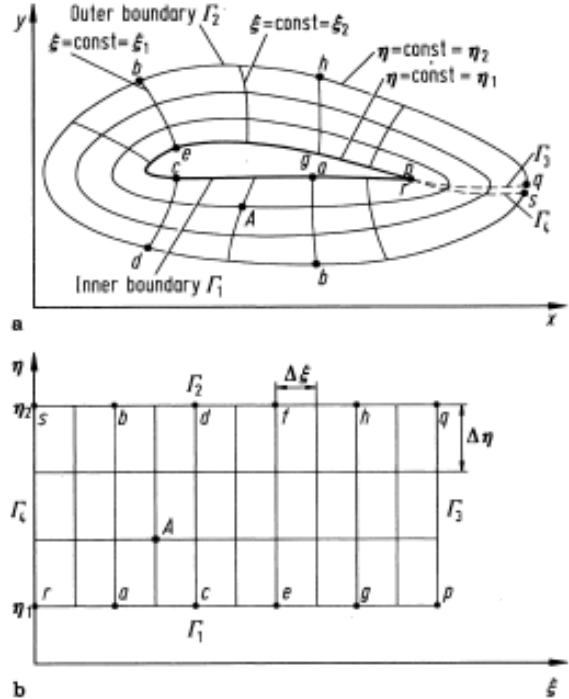


Fig. 6.5 A simple boundary-fitted coordinate system. (a) Physical plane. (b) Computational plane

الخطوط المنتشرة على الحدود الداخلية Γ_1 والتي تتقاطع و الحدود الخارجية Γ_2 هي خطوط ξ ثابت، مثل خط ef ذو $\xi = \xi_1 = \xi_2$ ثابت. (لاحظ أن في Fig. 6.6(a) خطوط من η ثابتة ترافق الجنيح تماما، مثل الكثير من الدوائر الممدودة. وتسمى مثل هذه الشبكة '0'، صلة اخرة للشبكة المنحنية يمكن أن تكون $\eta = \eta_1 = \eta_2$ خطوط ثابتة متابعة

للمجرى إلى اليمين، و ليست مرفقة تماما بالجنيح (إلا على الحدود الداخلية Γ_1). وتسمى مثل هذه الشبكة: الشبكة 'c'. سوف نرى مثالا على نوع الشبكة 'c' قريبا).

Fig. 6.6 (a) Physical plane.
(b) Computational plane



السؤال: ما هو التحويل الذي يمكن أن يلقي الشبكة المنحنية في (Fig.6.6a) في شبكة موحدة في التخطيط الحسابي كما في (Fig.6.6b)؟ للإجابة على هذا السؤال، لاحظ في (Fig.6.6a) أن طول الحدود الداخلية Γ_1 ، وتعرف الإحداثيات الفيزيائية للجسم:

$$(x, y) \text{ معروف على طول } \Gamma_1$$

وبالمثل، الإحداثيات الفيزيائية للحدود الخارجية Γ_2 معروفة أيضا، لأن Γ_2 هو مجرد حلقة تقريبية تم رسمها بشكل عشوائي حول الجنيح. مرة واحدة يتم تحديد هذه الحلقة Γ_2 ، ثم الإحداثيات الفيزيائية تصبح معروفة على طول ذلك:

$$(x, y) \text{ معروفة على طول } \Gamma_2$$

هذا يلمح لوجود مشكلة في قيمة الحدود حيث نعرف الشروط الحدودية (وهي قيم x و y) في كل مكان على طول الحدود. أذكر من Sect. 4.3.3 أن حل المعادلات التفاضلية الجزئية الإهليلجية (elliptic) الشكل يتطلب مواصفات شروط الحدود في كل مكان على طول الحدود داخل المجال. لذلك، دعونا ننظر للتحويل في Fig. 6.6 الذي تحدده

Sect. 6.4. المعادلة التفاضلية الجزئية الإهليلجية الشكل (على النقيض من علاقة جبرية كما هو موضح في المقطع. 6.4

(6.4). واحد من أبسط المعادلات الإهليلجية الشكل هي معادلة لابلاس (Laplace):

حيث لدينا شروط الحدود ديريتشليت Dirichlet

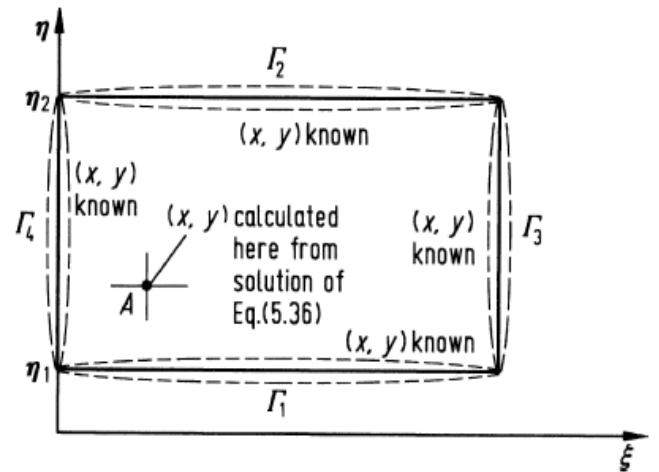
$$\Gamma_1 = \eta = \eta_1 \text{ ثابت على } \Gamma_1$$

$$\Gamma_2 = \eta = \eta_2 \text{ ثابت على } \Gamma_2$$

$$\xi = \xi(x, y) \text{ يتم تحديد على كلا } \Gamma_1 \text{ و } \Gamma_2$$

من المهم أن نأخذ في عين الاعتبار ما نقوم به هنا. المعادلات (6.25 a and b) لا علاقة لها بفيزياء مجال التدفق شيئاً. هم ببساطة المعادلات التفاضلية الجزئية الإهليلجية الشكل التي اخترناه لربط ξ و η ب x و y ، وبالتالي تشكل تحولا (المطابقة لواحدة واحدة من نقاط الشبكة) من التخطيط الفيزيائي إلى التخطيط الحاسوبي. لأنه هذا التحول يخضع للمعادلات الإهليلجية الشكل، هو مثال على الطبقة العامة من شبكة تسمى انشاء شبكة الإهليلجية (elliptic).

Fig. 6.7 Computational plane, illustrating the boundary conditions and an internal point



دعونا نلقي نظرة عن كثب على التخطيط الفيزيائي والحاسوبي المبين في Fig. 6.6. من أجل بناء شبكة مستطيلة في التخطيط الحاسوبي (Fig. 6.6b)، يجب أن يتم خفض التخطيط الفيزيائي (Fig. 6.6a) على حافة زائدة من الجنيح. هذا الخفض يمكن تصوره كاثنين من الخطوط المتراكبة على بعضها البعض: خط pq المرموز له بواسطة Γ_3 يمثل خط الحدود للحيز الفيزيائي فوق pq، وخط rs الذي نرمز اليه بواسطة Γ_4 يمثل خط الحدود للحيز الفيزيائي دون rs. في

التخطيط الفيزيائي، النقاط p و r هي نفس عينها، و q و s هما نفس النقطة. في (Fig. 6.6a) نبعث قليلا عن الوضوح. ومع ذلك، في التخطيط الحاسوبي، هذه النقاط كلها مختلفة. في الواقع، يتم الحصول على الشبكة في التخطيط الحاسوبي عبر تفصيل للشبكة الفيزيائية في التقطيع، ثم "إزالة تغليف" الشبكة من الجنيح. على سبيل المثال، سطح الجنيح في التخطيط الفيزيائي، ومنحنى $pgecar$ ، يصبح خط مستقيم أقل من الرمز بواسطة Γ_1 في التخطيط الحاسوبي. وبالمثل، فإن الحدود الخارجية $ghfdbs$ تصبح خط مستقيم علوي نمرز له بواسطة Γ_2 في التخطيط الحاسوبي. يشكل الجانبين الأيمن والأيسر من المستطيل في التخطيط الحاسوبي قطع من التخطيط الفيزيائي؛ الجانب الأيسر هو خط rs يرمز اليه بواسطة Γ_4 في Fig. 6.6(b)، وعلى الجانب الأيمن هو خط pq يرمز اليه بواسطة Γ_3 في Fig. 6.6(b). ويُرسَم التخطيط الحاسوبي مرة أخرى في Fig. 6.7. نحن هنا نؤكد معرفة قيمة (x, y) على طول كل الحدود الأربعة، Γ_1 ، Γ_2 ، Γ_3 و Γ_4 . الجانب الرئيسي لمنهج انشاء شبكة الإهليلجية (elliptic grid generation) الشكل هو أنه، مع شروط الحدود يتم حل (Eqs. 6.35a and b) لقيمة (x, y) التي تنطبق على جميع النقاط الداخلية. وتعطى مثالا على مثل هذه النقاط الداخلية من خلال النقطة (A) في Fig. 6.7، والتي تتطابق مع نفس النقطة (A) في (Figs. 6.6(a) and (b)). في الواقع، لحل المعادلات نعتمد على معكوس (Eqs. 6.35a and b)، وهذه المعادلات التي تم الحصول عليها من المعادلات 6.35a and b تتبادل في المتغيرات التابعة والمستقلة. والنتيجة هي:

$$\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} = 0 \quad (6.36a)$$

$$\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi \partial \eta} + \alpha \frac{\partial^2 y}{\partial \eta^2} = 0 \quad (6.36b)$$

where

$$\alpha = \left(\frac{\partial x}{\partial \eta} \right)^2 + \left(\frac{\partial y}{\partial \eta} \right)^2$$

$$\beta = \left(\frac{\partial x}{\partial \xi} \right) \left(\frac{\partial x}{\partial \eta} \right) + \left(\frac{\partial y}{\partial \xi} \right) \left(\frac{\partial y}{\partial \eta} \right)$$

$$\gamma = \left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2$$

نلاحظ في (Eqs. 6.36a and b) أن x و y يتم التعبير عنهم الآن كمتغيرات تابعة. نعود مرة أخرى إلى Fig. 6.7, Eqs. (6.36a and b) تحل هذه المعادلة، بالموازاة مع شروط الحدود نظرا ل (x, y) على Γ_1 ، Γ_2 ، Γ_3 و Γ_4 ، للحصول على

قيمة (x, y) التي تتوافق مع نقاط الشبكة المتباعدة بشكل موحد في التخطيط الحاسوبي (ξ, η) . وهكذا، فإن أي نقطة تقع في شبكة معينة في التخطيط الحاسوبي (ξ_i, η_j) تتوافق مع نقطة في شبكة حسابية في الحيز الفيزيائي (x_i, y_j) .

حل (Eqs. (6.36a and b) أن يتم بحل هذه الفروق المحدودة المناسبة للمعادلات الإهليلجية الشكل. على سبيل المثال، تقنيات الاسترخاء مستعملة كثيرا لمثل هذه المعادلات. لاحظ أن التحول المذكور أعلاه، يستخدم المعادلة التفاضلية الجزئية الإهليلجية الشكل لتوليد الشبكة، لا تنطوي على تعابير تحليلية مغلقة في المعادلات التحليلية؛ بدلا من ذلك، فإنها تنتج مجموعة من الأرقام والتي تحدد نقاط الشبكة (x_i, y_j) في الحيز الفيزيائي و التي تتوافق مع نقطة في شبكة معينة (ξ_i, η_j) في الحيز الحاسوبي. في المقابل، يتم الحصول على المقاييس في المعادلات التي تحكم التدفق (التي تحل في التخطيط الحاسوبي)، مثل $\partial \xi / \partial x$ ، $\partial \eta / \partial y$ ، وما إلى ذلك من الفروق المحدودة. وكثيرا ما تستخدم العناصر المنتهية والمراكز المنتهية لهذا الغرض، فإن المنحني و نظام إحداثيات الحدود المركبة، المبينين في Fig. 6.6(a) وتوضح ببساطة المعنى النوعي لأغراض وتعليمات معينة. في الحقيقة الشبكة المتولدة عن وجود الجنيح باستخدام منهج انشاء شبكة الإهليلجية (elliptic grid generation) الشكل مبين أعلاه في Fig. 6.8، انظر المرجع [7]. باستخدام مخطط انشاء شبكة Thompson (المرجع [6])، ([7]) التي ولدت نظام الإحداثيات في الحدود المجهزة حول جنيح مايلي Miley. (وجنيح مايلي Miley هو الجنيح المصمم خصيصا لتطبيقات قاعدة عدد رينولدز Reynolds من قبل ستان مايلي Stan Miley في جامعة ولاية ميسيسيبي Mississippi). في Fig. 6.6 البقعة البيضاء في منتصف الشكل هي الجنيح، والشبكة تنتشر بعيدا عن الجنيح في كل الاتجاهات.

في المرجع [7] تدفقات قاعدة رقم رينولدز Reynolds على الجنيحات خلال الوقت تعتمد على حل الفروق المحدودة في معادلات الانضغاط لنافيير ستوكس Navier-Stokes (وسناقش مثل هذه الحلول المعتمدة على الزمن في Chap. 7). التيار الحر الذي هو دون سرعة الصوت، وبالتالي يجب وضع الحدود الخارجية بعيدا عن الجنيح بسبب انتشارات بعيدة المدى من اضطرابات في تدفق دون سرعة الصوت. وترد التفاصيل من الشبكة في المحيط القريب من الجنيح في Fig. 6.9. نلاحظ في كل من Figs. 6.8 and 6.9 أن نوع الشبكة هو "C"، على النقيض من نوع الشبكة '0' التي رسمت في Fig. 6.6. نوضع حد لهذا القسم من خلال التأكيد مرة أخرى على أن انشاء شبكة الإهليلجية (elliptic)

(grid generation)، مع حل لها من المعادلات التفاضلية الجزئية الإهليلجية الشكل للحصول على نقاط الشبكة الداخلية، منفصل تماما عن حل الفروق المحدودة من المعادلات التي تحكمها.

أولا يتم إنشاء شبكة، قبل محاولة أي حل للمعادلات التي تحكمها. ثم استخدام معادلة لابلاس Laplace (المعادلات (6.35 a and b)) للحصول على هذه الشبكة، لا توجد اي علاقة مع الجوانب الفيزيائية لمجال التدفق الفعلي. هنا، نستخدم معادلة لابلاس Laplace ببساطة لتوليد الشبكة فقط.

الآن نعتبر أن التدفق يجري خلال قناة تتباعد كما هو مبين في Fig. 6.5(a). منحنى de هو الجدار العلوي من القناة، وخط fg هو خط المنتصف لهذا التدفق. الشبكة المستطيلة البسيطة في التخطيط الفيزيائي ليست مناسبة، للأسباب التي ذكرناها في المقطع 6.1 (Sect. 6.1) بدلا من ذلك، سنستخدم الشبكة المنحنية في Fig. 6.5(a) التي تسمح لكل من de الحدود العليا و fg المنتصف أن تكون خطوط منسقة، بما يناسب بالضبط هذه الحدود. في المقابل، فإن شبكة الخطوط المنحنية في Fig. 6.5(a) يجب أن تتحول إلى شبكة مستطيلة في التخطيط الحاسوبي، Fig. 6.5(b). ويمكن تحقيق ذلك على النحو التالي. السماح $ys = f(x)$ ليكون تنسيق من المساحة de العلوي في Fig. 6.5(a). ثم التحول التالي سوف يؤدي إلى شبكة مستطيلة في البعد (ξ, η) :

$$\xi = x$$

$$\eta = y/ys \quad \text{where } ys = f(x)$$

ما سبق هو مثال بسيط من الحدود المركبة على نظام الإحداثيات. ويرد مثال أكثر تطورا في Fig. 6.6، و وضع القضية موضح في Fig. 6.2.

لنظر في أمر الجنيح الوارد في Figure 6.6(a). نظام منحنى الأضلاع يلف حول الجنيح، حيث تنسيق الخط $\eta = \eta_1 =$ ثابت على سطح الجنيح. هذه هي الحدود الداخلية للشبكة، المعروفة ب Γ_1 . و الحدود الخارجية من الشبكة تعرف ب Γ_2 في Figure 6.6(a)، وتُعطى بواسطة $\eta = \eta_2 =$ ثابت على سطح الجنيح.

الخطوط المنتشرة على الحدود الداخلية Γ_1 والتي تتقاطع و الحدود الخارجية Γ_2 هي خطوط ξ الثابت، مثل خط ef ذو $\xi = \xi_1 = \xi_2 =$ ثابت. (لاحظ أن في Fig. 6.6(a) خطوط من η ثابتة ترافق الجنيح تماما، مثل الكثير من الدوائر الممدودة. وتسمى مثل هذه الشبكة '0'، صلة اخرة للشبكة المنحنية يمكن أن تكون $\eta =$ خطوط ثابتة متابعة للمجرى إلى

اليمين، و ليست مرفقة تماما بالجنيح (إلا على الحدود الداخلية Γ_1). وتسمى مثل هذه الشبكة: الشبكة 'C'. سوف نرى مثالا على نوع الشبكة 'C' قريبا.)

السؤال: ما هو التحول الذي يمكن أن يلقي الشبكة المنحنية في Fig.6.6(a) في شبكة موحدة في التخطيط الحسابي كما في Fig.6.6(b)؟ للإجابة على هذا السؤال، لاحظ في Fig. 6.6(a) أن طول الحدود الداخلية Γ_1 ، وتعرف الإحداثيات الفيزيائية للجسم: (x, y) معروف على طول Γ_1

وبالمثل، الإحداثيات الفيزيائية للحدود الخارجية Γ_2 معروفة أيضا، لأن Γ_2 هو مجرد حلقة تقريبية تم رسمها بشكل عشوائي حول الجنيح. مرة واحدة يتم تحديد هذه الحلقة Γ_2 ، ثم الإحداثيات الفيزيائية تصبح معروفة على طول ذلك: (x, y) معروفة على طول Γ_2

هذا يلزم لوجود مشكلة في قيمة الحدود حيث نعرف الشروط الحدودية (وهي قيم x و y) في كل مكان على طول الحدود. أذكر من Sect. 4.3.3 أن حل المعادلات التفاضلية الجزئية الإهليلجية الشكل يتطلب مواصفات شروط الحدود في كل مكان على طول الحدود داخل المجال. لذلك، دعونا ننظر للتحويل في Fig. 6.6 الذي تحدده المعادلة التفاضلية الجزئية الإهليلجية الشكل (على النقيض من علاقة جبرية كما هو موضح في المقطع 6.4 Sect. 6.4). واحد من أبسط المعادلات الإهليلجية الشكل هي معادلة لابلاس Laplace:

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0 \quad (6.35a)$$

$$\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = 0 \quad (6.35b)$$

حيث لدينا شروط الحدود ديريتشليت Dirichlet

$$\Gamma_1 = \eta = \eta_1 \text{ ثابت على } \Gamma_1$$

$$\Gamma_2 = \eta = \eta_2 \text{ ثابت على } \Gamma_2$$

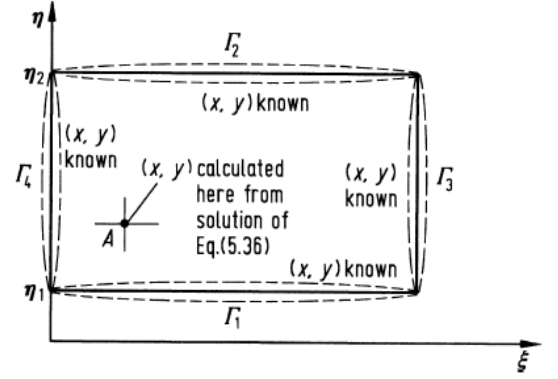
و $\xi = \xi(x, y)$ يتم تحديد على كلا Γ_1 و Γ_2

من المهم أن نأخذ في عين الاعتبار ما نقوم به هنا. المعادلات (6.35a and b) لا علاقة لها بفيزياء مجال التدفق شيئاً. هم ببساطة المعادلات التفاضلية الجزئية الإهليلجية الشكل التي اخترناها لربط ε و η ب x و y ، وبالتالي تشكل تحولا (المطابقة لواحدة واحدة من نقاط الشبكة) من التخطيط الفيزيائي إلى التخطيط الحاسوبي. لأنه هذا التحول يخضع للمعادلات الإهليلجية الشكل، هو مثال على الطبقة العامة من شبكة تسمى انشاء شبكة الإهليلجية الشكل. وقد استُخدمت انشاء شبكة الإهليلجية الشكل عمليا من قبل جو تومسون Joe Thompson في جامعة ولاية Mississippi، ووصفت بالتفصيل في المرجع [6].

دعونا نلقي نظرة عن كثب على التخطيط الفيزيائي والحاسوبي المبين في Fig. 6.6. من أجل بناء شبكة مستطيلة في التخطيط الحاسوبي (Fig. 6.6b)، يجب أن يتم خفض التخطيط الفيزيائي (Fig. 6.6a) على حافة زائدة من الجنيح. هذا الخفض يمكن تصوره كاثنين من الخطوط المتراكبة على بعضها البعض: خط pq المرموز له بواسطة Γ_3 يمثل خط الحدود للحيز الفيزيائي فوق pq ، وخط rs الذي نرمز اليه بواسطة Γ_4 يمثل خط الحدود للحيز الفيزيائي دون rs . في التخطيط الفيزيائي، النقاط p و r هي نفس عينها، و q و s هما نفس النقطة. في Fig. 6.6(a) نبعد قليلا عن الوضوح. ومع ذلك، في التخطيط الحاسوبي، هذه النقاط كلها مختلفة. في الواقع، يتم الحصول على الشبكة في التخطيط الحاسوبي عبر تفصيل للشبكة الفيزيائية في التقطيع، ثم "إزالة تغليف" الشبكة من الجنيح. على سبيل المثال، سطح الجنيح في التخطيط الفيزيائي، ومنحنى $pqecar$ ، يصبح خط مستقيم أقل من الرمز بواسطة Γ_1 في التخطيط الحاسوبي. وبالمثل، فإن الحدود الخارجية $ghfdbs$ تصبح خط مستقيم علوي نرمز له بواسطة Γ_2 في التخطيط الحاسوبي. يشكل الجانبين الأيمن والأيسر من المستطيل في التخطيط الحاسوبي قطع من التخطيط الفيزيائي؛ الجانب الأيسر هو خط rs يرمز اليه بواسطة Γ_4 في Fig. 6.6(b)، وعلى الجانب الأيمن هو خط pq يرمز اليه بواسطة Γ_3 في Fig. 6.6(b). ويُرسم التخطيط الحاسوبي مرة أخرى في Fig. 6.7. نحن هنا نؤكد معرفة قيمة (x, y) على طول كل الحدود الأربعة، Γ_1 ، Γ_2 ، Γ_3 و Γ_4 . الجانب الرئيسي لمنهج انشاء شبكة الإهليلجية الشكل هو أنه، مع شروط الحدود يتم حل Eqs. (6.35a and b) لقيمة (x, y) التي تنطبق على جميع النقاط الداخلية. وتعطى مثالا على مثل هذه النقاط الداخلية من خلال النقطة (A) في Fig. 6.7، والتي تتطابق مع نفس النقطة (A) في Figs. 6.6(a) and (b). في الواقع، لحل المعادلات نعتمد

على معكوس (6.35a and b) Eqs.، وهذه المعادلات التي تم الحصول عليها من (6.35a and b) Eqs. تتبادل في المتغيرات التابعة والمستقلة. والنتيجة هي:

Fig. 6.7 Computational plane, illustrating the boundary conditions and an internal point



$$\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} = 0 \quad (6.36a)$$

$$\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 y}{\partial \eta^2} = 0 \quad (6.36b)$$

where

$$\alpha = \left(\frac{\partial x}{\partial \eta} \right)^2 + \left(\frac{\partial y}{\partial \eta} \right)^2$$

$$\beta = \left(\frac{\partial x}{\partial \xi} \right) \left(\frac{\partial x}{\partial \eta} \right) + \left(\frac{\partial y}{\partial \xi} \right) \left(\frac{\partial y}{\partial \eta} \right)$$

$$\gamma = \left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2$$

نلاحظ في (6.36a and b) Eqs. أن x و y يتم التعبير عنهم الآن كمتغيرات تابعة. نعودة مرة أخرى إلى Fig. 6.7، Eqs. (6.36a and b) تحل هذه المعادلة، بالموازاة مع شروط الحدود نظراً لـ (x, y) على Γ_1 ، Γ_2 ، Γ_3 و Γ_4 ، للحصول على قيمة (x, y) التي تتوافق مع نقاط الشبكة المتباعدة بشكل موحد في التخطيط الحاسوبي (ξ, η) . وهكذا، فإن أي نقطة تقع في شبكة معينة في التخطيط الحسابي (ξ_i, η_j) تتوافق مع نقطة في شبكة حسابية في الحيز الفيزيائي (x_i, y_j) .

حل (6.36a and b) Eqs. أن يتم بحل هذه الفروق المحدودة المناسبة للمعادلات الإهليلجية الشكل. على سبيل المثال، تقنيات الاسترخاء مستعملة كثيراً لمثل هذه المعادلات. لاحظ أن التحول المذكور أعلاه، يستخدم المعادلة التفاضلية الجزئية الإهليلجية الشكل لتوليد الشبكة، لا تنطوي على تعابير تحليلية مغلقة في المعادلات التحليلية؛ بدلاً من ذلك، فإنها تنتج مجموعة من الأرقام والتي تحدد نقاط الشبكة (x_i, y_j) في الحيز الفيزيائي و التي تتوافق مع نقطة في

شبكة معينة (η_j, ξ_i) في الحيز الحاسوبي. في المقابل، يتم الحصول على المقاييس في المعادلات التي تحكم التدفق (التي تحل في التخطيط الحاسوبي)، مثل $\partial \xi / \partial x$ ، $\partial \eta / \partial y$ ، وما إلى ذلك من الفروق المحدودة. وكثيرا ما تستخدم العناصر المنتهية والمراكز المنتهية لهذا الغرض، فإن المنحني و نظام إحداثيات الحدود المركبة، المبينين في Fig. 6.6(a) وتوضح ببساطة المعنى النوعي لأغراض وتعليمات معينة. في الحقيقة الشبكة المتولدة عن وجود الجنيح باستخدام منهج انشاء شبكة الإهليلجية الشكل مبين أعلاه في Fig. 6.8، انظر المرجع. [7]. باستخدام مخطط انشاء شبكة طومسون Thompson (المرجع [6])، ([7]) التي ولدت نظام الإحداثيات في الحدود المجهزة حول جنيح مايلي Miley. (وجنيح مايلي Miley هو الجنيح المصمم خصيصا لتطبيقات قاعدة عدد رينولدز Reynolds من قبل ستان مايلي Stan Miley في جامعة ولاية ميسيسيبي Mississippi). في Fig. 6.6 البقعة البيضاء في منتصف الشكل هي الجنيح، والشبكة تنتشر بعيدا عن الجنيح في كل الاتجاهات.

في المرجع. [7] تدفقات قاعدة رقم رينولدز Reynolds على الجنيحات خلال الوقت تعتمد على حل الفروق المحدودة في معادلات الانضغاط لنافير ستوكس Navier-Stokes (وسنناقش مثل هذه الحلول المعتمدة على الزمن في Chap. 7). التيار الحر الذي هو دون سرعة الصوت، وبالتالي يجب وضع الحدود الخارجية بعيدا عن الجنيح بسبب انتشارات بعيدة المدى من اضطرابات في تدفق دون سرعة الصوت. وترد التفاصيل من الشبكة في المحيط القريب من الجنيح في Fig. 6.9. نلاحظ في كل من Figs. 6.8 and 6.9 أن نوع الشبكة هو "C" ، على النقيض من نوع الشبكة '0' التي رسمت في Fig. 6.6. نوضع حد لهذا القسم من خلال التأكيد مرة أخرى على أن انشاء شبكة الإهليلجية (elliptic grid generation)، مع حل لها من المعادلات التفاضلية الجزئية الإهليلجية الشكل للحصول على نقاط الشبكة الداخلية، منفصل تماما عن حل الفروق المحدودة من المعادلات التي تحكمها.

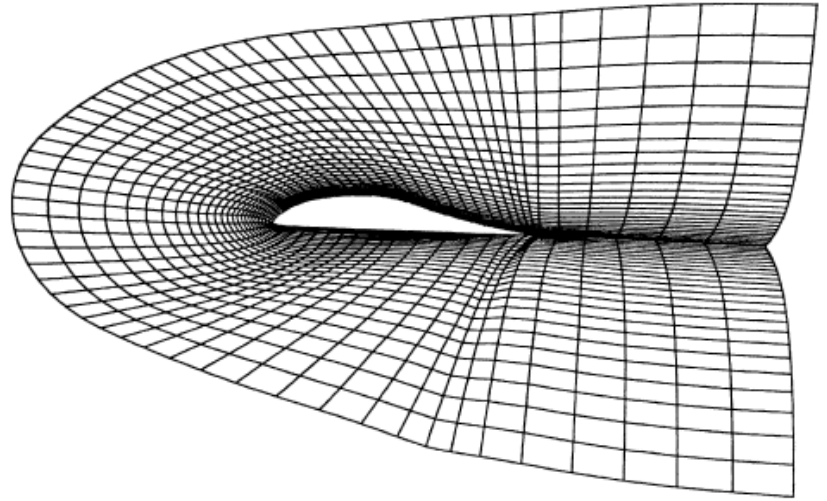


Fig. 6.9 A detail of the boundary fitted grid (from Ref. [7])

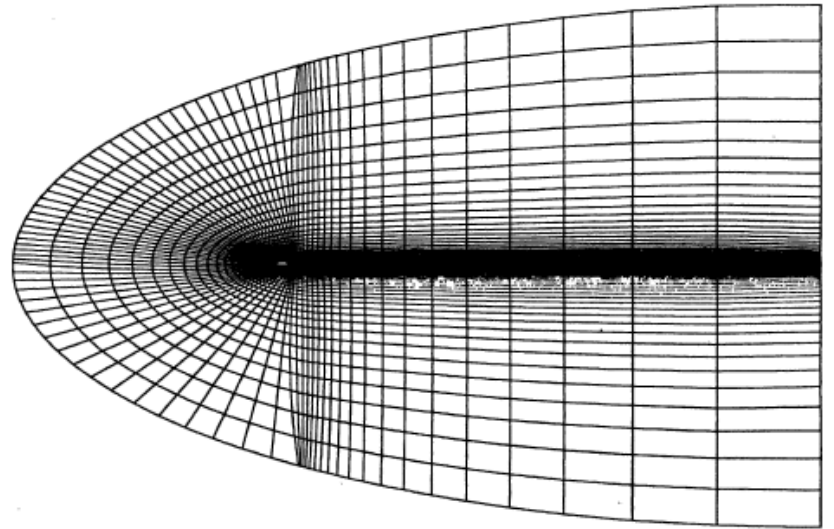


Fig. 6.8 Boundary fitted grid (from Ref. [7])

أولاً يتم إنشاء شبكة، قبل محاولة أي حل للمعادلات التي تحكمها. ثم استخدام معادلة لابلاس Laplace (المعادلة (6.35a and b)) للحصول على هذه الشبكة، لا توجد أي علاقة مع الجوانب الفيزيائية لمجال التدفق الفعلي. هنا، نستخدم معادلة لابلاس Laplace ببساطة لتوليد الشبكة فقط.

22.6 الشبكة التكيفية (Adaptive Grid)

الشبكة التكيفية هي شبكة لعدة شبكات (network) حيث تُكثف نقاط للشبكة (grid points) تلقائياً في مناطق ذات مجال تدفق عالي. الحل لخصائص حقل التدفق (flow) في نقاط الشبكة تكون في التخطيط الفيزيائي. شبكة

التكيف تتطور مع الوقت بالتزامن مع وقت حل يعتمد على المعادلات التي تحكم مجال التدفق، والذي يحسب متغيرات مجال التدفق في مراحل من الوقت. أثناء الحل، حيث تكون نقاط الشبكة في التخطيط الفيزيائي في مثل هذه الحالة "للتكيف" مع المناطق ذات درجات التدفق العالي. وبالتالي، فإن نقاط الشبكة الفعلية في التخطيط الفيزيائي هي باستمرار في الحركة خلال إيجاد حل لمجال التدفق، وتصبح ثابتة فقط عندما يقترب التدفق الى حالة مستقرة. وبالتالي، فخلافا لإنشاء شبكة الإهليلجية الشكل المناقشة في المقطع 6.5 حيث الحل في انشاء الشبكة منفصل تماما عن الحل في مجال التدفق، و الذي يرتبط بالشبكة التكيفية ارتباطا وثيقا مع حل حقل التدفق، الذي يتغير مع التغيرات في مجال التدفق. الهدف من مزايا وضع شبكة التكيف لجميع نقاط الشبكة في المناطق التي يتم فيها حدوث "العمل". هذه المزايا هي: (1) زيادة الدقة لعدد محدد من نقاط الشبكة، أو (2)، لدقة معينة، وهناك حاجة إلى نقاط أقل في الشبكة. شبكات التكيف لا تزال جديدة للغاية في CFD، سواء أنخزت هذه المزايا أم لا فهي غير راسخة. مثال بسيط على شبكة التكيف الذي استخدم من قبل كوردا Corda [8] من أجل حل تدفق لزج أسرع من الصوت. والتحول معرب عنه في شكل:

$$\Delta x = \frac{B\Delta\xi}{1 + b \frac{\partial g}{\partial x}} \quad (6.37)$$

$$\Delta y = \frac{C\Delta\eta}{1 + c \frac{\partial g}{\partial y}} \quad (6.38)$$

حيث g هو متغير حقل التدفق الأصلي، مثل p, ρ أو T . إذا $g = p$ ، اذا Eqs. (6.37) and (6.38) حيث مجموعة نقاط الشبكة تقع في مناطق ذات فروقات كبيرة في الضغط؛ إذا كانت $T = g$ فإن نقاط الشبكة العنقودية تتمركز في مناطق ذات فروقات حرارية مرتفعة وهكذا دواليك. في Eqs. (6.37) و (6.38)، يتم إصلاح $\Delta\xi$ و $\Delta\eta$ ، على شبكة موحدة في التخطيط الحاسوبي (ξ, η) ، و b و c ثوابت مختارة لزيادة أو تقليل تأثير التدرج مع تغيير تباعد الشبكة في التخطيط الفيزيائي، و B و C و عوامل نطاق و Δy و Δx هي شبكة المباعدة الجديدة في التخطيط الفيزيائي. لأن $\partial g / \partial x$ و $\partial g / \partial y$ تتغير مع مرور الوقت خلال حل يعتمد على الوقت من مجال التدفق، فمن الواضح ان Δx و Δy تتغير مع الوقت، أي تتحرك نقاط الشبكة في الحيز الفيزيائي بشكل واضح، في مناطق تدفق حيث $\partial g / \partial x$ و $\partial g / \partial y$ كبيرة. Eqs. (6.37) and (6.38) تسفر عن قيمة صغيرة من Δx و Δy ل $\Delta\xi$ و $\Delta\eta$ معين. هذه هي آلية مجموعات نقاط الشبكة. هذه

هس آلية التعامل مع شبكة التكيف، وتتكون من التخطيط الحاسوبي من نقاط ثابتة في البعد (ξ, η) ؛ حيث يتم إصلاح هذه النقاط في الوقت المناسب، أي أنها لا تتحرك في البعد الحاسوبي. وعلاوة على ذلك، $\Delta \xi$ موحدة، و $\Delta \eta$ موحدة. وبالتالي، فإن التخطيط الحاسوبي هو نفسه كما ناقشنا في الأقسام السابقة.

تحل المعادلات التي تحكم التدفق في التخطيط الحاسوبي، حيث يتم تحويل المشتقات X, Y و t وفقاً لـ (6.2), (6.3) Eqs. (6.5) and (6.5). على وجه دراسة التحول الذي قدمته المعادلة (6.5) لمشتقات الوقت. في حالة شبكات الضغوط أو الحدود المجهزة كما نوقش في (Sects. 6.4 and 6.5) على التوالي، وكانت مقاييس $\partial \xi / \partial t$ و $\partial \eta / \partial t$ صفر، والمعادلة (6.5) تنتج $\partial / \partial t = \partial / \partial \tau$. لشبكة تكيفية،

$$\frac{\partial \xi}{\partial t} \equiv \left(\frac{\partial \xi}{\partial t} \right)_{x,y}$$

and

$$\frac{\partial \eta}{\partial t} \equiv \left(\frac{\partial \eta}{\partial t} \right)_{x,y}$$

هذا محدود. لماذا؟ لأنه، على الرغم من أن نقاط الشبكة ثابتة في التخطيط الحاسوبي، إلا أنها تتحرك مع مرور الوقت في التخطيط الفيزيائي. المعنى الفيزيائي لـ x, y $(\partial \xi / \partial t)$ هو معدل تغير ξ مع الوقت في الموقع ثابت في التخطيط الفيزيائي. وبالمثل، فإن المعنى الفيزيائي لـ $(\partial \eta / \partial t)_{x,y}$ هو معدل تغير η مع الوقت في الموقع ثابت في التخطيط الفيزيائي. تخيل أن عينيك تنظر نقطة ثابتة (x, y) في التخطيط الفيزيائي. بوصفها دالة من الزمن، قيمة ξ و η المرتبطة بثوابت نقطة (x, y) سوف تتغير. هذا هو السبب أن $\partial \xi / \partial t$ و $\partial \eta / \partial t$ محدودة. و بالتالي، عند التعامل مع معادلات التدفق التي تحولت في التخطيط الحاسوبي، جميع المصطلحات الثلاثة على الجانب الأيمن من المعادلة (6.5) تكون محدودة، ويجب تضمينها في المعادلات التحويلية. في هذا الشكل، مقاييس الوقت $\partial \xi / \partial t$ و $\partial \eta / \partial t$ تأخذ تلقائياً بعين الاعتبار حركة شبكة التكيف خلال حل المعادلات التي تحكم التدفق.

قيمة مقاييس الوقت في الشكل المبين في المعادلة (6.5) صعبة التقييم. من ناحية أخرى، فإن مقاييس الوقت ذات الصلة:

$$\left(\frac{\partial x}{\partial t} \right)_{\xi,\eta} \quad \text{and} \quad \left(\frac{\partial y}{\partial t} \right)_{\xi,\eta}$$

هي أسهل بكثير للتقييم، لأنها تأتي من

$$\left(\frac{\partial x}{\partial t}\right)_{\xi,\eta} \approx \frac{\Delta x}{\Delta t} \quad (6.39)$$

and

$$\left(\frac{\partial y}{\partial t}\right)_{\xi,\eta} \approx \frac{\Delta y}{\Delta t} \quad (6.40)$$

حيث يتم الحصول على Δx و Δy مباشرة من صيغ التحويل الواردة في Eqs. (6.37) and (6.38) على التوالي. دعونا نعتبر على العلاقة بين هاتين المجموعتين من مقاييس الزمن. انظر التالي

$$x = x(\xi, \eta, \tau)$$

Hence

$$dx = \left(\frac{\partial x}{\partial \xi}\right)_{\eta,\tau} d\xi + \left(\frac{\partial x}{\partial \eta}\right)_{\xi,\tau} d\eta + \left(\frac{\partial x}{\partial \tau}\right)_{\xi,\eta} d\tau$$

From this result, we write

$$\left(\frac{\partial x}{\partial t}\right)_{x,y}^0 = \left(\frac{\partial x}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial t}\right)_{x,y} + \left(\frac{\partial x}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial t}\right)_{x,y} + \left(\frac{\partial x}{\partial \tau}\right)_{\xi,\eta} \left(\frac{\partial \tau}{\partial t}\right)_{x,y}^1$$

or

$$-\left(\frac{\partial x}{\partial \tau}\right)_{\xi,\eta} = \left(\frac{\partial x}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial t}\right)_{x,y} + \left(\frac{\partial x}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial t}\right)_{x,y} \quad (6.41)$$

ملاحظة نحن نضع السفلية على المشتقات الجزئية لتجنب أي التباس حول اي المتغيرات تبقى ثابتة. الآن نرى:

$$y = y(\xi, \eta, \tau)$$

Hence:

$$dy = \left(\frac{\partial y}{\partial \xi}\right)_{\eta,\tau} d\xi + \left(\frac{\partial y}{\partial \eta}\right)_{\xi,\tau} d\eta + \left(\frac{\partial y}{\partial \tau}\right)_{\xi,\eta} d\tau$$

Thus, from this result we write

$$\left(\frac{\partial y}{\partial t}\right)_{x,y}^0 = \left(\frac{\partial y}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial t}\right)_{x,y} + \left(\frac{\partial y}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial t}\right)_{x,y} + \left(\frac{\partial y}{\partial \tau}\right)_{\xi,\eta} \left(\frac{\partial \tau}{\partial t}\right)_{x,y}^1$$

or

$$-\left(\frac{\partial y}{\partial \tau}\right)_{\xi,\eta} = \left(\frac{\partial y}{\partial \xi}\right)_{\eta,\tau} \left(\frac{\partial \xi}{\partial t}\right)_{x,y} + \left(\frac{\partial y}{\partial \eta}\right)_{\xi,\tau} \left(\frac{\partial \eta}{\partial t}\right)_{x,y} \quad (6.42)$$

Solve Eqs. (6.41) and (6.42) for $\left(\frac{\partial \xi}{\partial t}\right)_{x,y}$

$$\left(\frac{\partial \xi}{\partial t}\right)_{x,y} = \frac{\begin{vmatrix} -\left(\frac{\partial x}{\partial \tau}\right)_{\xi,\eta} & \left(\frac{\partial x}{\partial \eta}\right)_{\xi,\tau} \\ -\left(\frac{\partial y}{\partial \tau}\right)_{\xi,\eta} & \left(\frac{\partial y}{\partial \eta}\right)_{\xi,\tau} \end{vmatrix}}{\begin{vmatrix} \left(\frac{\partial x}{\partial \xi}\right)_{\eta,\tau} & \left(\frac{\partial x}{\partial \eta}\right)_{\xi,\tau} \\ \left(\frac{\partial y}{\partial \xi}\right)_{\eta,\tau} & \left(\frac{\partial y}{\partial \eta}\right)_{\xi,\tau} \end{vmatrix}}$$

وإذا سلمنا بأن $t = \tau$ ، وأن القاسم المشترك هو مصفوفه جاكوبي J Jacobian ، تصبح المعادلة أعلاه (باسقاط السفلية) كالتالي:

$$\frac{\partial \xi}{\partial t} = \frac{1}{J} \left[-\left(\frac{\partial x}{\partial t}\right) \left(\frac{\partial y}{\partial \eta}\right) + \left(\frac{\partial y}{\partial t}\right) \left(\frac{\partial x}{\partial \eta}\right) \right] \quad (6.43)$$

إذا حلينا Eqs. (6.41) and (6.42) ل $\left(\frac{\partial \eta}{\partial t}\right)_{x,y}$ نجده بالطريقة الأمثل:

$$\frac{\partial \eta}{\partial t} = \frac{1}{J} \left[\left(\frac{\partial x}{\partial t}\right) \left(\frac{\partial y}{\partial \xi}\right) - \left(\frac{\partial y}{\partial t}\right) \left(\frac{\partial x}{\partial \xi}\right) \right] \quad (6.44)$$

دعونا نستجمع الأفكار. للحصول على شبكة التكيف، للمعادلات التي تحكم التدفق، سنحول الحل للتخطيط الحاسوبي (ξ, η) ، حيث من الواجب توافر كل الشروط لتحويل الوقت التي قدمتها المعادلة (6.5). بالنسبة لمقاييس الوقت، $\partial \xi / \partial t$ و $\partial \eta / \partial t$ ، في المعادلة (6.5) فإنه يمكن التعبير عنهم من حيث $\partial x / \partial t$ و $\partial y / \partial t$ من خلال Eqs. (6.43) and (6.44). هذه المقاييس الزمنية الجديدة يمكن بدورها أن تحسب بسهولة من خلال Eqs. (6.39) and (6.40) ، حيث يتم إعطاء Δx و Δy قبل التحول الأساسي في Eqs. (6.37) and (6.38). ونعطي مثالاً على شبكة التكيف لتدفق لزوج أسرع من الصوت يتحرك باتجاه خلفي حسب Fig. 6.10، التي أُخذت من عمل كوردا Corda [8]. التدفق هو من اليسار إلى اليمين. لاحظ أن الشبكة العنقودية تتشابك حول موجة التوسع من الزاوية العليا وحول موجة الصدمة المرتكزة حول تيار التدفق. من المثير للاهتمام أن نلاحظ أن الشبكة التكيف في حد ذاتها هي نوع من "تصور طريقة تدفق الحقل" حيث تساعد على تحديد موقع الأمواج وفروقات أخرى في التدفق.

وكملاحظة أخيرة، هناك العديد من الأساليب المختلفة لتوليد شبكات التكيف. المناقشة الواردة أعلاه هي مجرد غيض من فيض؛ تقوم على الأفكار التي قدمها دواير Dwyer وآخرون. في المرجع. [9]. لمناقشة أكثر شمولية حول شبكات التكيف، وانشاء شبكة بشكل عام، انظر المرجع. [1].

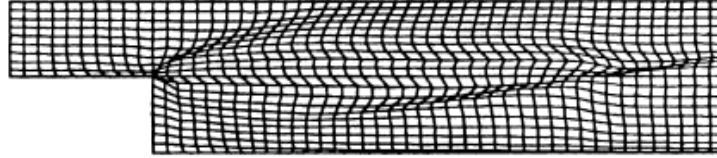


Fig. 6.10 Adapted grid for the rearward-facing step problem (from Corda, Ref. [8])

References

1. Anderson, D.A., Tannehill, John C. and Pletcher, Richard H., *Computational Fluid Mechanics and Heat Transfer*, McGraw-Hill, New York, 1984.
2. Sullins, G.A., Anderson, J.D., Jr. and Drummond, J.P., 'Numerical Investigation of Supersonic Base Flow with Parallel Injection,' AIAA Paper No. 82-1001.
3. Sullins, G.A., Numerical Investigation of Supersonic Base Flow with Tangential Injection, M.S. Thesis, Department of Aerospace Engineering, University of Maryland, 1981.
4. Holst, T.L., 'Numerical Solution of Axisymmetric Boattail Fields with Plume Simulators,' AIAA Paper No. 77-224, 1977.
5. Roberts, B.O., 'Computational Meshes for Boundary Layer Problems,' *Lecture Notes in Physics*, Springer-Verlag, New York, 1971, pp. 171-177.
6. Thompson, J.F., Thames, F.C. and Mastin, C.W., 'Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate Systems for Fields Containing Any Number of Arbitrary Two-Dimensional Bodies,' *Journal of Computational Physics*, Vol. 15, pp. 299-319, 1974.
7. Wright, Andrew F., A Numerical Investigation of Low Reynolds Number Flow Over an Airfoil, M.S. Thesis, Department of Aerospace Engineering, University of Maryland, 1982.
8. Corda, Stephen, Numerical Investigation of the Laminar, Supersonic Flow over a Rearward-Facing Step Using an Adaptive Grid Scheme, M.S. Thesis, Department of Aerospace Engineering, University of Maryland, 1982.
9. Dwyer, H.A., Kee, R.J. and Sanders, B.R., 'An Adaptive Grid Method for Problems in Fluid Mechanics and Heat Transfer,' AIAA Paper No. 79-1464, 1979.

23 : بعض التطبيقات المحددة للسريان (Explicit Finite Difference Methods) طرق الفروق المحدودة الواضحة اللزجي واللازجي⁶

23.1 مدخل (Introduction)

في هذا الفصل نحن سنقوم بجولة شاملة حول ديناميكيات الموائع الحاسوبية (computational fluid dynamics) من خلال مناقشة بعض التطبيقات (applications) من طرق الفرق المحدودة الواضحة (explicit finite difference methods) لأمثلة مختارة لسريان (flows) غير لزجي (inviscid) ولزجي (viscous). هذه الأمثلة مأخوذة من النتائج التي حصل عليها J.D. Anderson, Jr. و طلابه. المقصود هو التوضيح ما يمكن القيام به من قبل الطلاب نوعاً ما مبتدئين غير متمكنين جيداً من أفكار لديناميكيات الموائع الحاسوبية (CFD).

وعلاوة على ذلك، في جميع الحالات يتم القيام بالتطبيقات (applications) مع برامج كمبيوتر (computer programs) مصممة تماماً ومكتوبة من قبل كل طالب. هذا وتتابع الفكرة التعليمية أن كل طالب يجب أن يكون لديه تجربة بدء من ورقة وقلم، بكتابة المعادلات الأساسية (governing equations). وضع الحل العددي (numerical solution) المناسب لهذه المعادلات، وكتابة برنامج C (C program)، ووضع البرنامج في الكمبيوتر، ومن ثم المرور بجميع التجارب والمحن لجعل البرنامج يعمل بشكل صحيح. هذا هو جانب هام من تعليم ديناميكيات الموائع الحاسوبية (CFD).

قبل أن نناقش بعض الأمثلة عن ذلك، من المهم أن نصف آلية (mechanism) حسابات الفرق المحدود الصريح (explicit finite-difference calculations)، ثم التمييز بين النهج الصريح (explicit) والضمني (implicit) في القسم 5.3، التي ينبغي أن يعاد النظر فيها قبل التقدم أكثر في هذا الفصل. في المقاطع القليلة المقبلة، سوف نقوم بوصف الطرق المعلنة (explicit methods) المبسطة و الواضحة نوعاً ما. أما بالنسبة للطرق الضمنية (implicit methods) فلن تتم مناقشتها هنا.

⁶ معظم هذه الفقرة من

[Wendt 2009], Ch. 7 (Author: Anderson jr.)

وأخيراً، فإن الأمثلة التي تمت مناقشتها في هذا الفصل تتضمن كل طريقة تعتمد على الوقت، أي السير قدماً في خطوات من الزمن (forward marching in steps of time). الغالبية العظمى من الحلول التي تعتمد على الزمن (time dependent solutions) يكون هدفها حل حقل السريان الثابت الحالة (steady-state flow field) والتي تقترب من الحل عندما يكون الوقت كبيراً، وهنا، فإن الوقت هو مجرد وسيلة لتحقيق هذه الغاية. في تطبيقات (applications) أخرى، يتم استخدام الطريقة التي تعتمد على الزمن لحساب العوابر الحالية (actual transients) في سريان متقلب (unsteady flow).

وهناك أمثلة من الاثنين قد اعطيا هنا. نلاحظ، مع ذلك، أنه على الرغم من أن المقاطع التالية تعالج السير إلى الأمام (marching forward) بالنسبة للوقت (time)، يتم تطبيق نفس التقنيات (techniques) بسهولة لحساب السريان الثابت (steady flow) حيث يتم السير المكاني (spatial marching) على طول بعض محاور التنسيق (coordinate axis). لقد رأينا في الفصل 4 أن السير إلى الأمام (forward marching) من هذا القبيل (في الزمان أو المكان) هو مناسب عندما تكون المعادلات الأساسية (governing equations) قطعية (hyperbolic) أو قطعية مكافئة (parabolic).

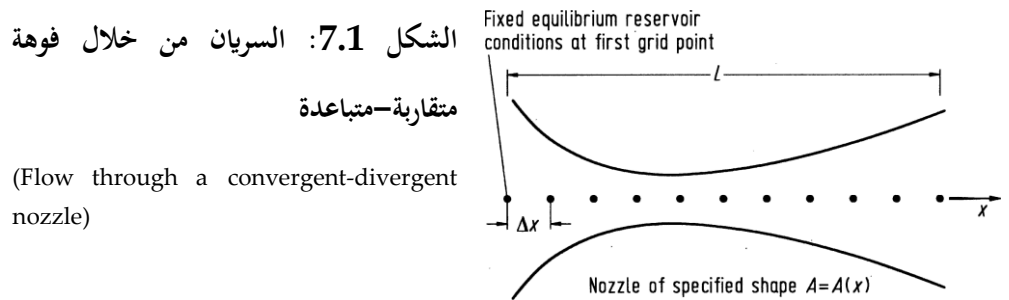
23.2 طريقة لاكس واندروف (The Lax- Wendroff Method)

دعونا نصف هذه الطريقة من خلال النظر إلى مشكلة بسيطة لديناميك الغاز (gas-dynamic problem)، وهي مشكلة سريان دون سرعة الصوت - الأسرع من الصوت من الصوت من خلال فوهة متقاربة- متباعدة (subsonic-supersonic isentropic flow through a convergent-divergent nozzle)، كما هو واضح في الشكل 7.1. هنا، من فوهة توزيع منطقة محددة، $A=A(x)$ ، يكون مُعطى، و تكون ظروف الخزان (reservoir conditions) معروفة. دعونا ننظر إلى حل شبه أحادي البعد (quasi-one-dimensional)، حيث متغيرات (variables) مجال السريان (flow field) مرتبطة (functions) ب x (في حالة ثابتة (steady state)). للحصول على غاز (gas) مثالي بالنسبة للوحدات الحرارية (calorically)، والحل لهذا السريان (flow) هو كلاسيكي (classical)، ويمكن العثور عليه في أي نص كتاب جريان قابل للانضغاط (compressible flow) (انظر على سبيل المثال المرجع [1، 2]). نستخدم هذا المثال هنا فقط لأنه

وسيلة ممتازة لتعريف ووصف فلسفة الاختلاف المحدودة المعتمدة على وقت (time-dependent finite-difference philosophy).

تنقسم الفوهة (nozzle) إلى عدد من نقاط الشبكة (grid points) في اتجاه x ، كما هو مبين في الشكل. 7.1، والتباعد (spacing) بين نقاط الشبكة المتجاورة هو Δx . لنفترض الآن قيم (values) متغيرات (variables) مجال السريان (flow field) في جميع نقاط الشبكة، والنظر في هذا السريان (flow) بصورة عشوائية (arbitrarily) بل يفترض كشرط (condition) أولي في الزمن $t = 0$ بشكل عام، فإن هذه القيم لا يفترض أن تكون على وجه الدقة حالة استقرار (steady-state) للنتائج (steady-state results)، بل على وجه الدقة حالة استقرار النتائج (steady-state results) هي ما نسعى لحسابها.

لنعتبر نقاط الشبكة (grid point)، ونعتبر النقطة i . و نترك g_i دلالة على متغير مجال السريان (flow field variable) عند هذه النقطة (g_i) قد تكون الضغط (pressure)، الكثافة (density)، السرعة (velocity)، وغيرها). هذا المتغير g_i سوف يكون دالة الزمن (function of time)، ومع ذلك، ونحن نعلم g_i في الوقت $t = 0$ ، أي أننا نعرف $g_i(0)$ لأننا نفترض القيم لجميع متغيرات مجال السريان (flow field variables) في جميع النقاط في الوقت الأولي (the initial time) $t = 0$.



نحن نحسب الآن قيمة جديدة من g_i في وقت $t + \Delta t$ ، وانطلاقاً من الشروط الأولية (initial conditions)، في المرات الأولى الجديدة $t + \Delta t = 0 + \Delta t$. هنا، Δt هو زيادة صغيرة في الوقت لمناقشتها في وقت لاحق. يتم الحصول

على قيمة جديدة (new value) من g_i ، أي $g_i(t + \Delta t)$ ، من توسيع سلسلة تايلور (Taylor's series expansion) في الوقت مع مرور الوقت كما:

$$g_i(t + \Delta t) = g_i(t) + \left(\frac{\partial g}{\partial t}\right)_i \Delta t + \left(\frac{\partial^2 g}{\partial t^2}\right)_i \frac{(\Delta t)^2}{2} + \dots$$

أو، باستخدام ترميز موحد بالنسبة للوقت باعتبارها مرتفع

$$g_i^{t+\Delta t} = g_i^t + \left(\frac{\partial g^t}{\partial t}\right)_i \Delta t + \left(\frac{\partial^2 g^t}{\partial t^2}\right)_i \frac{(\Delta t)^2}{2} + \dots \quad (7.1)$$

هنا $g_i[t+\Delta t]$ هي قيمة g في النقطة i من الشبكة في وقت $t + \Delta t$ ؛ $(\partial g / \partial t)_i[t]$ هو الأول من جزئية g تقويمها في النقطة i من الشبكة في الزمن t ، وما إلى ذلك في المعادلة (7.1)، وتصبح g_i معروفة و Δt محددة. لذلك، يمكننا استخدام المعادلة (7.1) لحساب $g_i^{t+\Delta t}$.

لحساب $g_i^{t+\Delta t}$ إذا كان لنا أن يكون بين أرقام لمشتقات $(\partial g / \partial t)_i[t+\Delta t]$ فإنه و $(\partial^2 g / \partial t^2)_i^{t+\Delta t}$ فإنه يتم الحصول على أرقام لمشتقات بذلك من فيزياء التدفق كما وردت في المعادلات التي تحكم التدفق. (ملاحظة أن المعادلة (7.1) هي ببساطة رياضيات، والتي في حد ذاتها بالتأكيد ليست كافية لحل المشكلة) والمعادلات التي تحكم التدفق لتدفق شبه أحادي الأبعاد من خلال فوهة هي (14):

$$\text{Continuity: } \frac{\partial \rho}{\partial t} = -\frac{1}{A} \frac{\partial(\rho u A)}{\partial x} \quad (7.2)$$

$$\text{Momentum: } \frac{\partial u}{\partial t} = -\frac{1}{\rho} \left(\frac{\partial p}{\partial x} + \rho u \frac{\partial u}{\partial x} \right) \quad (7.3)$$

$$\text{Energy: } \frac{\partial e}{\partial t} = -\frac{1}{\rho} \left[p \frac{\partial u}{\partial x} + \rho u \frac{\partial(\ln A)}{\partial x} + \rho u \frac{\partial e}{\partial x} \right] \quad (7.4)$$

لاحظ ان المعادلات (7.2) و (7.3) و (7.4) المكتوبة مع مشتقات الوقت على الجانب الأيسر، والمشتقات المكانية على الجانب الأيمن. ل هذه اللحظة، دعونا نحسب الكثافة، أي $g \equiv \rho$ ، ودعونا ننظر فقط للمعادلة الاستمرارية، المعادلة (7.2). توسيع الجانب الأيمن من المعادلة (7.2)، نحصل على

$$\frac{\partial \rho}{\partial t} = -\frac{1}{A} \rho u \frac{\partial A}{\partial x} - u \frac{\partial \rho}{\partial x} - \rho \frac{\partial u}{\partial x} \quad (7.5)$$

في وقت $t = 0$ ، نفترض المتغير هو مجال تدفق، ومن هنا يمكننا استبدال المشتقات مع وجود الاختلافات المكانية المركزية:

$$\left(\frac{\partial \rho}{\partial t}\right)_i^t = -\frac{1}{A} \rho_i^t u_i^t \left(\frac{A_{i+1} - A_{i-1}}{2\Delta x}\right) - u_i^t \left(\frac{\rho_{i+1}^t - \rho_{i-1}^t}{2\Delta x}\right) - \rho_i^t \left(\frac{u_{i+1}^t - u_{i-1}^t}{2\Delta x}\right) \quad (7.6)$$

المعادلة (7.6) تعطينا الرقم $(\partial \rho / \partial t)_i[t]$ ، والذي يتم إدراجه في المعادلة (7.1)، ولكن لإكمال المعادلة (7.1)، نحن بحاجة إلى المشتق الجزئي الثاني أيضا، وهو $(\partial^2 \rho / \partial t^2)_i[t]$. للحصول على هذا، تُفرق معادلة الاستمرارية، Eq. (7.5)، فيما يتعلق بالوقت:

$$\frac{\partial^2 \rho}{\partial t^2} = -\frac{1}{A} \left[\frac{\partial A}{\partial x} \left(\rho \frac{\partial u}{\partial t} + u \frac{\partial \rho}{\partial t} \right) \right] - u \frac{\partial^2 \rho}{\partial x \partial t} - \left(\frac{\partial \rho}{\partial x} \right) \left(\frac{\partial u}{\partial t} \right) - \rho \frac{\partial^2 u}{\partial x \partial t} - \left(\frac{\partial u}{\partial x} \right) \left(\frac{\partial \rho}{\partial t} \right) \quad (7.7)$$

أيضا، تفرق معادلة الاستمرارية، (7.5)، بالنسبة لـ x

$$\frac{\partial^2 \rho}{\partial t \partial x} = -\frac{1}{A} \left[\rho u \frac{\partial^2 A}{\partial x^2} + \left(\frac{\partial A}{\partial x} \right) \left(\rho \frac{\partial u}{\partial x} + u \frac{\partial \rho}{\partial x} \right) \right] - u \frac{\partial^2 \rho}{\partial x^2} - \left(\frac{\partial \rho}{\partial x} \right) \left(\frac{\partial u}{\partial x} \right) - \rho \frac{\partial^2 u}{\partial x^2} - \left(\frac{\partial u}{\partial x} \right) \left(\frac{\partial \rho}{\partial x} \right) \quad (7.8)$$

يعمل هذا الإجراء الآن على النحو التالي:

(1) في المعادلة (7.8)، يستعاض عن المشتقات على الجانب الأيمن مع وجود اختلافات المركزية، مثل

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{u_{i+1}^t - u_{i-1}^t}{2\Delta x} \\ \frac{\partial^2 u}{\partial x^2} &= \frac{u_{i+1}^t - 2u_i^t + u_{i-1}^t}{(\Delta x)^2} \\ \text{etc.} \end{aligned}$$

هذا يوفر الآن عدد لـ $(\partial^2 \rho / \partial t \partial x)_i[t]$ من المعادلة (7.8).

(2) تدرج هذا العدد لـ $(\partial^2 \rho / \partial t \partial x)_i[t]$ في المعادلة (7.7). كما في المعادلة (7.7)، وأرقام لـ $\partial u / \partial t$ و $\partial^2 u / \partial x \partial t$ يتم الحصول على معادلة الزخم عبر علاج المعادلة (7.3)، على نحو كان يعالج بالضبط نفس معادلة الاستمرارية أعلاه. لن نعطي تفاصيل هذه المعادلة هنا. (7.7)، لعدد $(\partial \rho / \partial t)$ متاحة بالفعل، وهما من المعادلة (7.6). والنتيجة الصافية هي أن لدينا الآن عدد لـ $(\partial^2 \rho / \partial t^2)_i[t]$ ، الذي تم الحصول عليه من المعادلة (7.7).

(3) تضاف لهذا العدد $(\partial^2 \rho / \partial t^2)_i[t]$ المعادلة (7.1) تذكر أن $\rho \equiv g$ لهذه القضية.

(4) لإدراج رقم $t_i(\partial p/\partial t)$ ، التي تم الحصول عليه من المعادلة (7.6)، في المعادلة (7.1).

(5) كل كمية على الجانب الأيمن من المعادلة (7.1) ومن المعروف الآن. هذا يسمح بحساب الكثافة $\rho(t+\Delta t)$ من المعادلة (7.1). هذا هو

في الواقع ما كنا نريده. والآن لدينا كثافة في النقطة i من الشبكة في الخطوة التالية في الوقت المناسب، $t+\Delta t$.

(6) نفذ الإجراء أعلاه عند كل نقطة في الشبكة للحصول على $\rho(t+\Delta t)$ في كل مكان في جميع أنحاء الفوهة.

(7) تنفيذ الإجراءات المذكورة أعلاه على معادلات الزخم والطاقة للحصول على $e(t+\Delta t)$ and $u(t+\Delta t)$ في كل مكان في جميع أنحاء

الفوهة. لدينا الآن مجرى السريان الكامل في وقت $(t+\Delta t)$ ، تم الحصول عليها من معرفة مجرى السريان في الزمن t . (نذكر أنه تبدأ العملية

في $t=0$ مع الظروف الأولية المفترضة.)

(8) نكرر العملية المذكورة أعلاه بالنسبة لعدد كبير من الخطوات في الزمن. في كل خطوة زمنية، فإن خصائص التدفق في جميع نقاط الشبكة

تتغير من وقت لآخر، في فترات زمنية طويلة، هذه التغييرات تصبح صغيرة جدا، ويتم التعامل معها باعتبارها حالة مستقرة. هذه الحالة

المستقرة هي النتيجة المرجوة، وهذه التقنية المعتمدة مع الزمن هي مجرد وسيلة لتحقيق هذه الغاية.

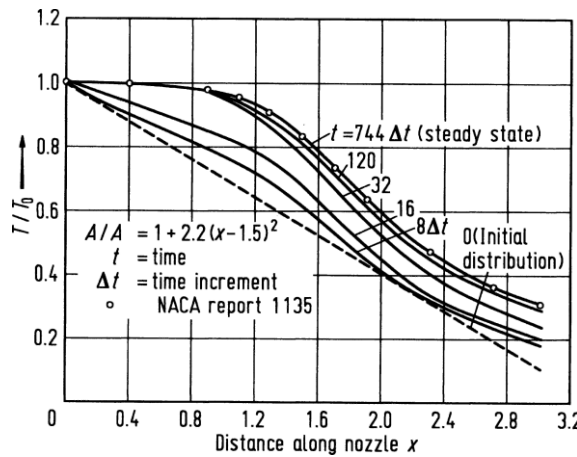
الشكل 7.2:

الحالة العابرة و الحالة النهائية المستقرة

لتوزيعات درجة الحرارة للغاز بالوحدات

الحرارية المثالية لذلك في الوقت الحالي

يتم الحصول عليها ، $\gamma = 1.4$



Transient and final steady-state temperature distributions for a calorically perfect gas obtained from the present time dependent analysis, $\gamma = 1.4$

ويتضح من سلوك هذا النوع من الحل في Figs. 7.2 and 7.3 في الشكل 7.2، الذي يظهر توزيع درجات الحرارة

لفوهة معينة. خط متقطع مسمى $t=0$ هو الذي يقترض القيم ل T في البداية في جميع أنحاء الفوهة. والمنحنى فوقه

المسمى $8\Delta t$ يمثل توزيع درجات الحرارة بعد خطوة الوقت الثامنة في أعقاب الإجراء أعلاه. المنحنيات المسماة $16\Delta t$

و $32\Delta t$ هي نتائج مماثلة بعد خطوات في الوقت 16 و 32 على التوالي. نلاحظ أن توزيع درجات الحرارة قد تغير بسرعة

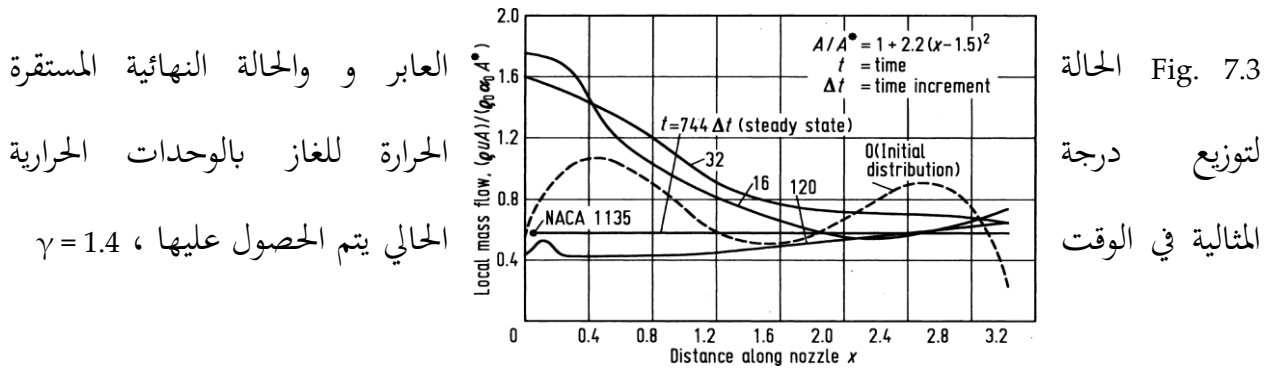
في التوزيع الأولي حيث $t=0$. في أوقات لاحقة، التغييرات تصبح أصغر؛ لاحظ أن المنحنى المسمى $120\Delta t$ لا

يختلف كثيرا عن $32\Delta t$. أخيرا، وبعد 744 خطوة من الوقت، تصبح التغييرات صغيرة جدا لدرجة أن توزيع درجات

الحرارة يصبح في حالة مستقرة. والمطلوب هذه الحالة المستقرة للحل. نلاحظ أن الحالة الثابتة التي تم الحصول عليها

عدديا، تتفق تماما مع النتائج الكلاسيكية، ويمكن الحصول عليها من المراجع [1 و 3]، ومن المرجع [4].

الشكل 7.3 يوضح الاختلاف في التدفق الشامل، م، من خلال الفوهة. الخط المتقطع هو م بما يتفق مع الظروف الأولية بافتراض $t=0$ في المنحنى المسماة $16\Delta t$ و $32\Delta t$ تثبت بوضوح الاختلافات غير المنتظمة في م في أوقات مبكرة.



ومع ذلك، بعد 120 خطوة زمنية م أصبحت أكثر استقراراً ، وبعد 744 خطوة زمنية قد وصلت إلى حالة مستقرة. هذا التوزيع للحالة المستقرة م هو على التوالي، خط أفقي، كما ينبغي أن يكون التدفق مستمر، حيث $m = \text{المستمر}$ (ثابت) من خلال الفجوة، هو القيمة الصحيحة من التدفق الشامل، بالمقارنة مع النتائج من المرجع [4]. وصف الأسلوب أعلاه، وذلك باستخدام المعادلة (7.1)، والذي هو أول ثلاثة شروط لتوسيع سلسلة تايلور Taylor's series، وحيث كل من المشتقات الجزئية الأولى والثانية في المعادلة (7.1) يتم العثور عليها من خلال الفروق المحدودة، في المشتقات المكانية في المعادلات التي تحكم التدفق مع وجود اختلافات مركزية، يتم استدعاء أسلوب Lax-Wendroff. نلاحظ أن هذه الطريقة من الدرجة الثانية من الدقة، من المعادلة (7.1). كان يعمل هذا الأسلوب بكثير من النجاح في أواخر 1960 حتى قدمت نسخة أكثر تطوراً في نفس الفكرة من قبل ماكورماك MacCormack في عام 1969. هذا هو موضوع الجزء التالي. لمزيد من المعلومات حول أسلوب Lax-Wendroff كما ينطبق على مشكلة فوهة، انظر المراجع [5,6].

MacCormack's Method 23.3

طريقة ماكورماك (MacCormack's method)، قدمت للمرة الأولى في عام 1969 (انظر المرجع [7])، كانت طريقة الفروق المحدودة الصريحة، الأكثر بساطة بالنسبة لحل تدفقات السوائل. وترتبط ارتباطاً وثيقاً بطريقة Lax-Wendroff،

ولكن هي أسهل للتطبيق. دعونا نستخدم نفس مشكلة الفوهة المناقشة في المقطع. 7.2 لتوضيح طريقة ماكورماك MacCormack's method في هذا الباب. طريقة ماكورماك، على غرار طريقة Lax-Wendroff، وتستند على توسع سلسلة تايلور Taylor's series في الوقت المناسب. ومرة أخرى، كما هو الحال في المقطع. 7.2، دعونا ننظر للكثافة عند النقطة i.

$$\rho_i^{t+\Delta t} = \rho_i^t + \left(\frac{\partial \rho}{\partial t} \right)_{\text{ave}} \Delta t \quad (7.9)$$

المعادلة (7.9) هي اقتطاع سلسلة تايلور (Taylor's series)، والذي يبدو من الدرجة الأولى دقيق.

ومع ذلك، $(\partial \rho / \partial t)_{\text{ave}}$ هو مشتق متوسط الوقت الذي يستغرقه بين الزمن t و $t + \Delta t$. يتم تقييم هذا المشتق في مثل هذه الحالة عبر حساب $\rho_i^{t+\Delta t}$ من المعادلة (7.9) التي تصبح دقيقة عند الدرجة الثانية. متوسط مشتق الوقت في المعادلة (7.9) يتم تقييمه من فكرة التنبؤ والتصحيح كما تَوابع الخطوة المتنبأة.

الخطوة المتنبأة (predictor step)

نكرر معادلة الاستمرارية، المعادلة (7.5) أدناه:

$$\frac{\partial \rho}{\partial t} = -\frac{1}{A} \rho u \frac{\partial A}{\partial x} - u \frac{\partial \rho}{\partial x} - \rho \frac{\partial u}{\partial x} \quad (7.5 \text{ repeated})$$

في المعادلة (7.5)، حساب المشتقات المكانية من القيم المعروفة لمجال التدفق في الزمن t باستخدام الاختلافات. وهذا هو، في المعادلة (7.5)،

$$\left(\frac{\partial \rho}{\partial t} \right)_i^t = -\frac{1}{A} \left[\rho_i^t u_i^t \left(\frac{A_{i+1} - A_i}{\Delta x} \right) \right] - u_i^t \left(\frac{\rho_{i+1}^t - \rho_i^t}{\Delta x} \right) - \rho_i^t \left(\frac{u_{i+1}^t - u_i^t}{\Delta x} \right) \quad (7.10)$$

الحصول على القيمة المتوقعة للكثافة، $\bar{\rho}_i^{t+\Delta t}$ ، من حيث التعبيرين الأولين من سلسلة تايلور Taylor's series، على النحو التالي

$$\bar{\rho}_i^{t+\Delta t} = \rho_i^t + \left(\frac{\partial \rho}{\partial t} \right)_i^t \Delta t \quad (7.11)$$

في المعادلة (7.11)، ρ_i^t معروفة، و $(\partial \rho / \partial t)_i^t$ هو العدد المعروف من المعادلة (7.10)؛

وبالتالي، يتم الحصول على $Q_i^{t+\Delta t}$ بسهولة. بطريقة مماثلة، في معادلات الزخم والطاقة، وتوقع قيمة المتغيرات تدفق أخرى مثل $\bar{u}_i^{t+\Delta t}$ ، $\bar{e}_i^{t+\Delta t}$ ، الخ نحصل عليها.

خطوة مصححة (corrector step)

خطوة مصححة هنا، نحن أولاً نحصل على القيمة المتوقعة لمشتقات الوقت، $(\partial Q/\partial t)_i^{t+\Delta t}$ ، عن طريق استبدال القيم المتوقعة ل $Q_i^{t+\Delta t}$ ، $u_i^{t+\Delta t}$ ، وما إلى ذلك في المعادلة 7.5.

$$\overline{\left(\frac{\partial \rho}{\partial t}\right)}_i^{t+\Delta t} = -\frac{1}{A} \bar{\rho}_i^{t+\Delta t} \bar{u}_i^{t+\Delta t} \left(\frac{A_i - A_{i-1}}{\Delta x}\right) - \bar{u}_i^{t+\Delta t} \left(\frac{\bar{\rho}_i^{t+\Delta t} - \bar{\rho}_{i-1}^{t+\Delta t}}{\Delta x}\right) - \bar{\rho}_i^{t+\Delta t} \left(\frac{\bar{u}_i^{t+\Delta t} - \bar{u}_{i-1}^{t+\Delta t}}{\Delta x}\right) \quad (7.12)$$

الآن نحسب متوسط مشتق الوقت الحسابي بين Eqs. (7.10) and (7.12)، أي

$$\left(\frac{\partial \rho}{\partial t}\right)_{\text{ave}} = \frac{1}{2} \left[\left(\frac{\partial \rho}{\partial t}\right)_i^t + \overline{\left(\frac{\partial \rho}{\partial t}\right)}_i^{t+\Delta t} \right] \quad (7.13)$$

حيث أرقام للمصطلحين على الجانب الأيمن من المعادلة (7.13) تأتي من Eqs (7.10) and (7.12) على التوالي. وأخيراً، فإننا نحصل على قيمة تصحيح ل $Q_i^{t+\Delta t}$ من المعادلة (7.9)، ونكرر التالي:

$$\rho_i^{t+\Delta t} = \rho_i^t + \left(\frac{\partial \rho}{\partial t}\right)_{\text{ave}} \Delta t \quad (7.9 \text{ repeated})$$

يتم تنفيذ نهج التنبؤ والتصحيح أعلاه بالنسبة لجميع نقاط الشبكة في جميع أنحاء الفوهة، ويطبق في الوقت نفسه على معادلات الزخم والطاقة من أجل توليد $u_i^{t+\Delta t}$ و $e_i^{t+\Delta t}$. في هذا المجال، مجال التدفق من خلال فوهة كامل في الزمن $t + \Delta t$ يتم احتسابها. ويتكرر هذا بالنسبة لعدد كبير من خطوات الوقت حتى يتم تحقيق حالة مستقرة، تماماً كما هو

الحال بالنسبة للطريقة Lax Wendroff وصفها في الطائفة 7.2.

تقنية ماكورماك MacCormack's technique كما هو مذكور أعلاه، لأنه يستخدم من خطوتين تسلسل التنبؤ والتصحيح مع وجود اختلافات الأمام على التنبؤ والخلافات المؤخرة على مصحح، هو وسيلة دقيقة من الدرجة الثانية. لذلك، فإنه لديه نفس الدقة كأسلوب Lax-Wendroff التي وصفها في المقطع 7.2. ومع ذلك، أسلوب

ماكورماك MacCormack method هو أسهل بكثير للتطبيق، لأنه ليست هناك حاجة لتقييم مشتقات الوقت الثانية كما كان الحال بالنسبة لطريقة Lax-Wendroff. لرؤية هذا بوضوح أكثر، راجع Eqs. (7.7) and (7.8)، وهي مطلوبة للأسلوب Lax-Wendroff. وتمثل هذه المعادلات عدد كبير من الحسابات الإضافية. وعلاوة على ذلك، لمشكلة ديناميكية السوائل الأكثر تعقيدا، والتفريق بين الاستمرارية، والزخم والطاقة. للحصول على المشتقات الثانية للمعادلة، أولا فيما يتعلق بالوقت، وبعد ذلك مشتقات مختلطة فيما يتعلق بالزمن والمكان، ويمكن أن تكون مملة للغاية، و يوفر مصدرا إضافيا للخطأ البشري. لا تتطلب طريقة ماكورماك MacCormack's technique في مثل هذه المشتقات الثانية، وبالتالي لا يتعامل مع معادلات مثل Eqs. (7.7) and (7.8).

وقدم بعض الملاحظات فيما يتعلق بتطبيق معين على شبه بعد واحد تدفق فوهة كما هو مبين في الشكل 7.1. على حدود التدفق (نقطة الشبكة الأولى في اليسار)، قيم p, T, ρ يتم إصلاحها، بغض النظر عن الوقت، ويفترض أن تكون قيم الخزان. يتم حساب سرعة التدفق، والتي هي قيمة صغيرة جدا دون سرعة الصوت، عبر استخدام النقاط الداخلية المجاورة، أو يمكن تقييمها من معادلة الزخم بتطبيقها عند نقطة الشبكة الأولى باستخدام الاختلافات من جانب واحد. على حدود التدفق (نقطة الشبكة الماضية في الحق في الشكل 7.1)، ويتم الحصول على جميع المتغيرات التابعة من استقراء خطية من النقاط الداخلية المجاورة، أو من خلال تطبيق المعادلات التي تحكم في هذه المرحلة، وذلك باستخدام الاختلافات من جانب واحد.

وأخيرا، نلاحظ أن النتائج التي تم الحصول عليها من طريقة Lax-Wendroff ومن أسلوب ماكورماك MacCormack method متطابقة تقريبا. على سبيل المثال، تتم مقارنة هاتين الطريقتين للاسترخاء الاهتزازي، ارتفاع في درجة الحرارة، وعدم توازن تدفق الفوهة في المرجع [8]. لا يوجد فرق بين المجموعتين من النتائج.

23.4 Stability Criterion مقياس الاستقرار

دراسة المعادلة (7.1)، هو أمر حيوي لطريقة Lax-Wendroff. نلاحظ أنها تتطلب مواصفات لزيادة الوقت، Δt . دراسة (7.9) و (7.11) Eqs. (7.9) and (7.11)، والتي تعتبر حيوية لطريقة ماكورماك MacCormack method. أنها تتطلب أيضا مواصفات لزيادة الوقت، Δt . للحصول على طرق واضحة، فإن قيمة Δt لا يمكن أن تكون عشوائية، بل يجب أن تكون أقل من المسموح به بنسبة للقيمة القصوى لتحقيق الاستقرار. التطبيقات التي تعتمد

على الوقت التي تم وصفها في المقاطع 7.2 و 7.3 تتعامل مع المعادلات التي تحكم التدفق القطعي (hyperbolic) فيما يتعلق بالوقت. راجع المقطع 5.4 التعامل مع معايير الاستقرار لمثل هذه المعادلات. Δt يجب أن تنصاع ل معيار- ما يسمى المعيار كورانت-فريدريكس-لوي CFL Courant-Friedrichs-Lewy. ويتجسد هذا في المعادلة (5.47)، والتي كانت مستمدة من معادلة نموذج بسيط التي قدمتها المعادلة (5.42). هذه معادلة الموجة الخطية، حيث c هي سرعة انتشار الموجات. اذا تمت الموجة من خلال نشر الغاز التي لديها بالفعل سرعة u ، ثم ستتحرك الموجة في سرعة $(u+c)$ نسبة إلى المناطق المحيطة الثابتة. لمثل هذه الحالة، المعادلة (5.47) تصبح:

$$\Delta t = C \left(\frac{\Delta x}{u+c} \right); \quad C \leq 1 \quad (7.14)$$

حيث C هو عدد كورانت Courant number ، و c هي سرعة الصوت، $(\partial p / \partial \rho)_s = c$. Eq. (7.14) هو المعيار CFL المناسب لحلول أحادية البعد، صريحة من فوهة التدفقات التي تم مناقشتها في المقاطع 7.2 و 7.3. لمعيار CFL الذي قدمته المعادلة (7.14) يقول أن خطوة الوقت صريحة يجب أن لا تكون أكبر من الوقت اللازم لموجة الصوت لنشر شبكة من نقطة واحدة إلى أخرى. وقد تم تجربة هذا البلاغ بأن C يجب أن تكون الأقرب امكانية إلى الوحدة، ولكن اعتمادا على التطبيق الفعلي، القيمة القصوى للاستقرار المتاحة ل C في الوقت الصريح تعتمد على حسابات الفرق المحدودة يمكن أن تختلف من حوالي 0.5-1.0. نأخذ في الاعتبار أن معايير الاستقرار تتضح من Eqs. (5.47) and (7.14) تستند إلى تحليل المعادلات الخطية. من ناحية أخرى، فإن المعادلات التي تحكم تدفق السوائل العام هي خطية غير عالية. لذلك، لن نتوقع معايير CFL للتطبيق بالضبط لمثل هذه الحالات؛ بدلا من ذلك، فإنه يوفر تقدير معقول لل Δt لمشكلة غير خطية معينة، ونتيجة لذلك قيمة الرقم كورانت Courant number في المعادلة (7.14) يمكن أن ينظر إليها باعتبارها عامل متغير في التجربة قابلة للتعديل adjustable parameter للتعويض عن تلك غير التخطيطية. العودة لحظة لتطبيق تدفق فوهة مناقشتها في الطوائف 7.2 و 7.3. هنا، في أي وقت من الأوقات t ، المعادلة (7.14) يتم تقييم في كل نقطة في الشبكة في جميع أنحاء التدفق. لأن u and c

تختلف مع x ، ثم القيمة المحلية لـ Δt المرتبطة بكل نقطة الشبكة ستكون مختلفة من نقطة إلى أخرى. قيم Δt يعملون فعلا في يكس. (7.1) و (7.9) Eqs. (7.1) and (7.9) للمضي قدما في مجال تدفق من خلال الخطوة التالية في الوقت المناسب يجب أن يكون الحد الأدنى لـ Δt محسوبة على جميع نقاط الشبكة.

[بعض التطبيقات CFD قد استخدمت " طريقة خطوة الوقت المحلي"، حيث يتم استخدام القيم المحلية من Δt في كل نقطة الشبكة في (7.1) و (7.9) Eqs. (7.1) و (7.9). في هذه الحالة، فإن الاختلافات العابرة the transient variations محسوبة على العديد من خطوات الوقت لا يخفون فيزيائيا. تم تطوير نوع من مجال تدفق "مشوه الوقت"، حيث كل متغيرات التدفق الجديدة المحسوبة لخطوة لاحقة تتعلق بالوقت فعلا للقيم الإجمالية المختلفة من الزمن. هذه " طريقة خطوة الوقت المحلي" تعطي نتائج قريبة للحالة المستقرة.

هناك حاجة لأقل مجموع خطوات الوقت للحصول على حالة مستقرة. من ناحية أخرى، احتساب العابرين ليس لها أي معنى فيزيائي، وبعض خبراء CFD يتساءل علنا عن الدقة الشاملة لمثل هذا الأسلوب، حتى بالنسبة لنتائج الحالة المستقرة النهائية.]

وأخيرا، نلاحظ أن لتدفق ثنائي أو ثلاثي البعد، هو امتدادا للمعادلة. (7.14) :

$$\Delta t = \text{Min}(\Delta t_x, \Delta t_y) \quad (7.15a)$$

$$\Delta t_x = C \frac{\Delta x}{u+c} \quad (7.15b)$$

$$\Delta t_y = C \frac{\Delta y}{v+c} \quad (7.15c)$$

23.5 تطبيقات مختارة من تقنيات المعتمدة على الزمن صريح (Explicit Time-Dependent Technique)

والغرض من هذا القسم هو لتوضيح بعض التطبيقات لهذه التقنية الواضحة، نعتمد الوقت الموضح في الأقسام السابقة من هذا الفصل. هذه التطبيقات تحتوي على العديد من ميزات CFD التي نوقشت طوال هذه الملاحظات.

23.5.1 Non-equilibrium Nozzle Flows

المراجع [5,6,8] يمثل أول تطبيق لهذه التقنية المعتمدة على الزمن للذبذبات على فوهة التدفقات غير المتوازنة. تحليل بحثي للتدفق المستمر في هذه التدفقات، والذي ينطوي قدما بمسيرة من الخزان للخروج من الفوهة. هذا التفرد يعقد إلى حد كبير الحالة المستقرة للحلول العددية للتدفق. من ناحية أخرى، أول تفسير في المراجع [5,6]، والحل العددي المعتمد على الزمن تلتف مثل في هذه المشاكل في منطقة الحل، ويشكل ذلك حلا عدديا بسيطا نسبيا للتحليل لمثل هذه الذبذبات في سريان فوهة التدفقات غير المتوازن و يتطلب ذلك إدراج معادلة معدل الذبذبات، مثل:

$$\frac{\partial e_{\text{vib}}}{\partial t} = \frac{1}{\tau} [(e_{\text{vib}})_{\text{eq}} - e_{\text{vib}}] - u \frac{\partial e_{\text{vib}}}{\partial x} \quad (7.16)$$

حيث e_{vib} هي قيمة محلية غير متوازنة لطاقة الذبذبات في وحدة الكتلة الجزيئية للغاز، $(e_{\text{vib}})_{\text{eq}}$ هي القيمة المتوازن المحلي، و τ هو وقت استرخاء الذبذبات التي هي وظيفة p المحلية و T . التحليل الكيميائي لتدفقات الفوهة غير المتوازنة يتطلب إدراج الأنواع لمعادلات الاستمرارية - واحد لكل الأنواع الكيميائية الموجودة في الغاز - والتي هي من النموذج:

$$\frac{\partial \eta_i}{\partial t} = \dot{w}_i - u \frac{\partial \eta_i}{\partial x} \quad (7.17)$$

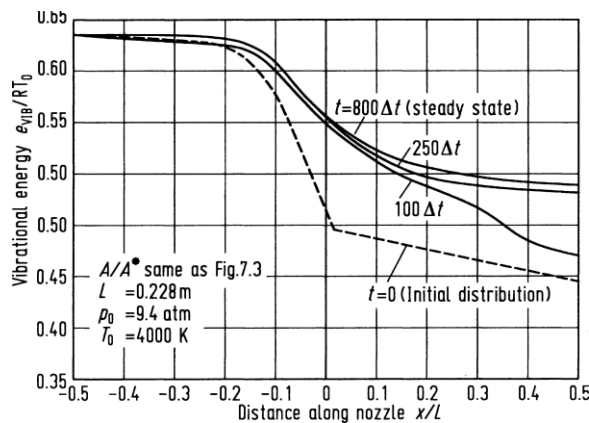
هو w_i (من خليط)، i الأنواع في وحدة كتلة moles (مولات) mole-mass ratio هي نسبة المول_الكتلي η_i حيث . (finite-rate) بسبب التفاعلات الكيميائية المحدودة الصرف i (أو تلاشي الأنواع rate of formation معدل تكوين ينطوي الثوابت الكيميائية ومعدل تركيز المحلية من الأنواع الكيميائية. لتطوير التمهيدي من المعادلات. w_i شكل (7.16) و (7.17)، انظر الفصول. 13 و 14 من المرجع. [3]. نلاحظ أنه، وعلى نفس المنوال المعادلات. (7.16) و (7.17) مكتوبة في شكل مشتق الوقت على الجانب Eqs (7.2)، (7.3) و (7.4) ويكس. في η_i و e_{vib} الأيسر، والمشتقات المكانية على الجانب الأيمن. في المقابل، يتم حساب المتغيرات غير المتوازنة من المعادلات. (7.2)، (7.3) و (7.4). في الواقع، من أجل e ، u and q الخطوات من الوقت في نفس منوال

(7.2)، (7.3)، (7.4)، (7.16) و (7.17)، Eqs. الحل المعتمد على الزمن غير المتوازن لتدفقات الفوهة يكس تحل بنفس الطريقة في كل خطوة إلى جانب الوقت كما هو موضح في الطوائف. 7.2 و 7.3. ومع ذلك، هناك قيد واحد إضافي للاستقرار الناجم عن الظواهر غير المتوازن. ل الحلول الصريحة من التدفقات غير المتوازنة، بالإضافة أيضا أقل من الوقت المخصص ل أسرع Δt التي نوقشت في الفرع 7.4، يجب أن تكون قيمة CFL إلى معيار معدل محدود يجري في النظام. وهذا هو

$$\Delta t < B\Gamma$$

هو وقت الاسترخاء الكيميائي الفعال. (وانظر الأحكام [5، 6] $\Gamma = (\partial w_i / \partial \eta_i)^{-1}$ لذبذبات عدم التوازن، $\Gamma = \tau$ حيث لمزيد من التفاصيل) لهذه المشكلة، تحول الشبكة أمر غير ضروري، والحيز الفيزيائي والحاسوبي هي واحدة في داخل نفسه.

Fig. 7.4 Transient and final distributions for the non-N2 obtained from the present



steady-state evib equilibrium expansion of time-dependent analysis

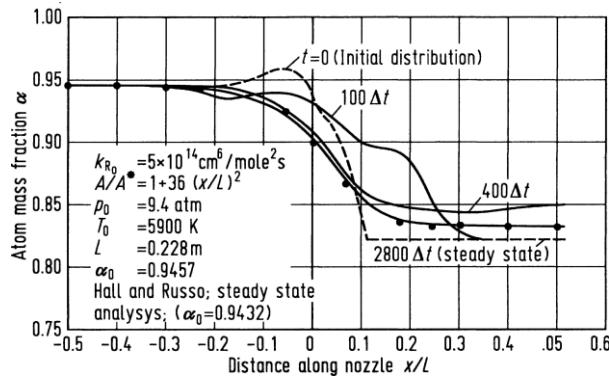
الاستقرار الأخيرة لتوزيعات evib الحصول عليها من التحليل

الشكل. 7،4 عابر و حالة للتوسع غير المتوازن ل N2 تم المعتمدة على الزمن الحاضر

في الشكل. 7.4 و LAX-Wendroff وتظهر النتائج التي تم الحصول عليها نموذجية مع تقنية تعتمد على الوقت النقي في الشكل. 7.4. هنا، يظهر N2 7.5، من المرجع. [5]. ويتضح حالة التوسع غير المتوازن للذبذبات ل بوصفها وظيفة من المسافة من خلال الفوهة. خط متقطع evib طبيعة المعتمدة على الزمن من قيمة غير متوازنة ل . توزيعات المتوسطة بعد 100 و 250 خطوة ، وتظهر، جنبا إلى جنب مع $T=0$ يمثل التوزيع الأولي المفترضة في حالة ثابتة للخطوات النهائية بعد 800 خطوة زمن. وهناك حالة مختلفة مشتقة، وهي ان من عدم التوازن الكيميائي يرد توسيع نشر الأكسجين، في الشكل. 7.5. هنا، خط متقطع يمثل الاختلاف يفترض في البداية من

. وتظهر منحنيات وسيطة بعد 100 و 400 خطوة $T=0$ جزء من كتلة الأكسجين الذري من خلال فوهة في زمنية، جنباً إلى جنب مع المباراة النهائية، تقارب ل حالة مستقرة بعد 2800 خطوة زمنية. هذا النهائي التوزيع ل [9]، والذي Hall and Russo حالة مستقرة يتفق تماماً مع حل تدفق مطرد في وقت سابق قام بها هال وروسو يظهر كالدوائر الصلبة في الشكل. 7.5

Fig. 7.5 Transient and final fraction distributions for the expansion of dissociating the present time-dependent state distribution is compared analysis of Ref. [9]



steady-state atom mass non-equilibrium oxygen obtained from method; the steady with the steady-flow

استقرار توزيعات جزء كتلة الذرة عدم متوازن النأي الأكسجين

الشكل. 7.5 حالة العابرة والأخيرة لتوسيع

لذلك تم الحصول عليها من طريقة تعتمد على الوقت الحالي؛ وبالمقارنة توزيع الحالة مطرد مع تحليل تدفق مستمر من المرجع. [9]

23.5.2 Flow Field over a Supersonic Blunt Body

ونحن نفترض تدفق غير لزج. وبالتالي يتم تمثيل المعادلات التي تحكم التدفق بواسطة المعادلة (2.65) مع F ، U ، G ، H التي قدمتها التعبيرات غير اللزجة في الطائفة. 2.9. لهذه القضية، قوات الجسم تكاد لا تذكر، وبالتالي يظهر $J=0$. التخطيط الفيزيائي في الجزء العلوي من الشكل. 7.6. BC منحنى الجسم ومنحنى AD هو موجة صدمة. يتم إعطاء الإحداثيات X من الصدمة والجسم عن طريق s and b على التوالي. ونظراً لمسافة صدمة الانفصال المحلية التي كتبها $\delta = s - b$. خلال الحل الذي يعتمد على الوقت، الجسد ثابت، وبالتالي $b = b(y)$. ومع ذلك، فإن موجة صدمة تغيير شكل ومكان مع مرور الوقت، وبالتالي $s = s(y, t)$. لذلك،

$$\delta(y, t) = s(y, t) - b(y) \quad (7.18)$$

الحيز الحاسوبي (ξ, η) هو مبين في الشكل 7.6b، ويتم الحصول عليها من التحول

$$\xi = \frac{x-b}{\delta}; \quad \eta = y; \quad \tau = t \quad (7.19)$$

حيث يتم الحصول على δ من المعادلة (7.18). لاحظ أن هذا التحول هو مثال على نظام احداثيات الحدود المجهزة كما نوقش في الطائفة 5.5 النتائج نمذجي، تم الحصول عليها من المرجع [10]، وتظهر في الشكل 7.7، 7.8 و 7.9.

تم الحصول على هذه النتائج باستخدام طريقة Lax-Wendroff. في الشكل 7.7، ويتضح من موجة الحركة التي تتغير مع الوقت، بدءاً من قيمتها المفترضة في البداية $t=0$ ، وتتقدم على شكل حالة مستقرة، بعد 500 خطوة وقت. وتظهر اختلافات الوقت لمنتصف centreline موجة السرعة و نقطة الركود الضغط stagnation point pressure في الشكل 7.8 و 7.9 على التوالي. نلاحظ في كل الأشكال الثلاثة 7.7، 7.8 و 7.9، أن أكثر التغيرات السريعة تحدث في العصور الأولى، واقترب من حالة مستقرة بدلا مقارب في بعض الأحيان.

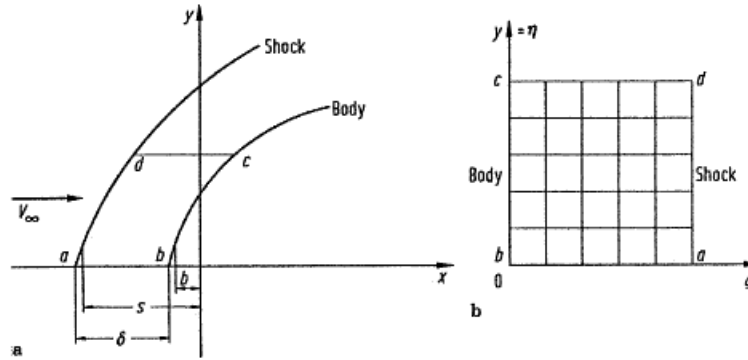


Fig. 7.6 Coordinate system for the blunt body problem

Fig. 7.7 Time-dependent shock wave motion, parabolic cylinder, $M_\infty = 4$

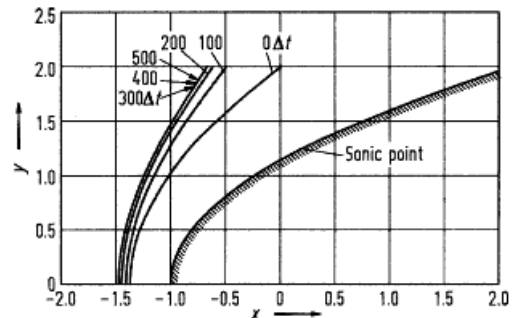


Fig. 7.8 Time variation of wave velocity; parabolic cylinder, $M_{\infty} = 4$

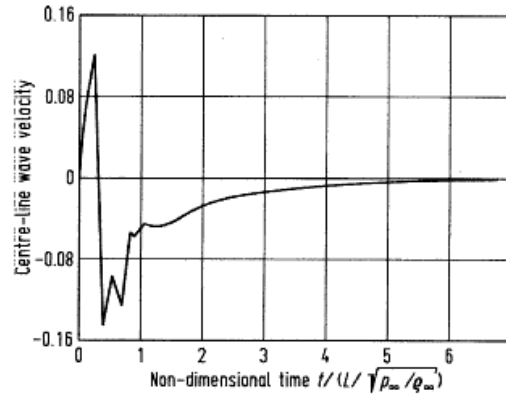
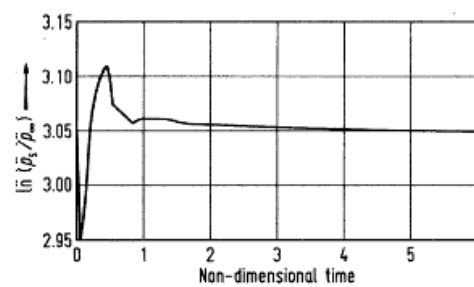


Fig. 7.9 Time variation of stagnation point pressure; parabolic cylinder, $M_{\infty} = 4$



23.5.3 Internal Combustion Engine Flows

النظر في التدفق داخل محرك الاحتراق الداخلي والتي على غرار هندسة مكبس الأسطوانة piston cylinder المبين في الشكل 7.10. المكبس يتحرك صعودا وهبوطا داخل الاسطوانة، والتدفق يدخل من خلال صمام السحب والمخارج من خلال صمام أمان exhaust valve. مجال التدفق في هذه المشكلة هو متقلب حقا، والهدف من ذلك حساب هذا التدفق غير المستقر من خلال هذه التقنية المعتمدة على الزمن. هنا، لا يتم الحصول على أي وقت مضى أي حالة مقارنة ثابتة، بل يتم احتساب تدفق دوري تكرر المجال على مدى دورة مدتها أربع أشواط كاملة من الضغط، الطاقة، و العادم. سننظر تدفق غير لزج inviscid، وبالتالي هي التي تحكم المعادلات المعادلة (2.65) و U, F, G, H ناقلات العمود من الطائفة 2،9 ويستخدم لحدود المجهزة لسريان غير لزج. نظام الإحداثيات، حيث تحول هو

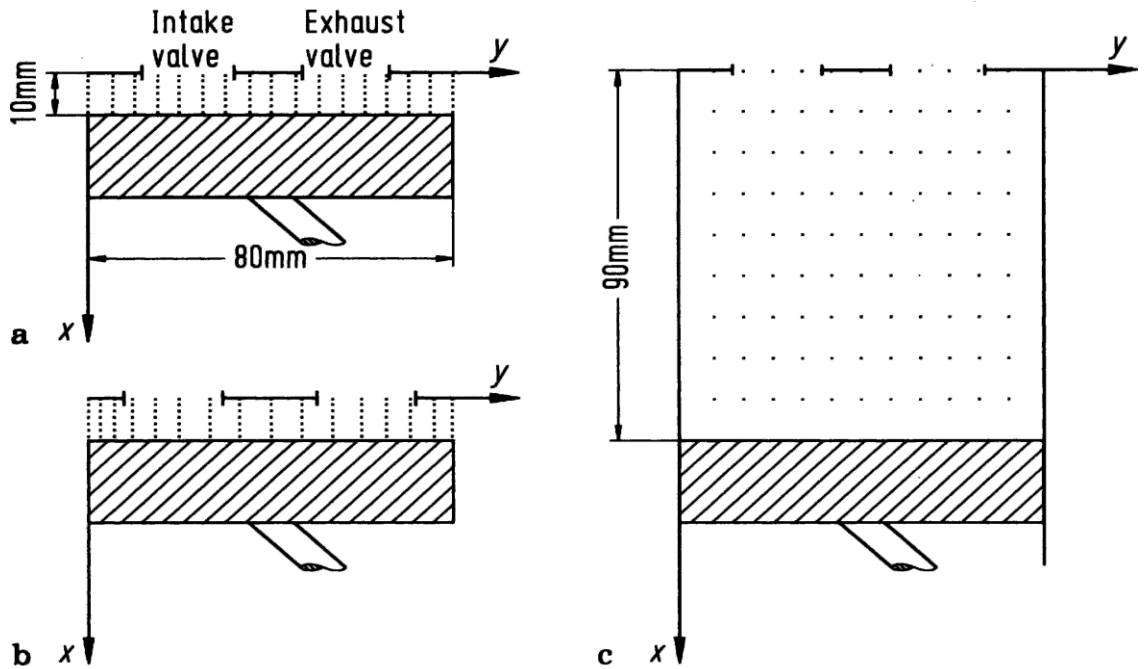
$$\xi = x/H(t); \eta = y, \tau = t$$

اسطوانة مكبس ثنائي الأبعاد نموذج متحرك يظهر ترتيب الشبكة (أ) المكبس IC الشكل 7،10 هندسة

10 × 17 TDC نقاط الشبكة متباعدة بشكل موحد؛ (ب) وضعه على المكبس 10 × 17 TDC وضعه على نقاط الشبكة (ب) وضعه على المكبس 10 × 17 TDC وضعه على نقاط الشبكة متباعدة بشكل موحد؛ (ج) وضع المكبس في (Y-نقاط الشبكة متباعدة بنسب مختلفة (فقط في اتجاه متباعدة بشكل موحد

وحيث $H(t)$ هي مسافة زمنية تتراوح بين الجزء العلوي من الاسطوانة والجزء العلوي من المكبس. نلاحظ في الشكل 7،10 تظهر أن احداثيات x على طول المحور العمودي للاسطوانة، والاحداثيات y هي في الاتجاه شعاعي عبر الاسطوانة لهذا التدفق في الشكل 7،11، 7،12، 7،13 و 7،14، مأخوذة من المرجع. [11]. ويتم الحل باستخدام تقنية ماكورماك MacCormack's technique على النحو المبين في الفرع 7.3. أرقام 7،11، 7،12، 7،13، 7،14 وإظهار حقل التدفق المرتبط بمركز امتصاص الحركة المتكررة، ثلاثة مواقع للمكبس خلال تكرار ضغط، بالقرب من مركز الطاقة، والموقع الوسيط من خروج الضربة، على التوالي. لاحظ أن يتم إنشاء دورة التدفق أثناء تناول امتصاص الضربة، وأن دورة هذا التدفق استمرت طوال دورة.

Fig. 7.10

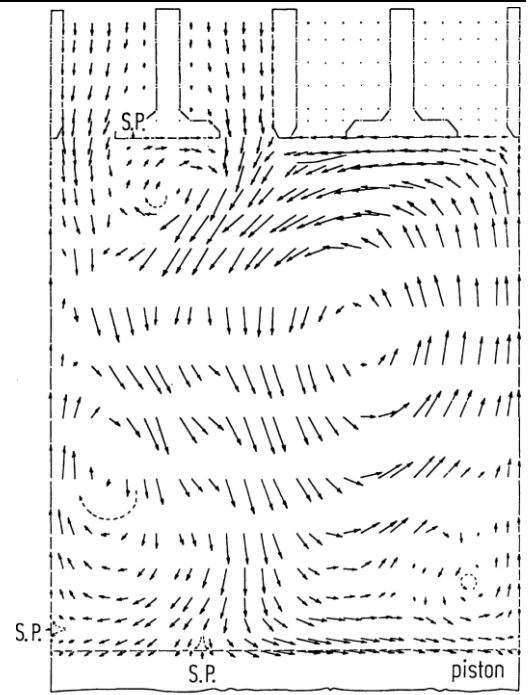


23.5.4 Supersonic Viscous Flow over a Rearward-Facing Step With Hydrogen Injection

النظر في تدفق لزج ثنائي الأبعاد الأسرع من الصوت على مدى المؤخرة التي تواجه الخطوة، حيث يتم حقن H_2 في تدفق المصب أن الخطوة رسمت في الشكل. 7،15. على عكس الأمثلة المذكورة أعلاه، تتناول هذه القضية في حل المعادلات نافير ستوكس Navier-Stokes كاملة، التي قدمتها المعادلة. (2.65) مع U ، F ، G ناقلات العمود الواردة في جوهر الفرع. 2،9 لتدفق لزج. يتم تعديل هذا النظام قليلا لوجود نشر الشامل، الذي يضيف مصطلح نشرها في معادلة الطاقة، ويضيف معادلة أخرى، وهما معادلة الاستمرارية مع الأنواع النشر. (انظر الحكام. [12]، [13] لمزيد من التفاصيل.) هذه التقنية تستخدم العددية هنا طريقة ماكورماك MacCormack's method المناقشة في الطائفة. 7.3. الحسابات الحالية الموجودة على الشبكة الموحدة في جميع أنحاء الحيز الفيزيائي. في تركيبة مع هندسة مستطيلة الموجودة بالفعل في الحيز الفيزيائي (كما يمكن أن يرى من خلال دراسة الشكل 7.15)، وهذا يعني أن لا حاجة لتحويل الشبكة.

نتائج نموذجية تم الحصول عليها من الحكام. [13، 12] وترد في الشكل. 7،16، 7،17، 7،18، 7.19 و. في الشكل. 7،16، يظهر رسم تخطيطي لناقل السرعة في حال عدم حقن H_2 . عدد ماخ Mach number الخارجي 2.19، وعدد رينولدز Reynolds number على أساس ارتفاع الخطوة 70,000. وتشمل هذه الحسابات أيضا نمودجا الاضطراب على غرار تلك ل بالدوين ووماكس Baldwin and Lomax [14]. لاحظ إعادة تدوير التدفق المفصول فقط لمصب الخطوة. الشكل 7.17 هو مخطط ناقل السرعة مع حقن H_2 .

وينظر الآن إعادة تدوير التدفقات المفصولة بين الخطوة وتدفق H_2 ، بالإضافة كما المصب من التدفق. الشكل 7،18 يظهر عدد ماخ Mach number لحدود التدفق (خطوط رقم ماخ الثابتة constant Mach number). الرقم 7.19 يوضح معالم ثابتة جزء H_2 الشامل، وهذا الرقم يعمل على تحديد مدى وشكل تدفق طائرة.



Scale: → = 0.1 V_f

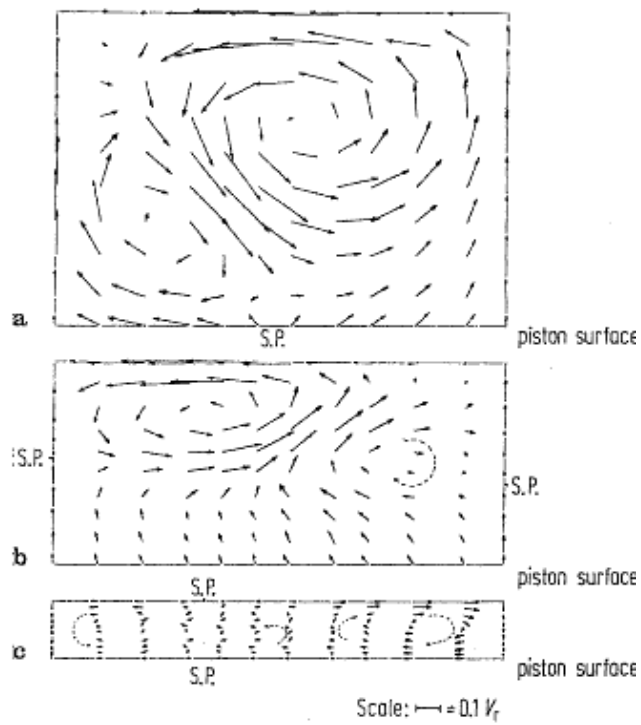


Fig. 7.12 Velocity distributions on compression stroke for the manifold-valve-engine model, 12×12 mesh.
 (a) $X_p^* = 5.63$, $CA = 261^\circ$, $t = 14.5 \text{ msec} = 3970 \Delta t$;
 (b) $X_p^* = 3.56$, $CA = 291^\circ$, $t = 16.2 \text{ msec} = 4250 \Delta t$;
 (c) $X_p^* = 1.0$, $CA = 359^\circ$, $t = 19.9 \text{ msec} = 6300 \Delta t$

Fig. 7.13 Velocity pattern near end of power stroke; $X_p^* = 8.99$, $CA = 539^\circ$, $t = 29.9 \text{ msec} = 9950 \Delta t$

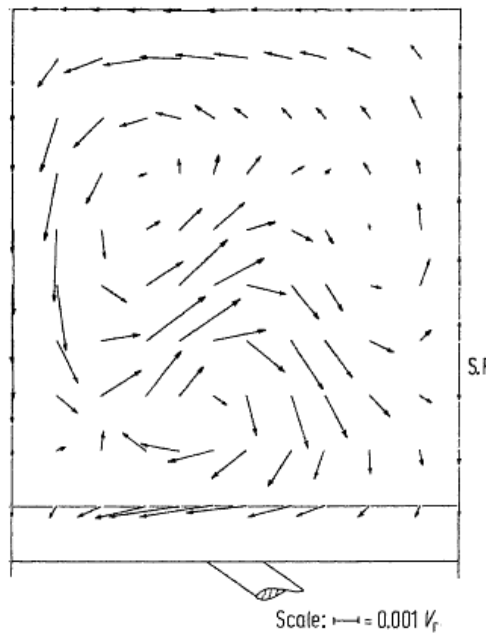


Fig. 7.14 Velocity distribution on exhaust stroke; $X_p^* = 6.99$, $CA = 600^\circ$, $t = 33.3 \text{ msec} = 11560 \Delta t$, $30 \times 22 \text{ mesh}$

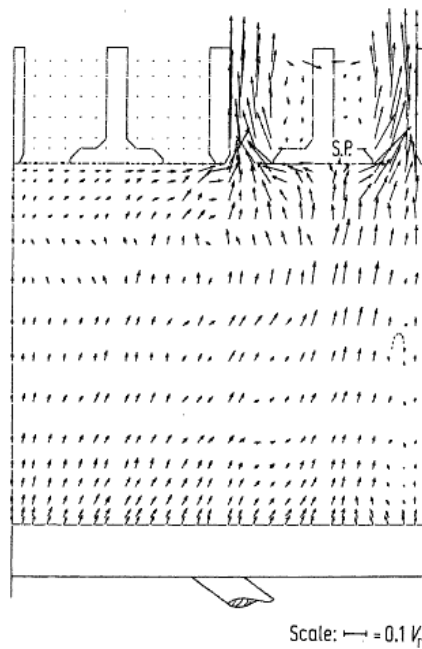
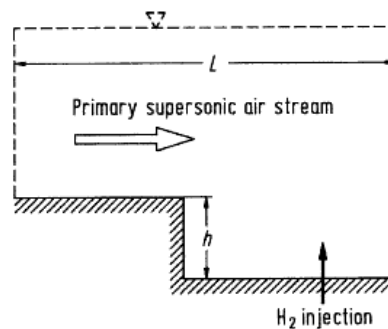


Fig. 7.15 Rearward facing step geometry



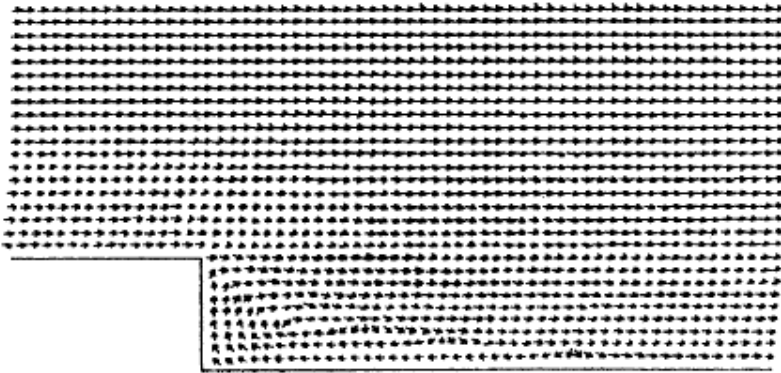


Fig. 7.16 Velocity vectors with no H₂ injection

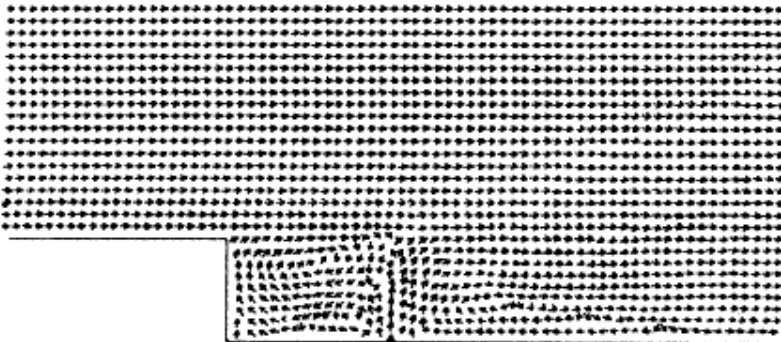


Fig. 7.17 Velocity vectors with H₂ injection

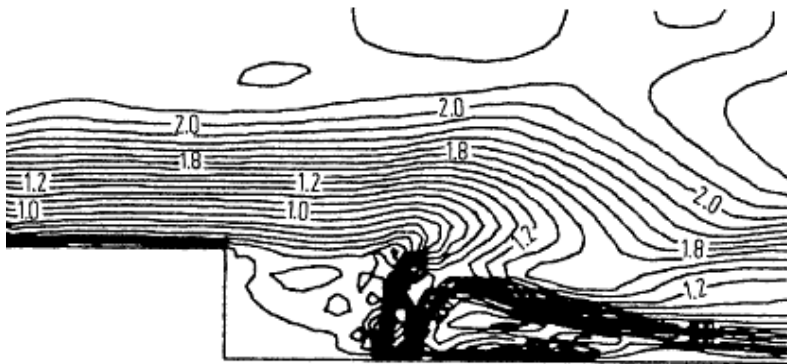


Fig. 7.18 Lines of constant Mach number with H₂ injection



Fig. 7.19 Lines of constant H₂ mass fraction

23.5.5 Supersonic Viscous Flow over a Base

بطريقة ذات صلة إلى حد ما، والنظر في تدفق الصوت لزوج لأكثر من قاعدة، كما هو موضح في الشكل 7،20. هنا، يتم استخدام اللزج كما نوقش في الفرع 7.5.4 أعلاه. ومع ذلك، لهذا الحساب يستخدم شبكة تمتد، على النحو الوارد بالتفصيل في الفرع 4. الشكل 6.4. مرة أخرى، تقنية ماكورماك MacCormack's technique هي المستخدمة. بعض نتائج العينة من الحكام [15،16] 7،21 7،22 والتي لا تتعامل مع أي حقن ثانوي في القاعدة. الشكل 7.21 يوضح الرسم التخطيطي لناقل السرعة لحالة مع عد الخارج من 2.25 وعدد رينولدز Reynolds number من 477 000 استنادا إلى ارتفاع القاعدة. لاحظ إعادة تدوير المصب تدفق فصل 7.22 يوضح معالم الضغط المستمر في التدفق، و تعتبر موجة التوسع قاب قوسين أو أدنى مع إعادة الضغط على وينظر بشكل واضح المصب من قاعدة بوضوح. الأشكال 7،23 و 7،24 تظهر نفس النوع من النتائج، باستثناء الآن لحالة حقن الهواء من وسط الق الحقن كثيرا في حقل التدفق، كما يمكن أن يرى في المقارنة مع الشكل 7،22 و 21، 7.

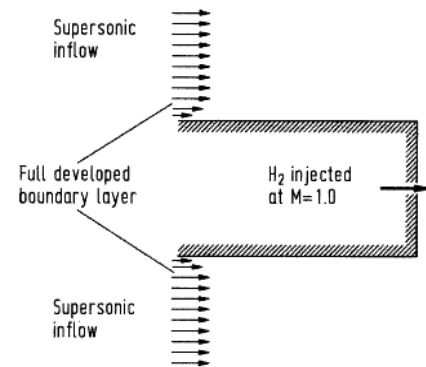


Fig. 7.20 Base flow with mass injection

Fig. 7.21 Velocity vectors with no base injection

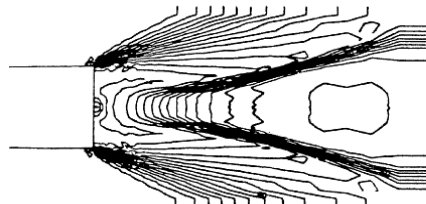
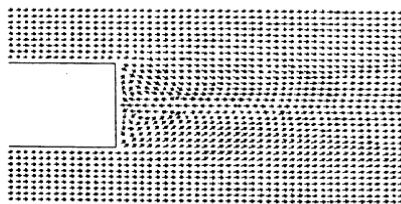


Fig. 7.22 Lines of constant pressure with no base injection

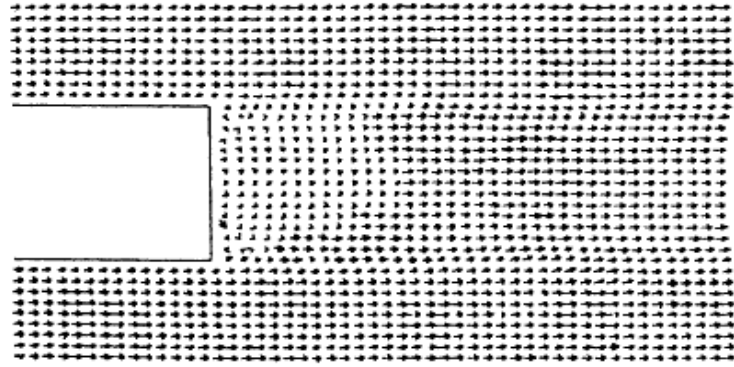


Fig. 7.23 Velocity vectors with injection from the center of the base

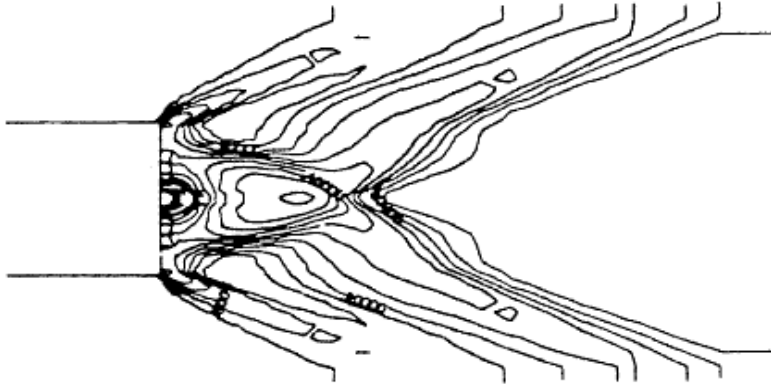


Fig. 7.24 Lines of constant pressure with injection from the center of the base

في السنوات الأخيرة، وقد تم نشر بعض النصوص الحديثة على CFD (المراجع [19-23])؛ ينصح بهذه النصوص للدراسات المتقدمة في هذا الموضوع. على وجه الخصوص، مجلدي فليتشر Fletcher (المراجع [19، 20]) تحتوي على مناقشة جيدة نظرية للموضوع. من ملاحظة خاصة هي مجلدين من قبل هيرش Hirsch (المراجع [21، 22]).، وهذه الكميات تمثل عرضاً رسمياً لأساسيات الرياضيات العددية للـ CFD، والتقنيات الحديثة المستخدمة في CFD، وكيفية استخدام هذه التقنيات في مختلف التطبيقات العملية. إشارة [23]، من خلال هوفمان Hoffmann، هو عرض هش من CFD للاستخدام من قبل المهندسين. ويوصى جميع هذه الكتب لمزيد من الدراسة المتقدمة لديناميكيات السوائل الحسابية. أيضاً، لعرض موسع للابتدائية والأفكار التمهيديّة الواردة في هذا الكتاب، فضلاً عن مناقشة مطولة للفلسفة العامة للـ CFD ودورها في مجال الهندسة الحديثة، راجع كتاب من قبل المؤلف الحالي (المراجع [24])؛ هذا هو مكتوب لدورة الجامعيين على مستوى رفيع في CFD، ويفترض على الإطلاق أي معرفة مسبقة للموضوع. يتمنى لك هذا الكاتب قراءة سعيدة، والحوسبة سعيدة في مزيد من البعثات الخاصة بك في عالم ديناميات الموائع الحسابية.

23.5.6 References

1. Anderson, John D., Jr., Fundamentals of Aerodynamics, 2nd Edition McGraw-Hill, New York, 1991.

2. Anderson, John D., Jr., 'Computational Fluid Dynamics—An Engineering Tool?' in A.A. Pouring (ed.), *Numerical Laboratory Computer Methods in Fluid Dynamics*, ASME, New York, 1976, pp. 1–12.
3. Anderson, J.D., Jr., *Modern Compressible Flow: With Historical Perspective*, 2nd Edition McGraw-Hill, New York, 1990.
4. Ames Research Staff, 'Equations, Tables, and Charts for Compressible Flow,' NACA Report 1135, 1953.
5. Anderson, J.D. Jr., 'A Time-Dependent Analysis for Quasi-One-Dimensional Nozzle Flows with Vibrational and Chemical Nonequilibrium,' NOLTR 69-52, Naval Ordnance Laboratory, White Oak, MD, 1969.
6. Anderson, J.D., Jr., 'A Time-Dependent Analysis for Vibrational and Chemical Nonequilibrium Nozzle Flows,' *AIAA Journal*, Vol. 8, No. 3, March 1970, pp. 545–550.
7. MacCormack, R.W., 'The Effect of Viscosity in Hypervelocity Impact Cratering,' *AIAA Paper No. 69-354*, 1969.
8. Anderson, J.D., Jr., 'Time-Dependent Solutions of Nonequilibrium Nozzle Flow—A Sequel,' *AIAA Journal*, Vol. 5, No. 12, Dec. 1970. pp. 2280–2282.
9. Hall, J.G. and Russo, A.L., 'Studies of Chemical Nonequilibrium in Hypersonic Nozzle Flows,' AFOSR TN 59-1090, Cornell Aeronautical Laboratory Report AD-1118-A-6, November 1969.
10. Anderson, J.D., Jr., 'On Hypersonic Blunt Body Flow Fields Obtained with a Time-Dependent Technique,' NOLTR 68-129, Naval Ordnance Laboratory, White Oak, MD, August 1968.
11. Dallospedale, C.L., 'A Numerical Solution for the Two-Dimensional Flowfield in an Internal Combustion Engine with Realistic Valve-Geometry,' M.S. Thesis, Department of Aerospace Engineering, University of Maryland, College Park, MD, 1978.

24 (الأحجام المحدودة) Finite volumes

24.1 نظرة عامة

وتعتمد طرائق الحجم المحدودة على تجزيء المعادلات إلى معادلات تكاملية

$$\frac{d}{dt} \int_{CV} \rho \phi dV + \underbrace{\int_{CS} \rho \phi (\vec{v} \cdot \vec{n}) dA}_{\text{Advective (convective) fluxes}} = \underbrace{-\int_{CS} \vec{q}_\phi \cdot \vec{n} dA}_{\text{Other transports (diffusion, etc)}} + \underbrace{\sum \int_{CV} s_\phi dV}_{\text{Sum of sources and sinks terms (reactions, etc)}}$$

في أمثلتنا اللاحقة سوف نستخدم هذه المعادلة:

$$\frac{d}{dt} \int_{V(t)} \rho \phi dV + \int_{S(t)} \rho \phi (\vec{v} \cdot \vec{n}) dA = -\int_{S(t)} \vec{q}_\phi \cdot \vec{n} dA + \int_{V(t)} s_\phi dV$$

نفترض أن الحجم لا يتغير

حيث $V(t)$ هو أي حجم منفصل، و الآن سوف

$$\frac{d\Phi}{dt} + \int_S \vec{F}_\phi \cdot \vec{n} dA = S_\phi$$

مع الوقت إذا:

$$V(t) = V$$

$$\frac{d}{dt} \int_V \rho \phi dV + \int_S \rho \phi (\vec{v} \cdot \vec{n}) dA = -\int_S \vec{q}_\phi \cdot \vec{n} dA + \int_V s_\phi dV$$

وتصبح المعادلة كالتالي:

طريقة الوقت السائر " يحتاج استخدامها إلى حساب التكامل $\Phi = \int \rho \phi dV$ للقيام بحل المعادلة

$$\frac{d}{dt} \int \rho \phi dV = \frac{d\Phi}{dt}$$

إجمالي تقديرات التدفق F_ϕ المطلوبة في حدود كل CV

$$\int_S \vec{F}_\phi \cdot \vec{n} dA = \int_S \rho \phi (\vec{v} \cdot \vec{n}) dA + \int_S \vec{q}_\phi \cdot \vec{n} dA$$

e.g. $F_\phi = \text{advection} + \text{diffusion fluxes}$

يجب أن تكون متكاملة كلياً على مدى المصدر (مجموع المصادر) على كل CV

$$S_\phi = \int S_\phi dV$$

وبالتالي يصبح:

هذه الاحتياجات تؤدي إلى العناصر الأساسية لمخطط FV، ولكن نحن بحاجة إلى ربط Φ و S_ϕ .

"طريقة الوقت-السائر" لمعادلة CV:

$$\frac{d\Phi}{dt} + \int_S \vec{F}_\phi \cdot \vec{n} dA = S_\phi$$

متوسط S_ϕ :

إجمالي / صافي التدفق من خلال CV الحدود هو مجموع التكاملات:

$$\int_S \vec{F}_\phi \cdot \vec{n} dA = \sum_k \int_{S_k} f_\phi dA$$

لحساب السطح المتكامل، هناك حاجة لمعرفة ϕ في كل مكان على السطح، ولكن ϕ لا نعرفها إلا في المركز العقدي للقيم

السطوح 1D (2D CV)

• الهدف: تقدير

• أبسط تقريب: $F_e = \phi f_\phi dA$

قاعدة نقطة المنتصف (ثاني أمر)

مركز الخلية (قيمه تقريبا تعادل

- ويقترَب F_e من الكمية المتكاملة في

قيمة المتوسط على السطح)

$$V \frac{d\bar{\Phi}}{dt} + \int_S \vec{F}_\phi \cdot \vec{n} dA = S_\phi$$

من الحصول عليها عن طريق

- بيد أن f_e غير متوفرة، فإنه لا بد

الاستيفاء

آخر أجل تقريب 2: حكم شبه منحرف

$$\underline{F_e} = \int_{S_e} f_\phi dA = \bar{f}_e S_e = f_e S_e + O(\Delta y^2) \approx \underline{f_e} S_e$$

- ويقترَب F_e على النحو التالي:

- في هذه الحالة، فمن تدفقات في زوايا f_{se} و f_{ne} التي تحتاج إلى الحصول عليها عن طريق الاستيفاء العليا تقريب

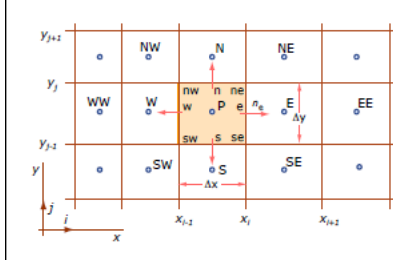
التكاملات على سطح تتطلب

- حكم سمبسون (أجل التقريب الرابع):

القيم اللازمة في 3 مواقع

- للحفاظ على دقة جزء لا يتجزأ: على سبيل المثال استخدام متعدد الحدود مكعب لتقدير هذه القيم $\bar{\Phi}_p$'s

من قريب:



الهدف: تقدير $F_e = \oint f_\phi dA$ ل 3 D CV

• أبسط تقريب: لا تزال قاعدة نقطة المنتصف (nd2 أمر)

$$S_\phi = \int_V s_\phi dV$$

- ويقترب الحديد على النحو التالي:

$$\bar{\Phi} = \frac{1}{V} \int_V \rho \phi dV$$

• النظام العالي للتقريب ممكن ولكن أكثر تعقيدا لتنفيذ ل 3D CV

• التكامل السهل يفترض الاختلاف من f_e على سطح 2D أن يكون شكل معين سهل الدمج، على سبيل المثال

2D الاستيفاء متعدد الحدود، ثم التكامل

الهدف: تقدير

• أبسط تقريب: المنتج من حجم CV مع القيمة المتوسطة من الكمية المتكاملة (يقترب من القيمة في مركز العقدة

(P)

- يقترب S_p على النحو التالي:

$$S_p = \int_V s_\phi dV = \bar{s}_p V \approx s_p V$$

• إذا تطابق S_p هو ثابت أو الخطية داخل CV

• الترتيب الثاني على خلاف ذلك

• ارتفاع التقريبية لأن:

- Φ تتطلب القيم في مواقع أخرى من P

- لأن التقريب حصل إما عن طريق تعريف القيم العقدية أو باستخدام وظائف شكل / متعددة الحدود

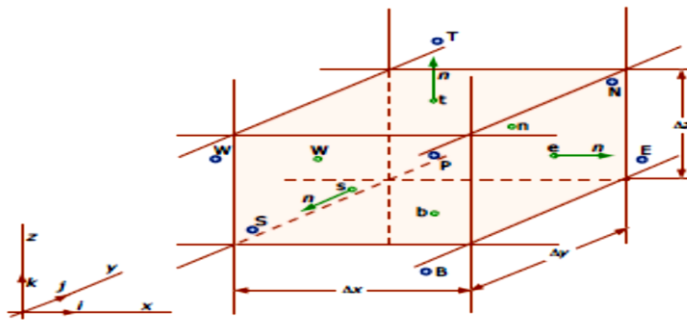
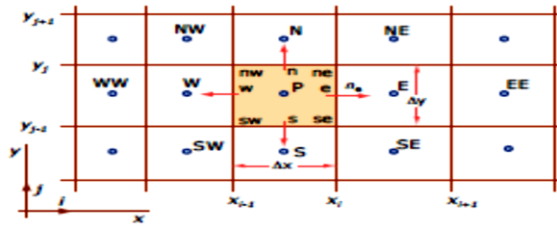
• النظر في القضية 2D (حجم لا يتجزأ من سطح يتجزأ) باستخدام وظائف الشكل

- ثنائي التريبية، وظيفة الشكل: يؤدي إلى التقريب الرابع (9 معاملات)

- 9 معاملات التي تم الحصول عليها بواسطة $s(x,y)$ المناسبة ل 9 مواقع عقدية (مركز، زوايا، المتوسط)،

- للشبكة الديكارتية Cartesian grid ، وهذا يعطي:

أربعة معاملات (الإعتماد الخطية)، لكنها لا تزال تعتمد على القيم العقدية 9



• في حالة ثنائية الأبعاد على سبيل المثال:

- للشبكة الديكارتية Cartesian grid موحدة، واحد يحصل على 2D لا يتجزأ بوصفها وظيفة من القيم العقدية 9:

القيمة الوحيدة المتاحة في العقدة P . يجب الحصول على القيم في مواقع السطح

يجب إيجاد 4 على الأقل من أجل استيفاء دقيق للتقريب المتكامل

• حالة 3D:

- تقنيات مشابهة لحالة 2D ولكن نعلم التقريب للمستوى الرابع

- للطلب العالي



Approx. of Surface/Volume Integrals: Classic symbolic formulas

• الصيغ التقريب

المتكاملة هي

أكثر تعقيدا.

• Surface Integrals $F_e = \int_{S_e} f_\phi dA$

- 2D problems (1D surface integrals)

• Midpoint rule (2nd order): $F_e = \int_{S_e} f_\phi dA = \bar{f}_e S_e = f_e S_e + O(\Delta y^2) \approx f_e S_e$

• Trapezoid rule (2nd order): $F_e = \int_{S_e} f_\phi dA \approx S_e \frac{(f_{ne} + f_{se})}{2} + O(\Delta y^2)$

• Simpson's rule (4th order): $F_e = \int_{S_e} f_\phi dA \approx S_e \frac{(f_{ne} + 4f_e + f_{se})}{6} + O(\Delta y^4)$

• الاستيفاء من

قيم العقدة هو

أكثر تعقيدا.

- 3D problems (2D surface integrals)

• Midpoint rule (2nd order): $F_e = \int_{S_e} f_\phi dA \approx S_e f_e + O(\Delta y^2, \Delta z^2)$

• Higher order more complicated to implement in 3D

• Volume Integrals: $S_\phi = \int_V s_\phi dV$, $\bar{\Phi} = \frac{1}{V} \int_V \rho \phi dV$

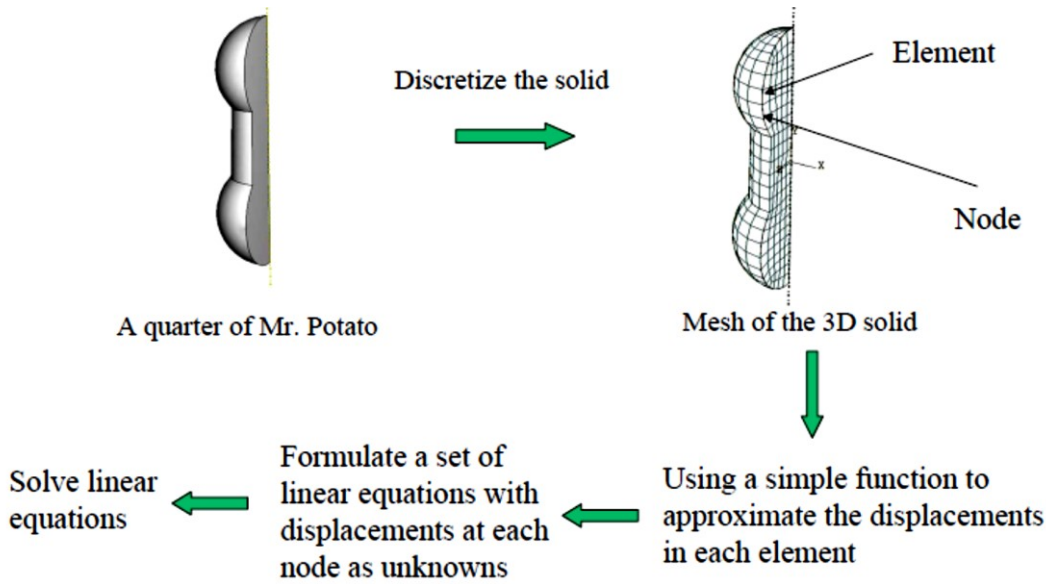
- 2D/3D problems, Midpoint rule (2nd order): $S_\phi = \int_V s_\phi dV = \bar{s}_\phi V \approx s_p V$

- 2D, bi-quadratic (4th order, Cartesian): $S_\phi = \frac{\Delta x \Delta y}{36} [16s_p + 4s_e + 4s_n + 4s_w + 4s_e + s_{se} + s_{sw} + s_{ne} + s_{nw}]$

25 العناصر المحدودة:

25.1 مدخل الى العناصر المحدودة (Finite elements)

علينا أن نحول المعادلة إلى معادلات منفصلة لتطبيق طريقة العناصر المحدودة .



ونحن نعرف المعادلة ولكن لا يمكننا حلها.



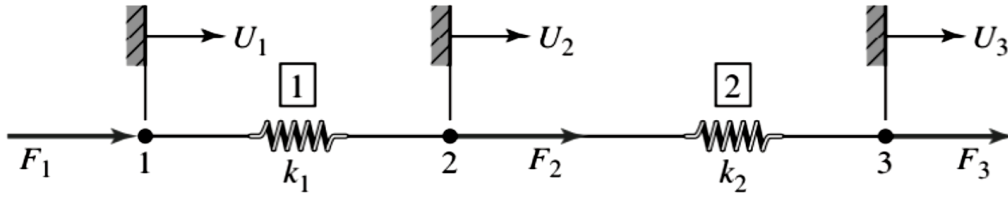
مع:

$$[\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{F}\} \Rightarrow \{\mathbf{u}\} = [\mathbf{K}]^{-1}\{\mathbf{F}\}$$

Property
Behavior
Action

	Property [K]	Behavior {u}	Action {F}
Elastic	stiffness	displacement	force
Thermal	conductivity	temperature	heat source
Fluid	viscosity	velocity	body force
Electrostatic	dielectric permittivity	electric potential	charge

يجب أن نحصل على الشكل التالي:



كتابة المعادلات لكل نابض على شكل مصفوفة:

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{Bmatrix} u_1^{(1)} \\ u_2^{(1)} \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix}$$

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_1^{(2)} \\ u_2^{(2)} \end{Bmatrix} = \begin{Bmatrix} f_2^{(2)} \\ f_3^{(2)} \end{Bmatrix}$$

للبداً بجمع معادلات التوازن التي تصف سلوك النسق المؤلف من نابضين ، تتم كتابة شروط التحرك المتوافق ، التي تبدأ بالتأثر بحركة النابض الواحد وتنتقل لتؤثر على النابض الآخر، على النحو التالي:

و بالتالي نحصل على الآتي:

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix}$$

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} f_2^{(2)} \\ f_3^{(2)} \end{Bmatrix}$$

$$u_1^{(1)} = U_1 \quad u_2^{(1)} = U_2 \quad u_1^{(2)} = U_2 \quad u_2^{(2)} = U_3$$

هنا نستخدم تدوين ف_ط (ي) لتمثيل القوة المبذولة على العنصر ي في عقدة

ط.

توسيع كل المعادلة في شكل مصفوفة:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} + f_2^{(2)} \\ f_3^{(2)} \end{Bmatrix}$$

هنا نجمع المصفوفات التي يمثل فيها عدد الأسطر، عدد النوابض في كل نسق:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ 0 \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} 0 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ f_2^{(2)} \\ f_3^{(2)} \end{Bmatrix}$$

وبعد ذلك، نشير إلى مخططات الجسم المتحررة من العقد الثلاثة:

$$f_1^{(1)} = F_1 \quad f_2^{(1)} + f_2^{(2)} = F_2 \quad f_3^{(2)} = F_3$$

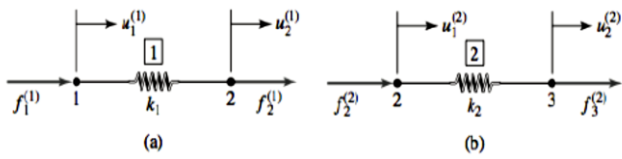
$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \quad \text{الشكل النهائي:}$$

حيث مصفوفة ثوابت الجساءة لكل من النابضين هي:

$$[K] = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix}$$

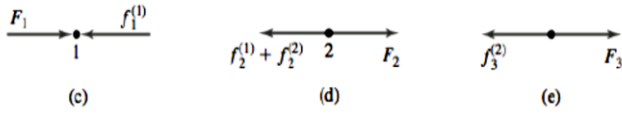
هذا هو الحل لمثال الدعامات

اثنين من العناصر متساوية الطول $L/2$ مع تحرك عقدة الربط العنقدي المرتبطة



بها. لعنصر $A1$

$$8A1 = 7A0 /$$



أما بالنسبة للعنصر 2، فلدينا:

$$k_1 = \frac{A_1 E}{L_1} = \frac{7A_0 E}{8(L/2)} = \frac{7A_0 E}{4L}$$

بما أن وسط القضيب لا يتعرض لأي نوع من الأحمال ،

$$A_1 = \frac{5A_0}{8} \quad \text{and} \quad k_2 = \frac{A_2 E}{L_2} = \frac{5A_0 E}{8(L/2)} = \frac{5A_0 E}{4L}$$

تصبح معادلات التوازن على العنصرين:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ 0 \\ P \end{Bmatrix}$$

تطبيق شرط القيد $U_1 = 0$ النتيجة في:

$$\begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ P \end{Bmatrix}$$

25.2 مدخل الي طريقة العناصر المنتهية (FEM) في ديناميكيات الموائع الحاسوبية (CFD)⁷

طريقة العناصر المنتهية (Finite element method) أو يطلق عليها أيضاً تحليل العناصر المنتهية هي طريقة تحليل عددي لإيجاد الحلول التقريبية للمعادلات التفاضلية الجزئية بالإضافة إلى الحلول التكاملية.

أول سمة أساسية هي أن المجال المتواصل ، أو الحقل ، يتم تقسيمهم إلى خلايا تسمى العناصر التي تشكل الشبكة. العناصر (في مساحة ذات بعدين) لديها الثلاثي الشكل و الرباعي الشكل ، ويمكن أن تكون مستقيمة أو منحنية. الشبكة نفسها لا يلزم أن تكون منظمة. مع شبكات غير منتظمة وخلايا منحنية ، يمكن التعامل مع هندسيات معقدة بكل سهولة.

والسمة الأساسية الثانية من طريقة العناصر المنتهية هي أن حل مشكلة منفصلة يفترض بداهة أن يكون النموذج مُعدًا . الحل يجب أن ينتمي إلى فضاء الوظيفة ، والتي بنيت من خلال تغيير القيم لوظيفة بطريقة معينة، على سبيل المثال خطيا أو تربيعيا بين القيم في نقاط عقدية.

النقاط العقدية ، أو العقد ، هي نقط نموذجية من العناصر مثل القمم ، نقاط جانب المنتصف ، نقاط منتصف العنصر ، وما إلى ذلك ونظرا لهذا الخيار يكون تمثيل الحل مرتبط بشدة مع التمثيل الهندسي داخل المجال. و السمة الأساسية الثالثة هي ان طريقة العناصر المنتهية لا تهتم بالحل في المعادلات التفاضلية الجزئية بحد ذاتها وانما تهتم بالحل بالاعتماد على المعادلات التكاملية.

السهولة في الحصول على قدر عال من الدقة و السهولة في تنفيذ شروط الحدود تشكل ميزة هامة ثانية لطريقة العناصر المنتهية. و السمة الأساسية الأخيرة لطريقة العناصر المنتهية هي الطريقة النموذجية التي يتم الحصول على التفريد منها. هذه المعادلات المفردة بُنيت من قبل مساهمات على مستوى العنصر والتي يتم تجميعها بعد ذلك.

25.3 شرح طريقة العناصر المنتهية

سوف نستخدم مثالين بسيطين لشرح طريقة العناصر المنتهية، والتي من خلالها من الممكن استخلاص الطريقة العامة. في النقاش التالي، يجب على القارئ أن يكون متفههما لمبادئ [علم الحسبان](#) و [الجبر الخطي](#).

P1 هي مسألة أحادية البعد، معطاة على الشكل التالي:

$$P1 : \begin{cases} u'' = f \text{ in } (0, 1), \\ u(0) = u(1) = 0, \end{cases}$$

حيث f معلوم و u هو تابع مجهول للمتحول x ، و u'' هو المشتق الثاني للتابع u بالنسبة للمتحول x .

المسألة ثنائية البعد البسيطة هي مسألة ديركلت (Dirichlet) وتعطى على الشكل التالي:

$$P2 : \begin{cases} u_{xx} + u_{yy} = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

حيث Ω هي منطقة مفتوحة متصلة في بالسطح الثنائي البعد (x, y) الذي تكون حدوده $\partial\Omega$ هي عبارة عن مضلع ذو شكل معين. و u_{xx} و u_{yy} هي المشتقات الثانية للمتحولين x و y على الترتيب.

من الممكن حل مسألة أحادية البعد بحساب المشتق العكسي. لكن هذه الطريقة في حل مسألة القيمة الحدية (boundary value problem) تصلح لحل المسائل أحادية البعد ولا يمكن تعميمها إلى مسائل ذات أبعاد أعلى أو مثال لها الشكل $u + u'' = f$ ولهذا السبب كان من الضروري تطوير طريقة العناصر المنتهية، بدءاً من البعد الأحادي وتعميمها على الأبعاد الأعلى.

الشرح هنا سوف يتم على مرحلتين، المرحتين الأساسيتين الواجب تطبيقهما لحل مسألة القيمة الحدية باستخدام طريقة العناصر المنتهية:

الخطوة الأولى: تبسيط مسألة القيمة الحدية (boundary value problem) إلى شكل بسيط تنتفي معه الحاجة إلى استخدام الحاسب للحل، بل يكون من الممكن حلها يدوياً باستخدام الورقة والقلم.

الخطوة الثانية: هي التقطيع، حيث يتم تجزئة الشكل إلى عناصر منتهية وحل كل عنصر على حدة.

بعد هذه الخطوة سيكون لدينا صيغة متكاملة لحل مسائل ذات درجات عالية لكن يجب أن تكون خطية وحلها ستكون تقريبية لمسألة القيمة الحدية. ومن ثم يتم برمجة هذه الطريقة على الحاسوب.

25.4 الصيغة المتحولية (variational formulation)

Variational formulation = The minimization of an energy integral over the domain.

الصيغة المتحولية هي صيغة طبيعية تكاملية لطريقة العناصر المنتهية (FEM) و لكن في ميدان الميكانيك الموائع - بشكل عام - من غير الممكن وضع الصيغة المتحولية (variational formulation).

الخطوة الأولى هي تحويل P1 و P2 إلى مكافئتها المتحولية. إذا كان u هو حل لـ P1 ، عندها من أجل أي دالة متصلة v تتحقق شروط الانتقال الحدي، مثلاً $v = 0$ عند $x = 0$ و $x = 1$ ، يكون لدينا (1)

وبشكل معاكس، من أجل قيمة معطاة لـ u فإن (1) تكون محققة من أجل أي دالة متصلة $v(x)$ وعندها من الممكن أن يبرهن أن u ستكون حلاً لـ P1 برهان هذا ليس بالأمر السهل وهو يعتمد على فضاء سوبوليف. (وباستخدام التكامل بالأجزاء على يمين المعادلة (1) سنحصل على مايلي

$$\text{حيث تم افتراض أن } v(0) = v(1) = 0.$$

برهان يظهر وجود حل وحيد

مثل هذه التوابع تكون ضعيفة. (بحيث (0,1) هو عبارة عن تابع مستمر مطلق للثنائية $H_0^1(0, 1)$ من الممكن اعتبار أن الذي جداء داخلي ومن ثم تعرف ϕ (قابلة للاشتقاق مرة واحدة) وتكشف عن الخريطة الخطية الثنائية المتناظرة الجداء هو أيضاً $\int_0^1 f(x)v(x)dx$ ومن ناحية أخرى، فإن الطرف الأيسر. فضاء هلبرت إلى $H_0^1(0, 1)$ يحول على فضاءات هلبرت يظهر أنه لمبرهنة تمثيل رايسز وتطبيق $L^2(0, 1)$ الفضاء الداخلي، ولكن هذه المرة على P1. (2) وبالتالي يحل المسألة u يوجد حل وحيد

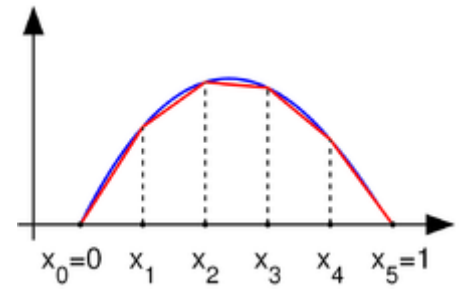
الصيغة المتحولية لـ P2

إذا تم التكامل بالأجزاء باستخدام مبرهنة غرين حيث نجد أنه إذا كان u هو حل لـ P2 ، فإنه من أجل أي v يكون

$$\int_{\Omega} f v ds = - \int_{\Omega} \nabla u \cdot \nabla v ds = -\phi(u, v),$$

حيث ∇ تحقق الترج وترمز إلى الجداء الداخلي في المستوي ثنائي البعد.

25.5 التقطيع (Discretization)



التابع H^1_0 مع القيم الصفرية عند نقاط النهاية (زرقاء)، والتقريب الخطي الجزئي للمنحنى (اللون الأحمر).

الفكرة الأساسية من طريقة العناصر المنتهية هي استبدال المسألة الخطية ذات الأبعاد اللانهائية: أوجد قيمة $u \in H^1_0$

بحيث أن صيغة بعدية منتهية:

(3) أوجدت $u \in V$ such that

$$\forall v \in V, -\phi(u, v) = \int f v$$

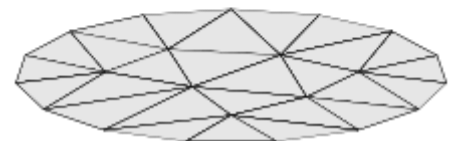
حيث V هو فضاء جزئي خطي ذو عدد أبعاد منتهية من H^1_0 . هناك العديد من الخيارات لـ V . لكن في طريقة العناصر

المنتهية نعتبر V على أنها فضاء للأجزاء الخطية للتابع.

في المسألة P1، نأخذ المقطع $(0,1)$ باختيار n قيم من $0 = x_0 < x_1 < \dots < x_n < x_{n+1} = 1$ ونعرف V على الشكل:

$$V = \{v : [0, 1] \rightarrow \mathbb{R} : v \text{ is continuous, } v|_{[x_k, x_{k+1}]} \text{ is linear for } k = 0, \dots, n, \text{ and } v(0) = v(1) = 0\}$$

$$\forall v \in H^1_0, -\phi(u, v) = \int f v$$



حيث نعرف $x_0 = 0$ و $x_{n+1} = 1$ لاحظ أن التتابع في V هي توابع غير قابلة للاشتقاق بالاعتماد على التعريف المبدئي للحسابان. إذا كان $v \in V$ فإن المشتق يكون عادة غير معرف عند أي $x = x_k, k = 1, \dots, n$. لكن يوجد مشتق عند كل قيمة للمتحول x ومن الممكن استخدام هذا المشتق لغرض التكامل بالأجزاء.

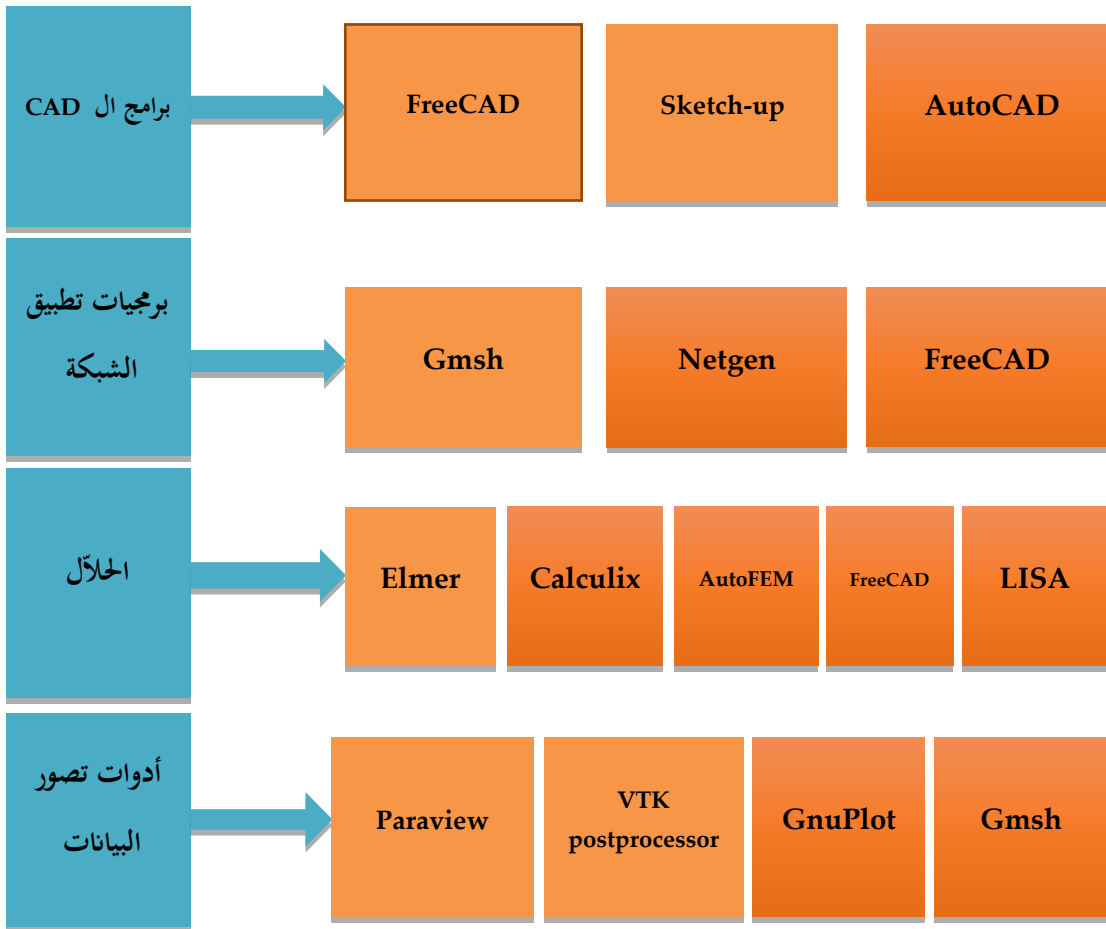
تابع خطي مقطوع في مستوي ثنائي الأبعاد.

من أجل المسألة P2 نحتاج أن تكون V عبارة عن مجموعة من التتابع من Ω في الشكل الموضح على اليسار، يظهر تثليث مضلعي لمنطقة مضلعية من 15 ضلع Ω في المستوي (في الأسفل)، والتتابع الخطي الجزأ (ملوناً، في الأعلى) لهذا المضلع الذي يكون خطياً على كل مثلث من التثليث. حيث أن الفضاء V سيحتوي على توابع تكون خطية على كل مثلث من التثليث المختار.

تظهر V مكتوبة على الشكل V_h في بعض المراجع، وذلك بسبب أنه يوجد هدف في الحصول على حلول أدق وأدق للمسألة المتقطعة (3) الذي سيكون إلى حد ما سيؤدي إلى حد المسألة الأصلية في إيجاد القيم الحدية للمسألة P2. يتم عنونة التثليث باستخدام مُعامل ذو قيمة حقيقية $h > 0$ والذي يكون ذو قيمة صغيرة. سوف يتم ربط هذا المعامل بحجم أكبر مثلث وسطي الحجم في التثليث. وعندما نزيد تجزئة التثليث فإن فضاء التقطيع الخطي V يجب أن يتغير مع h كما يوضح الترميز V_h .

26 البرمجيات المستخدمة في النمذجة والمحاكاة

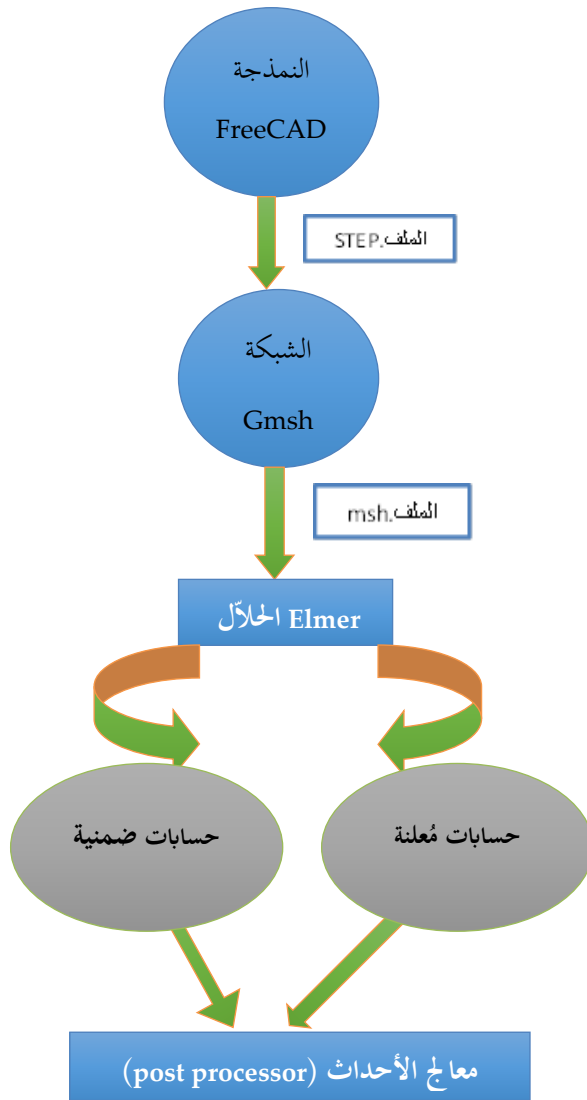
استخدمنا العديد من البرامج في هذه الدراسة ، و اعتمدنا استراتيجيات مختلفة لتحقيق هذا العمل (بشأن التصميم، و وضع الشبكة ، و الحل و التصور للنتائج) .
 في التخطيطي المبين أدناه وصف لسلسلة من الأدوات المستخدمة . تم تلوين الأدوات المعتمدة في هذه الأطروحة باللون الأخضر و اللاتي ملونة باللون الأحمر كانت معتمدة لفترة لا بأس بها من أجل التجربة ولكن في النهاية لم يتم اعتمادها إما لأنها ليست مجانية أو لأنها لاتعتبر من المصادر المفتوحة أو لأنها محدودة جدا ولا يمكن أن تدعم قيمة كبيرة من البيانات.

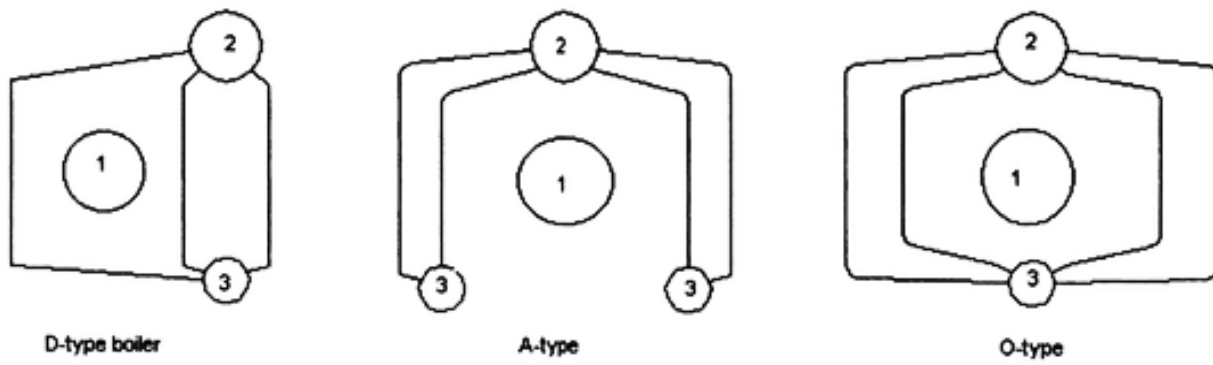


26.1 تنسيق الملفات (format of files)

واحدة من الصعوبات التي واجهتنا هي مشكلة الانتقال من برنامج إلى آخر .

عادةً يحفظ البرنامج تنسيقاً لا يستجيب له البرنامج الآخر، و هنا تظهر الحاجة لاكتشاف ما هي الصيغة المقبولة من قبل البرنامج كمدخل، و البرامج التعليمية لا تذكر هذه التفاصيل وهنا يبدأ العمل لاكتشاف الشكل المناسب .





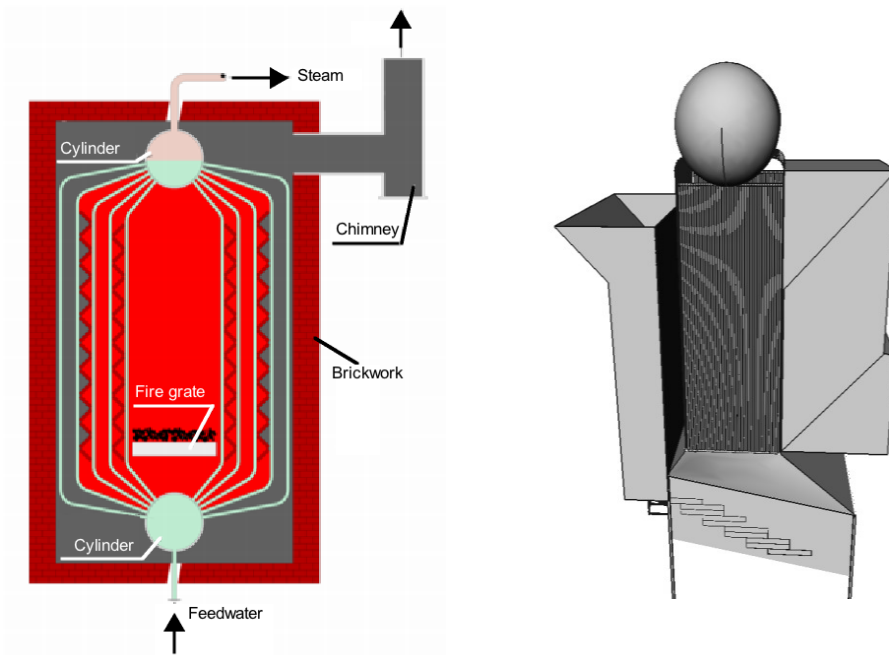
2.2.1: A-, D- and O-type boiler configurations. 1. Burner; 2. Steam drum; 3. Water drum

Figure

عدة نماذج موجودة في الدراسات، وقد اعتمدنا في هذه الدراسة الشكل الأكثر بساطة لتسهيل عملية التصنيع سيما و أن صناعة المحرقة ستكون محلية.

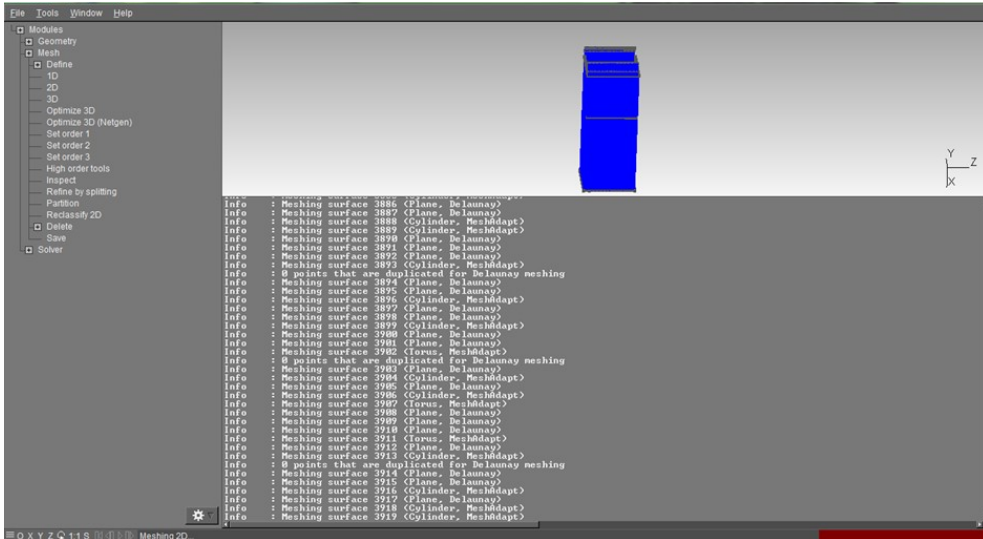
26.3 تطبيق الشبكة على النموذج

لوضع الشبكة على نموذج المحرقة المصمم عبر برنامج FreeCAD، اعتمدنا بداية البرنامج عينه أقصد FreeCAD ، ولكن تبين لنا أن هذا البرنامج غير قادر على إنجاز الشبكة على كامل النموذج وإنما على عنصر واحد فقط، هذا البرنامج لازال تحت التطوير و ربما في السنوات المقبلة يصبح قادرا على القيام بمثل هكذا مهمة.



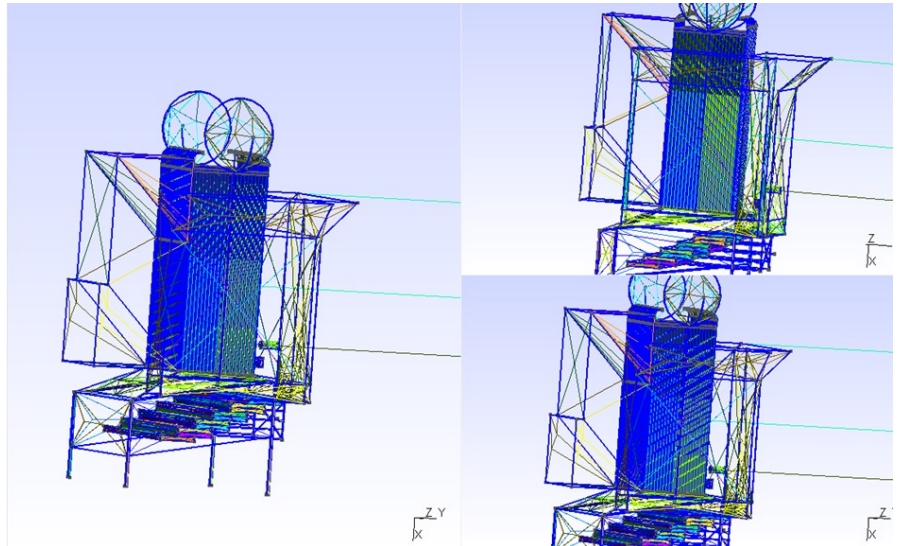
وبدأنا البحث عن برمجيات قادرة على القيام بما عجز عنه برنامج FreeCAD ، وبالفعل وجدنا العديد من البرمجيات منها Netgen و Gmsh. جميعها مجانية وتندرج تحت المصادر المفتوحة.

حاولنا كل هذه البرامج و وجدنا أن الأفضل هو Gmsh من وجهة نظر السرعة و إمكانية تحديد نوع الشكل في FEM وقدرته على وضع الشبكة على نموذج معقد في وقت قصير نسبيا مقارنة مع البرمجيات الأخرى .



يُني Gmsh حول أربع وحدات: الهندسة ، التشبيك ، الحلّالاً ومعالج الأحداث . يمكن السيطرة على كل وحدة إما بشكل تفاعلي باستخدام واجهة المستخدم الرسومية أو باستخدام لغة البرمجة . تصميم جميع الوحدات الأربع يعتمد على فلسفة بسيطة تكون سريعة وخفيفة و سهولة الاستعمال .

- السرعة : على جهاز كمبيوتر شخصي قياسي في أي لحظة معينة من الزمن ينبغي إطلاق Gmsh على الفور ، وتكون قادرة على وضع الشبكة بسرعة تصل إلى وضع مليون رباعي الأسطح في دقيقة واحدة .
- الذاكرة: يجب أن يكون أثر الذاكرة من تطبيق الحد الأدنى و يجب أن يكون رمز مصدر صغير بما فيه الكفاية بحيث مطور واحد يمكن أن يفهم ذلك. تثبيت أو تشغيل البرنامج يجب أن لا يعتمد على أي حزمة برامج طرف ثالث غير متوفرة على نطاق واسع .
- سهولة الاستعمال : تصميم واجهة المستخدم الرسومية تسمح للمستخدم الجديد بإنشاء شبكات بسيطة في غضون دقائق



Elmer 26.4/الحلّال

Elmer هو مزيج من برامج مختلفة تهدف إلى محاكاة مشاكل فيزيائية باستخدام طريقه العناصر المحددة (FEM) . ثلاثة من هذه البرامج هي: ElmerGUI ، ElmerSolver ، ElmerPost . إلر هو برنامج مفتوح المصدر ، الذي صدر تحت رخصة جنو العمومية (GPL) .

Elmer يمكن استخدامه بطريقتين مختلفتين :

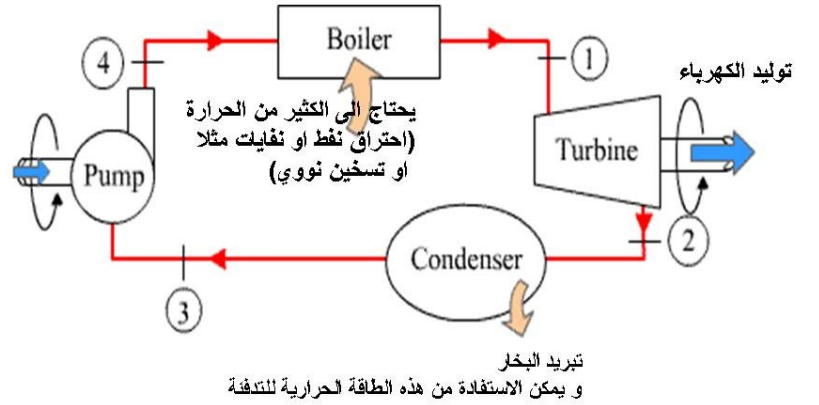
- باستخدام واجهة المستخدم الرسومية (GUI) . (يمكن إنشاء ملف نص الأمر بعد جلسة GUI) .
 - باستخدام ملف نص الأمر
- Elmer لا يملك القدرة لتوليد الهندسة و التشبيك . ولذلك، كإجراء عام، يجب أن يتم استيراد الهندسة و الشبكة إلى Elmer .
- Elmer يقبل الهندسة وشبكات مختلفة الأشكال . من بينها، فإنه يقبل شكل شبكة GMSH .
- في أطروحة الماجستير هذه واحدة من المهام الأكثر أهمية هو تحديد موقع المنطقة التي تتعرض لضغوط عالية .

27 استخدام برامج لا تحتاج الى رخصة فى ميدان ديناميكيات الموائع الحاسوبية

27.1 تحسب سريان الماء داخل محطة طاقة تعمل على البخار ببرامج جاهزة

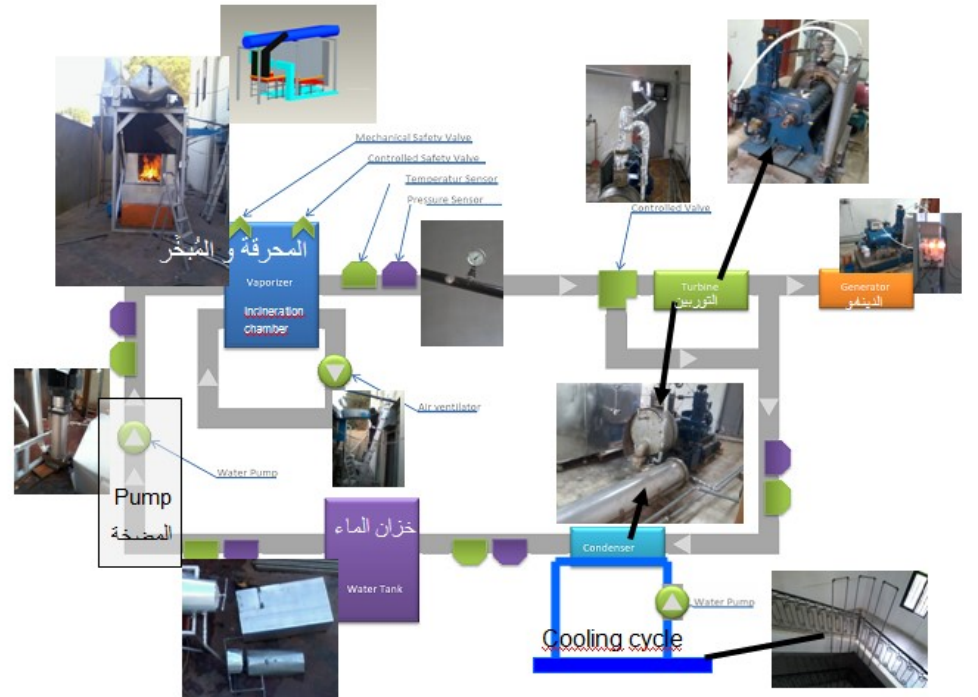
محطة طاقة مع توربين تعمل على البخار بشكل عام

- دورة الماء مُغلقة و تتغير حالة الماء ما بين سائل و بخار.
- وظيفة المحطة هي نقل الطاقة الحرارية الى طاقة كهربائية.



27.1.1 محطة طاقة عن طريق حرق النفايات لتبخير الماء قرب طرابلس الشام

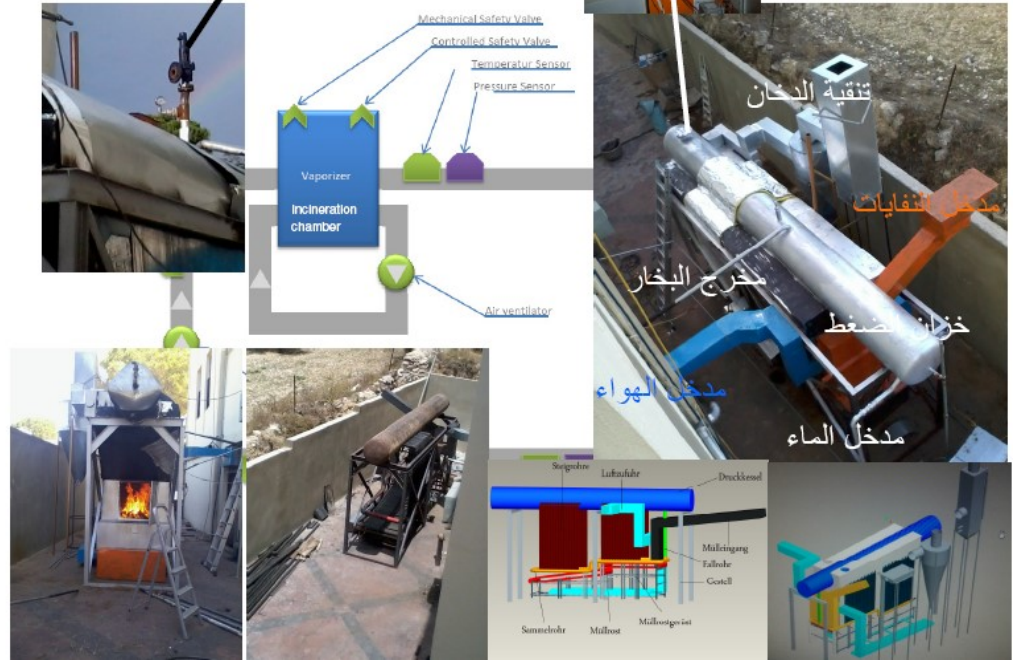
27.1.1



تدخل النفايات الى المحرقة عن طريق المدخل المخصص لها. تحرق النفايات فيتسخن الماء الموجود في الخزان فوق المحرقة حتى يصل الماء الى درجة التبخر. عندما يصل ضغط البخار الى 14 بار تُفتح الصمامة والبخار يجري الى التوربين

استخدام برامج لا تحتاج الى رخصة في ميدان ديناميكيات الموائع الحسابية

ويولد الكهرباء. يخرج البخار من التوربين الى المكثف حيث يرجع ماءً. هذه الماء تعود الى الخزان البارد و منه عن طريق المضخة مرة اخرى الى خزان المبخر.



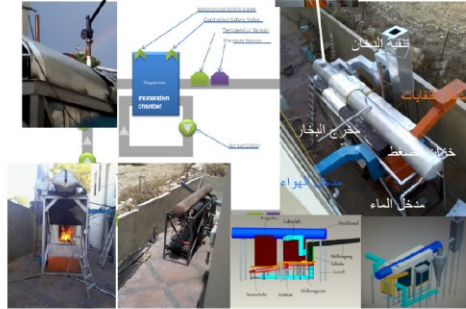
محطة الطاقة التجارية في راسنحاش - البتون قرب طرابلس في شمال لبنان تولد كهرباء عن طريق حرق الخشب او النفايات



www.temo-ek.de

Ras Nhache/Batroun - Tripoli, 11th Jan 2015

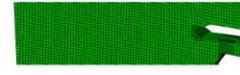
TEMO-IPP Incineration Demonstration Plant Ras Nhache/Batroun, Lebanon



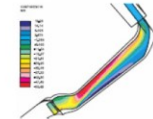
Upscaled vaporizer train element (TEMO-IPP has to be upscaled in such a way) (picture is from Dr.-Ing. M. Franz, "Dampferzeuger", www.axpoholz.ch/Dampferzeuger.pdf)

Vaporizer of TEMO-IPP incineration demonstration plant at Ras Nhache/Batroun

CFD Analysis step 1: Upscaling CAD Model of vaporizer (to be done by student working on Master Thesis Mechanical Analysis of an upscaled version of the Vaporizer (pressure vessel and circulation tubes) of the incineration pilot power plant TEMO-IPP)



CFD Analysis step 2: Grid generation



CFD Analysis step 3: Calculated water/steam flow

Master Thesis

Computational Fluid Dynamics (CFD) Analysis for Water/Steam flow in an upscaled version of the vaporizer of incineration power plant TEMO-IPP

To be able to upscale the TEMO-IPP incineration plant to a commercial incineration plant (about 40 MW) in Tripoli or elsewhere in North Lebanon critical components shall be verified by Computational Fluid Dynamics with the tool Abaqus. The main critical component is the pressure vessel with about 100 bar pressure difference. Working packages:

1. CAD Modeling	2. Mesh Generation	3. Solver	4. Visualization	5. Documentation
Upscaling CAD Model with ProE (to be done by other student –see above)	A mesh generation C++ code shall be taken from the open source code OpenFoam and migrated to TEMO_IPP-CFD tool.	A finite difference and a finite volume C++ code shall be taken from the open source code OpenFoam and migrated to TEMO_IPP-CFD tool.	Shall be done with the tool Paraview	
	4 weeks	6 weeks	4 weeks	3 weeks

Keywords: Alternative Energy, Steam Generation in power plant, Computational Fluid Dynamics (CFD), OpenFoam, C++

Contact: Samir Mourad, Email: samir.mourad@aecenar.com

حل المسألة 27.1.3

العمل على برمجيات FreeCAD, Gmsh, Elmer, لدراسة السلوك الميكانيكي و حركة البخار على حد سواء في المبخر.

دراستنا هي جريان الماء داخل انابيب محرقة لمحطة طاقة تعمل على حرق النفايات، لذلك يجب علينا ادخال تصميم جزء من هذه المحطة. هذا التصميم هو تصميم انشئ ببرنامج FreeCAD ولذلك علينا ان ننقل تصميم FreeCAD إلى OpenFOAM قبل التشغيل البرنامج.

OpenFOAM للحل:

مشكلتنا الآن هو كيف يمكننا أن نفعل هذا النقل:

أولاً؛ نفتح تصميم freeCAD على OpenFOAM ونحاول استخدام file.VTK لكننا لا نحصل على نتيجة.

ثانياً؛ نحاول نقل الملف على paraview ثم على OpenFOAM، لكننا لا نحصل على نتيجة أيضاً.

ثالثاً؛ نحن نبحث على الانترنت عن بعض الرموز، ونحن نحاول التحقق من ذلك، ولكن لا نتيجة.

رابعاً؛ نحاول إنشاء مجلد جديد نسميه اسطوانة للقيام ببعض التجارب، و نقدم الشروط بالاحرف الاولى (p-U)،

وحالة النظام (fvSchemes- fvSolutions- controlDict)، ولكن في polyMesh في مجلد الثوابت ندرج

الإحداثيات الجديدة ل الملف freeCAD غير المقروء من قبل OpenFOAM.

وجدنا رمز stl. لكن نستنتج أن هذا الرمز هو رمز عكسي يمكننا من النقل من OpenFOAM إلى

freeCAD.

نحن نحاول نقل احداثيات freeCAD ل OpenFAOM مباشرة ولكن البرنامج لا يقرأها.

حولنا ملف البرنامج FOAM ل vtkp . (foamToVTKP) والبرنامج لا يزال غير مقروء.

نحن نبحت كيف يمكننا قراءة رموز freeCAD باستخدام Visual C++ ولكن Visual C++ لا يمكن فتح

رموز freeCAD.

نستخدم (.ast) رمز للملف لكنها ليست مقروءة من OpenFOAM.

ندرج ماكرو macro في FreeCAD لعرض إحداثيات محطة الحرق للطاقة في OpenFAOM ولكن لا يمكن

قراءة الإحداثيات.

باستخدام Gmsh in OpenFOAM محلاً:

وجدنا أن Gmsh يجتمع مع OpenFOAM بالتالي فإننا نثبت Gmsh في Linux.

تركيب Gmsh:

الطريقة الأولى لحل:

1. فتح محطة (استخدام سطر الأوامر) في إطار Linux

2. تصور الدليل README.text

3. تشغيل برنامج

• إنشاء دليل البناء (build): MKDIR بناء.

• cmake تشغيل من ضمن الدليل بناء: cd build

cmake

4. بناء Gmsh باستخدام واجهة المستخدم الرسومية ل CMake.

• CMake ملء -----في.

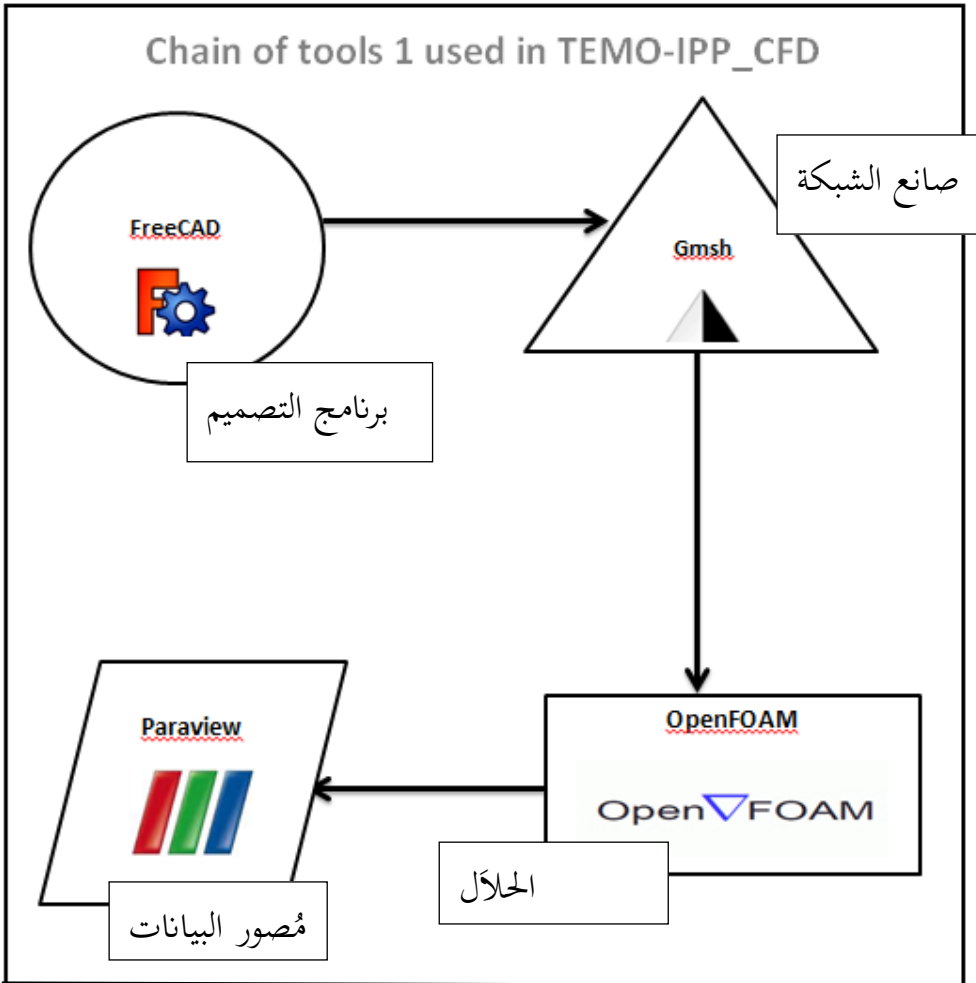
• إضافة الدخول ("CMake_PREFIX_PATH "PATH).

• "تكوين" من اختيار المترجم.

• لدينا لإعادة تشغيل "تكوين" في كل مرة نغير بعض الخيارات.

• "إنشاء".

• بناء Gmsh باستخدام مترجم المختار.



Chain 1: باستخدام 1 سلسلة الأدوات

نقوم بتحميل نسخة Gmsh الجديد (gmsH-2.6.1-source.tgz)

تشغيل gmsH:

الملف المحمل -gmsH-2.6.1-source.tgz

ثم يتم بناؤه في دليل البناء (build) منفصل ونحولنا مع:

MKDIR build-gmsH

Cd build-gmsH

تم تكوين GMSH مع:

```
ccmake -i ../gmsh-2.6.1-source
```

ثم 'c' لتكوين، 'c' مرة أخرى لتكوين، 'g' لتوليد. إذا واجهت 'مساعدة' الشاشات، اضغط على 'e' للخروج منها.

ثم يتم ترجمة GMSH وتثبيتها مع:

```
make
```

```
sudo make install
```

```

CMake Error at /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineSystem.cmake:138 (FILE):
file Internal CMake error when trying to open file:
/home/meae/gmsh-2.6.1-source/CMakeFiles/CMakeOutput.log for writing.
Call Stack (most recent call first):
  CMakeLists.txt:17 (project)

CMake Error: Could not open file for write in copy operation /home/meae/gmsh-2.6.1-source/CMakeFiles/CMakeSystem.cmake.tmp

CMake Error: : System Error: No such file or directory

CMake Error at /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineSystem.cmake:150 (CONFIGURE_FILE):
configure_file Problem configuring file
Call Stack (most recent call first):
  CMakeLists.txt:17 (project)

CMake Error at /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCompilerId.cmake:63 (FILE):
file problem creating directory:
/home/meae/gmsh-2.6.1-source/CMakeFiles/CompilerIdCXX
Call Stack (most recent call first):
  /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCompilerId.cmake:25 (CMAKE_DETERMINE_COMPILER_ID_BUILD)
  /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCXXCompiler.cmake:128 (CMAKE_DETERMINE_COMPILER_ID)
  CMakeLists.txt:17 (project)

CMake Error at /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCompilerId.cmake:63 (FILE):
file problem creating directory:
/home/meae/gmsh-2.6.1-source/CMakeFiles/CompilerIdCXX
Call Stack (most recent call first):
  /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCompilerId.cmake:25 (CMAKE_DETERMINE_COMPILER_ID_BUILD)
  /home/meae/OpenFOAM/ThirdParty-1.6/cmake-2.6.4/platforms/linux/share/cmake-2.6/Modules/CMakeDetermineCXXCompiler.cmake:128 (CMAKE_DETERMINE_COMPILER_ID)
  CMakeLists.txt:17 (project)

Errors occurred during the last pass

CMake Version 2.6 - patch 4
Press [e] to exit help

```

Figure 27.1.3-1: تحميل برنامج Gmsh على نظام التشغيل Linux- Red Hat

وجدنا بعض المشاكل التي تواجهنا لتثبيت gmsh على Redhat Linux التي لا تحتوي على "تكوين" وعلينا تثبيت cmake التي لا تتطابق مع نسختنا Redhat.

نقوم بتحميل Linux-gmsh-2.9.3 ، في نسخة Ubuntu ، ثم نجد ملفين (ben و share) إدخال ملف ben وجدنا (gmsh*) نكتب (./gmsh) و gmsh تثبت.

```

-rw-rw-r-- 1 iap iap 100505 May 6 09:56 Re_El_Maoun_04052015.pdf
-rw-rw-r-- 1 iap iap 0 Mar 27 2014 touch6517
iap@iap-HP-G62-Notebook-PC:~/Downloads$ cd CMakeFiles/
iap@iap-HP-G62-Notebook-PC:~/Downloads/CMakeFiles$ ll
total 12
drwxrwxr-x 2 iap iap 4096 May 25 11:06 ./
drwxr-xr-x 6 iap iap 4096 May 25 11:09 ../
-rw-rw-r-- 1 iap iap 85 May 25 11:09 cmake.check_cache
iap@iap-HP-G62-Notebook-PC:~/Downloads/CMakeFiles$ cd ..
iap@iap-HP-G62-Notebook-PC:~/Downloads$ cmake -i gmsh
gmsh_2.8.5+dfsg-1.1ubuntu1.dsc gmsh-2.9.3-Linux64.tgz
gmsh_2.8.5+dfsg.orig.tar.xz gmsh-build/
gmsh-2.9.3-Linux/
iap@iap-HP-G62-Notebook-PC:~/Downloads$ cmake -i gmsh
gmsh_2.8.5+dfsg-1.1ubuntu1.dsc gmsh-2.9.3-Linux64.tgz
gmsh_2.8.5+dfsg.orig.tar.xz gmsh-build/
gmsh-2.9.3-Linux/
iap@iap-HP-G62-Notebook-PC:~/Downloads$ cmake -i gmsh-
gmsh-2.9.3-Linux/ gmsh-2.9.3-Linux64.tgz gmsh-build/
iap@iap-HP-G62-Notebook-PC:~/Downloads$ cmake -i gmsh-2.9.3-Linux
Would you like to see advanced options? [No]:
Please wait while cmake processes CMakeLists.txt files....

CMake Error: The source directory "/home/iap/Downloads/gmsh-2.9.3-Linux" does no
t appear to contain CMakeLists.txt.
Specify --help for usage, or press the help button on the CMake GUI.

iap@iap-HP-G62-Notebook-PC:~/Downloads$ cd gmsh-2.9.3-Linux/
iap@iap-HP-G62-Notebook-PC:~/Downloads/gmsh-2.9.3-Linux$ ll
total 16
drwxrwxr-x 4 iap iap 4096 May 25 11:08 ./
drwxr-xr-x 6 iap iap 4096 May 25 12:42 ../
drwxrwxr-x 2 iap iap 4096 May 25 10:39 bin/
drwxrwxr-x 4 iap iap 4096 May 25 10:39 share/
iap@iap-HP-G62-Notebook-PC:~/Downloads/gmsh-2.9.3-Linux$ cd bin/
iap@iap-HP-G62-Notebook-PC:~/Downloads/gmsh-2.9.3-Linux/bin$ ll
total 66408
drwxrwxr-x 2 iap iap 4096 May 25 10:39 ./
drwxrwxr-x 4 iap iap 4096 May 25 11:08 ../
-rwxr-xr-x 1 iap iap 67972608 Apr 18 10:45 gmsh*
-rw-r--r-- 1 iap iap 19059 Mar 17 18:03 onelab.py
iap@iap-HP-G62-Notebook-PC:~/Downloads/gmsh-2.9.3-Linux/bin$ ./gmsh

```

Figure 27.1.3-2: تحميل Gmsh على نظام التشغيل Linux- Ubuntu 14.04

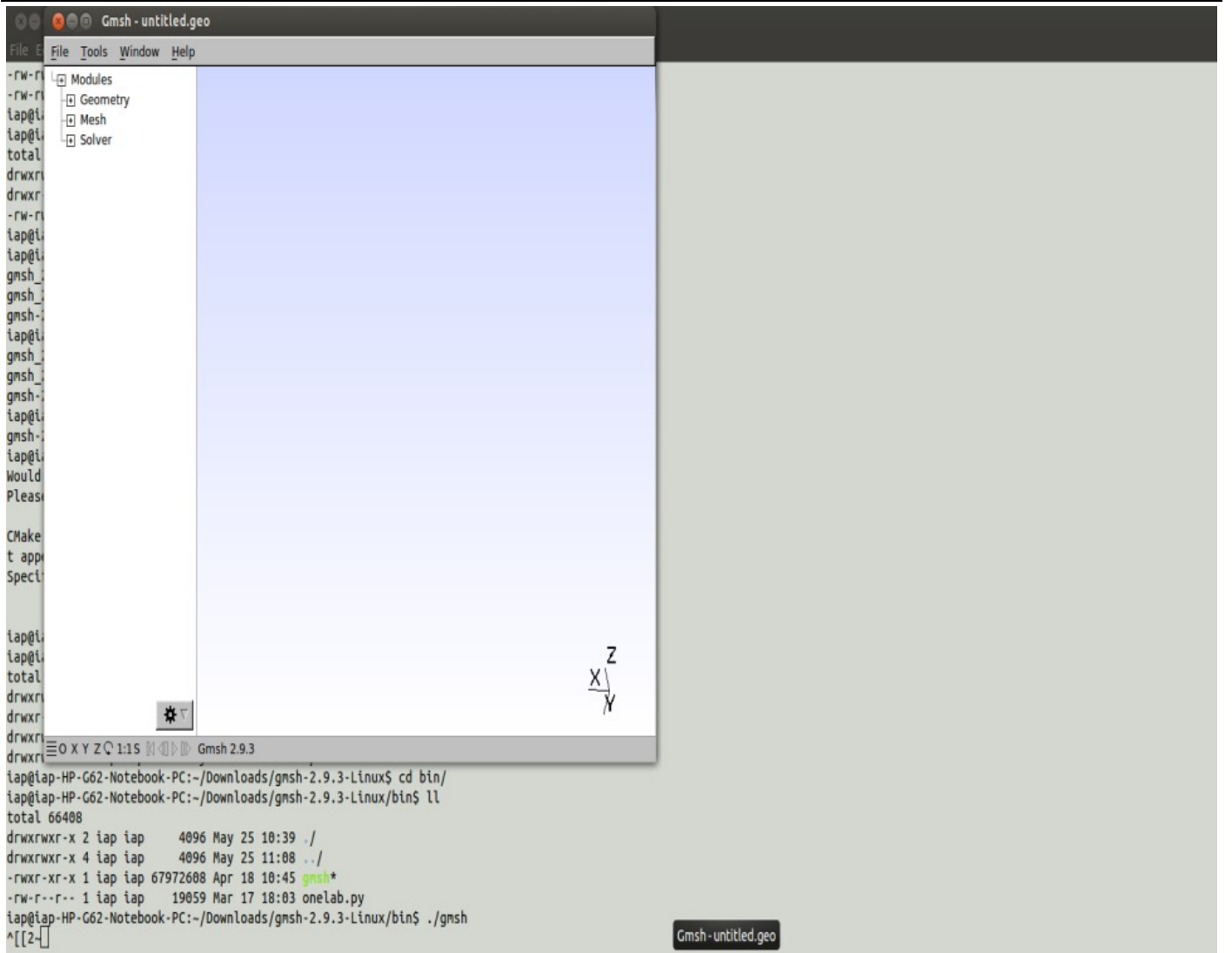


Figure 27.1.3-3: واجهة برمجية Gmsh على نظام التشغيل Linux_Ubuntu 14.04

تصميم محطة للطاقة للحرق في برنامج FreeCAD:

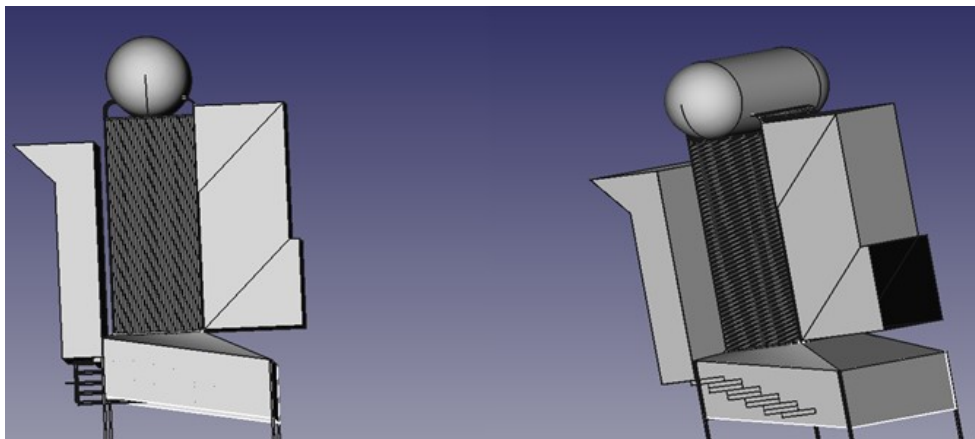
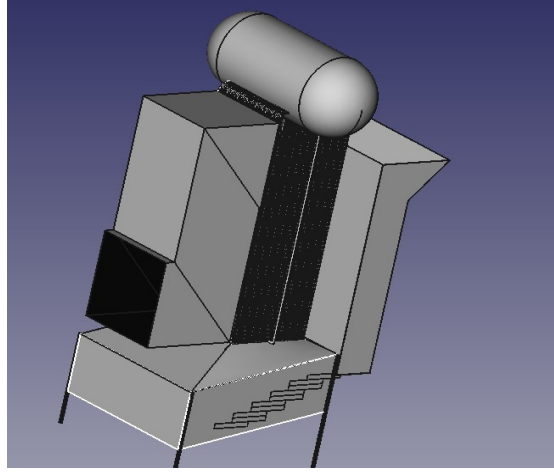
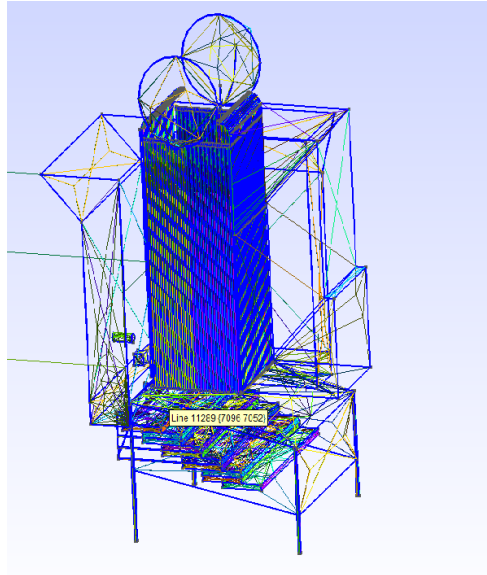


Figure 27.1.3-4: التصميم FreeCAD



التصميم FreeCAD Figure 27.1.3-5:

نحصل على شبكة باستخدام Gmsh:



Gmsh التشبيك للتصميم عبر Figure 27.1.3-6:

A. الآن نحاول نقل شبكة لOpenFOAM:

1. gmshToFoam: لا يستجيب.

2. cp -r \$FOAM_TUTORIALS/incompressible/icoFoam/cavity/* ./files.msh {name}: لا يستجيب.

B. لقراءة gmsh ملف msh. التي كتبها OpenFOAM نتبع الأوامر:

Gmsh main.geo -3 0 file.msh .1

case-vaporisor gmshToFoam file.msh .2

blockMesh .3

icoFoam .4

paraFoam .5

ولكنه لا يؤثر.

C. نغير file.msh إلى file.STL لحلها باستخدام snappyMesh ولكنه لا يؤثر.

D. تتبع طريقة أخرى:

1. جعل مجلد جديد في icoFoam

2. نسخ الظروف الأولية في هذا المجلد من آخر وجود البرنامج التعليمي

3. نسخ file.msh في هذا المجلد

4. كتابة fluentMeshToFoam file.msh

5. icoFoam

6. paraFoam

ولكننا نرى أن علينا جعل الشروط الحدية التصميم (حدود، نقط، وجوه...).

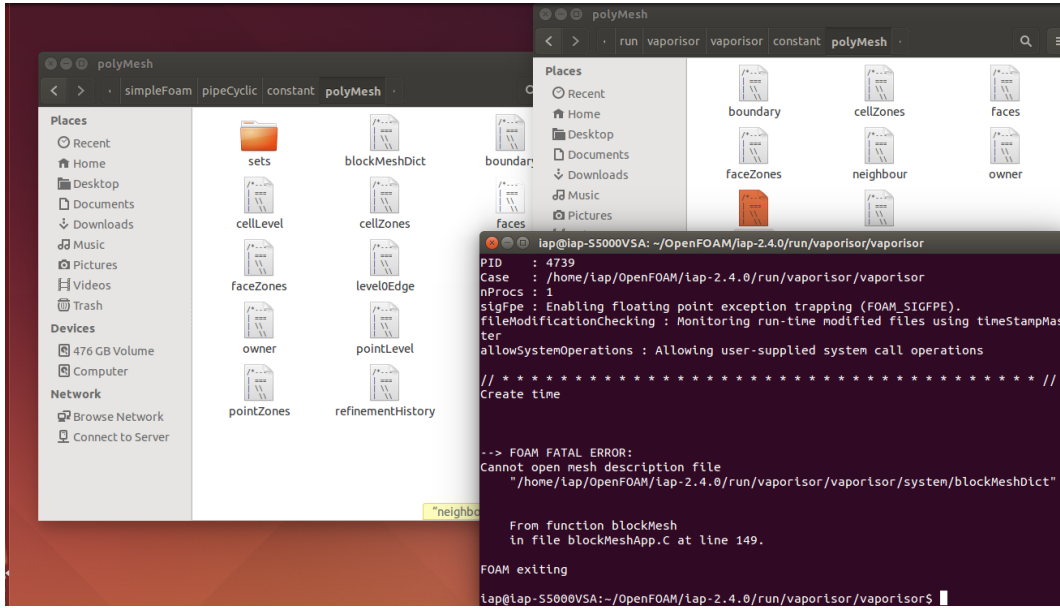


Figure 27.1.3-7: واجهة Linux-Ubuntu

:
 ونحن نحاول العثور على هذه الحدود من file.geo المُنْتِجَة في Gmsh أو
 الشبكة file.msh صنع في Gmsh أيضا.
 ونحن نحاول الآن نسخة جديدة من Gmsh (Gmsh 2.3) ونبدأ مع أنبوب
 كمثال:

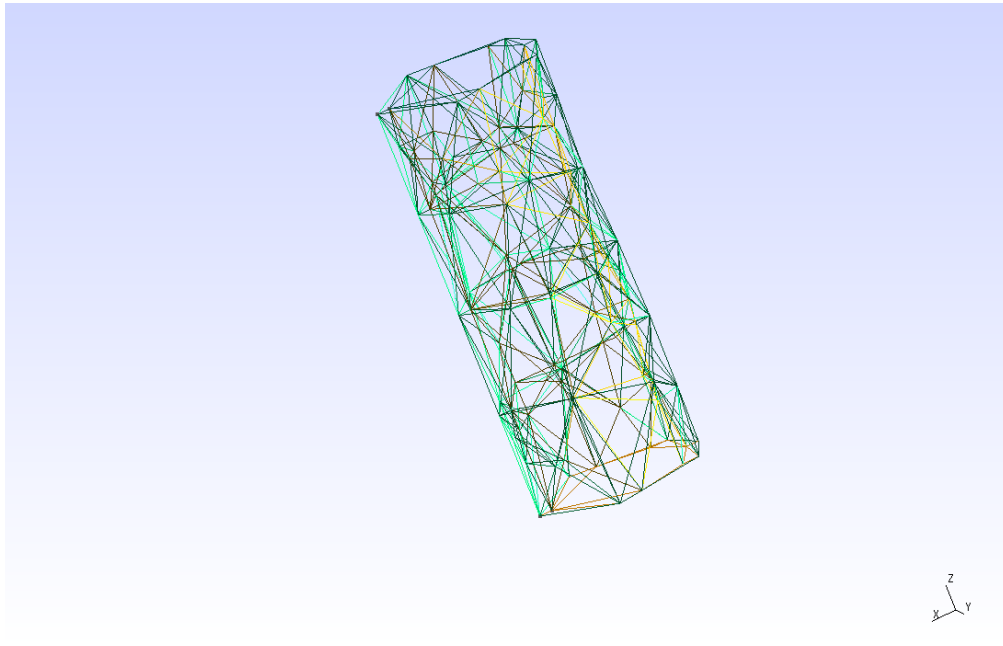


Figure 27.1.3-8: المثال لتشيبيك الأنبوب في Gmsh


```

Unhanded element 1 at line 156
Unhanded element 1 at line 157
Unhanded element 1 at line 158
Unhanded element 1 at line 159
Unhanded element 1 at line 160
Unhanded element 1 at line 161
Unhanded element 1 at line 162
Unhanded element 1 at line 163
Unhanded element 1 at line 164
Unhanded element 1 at line 165
Unhanded element 1 at line 166
Unhanded element 1 at line 167
Mapping region 0 to Foam patch 0
Mapping region 0 to Foam cellZone 0
Cells:
  total:365
  hex :0
  prisms:0
  pyr :0
  tet :365

CellZones:
Zone  Size
0     365

Skipping tag at line 764
Patch 0 gets name patch0

--> FOAM Warning :
From function polyMesh::polyMesh(... construct from shapes...)
in file meshes/polyMesh/polyMeshFromShapeMesh.C at line 627
Found 230 undefined faces in mesh; adding to default patch.
Finding faces of patch 0

FaceZones:
Zone  Size

Writing zone 0 to cellZone cellZone_0 and cellSet
End

iap@iap-S5000VSA:~/OpenFOAM/iap-2.4.0/run/vaporisor/cylindre$ icoFoam

```

Figure 27.1.3-11: النتيجة 2 تطبيق gmshToFOAM

لكننا لا نحصل على المعلومات خلال الوقت عندما نطبق icoFoam:

```

=====
      \   / F i e l d           | OpenFOAM: The Open Source CFD Toolbox
      /   \ O peration         | Version: 2.4.0
      /     \ A n d             | Web: www.OpenFOAM.org
      \     / M anipulation    |
      \___/ |-----|
Build : 2.4.0-f0842aea0e77
Exec   : icoFoam
Date   : Jul 08 2015
Time   : 10:22:34
Host   : "iap-S5000VSA"
PID    : 4476
Case   : /home/iap/OpenFOAM/iap-2.4.0/run/vaporisor/test
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations

// * * * * *
Create time

Create mesh for time = 0

Reading transportProperties

Reading field p

--> FOAM FATAL IO ERROR:
Cannot find patchField entry for patch0

file: /home/iap/OpenFOAM/iap-2.4.0/run/vaporisor/test/0/p.boundaryField from line 11 to line 39.

From function GeometricField<Type, PatchField, GeoMesh>::GeometricBoundaryField::readField(const DimensionedField<Type, GeoMesh>&, const dictionary&)
in file /home/openfoam/OpenFOAM/OpenFOAM-2.4.0/src/OpenFOAM/lninclude/GeometricBoundaryField.C at line 289.

FOAM exiting

iap@iap-S5000VSA:~/OpenFOAM/iap-2.4.0/run/vaporisor/test$

```

Figure 27.1.3-12: النتيجة 3 تطبيق gmshToFOAM

وOpenFOAM لا يستجيب، ونحن لا يمكننا قراءة تصميم gmsh للشبكة في OpenFOAM. قد يكون ذلك للأسباب التالية:

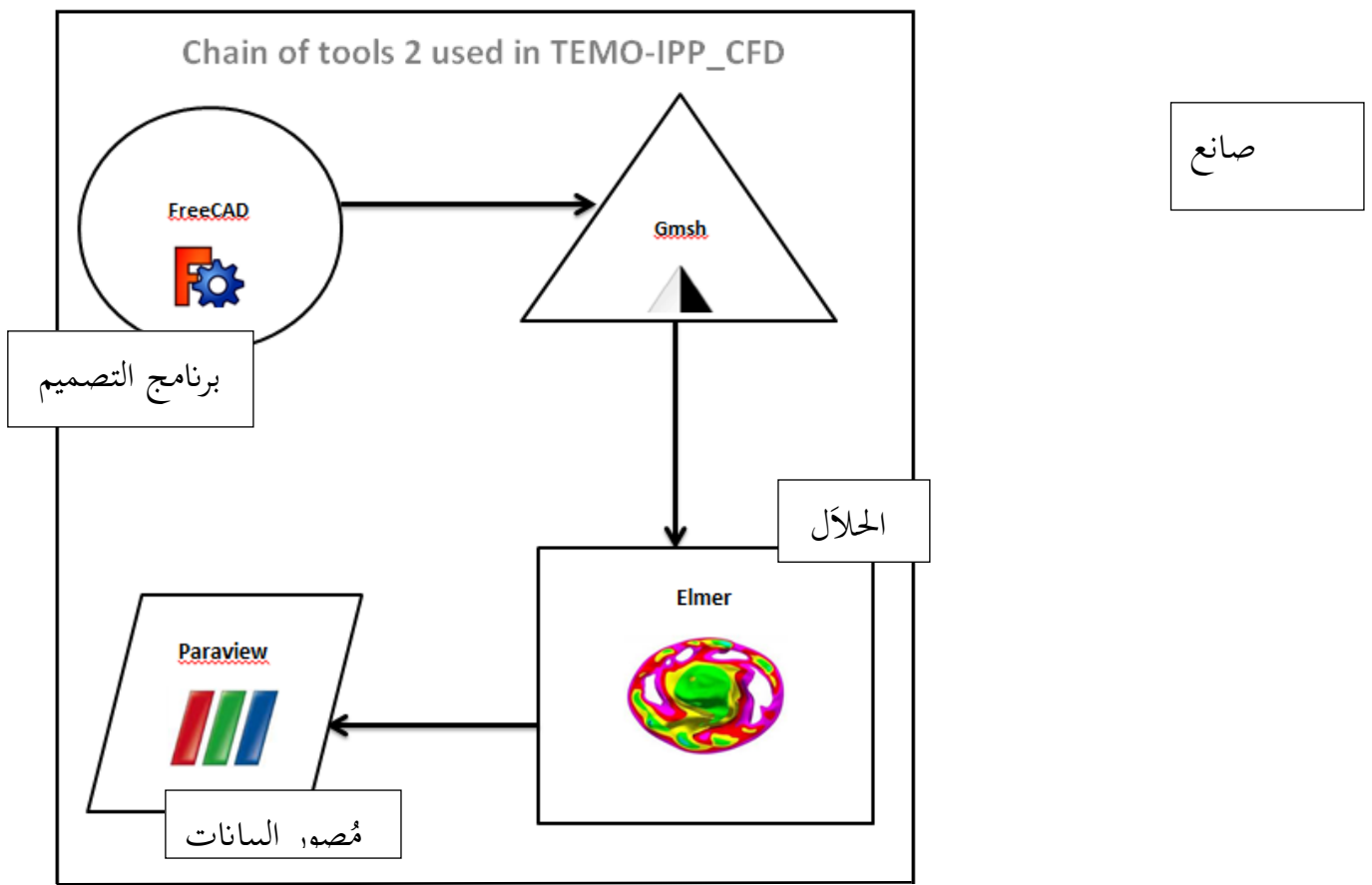
• النسخة OpenFOAM ليست كاملة ولكن هذا ليس منطقياً لأنه يستخدم في البرامج التعليمية على شبكة الانترنت.

• لا يعمل الأمر command بعد الآن، والتي هي أكثر منطقية لأننا حاولنا العديد من وسائل لتطبيق الأمر ولا شيء يحدث.

• نسخة Gmsh لا تعمل ولكن هذا ليس من المنطقي جداً لأننا نستخدم نسخ كثيرة من Gmsh.

استخدام برنامج Elmer

27.1.3.2



Chain 2: باستخدام 2 سلسلة الأدوات

هنا نجرب برمجيات أخرى حيث أن Elmer يستطيع قراءة بيانات التصميم للأنبوب:

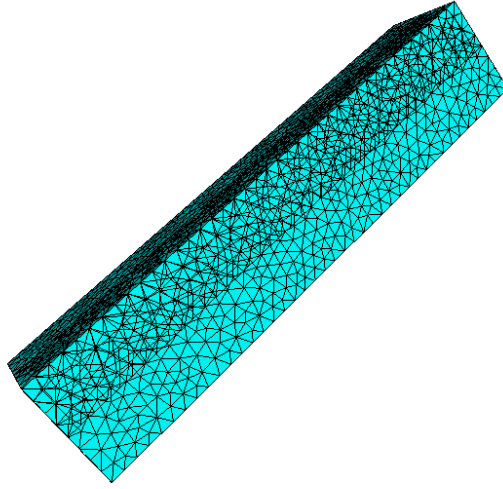


Figure 27.1.3-13: Elmer الأنبوب في

- تصميم الأنابيب مع الشبكة على gmsht وتوفير مثل شكل file.msh
 - الشروط الأولية
 - اختيار معادلة نافير ستوكس Navier-Stokes
 - تحديد استخدام المواد (الماء درجة حرارة الغرفة) للواجهة الداخلية للأنبوب، والستانيلس ستيل للواجهة الخارجية (...)
 - تحديد الحدود في التصميم قبل إدخال حالة كل الحدود
 - تشغيل بدء المحلل
 - ثم حدد بداية ElmerPost أو ElmerVTK.
- قبل أن تنتقل إلى النتائج، علينا أن نعرف كيف نحصل على الشروط الأولية:
- نحن في حاجة إلى توربينة تعمل على البخار لتولد 30.2 ميغا واط.
- مواصفات البخار الحي الذي يدخل إلى التوربينة: ضغطه 120 بار، حرارته 520 درجة مئوية، مع معدل تدفق للبخار يساوي 58 كغ في الثانية.

نحن نستخدم الماء، فالكثافة تساوي 1000 كغ / في المتر المكعب. يمكننا أن نستنتج أن حجم التدفق س يساوي 0.058 متر مكعب / ثانية.

$$Q = \frac{\text{الحجم}}{\text{الوقت}} = \frac{\text{المساحة} * \text{الارتفاع}}{\text{الوقت}} = \text{المساحة} * \text{سرعة}.$$

$$\text{الشعاع}^2 * \pi = 0.03 \text{ متر مربع}$$

إذا السرعة تساوي 19.44 م / ث.

نحصل على قيم السرعة للبخار عبر (start solver) والتصوير عبر (ElmerPost):

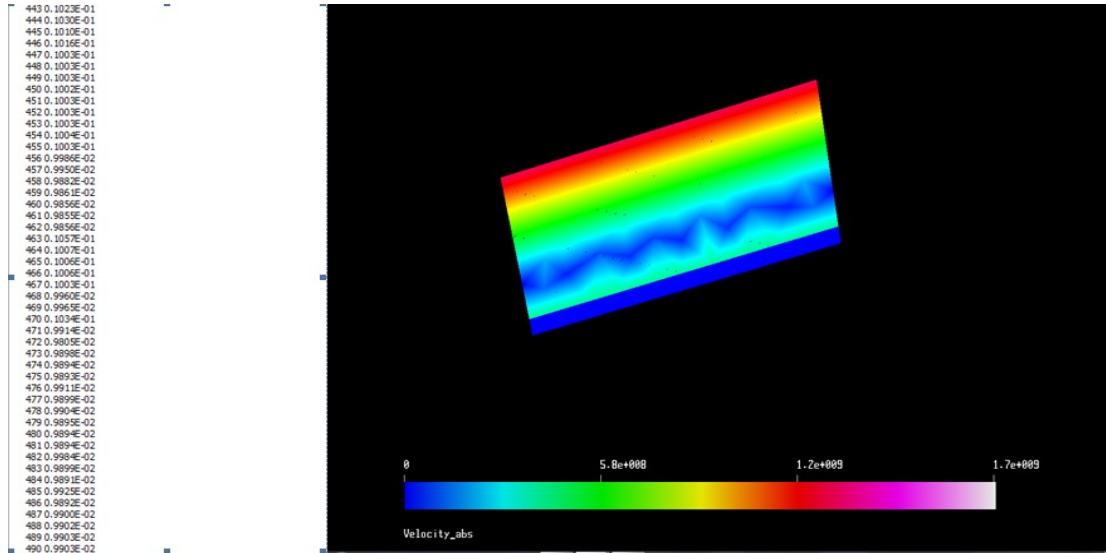


Figure 27.1.3-14: قيم السرعة

تتغير السرعة حسب النموذج التالي:

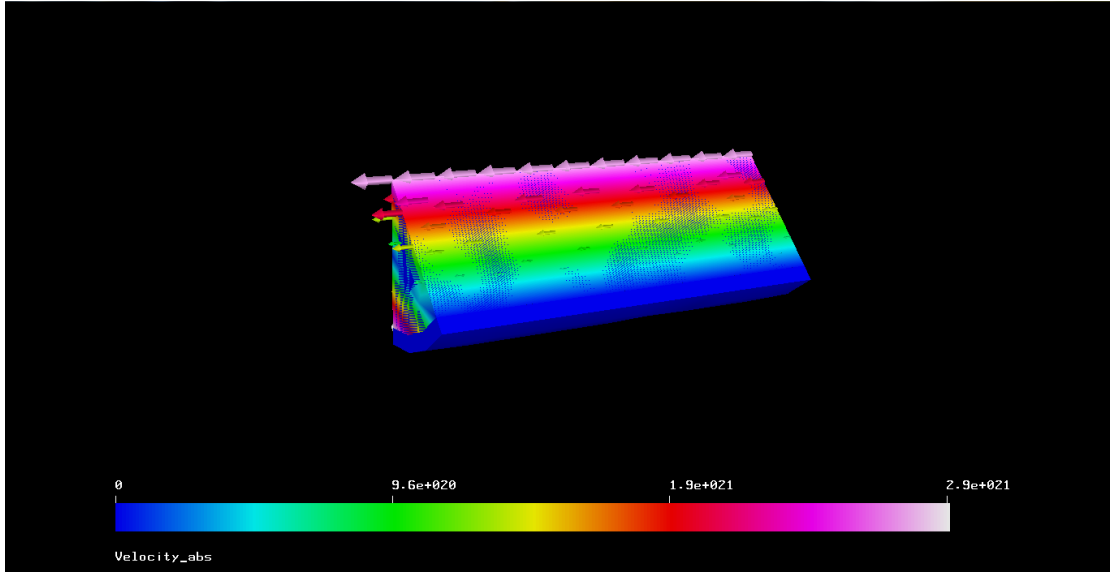


Figure 27.1.3-15: قيم السرعة مرموزة بموجه

الحد الأقصى في المنتصف يقل مع السير نحو حدود الأنابيب والسبب هو احتكاك البخار مع المادة المصنوع منها الأنبوب مما يخفف السرعة عند الأطراف

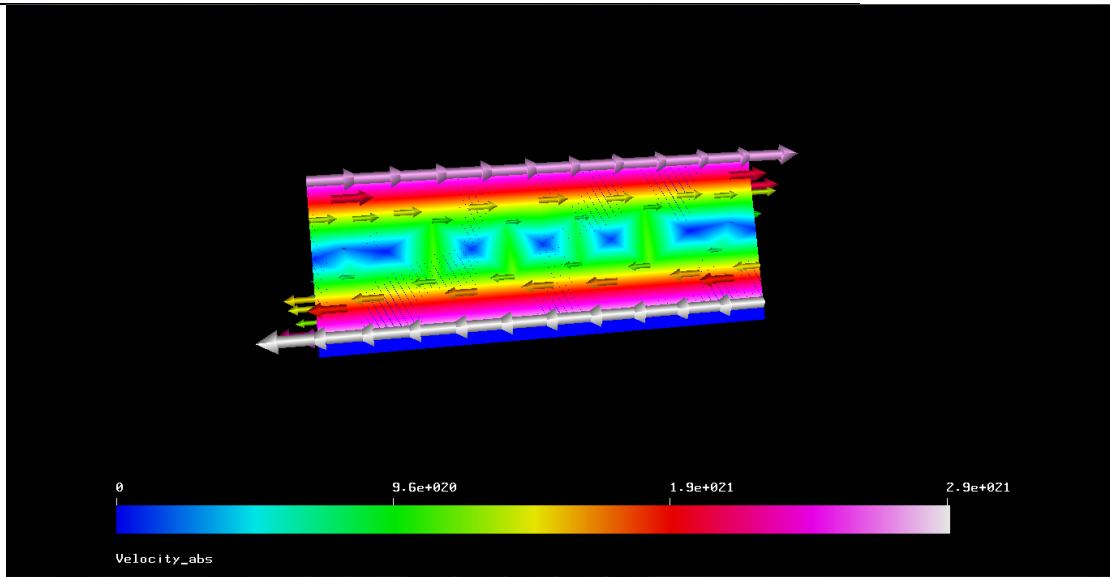


Figure 27.1.3-16: نتائج الاحتكاك

الآن نُدخل تصميم محطة الطاقة لبرنامج Elmer ولكن من الصعب حاليا أن ننقل تصميم كامل لذا نُدخل مسار الماء فقط الموضح في الشكل التالي:

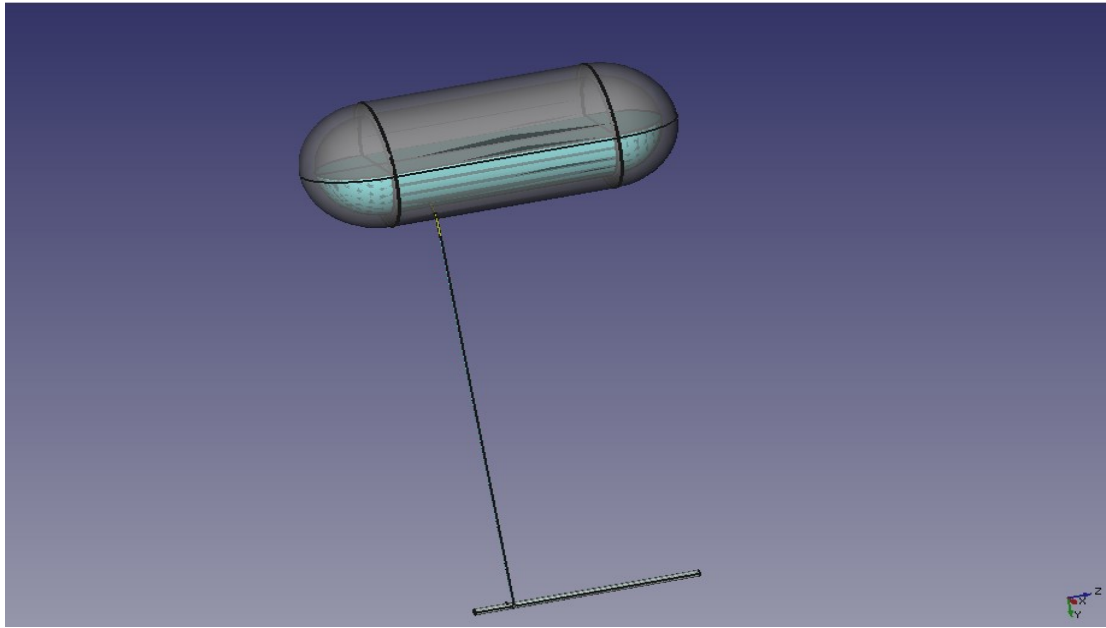


Figure 27.1.3-17: The studied design

علينا أن نعرف بعض الملاحظات:

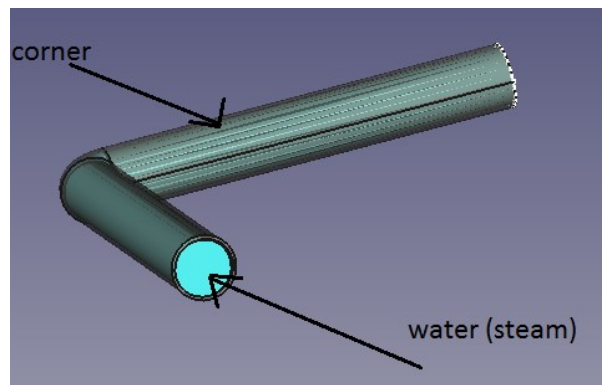


Figure 27.1.3-18: أنبوب مع كوع

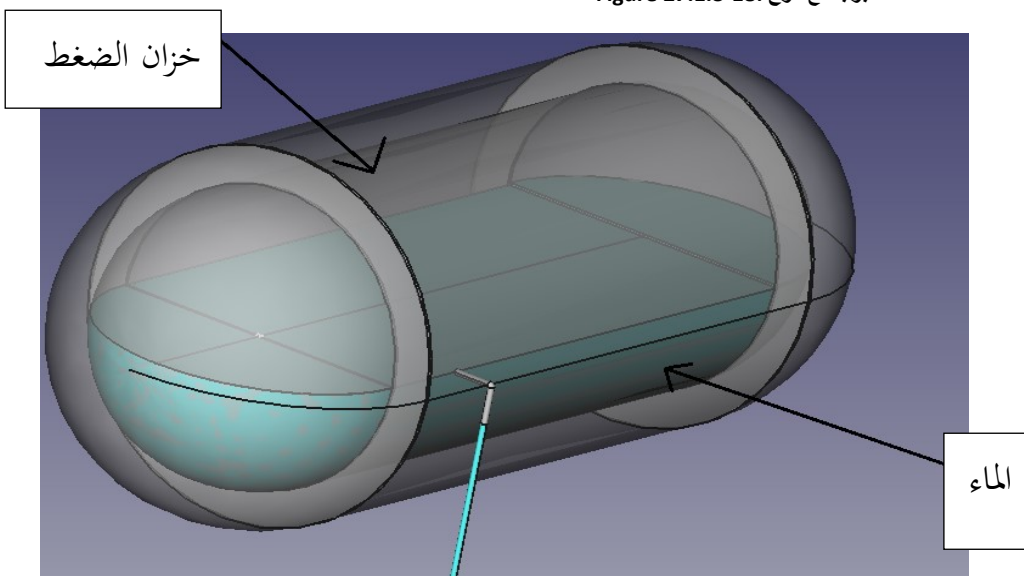


Figure 27.1.3-19: Noted drump

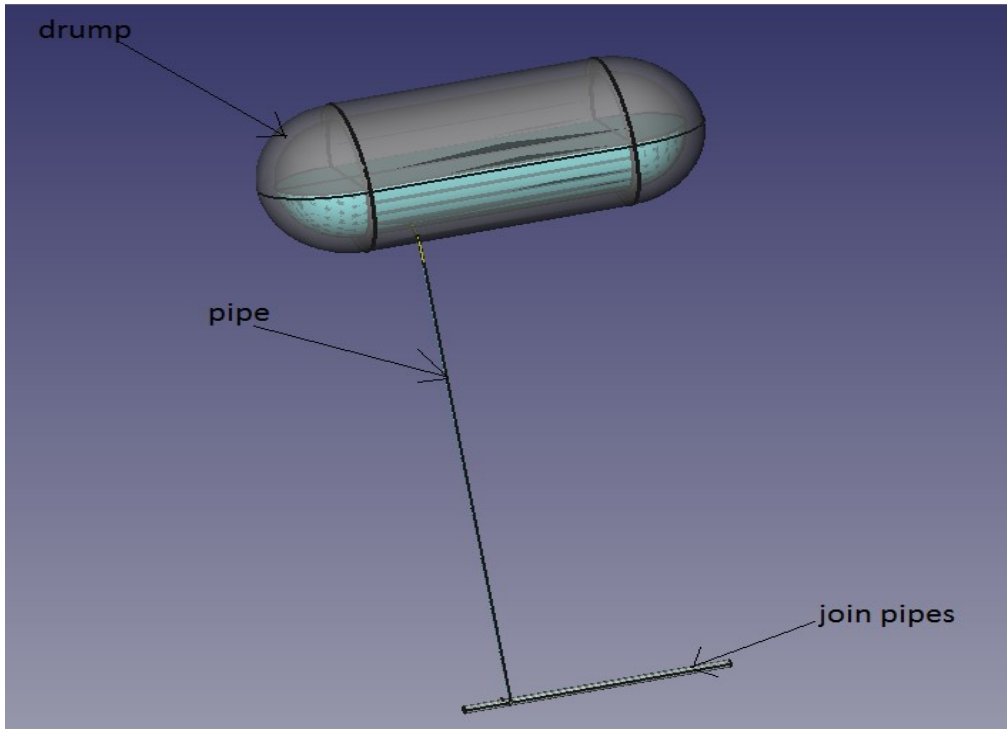


Figure 27.1.3-20: Noted design

دراستنا صعبة في جهاز كمبيوتر شخصي. لذلك نحن نقوم بالدراسات في خادم quadcore مربوط في أجهزتنا الشخصية. لذلك نجعل الدراسة في الخادم وننقل النتيجة (الملفات والأرقام) لأجهزة الكمبيوتر الشخصية. لعرض ملفات Elmer المعروضة في القرص المحلي (C):

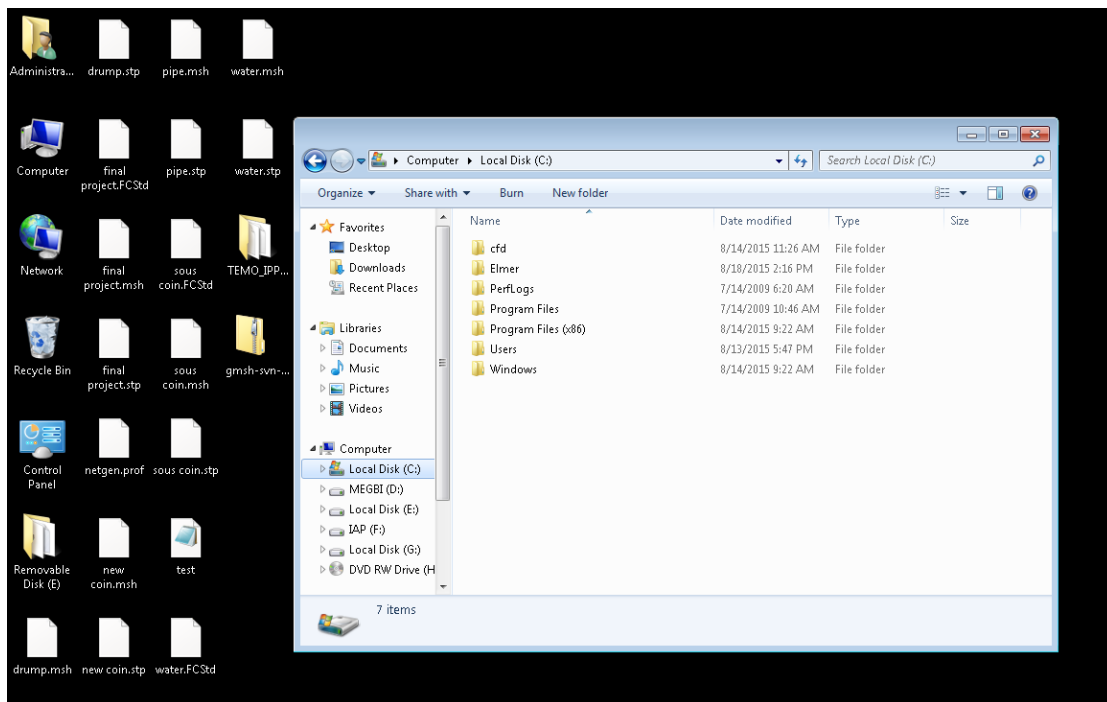
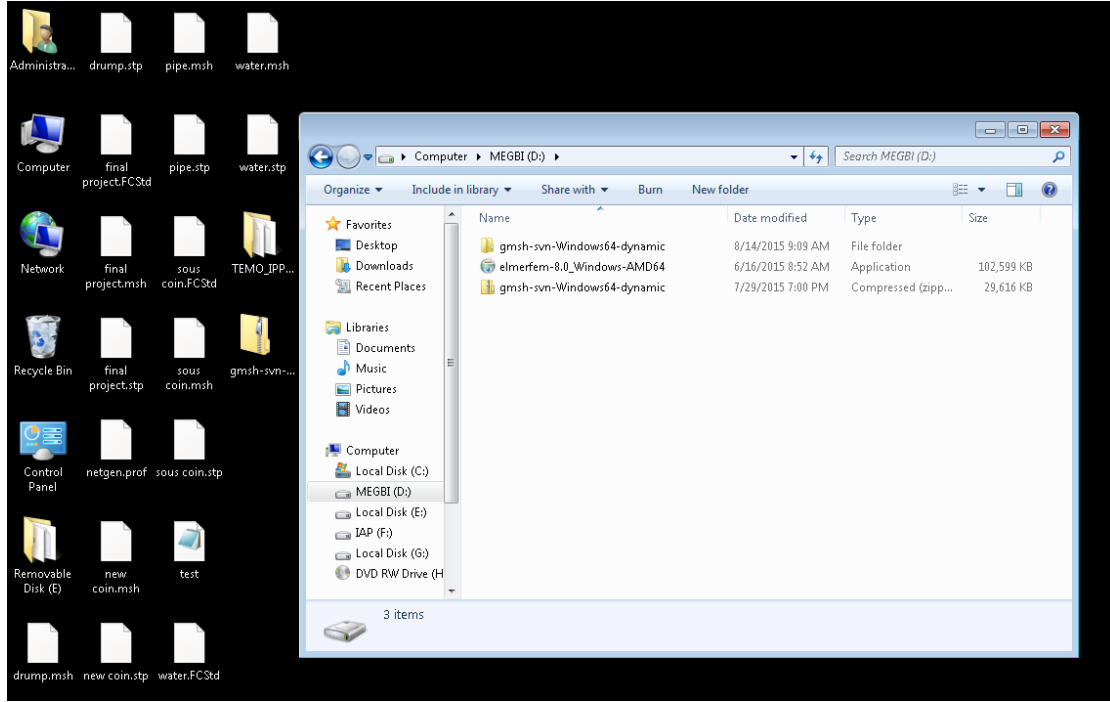


Figure 27.1.3-21: واجهة الخادم 1

ندخول إلى القرص المحلي (C) ثم إلى المجلد Elmer ثم نختار اسم الملف الذي نحتاج إليه.
ومكان Gmsh و Elmer كان في (D) MEGBI:



واجهة الخادم 2: Figure 27.1.3-22

كما نرى ملفات FreeCAD و Gmsh تقع في سطح المكتب ولكن يمكننا نقلها إلى مجلد خاص لنتمكن من تسمية FreeCAD أو Gmsh يمكننا إنشاء مجلد لكل نوع من الملفات.
من المهم أن نقول أنه علينا رسم الماء مثل المواد، لأننا نضع الشروط على الماء (أو البخار وفقاً لدرجة الحرارة) في برنامج Elmer. لذلك التصميم سيكون:

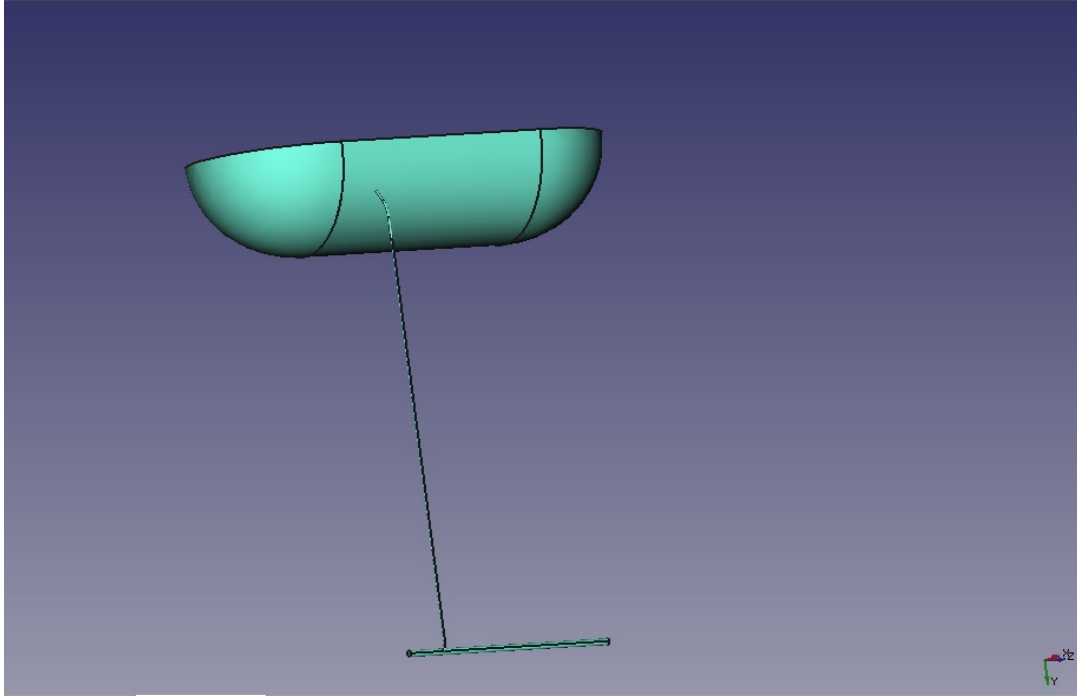


Figure 27.1.3-23: مسار البخار

الآن علينا أن نجزي التصميم باستخدام gmsh أو Elmer، ولكن Elmer غير قادر على تجزئة تصميم كبير لذلك نستخدم gmsh:

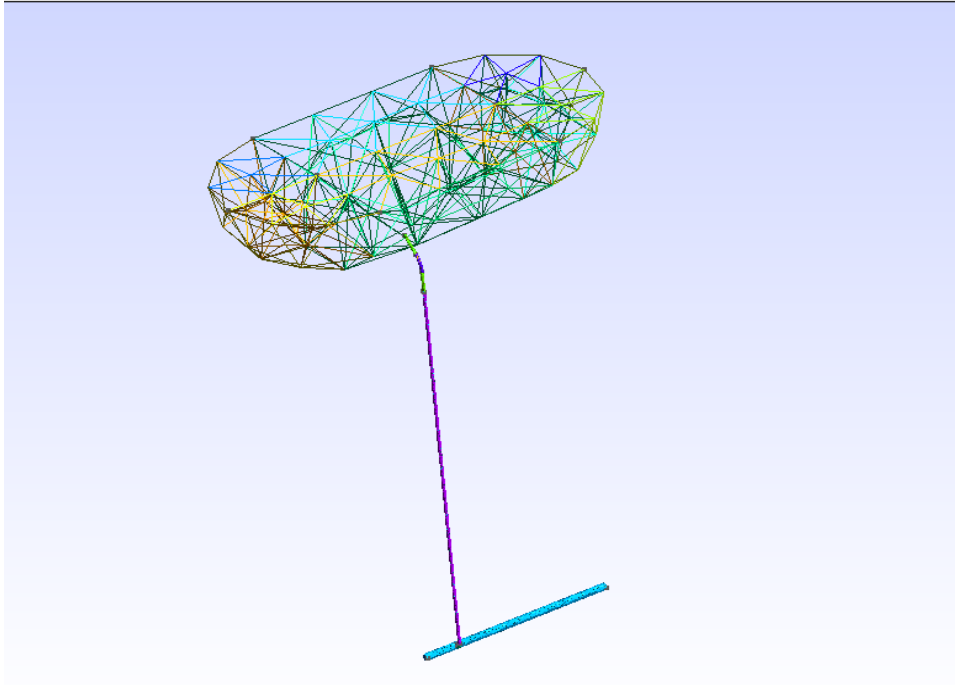


Figure 27.1.3-24: تشبيك مسار البخار في

ندخل التصميم إلى برنامج Elmer مع الشروط الأولية، ومعادلات السرعة، وشروط الحدود التي نحددها في

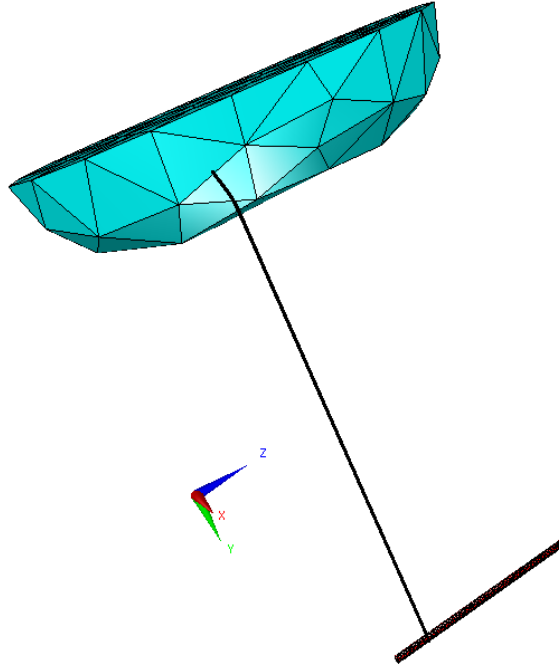


Figure 27.1.3-25: Elmer بخار الماء في

بعد تشغيل البرنامج وفقا لطريقة العناصر المحدودة، نحصل على الملفات انظر Figure 11.1.1.3-26:

Name	Date modified	Type	Size
case.ep	18/8/2015 11:46 AM	EP File	1,564 KB
case.sif	20/8/2015 11:51 PM	SIF File	3 KB
egproject	17/8/2015 11:02 AM	XML File	96 KB
ELMERSOLVER_STARTINFO	20/8/2015 11:51 PM	File	1 KB
mesh.boundary	18/8/2015 11:44 AM	BOUNDARY File	322 KB
mesh.elements	18/8/2015 11:44 AM	ELEMENTS File	431 KB
mesh.header	18/8/2015 11:44 AM	HEADER File	1 KB
mesh.nodes	18/8/2015 11:44 AM	NODES File	161 KB
netgen.prof	20/8/2015 11:57 PM	PROF File	1 KB
water.FCStd	17/8/2015 11:02 AM	FCSTD File	11 KB
water.msh	17/8/2015 11:02 AM	MSH File	1,028 KB
water.stp	17/8/2015 11:02 AM	STP File	76 KB

Figure 27.1.3-26: Elmer الملفات التي نحصل عليها من برنامج

Case.ep هو الملف الذي يحتوي على قيم السرعة والضغط.

Case.sif هو الملف الذي يحتوي على الشروط التي قمنا بتحديددها في البرنامج.
Mesh.boundary هو الملف الذي يحتوي على عدد من العناصر الحدودية، وعدد من العناصر التي تنتمي إلى الحدود، والعناصر المحيطة للحدود، نوع من رموز العناصر، والعقد من العناصر.
Mesh.elements هو الملف الذي يحوي نوع المواد المستخدمة في الدراسة مثلا هنا الستاينلس ستيل و الماء .
Mesh.header هو الملف الذي يحتوي على عدد العقد، عدد من العناصر، وعدد من عناصر الحدود.
Mesh.node هو الملف الذي يحتوي على عدد العقد، مؤشر العقد المتوازي، ويحوي تنسيق العقد.
Water.FCStd هو ملف تصميم FreeCAD.
Water.stp هو ملف تصميم gms.
water.msh هو ملف تشبيك Elmer.
تغير الألوان يعبر عن تغير قيم الضغط و الحرارة :

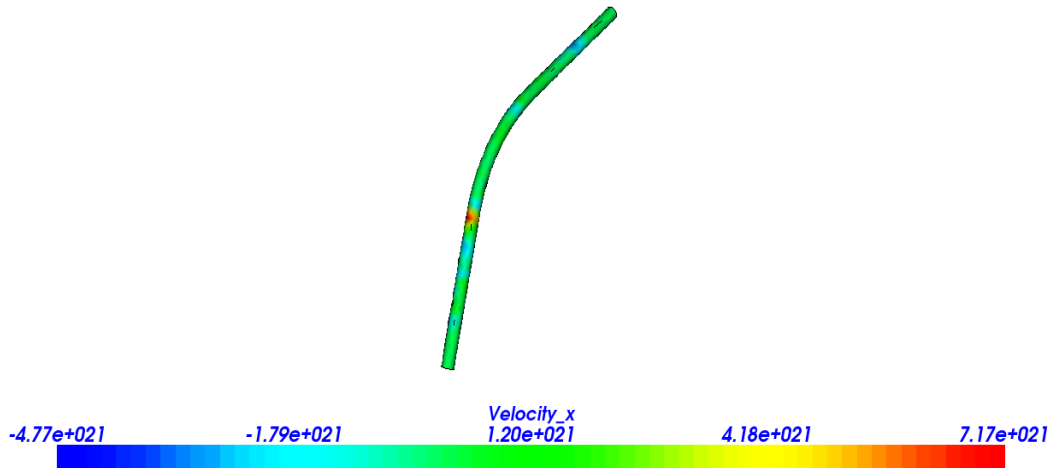


Figure 27.1.3-27: قيم تغير السرعة

هذا الشكل من قيم السرعة يدل على أن اللون الأزرق يحدد قيمة الحد الأدنى من سرعة. ثم تزيد القيمة لتصل إلى الحد الأقصى في اللون الأحمر.

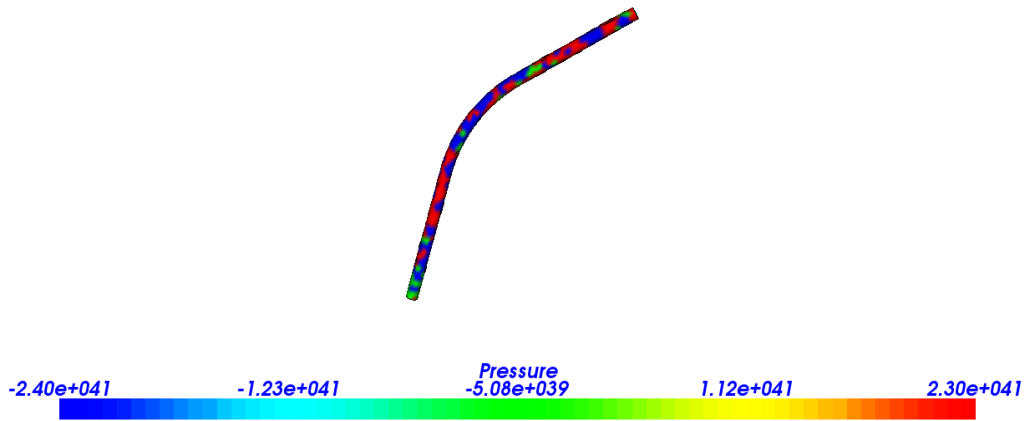


Figure 27.1.3-28: قيم تغير الضغط

هذا الشكل من قيم الضغط يدل على أن الضغط هو الحد الأدنى في اللون الأزرق أيضا، ويزيد حتى يصل إلى القيمة القصوى في اللون الأحمر.

لذلك علينا دراسة الأماكن ذات اللون الأخضر الأصفر، والأحمر في السرعة والضغط لمعرفة مكان الضعف في التصميم

على سبيل المثال نرى تغير السرعة في :
الركن:

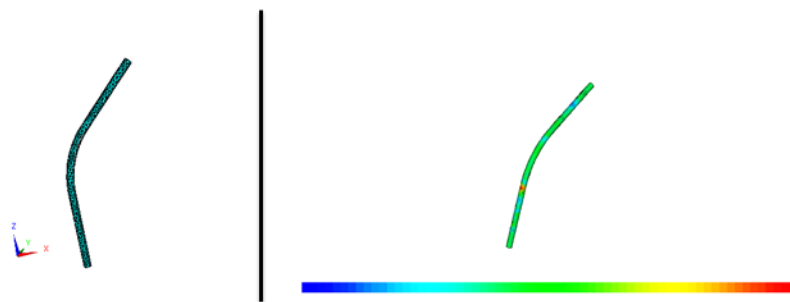


Figure 27.1.3-29: تغير السرعة في الركن

الأنابيب:

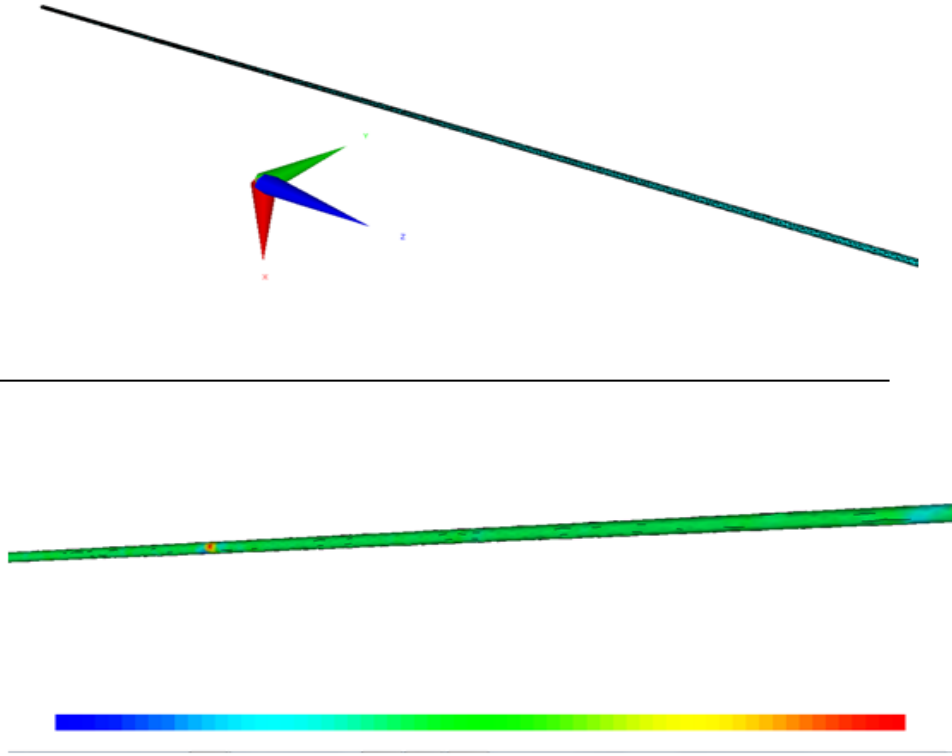


Figure 27.1.3-30: تغير السرعة في الأنبوب

مسار المياه:

المسار الأول عندما تسير المياه من خزان الضغط الى الأنابيب ثم إلى الأنبوب الذي تصب فيه الأنابيب:

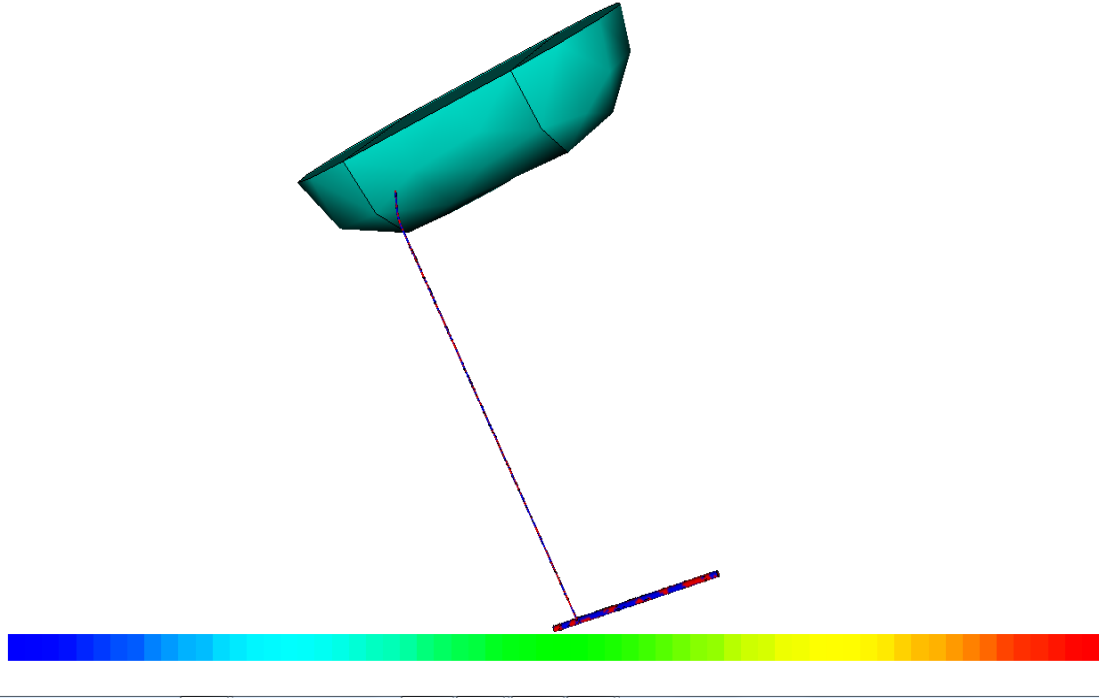


Figure 27.1.3-31: قيم السرعة في المسار الأول

المسار الثاني هو المسار العكسي أي من المصب إلى خزان الضغط.

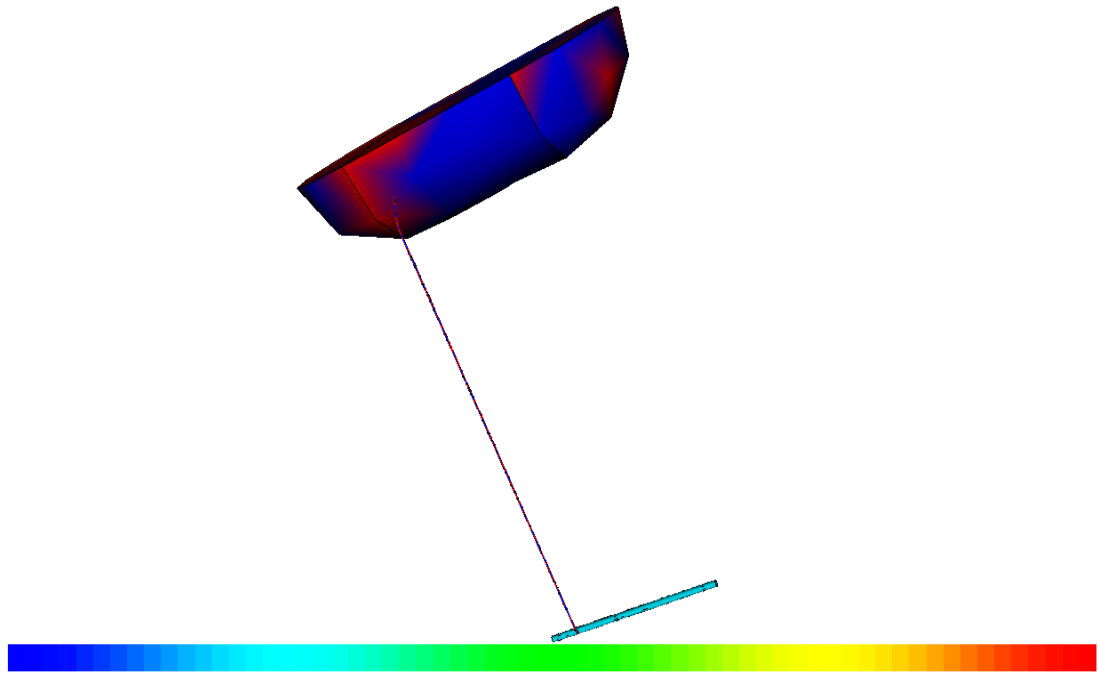


Figure 27.1.3-32: قيم السرعة في المسار الثاني

يمكننا أن نستنتج أن السرعة هي القصوى في الركن، وفي خزان الضغط عند ارتفاع منسوب المياه، وفي المصب .
لذلك علينا رعاية المواد عندما نقوم بتصميم محطة توليد الكهرباء.
الآن ننتقل إلى قيم الضغط .

في الركن:

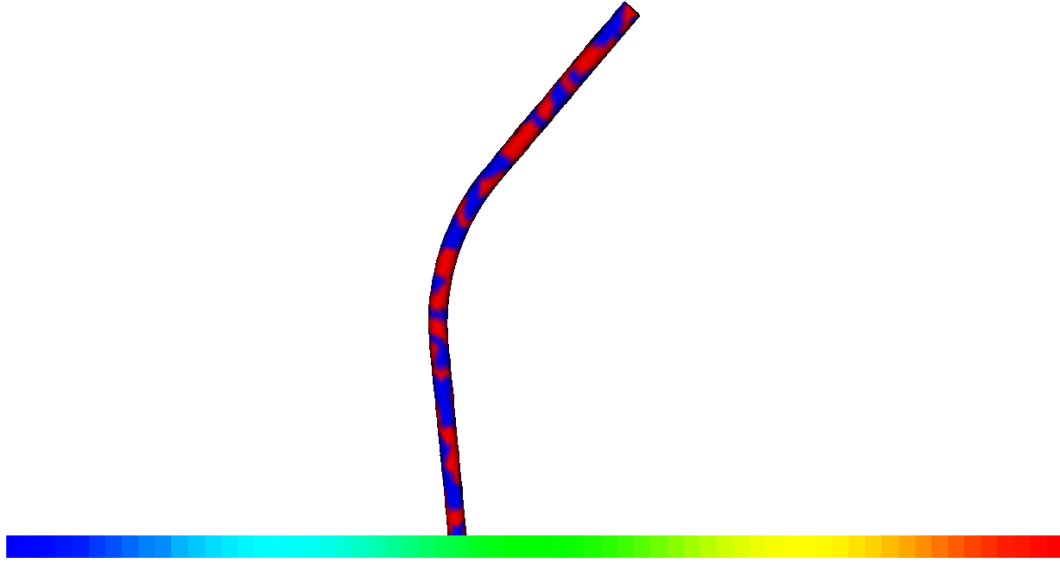


Figure 27.1.3-33: تغير الضغط في الركن

في الأنابيب:

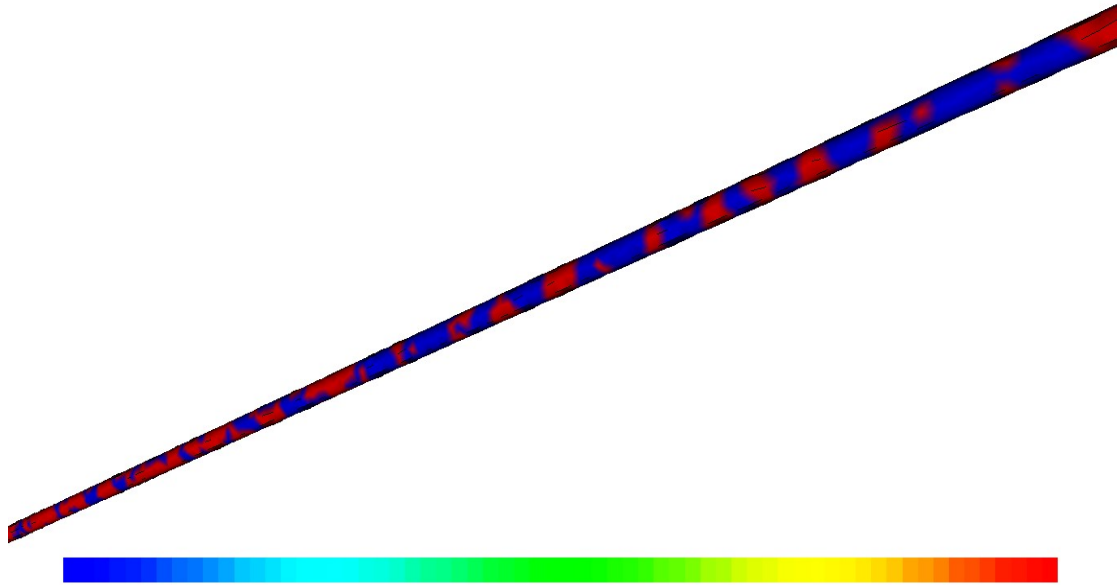


Figure 27.1.3-34: تغير الضغط في المسار الأول:

مسار المياه:

المسار الأول عندما تسير المياه من خزان الضغط في الأنابيب ثم إلى الأنبوب الذي يجمع الأنابيب (المصب):

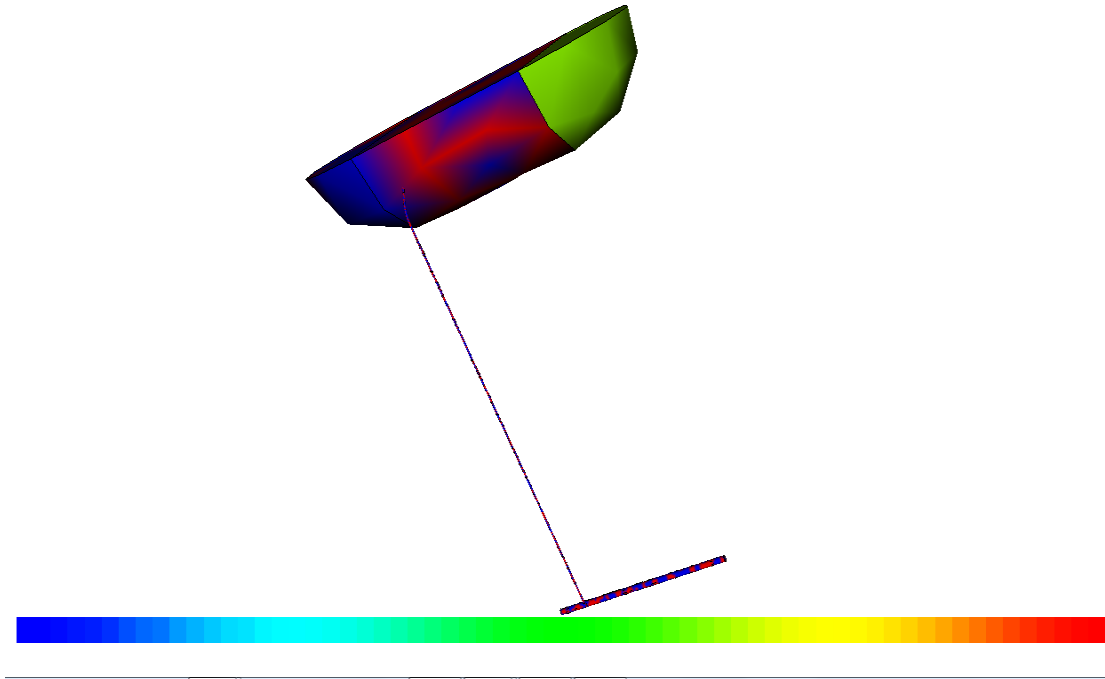


Figure 27.1.3-35: تغير الضغط في المسار الأول

المسار الثاني عندما تسير المياه من المصب إلى الأنبوب ثم إلى خزان الضغط:

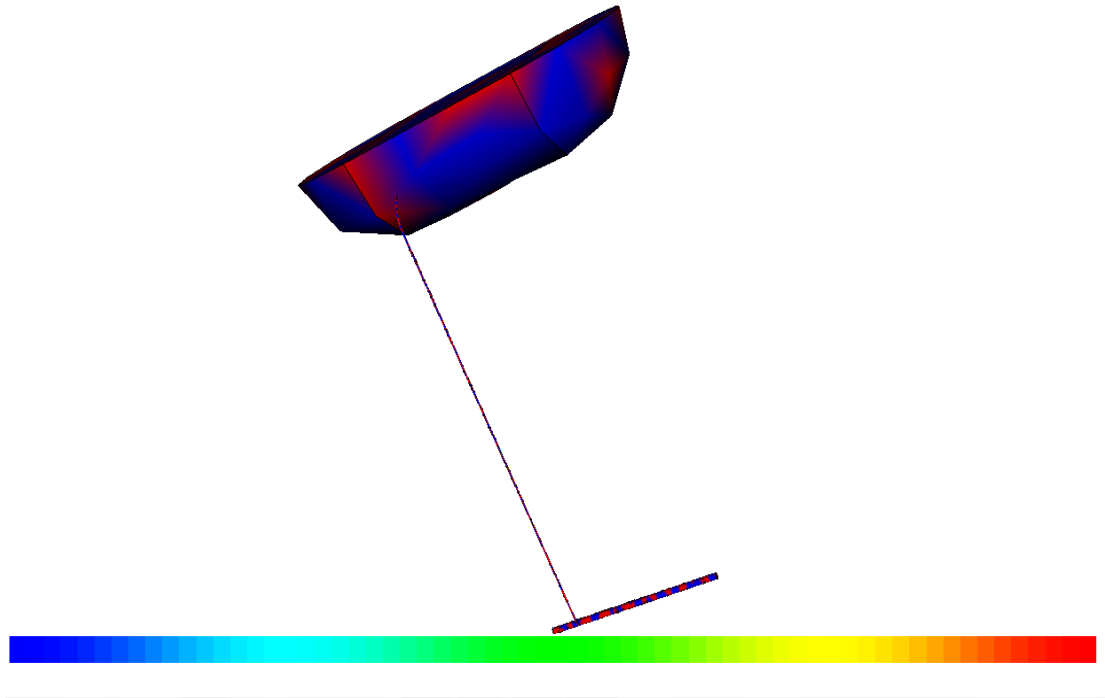


Figure 27.1.3-36: تغير الضغط في المسار الثاني

يمكننا أن نستنتج أن الضغط مرتفع في كل جزء من التصميم؛ لذلك علينا اختيار مواد متينة قادرة على تحمل درجات عالية من الضغط في محطة توليد الكهرباء .

من المهم أن نقول أن الملفات التي تتضمن معلومات التصميم (سرعة والقيم الضغط) نجدها على الموقع

<http://www.aecenar.com/publications>

27.1.4 مراجع

- Introduction to Finite Element Analysis (FEA) or Finite Element Method (FEM)
- **Finite Element Analysis**
(MCEN 4173/5173)
Fall, 2006
Instructor: Dr. H. "Jerry" Qi

27.2 انشاء برنامج لتحليل مسألة ما في ميدان ديناميكيات الموائع الحاسوبية (د.م.ج.).

نبدأ بكتابة المعادلات التي تحكم المسألة المطروحة، ثم نوجد الحلول العددية لهذه المعادلات .

نكتب برنامج C++ (باستعمال المكتبة الموجودة في برنامج OpenFOAM) وندخل الرموز إلى الجهاز وهكذا يعمل البرنامج جيداً. ويمكن استخدام البرنامج المفتوح OpenFoam لهذا الغرض.

27.2.1 تحسيب السريان في زاوية باستخدام OpenFOAM

علينا إدراج البرنامج في قائمة البدلاء في FreeCAD باستخدام رموز OpenFOAM.

أولاً؛ يجب أن نعلم رموز OpenFOAM في Linux. الاوامر الرئيسية مصنفة في هذا الجدول:

Command	Description
cd	Changes Directory to dirname
cp	Copy source file into destination
mkdir	Create a new directory dirname
mv	Move (Rename) an oldname to newname.
pwd	Print current working directory.
rm	Remove (Delete) filename
rmdir	Delete an existing directory provided it is empty.
vi	Opens vi text editor

Table 27.2.1-1: Linux الأوامر في نظام التشغيل

نكتب البرنامج على OpenFOAM عندما نقوم بتشغيله نحصل على النتائج التالية:

```

ogin as: meae
meae@192.168.1.1's password:
ast login: Sat Mar 28 13:38:24 2015 from 192.168.1.2
meae@server ~]$ pwd
/home/meae
meae@server ~]$ cd ..
meae@server home]$ ll
total 36
-rwx----- 2 bkerdi bkerdi 4096 Mar 29 12:45 bkerdi
-rwx----- 14 fchaar fchaar 4096 Mar 29 17:21 fchaar
-rwx----- 2 fhamed fhamed 4096 Apr 14 10:29 fhamed
-rwx----- 32 iap iap 4096 Apr 11 12:20 iap
-rwxr-xr-x 31 meae meae 4096 Apr 14 11:12 meae
-rwx----- 17 megbi megbi 4096 Mar 27 10:02 megbi
meae@server home]$ meae
bash: meae: command not found
meae@server home]$ cd meae
meae@server ~]$ ll
total 75488
-rwxr-xr-x 1 root root 28377109 May 15 2013 150513TEMO_last_-STPP_Report4_en
i_arab.pdf
-rwxr-xr-x 12 meae meae 4096 Dec 31 2013 Central_Library
-rwxr-xr-x 2 megbi megbi 4096 Apr 3 2010 Desktop
-rwxr-xr-x 1 meae meae 57782 Apr 23 2010 IAP-Logo.JPG
-rwxrwxrwx 1 meae meae 64 Aug 1 2014 link to scilab -> /home/meae/sci
slab-x11-4.3-1.e15.i386.rpm_FILES/usr/bin/scilab
-rwxr-xr-x 1 meae meae 20456732 Feb 21 2010 martin_liu_dissertation_num_bren
kammer.pdf
-rwxr-xr-x 5 meae meae 4096 Jul 14 2010 OpenFOAM
-rwxrwxr-x 2 meae meae 4096 Jul 18 2011 pluto
-rwxr-xr-x 1 meae meae 386195 May 14 2010 promotion1_fzk
-rwxr-xr-x 1 meae meae 19379104 Jan 25 2014 scicoslab-x11-4.3-1.e15.i386.rpm
-rwxr-xr-x 3 meae meae 4096 Jan 25 2014 scicoslab-x11-4.3-1.e15.i386.rpm
FILES
-rwxrwxr-x 2 meae meae 4096 Aug 1 2014 spa.environ
-rwxrwxr-x 4 meae meae 4096 Jul 17 2011 tools
-rwxrwxr-x 2 meae meae 4096 Jun 21 2010 uebung
-rwx----- 5 meae meae 4096 Jan 25 2014 usr
-rwxrwxr-x 15 meae meae 4096 Jul 18 2011 xemacs-21.4.20
-rwxr-xr-x 1 meae meae 8408589 Jul 13 2010 xemacs-21.4.20.tar.tar
meae@server ~]$ cd OpenFOAM/
meae@server OpenFOAM]$ ll
total 366500
-rw----- 1 meae meae 310 Jun 12 2010 Installation Notes

```

Figure 27.2.1-1:1 OpenFOAM نتایج

```
[fhamed@server meae]$ ll
total 75488
-rwxr-xr-x 1 root root 28377109 May 15 2013 150513TEMO_last_-STPP_Report4_en
gl_arab.pdf
drwxr-xr-x 12 meae meae 4096 Dec 31 2013 Central_Library
drwxr-xr-x 2 megbi megbi 4096 Apr 3 2010 Desktop
-rwxr-xr-x 1 meae meae 57782 Apr 23 2010 IAP-Logo.JPG
lrwxrwxrwx 1 meae meae 64 Aug 1 2014 link to scilab -> /home/meae/sci
coslab-x11-4.3-1.el5.i386.rpm FILES/usr/bin/scilab
-rwxr-xr-x 1 meae meae 20456732 Feb 21 2010 martin_liu_dissertation_num_bren
nkammer.pdf
drwxr-xr-x 5 meae meae 4096 Jul 14 2010 OpenFOAM
drwxrwxr-x 2 meae meae 4096 Jul 18 2011 pluto
-rwxr-xr-x 1 meae meae 386195 May 14 2010 promotion1_fzk
-rwxr-xr-x 1 meae meae 19379104 Jan 25 2014 scicoslab-x11-4.3-1.el5.i386.rpm
drwxr-xr-x 3 meae meae 4096 Jan 25 2014 scicoslab-x11-4.3-1.el5.i386.rpm
_FILES
drwxrwxr-x 2 meae meae 4096 Aug 1 2014 spa.environ
drwxrwxr-x 4 meae meae 4096 Jul 17 2011 tools
drwxrwxr-x 2 meae meae 4096 Jun 21 2010 uebung
drwx----- 5 meae meae 4096 Jan 25 2014 usr
drwxrwxr-x 15 meae meae 4096 Jul 18 2011 xemacs-21.4.20
-rwxr-xr-x 1 meae meae 8408589 Jul 13 2010 xemacs-21.4.20.tar.tar
[fhamed@server meae]$ cd OpenFOAM/
[fhamed@server OpenFOAM]$ ll
total 366500
-rw----- 1 meae meae 310 Jun 12 2010 Installation Notes
-rw----- 1 meae meae 0 Jun 12 2010 Installation Notes~
drwxrwxr-x 3 meae meae 4096 Jul 14 2010 meae-1.6
drwxrwxr-x 11 meae meae 4096 Sep 24 2010 OpenFOAM-1.6
-rwxr-xr-x 1 meae meae 241760751 Jun 9 2010 OpenFOAM-1.6.General.gtgz
drwxrwxr-x 15 meae meae 4096 Jun 9 2010 ThirdParty-1.6
-rwxr-xr-x 1 meae meae 133110883 Jun 9 2010 ThirdParty-1.6.General.gtgz
[fhamed@server OpenFOAM]$ cd OpenFOAM-1.6
[fhamed@server OpenFOAM-1.6]$ ll
total 81260
-rwxr-x--- 1 meae meae 366 Jul 24 2009 Allwmake
drwxrwxr-x 6 meae meae 4096 Jun 9 2010 applications
drwxrwxr-x 4 meae meae 4096 Jun 9 2010 bin
-rw-r----- 1 meae meae 17994 May 1 2008 COPYING
drwxrwxr-x 5 meae meae 4096 Jun 9 2010 doc
drwxrwxr-x 4 meae meae 4096 Jun 9 2010 etc
drwxrwxr-x 4 meae meae 4096 Jun 9 2010 lib
drwxrwxr-x 5 meae meae 4096 Jun 9 2010 OpenFOAM-1.6
```

Figure 27.2.1-2: 2 OpenFOAM نتائج

```
total 366500
-rw-r----- 1 meae meae      310 Jun 12 2010 Installation Notes
-rw-r----- 1 meae meae       0 Jun 12 2010 Installation Notes-
drwxrwxr-x 3 meae meae    4096 Jul 14 2010 meae-1.6
drwxrwxr-x 11 meae meae    4096 Sep 24 2010 OpenFOAM-1.6
-rwxr-xr-x 1 meae meae 241760751 Jun  9 2010 OpenFOAM-1.6.General.gtgz
drwxrwxr-x 15 meae meae    4096 Jun  9 2010 ThirdParty-1.6
-rwxr-xr-x 1 meae meae 133110883 Jun  9 2010 ThirdParty-1.6.General.gtgz
[fhamed@server OpenFOAM]$ cd OpenFOAM-1.6
[fhamed@server OpenFOAM-1.6]$ ll
total 81260
-rwxr-xr-x 1 meae meae     366 Jul 24 2009 Allwmake
drwxrwxr-x 6 meae meae    4096 Jun  9 2010 applications
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 bin
-rw-r----- 1 meae meae   17994 May  1 2008 COPYING
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 doc
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 etc
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 lib
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 OpenFOAM-1.6
-rwxr-xr-x 1 meae meae 41587474 Jun  8 2010 OpenFOAM-1.6.linuxGccDFOpt.gtgz
-rwxr-xr-x 1 meae meae 41315397 Jun  9 2010 OpenFOAM-1.6.linuxGccSFOpt.gtgz
-rw-r----- 1 meae meae     8813 Jul 27 2009 README
-rw-r----- 1 meae meae    15311 Jul 27 2009 README.html
-rw-r----- 1 meae meae    18461 Jul 27 2009 ReleaseNotes-1.6
-rw-r----- 1 meae meae    32656 Jul 27 2009 ReleaseNotes-1.6.html
drwxrwxr-x 28 meae meae    4096 Jun  9 2010 src
drwxrwxr-x 15 meae meae    4096 Jun  9 2010 tutorials
drwxrwxr-x 6 meae meae    4096 Jun  9 2010 vmake
[fhamed@server OpenFOAM-1.6]$ cd tutorials/
[fhamed@server tutorials]$ ll
total 132
-rwxr-xr-x 1 meae meae   1779 May 13 2009 Allclean
-rwxr-xr-x 1 meae meae   3011 May 13 2009 Allrun
-rwxr-xr-x 1 meae meae   5710 May 13 2009 Alltest
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 basic
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 combustion
drwxrwxr-x 10 meae meae    4096 Jun  9 2010 compressible
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 discreteMethods
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 DNS
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 electromagnetics
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 financial
drwxrwxr-x 8 meae meae    4096 Jun  9 2010 heatTransfer
drwxrwxr-x 13 meae meae    4096 Jun  9 2010 incompressible
drwxrwxr-x 6 meae meae    4096 Jun  9 2010 lagrangian
```

Figure 27.2.1-3:3 OpenFOAM نتائج

```
drwxrwxr-x 10 meae meae    4096 Jun  9 2010 multiphase
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 stressAnalysis
[fhamed@server tutorials]$ cd incompressible/
[fhamed@server incompressible]$ ll
total 88
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 boundaryFoam
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 channelFoam
drwxrwxr-x 6 meae meae    4096 Jun  9 2010 icoFoam
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 MRFSimpleFoam
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 nonNewtonianIcoFoam
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 pimpledByMFoam
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 pimpleFoam
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 pisoFoam
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 shallowWaterFoam
drwxrwxr-x 6 meae meae    4096 Jun  9 2010 simpleFoam
drwxrwxr-x 4 meae meae    4096 Jun  9 2010 simpleSRFFoam
[fhamed@server incompressible]$ cd icoFoam/
[fhamed@server icoFoam]$ ll
total 56
-rwxr-xr-x 1 meae meae     381 Feb 17 2009 Allclean
-rwxr-xr-x 1 meae meae    2797 Jul  9 2009 Allrun
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 cavity
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 cavityClipped
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 cavityGrade
drwxrwxr-x 5 meae meae    4096 Jun  9 2010 elbow
-rw-r----- 1 meae meae     160 Jul  9 2009 resetFixedWallsScr
[fhamed@server icoFoam]$ cd cavity
[fhamed@server cavity]$ ll
total 24
drwxrwxr-x 2 meae meae    4096 Jun  9 2010 0
drwxrwxr-x 3 meae meae    4096 Jun  9 2010 constant
drwxrwxr-x 2 meae meae    4096 Jun  9 2010 system
[fhamed@server cavity]$ cd constant/
[fhamed@server constant]$ ll
total 16
drwxrwxr-x 2 meae meae    4096 Jun  9 2010 polyMesh
-rw-r----- 1 meae meae     917 Jul 23 2009 transportProperties
[fhamed@server constant]$ cd polyMesh/
[fhamed@server polyMesh]$ ll
total 16
-rw-r----- 1 meae meae   1346 Jul 24 2009 blockMeshDict
-rw-r----- 1 meae meae   1228 Jul 23 2009 boundary
[fhamed@server polyMesh]$ cd blockMeshDict
-bash: cd: blockMeshDict: Not a directory
```

Figure 27.2.1-4: 4 OpenFOAM نتائج

```

drwxrwxr-x 2 meae meae 4096 Jun  9 2010 system
[fname@server cavity]$ cd constant/
[fname@server constant]$ ll
total 16
drwxrwxr-x 2 meae meae 4096 Jun  9 2010 polyMesh
-rw-r----- 1 meae meae 917 Jul 23 2009 transportProperties
[fname@server constant]$ cd polyMesh/
[fname@server polyMesh]$ ll
total 16
-rw-r----- 1 meae meae 1346 Jul 24 2009 blockMeshDict
-rw-r----- 1 meae meae 1228 Jul 23 2009 boundary
[fname@server polyMesh]$ cd blockMeshDict
-bash: cd: blockMeshDict: Not a directory
[fname@server polyMesh]$ chown blockMeshDict
chown: missing operand after 'blockMeshDict'
Try 'chown --help' for more information.
[fname@server polyMesh]$ chown --help blockMeshDict
Usage: chown [OPTION]... [OWNER]][:[GROUP]] FILE...
       or: chown [OPTION]... --reference=RFILE FILE...
Change the owner and/or group of each FILE to OWNER and/or GROUP.
With --reference, change the owner and group of each FILE to those of RFILE.

-c, --changes          like verbose but report only when a change is made
--dereference         affect the referent of each symbolic link, rather
                    than the symbolic link itself (this is the default)
-b, --no-dereference  affect each symbolic link instead of any referenced
                    file (useful only on systems that can change the
                    ownership of a symlink)
--from=CURRENT_OWNER:CURRENT_GROUP
                    change the owner and/or group of each file only if
                    its current owner and/or group match those specified
                    here. Either may be omitted, in which case a match
                    is not required for the omitted attribute.
--do-preserve-root    do not treat '/' specially (the default)
--preserve-root       fail to operate recursively on '/'
-f, --silent, --quiet suppress most error messages
--reference=RFILE     use RFILE's owner and group rather than
                    the specifying OWNER:GROUP values
-R, --recursive       operate on files and directories recursively
-v, --verbose         output a diagnostic for every file processed

The following options modify how a hierarchy is traversed when the -R
option is also specified. If more than one is specified, only the final
one takes effect.

```

نتائج OpenFOAM 5: 27.2.1-5

بعد إدخال الملفات في التجويف cavity علينا تشغيل البرنامج باستخدام Allrun و بالتالي نحصل على قيم الضغط على سبيل المثال في كل نقطة رأينا قيم الضغط المرتفعة:

```

-rw-rw-rw- 1 meae meae 5137 Apr 16 09:31 p
-rw-rw-r-- 1 meae meae 10757 Apr 16 09:10 phi
-rw-rw-r-- 1 meae meae 11110 Apr 16 09:10 U
drwxrwxr-x 2 meae meae 4096 Apr 16 09:10 uniform
[meae@server 0.5]$ vi p
-0.119397
-0.1159351
-0.1174909
-0.1172752
-0.161142
-0.142923
-0.119753
-0.0927715
-0.0625719
-0.0294421
0.00637862
0.0444789
0.0840605
0.123708
0.161142
0.193091
0.215673
0.224082
0.208056
0.160638
-0.188623
-0.232236
-0.243933
-0.233942
-0.213689
-0.187149
-0.156367
-0.122495
-0.0858922
-0.0464664
-0.0040087
0.0415445
0.0898478
0.139835
0.189324
0.234695
0.271106
0.29167
0.281748
0.229781
-0.283968

```

OpenFAOM القيم التي حصلنا عليها من: 27.2.1-6

ثانياً؛ علينا أن نصور البرنامج باستخدام المصور للنتائج

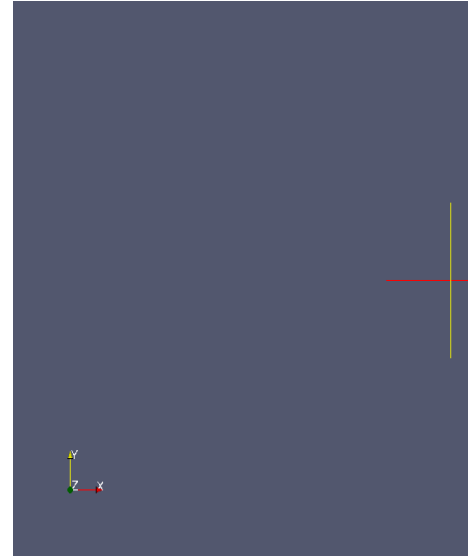


Figure 27.2.1-7: Praview في windows

ولكن مشكلتنا هي نقل البيانات من Linux OpenFOAM إلى windows paraview. علينا أن نجد صيغة لنقل البيانات.

1. نرى صيغة VTK للنقل:

```
-bash: touch.OpenFoam: command not found
[meae@server icoFoam]$ foamToVTK
-----
//
// Field      | OpenFOAM: The Open Source CFD Toolbox
// Operation  | Version: 1.6
// Author    | Web:      www.OpenFOAM.org
// Manipulation |
//
Build : 1.6-53b7f692aa41
Exec  : foamToVTK
Date  : Apr 14 2015
Time  : 12:39:53
Host  : server
PID   : 5342
Case  : /home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam
nProcs : 1
SigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).

// * * * * *
Create time

cannot open file

file: /home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/system/controlDict at line 0.

From function regIOobject::readStream()
in file db/regIOobject/regIOobjectRead.C at line 62.

FOAM exiting

[meae@server icoFoam]$ LL
-bash: LL: command not found
[meae@server icoFoam]$ ll
total 56
-rwxr-x--- 1 meae meae 381 Feb 17 2009 Allclean
-rwxr-x--- 1 meae meae 2797 Jul 9 2009 Allrun
drwxrwxr-x 5 meae meae 4096 Apr 14 10:50 cavity
drwxrwxr-x 5 meae meae 4096 Jun 9 2010 cavityClipped
drwxrwxr-x 5 meae meae 4096 Jun 9 2010 cavityGrade
drwxrwxr-x 5 meae meae 4096 Jun 9 2010 elbow
```

Figure 27.2.1-8: نتائج VTK

ثانياً؛ نعود إلى المحلل ونختار ظروف دراستنا (icoFoam → incompressible) للحصول على الضغط والسرعة وقيم phi:

ثم ننسخ الملف محاولين فتحه باستخدام paraview لكننا لم نر تجويف cavity.

نحاول الآن تشغيل OpenFOAM على windows لتصور البرنامج من OpenFOAM على windows paraview للتصور.

أولاً؛ ننقر على blockMesh في شبكة المرافق لتجزئة برنامج التجويف.

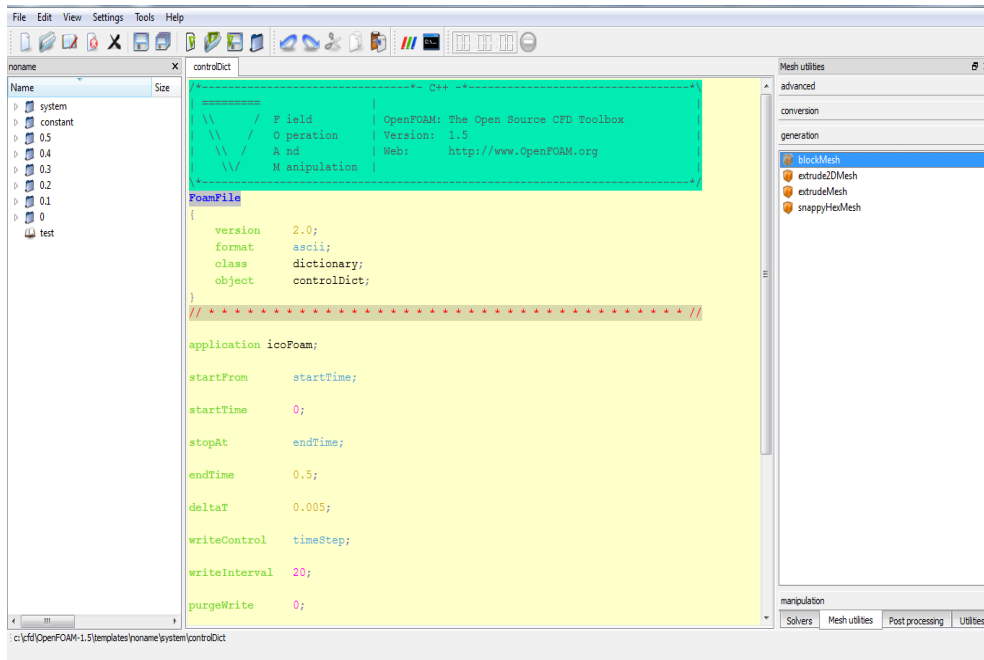


Figure 27.2.1-9: 1 OpenFOAM في نتائج windows

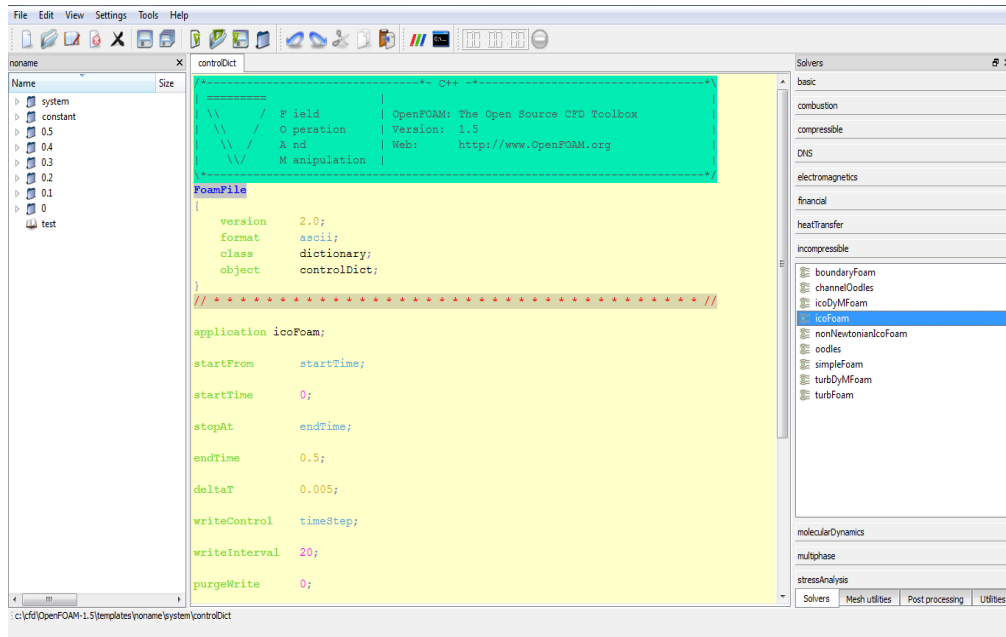


Figure 27.2.1-10:2 OpenFOAM نتائج في windows

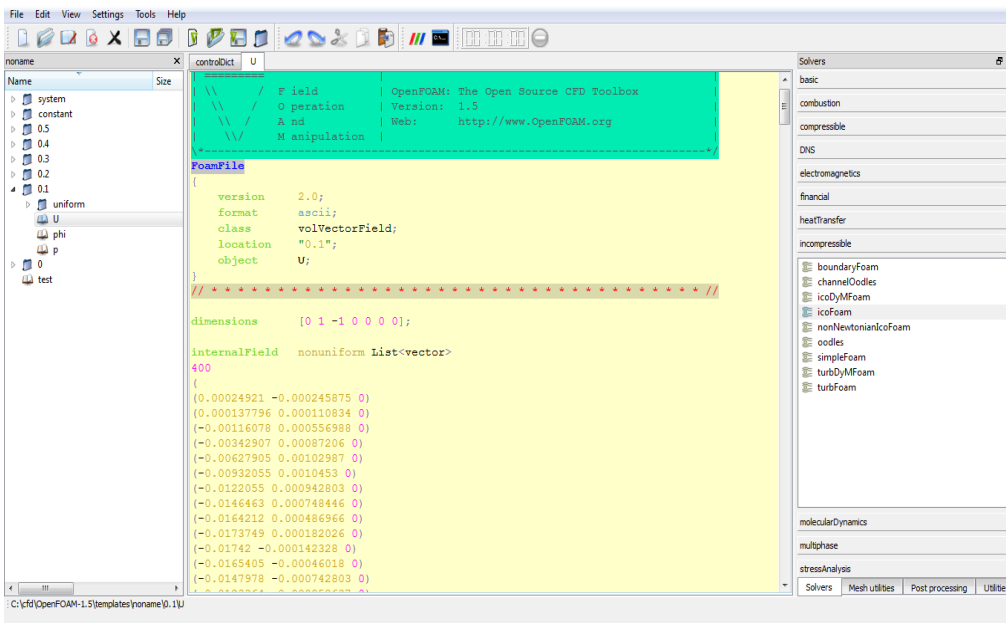


Figure 27.2.1-11: قيم السرعة عبر OpenFOAM في windows

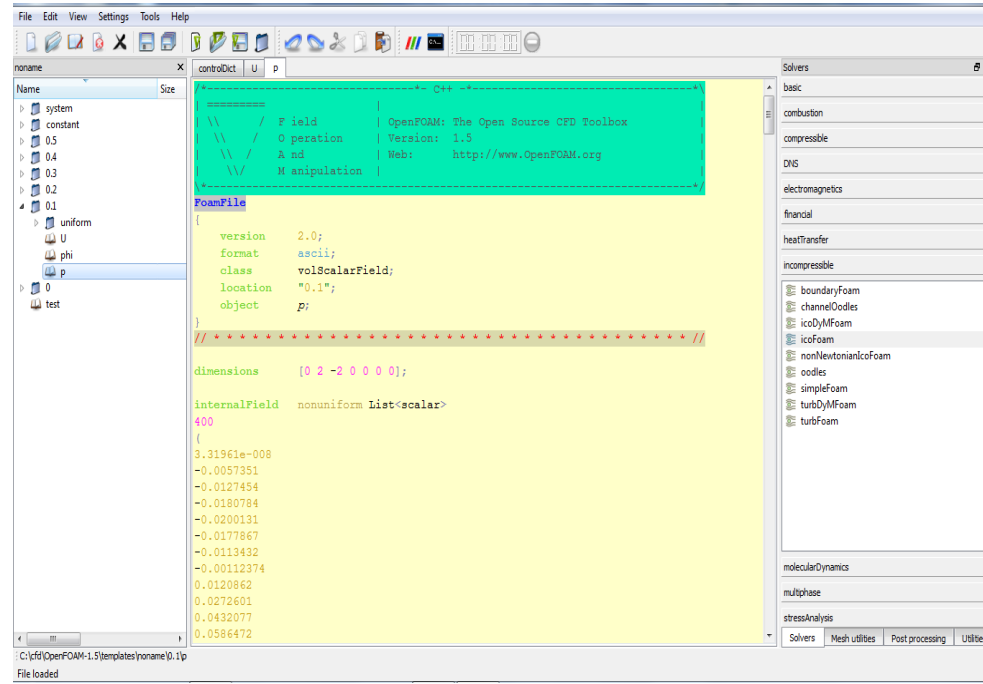


Figure 27.2.1-12: windows في OpenFOAM قيم الضغط عبر

علينا حفظ البرنامج بعد كل خطوة.

بعدها يجب أن نصور النتيجة باستخدام paraview التي تتعلق على OpenFOAM التي كتبها paraFoam.

عندما لا يستجيب paraFOAM يمكننا تصور البرنامج في paraview باستخدام foamToVTK -ascii (مع windows OpenFOAM و windows paraview):

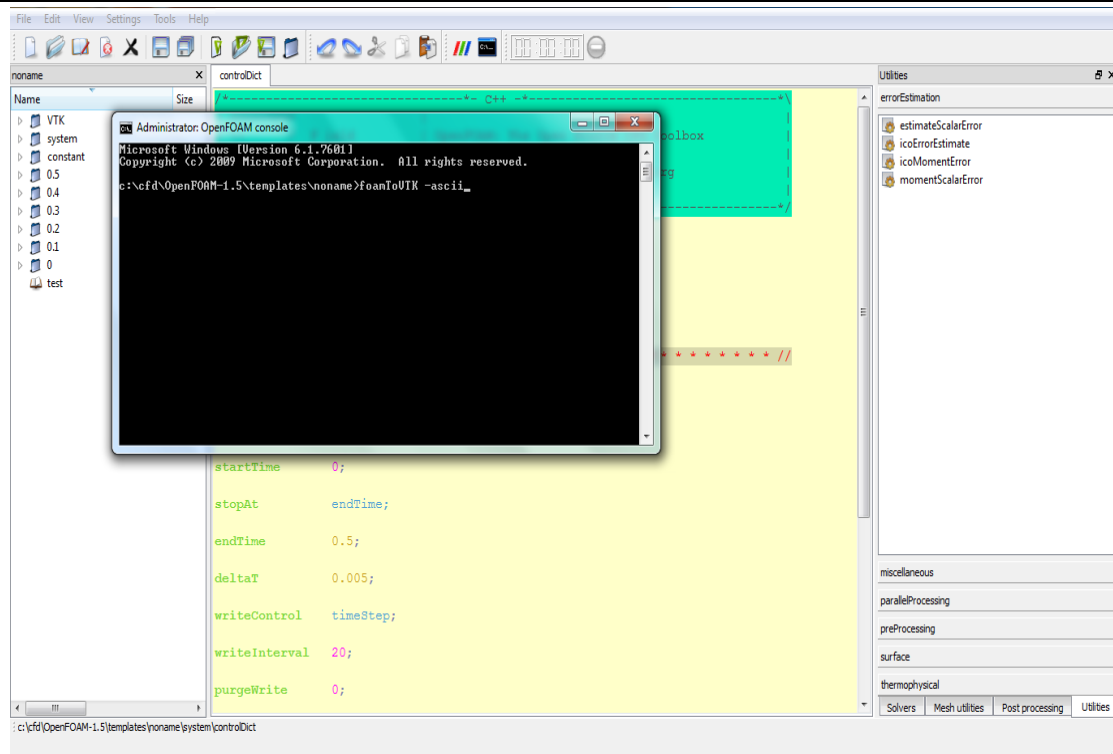


Figure 27.2.1-13: تطبيق VTK على windows

عندما نقوم بتشغيل البرنامج نحصل على:

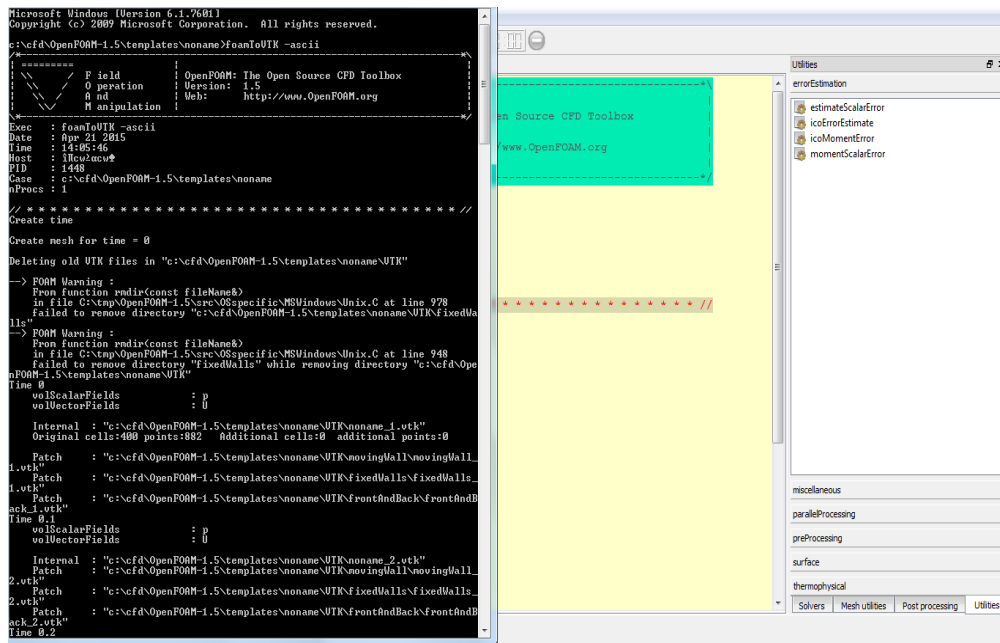


Figure 27.2.1-14: تشغيل VTK في windows 1

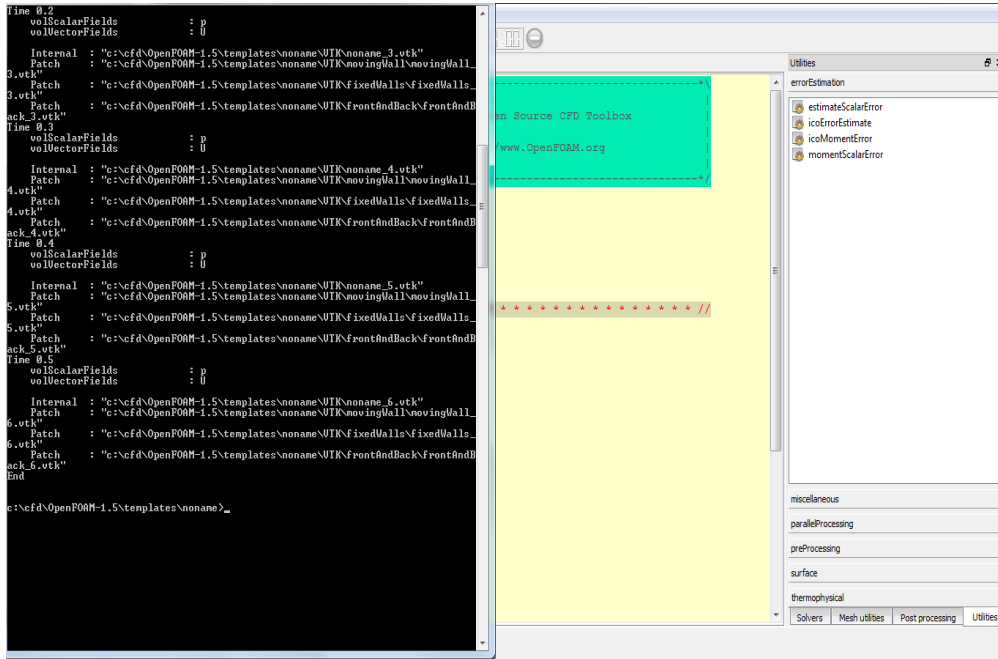


Figure 27.2.1-15: تشغيل VTK في windows 2

الآن نفتح ملف VTK في paraview للحصول على هذا الشكل: للضغط: شكل الضغط المتقطع و المتواصل

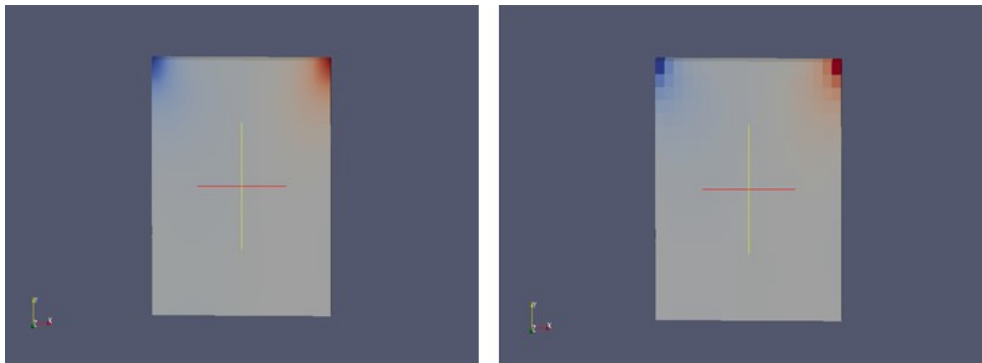


Figure 27.2.1-16: شكل الضغط المتقطع والمتواصل

للسرعة: شكل السرعة المتقطعة و المتواصلة:

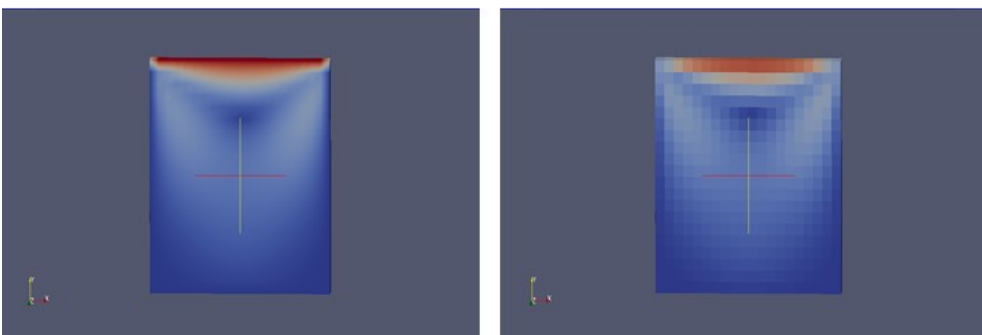


Figure 27.2.1-17: شكل السرعة المتقطعة و المتواصلة

ويمكننا أيضا تصور البرنامج تشغيل Linux OpenFOAM على windows paraview باستخدام

:foamToVTK -ascii

```
(mease@server cavity)$ foamToVTK -ascii
-----*
|=====|
| \\ / F ield | OpenFOAM: The Open Source CFD Toolbox |
| \\ / O peration | Version: 1.6 |
| \\ / A nd | Web: www.OpenFOAM.org |
| \\ / M anipulation | |
|-----*|
Build : 1.6-53b7f692aa41
Exec : foamToVTK -ascii
Date : Apr 21 2015
Time : 14:51:38
Host : server
PID : 6709
Case : /home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity
nProcs : 1
SigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).

// ***** //
Create time

Create mesh for time = 0

Deleting old VTK files in "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK"

Time: 0
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_0.vtk"
Original cells:400 points:882 Additional cells:0 additional points:0

Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_0.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_0.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_0.vtk"
Time: 0.1
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_20.vtk"
Time: 0.2
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_40.vtk"
Time: 0.3
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_60.vtk"
Time: 0.4
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_80.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_80.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_80.vtk"
```

Figure 27.2.1-18: تطبيق VTK في Linux 1

```
Time: 0.1
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_20.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_20.vtk"
Time: 0.2
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_40.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_40.vtk"
Time: 0.3
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_60.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/frontAndBack/frontAndBack_60.vtk"
Time: 0.4
volScalarFields : p
volVectorFields : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/cavity_80.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/movingWall/movingWall_80.vtk"
Patch : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFoam/cavity/VTK/fixeWalls/fixeWalls_80.vtk"
```

Figure 27.2.1-19: تطبيق VTK في Linux 2

```

Time: 0.1
volScalarFields      : p
volVectorFields      : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/cavity_20.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/movingWall/movingWall_20.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/fixedWalls/fixedWalls_20.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/frontAndBack/frontAndBack_20.vtk"
Time: 0.2
volScalarFields      : p
volVectorFields      : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/cavity_40.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/movingWall/movingWall_40.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/fixedWalls/fixedWalls_40.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/frontAndBack/frontAndBack_40.vtk"
Time: 0.3
volScalarFields      : p
volVectorFields      : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/cavity_60.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/movingWall/movingWall_60.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/fixedWalls/fixedWalls_60.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/frontAndBack/frontAndBack_60.vtk"
Time: 0.4
volScalarFields      : p
volVectorFields      : U

Internal : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/cavity_80.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/movingWall/movingWall_80.vtk"
Patch    : "/home/meae/OpenFOAM/OpenFOAM-1.6/tutorials/incompressible/icoFo
am/cavity/VTK/fixedWalls/fixedWalls_80.vtk"

```

Figure 27.2.1-20: تطبيق VTK في Linux 3

28 Numerical Combustion) لمحات عن الحرق الحسابي

من:

Peter Gerlinger, **Numerische Verbrennungssimulation** - Effiziente numerische Simulation turbulenter Verbrennung, 2008

Teil I Turbulente Strömung und Verbrennung

1	Einleitung	3
1.1	Bemerkungen zur Verbrennungssimulation	5
1.1.1	Brutto-Reaktionen und Flame-Sheet-Modell	6
1.1.2	Eddy-Breakup- und Eddy-Dissipation-Modell	6
1.1.3	Chemisches Gleichgewicht	6
1.1.4	Tabellierungstechniken	7

28.1 بعض ملاحظات بالنسبة لمحاكاة الحرق

28.1.1 (brutto reactions) و (Flame Sheet Model)

The flame-sheet model allows a complete decoupling of the modeling of the formation and destruction of species from the modeling of the flow and mixing process.

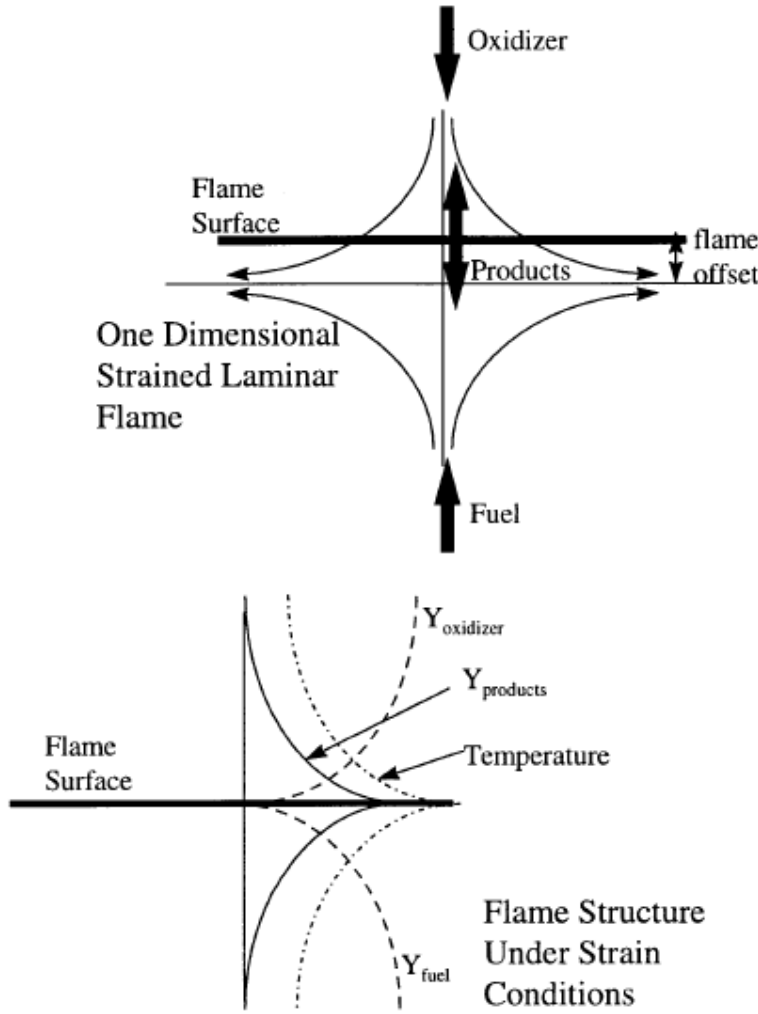


Fig. 11.1: The flame sheet model. From [Akinyemi 1997]

في نمذجة الحرق عن طريق صفحة الاله (Flame sheet model) يفترض ان الاتفاعلات الكيميائية (chemical reactions) يمكن ان يجزء في صُحف يعمل فيه الافتعال، و يفترض ايضاً ان هذه الصحف لها طخانة ضئيلة مقارنة مع امداد السريان (flow) و عملية الخلط.

28.2 اساسيات الحرق (Basics of Combustion)

From [Strauss], 111-112:

Bei der Verbrennung handelt es sich um die Hochtemperatur-Oxidation eines Brennstoffes, bei der im wesentlichen Kohlenstoff und Wasserstoff, die in verschiedener Form im Brennstoff enthalten sind, mit Sauerstoff exotherm reagieren. Eine Verbrennung heißt vollständig oder vollkommen, wenn alle brennbaren Bestandteile in ihre höchste Oxidationsstufe überführt werden. Jede Verbrennung wird durch eine Zündung eingeleitet. Unter der Zündtemperatur versteht man diejenige Temperatur, bei der mehr Wärme durch die Reaktion freigesetzt als durch Strahlung an die Umgebung abgegeben wird, so daß sich die Verbrennung von selbst erhält. Die Zündtemperatur ist im strengen Sinn kein Stoffparameter, sie wird aber als Erfahrungswert bei der Auslegung von Feuerungen und Sicherheitseinrichtungen immer wieder herangezogen. Die Zündtemperaturen der verschiedenen Brennstoffe weisen erhebliche Unterschiede auf und sind darüber hinaus abhängig von der Brennkammerbeschaffenheit sowie den

Reaktionsparametern Druck, Sauerstoffpartialdruck, der katalytischen Wirksamkeit organischer Bestandteile und der spezifischen Oberfläche des Brennstoffes.

من:

Peter Gerlinger, **Numerische Verbrennungssimulation** - Effiziente numerische Simulation turbulenter Verbrennung, 2008

2 Grundlagen der Verbrennung	11
2.1 Bilanzgleichungen reaktiver Strömungen	11
2.1.1 Wahl des Gleichungssystems	14
2.1.2 Vernachlässigung unbedeutender Terme	16
2.1.3 Kompressibilität	17
2.2 Thermodynamische Beziehung	19
2.3 Diffusiver Transport	20
2.4 Stoffwerte	23
2.4.1 Reine Stoffe	24
2.4.2 Gasmische	24
2.5 Chemische Kinetik	25
2.5.1 Chemische Umsatzraten	25
2.5.2 Reaktionsmechanismen	30

From Theroretical and Numerical Combustion (Thierry Poinso, Denis Veynante)

1 Conservation equations for reacting flows	1
1.1 General forms	1
1.1.1 Choice of primitive variables	1
1.1.2 Conservation of momentum	12
1.1.3 Conservation of mass and species	13
1.1.4 Diffusion velocities and Fick's law	13
1.1.5 Global mass conservation and correction velocity	14
1.1.6 Conservation of energy	16
1.2 Usual simplified forms	21
1.2.1 Constant pressure flames	21
1.2.2 Equal heat capacities for all species	22
1.2.3 Constant heat capacity for the mixture only	23
1.3 Summary of conservation equations	24

و هنالك المسائل التالية:

- mass transfer⁸
- معادلات الاستمرارية لسرايين تفاعلية (Conservation equations for reacting flows)
- Some Important Chemical Mechanisms (e.g. the H₂-O₂ System)⁹
- Laminar premixed flames and Laminar Diffusion flames
- Droplet Evaporation and Burning
- Introduction to Turbulent Flows
- Turbulent Premixed and Nonpremixed flames
- Burning of solids

⁸ From [Turns], pp. 83-105

⁹ From [Turns], 148-152

مراجع

Fluid Dynamics

- 1) [Ganzer 1987] Uwe Ganzer, *Gasdynamik*, Springer-Verlag 1987
- 2) [Wendt 2009] John F. Wendt, *Computational Fluid Dynamics – an Introduction (a von Karman Institute Book)*, Third Edition, 2009, Springer Verlag
- 3) [Siddiq]
[صديق] محمد هاشم الصديق (الإستاذ المشارك بشعبة هندسة الموائع قسم الهندسة الميكانيكية / كلية الهندسة والعمارة، جامعة الخرطوم، msiddiq@yahoo.com)، ميكانيك الموائع، الإصدار الثانية، 2006

Computational Fluid Dynamics

- 1) [Anderson 1991] Anderson, John D., Jr., *Fundamentals of Aerodynamics*, 2nd Edition McGraw-Hill, New York, 1991
- 2) [Ferziger, Peric] J. Ferziger und M. Peric, *Numerische Strömungsmechanik*, 2008, Springer Verlag.
- 3) [Wessling] Pieter Wesseling, *Principles of Computational Fluid Dynamics*, 2000, Springer Verlag.
- 4) http://en.wikipedia.org/wiki/Computational_fluid_dynamics

Numerical Combustion

- 1) [Strauss] K. Strauss, *Kraftwerkstechnik - zur Nutzung fossiler, nuklearer und regenerativer Energiequellen*, Springer-Verlag, 2006
- 2) [Poinsot, Veynante] Thierry Poinsot, Denis Veynante; *Theoretical and Numerical Combustion*
- 3) [Turns] Stephen R. Turns; *Introduction to Combustion – Concepts and Applications*, 2nd edition
- 4) [Akinyemi 1997] O. Akinyemi, *A flame Sheet Model of Combustion and NO Formation in Diesel Engines*, PhD thesis, MIT, June 1997

29) ملحقات Appendices(

29.1 ملحق أ: مضمون كتاب "ميكانيك الموائع" لمحمد هاشم الصديق

مضمون [صديق] محمد هاشم الصديق (الإستاذ المشارك بشعبة هندسة الموائع قسم الهندسة الميكانيكية / كلية الهندسة والعمارة، جامعة الخرطوم، msiddiq@yahoo.com)، ميكانيك الموائع، الإصدار الثانية، 2006. هو التالي:

الصفحة	العنوان	القسم	الباب
1	تعريفات أساسية		1
9	مسائل		
11	المعادلات الأساسية في ميكانيكا الموائع		2
11	متجه السران	2.1	
13	حفظ الكتلة	2.2	
16	حفظ الطاقة	2.3	
20	حفظ كمية التحرك	2.4	
24	مسائل		
27	التحليل البعدي والتمدج		3
27	أسس التحليل البعدي	3.1	
31	بعض المقادير اللابعدية ذات الأهمية في ميكانيكا الموائع	3.2	
32	التمدج	3.3	
34	مسائل		
35	السران اللا إنضغاطي في الأنابيب		4
35	أثر الاحتكاك على السران في الأنابيب	4.1	
41	القوا قد الموضعية في الأنابيب	4.2	
44	الأنابيب المتفرعة	4.3	
47	مسائل		
49	ميكانيكا الموائع عند الاتزان النسبي		5
49	المعادلة الأساسية	5.1	
50	توزيع الضغط في مجال تنائي الأبعاد لسائل في حاوية تتحرك بتسارع ثابت	5.2	
54	توزيع الضغط في سائل ساكن	5.4	
56	الطفو	5.5	
59	الهيدرومتر	5.6	
61	استقرار الأجسام الطافية	5.8	
64	مسائل		
66	طرق القياس		6
66	مقدمة	6.1	
67	أجهزة قياس الضغط	6.2	
71	أجهزة قياس معدل السران	6.3	
75	الدفع		7
75	الدفع النفاث	7.1	
78	الدفع الصاروخي	7.2	
79	الدفاع	7.3	
86	طرق الدفع النفاث	7.4	
87	مسائل		

88	حفظ كمية التحرك في الصورة التفاضلية		8
88	الصورة العامة للمعادلات	8.1	
90	حالات خاصة	8.2	
91	حل معادلات نافير - ستوكس	8.3	
101	تحسين حركة الموائع	8.4	
103	مسائل		
105	الإعاقة		9
105	مقدمة	9.1	
105	معادلات الطبقة الجدارية	9.2	
109	حل فون-كارمن عند ممال الضغط صفر	9.3	
120	الطبقة الجدارية بممال ضغط لا صفري	9.4	
122	الفصل و الإعاقة الضغطية في السريان الخارجي	9.5	
128	التحكم في الطبقة الجدارية	9.6	
132	مسائل		
134	الرفع		10
134	مقدمة	10.1	
142	إختزال معادلات نافير - ستوكس لحالة السريان الللزجي	10.2	
146	السريان اللادوراني عبر اسطوانة	10.3	
155	الرفع على الجنيح	10.4	
160	مسائل		
162	السريان الانضغاطي للغاز		11
163	مقدمة	11.1	
166	حركة الموجات الصوتية	11.2	
172	السريان اللاتديدي	11.3	
192	مسائل		
194	الصدمة المتعامدة	11.4	
208	مسائل		
209	السريان الاحتكاكي	11.5	
224	مسائل		
225	السريان اللاكظمي	11.6	
234	مسائل		
235	قياس السرعة في السريان الانضغاطي	11.7	

239	فوائيم خواص الماء و الجو القياسي	الملحق أ
240	بعض العلاقات الرياضية ذات الصلة	الملحق ب
241	معامل الاحتكاك f للأنابيب	الملحق ج
245	فوائيم السريان الانضغاطي للهواء	الملحق د
252		الرموز
254		مراجع
256		معجم

29.2 ملحق ب: مضمون كتاب [Ferziger, Peric]

مدخل الى التحليل العددي (بالإنجليزية: Numerics)

(بالإنجليزية: Components of a numerical method)

(بالإنجليزية: Mathematical model)

(Discretization method: بالإنجليزية)

(Coordinate and base vector systems: بالإنجليزية)

(Numerical mesh: بالإنجليزية)

(Finite Approximations: بالإنجليزية)

(Solution method: بالإنجليزية)

(Convergence criteria: بالإنجليزية)

اساسيات ديناميك الحرارة (بالإنجليزية: Thermodynamics)

(Finite Difference Methods: بالإنجليزية)

(Finite Volume Methods: بالإنجليزية)

طريقة العناصر المنتهية (FEM)

(Solving linear equation systems: بالإنجليزية)

(Solving the Navier-Stokes Equations: بالإنجليزية)

(Computation Methods for complex flow areas: بالإنجليزية)

(Simulation of turbulence: بالإنجليزية)

(Compressible Fluids: بالإنجليزية)

(Efficiency and accuracy: بالإنجليزية)

(Special Topics: بالإنجليزية)

(Combustion: بالإنجليزية)

30 Dictionary engl.-arabic(قاموس انجليزي - عربي)

A

absolute	مطلق
absolute pressure	ضغط مطلق
accuracy	دقة
acceleration	تسارع
adiabatic	كظيم
aerofoil, airfoil	جنح
algebraic difference quotients	فُرُق لمقسومات الجبرية
angle of attack	زاوية الهجوم
apparent turbulent stress	اجهاد موري ظاهري
Apparent viscosity	لزوجة ظاهرية
aspect ratio	نسبة باعية
atmosphere	جو

B

back pressure	ضغط نهائي
barometer	بارومتر
bearing	محمل
Bernouli	برنولي

boundary	الحدود
Blasius	بلازيوس
Boundary conditions	الشروط الحدودية
body force	قوة جسمية
boundary layer	طبقة جدارية
Bourdon	بوردون
Bourdon gauge	مضغاط بوردون
Boussinesq	بوسينيسك
branched pipes	أنابيب تتفرعة
Buckingham theorem	نظرية بكنغهام
buoyancy	طفو
Buoyancy center	مركز الطفو

C

calculation	حساب
characteristic lines	الخطوط المميزة
chemical reactions	تفاعلات كيميائية
choked	شرق
chord	وتر
circulation	تدوير
Colebrooke-White	كلبروك-وايت

Combined boundary layer	طبقة جدارية هجين
combustion chamber	غرفة احتراق
compressible	انضغاطي
compressor	ضاغط
Computational fluid dynamics	حركية الموائع التحسببية
conservation laws	قوانين الحفظ
constriction flow meters	مقاييس السريان الزامة
container	حاوية
continuity	الاستمرارية
continuum	كمية متصلة
convective	حملية
convergent	لام
convergent-divergent	لام-ناشر
configure	تكوين
conservation form	الشكل التحفظي
continuity equation	معادلة الاستمرارية
control volume	حجم التحكم
coordinate system	نظام إحداثي

corner	زاوية
correction factor	معامل تصحيح
couette flow	سريان كوييت
couple	مزدوج
critical	حرج
crude oil	نפט خام
current	تيار
cylinder	أسطوانة

D

Darcy formula	معادلة دارسي
dependent variables	والمتغيرات التابعة
density	كثافة
derivate	المتفرعة
derivative	المشتق
determinant	المحددة
diffuser, divergent	ناشر
differential	تفاضلي
difference	الفرق
difference expressions	تعايير الفروق
difference quotients	مقسومات فرقية

dimensional	بُعدي
dimensional analysis	تحليل بعدي
dimensionless	لا بعدي
discretization	تجزئة - تفريز
discriminant	المتميّز
displacement	ازاحة
distinct	متميز
divergence theorem	نظرية التباعد
downstream	سفلي
drag	اعاقة
duct	مجرى
dynamic	حركي
dynamic pressure	ضغط حركي
dynamic similarity	تشابه حركي
dynamic viscosity	لزوجة حركية

E

efficiency	كفاءة
elbow	كوع
Elliptic (partial differential) equations	معادلات القطع الناقص

ellipse	الاهليج
energy	طاقة
enthalpy	محتوى حراري
entropy	تبديد
equilibrium	توازن
error	خطأ
exit pressure	ضغط خروجي
expansion waves	موجات تمديدية
explicit	صريح

F

Fanno flow	سريان فانو
flow coefficient	معامل السريان
flow	سريان
flow field	مجال السريان
finite-difference methods	طرق الفرق المحدود
finite element	أعضاء محددة
finite volumes	أحجام محددة
finite differences	فروق محددة
fluid dynamics	حركية الموائع
fluid	الموائع
flux	سريان

flux vector	متجه السريان
formal	شكلي
forward difference	الفرق إلى الأمام
free path	مسار حر
friction	احتكاك
friction factor, ϕ	معامل الاحتكاك ϕ
Froude number	عدد فرود
Froude theorem	نظرية فرود
function of	دالة ل

G

gas	غاز
gauge	قياسي
gauge pressure	ضغط قياسي
geometric similarity	تشابه هندسي
generate	انشاء
gradient	ممال
governing equation	معادلة اساسية
grid	شبكة

H

head	سمت
------	-----

heat transfer	انتقال حرارة
helicopter	حواطة
hydraulic diameter	قطر سريان
hydrodynamic lubrication	تزييق حركي
hydrometer	مقياس الكثافة
hyperbolic	مغرق
hyperbolic (partial differential) equations	معادلات القطع الزائد

I

ideal	مثالي
incorporate	دمج
incompressible	لا انضغاطي
inequality	متباينة
infinitesimal	موحل في الصغر
initial conditions	الشروط الاولية
insulated	معزول
integral	تكامللي
internal energy	طاقة داخلية
inviscous, inviscid	لا لزجي
isentropic	لا تبديدي
isotropic	لا اتجاهي

iterative	تكرري
irrotational	لا دوراني
installation	تثبيت
integral form	شكل لا يتجزأ
integral	تكاملي
inviscid	لا لزجي

J

jet	نفث
jet engine	محرك نفث

K

kinematic viscosity	لزوجة كينماتية
Kutta - Joukowski	كوتا-يوكوفسكي

L

laminar	صفائحي
Laplace equations	معادلات لابلاس
lift	رفع
linear algebra	علم الحساب الجبر الخطي
loss	فقد
lubricant	مزلق
lubrication	تزييق

M

Mach angle	زاوية ماخ
Mach cone	مخروط ماخ
Mach number	عدد ماخ
manipulation	تلاعب
manometer	مضغاط سائلي
mass	كتلة
mass flow rate	معدل سريان كتلي
matrix	مصفوفة
metacentre	مركز التارجح
metacentric radius	نصف القطر التارجحي
minor losses	فواقد موضعية
model	نموذج
modeling	نمذجية
molecular	جزيئي
moment	العزم
momentum	كمية التحرك

N

Navier-Stokes equations	معادلات نافير ستوكس
Newtonian flow	سريان نيوتوني
non-Newtonian flow	سريان لا نيوتوني
normal	عمودية

normal shock	صدمة متعامدة
nozzle	منفث
numerical methods	طرق عددية
numerical analysis	التحليل العددي

O

oblique shock	صدمة مائلة
one dimensional	أحادي البعد
orifice	فوهة
order of magnitude	القيمة الأسية

P

Panel	مؤطرة
parabolic	قطعي مكافئ
parabolic (partial differential) equations	معادلات القطع المكافئ
paraboloid	المقطع المكافئ
partial derivate	المشتق الجزئي
partial differential equations	المعادلات التفاضلية الجزئية
perfect gas	غاز كامل
physical similarity	تشابه فيزيائي
pipe	أنبوب
piston	مكبس

pitot-static tube	أنبوب الضغط الحركي
plane (e.g. xy plane)	مستو (مثلا مستو xy)
plate	لوح
polynomial equation	معادلة متعددة الحدود
Potential flow	سريان كمون
potential function	دالة كمون
power	قدرة
Prandtl	براندل
pressure	ضغط
pressure drag	اعاقة ضغطية
propeller	دفاع
property	خاصية
propulsion	دفع
prototype	طراز بدائي
pump	مضخة

R

ramjet	نفث تضاعطي
Rayleigh flow	سريان ريلي
(chemical) reaction	تفاعل كيميائي
Rectangular	مستطيلي

reference point	نقطة مرجعية
relative density	كثافة نسبية
relative mass	كتلة نسبية
relative weighing	موازنة حدية
reservoir	مستودع
Reynolds number	عدد رينولز
Reynolds stresses	اجهاد رينولز
rocket	صاروخ
rotation	دوران
roughness height	ارتفاع الخشونة

S

scalar	مقداري
sensitivity	حساسية
separation	فصل
shear	قص
shear stress	الإجهاد القصي
SI units	النظام العالمي للوحدات
similarity	تشابه
simulation	محاكاة
sink	مصب
skin drag	اعاقة جلدية

skin-drag coefficient	معامل الاعاقة الجلدية
slope	ميل
sonic	صوتي
source	منبع
span	باع
specific heat	حرارة نوعية
specific-heat ratio	نسبة الحرارة النوعية
speed of sound	سرعة الصوت
stability	الاستقرار
stagnation	ركود
stagnation pressure	ضغط كودي
stall	انهيار
stall point	نقطة الانهيار
static	سكوني
steady	رتيب
steady-state	حالة مستقرة
stream function	دالة السريان
streamline	خط الانسياب
Stress	اجهاد
subsonic	دون صوتي
Substantial Derivate	الاشتقاق الكبير

suction	سحب
supersonic	فوق صوتي
surface force	قوة سطحية
symmetric	متماثل
System	منظومة- نظام
Symbol (Mathematical symbol)	رمز رياضي
Swamsee	سوامي

T

tangential	مماسية
term (mathematical term)	حد رياضي او جملة
Thermodynamics	الحركية الحرارية
three dimensional	ثلاثي الأبعاد
throat	حلق
thrust	دفع
time-dependend method	طريقة تعتمد الوقت
total head	سمت كلي
total-drag coefficient	معامل الاعاقة الكلي
Transient	عابر
turbine	عنفة
turbulence	مور

turbulence intensity	حدة التمور
turbulent	مائر
two dimensional	ثنائي البعد

U

uniform flow	سريان منتظم
unsteady	غير رتيب

V

vacuum	فراغ
valve	صمام
variable x	متغير x
vector operator	عامل المتجه
venturi	فنشوري
volume flow rate	معدل السريان الحجمي
Von Karman	فون كارمن
viscous, viscid	لزجي
visualization	تصوير
vortex	دوامة مائية

W

wave	موجة
------	------

انظر ايضا مجمع اللغة العربية

**MEAE-CFDNC (Computational Fluid Dynamics and Numerical Combustion) Code
(2019)**



طاقة الشمال
North Lebanon Alternative Power
www.nlap-lb.com

برنامج رقمي هدفه حساب ومحاكاة عملية الحرق و ديناميكية الموانع الحسابية (CFD)
Framework + Graphical User Interface (GUI), CFD + Numerical combustion

مختل: استجابة لتجاجة المستخدم المتكبرة والتطورات الأخيرة في مجالات ديناميكية الموائع العددية (CFD) ونسجة عملية الاحتراق (numerical combustion) التي تشمل على عدد من العمليات الفيزيائية والكيميائية المعقدة والبرنامج بشكل وثيق بقرر برمجة كود لديه القدرة على حساب مثل هذه النسجات في المحارق.. وتشمل ديناميكيات عابرة ثلاثية الأبعاد لتبخر بخاخ الوقود تتفاعل مع تدفق الغازات المتعددة المتكونة التي تمر بالاختلاط، الإشتعال، التفاعلات الكيميائية، ونقل الحرارة.

Main equations used in our program:

1. Continuity equation (mass conservation)
2. Momentum equation
3. Energy Conservation
4. Chemical equations
5. Mixture characteristics

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0$$

$$\frac{\partial}{\partial t} \rho u_j + \frac{\partial}{\partial x_i} \rho u_i u_j = - \frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho \sum_{k=1}^N Y_k f_{k,j}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i E) = \dot{Q}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$$

$$\dot{\omega}_k = \sum_{j=1}^M \dot{\omega}_{k,j} = W_k \sum_{j=1}^M \nu_{k,j} Q_j$$

$$Q_j = K_{f,j} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu_{k,j}} - K_{r,j} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu_{k,j}}$$

$$K_{f,j} = A_{f,j} T^{\beta_j} \exp\left(-\frac{E_{a,j}}{RT}\right) = A_{f,j} T^{\beta_j} \exp\left(-\frac{T_{a,j}}{T}\right)$$

$$K_{r,j} = \left(\frac{p_e}{RT}\right)^{\sum_{i=1}^N \nu_{i,j}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right)$$

$$\rho_m = (\rho_1 V_1 + \rho_2 V_2 + \dots + \rho_n V_n) / (V_1 + V_2 + \dots + V_n)$$

$$\mu_{ga} = \frac{\sum_{i=1}^N \nu_i \mu_i \sqrt{M_{g,i}}}{\sum_{i=1}^N \nu_i \sqrt{M_{g,i}}}$$

The equation below are then discretized by the finite differential method and implemented in our C++ program. The values at t+Δt are obtained from a Taylor's series expansion in time as:

$$s_i^{t+\Delta t} = s_i^t + \left(\frac{\partial s_i}{\partial t}\right) \Delta t + \left(\frac{\partial^2 s_i}{\partial t^2}\right) \frac{(\Delta t)^2}{2} + \dots$$

Classes of our program:

Our program is composed of 4 classes:

- Fuel: calculate the characteristics of species used.
- Chemical: calculate the chemical constants of the equations.
- Mix: calculate the characteristics of the mixture.
- MIXTURE: calculate density, velocity and internal energy of the mixture

Application : Hydrogen Combustion
2H2 + O2 -> 2H2O

The geometry sustainable in our program is a cube. Each side is divided to 5 parts (X, Y and Z) so it's a total of 125 points to be calculated. The fuel entrance is a square fixed at (2;0;1), (2;0;2), (3;0;1), (3;0;2) and the oxidizer entrances are the rest of the points with y=0. The points are shown in the figure beside.

Meshing

Program Input: (Characteristics)

Results presented on Paraview

Paraview support .csv files. Examples from our results are presented below. Density:

Velocity:

Program Output: (density, velocity, energy)

Maryam Abdel-Karim

@AECENAR/ May 2019

Numerical Combustion

برنامج رقمي هدفه حساب ومحاكاة عملية الحرق وديناميكية الموائع

(CFD) الحسابية

Last update: 13.05.19

Author

Maryam ABDEL-KARIM

Error! No text of specified style in document.

Content

31 Introduction مدخل

استجابة لحاجة المستخدم الكبيرة والتطورات الأخيرة في مجالات ديناميكية السوائل العددية (CFD) ونمذجة عملية الاحتراق (numerical combustion) التي تشمل على عدد من العمليات الفيزيائية والكيميائية المعقدة والمرتبطة بشكل وثيق تقرر برمجة كود لديه القدرة على حساب مثل هذه التدفقات في المحارق. وتشمل ديناميكيات عابرة ثلاثية الأبعاد لتبخير بخاخ الوقود تتفاعل مع تدفق الغازات المتعددة المكونات التي تمر بالاختلاط، الاشتعال، التفاعلات الكيميائية، ونقل الحرارة.

32 Basics¹⁰

32.1 General forms

Combustion involves multiple species reacting through multiple chemical reactions. The Navier-Stokes equations apply for such a multi-species multi-reaction gas but they require some additional terms. Species are characterized through their mass fractions Y_k for $k=1$ to N where N is the number of species in the reacting mixture. The mass fractions Y_k are defined by:

$$Y_k = \frac{m_k}{m}$$

Where m_k is the mass of species k present in a given volume V and m is the total mass of gas in this volume.

Going from non-reacting flow to combustion requires solving for $N+5$ variables instead of 5.

For a mixture of N perfect gases, total pressure is the sum of partial pressures:

$$p = \sum_{k=1}^N p_k \quad \text{where} \quad p_k = \rho_k \frac{R}{W_k} T$$

Where W_k is the atomic weight of species k . the mean molecular weight of the mixture is given by:

$$\frac{1}{W} = \sum_{k=1}^N \frac{Y_k}{W_k}$$

The enthalpy is assumed by:

$$h_k = \underbrace{\int_{T_0}^T C_{p,k} dT}_{\text{sensible}} + \underbrace{\Delta h_{f,k}^o}_{\text{chemical}}$$

$T_0=298.15$ k (reference temperature).

The formation enthalpies $\Delta h_{f,k}^o$ are the enthalpies needed to form 1 kg of species k at the reference temperature.

¹⁰ "Theoretical and Numerical Combustion" - By Thierry Poinsot, Denis Veynante

Substance	Molecular weight W_k (kg/mole)	Mass formation enthalpy $\Delta h_{f,k}^o$ (kJ/kg)	Molar formation enthalpy $\Delta h_{f,k}^{o,m}$ (kJ/mole)
CH_4	0.016	-4675	-74.8
C_3H_8	0.044	-2360	-103.8
C_8H_{18}	0.114	-1829	-208.5
CO_2	0.044	-8943	-393.5
H_2O	0.018	-13435	-241.8
O_2	0.032	0	0
H_2	0.002	0	0
N_2	0.028	0	0

Table 1.2: Formation enthalpies (gaseous substances) at $T_0 = 298.15\text{ K}$.

The heat capacities at constant pressure of species k (C_{pk}) are

$$C_{pk}^m = 3.5R \quad \text{and} \quad C_{pk} = 3.5R/W_k$$

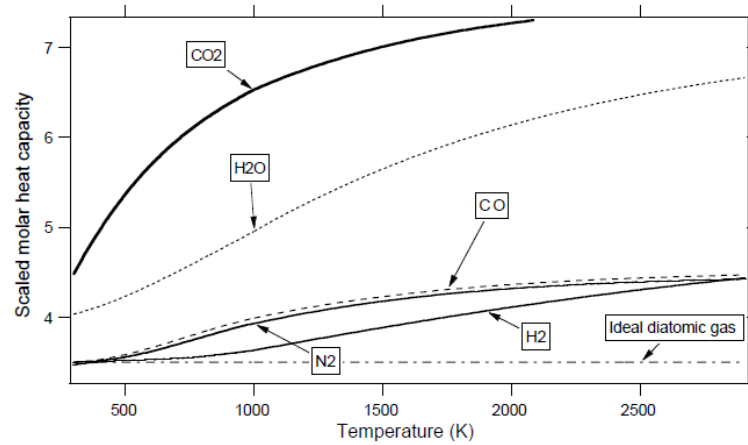


Figure 1.1: Scaled molar heat capacities at constant pressure C_{pk}^m/R of CO_2 , CO , H_2O , H_2 and N_2 .

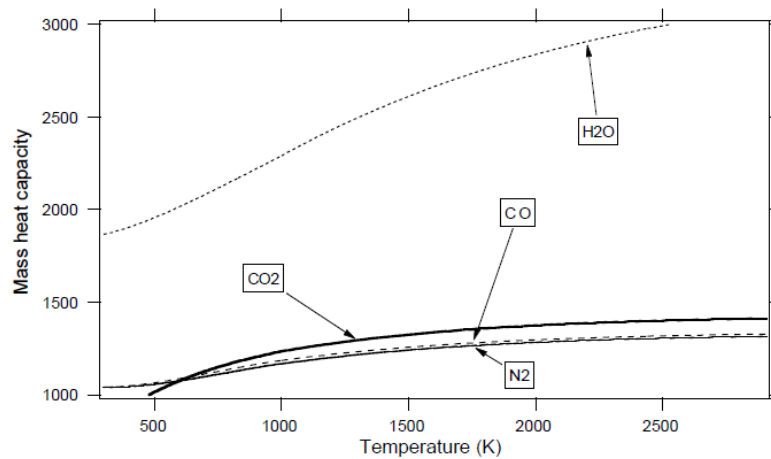


Figure 1.2: Mass heat capacities ($J/(kgK)$) at constant pressure C_{pk} of CO_2 , CO , H_2O and N_2 .

The mass heat capacities C_{vk} at constant volume are related to the C_{pk} by:

$$C_{pk} - C_{vk} = R/W_k$$

32.2 Different forms of energy equations

Form	Energy	Enthalpy
Sensible	$e_s = h_s - p/\rho = \int_{T_0}^T C_v dT - RT_0/W$	$h_s = \int_{T_0}^T C_p dT$
Sensible+Chemical	$e = h - p/\rho = e_s + \sum_{k=1}^N \Delta h_{f,k}^0 Y_k$	$h = h_s + \sum_{k=1}^N \Delta h_{f,k}^0 Y_k$
Total Chemical	$e_t = h_t - p/\rho = e_s + \sum_{k=1}^N \Delta h_{f,k}^0 Y_k + \frac{1}{2} u_i u_i$	$h_t = h_s + \sum_{k=1}^N \Delta h_{f,k}^0 Y_k + \frac{1}{2} u_i u_i$
Total non Chemical	$E = H - p/\rho = e_s + \frac{1}{2} u_i u_i$	$H = h_s + \frac{1}{2} u_i u_i$

e_t	$\rho \frac{De_t}{Dt} = -\frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$
h_t	$\rho \frac{Dh_t}{Dt} = \frac{\partial p}{\partial t} - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j} (\tau_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$
e	$\rho \frac{De}{Dt} = -\frac{\partial q_i}{\partial x_i} + \sigma_{ij} \frac{\partial u_i}{\partial x_j} + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} V_{k,i}$
h	$\rho \frac{Dh}{Dt} = \frac{Dp}{Dt} - \frac{\partial q_i}{\partial x_i} + \tau_{ij} \frac{\partial u_i}{\partial x_j} + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} V_{k,i}$
e_s	$\rho \frac{De_s}{Dt} = \dot{\omega}_T + \frac{\partial}{\partial x_i} (\lambda \frac{\partial T}{\partial x_i}) - \frac{\partial}{\partial x_i} (\rho \sum_{k=1}^N h_{s,k} Y_k V_{k,i}) + \sigma_{ij} \frac{\partial u_i}{\partial x_j} + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} V_{k,i}$
h_s	$\rho \frac{Dh_s}{Dt} = \dot{\omega}_T + \frac{Dp}{Dt} + \frac{\partial}{\partial x_i} (\lambda \frac{\partial T}{\partial x_i}) - \frac{\partial}{\partial x_i} (\rho \sum_{k=1}^N h_{s,k} Y_k V_{k,i}) + \tau_{ij} \frac{\partial u_i}{\partial x_j} + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} V_{k,i}$
E	$\rho \frac{DE}{Dt} = \dot{\omega}_T + \frac{\partial}{\partial x_i} (\lambda \frac{\partial T}{\partial x_i}) - \frac{\partial}{\partial x_i} (\rho \sum_{k=1}^N h_{s,k} Y_k V_{k,i}) + \frac{\partial}{\partial x_j} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$
H	$\rho \frac{DH}{Dt} = \dot{\omega}_T + \frac{\partial p}{\partial t} + \frac{\partial}{\partial x_i} (\lambda \frac{\partial T}{\partial x_i}) - \frac{\partial}{\partial x_i} (\rho \sum_{k=1}^N h_{s,k} Y_k V_{k,i}) + \frac{\partial}{\partial x_j} (\tau_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$

Table 1.6: Enthalpy and energy forms and corresponding balance equations. The $V_{k,i}$ are the diffusion velocities. The $f_{k,i}$'s are volume forces acting on species k in direction i . \dot{Q} is the volume source term. q_i is the enthalpy flux defined by $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^N h_k Y_k V_{k,i}$. The viscous tensors are defined by $\tau_{ij} = -2/3 \mu \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$ and $\sigma_{ij} = \tau_{ij} - p \delta_{ij}$. The heat release $\dot{\omega}_T$ is $-\sum_{k=1}^N \Delta h_{f,k}^0 \dot{\omega}_k$. For any energy or enthalpy f : $\rho \frac{Df}{Dt} = \rho (\frac{\partial f}{\partial t} + u_i \frac{\partial f}{\partial x_i}) = \frac{\partial \rho f}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i f)$.

32.3 Viscous Tensor

The velocity components are called u_i for $i=1$ to 3. The viscous tensor is defined by:

$$\tau_{ij} = -\frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

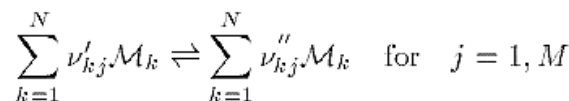
Where δ_{ij} is the Kronecker symbol $\delta_{ij}=1$ if $i=j$, 0 otherwise.

Viscous and pressure tensors are often combined into:

$$\sigma_{ij} = \tau_{ij} - p \delta_{ij} = -p \delta_{ij} - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

32.4 Chemical kinetics

Consider a chemical system of N species reacting through M reactions:



For simplicity, only mass reaction rates are used. For species k , this rate is the sum of rates produced by all M reactions:

$$\dot{\omega}_k = \sum_{j=1}^M \dot{\omega}_{kj} = W_k \sum_{j=1}^M \nu_{kj} Q_j \quad \text{with} \quad \frac{\dot{\omega}_{kj}}{W_k \nu_{kj}} = Q_j$$

The progress rate Q_j of reaction j is written:

$$Q_j = K_{fj} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu'_{kj}} - K_{rj} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu''_{kj}}$$

Where K_{fj} and K_{rj} are the forward and reverse rates of reaction j .

They are usually modeled using the empirical Arrhenius law:

$$K_{fj} = A_{fj} T^{\beta_j} \exp\left(-\frac{E_j}{RT}\right) = A_{fj} T^{\beta_j} \exp\left(-\frac{T_{aj}}{T}\right)$$

An example of a kinetic scheme for H_2 - O_2 combustion proposed in the table below. First, elements and species which have been retained for the scheme listed. For each reaction, the table then gives A_{fj} in cgs units, β_j and E_j in cal/mole. The backwards rates K_{rj} are computed from the forward rates through the equilibrium constants:

$$K_{rj} = \frac{K_{fj}}{\left(\frac{p_a}{RT}\right)^{\sum_{k=1}^N \nu_{kj}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right)}$$

```

ELEMENTS
  H  O  N
END
SPECIES
H2 O2 OH O H H2O HO2 H2O2 N N2 NO
END
REACTIONS
H2+O2=OH+OH          1.700E13    0.0    47780.
H2+OH=H2O+H          1.170E09    1.30   3626.
H+O2=OH+O             5.130E16   -0.816  16507.
O+H2=OH+H             1.800E10    1.0    8826.
H+O2+M=HO2+M         2.100E18   -1.0     0.
  H2/3.3/ O2/0./ N2/0./ H2O/21.0/
H+O2+O2=HO2+O2       6.700E19   -1.42    0.
H+O2+N2=HO2+N2       6.700E19   -1.42    0.
OH+HO2=H2O+O2        5.000E13    0.0    1000.
H+HO2=OH+OH          2.500E14    0.0    1900.
O+HO2=O2+OH          4.800E13    0.0    1000.
OH+OH=O+H2O          6.000E08    1.3     0.
H2+M=H+H+M           2.230E12    0.5   92600.
  H2/3./ H/2./ H2O/6.0/
O2+M=O+O+M           1.850E11    0.5   95560.
H+OH+M=H2O+M         7.500E23   -2.6     0.
  H2O/20.0/
HO2+H=H2+O2          2.500E13    0.0    700.
HO2+HO2=H2O2+O2      2.000E12    0.0     0.
H2O2+M=OH+OH+M       1.300E17    0.0   45500.
H2O2+H=H2+HO2        1.600E12    0.0    3800.
H2O2+OH=H2O+HO2      1.000E13    0.0    1800.
END
    
```

Table 1.4: Chemical scheme for H_2 - O_2 combustion (Miller et al.³²²). For each reaction, the table provides respectively A_{fj} (cgs units), β_j and E_j (cal/mole).

32.5 Reacting flow conservation equations

The governing conservation equations for reacting flow are shown below:

<p>Mass</p> $\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0$
<p>Species: for $k = 1$ to $N - 1$ (or N if total mass is not used)</p> <p>With diffusion velocities:</p> $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_{k,i}) Y_k) = \dot{\omega}_k$ <p>With Fick's law:</p> $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_i^c) Y_k) = \frac{\partial}{\partial x_i} (\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k \text{ and } V_i^c = \sum_{k=1}^N D_k \frac{\partial Y_k}{\partial x_i}$
<p>Momentum</p> $\frac{\partial \rho u_j}{\partial t} + \frac{\partial}{\partial x_i} \rho u_i u_j = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho \sum_{k=1}^N Y_k f_{k,j}$
<p>Energy (sum of sensible and kinetic)</p> $\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$ <p>with $\dot{\omega}_T = -\sum_{k=1}^N \Delta h_{f,k}^o \dot{\omega}_k$ and $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^N h_k Y_k V_{k,i}$</p>

Table 1.7: Conservation equations for reacting flows: the energy equation may be replaced by any of the equations given in Table 1.6. \dot{Q} is the external heat source term and f_k measures the volume forces applied on species k .

Mass $\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0$
Species: For $k = 1$ to $N - 1$ (or N if total mass is not used) With diffusion velocities: $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_{k,i})) Y_k = \dot{\omega}_k$ With Fick's law: $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_i^c) Y_k) = \frac{\partial}{\partial x_i} (\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k \text{ and } V_i^c = \sum_{k=1}^N D_k \frac{\partial Y_k}{\partial x_i}$
Momentum $\frac{\partial}{\partial t} \rho u_j + \frac{\partial}{\partial x_i} \rho u_i u_j = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i}$
Energy (sum of sensible and kinetic) $\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i} \text{ with } \dot{\omega}_T = -\sum_{k=1}^N \Delta h_{f,k}^o \dot{\omega}_k$ Or temperature $\rho C_p \frac{DT}{Dt} = \dot{\omega}'_T + \frac{\partial}{\partial x_i} (\lambda \frac{\partial T}{\partial x_i}) - \rho \frac{\partial T}{\partial x_i} \left(\sum_{k=1}^N C_{p,k} Y_k V_{k,i} \right)$ with $\dot{\omega}'_T = -\sum_{k=1}^N h_k \dot{\omega}_k$ and $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^N h_k Y_k V_{k,i}$

Table 1.8: Conservation equations for constant pressure, low Mach number flames.

32.6 Boundary Conditions

In first step, two classes of boundary conditions must be distinguished:

- Physical boundary conditions.
- Soft or numerical boundary conditions.

Physical boundary conditions specify the known physical behavior of one or more of the dependent variables at the boundaries. For example, specification of the inlet longitudinal velocity on a boundary is a physical boundary condition. These conditions are independent of the numerical method used to solve the relevant equations. The number of necessary and sufficient

physical boundary conditions for well-posedness should match theoretical results as summarized in the table below:

Boundary type:	EULER	NAVIER STOKES	NAVIER STOKES
	Non-reacting	Non-reacting	Reacting
Supersonic inflow	5	5	5 + N
Subsonic inflow	4	5	5 + N
Supersonic outflow	0	4	4 + N
Subsonic outflow	1	4	4 + N

Table 9.1: Number of physical boundary conditions required for well-posedness (three-dimensional flow). N is the number of reacting species.

A boundary condition is called “numerical” when no explicit physical law fixes one of the dependent variables, but the numerical implementation requires to specify something about this variable. Variables which are not imposed by physical boundary conditions must be computed on the boundaries by solving the same conservation equations as in the domain.

32.6.1 Reacting Navier-Stokes equations near a boundary

The method is first derived using the following assumptions:

- All gases have the same constant heat capacity ($C_{pk} = C_p$) and Y is constant.
- Volume forces are neglected ($f_k=0$) like volume heat sources ($\dot{Q} = 0$)
- Fick’s law without correction velocity is used for diffusion velocities.

Under these assumptions, the fluid dynamics equations derived before are written:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0$$

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial}{\partial x_i}[u_i(\rho E + p)] = -\frac{\partial}{\partial x_i}(q_i) + \frac{\partial}{\partial x_i}(u_i \tau_{ij}) + \dot{\omega}_T$$

$$\frac{\partial(\rho u_j)}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i u_j) + \frac{\partial p}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_i} \quad \text{for } i = 1, 3$$

$$\frac{\partial(\rho Y_k)}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i Y_k) = \frac{\partial}{\partial x_i}(M_{ki}) - \dot{\omega}_k \quad \text{for } k = 1, N$$

where E is the total energy (without chemical term) defined in Table 1.3:

$$E = e_s + \frac{1}{2}u_k u_k = \int_0^T C_v dT + \frac{1}{2}u_k u_k = C_v T + \frac{1}{2}u_k u_k$$

The molecular fluxes of heat (q_i) and of species (M_{ki}) in direction i are defined by:

$$q_i = -\lambda \frac{\partial T}{\partial x_i} \quad \text{and} \quad M_{ki} = \rho D_k \frac{\partial Y_k}{\partial x_i}$$

32.6.2 Comparison between NSCBC implementation for Euler and Navier-Stocks

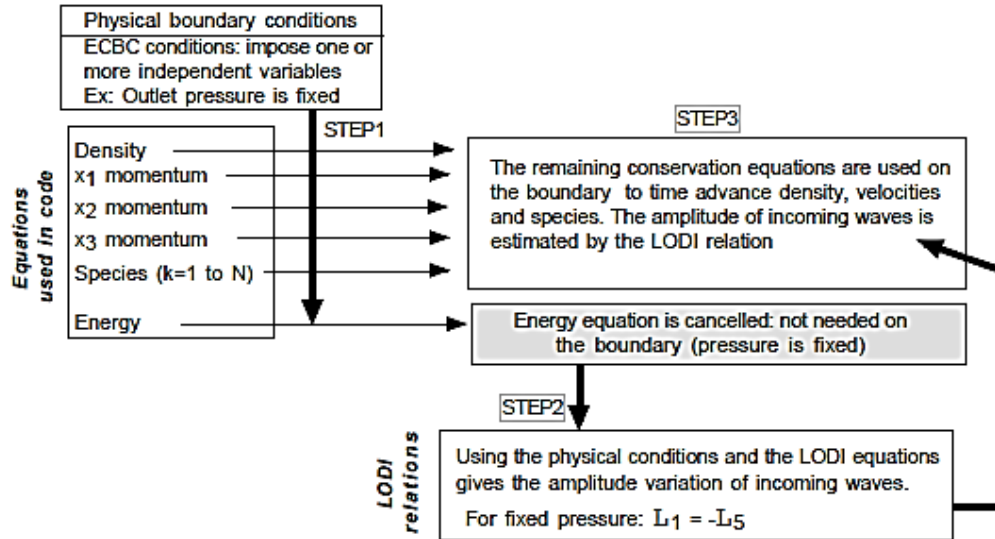


Figure 9.3: NSCBC implementation for Euler equations. Example for a fixed pressure outlet.

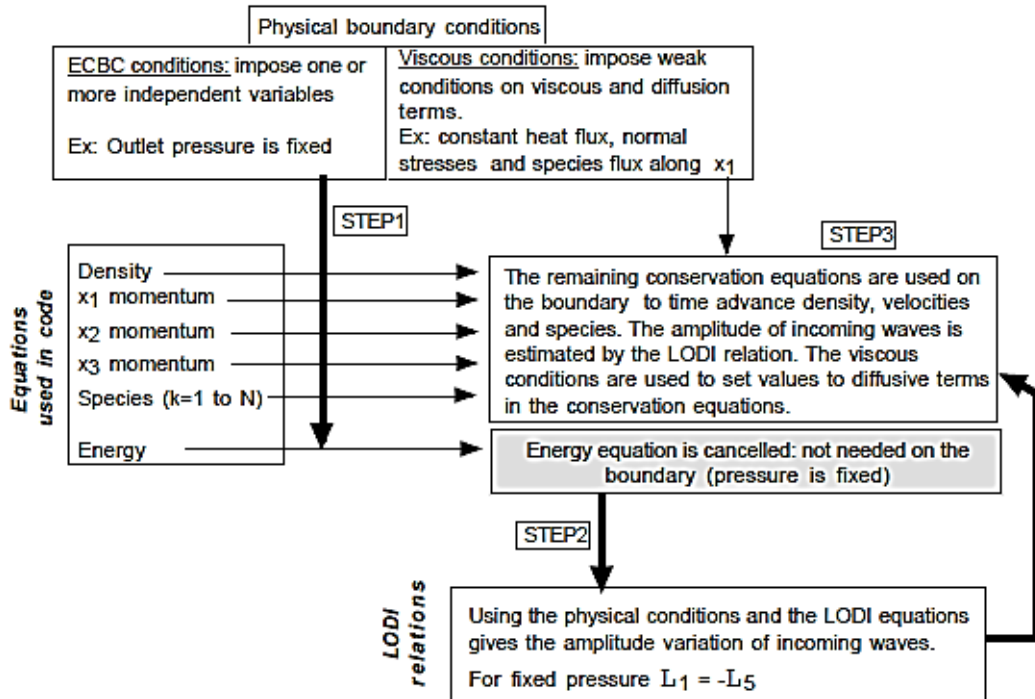


Figure 9.4: NSCBC implementation for Navier-Stokes equations. Example for a fixed pressure outlet.

Step 1 and 2 are the same for Euler and Navier-Stokes equations.

	Euler		Navier -Stokes with N species			
	ECBC Conditions	Total Nbr	ECBC Conditions	Viscous Conditions	Reaction Condition	Total Nbr
SI-1	u_i, T, Y_k imposed		u_i, T, Y_k imposed	No	No	4+N
		4+N	4+N	0	0	
SI-2	u_i, ρ, Y_k imposed		u_i, ρ, Y_k imposed	$\frac{\partial \tau_{11}}{\partial x_1} = 0$	No	5+N
		4+N	4+N	1	0	
SI-3	$u_1 - 2\frac{c}{\gamma-1}, u_2,$ u_3, s, Y_k imposed		$u_1 - 2\frac{c}{\gamma-1}, u_2,$ u_3, s, Y_k imposed	$\frac{\partial \tau_{11}}{\partial x_1} = 0$	No	5+N
		4+N	4+N	1	0	
SI-4	No reflected wave		No reflected wave	$\frac{\partial \tau_{11}}{\partial x_1} = 0$	No	5+N
		4+N	4+N	1	0	

Table 9.2: Physical boundary conditions for three-dimensional reacting flows. Subsonic inflow. The total number of species is N . The boundary is normal to the x_1 axis.

		Euler		Navier-Stokes with N species			
		ECBC Condition	Total Nbr	ECBC Conditions	Viscous Conditions	Reaction Condition	Total Nbr
B2	Perfectly non reflecting outflow	No reflection		No reflection	$\frac{\partial \pi_{12}}{\partial x_1} = 0$ $\frac{\partial \pi_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	4+N
			1	1	3	N	
B3	Partially non reflecting outflow	P infinity imposed		P at infinity imposed	$\frac{\partial \pi_{12}}{\partial x_1} = 0$ $\frac{\partial \pi_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	4+N
			1	1	3	N	
B4	Subsonic reflecting outflow	P outlet imposed		P outlet imposed	$\frac{\partial \pi_{12}}{\partial x_1} = 0$ $\frac{\partial \pi_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	4+N
			1	1	3	N	
NSW	Isothermal no slip wall			$u_i = 0$ T imposed		$M_{k1} = 0$	4+N
				4	0	N	
ASW	Adiabatic slip wall			Zero normal velocity	$q_1 = 0$	$M_{k1} = 0$	4+N
				3	1	N	

Table 9.3: Physical boundary conditions for three-dimensional reacting flows: subsonic outflow and walls. The total number of species is N . The boundary is perpendicular to the x_1 axis.

32.6.3 Examples of implementation

It is useful to go into more details by presenting the practical implementation of the NSCBC method in the following typical situations:

- A subsonic inflow with fixed velocities (SI-1)
- A subsonic non-reflecting inflow (SI-4)
- Non-reflecting outflows (B2 and B3)
- A subsonic reflecting outflow (B4)
- An isothermal no-slip wall (NSW)
- An adiabatic slip wall (ASW)

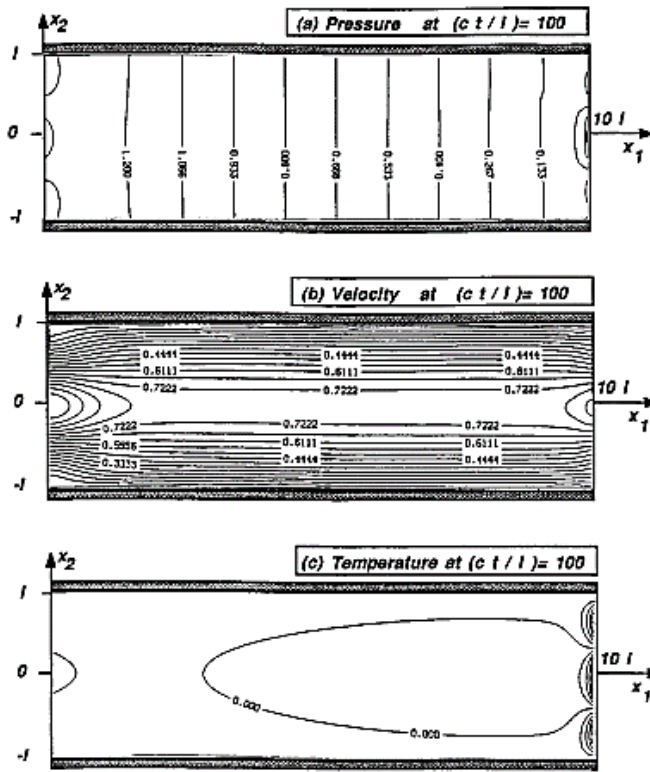


Figure 9.24: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet condition B1.

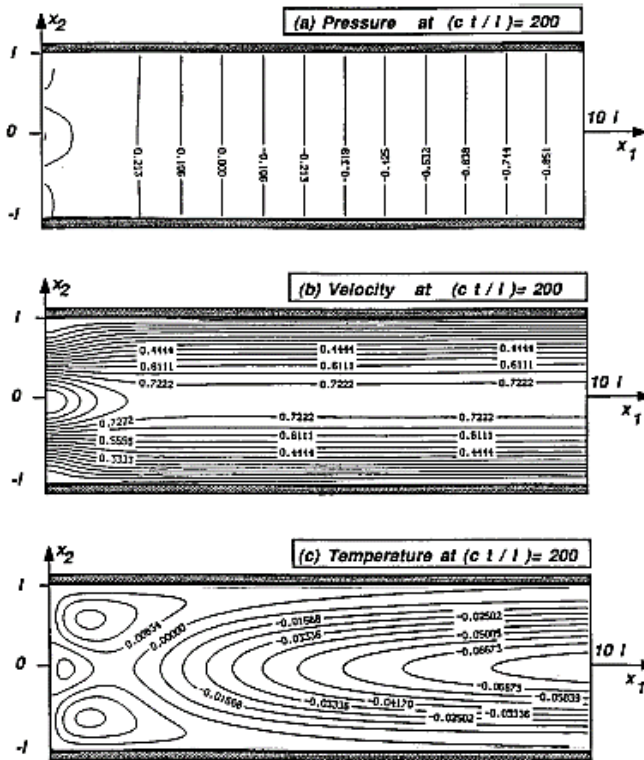


Figure 9.26: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet NSCBC condition B3 ($\sigma = 0.15$).

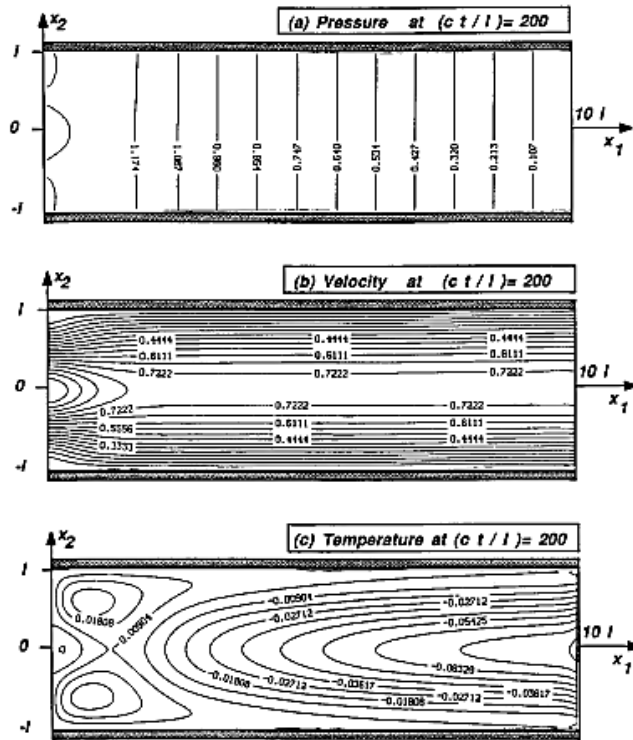


Figure 9.27: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet NSCBC condition B4.

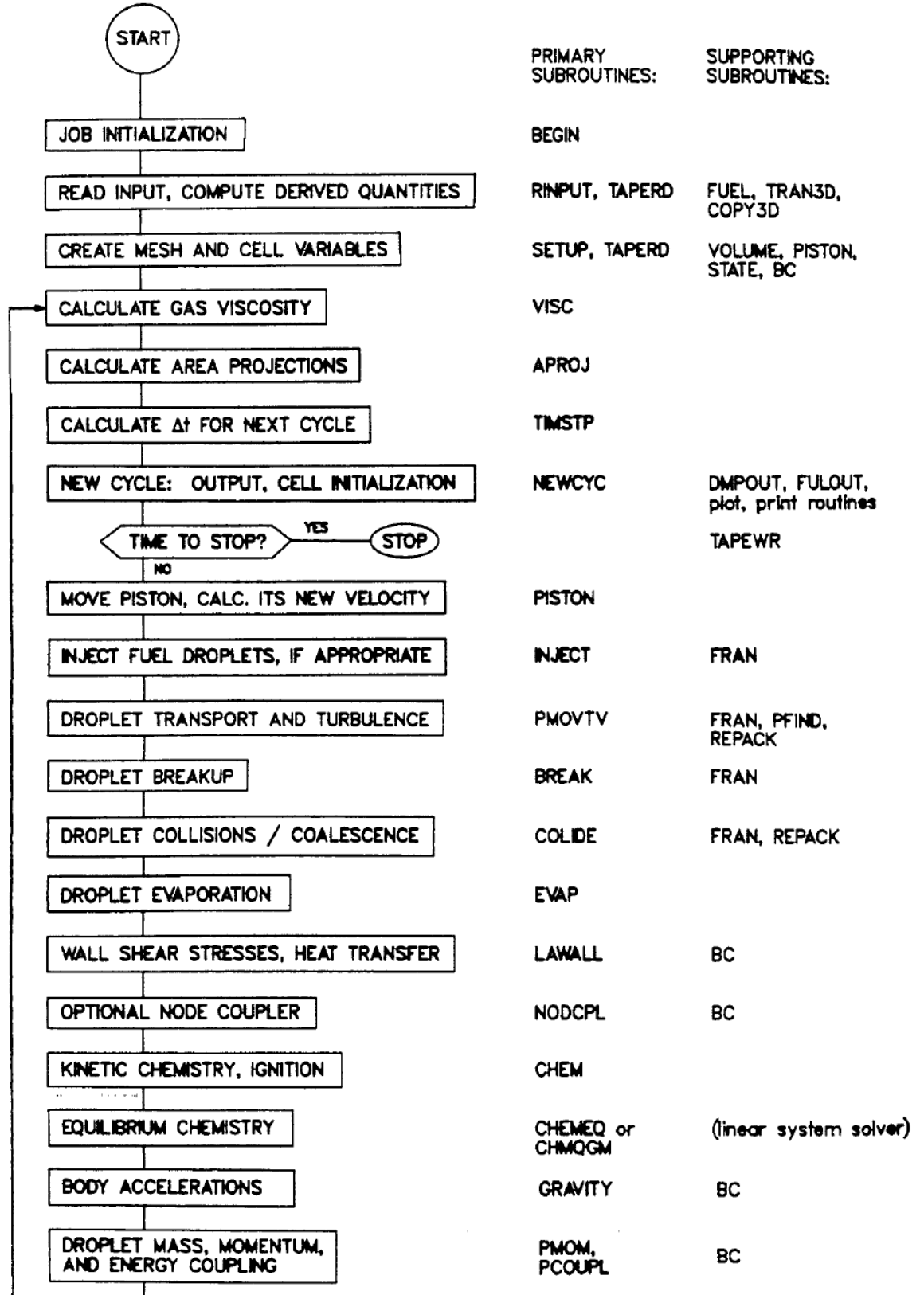
32.7 Conservation equation Comparison between KIVAII and Poinso

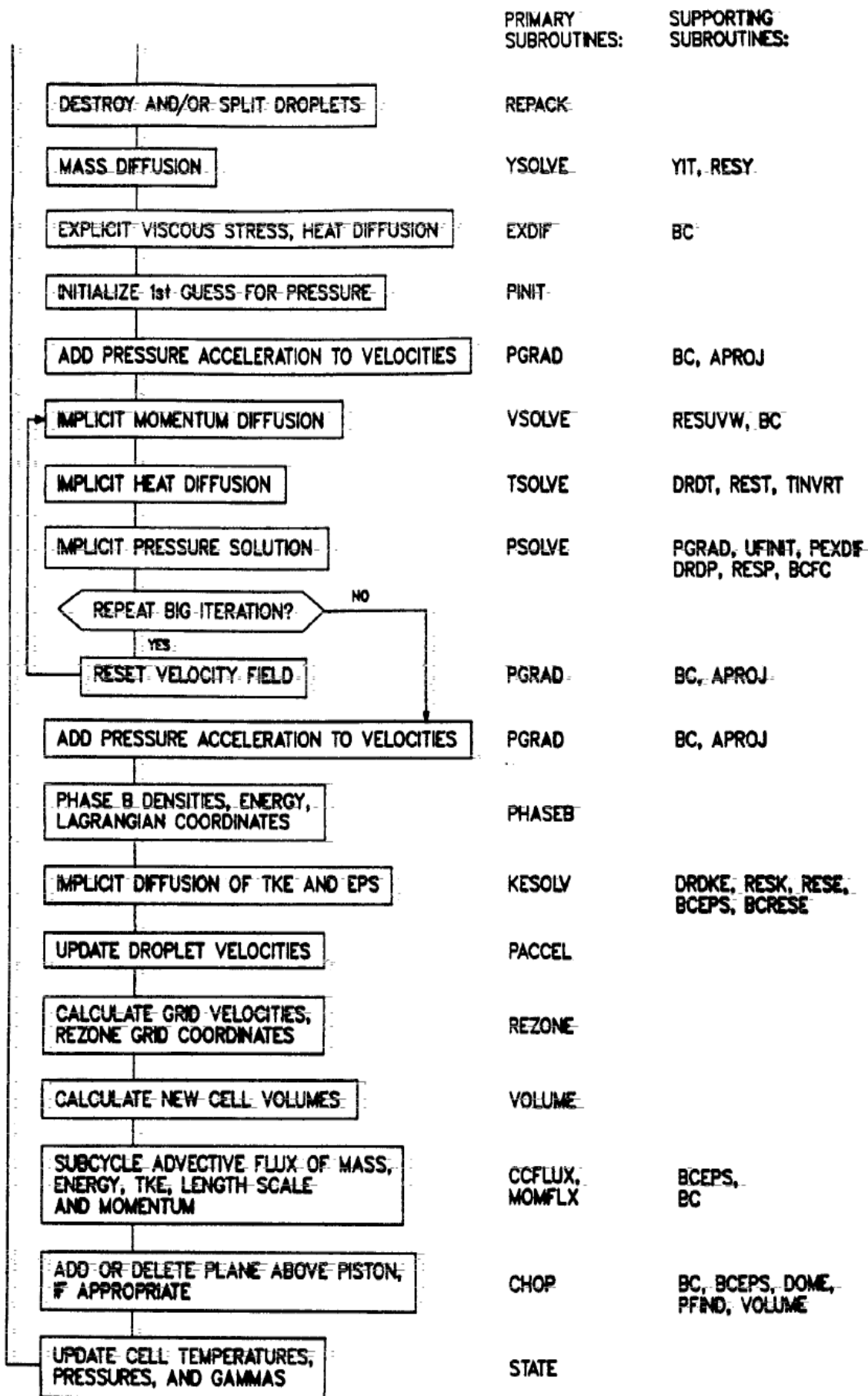
Governing equation	Computational fluid dynamics	KIVA II Program	Poinso
Mass conservation	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \dot{\rho}^s$	$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0$ <p>Species: for $k = 1$ to $N - 1$ (or N if total mass is not used)</p> <p>With diffusion velocities:</p> $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_{k,i}) Y_k) = \dot{\omega}_k$ <p>With Fick's law:</p> $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i} (\rho (u_i + V_i^c) Y_k) = \frac{\partial}{\partial x_i} (\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k \text{ and } V_i^c = \sum_{k=1}^N D_k \frac{\partial Y_k}{\partial x_i}$
Momentum equation	$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$ $\frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$ $\frac{\partial(\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$	$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\frac{1}{\alpha} \nabla p - A \nabla (2/3 \rho k) + \nabla \cdot \boldsymbol{\sigma} + \mathbf{F}^s + \rho \mathbf{g}$ $\mathbf{J} = -K \nabla T - \rho D \sum_m h_m \nabla (\rho_m / \rho)$	$\frac{\partial}{\partial t} \rho u_j + \frac{\partial}{\partial x_i} \rho u_i u_j = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho \sum_{k=1}^N Y_k f_{k,j}$
Energy equation	$\frac{\partial(\rho e)}{\partial t} + \nabla \cdot (\rho e \vec{V}) = \rho \dot{q} + \frac{\partial}{\partial x} (k \frac{\partial T}{\partial x}) + \frac{\partial}{\partial y} (k \frac{\partial T}{\partial y}) + \frac{\partial}{\partial z} (k \frac{\partial T}{\partial z})$ $- \rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)^2$ $+ \mu \left[2 \left(\frac{\partial u}{\partial x} \right)^2 + 2 \left(\frac{\partial v}{\partial y} \right)^2 + 2 \left(\frac{\partial w}{\partial z} \right)^2 \right]$ $+ \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right)^2$	$\frac{\partial(\rho I)}{\partial t} + \nabla \cdot (\rho \mathbf{u} I) = -\rho \nabla \cdot \mathbf{u} + (1 - A_0) \boldsymbol{\sigma} : \nabla \mathbf{u} - \nabla \cdot \mathbf{J} + A_0 \rho e + \dot{Q}^c + \dot{Q}^s$	$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i} (u_i + V_{k,i})$ <p>with $\dot{\omega}_T = -\sum_{k=1}^N \Delta h_{f,k}^o \dot{\omega}_k$ and $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^N h_k Y_k V_{k,i}$</p>
Chemical kinetics		$\dot{\omega}_r = k_{fr} \prod_m (\rho_m / W_m)^{a'_{mr}} - k_{br} \prod_m (\rho_m / W_m)^{b'_{mr}}$	$\dot{\omega}_k = \sum_{j=1}^M \dot{\omega}_{kj} = W_k \sum_{j=1}^M \nu_{kj} Q_j$ $Q_j = K_{fj} \Pi_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu_{kj}} - K_{rj} \Pi_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu_{kj}}$

33 KIVA II مخطط التدفق العام لبرنامج

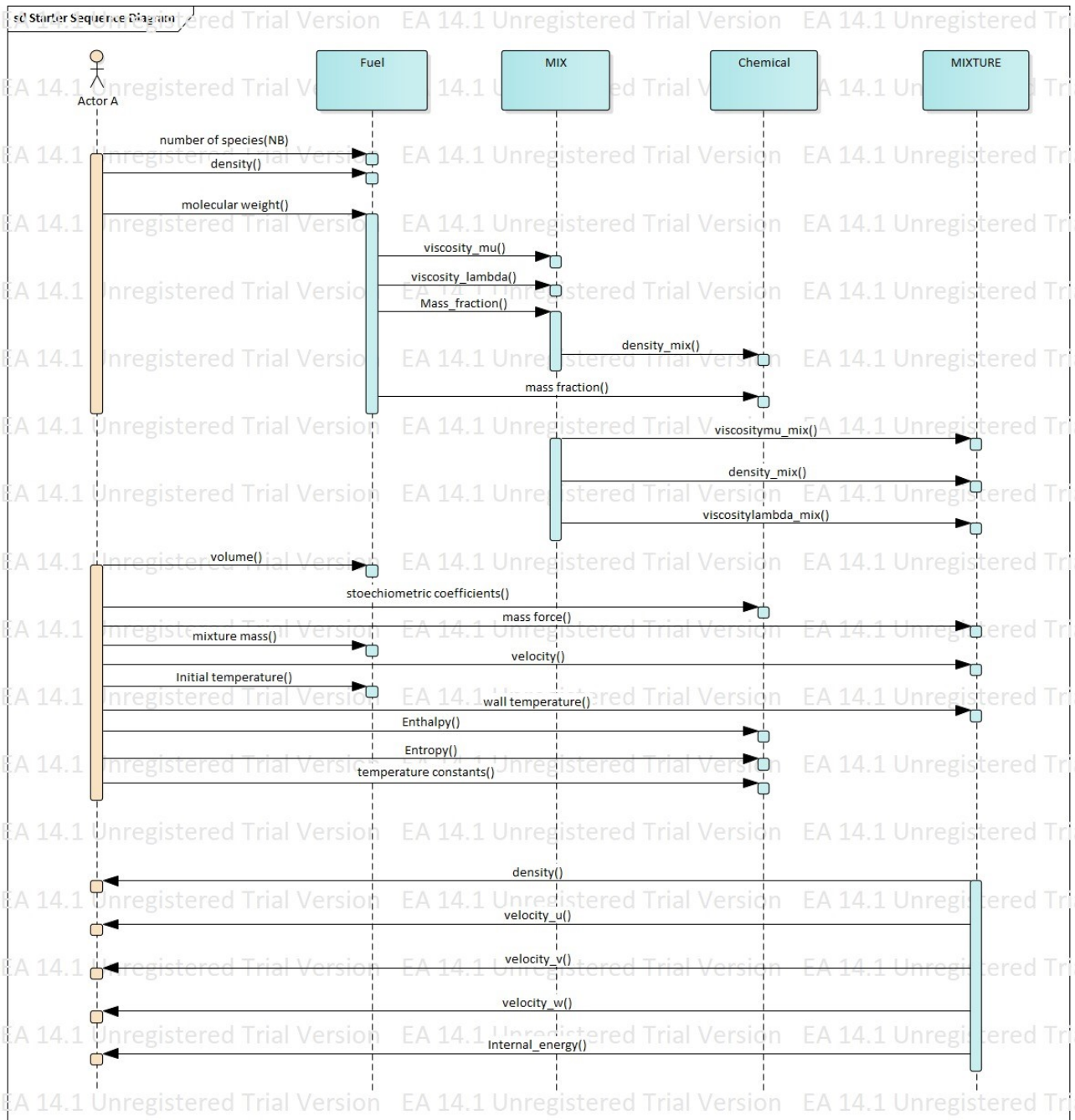
Our diagram is inspired from the KIVA II Diagrams

استوحى النموذج المعتمد في هذا الملف من البرامج التالية: KIVA II

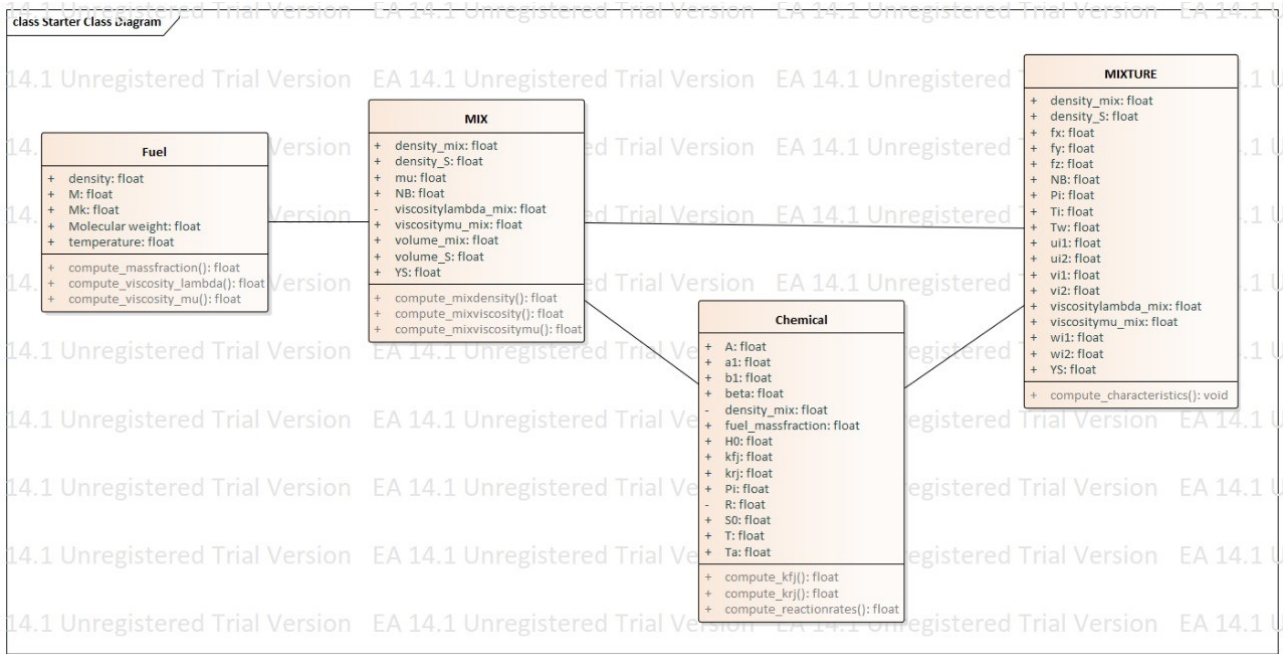




34 مخطط تسلسل البرنامج (sequence diagram)



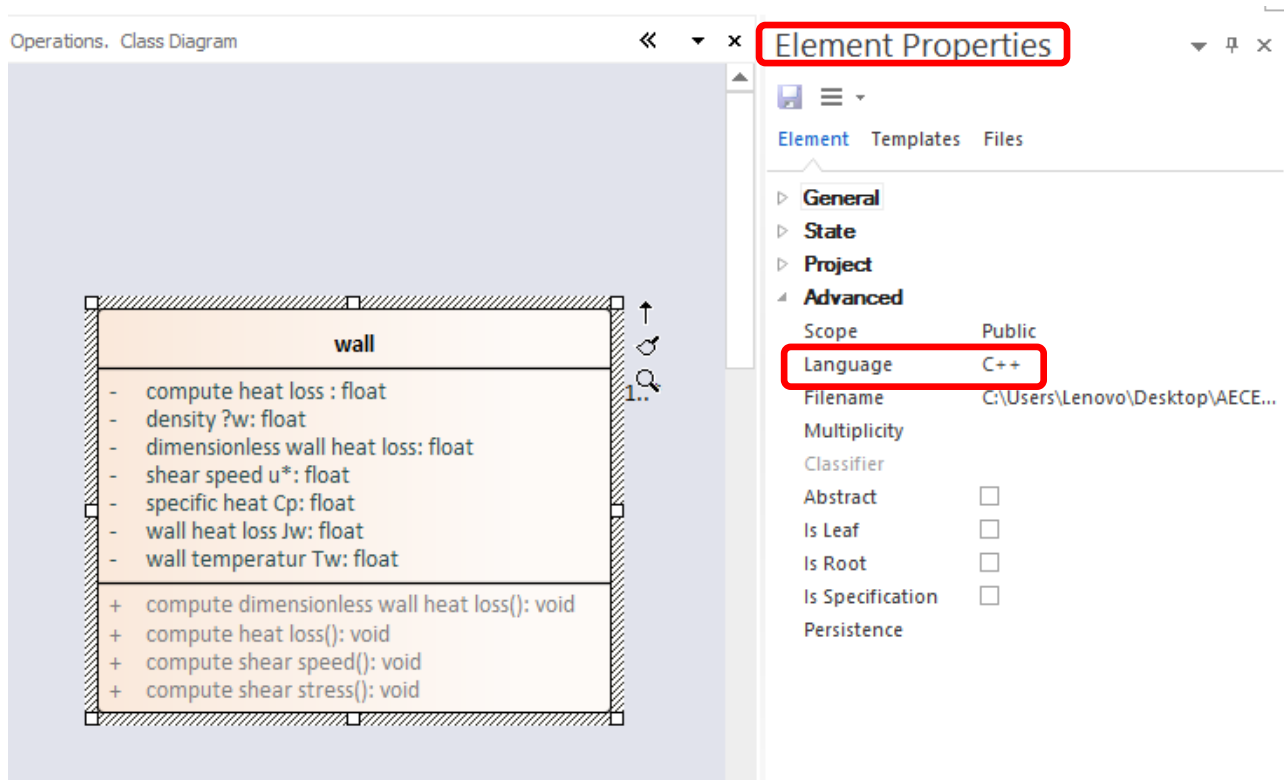
35 Class Diagram



36 Code generation

To generate your code from your class diagram, follow the steps below:

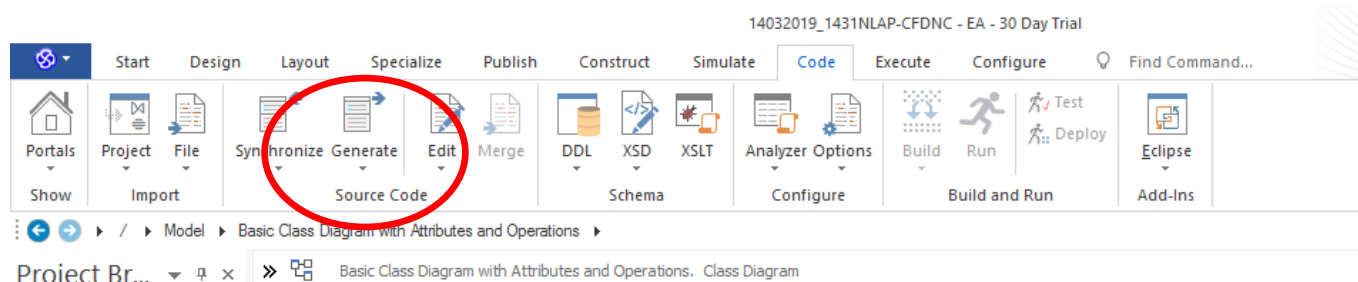
Before starting the code generation make sure to choose the language for your classes, in our case it's C++. First select the class then change the language in: "Element properties → Advanced → Language". Repeat this on all the classes that you want to generate code from.



Now that all the classes are ready follow these steps.

Step 1:

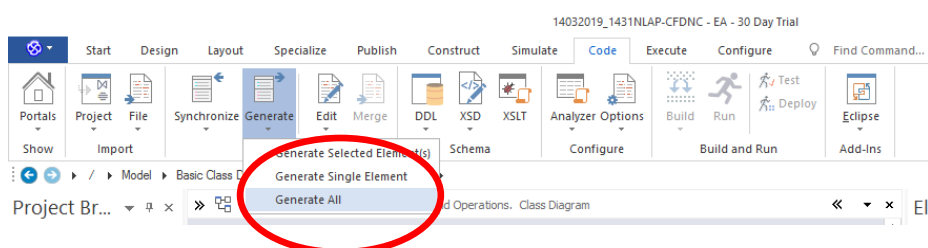
Go to "Code → Source Code → Generate" as shown in the image below



Step 2:

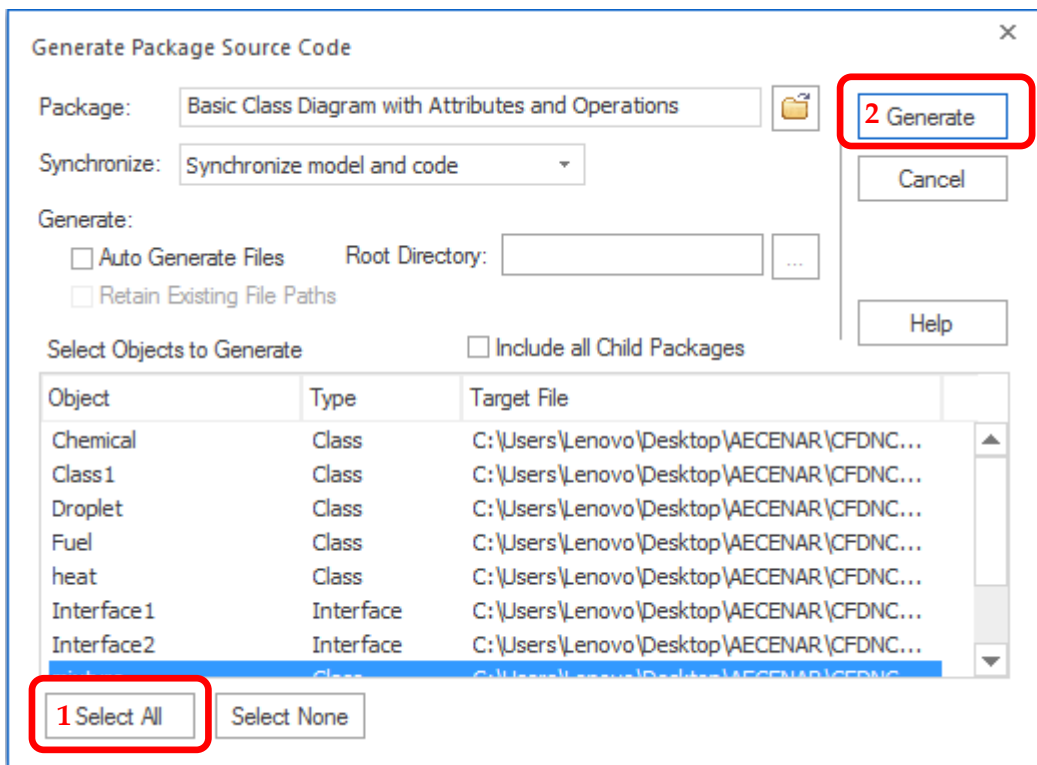
Select the type you want,

in our case "Generate all"



Step 3:

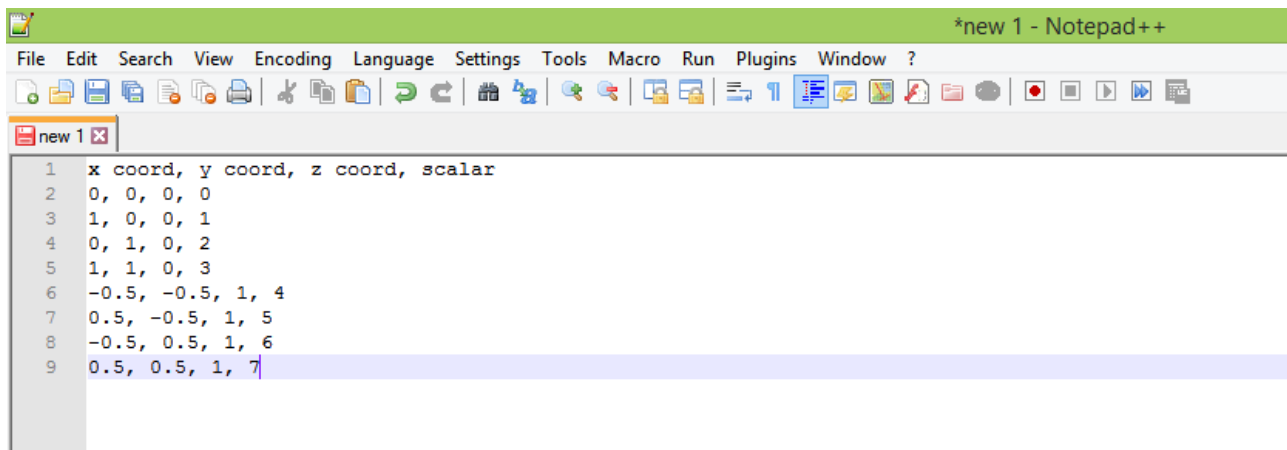
A dialog box will pop up. Choose the Select All button then Generate



The code will be generated after you choose the designated folder.

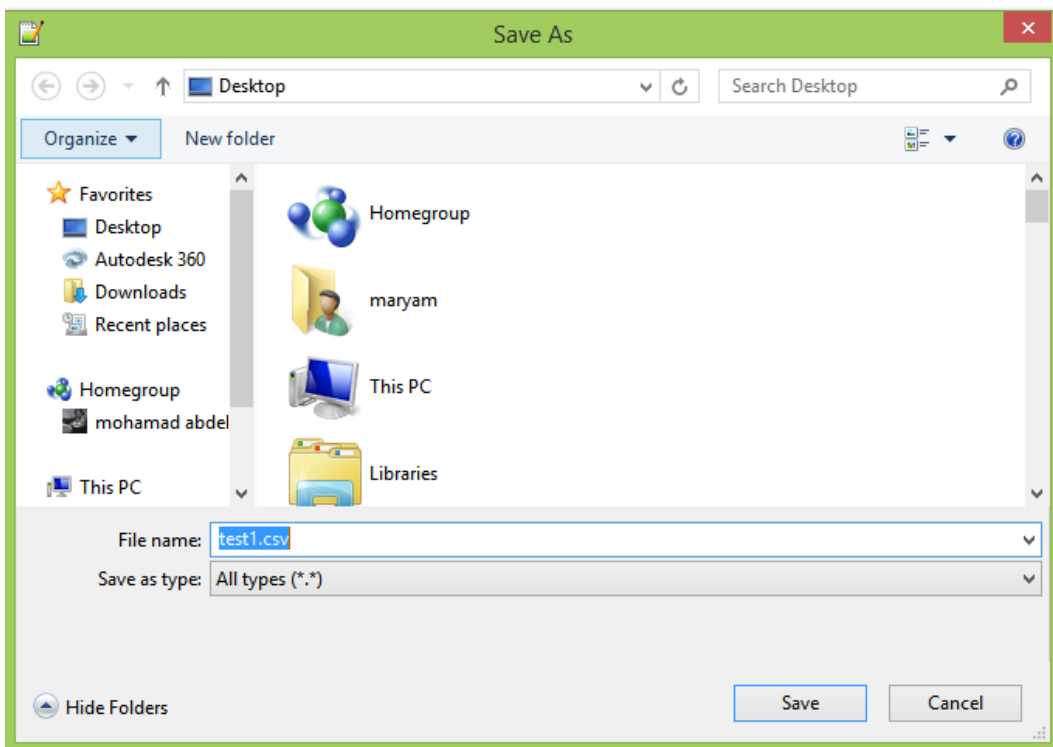
37 Para view Input files¹¹

The type of files we are using to read our solution via Para view is the .csv files (comma separated variables). In this section we'll show a simple example (8 points). First start with defining the .csv file using notepad++ as shown in the figure bellows.



```
*new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
1 x coord, y coord, z coord, scalar
2 0, 0, 0, 0
3 1, 0, 0, 1
4 0, 1, 0, 2
5 1, 1, 0, 3
6 -0.5, -0.5, 1, 4
7 0.5, -0.5, 1, 5
8 -0.5, 0.5, 1, 6
9 0.5, 0.5, 1, 7
```

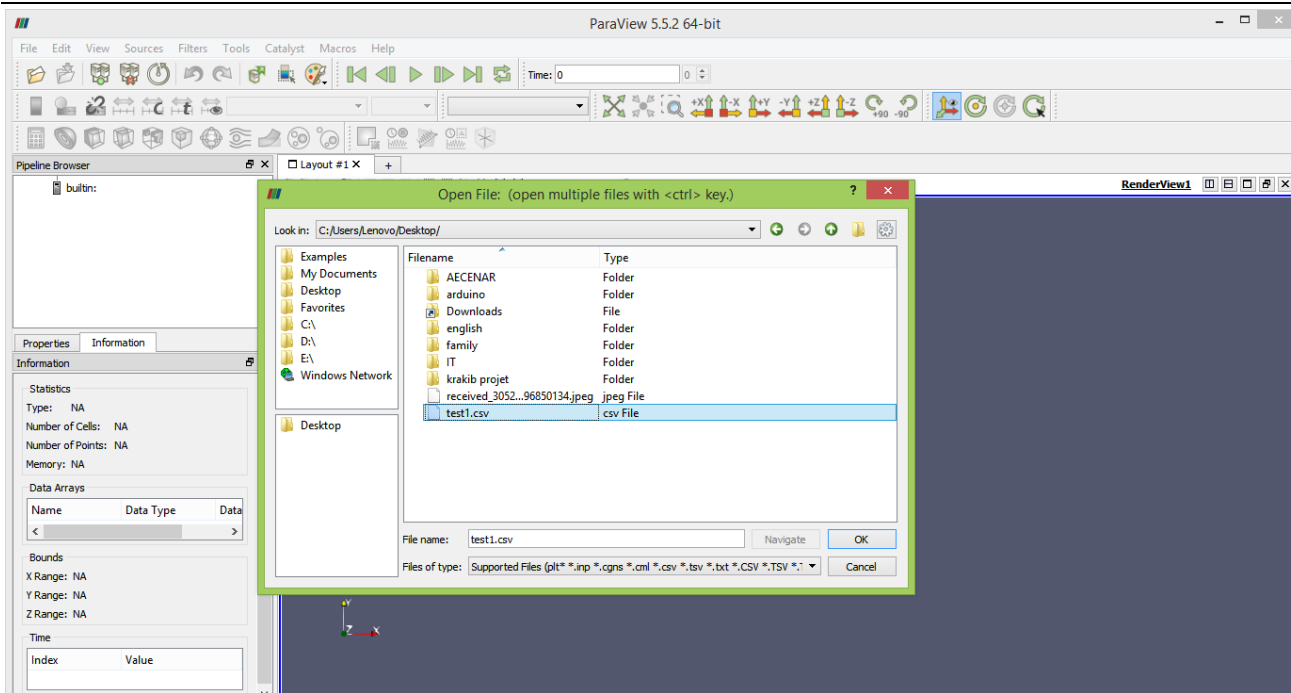
Then save your file as: All types (*.*) as shown in our example test1.csv.



Now it's ready to be opened in Para view.

Select the open button and choose your file .csv.

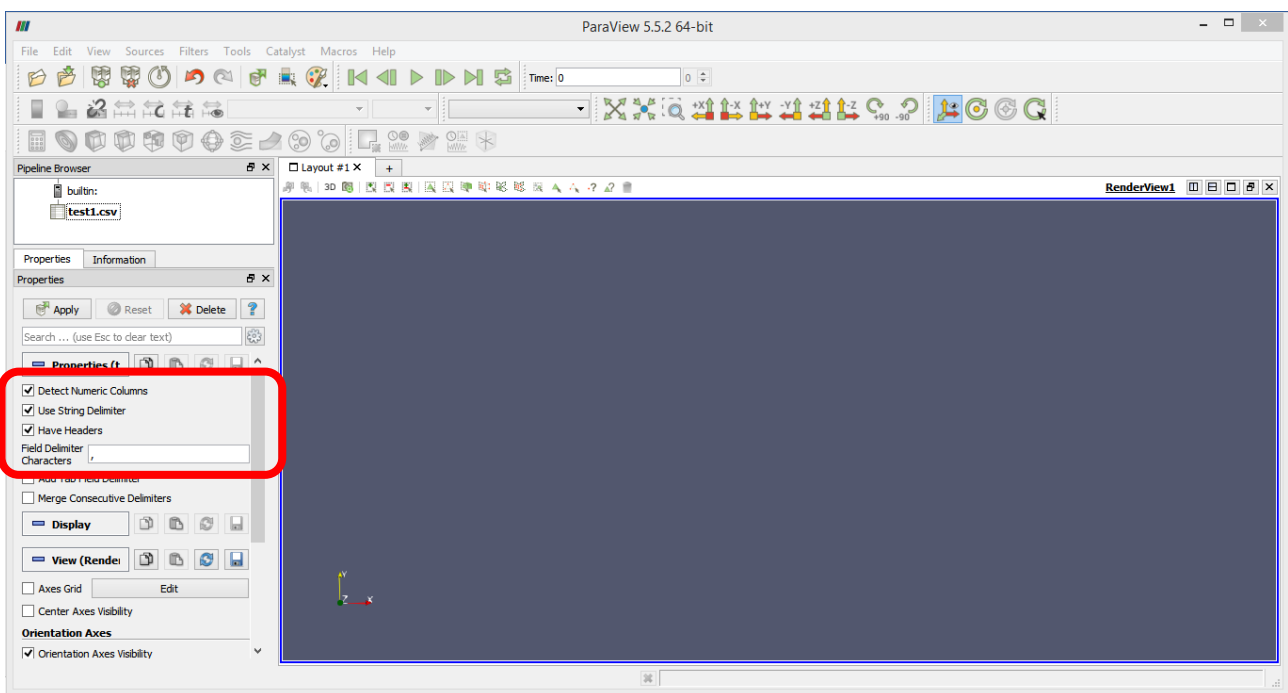
¹¹ https://www.paraview.org/Wiki/ParaView/Data_formats
<https://www.youtube.com/watch?v=mNR2Vn6r0io>



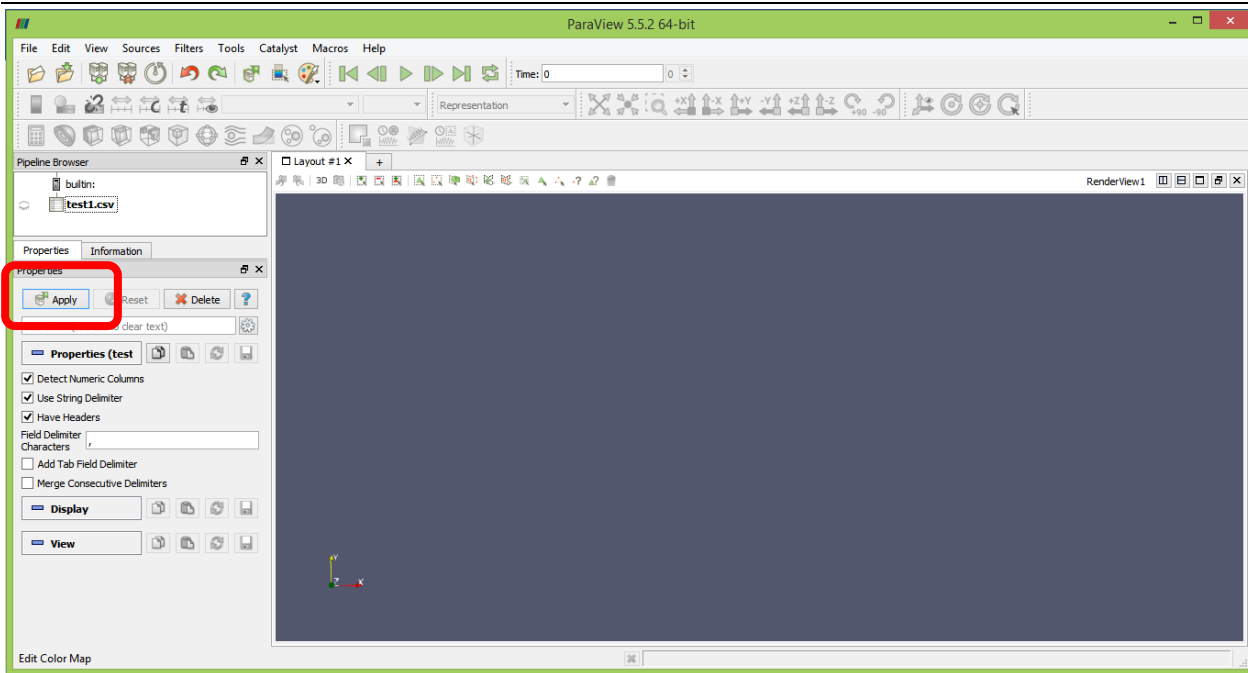
Start Para View, and read in this data. Note that the default settings should be used:

- Detect Numeric Columns ON
- Use String Delimiter ON
- Have Headers ON
- Field Delimiter Characters should be a comma - ','

(See figure below)

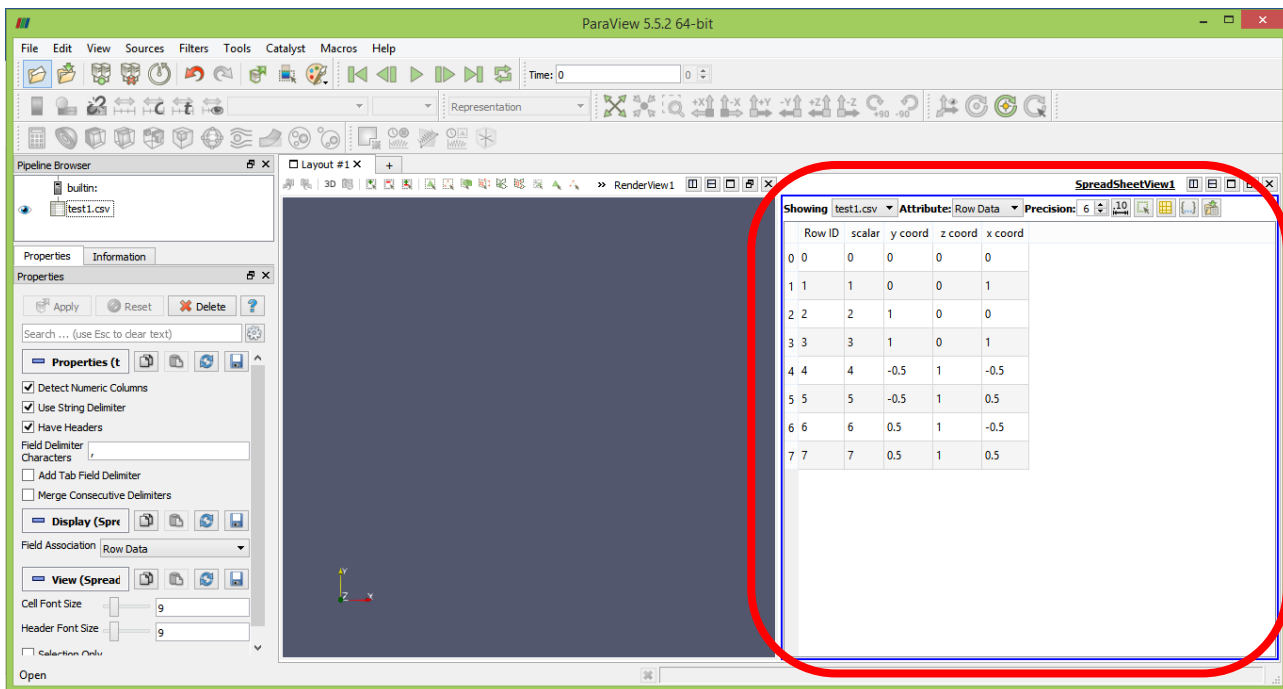


Then press apply.



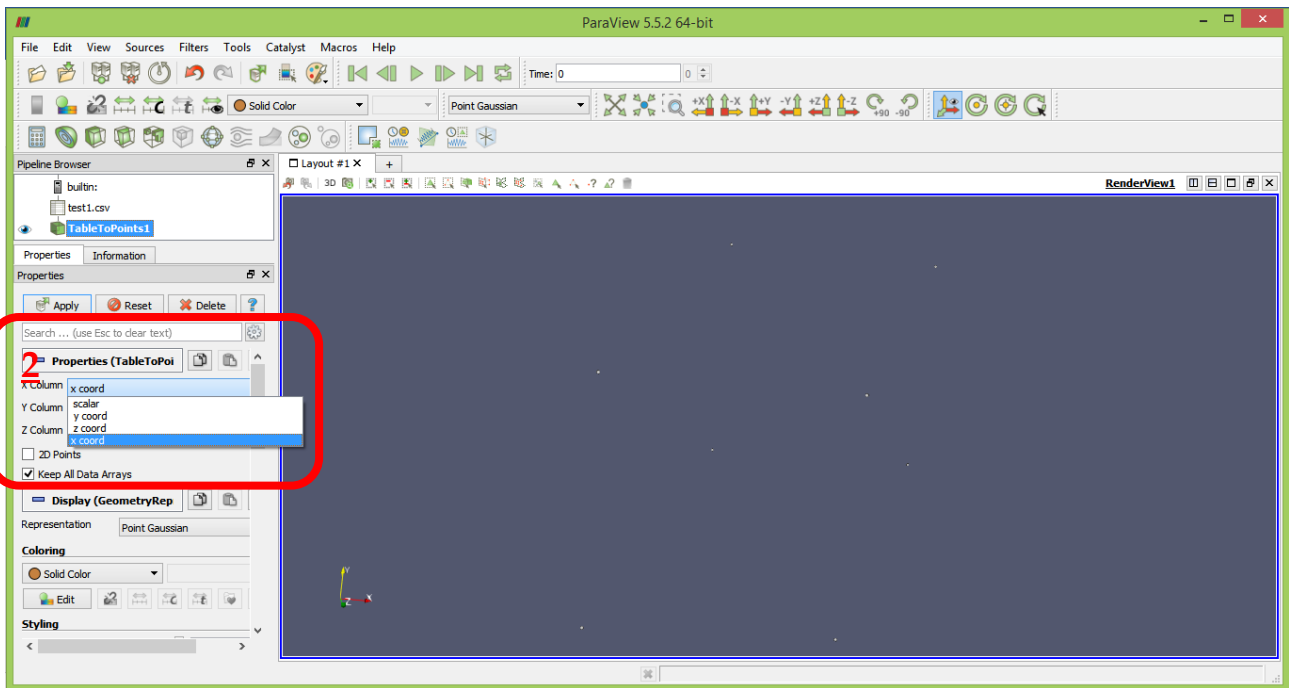
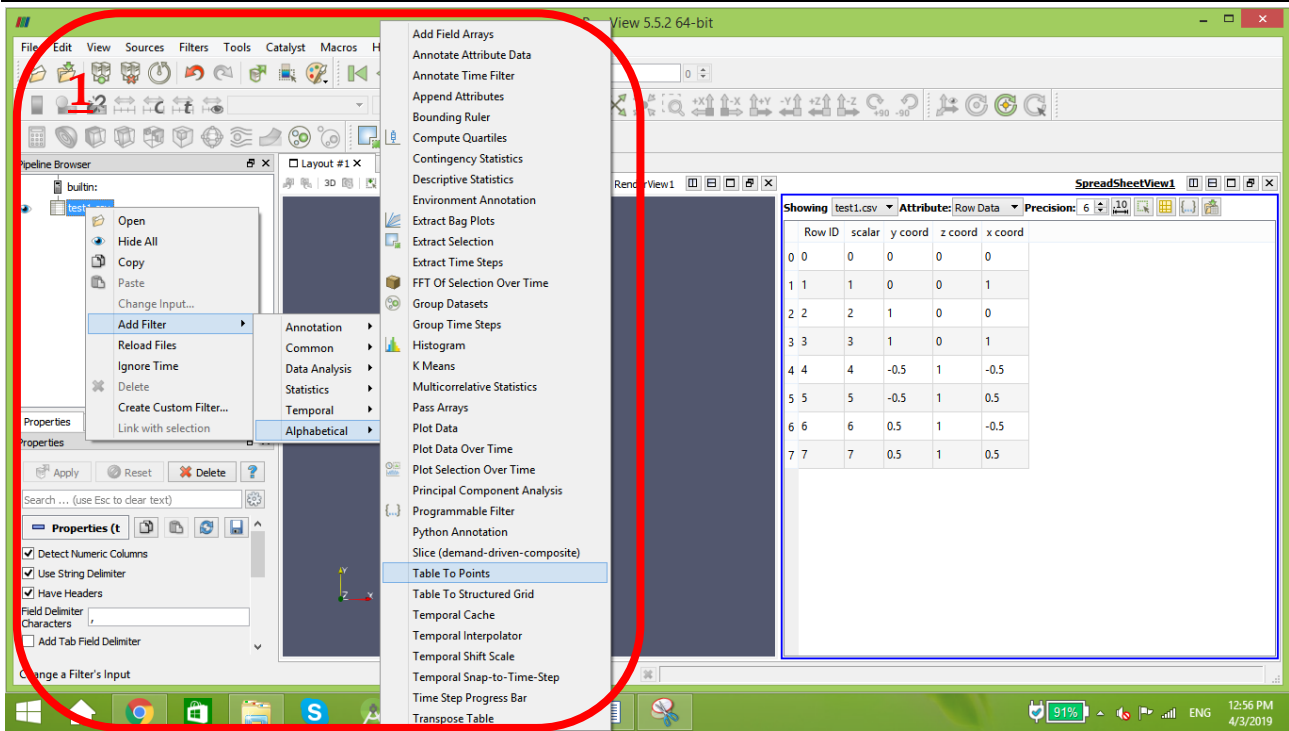
The data

should show up as a table.



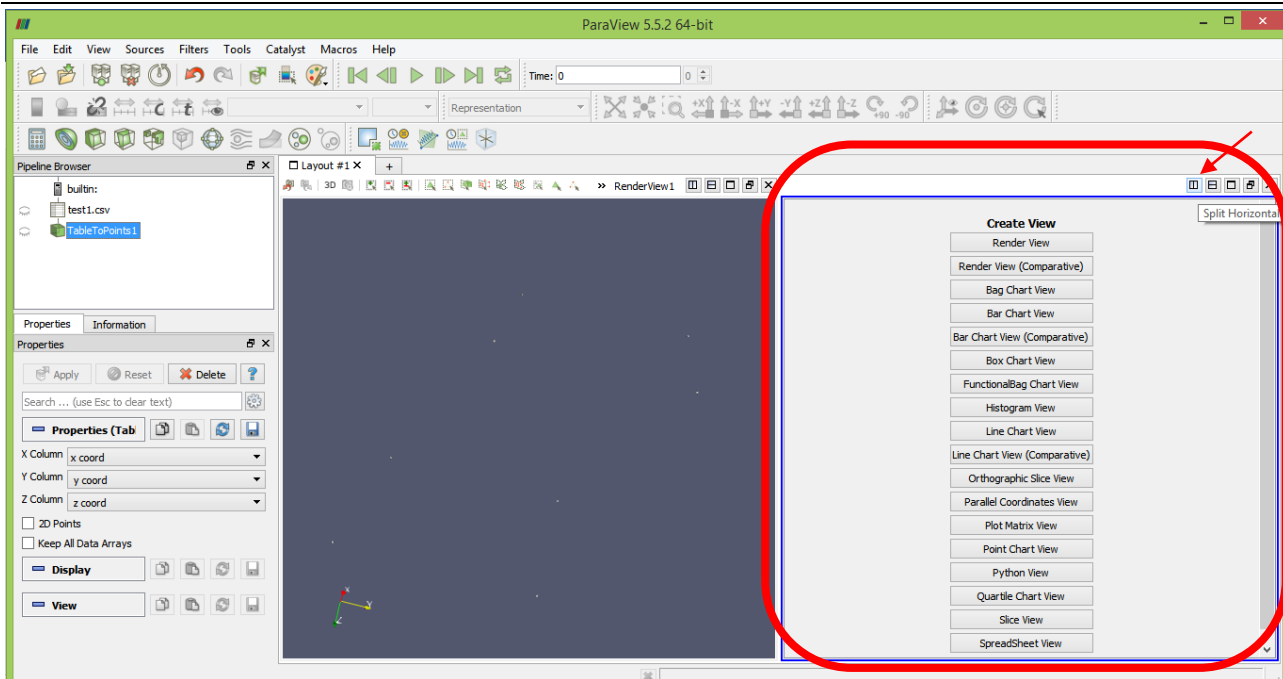
37.1 Displaying data as points

- Run the filter Filters/ Alphabetical/ Table to Points (right click on the table at the left as shown in the figure below).
- Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.



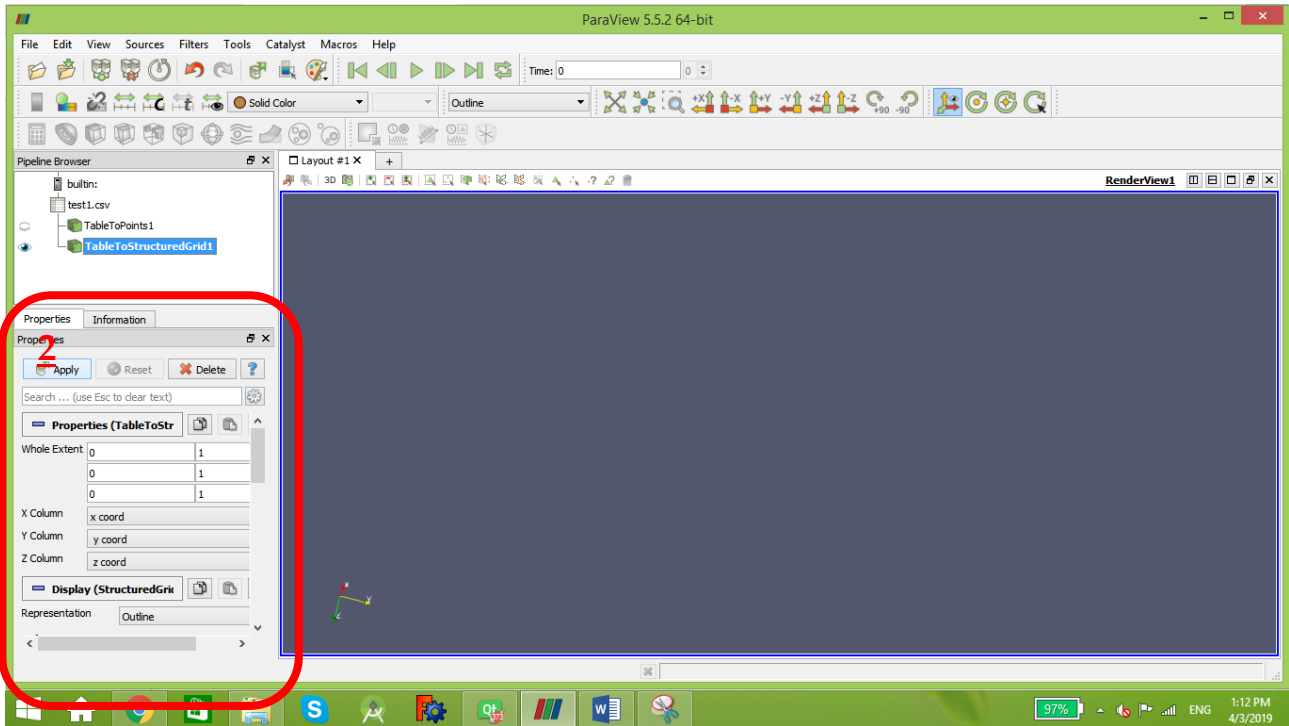
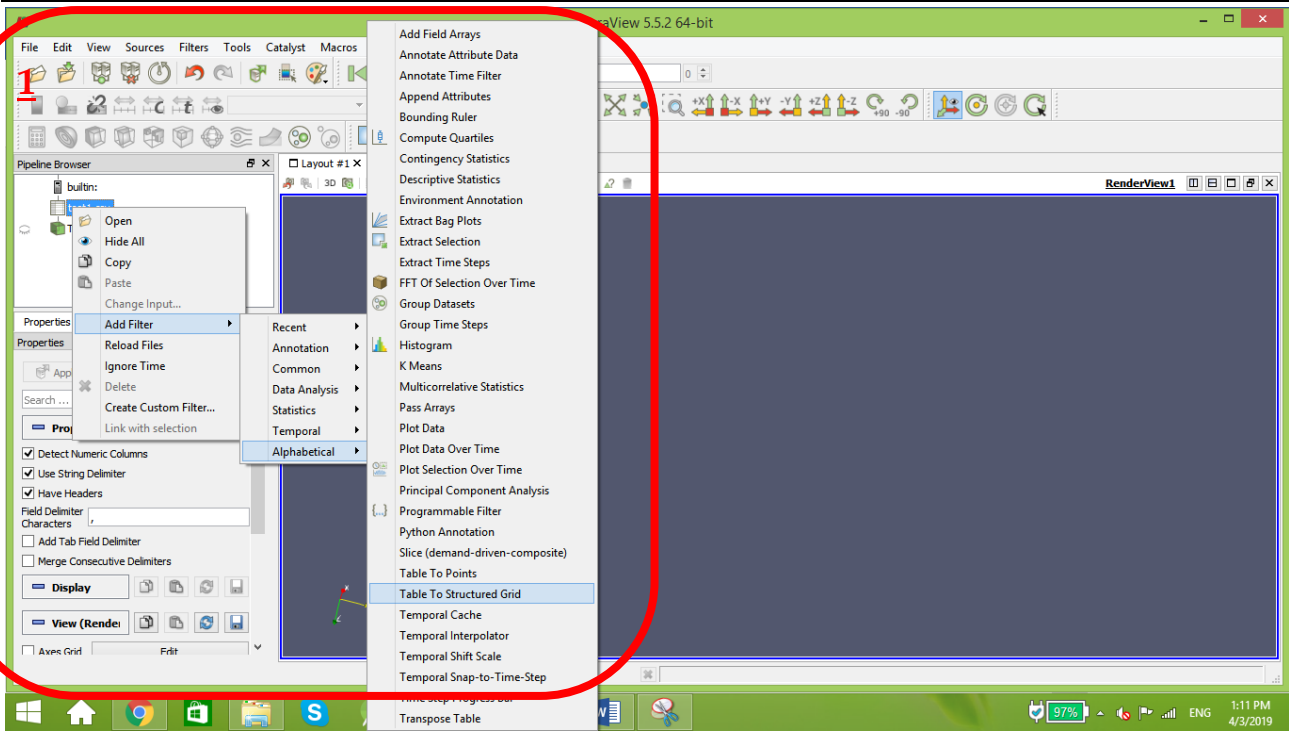
Press apply and the points are visible now.

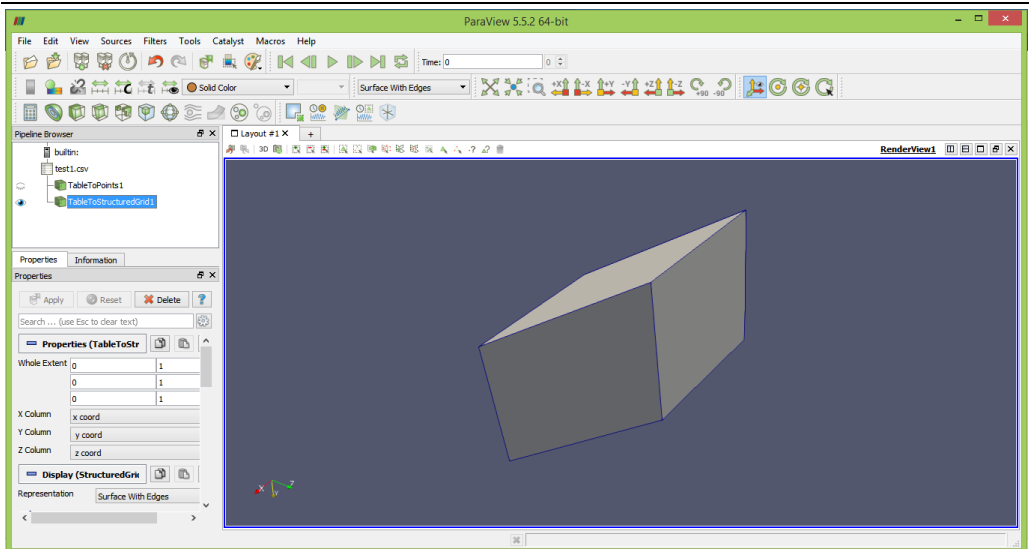
If your points didn't show up press on "split horizontal" button. And choose the desired view.



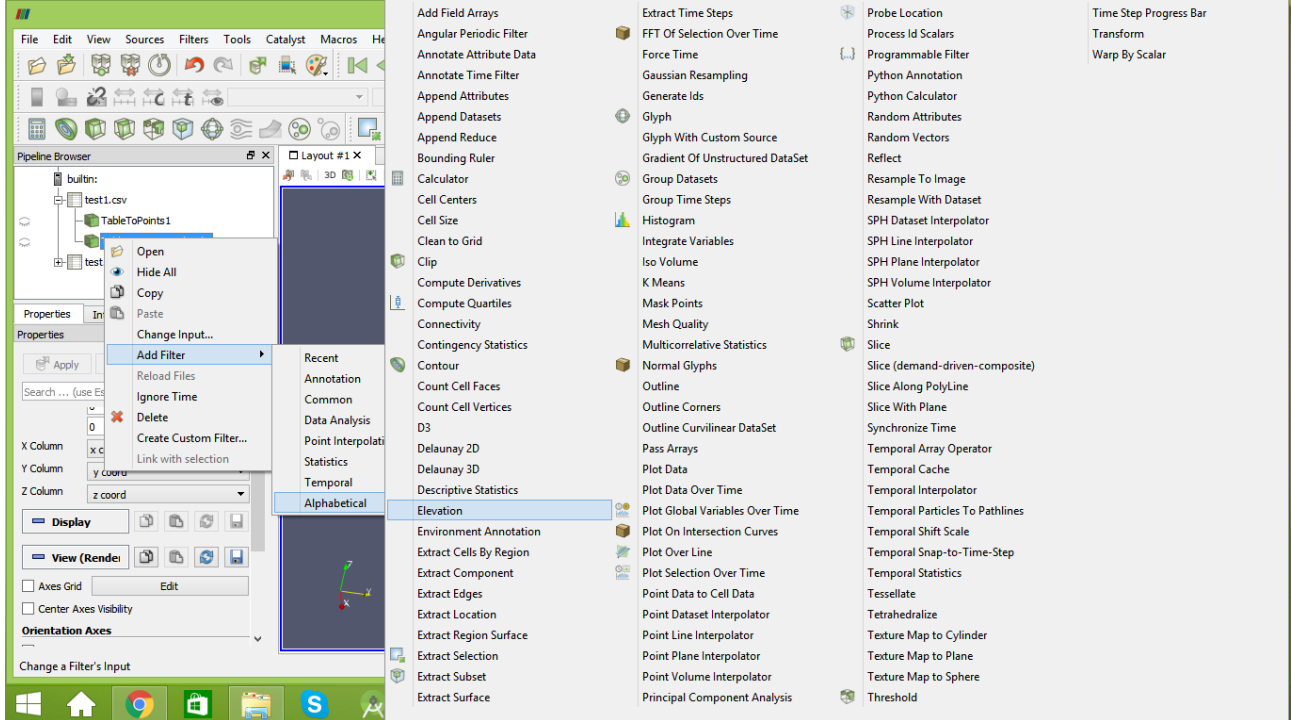
37.2 Displaying data as structured grid

1. Run the filter Filters/ Alphabetical/ Table to Structured Grid.
2. Tell Para View what extent, or array sizes, your data is in. For instance, the data above has 8 points, forming a leaning cube. Points arrays are in $X == \text{size } 2$, $Y == \text{size } 2$, and $Z == \text{size } 2$. In this example we will use C indexing for the arrays, thus they go from 0 to 1 (2 entries).
 - Whole extent is as follows:
 - 0 1
 - 0 1
 - 0 1
3. Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.

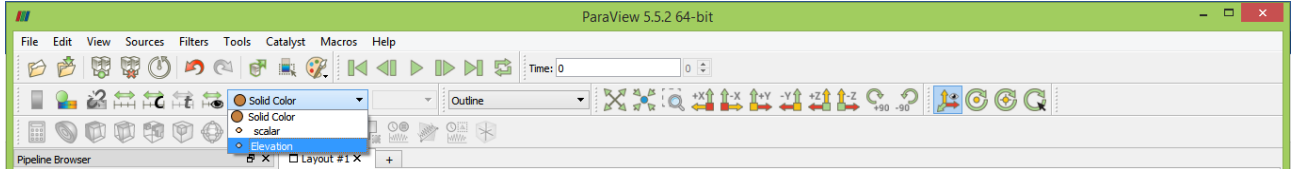




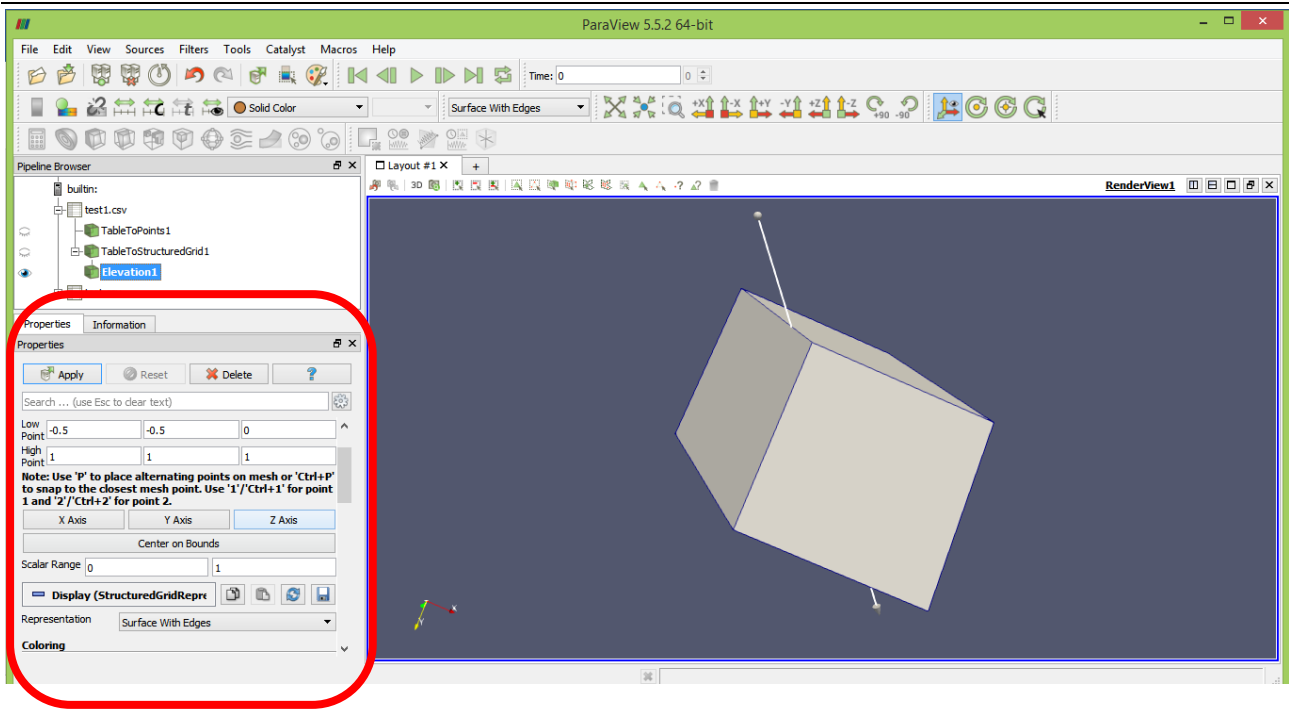
Now to represent your results with colors, right click on “table to structure” → add filter → Alphabetic → elevation.



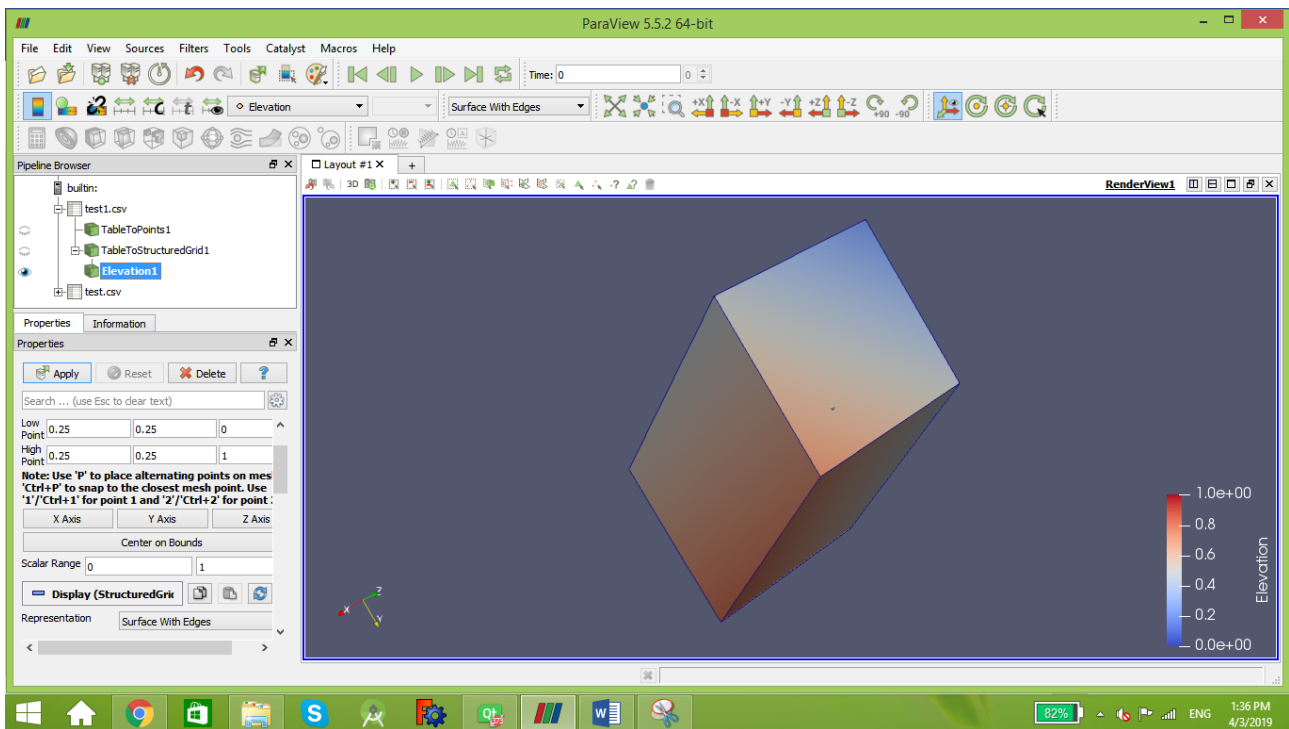
Make sure that you select the elevation button.



Finally choose the desired axis, then apply.



Now it's ready.



38 Discretization of partial differential equations

38.1 The continuity equation (mass conservation)

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0$$

$$\frac{\partial \rho}{\partial t} + \rho \frac{\partial u}{\partial x} + u \frac{\partial \rho}{\partial x} + \rho \frac{\partial v}{\partial y} + v \frac{\partial \rho}{\partial y} + \rho \frac{\partial w}{\partial z} + w \frac{\partial \rho}{\partial z} = 0$$

$$\begin{aligned} \frac{\rho_{ijk}^{t+1} - \rho_{ijk}^{t-1}}{2\Delta t} = & -\rho_{ijk}^t \left(\frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} \right) - u_{ijk}^t \left(\frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} \right) - \rho_{ijk}^t \left(\frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} \right) \\ & - v_{ijk}^t \left(\frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} \right) - \rho_{ijk}^t \left(\frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} \right) - w_{ijk}^t \left(\frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \right) \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 \rho}{\partial t^2} = & -\frac{\partial \rho}{\partial t} \frac{\partial u}{\partial x} - \rho \frac{\partial^2 u}{\partial x \partial t} - \frac{\partial u}{\partial t} \frac{\partial \rho}{\partial x} - u \frac{\partial^2 \rho}{\partial x \partial t} - \frac{\partial \rho}{\partial t} \frac{\partial v}{\partial y} - \rho \frac{\partial^2 v}{\partial t \partial y} - \frac{\partial v}{\partial t} \frac{\partial \rho}{\partial y} - v \frac{\partial^2 \rho}{\partial t \partial y} - \frac{\partial \rho}{\partial t} \frac{\partial w}{\partial z} - \rho \frac{\partial^2 w}{\partial t \partial z} \\ & - \frac{\partial w}{\partial t} \frac{\partial \rho}{\partial z} - w \frac{\partial^2 \rho}{\partial t \partial z} \end{aligned}$$

$$\begin{aligned} \frac{\rho_{ijk}^{t+1} - 2\rho_{ijk}^t + \rho_{ijk}^{t-1}}{\Delta t^2} = & -\frac{\rho_{ijk}^{t+1} - \rho_{ijk}^{t-1}}{2\Delta t} \times \frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} - \rho_{ijk}^t \frac{u_{i+1jk}^{t+1} - u_{i-1jk}^{t+1} - u_{i+1jk}^{t-1} + u_{i-1jk}^{t-1}}{4\Delta x \Delta t} \\ & - \frac{u_{ijk}^{t+1} - u_{ijk}^{t-1}}{2\Delta t} \times \frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} - u_{ijk}^t \frac{\rho_{i+1jk}^{t+1} - \rho_{i-1jk}^{t+1} - \rho_{i+1jk}^{t-1} + \rho_{i-1jk}^{t-1}}{4\Delta x \Delta t} \\ & - \frac{\rho_{ijk}^{t+1} - \rho_{ijk}^{t-1}}{2\Delta t} \times \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} - \rho_{ijk}^t \frac{v_{ij+1k}^{t+1} - v_{ij-1k}^{t+1} - v_{ij+1k}^{t-1} + v_{ij-1k}^{t-1}}{4\Delta y \Delta t} \\ & - \frac{v_{ijk}^{t+1} - v_{ijk}^{t-1}}{2\Delta t} \times \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - v_{ijk}^t \frac{\rho_{ij+1k}^{t+1} - \rho_{ij-1k}^{t+1} - \rho_{ij+1k}^{t-1} + \rho_{ij-1k}^{t-1}}{4\Delta y \Delta t} \\ & - \frac{\rho_{ijk}^{t+1} - \rho_{ijk}^{t-1}}{2\Delta t} \times \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} - \rho_{ijk}^t \frac{w_{ijk+1}^{t+1} - w_{ijk-1}^{t+1} - w_{ijk+1}^{t-1} + w_{ijk-1}^{t-1}}{4\Delta z \Delta t} \\ & - \frac{w_{ijk}^{t+1} - w_{ijk}^{t-1}}{2\Delta t} \times \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} - w_{ijk}^t \frac{\rho_{ijk+1}^{t+1} - \rho_{ijk-1}^{t+1} - \rho_{ijk+1}^{t-1} + \rho_{ijk-1}^{t-1}}{4\Delta z \Delta t} \end{aligned}$$

Species k continuity equation

$$\frac{\partial \rho Y_k}{\partial t} + \frac{\partial \rho u_i Y_k}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D_k \frac{\partial Y_k}{\partial x_i} \right) + \omega_k$$

$$\begin{aligned} Y_k \frac{\partial \rho}{\partial t} + \rho \frac{\partial Y_k}{\partial t} + \frac{\partial \rho u Y_k}{\partial x} + \frac{\partial \rho v Y_k}{\partial y} + \frac{\partial \rho w Y_k}{\partial z} \\ = \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2} \right) + D_k \frac{\partial Y_k}{\partial x} \frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2} \right) + D_k \frac{\partial Y_k}{\partial y} \frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2} \right) + D_k \frac{\partial Y_k}{\partial z} \frac{\partial \rho}{\partial z} + \omega_k \end{aligned}$$

$$\begin{aligned}
& Y_k \frac{\partial \rho}{\partial t} + \rho \frac{\partial Y_k}{\partial t} + \frac{\partial \rho u Y_k}{\partial x} + \frac{\partial \rho v Y_k}{\partial y} + \frac{\partial \rho w Y_k}{\partial z} \\
&= \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2} \right) + D_k \frac{\partial Y_k}{\partial x} \frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2} \right) + D_k \frac{\partial Y_k}{\partial y} \frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2} \right) + D_k \frac{\partial Y_k}{\partial z} \frac{\partial \rho}{\partial z} + \omega_k \\
\rho \frac{\partial Y_k}{\partial t} &= -Y_k \frac{\partial \rho}{\partial t} - \rho u \frac{\partial Y_k}{\partial x} - u Y_k \frac{\partial \rho}{\partial x} - \rho Y_k \frac{\partial u}{\partial x} - \rho v \frac{\partial Y_k}{\partial y} - v Y_k \frac{\partial \rho}{\partial y} - \rho Y_k \frac{\partial v}{\partial y} - \rho w \frac{\partial Y_k}{\partial z} - w Y_k \frac{\partial \rho}{\partial z} - \rho Y_k \frac{\partial w}{\partial z} \\
&+ \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2} \right) + D_k \frac{\partial Y_k}{\partial x} \frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2} \right) + D_k \frac{\partial Y_k}{\partial y} \frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2} \right) + D_k \frac{\partial Y_k}{\partial z} \frac{\partial \rho}{\partial z} + \omega_k \\
\rho \frac{\partial Y_k}{\partial t} &= -Y_k \frac{\partial \rho}{\partial t} - \rho u \frac{Y_{k_{i+1}j k}^t - Y_{k_{i-1}j k}^t}{2\Delta x} - u Y_k \frac{\rho_{i+1j k}^t - \rho_{i-1j k}^t}{2\Delta x} - \rho Y_k \frac{u_{i+1j k}^t - u_{i-1j k}^t}{2\Delta x} - \rho v \frac{Y_{k_{ij+1}k}^t - Y_{k_{ij-1}k}^t}{2\Delta y} \\
&- v Y_k \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - \rho Y_k \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} - \rho w \frac{Y_{k_{ijk+1}}^t - Y_{k_{ijk-1}}^t}{2\Delta z} - w Y_k \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \\
&- \rho Y_k \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} + \rho D_k \left(\frac{Y_{k_{i+1}j k}^t - Y_{k_{ijk}}^t + Y_{k_{i-1}j k}^t}{\Delta x^2} \right) + D_k \frac{Y_{k_{i+1}j k}^t - Y_{k_{i-1}j k}^t}{2\Delta x} \frac{\rho_{i+1j k}^t - \rho_{i-1j k}^t}{2\Delta x} \\
&+ \rho D_k \left(\frac{Y_{k_{ij+1}k}^t - Y_{k_{ijk}}^t + Y_{k_{ij-1}k}^t}{\Delta y^2} \right) + D_k \frac{Y_{k_{ij+1}k}^t - Y_{k_{ij-1}k}^t}{2\Delta y} \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} \\
&+ \rho D_k \left(\frac{Y_{k_{ijk+1}}^t - Y_{k_{ijk}}^t + Y_{k_{ijk-1}}^t}{\Delta z^2} \right) + D_k \frac{Y_{k_{ijk+1}}^t - Y_{k_{ijk-1}}^t}{2\Delta z} \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} + \omega_k
\end{aligned}$$

38.2 The momentum equation for fluid mixture (momentum conservation)

- **Velocity u:**

$$\begin{aligned}
& \frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho u v)}{\partial y} + \frac{\partial(\rho u w)}{\partial z} \\
&= -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x} \left(\lambda \nabla \cdot \vec{V} + 2\mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left[\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right] + \frac{\partial}{\partial z} \left[\mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \right] + \rho \sum_{k=1}^N Y_k f_x
\end{aligned}$$

$$\begin{aligned}
& \rho \frac{\partial u}{\partial t} + u \frac{\partial \rho}{\partial t} + 2\rho u \frac{\partial u}{\partial x} + u^2 \frac{\partial \rho}{\partial x} + \rho u \frac{\partial v}{\partial y} + \rho v \frac{\partial u}{\partial y} + uv \frac{\partial \rho}{\partial y} + \rho u \frac{\partial w}{\partial z} + \rho w \frac{\partial u}{\partial z} + uw \frac{\partial \rho}{\partial z} \\
&= -\frac{\partial p}{\partial x} + \lambda \frac{\partial^2 u}{\partial x^2} + \lambda \frac{\partial^2 v}{\partial x \partial y} + \lambda \frac{\partial^2 w}{\partial x \partial z} + 2\mu \frac{\partial^2 u}{\partial x^2} + \mu \frac{\partial^2 v}{\partial y \partial x} + \mu \frac{\partial^2 u}{\partial y^2} + \mu \frac{\partial^2 u}{\partial z^2} + \mu \frac{\partial^2 w}{\partial z \partial x} + \rho \sum_{k=1}^N Y_k f_x
\end{aligned}$$

$$\begin{aligned}
\rho \frac{\partial u}{\partial t} = & -u \frac{\partial \rho}{\partial t} - 2\rho u \frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} - u^2 \frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} - \rho u \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} - \rho v \frac{u_{ij+1k}^t - u_{ij-1k}^t}{2\Delta y} \\
& - uv \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - \rho u \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} - \rho w \frac{u_{ijk+1}^t - u_{ijk-1}^t}{2\Delta z} - uw \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \\
& - \frac{p_{i+1jk}^t - p_{i-1jk}^t}{2\Delta x} + \lambda \frac{u_{i+1jk}^t - u_{ijk}^t + u_{i-1jk}^t}{\Delta x^2} + \lambda \frac{v_{i+1j+1k}^t - v_{i+1j-1k}^t - v_{i-1j+1k}^t + v_{i-1j-1k}^t}{4\Delta x \Delta y} \\
& + \lambda \frac{w_{i+1jk+1}^t - w_{i+1jk-1}^t - w_{i-1jk+1}^t + w_{i-1jk-1}^t}{4\Delta x \Delta z} + 2\mu \frac{u_{i+1jk}^t - u_{ijk}^t + u_{i-1jk}^t}{\Delta x^2} \\
& + \mu \frac{v_{i+1j+1k}^t - v_{i+1j-1k}^t - v_{i-1j+1k}^t + v_{i-1j-1k}^t}{4\Delta x \Delta y} + \mu \frac{u_{ij+1k}^t - u_{ijk}^t + u_{ij-1k}^t}{\Delta y^2} \\
& + \mu \frac{u_{ijk+1}^t - u_{ijk}^t + u_{ijk-1}^t}{\Delta z^2} + \mu \frac{w_{i+1jk+1}^t - w_{i+1jk-1}^t - w_{i-1jk+1}^t + w_{i-1jk-1}^t}{4\Delta x \Delta z} + \rho \sum_{k=1}^N Y_k f_x
\end{aligned}$$

• **Velocity v:**

$$\begin{aligned}
\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho v u)}{\partial x} + \frac{\partial(\rho v^2)}{\partial y} + \frac{\partial(\rho v w)}{\partial z} \\
= -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x} \left[\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right] + \frac{\partial}{\partial y} \left(\lambda \nabla \cdot \vec{V} + 2\mu \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial z} \left[\mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \right] + \rho \sum_{k=1}^N Y_k f_{yk}
\end{aligned}$$

$$\begin{aligned}
\rho \frac{\partial v}{\partial t} + v \frac{\partial \rho}{\partial t} + \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial u}{\partial x} + uv \frac{\partial \rho}{\partial x} + 2\rho v \frac{\partial v}{\partial y} + v^2 \frac{\partial \rho}{\partial y} + \rho v \frac{\partial w}{\partial z} + \rho w \frac{\partial v}{\partial z} + vw \frac{\partial \rho}{\partial z} \\
= -\frac{\partial p}{\partial y} + \mu \frac{\partial^2 v}{\partial x^2} + \mu \frac{\partial^2 u}{\partial x \partial y} + \lambda \frac{\partial^2 u}{\partial y \partial x} + \lambda \frac{\partial^2 v}{\partial y^2} + \lambda \frac{\partial^2 w}{\partial y \partial z} + 2\mu \frac{\partial^2 v}{\partial y^2} + \mu \frac{\partial^2 v}{\partial z^2} + \mu \frac{\partial^2 w}{\partial z \partial y} + \rho \sum_{k=1}^N Y_k f_{yk}
\end{aligned}$$

$$\begin{aligned}
\rho \frac{\partial v}{\partial t} = & -v \frac{\partial \rho}{\partial t} - \rho u \frac{v_{i+1jk}^t - v_{i-1jk}^t}{2\Delta x} - \rho v \frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} - uv \frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} - 2\rho v \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} \\
& - v^2 \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - \rho v \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} - \rho w \frac{v_{ijk+1}^t - v_{ijk-1}^t}{2\Delta z} - vw \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \\
& - \frac{p_{ij+1k}^t - p_{ij-1k}^t}{2\Delta y} + \mu \frac{v_{i+1jk}^t - v_{ijk}^t + v_{i-1jk}^t}{\Delta x^2} + \mu \frac{u_{i+1j+1k}^t - u_{i+1j-1k}^t - u_{i-1j+1k}^t + u_{i-1j-1k}^t}{4\Delta x \Delta y} \\
& + \lambda \frac{u_{i+1j+1k}^t - u_{i+1j-1k}^t - u_{i-1j+1k}^t + u_{i-1j-1k}^t}{4\Delta x \Delta y} + \lambda \frac{v_{ij+1k}^t - v_{ijk}^t + v_{ij-1k}^t}{\Delta y^2} \\
& + \lambda \frac{w_{ij+1k+1}^t - w_{ij+1k-1}^t - w_{ij-1k+1}^t + w_{ij-1k-1}^t}{4\Delta y \Delta z} + 2\mu \frac{v_{ij+1k}^t - v_{ijk}^t + v_{ij-1k}^t}{\Delta y^2} \\
& + \mu \frac{v_{ijk+1}^t - v_{ijk}^t + v_{ijk-1}^t}{\Delta z^2} + \mu \frac{w_{ij+1k+1}^t - w_{ij+1k-1}^t - w_{ij-1k+1}^t + w_{ij-1k-1}^t}{4\Delta y \Delta z} + \rho \sum_{k=1}^N Y_k f_{yk}
\end{aligned}$$

• **Velocity w:**

$$\begin{aligned}
& \frac{\partial(\rho w)}{\partial t} + \frac{\partial(\rho w u)}{\partial x} + \frac{\partial(\rho v w)}{\partial y} + \frac{\partial(\rho w^2)}{\partial z} \\
&= -\frac{\partial p}{\partial z} + \frac{\partial}{\partial x} \left[\mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \right] + \frac{\partial}{\partial y} \left[\mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \right] + \frac{\partial}{\partial z} \left(\lambda \nabla \cdot \vec{V} + 2\mu \frac{\partial w}{\partial z} \right) + \rho \sum_{k=1}^N Y_k f_{zk} \\
\rho \frac{\partial w}{\partial t} + w \frac{\partial \rho}{\partial t} + \rho u \frac{\partial w}{\partial x} + \rho w \frac{\partial u}{\partial x} + u w \frac{\partial \rho}{\partial x} + \rho v \frac{\partial w}{\partial y} + \rho w \frac{\partial v}{\partial y} + v w \frac{\partial \rho}{\partial y} + 2\rho w \frac{\partial w}{\partial z} + w^2 \frac{\partial \rho}{\partial z} \\
&= -\frac{\partial p}{\partial z} + \mu \frac{\partial^2 u}{\partial x \partial z} + \mu \frac{\partial^2 w}{\partial x^2} + \mu \frac{\partial^2 v}{\partial y \partial z} + \mu \frac{\partial^2 w}{\partial y^2} + \lambda \frac{\partial^2 u}{\partial z \partial x} + \lambda \frac{\partial^2 v}{\partial z \partial y} + \lambda \frac{\partial^2 w}{\partial z^2} + 2\mu \frac{\partial^2 w}{\partial z^2} + \rho \sum_{k=1}^N Y_k f_{zk} \\
\rho \frac{w}{\partial t} &= -w \frac{\partial \rho}{\partial t} - \rho u \frac{w_{i+1jk}^t - w_{i-1jk}^t}{2\Delta x} - \rho w \frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} - u w \frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} - \rho v \frac{w_{ij+1k}^t - w_{ij-1k}^t}{2\Delta y} \\
&\quad - \rho w \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} - w v \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - 2\rho w \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} - w^2 \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \\
&\quad - \frac{p_{ijk+1}^t - p_{ijk-1}^t}{2\Delta z} + \mu \frac{u_{i+1jk+1}^t - u_{i+1jk-1}^t - u_{i-1jk+1}^t + u_{i-1jk-1}^t}{4\Delta x \Delta z} + \mu \frac{w_{i+1jk}^t - w_{ijk}^t + w_{i-1jk}^t}{\Delta x^2} \\
&\quad + \mu \frac{v_{ij+1k+1}^t - v_{ij+1k-1}^t - v_{ij-1k+1}^t + v_{ij-1k-1}^t}{4\Delta y \Delta z} + \mu \frac{w_{ij+1k}^t - w_{ijk}^t + w_{ij-1k}^t}{\Delta y^2} \\
&\quad + \lambda \frac{u_{i+1jk+1}^t - u_{i+1jk-1}^t - u_{i-1jk+1}^t + u_{i-1jk-1}^t}{4\Delta x \Delta z} + \lambda \frac{v_{ij+1k+1}^t - v_{ij+1k-1}^t - v_{ij-1k+1}^t + v_{ij-1k-1}^t}{4\Delta y \Delta z} \\
&\quad + \lambda \frac{w_{ijk+1}^t - w_{ijk}^t + w_{ijk-1}^t}{\Delta z^2} + 2\mu \frac{w_{ijk+1}^t - w_{ijk}^t + w_{ijk-1}^t}{\Delta z^2} + \rho \sum_{k=1}^N Y_k f_{zk}
\end{aligned}$$

38.3 The internal energy equation (energy conservation)

$$\frac{\partial(\rho e)}{\partial t} + \nabla \cdot (\rho e \vec{V}) = \dot{w}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_i} (\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{ki} (u_i + V_{ki})$$

$$\begin{aligned}
& \frac{\partial(\rho e)}{\partial t} + \frac{\partial \rho e u}{\partial x} + \frac{\partial \rho e v}{\partial y} + \frac{\partial \rho e w}{\partial z} \\
&= -\sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k - \frac{\partial q}{\partial x} - \frac{\partial q}{\partial y} - \frac{\partial q}{\partial z} + \frac{\partial((\tau_{ix} - p)u)}{\partial x} + \frac{\partial((\tau_{iy} - p)v)}{\partial y} + \frac{\partial((\tau_{iz} - p)w)}{\partial z} + \dot{Q} \\
&\quad + \rho \sum_{k=1}^N Y_k f_{kx} (u + V_{ki})
\end{aligned}$$

$$\begin{aligned}
& \frac{\partial(\rho e)}{\partial t} + \frac{\partial \rho e u}{\partial x} + \frac{\partial \rho e v}{\partial y} + \frac{\partial \rho e w}{\partial z} \\
&= -\sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k - \frac{\partial q}{\partial x} - \frac{\partial q}{\partial y} - \frac{\partial q}{\partial z} + \tau_{xx} \frac{\partial u}{\partial x} + \tau_{yx} \frac{\partial u}{\partial y} + \tau_{zx} \frac{\partial u}{\partial z} - p \frac{\partial u}{\partial x} + \tau_{xy} \frac{\partial v}{\partial x} + \tau_{yy} \frac{\partial v}{\partial y} \\
&\quad + \tau_{zy} \frac{\partial v}{\partial z} - p \frac{\partial v}{\partial y} + \tau_{xz} \frac{\partial w}{\partial x} + \tau_{yz} \frac{\partial w}{\partial y} + \tau_{zz} \frac{\partial w}{\partial z} - p \frac{\partial w}{\partial z} + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{ki} (u + V_{ki})
\end{aligned}$$

$$\begin{aligned}
& \frac{\partial(\rho e)}{\partial t} + \frac{\partial \rho e u}{\partial x} + \frac{\partial \rho e v}{\partial y} + \frac{\partial \rho e w}{\partial z} \\
&= - \sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k - \frac{\partial \left(-\lambda \frac{\partial T}{\partial x} + \sum_{k=1}^N h_k Y_k V_{k,x} \right)}{\partial x} - \frac{\partial \left(-\lambda \frac{\partial T}{\partial y} + \sum_{k=1}^N h_k Y_k V_{k,y} \right)}{\partial y} \\
&\quad - \frac{\partial \left(-\lambda \frac{\partial T}{\partial z} + \sum_{k=1}^N h_k Y_k V_{k,z} \right)}{\partial z} + \left(-\frac{2}{3} \mu \frac{\partial u_k}{\partial x} + \mu \left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x} \right) \right) \frac{\partial u}{\partial x} + \left(\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right) \frac{\partial u}{\partial y} \\
&\quad + \left(\mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \right) \frac{\partial u}{\partial z} + \left(\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right) \frac{\partial v}{\partial x} + \left(-\frac{2}{3} \mu \frac{\partial v_k}{\partial y} + \mu \left(2 \frac{\partial v}{\partial y} \right) \right) \frac{\partial v}{\partial y} + \left(\mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \right) \frac{\partial v}{\partial z} \\
&\quad + \left(\mu \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \right) \frac{\partial w}{\partial x} + \left(\mu \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \right) \right) \frac{\partial w}{\partial y} + \left(-\frac{2}{3} \mu \frac{\partial w_k}{\partial z} + \mu \left(2 \frac{\partial w}{\partial z} \right) \right) \frac{\partial w}{\partial z} \\
&\quad - p \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{ki} (u + V_{ki})
\end{aligned}$$

$$\begin{aligned}
& \rho \frac{\partial e}{\partial t} + e \frac{\partial \rho}{\partial t} + \rho e \frac{\partial u}{\partial x} + e u \frac{\partial \rho}{\partial x} + \rho u \frac{\partial e}{\partial x} + \rho e \frac{\partial v}{\partial y} + e v \frac{\partial \rho}{\partial y} + \rho v \frac{\partial e}{\partial y} + \rho e \frac{\partial w}{\partial z} + e w \frac{\partial \rho}{\partial z} + \rho w \frac{\partial e}{\partial z} = \\
&= - \sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k - \lambda \frac{\partial^2 T}{\partial x^2} - \frac{\partial}{\partial x} \sum_{k=1}^N h_k Y_k V_{k,x} - \lambda \frac{\partial^2 T}{\partial y^2} + \frac{\partial}{\partial y} \sum_{k=1}^N h_k Y_k V_{k,y} - \lambda \frac{\partial^2 T}{\partial z^2} \\
&\quad + \frac{\partial}{\partial z} \sum_{k=1}^N h_k Y_k V_{k,z} - \frac{2}{3} \mu \frac{\partial^2 u}{\partial x^2} + 2\mu \frac{\partial^2 u}{\partial x^2} + \mu \frac{\partial^2 u}{\partial y^2} + \mu \frac{\partial v}{\partial x} \frac{\partial u}{\partial y} + \mu \frac{\partial^2 u}{\partial z^2} + \mu \frac{\partial w}{\partial x} \frac{\partial u}{\partial z} + \mu \frac{\partial^2 v}{\partial x^2} + \mu \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \\
&\quad - \frac{2}{3} \mu \frac{\partial^2 v}{\partial y^2} + 2\mu \frac{\partial^2 v}{\partial y^2} + \mu \frac{\partial^2 v}{\partial z^2} + \mu \frac{\partial w}{\partial y} \frac{\partial v}{\partial z} + \mu \frac{\partial^2 w}{\partial x^2} + \mu \frac{\partial u}{\partial z} \frac{\partial w}{\partial x} + \mu \frac{\partial^2 w}{\partial y^2} + \mu \frac{\partial v}{\partial z} \frac{\partial w}{\partial y} - \frac{2}{3} \mu \frac{\partial^2 w}{\partial z^2} \\
&\quad + \mu 2 \frac{\partial^2 w}{\partial z^2} - p \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{kx} u + \rho \sum_{k=1}^N Y_k f_{kx} V_{kx} + \rho \sum_{k=1}^N Y_k f_{ky} v \\
&\quad + \rho \sum_{k=1}^N Y_k f_{ky} V_{ky} + \rho \sum_{k=1}^N Y_k f_{kz} w + \rho \sum_{k=1}^N Y_k f_{kz} V_{kz}
\end{aligned}$$

$$V_{k,i} Y_k = -D_k \frac{\partial Y_k}{\partial x_i}$$

$$\begin{aligned}
& \rho \frac{\partial e}{\partial t} + e \frac{\partial \rho}{\partial t} + \rho e \frac{\partial u}{\partial x} + e u \frac{\partial \rho}{\partial x} + \rho u \frac{\partial e}{\partial x} + \rho e \frac{\partial v}{\partial y} + e v \frac{\partial \rho}{\partial y} + \rho v \frac{\partial e}{\partial y} + \rho e \frac{\partial w}{\partial z} + e w \frac{\partial \rho}{\partial z} + \rho w \frac{\partial e}{\partial z} \\
&= - \sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k - \lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) - D_k \sum_{k=1}^N h_k \frac{\partial^2 Y_k}{\partial x^2} - D_k \sum_{k=1}^N h_k \frac{\partial^2 Y_k}{\partial y^2} - D_k \sum_{k=1}^N h_k \frac{\partial^2 Y_k}{\partial z^2} \\
&\quad - \frac{2}{3} \mu \frac{\partial^2 u}{\partial x^2} + 2\mu \frac{\partial^2 u}{\partial x^2} + \mu \frac{\partial^2 u}{\partial y^2} + \mu \frac{\partial v}{\partial x} \frac{\partial u}{\partial y} + \mu \frac{\partial^2 u}{\partial z^2} + \mu \frac{\partial w}{\partial x} \frac{\partial u}{\partial z} + \mu \frac{\partial^2 v}{\partial x^2} + \mu \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} - \frac{2}{3} \mu \frac{\partial^2 v}{\partial y^2} \\
&\quad + 2\mu \frac{\partial^2 v}{\partial y^2} + \mu \frac{\partial^2 v}{\partial z^2} + \mu \frac{\partial w}{\partial y} \frac{\partial v}{\partial z} + \mu \frac{\partial^2 w}{\partial x^2} + \mu \frac{\partial u}{\partial z} \frac{\partial w}{\partial x} + \mu \frac{\partial^2 w}{\partial y^2} + \mu \frac{\partial v}{\partial z} \frac{\partial w}{\partial y} - \frac{2}{3} \mu \frac{\partial^2 w}{\partial z^2} + \mu 2 \frac{\partial^2 w}{\partial z^2} \\
&\quad - p \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) + \dot{Q} - D_k \rho \sum_{k=1}^N \frac{\partial Y_k}{\partial x} f_{kx} + \rho \sum_{k=1}^N Y_k f_{kx} u + \rho \sum_{k=1}^N Y_k f_{ky} v - D_k \rho \sum_{k=1}^N \frac{\partial Y_k}{\partial y} f_{ky} \\
&\quad + \rho \sum_{k=1}^N Y_k f_{kz} w - D_k \rho \sum_{k=1}^N \frac{\partial Y_k}{\partial z} f_{kz}
\end{aligned}$$

$$\begin{aligned}
\rho \frac{\partial e}{\partial t} + e \frac{\partial \rho}{\partial t} = & -\rho e \frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} - eu \frac{\rho_{i+1jk}^t - \rho_{i-1jk}^t}{2\Delta x} - \rho u \frac{e_{i+1jk}^t - e_{i-1jk}^t}{2\Delta x} - \rho e \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} \\
& - ev \frac{\rho_{ij+1k}^t - \rho_{ij-1k}^t}{2\Delta y} - \rho v \frac{e_{ij+1k}^t - e_{ij-1k}^t}{2\Delta y} - \rho e \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} - ew \frac{\rho_{ijk+1}^t - \rho_{ijk-1}^t}{2\Delta z} \\
& - \rho w \frac{e_{ijk+1}^t - e_{ijk-1}^t}{2\Delta z} - \sum_{k=1}^N \Delta h_{fk}^0 \dot{w}_k \\
& - \lambda \left(\frac{T_{i+1jk}^t - 2T_{ijk}^t + T_{i-1jk}^t}{\Delta x^2} + \frac{T_{ij+1k}^t - 2T_{ijk}^t + T_{ij-1k}^t}{\Delta y^2} + \frac{T_{ijk+1}^t - 2T_{ijk}^t + T_{ijk-1}^t}{\Delta z^2} \right) \\
& - D_k \sum_{k=1}^N h_k \frac{Y_{k i+1jk}^t - 2Y_{k ij}^t + Y_{k i-1jk}^t}{\Delta x^2} - D_k \sum_{k=1}^N h_k \frac{Y_{k ij+1k}^t - 2Y_{k ij}^t + Y_{k ij-1k}^t}{\Delta y^2} \\
& - D_k \sum_{k=1}^N h_k \frac{Y_{k ijk+1}^t - 2Y_{k ijk}^t + Y_{k ijk-1}^t}{\Delta z^2} + \frac{4}{3}\mu \frac{u_{i+1jk}^t - 2u_{ijk}^t + u_{i-1jk}^t}{\Delta x^2} + \mu \frac{u_{ij+1k}^t - 2u_{ijk}^t + u_{ij-1k}^t}{\Delta y^2} \\
& + 2\mu \frac{v_{i+1jk}^t - v_{i-1jk}^t}{2\Delta x} \frac{u_{ij+1k}^t - u_{ij-1k}^t}{2\Delta y} + \mu \frac{u_{ijk+1}^t - 2u_{ijk}^t + u_{ijk-1}^t}{\Delta z^2} \\
& + 2\mu \frac{w_{i+1jk}^t - w_{i-1jk}^t}{2\Delta x} \frac{u_{ijk+1}^t - u_{ijk-1}^t}{2\Delta z} + \mu \frac{v_{i+1jk}^t - 2v_{ijk}^t + v_{i-1jk}^t}{\Delta x^2} + \frac{4}{3}\mu \frac{v_{ij+1k}^t - 2v_{ijk}^t + v_{ij-1k}^t}{\Delta y^2} \\
& + \mu \frac{v_{ijk+1}^t - 2v_{ijk}^t + v_{ijk-1}^t}{\Delta z^2} + 2\mu \frac{w_{ij+1k}^t - w_{ij-1k}^t}{2\Delta y} \frac{v_{ijk+1}^t - v_{ijk-1}^t}{2\Delta z} + \mu \frac{w_{i+1jk}^t - 2w_{ijk}^t + w_{i-1jk}^t}{\Delta x^2} \\
& + \mu \frac{w_{ij+1k}^t - 2w_{ijk}^t + w_{ij-1k}^t}{\Delta y^2} + \frac{4}{3}\mu \frac{w_{ijk+1}^t - 2w_{ijk}^t + w_{ijk-1}^t}{\Delta z^2} \\
& - p \left(\frac{u_{i+1jk}^t - u_{i-1jk}^t}{2\Delta x} + \frac{v_{ij+1k}^t - v_{ij-1k}^t}{2\Delta y} + \frac{w_{ijk+1}^t - w_{ijk-1}^t}{2\Delta z} \right) + \dot{Q} - D_k \rho \sum_{k=1}^N \frac{Y_{k i+1jk}^t - Y_{k i-1jk}^t}{2\Delta x} f_{kx} \\
& + \rho \sum_{k=1}^N Y_k f_{kx} u + \rho \sum_{k=1}^N Y_k f_{ky} v + -D_k \rho \sum_{k=1}^N \frac{Y_{k ij+1k}^t - Y_{k ij-1k}^t}{2\Delta y} f_{ky} + \rho \sum_{k=1}^N Y_k f_{kz} w \\
& - D_k \rho \sum_{k=1}^N \frac{Y_{k ijk+1}^t - Y_{k ijk-1}^t}{2\Delta z} f_{kz}
\end{aligned}$$

38.4 Chemical relations

38.4.1 Mass reaction rate

$$\dot{\omega}_k = \sum_{j=1}^M \dot{\omega}_{kj} = W_k \sum_{j=1}^M \nu_{kj} Q_j$$

38.4.2 Rate of progress of reaction j:

$$Q_j = K_{fj} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu'_{kj}} - K_{rj} \prod_{k=1}^N \left(\frac{\rho Y_k}{W_k} \right)^{\nu''_{kj}}$$

38.4.3 Rate constants:

$$K_{fj} = A_{fj} T^{\beta_j} \exp\left(-\frac{E_j}{RT}\right) = A_{fj} T^{\beta_j} \exp\left(-\frac{T_{a_j}}{T}\right)$$

$$K_{rj} = \frac{K_{fj}}{\left(\frac{p_a}{RT}\right)^{\sum_{k=1}^N \nu_{kj}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right)}$$

38.4.4 Mixture density:¹²

$$\rho_m = (\rho_1 V_1 + \rho_2 V_2 + \dots + \rho_n V_n) / (V_1 + V_2 + \dots + V_n)$$

38.4.5 Mixture viscosity¹³

$$\mu_{ga} = \frac{\sum_{i=1}^N y_i \mu_i \sqrt{M_{gi}}}{\sum_{i=1}^N y_i \sqrt{M_{gi}}}$$

38.4.6 Species viscosity

$$\mu_g = K_1 \exp(X \rho^Y), \dots$$

$$K_1 = \frac{(0.00094 + 2 \times 10^{-6} M_g) T^{1.5}}{(209 + 19 M_g + T)}$$

$$X = 3.5 + \frac{986}{T} + 0.01 M_g$$

- $Y = 2.4 - 0.2X$
- μ_g = gas viscosity, cp
- ρ = gas density, g/cm³
- p = pressure, psia
- T = temperature °R
- M_g = gas molecular weight = 28.967 γ_g

38.4.7 Pressure:

$$p = \rho \frac{R}{W} T \quad \frac{1}{W} = \sum_{k=1}^N \frac{Y_k}{W_k}$$

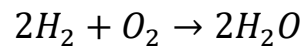
¹² https://www.engineeringtoolbox.com/gas-mixture-properties-d_586.html

¹³ https://petrowiki.org/Gas_viscosity

39 Application example: Hydrogene Oxygene Combustion

39.1 Hydrogen combustion characteristics

The global reaction for hydrogen combustion is as follow:



3 species are presented: H₂, O₂ and H₂O. Their characteristics are presented below.

39.1.1 Hydrogen characteristics

Values are considered at T=325 K

Density: $\rho=0.07603 \text{ kg/m}^3$

Molecular weight: M=2.016 g/mol

Mass: m=1 kg

Volume: V=1 m³

Reaction coefficients: a=2, b=0

39.1.2 Oxygen characteristics

Values are considered at T=325 K

Density: $\rho=1.2068 \text{ kg/m}^3$

Molecular weight: M=32 g/mol

Mass: m=1 kg

Volume: V=1 m³

Reaction coefficients: a=1, b=0

39.1.3 Water characteristics

Values are considered at T=325 K

Density: $\rho=0.6794 \text{ kg/m}^3$

Molecular weight: M=18 g/mol

Mass: m=1 kg

Volume: V=1 m³

Reaction coefficients: a=0, b=2

39.1.4 Mixture characteristics

Entropy differences: $\Delta S^0=-89 \text{ j/mol. k}$

Enthalpy differences: $\Delta H^0=-484000 \text{ j/mol}$

Activation energy: E=199911.52 j/mol

Chemical constant: A=1.7 E+13

Temperature constant: $\beta=0$

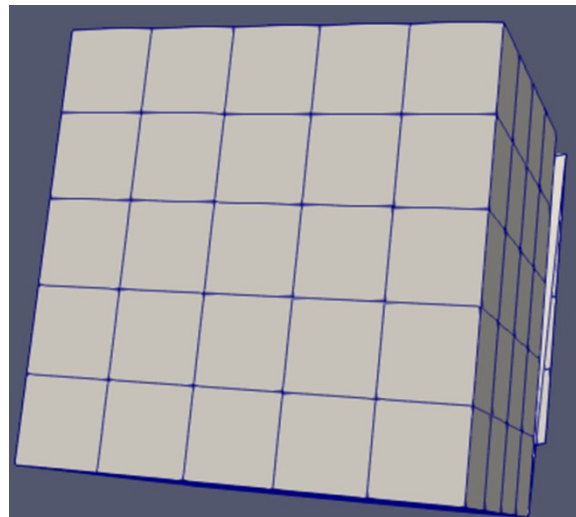
39.2 Program code

(the program used here isn't completely correct, an error analysis is needed)

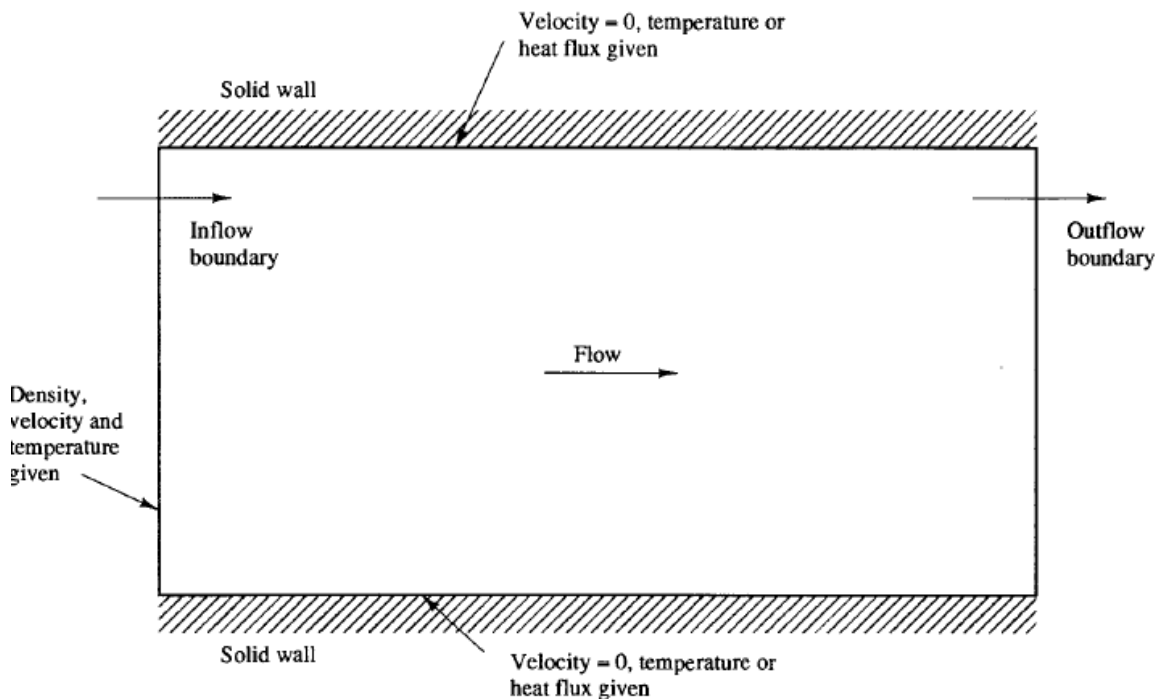
See Annex A.

The geometry adopted in our program is a cube. Each side is divided into 5 parts so it's a total of 125 points to be calculated. The fuel entrance is a square fixed at the following points (2,0,1), (2,0,2), (3,0,1) and (3,0,2). The rest of the points at $y=0$ are considered as the oxidizer entrance.

The meshing is shown in the figure below.

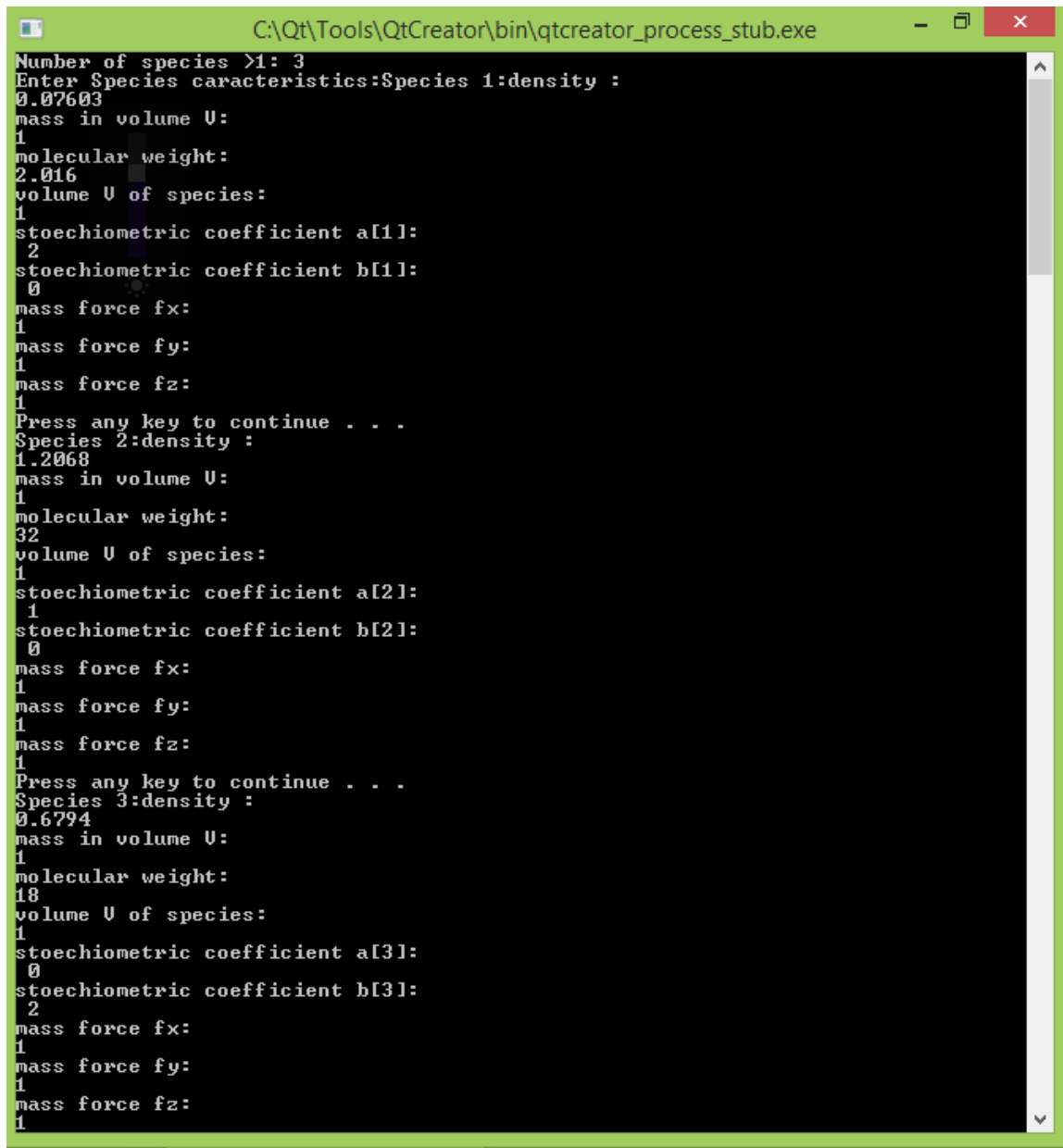


The boundary conditions are imposed as shown in the figure below¹⁴



¹⁴ "An introduction to computational fluid dynamics" H. K. VERSTEEG & W. MALALASEKERA

39.3 Program Input

A screenshot of a Qt Creator terminal window. The window title is "C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe". The terminal displays the following text:

```
Number of species >1: 3
Enter Species characteristics:Species 1:density :
0.07603
mass in volume U:
1
molecular weight:
2.016
volume U of species:
1
stoichiometric coefficient a[1]:
2
stoichiometric coefficient b[1]:
0
mass force fx:
1
mass force fy:
1
mass force fz:
1
Press any key to continue . . .
Species 2:density :
1.2068
mass in volume U:
1
molecular weight:
32
volume U of species:
1
stoichiometric coefficient a[2]:
1
stoichiometric coefficient b[2]:
0
mass force fx:
1
mass force fy:
1
mass force fz:
1
Press any key to continue . . .
Species 3:density :
0.6794
mass in volume U:
1
molecular weight:
18
volume U of species:
1
stoichiometric coefficient a[3]:
0
stoichiometric coefficient b[3]:
2
mass force fx:
1
mass force fy:
1
mass force fz:
1
```

```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Press any key to continue . . .
Mixture mass :
3
volume U of mixture:
3
activation energy E:
199911.52
Entropie:
-89
Enthalpi:
-484000
Rx constant A:
1700000000000000
Temp constant beta:
0
Enter Boundary conditions:
Inlet temperature:
300
inlet fuel Velocity u component:
83
inlet fuel Velocity v component:
90
inlet fuel Velocity w component:
100
inlet oxidizer Velocity u component:
2
inlet oxidizer Velocity v component:
4
inlet oxidizer Velocity w component:
7
Enter Boundary conditions:
wall temperature:
350
    
```

39.4 Program Results

```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Press any key to continue . . .
x,y,z,density,velocity_u,velocity_v,velocity_w
1.1.1.141.575180,1334.039673,1304.178711,3319.248047
1.1.2.229.875534,2565.744385,2687.626221,4863.949707
1.1.3.229.875534,3649.269043,3770.892578,5943.686035
1.1.4.141.575180,4729.168945,4850.921875,4824.661621
1.2.1.241.354294,5749.621094,6589.827637,7909.019531
1.2.2.329.654633,6981.762287,7974.874023,9454.932617
1.2.3.329.654633,8062.289551,9055.401367,10535.459961
1.2.4.241.354294,9142.159180,10136.561523,9417.179680
1.3.1.241.354294,10069.490234,10989.696289,12228.888672
1.3.2.329.654633,11300.049805,12293.161133,13773.219727
1.3.3.329.654633,12378.995117,13372.106445,14852.165039
1.3.4.241.354294,13457.940430,14452.342773,13732.960938
1.4.1.182.487396,14385.271484,14248.139648,16544.669922
1.4.2.270.787750,15615.831055,15631.604492,18089.001953
1.4.3.270.787750,16694.775391,16710.548828,19167.947266
1.4.4.182.487396,17776.326172,17793.392578,18051.347656
2.1.1.418.715302,19495.998047,19401.917969,21718.375000
2.1.2.507.015656,20731.013672,20795.736328,23273.103516
2.1.3.259.308990,20983.677734,21159.755859,23384.164062
2.1.4.171.008636,22067.078125,22237.818359,22263.990234
2.2.1.270.787750,23087.304688,23978.693359,25349.070312
2.2.2.359.088104,24322.318359,25365.322266,26896.994141
2.2.3.359.088104,25404.427734,26447.431641,27979.103516
2.2.4.270.787750,26487.830078,27530.832031,26863.494141
2.3.1.270.787750,27415.742188,28307.130859,29677.507812
2.3.2.359.088104,28650.755859,29693.759766,31225.431641
2.3.3.359.088104,29732.865234,30775.889141,32307.541016
2.3.4.270.787750,30016.267570,31059.269531,31191.921641
2.4.1.211.920837,31245.041016,31658.662109,34005.945312
2.4.2.300.221191,32980.054688,33045.289062,35553.871094
2.4.3.300.221191,34057.843750,34123.078125,36631.660156
2.4.4.211.920837,35134.917969,35200.152344,35509.722656
3.1.1.418.715302,36800.347656,36705.750000,39015.148438
3.1.2.507.015656,38029.035156,38093.242188,40563.546875
3.1.3.259.308990,38267.792969,38443.875000,40668.281250
3.1.4.171.008636,39344.867188,39515.609375,39541.781250
3.2.1.270.787750,40358.765625,41250.156250,42620.531250
3.2.2.359.088104,41587.453125,42630.457031,44162.128906
3.2.3.359.088104,42663.234375,43706.238281,45237.910156
3.2.4.270.787750,43740.304688,44783.308594,44115.972656
3.3.1.270.787750,44661.890625,45553.281250,46923.656250
3.3.2.359.088104,45890.578125,46933.582031,48465.253906
3.3.3.359.088104,46966.359375,48009.363281,49541.035156
3.3.4.270.787750,48043.427688,49086.433594,48419.097656
3.4.1.211.920837,48965.875000,48879.496894,51226.781250
3.4.2.300.221191,50194.562500,50259.796875,52768.370938
3.4.3.300.221191,51270.343750,51335.578125,53844.168156
3.4.4.211.920837,52347.417969,52412.652344,52722.222656
4.1.1.141.575180,52935.367188,53200.851562,55260.386719
4.1.2.229.875534,54164.054688,54579.988281,56801.210938
4.1.3.229.875534,55236.046875,55652.242188,57876.992188
    
```

```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
4,1,3,229.875534,55236.046875,55652.242188,57876.992188
4,1,4,141.575180,56313.117188,56728.152344,56754.281250
4,2,1,241.354294,57326.929688,58462.656250,59833.031250
4,2,2,329.654633,58555.617188,59842.957031,61374.628906
4,2,3,329.654633,59631.398438,60918.738281,62450.410156
4,2,4,241.354294,60708.472656,61995.808594,61328.472656
4,3,1,241.354294,61630.054688,62765.781250,64136.156250
4,3,2,329.654633,62858.742188,64146.082031,65677.750000
4,3,3,329.654633,63934.523438,65221.863281,66753.531250
4,3,4,241.354294,65011.597656,66298.937500,65631.593750
4,4,1,182.487396,65936.867188,66094.828125,68442.109375
4,4,2,270.787750,67178.210938,67487.781250,69996.359375
4,4,3,270.787750,68266.648438,68576.218750,71084.796875
4,4,4,182.487396,69356.382812,69665.945312,69975.515625
Press any key to continue
1,1,1,-2994276.750000,-385692096.000000,-224414784.000000,-1252813440.000000
1,1,2,-4899579.000000,-1683834240.000000,-1224401152.000000,-1088825856.000000
1,1,3,-4266908.000000,-1739932416.000000,-2387076096.000000,1490595200.000000
1,1,4,-2291173.750000,-1898906368.000000,-2873539840.000000,2527458304.000000
1,2,1,-7456676.500000,-4483110400.000000,-2289558016.000000,-10223842304.000000
1,2,2,-8073100.500000,-8712872960.000000,-3301702400.000000,-4076615424.000000
1,2,3,-7173767.500000,-11597544448.000000,-4239934208.000000,4732786688.000000
1,2,4,-3275619.250000,-11151495168.000000,-5236688896.000000,14178667520.000000
1,3,1,-8886769.000000,-13628271616.000000,3229327616.000000,-24389398528.000000
1,3,2,-8658380.000000,-22712696832.000000,4202843392.000000,-8597416960.000000
1,3,3,-7292427.000000,-27237795840.000000,4989513728.000000,9452112896.000000
1,3,4,-1984248.250000,-24074194944.000000,5944504832.000000,30233681920.000000
1,4,1,-5547210.000000,-21486802944.000000,23490904064.000000,-36363505664.000000
1,4,2,-3722049.000000,-35944792064.000000,39225692160.000000,-14586693632.000000
1,4,3,-1700839.125000,-41067790336.000000,44844208128.000000,15980102656.000000
1,4,4,2516511.500000,-32416546816.000000,37037469696.000000,42902323200.000000
2,1,1,-28067986.000000,-53337755648.000000,-51633471488.000000,-118751469568.000000
2,1,2,-20570214.000000,-60073312256.000000,-78126694400.000000,42417115136.000000
2,1,3,-7952650.500000,-7102490112.000000,-80469393408.000000,90864861184.000000
2,1,4,-3815113.250000,-7567226368.000000,-66719780864.000000,62020276224.000000
2,2,1,-12575873.000000,-8618021888.000000,41542221824.000000,-113897103360.000000
2,2,2,-7952018.000000,-9234143232.000000,46888349696.000000,-32409608192.000000
2,2,3,-8390575.000000,-10056889344.000000,-35396378624.000000,33907638272.000000
2,2,4,782204.375000,-10524505088.000000,-37909925888.000000,126766268416.000000
2,3,1,-14762243.000000,-12042323968.000000,22473662464.000000,-156028796928.000000
2,3,2,-8987528.000000,-12706935808.000000,25231097856.000000,-43594194944.000000
2,3,3,-5687045.500000,-13656201216.000000,27132487680.000000,45306441728.000000
2,3,4,4646202.500000,-14130817024.000000,29589311488.000000,171018584064.000000
2,4,1,-5710368.500000,-15436293120.000000,131461554176.000000,-170066886656.000000

```

```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
2,4,1,-5710368.500000,-15436293120.000000,131461554176.000000,-170066886656.0000
00
2,4,2,2653620.000000,-16091621376.000000,192079986688.000000,-55725838336.000000
2,4,3,6578786.500000,-17134432256.000000,204842303488.000000,59008507904.000000
2,4,4,12965449.000000,-17541638144.000000,163897065472.000000,185283903488.000000
0
3,1,1,-33754572.000000,185116295168.000000,-184051335168.000000,-382682529792.00
0000
3,1,2,-18013690.000000,198549946368.000000,-261454331904.000000,129374887936.000
000
3,1,3,-9458270.000000,19966375936.000000,-264970993664.000000,275282427904.000000
0
3,1,4,-3016610.250000,21867388928.000000,-210045386752.000000,196215095296.000000
0
3,2,1,-15694597.000000,22143356928.000000,123603509248.000000,-321412169728.0000
00
3,2,2,-6387816.000000,24451809280.000000,133084733440.000000,-86809010176.000000
3,2,3,-7409389.000000,25743345664.000000,-96028303360.000000,89172312064.000000
3,2,4,6396517.500000,28050311168.000000,-99690078208.000000,342479863808.000000
3,3,1,-18515686.000000,27210225664.000000,58807599104.000000,-389511118848.000000
0
3,3,2,-8064191.500000,29863753728.000000,63625408512.000000,-104470446080.000000
3,3,3,-3210342.250000,31309406208.000000,66607013888.000000,107042463744.000000
3,3,4,11747691.000000,33947805696.000000,70805266432.000000,412656304128.000000
3,4,1,-4731455.500000,34088067072.000000,313941262336.000000,-385419149312.000000
0
3,4,2,8804008.000000,37189496832.000000,444866199552.000000,-122242138112.000000
3,4,3,14276499.000000,38827012096.000000,464144302080.000000,127963373568.000000
3,4,4,22752110.000000,41852289024.000000,363873796096.000000,408926978048.000000
4,1,1,-7204224.000000,572271755264.000000,-339539361792.000000,-341710962688.000
000
4,1,2,-707065.125000,731261239296.000000,-487761608704.000000,-142475902976.0000
00
4,1,3,-2182722.000000,384252182528.000000,-507112390656.000000,146635587584.0000
00
4,1,4,6687717.000000,260918870016.000000,-382744297472.000000,359207895040.000000
0
4,2,1,-8699864.000000,432629940224.000000,-170697457664.000000,-578015789056.000
000
4,2,2,9492634.000000,604571566080.000000,-177234739200.000000,-164855119872.0000
00
4,2,3,15711604.000000,627010240512.000000,-183638491136.000000,172745768960.0000
00
4,2,4,26536082.000000,489013182464.000000,-188024143872.000000,609331380224.0000
00
4,3,1,-3829190.250000,500088111104.000000,114581274624.000000,-664072224768.0000
00

```

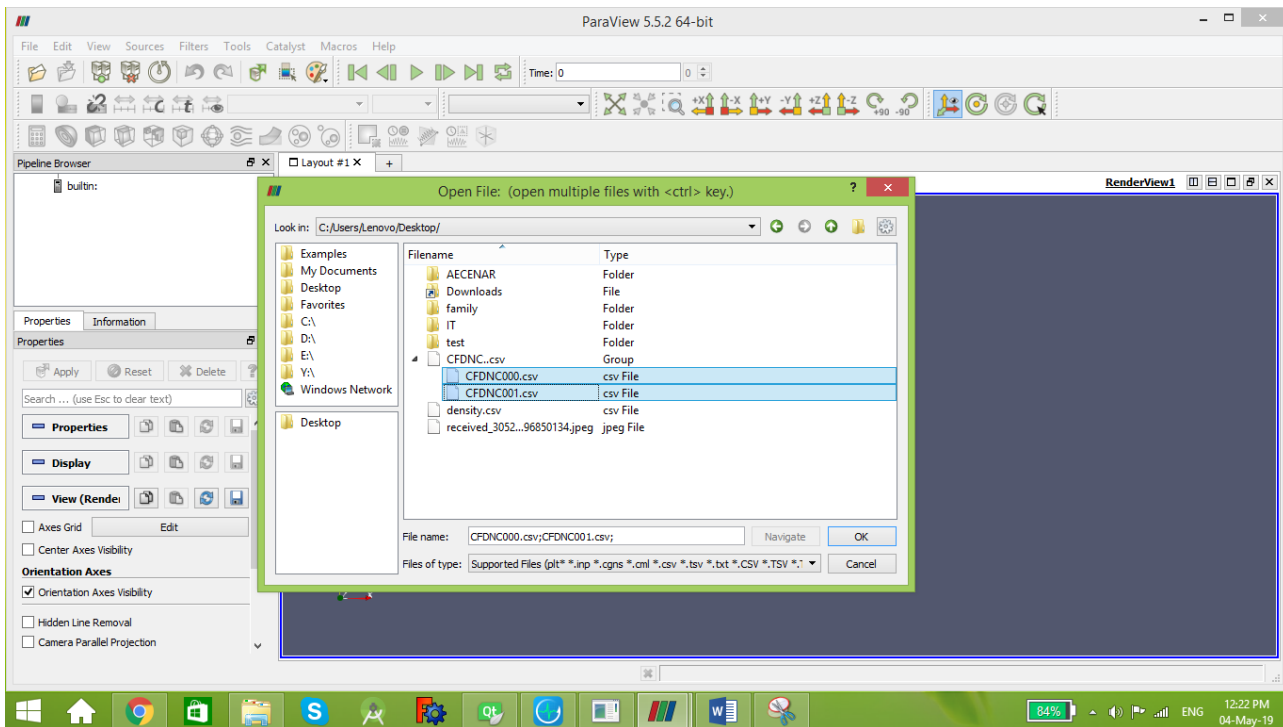
```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
00
4,3,1,-3829190.250000,500088111104.000000,114581274624.000000,-664072224768.0000
00
4,3,2,15962115.000000,696762040320.000000,121892421632.000000,-188706045952.0000
00
4,3,3,22644880.000000,720835510272.000000,126040203264.000000,197441716224.000000
0
4,3,4,34114928.000000,560857088000.000000,132321968128.000000,697905053696.000000
0
4,4,1,9317290.000000,446782275584.000000,512881229824.000000,-615627816960.000000
0
4,4,2,32844760.000000,665425936384.000000,738360033280.000000,-211618152448.0000
00
4,4,3,40061836.000000,687180939264.000000,762391363584.000000,226396782592.000000
0
4,4,4,41927180.000000,499296894976.000000,574797971456.000000,651601444864.000000
0
Press any key to continue . . .

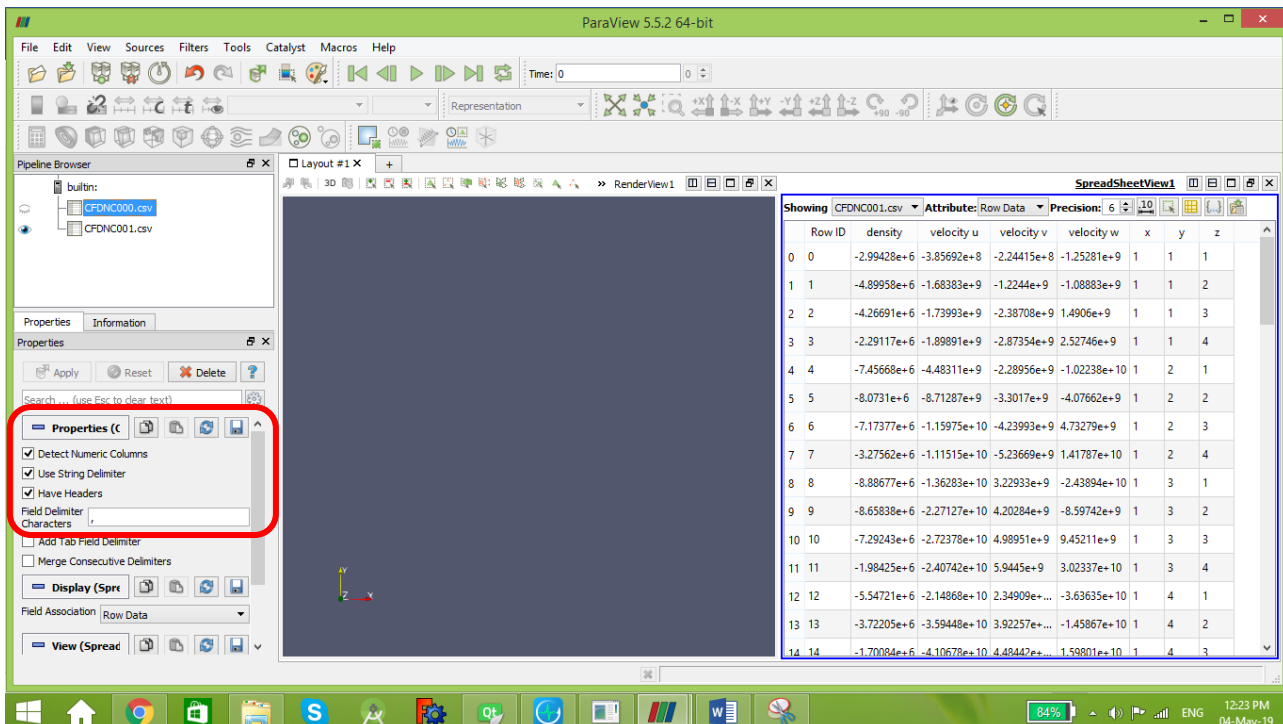
```

39.5 Results viewed on Paraview

39.5.1 First step: open your .csv files

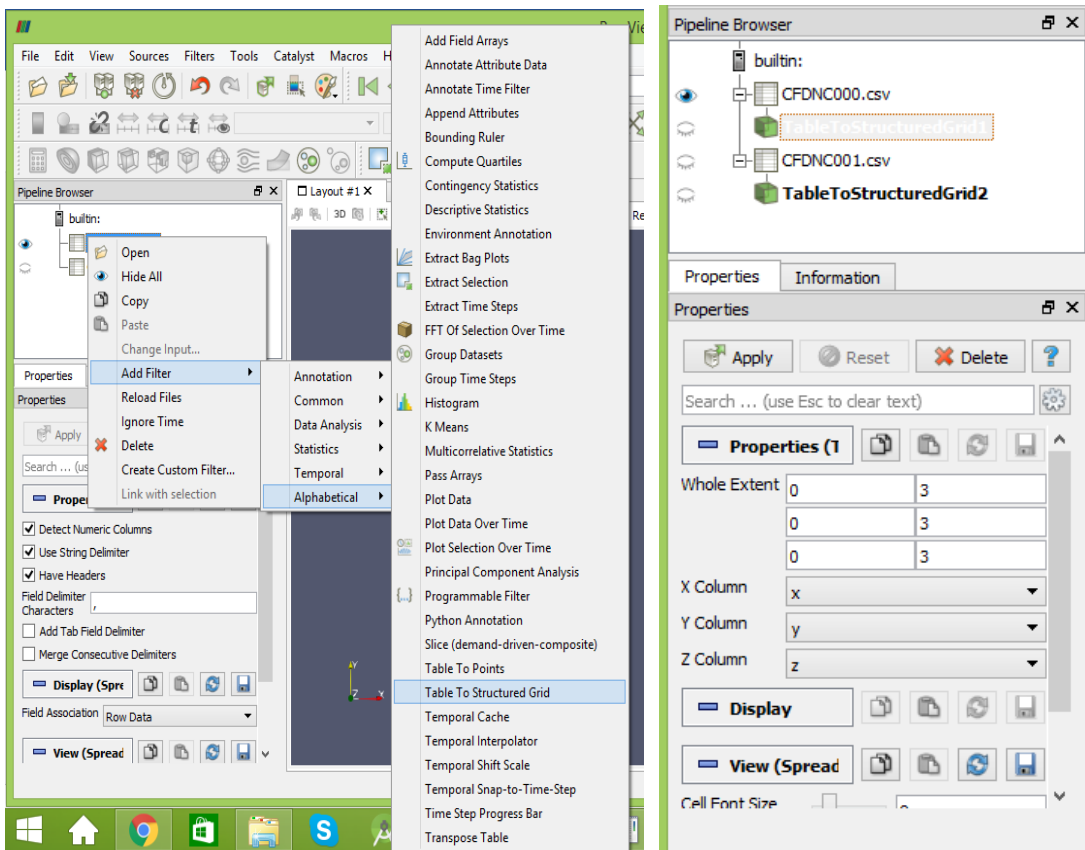


39.5.2 Second Step: make sure that the correct properties are enabled.

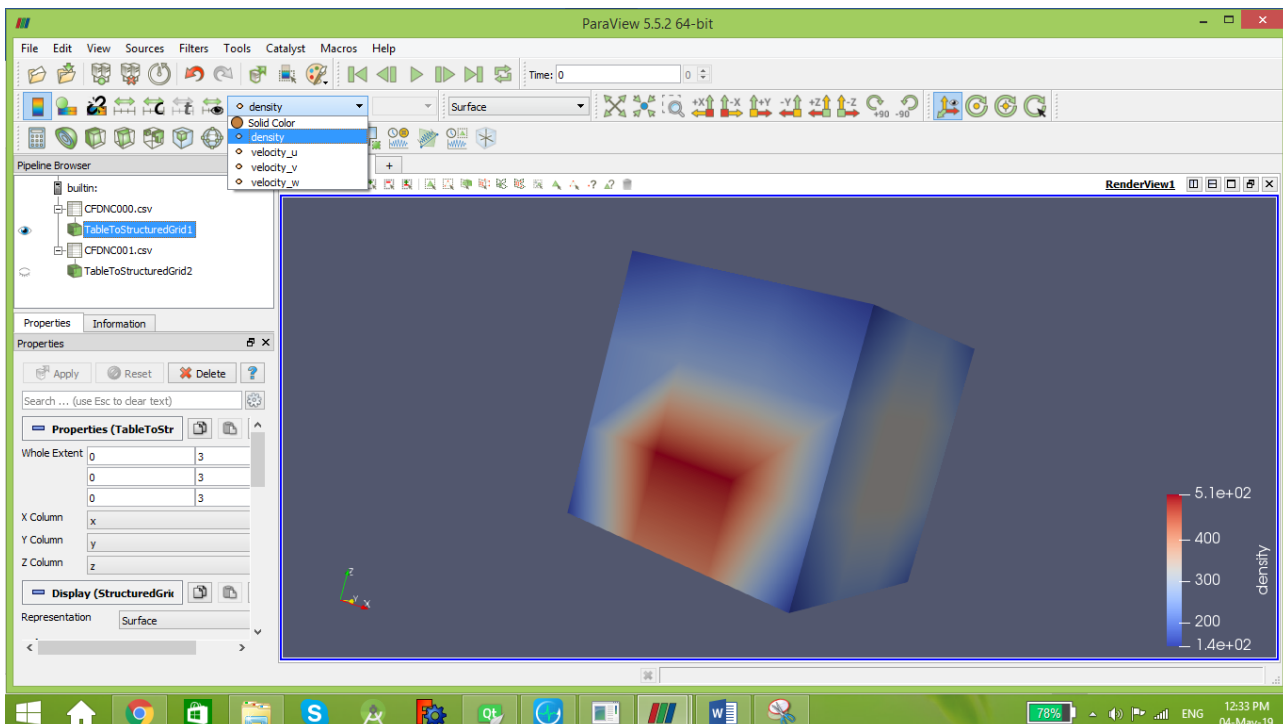


39.5.3 Third Step: Add filter for each file and fill with the right parameters:

Whole extent (0,3) for each (in our example the X, Y and Z axis are divided to 4 points), don't forget to define which columns of your data are the X, Y and Z columns.



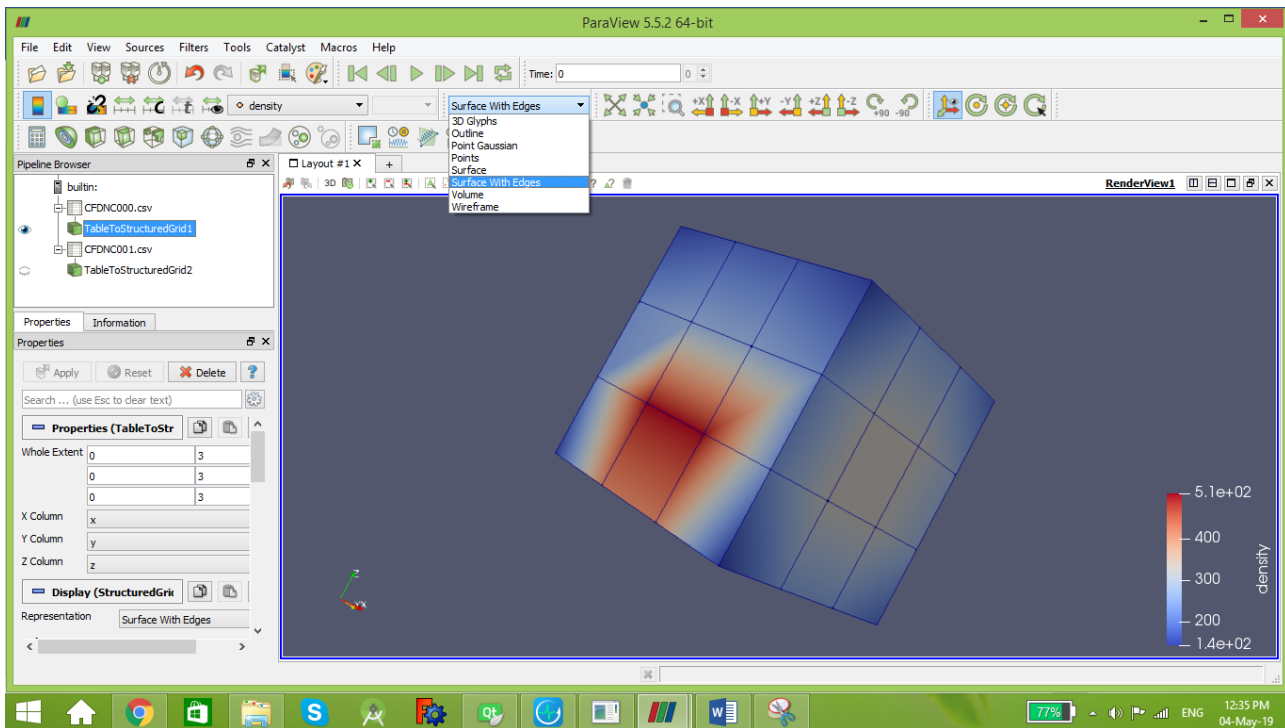
39.5.4 Fourth Step: Choose your variable and the desired view



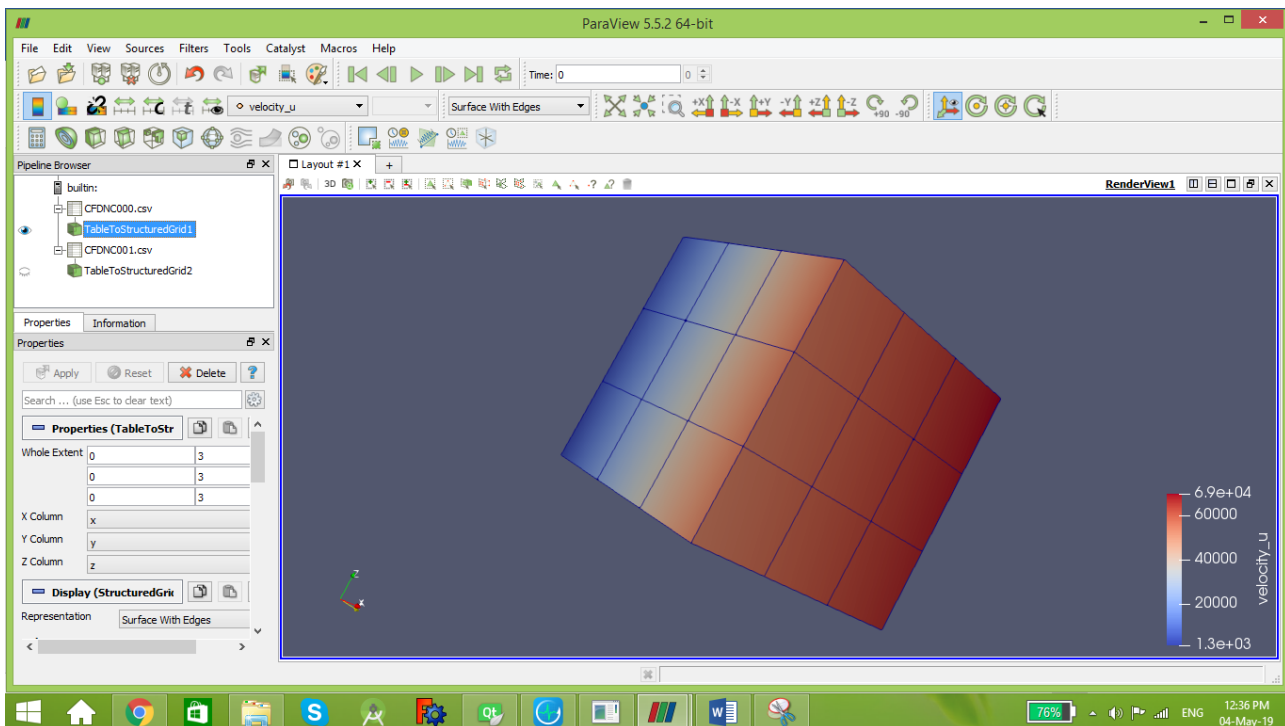
You can see your meshing by choosing the “surface with edges” option.

39.5.4.1 First time step:

Density:

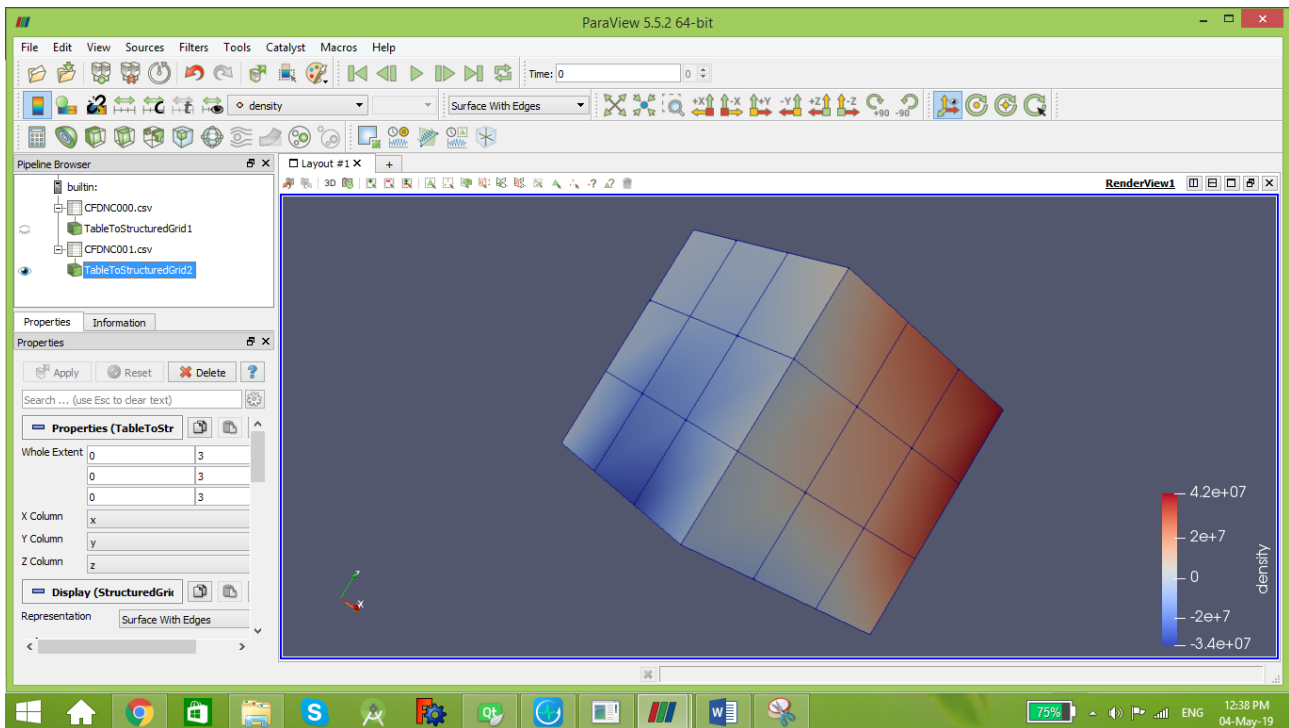


Velocity u:

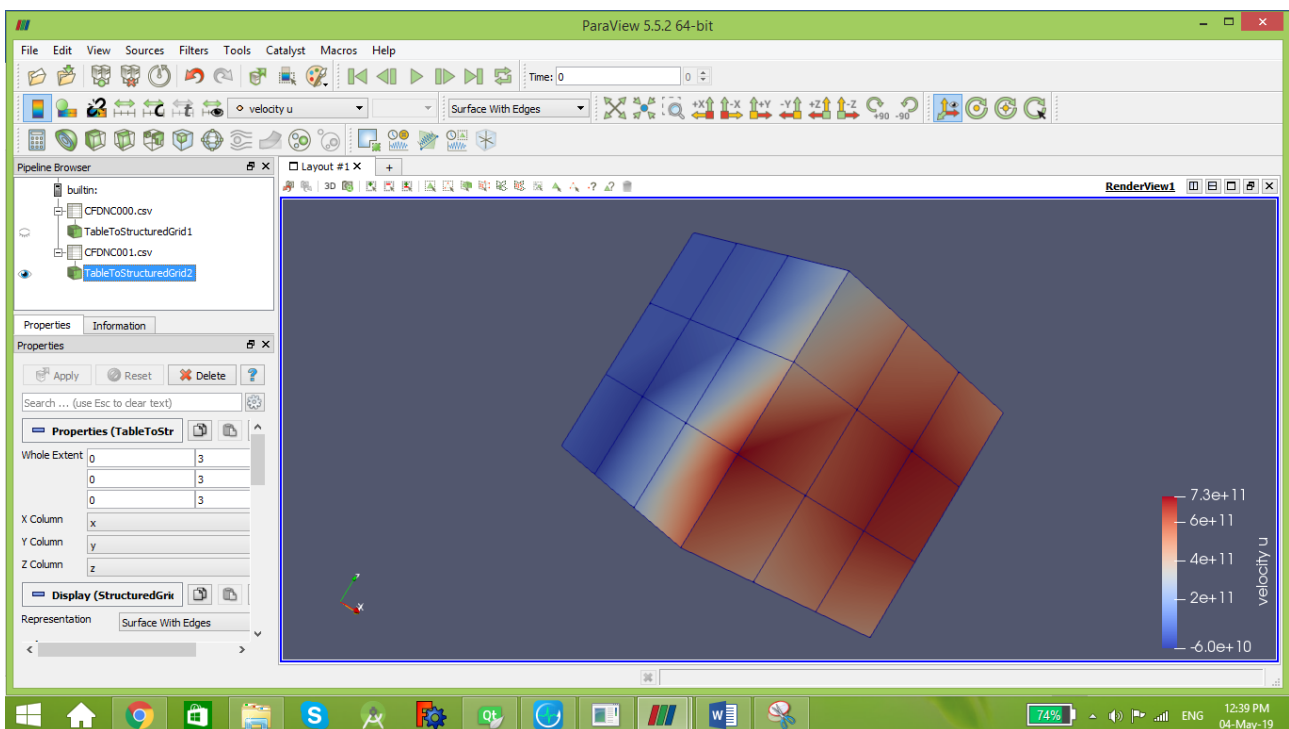


39.5.4.2 Second time step:

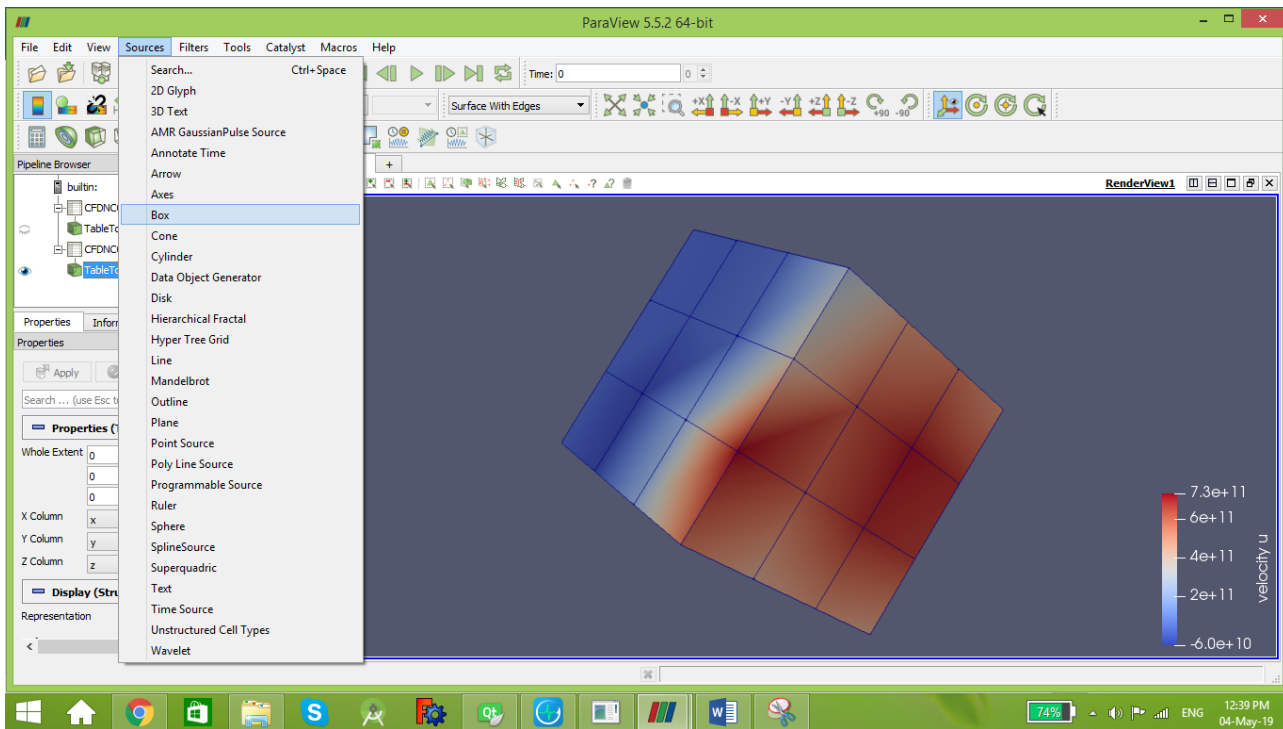
Density:



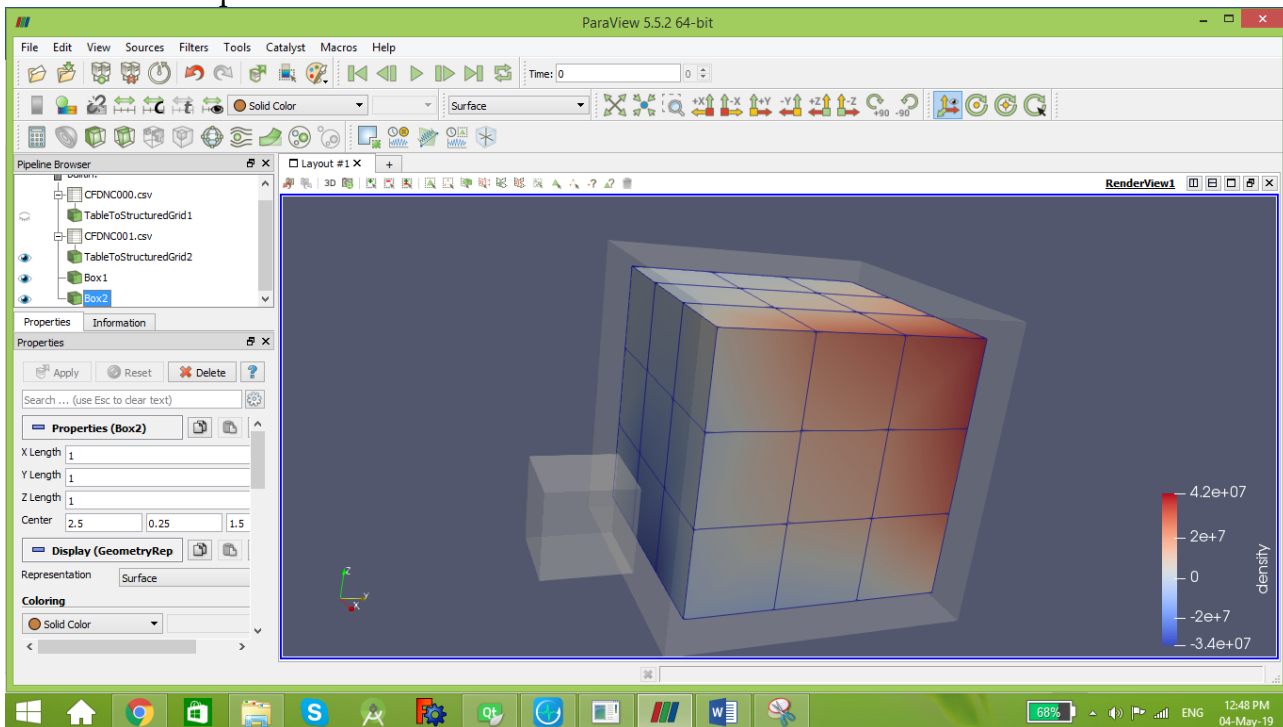
Velocity u:



39.5.5 Fifth Step(Optional): You can also add outline shape for your simulation



Choose the shape then its dimensions

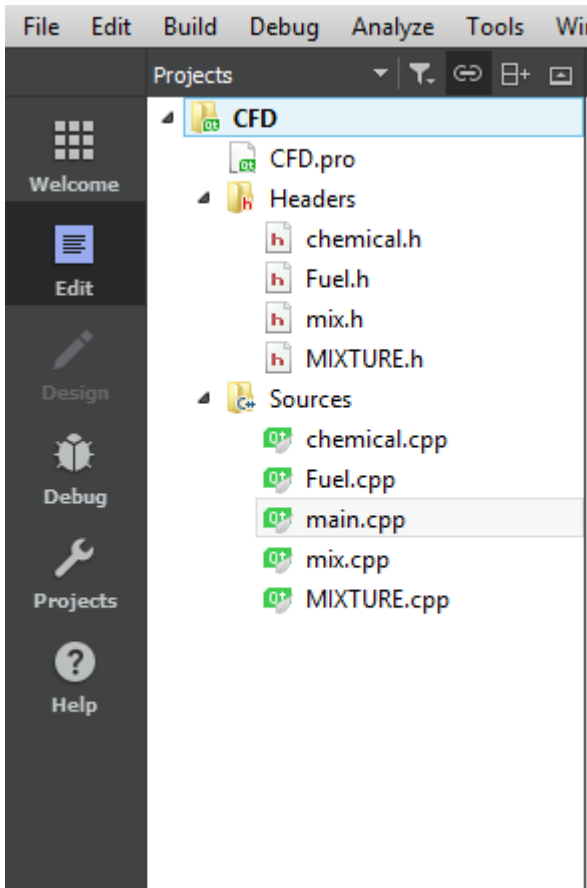


40 Annex A: Program code

Code used in chapter 8 application 1: Hydrogen combustion

Our program is composed of 4 classes:

1. Fuel: calculate the characteristics of species used.
2. Chemical: calculate the chemical constants of the procedure.
3. Mix: calculate the characteristics of the mixture.
4. MIXTURE: calculate density, velocity and internal energy of the mixture.



40.1 A.1. Class Fuel:

Fuel.h

```
#ifndef FUEL_H
#define FUEL_H
```

```
class Fuel
{
public:
    Fuel()
    {
    }
}
```

```

virtual ~Fuel()
{

}

float compute_viscosity_mu(float density, float W, float T);

float compute_viscosity_lambda(float A3, float mu);

float compute_massfraction(float mk, float m);

public:
float density;
float diffusivity;
float enthalpy;
char name;
float specific_heat;
float nu_air;
float viscosity_mu;
float viscosity_lambda;
float k;
float eps;
float Yk;
float mk; //mass of species k in a specific volume V
float m; //total mass of the mixture in the same volume V
};

#endif // FUEL_H

```

Fuel.Cpp

```

#include "Fuel.h"
#include "math.h"

float Fuel::compute_viscosity_mu(float density, float W, float T)
{
float X;
float Y;
float k;
k=((0.00094+2*pow(10,-6)*W*pow(T,1.5))/(209+19*W+T));
X=3.5+(986/T)+0.01*W;
Y=2.4-0.2*X;
viscosity_mu=k* exp (X*pow(density,Y));

return viscosity_mu;
}

float Fuel::compute_viscosity_lambda(float A3, float mu)
{
viscosity_lambda=A3*mu;

return viscosity_lambda;
}

float Fuel::compute_massfraction(float mk, float m)
{
Yk=mk/m;

return Yk;
}

```

40.2 A.2. Class Chemical:

Chemical.h:

```
#ifndef CHEMICAL_H
#define CHEMICAL_H

class Chemical
{
public:
    Chemical() {

    }
    virtual ~Chemical()
    {

    }
    float compute_reaction_rates(float f1, float f2, float a1, float b1, float Ws, float
kfj, float krj);
    float compute_Kfj(float A, float T, float beta, float Ta);
    float compute_krj(float kfj, float R, float T, float S0, float H0, float c, float Pi);

};

#endif // CHEMICAL_H
```

Chemical.Cpp:

```
#include "chemical.h"
#include "math.h"

float Chemical::compute_reaction_rates(float f1, float f2, float a1, float b1, float Ws,
float kfj, float krj) {
    float Qj;
    Qj=kfj*f1+krj*f2;

    return Qj;
}

float Chemical::compute_Kfj(float A, float T, float beta, float Ta) {
    float kfj;
    kfj=A*pow(T, beta)*exp(-Ta/T);
    return kfj;
}

float Chemical::compute_krj(float kfj, float R, float T, float S0, float H0, float c, float
Pi) {

    float krj;
    krj=(kfj/(pow((Pi/R*c), 1)*exp((S0/R)-(H0/R*T))));
    return krj;
}
```

40.3 A.3. Class MIX:**MIX.h:**

```

#ifndef MIX_H
#define MIX_H

class MIX
{
public:
    MIX() {

    }
    virtual ~MIX()
    {

    }

float compute_mixdensity(float* density_S, float* volume_S, float volume_mix, int NB);
float compute_mixviscositymu(float* mu, float* YS, float* wS, int NB);
float compute_mixviscositylambda(float viscositymu_mix, float A3);
public:

float density_S;
float density_mix;
float viscositymu_mix=0;
float f1=0;
float f2=0;

};

#endif // MIX_H

```

MIX.Cpp:

```

#include "mix.h"

float MIX::compute_mixdensity(float* density_S, float* volume_S, float volume_mix, int
NB) {

    float density_mix=0;
    for (int i=1; i<=NB; i++) {

        density_mix=density_mix+(density_S[i]*volume_S[i])*volume_mix;
    }

return density_mix;
}

float MIX::compute_mixviscositymu(float* mu, float* YS, float* wS, int NB) {

    float viscositymu_mix=0;
    float f1=0;
    float f2=0;
    for(int l=1; l<=NB; l++){
        f1=f1+YS[l]*mu[l]*wS[l];
        f2=f2+YS[l]*wS[l];
    }
}

```

```
        viscositymu_mix=f1/f2;
return viscositymu_mix;
}

float MIX::compute_mixviscositylambda(float viscositymu_mix, float A3){
float viscositylambda_mix=0;
    viscositylambda_mix=A3*viscositymu_mix;

    return viscositylambda_mix;
}
```

40.4 A.4. Class MIXTURE:

MIXTURE.h:

```
#ifndef MIXTURE_H
#define MIXTURE_H

#include<iostream>
#include<fstream>
#include<string>

class MIXTURE
{
public:
    MIXTURE()
    {

    }
    virtual ~MIXTURE()
    {

    }

    void compute_characteristic(float viscosity_mu,
                                float viscosity_lambda,
                                float density,float* density_S,
                                float ui1,float vi1,float wi1,float ui2,float vi2,float
wi2,float Ti,float Pi,float Tw, float* YS, float NB,float* fx, float* fy, float* fz);
};

#endif // MIXTURE_H
```

MIXTURE.Cpp:

```
#include "MIXTURE.h"

void MIXTURE::compute_characteristic(float viscosity_mu,float viscosity_lambda, float
density,float* density_S,float ui1,float vi1,float wi1,float ui2,float vi2,float
wi2,float Ti,float Pi,float Tw, float* YS,float NB,float* fx, float* fy, float* fz)
{
    int delta_x=1;
    int delta_y=1;
    int delta_z=1;
    int delta_t=1;
    float Fx=0;
    float Fy=0;
    float Fz=0;
    float factx=0;//energy eqt
```

```

float facty=0;//energy eqt
float factz=0;//energy eqt

float Rhomix[10][10][10][10];
float velocity_u[10][10][10][10];
float velocity_v[10][10][10][10];
float velocity_w[10][10][10][10];
float I[10][10][10][10];
float dRhomixdt[10][10][10][10];
float dudt[10][10][10][10];
float dvdt[10][10][10][10];
float dwdt[10][10][10][10];
float dIdt[10][10][10][10];
float P[10][10][10][10];
float T[10][10][10][10];

for (int t = 0; t < 3; t++){
    for (int i = 1; i < 5; i++)
    {
        for (int j = 1; j < 5; j++)
        {
            for (int k = 1; k < 5; k++){

                // initial conditions

                Rhomix[i][j][k][0]=density;
                velocity_u[i][j][k][0]=10;
                velocity_v[i][j][k][0]=20;
                velocity_w[i][j][k][0]=30;
                I[i][j][k][0]=10;
                P[i][j][k][0]=Pi;
                T[i][1][k][0]=Ti;

                //fuel inlet
                Rhomix[2][0][1][t]=density_S[1];
                velocity_u[2][0][1][t]=ui1;
                velocity_v[2][0][1][t]=vi1;
                velocity_w[2][0][1][t]=wi1;

                Rhomix[3][0][1][t]=density_S[1];
                velocity_u[3][0][1][t]=ui1;
                velocity_v[3][0][1][t]=vi1;
                velocity_w[3][0][1][t]=wi1;

                Rhomix[2][0][2][t]=density_S[1];
                velocity_u[2][0][2][t]=ui1;
                velocity_v[2][0][2][t]=vi1;
                velocity_w[2][0][2][t]=wi1;

                Rhomix[3][0][2][t]=density_S[1];
                velocity_u[3][0][2][t]=ui1;
                velocity_v[3][0][2][t]=vi1;
                velocity_w[3][0][2][t]=wi1;

                //oxidizer inlet
                Rhomix[1][0][1][t]=density_S[2];
                velocity_u[1][0][1][t]=ui2;
                velocity_v[1][0][1][t]=vi2;
                velocity_w[1][0][1][t]=wi2;
            }
        }
    }
}

```

```
Rhomix[1][0][2][t]=density_S[2];
velocity_u[1][0][2][t]=ui2;
velocity_v[1][0][2][t]=vi2;
velocity_w[1][0][2][t]=wi2;

Rhomix[1][0][3][t]=density_S[2];
velocity_u[1][0][3][t]=ui2;
velocity_v[1][0][3][t]=vi2;
velocity_w[1][0][3][t]=wi2;

Rhomix[1][0][4][t]=density_S[2];
velocity_u[1][0][4][t]=ui2;
velocity_v[1][0][4][t]=vi2;
velocity_w[1][0][4][t]=wi2;

Rhomix[2][0][3][t]=density_S[2];
velocity_u[2][0][3][t]=ui2;
velocity_v[2][0][3][t]=vi2;
velocity_w[2][0][3][t]=wi2;

Rhomix[2][0][4][t]=density_S[2];
velocity_u[2][0][4][t]=ui2;
velocity_v[2][0][4][t]=vi2;
velocity_w[2][0][4][t]=wi2;

Rhomix[3][0][3][t]=density_S[2];
velocity_u[3][0][3][t]=ui2;
velocity_v[3][0][3][t]=vi2;
velocity_w[3][0][3][t]=wi2;

Rhomix[3][0][4][t]=density_S[2];
velocity_u[3][0][4][t]=ui2;
velocity_v[3][0][4][t]=vi2;
velocity_w[3][0][4][t]=wi2;

Rhomix[4][0][1][t]=density_S[2];
velocity_u[4][0][1][t]=ui2;
velocity_v[4][0][1][t]=vi2;
velocity_w[4][0][1][t]=wi2;

Rhomix[4][0][2][t]=density_S[2];
velocity_u[4][0][2][t]=ui2;
velocity_v[4][0][2][t]=vi2;
velocity_w[4][0][2][t]=wi2;

Rhomix[4][0][3][t]=density_S[2];
velocity_u[4][0][3][t]=ui2;
velocity_v[4][0][3][t]=vi2;
velocity_w[4][0][3][t]=wi2;

Rhomix[4][0][4][t]=density_S[2];
velocity_u[4][0][4][t]=ui2;
velocity_v[4][0][4][t]=vi2;
velocity_w[4][0][4][t]=wi2;

Rhomix[i][5][k][t]=density;
velocity_u[i][5][k][t]=0;
velocity_v[i][5][k][t]=0;
velocity_w[i][5][k][t]=0;

//boundary conditions (walls)

Rhomix[0][j][k][t]=density;
Rhomix[5][j][k][t]=density;
```



```

velocity_u[0][j][k][t]=0;
velocity_v[0][j][k][t]=0;
velocity_w[0][j][k][t]=0;
velocity_u[5][j][k][t]=0;
velocity_v[5][j][k][t]=0;
velocity_w[5][j][k][t]=0;
T[0][j][k][t]=Tw;
T[5][j][k][t]=Tw;

Rhomix[i][j][0][t]=density;
Rhomix[i][j][5][t]=density;
velocity_u[i][j][0][t]=0;
velocity_v[i][j][0][t]=0;
velocity_w[i][j][0][t]=0;
velocity_u[i][j][5][t]=0;
velocity_v[i][j][5][t]=0;
velocity_w[i][j][5][t]=0;
T[i][j][0][t]=Tw;
T[i][j][5][t]=Tw;

for (int t = 0; t < 5; t++){
  for (int l=1;l<= NB;l++) {
    for (int i = 0; i <= 5; i++)
    {
      for (int j = 0; j <= 5; j++)
      {
        for (int k = 0; k <= 5; k++){

          factx=factx+(YS[l]*fx[l]*velocity_u[i][j][k][t]);
          facty=facty+(YS[l]*fy[l]*velocity_v[i][j][k][t]);
          factz=factz+(YS[l]*fz[l]*velocity_w[i][j][k][t]);
          Fx=Fx+YS[l]*fx[l];
          Fy=Fy+YS[l]*fy[l];
          Fz=Fz+YS[l]*fz[l];
        }
      }
    }
  }

  dRhomixdt[i][j][k][t]=(-Rhomix[i][j][k][t]*(velocity_u[i+1][j][k][t]-
velocity_u[i-1][j][k][t])/(2*delta_x)\
-velocity_u[i][j][k][t]*(Rhomix[i+1][j][k][t]-Rhomix[i-
1][j][k][t])/(2*delta_x)\
-Rhomix[i][j][k][t]*(velocity_v[i][j+1][k][t]-
velocity_v[i][j-1][k][t])/(2*delta_y)\
-velocity_v[i][j][k][t]*(Rhomix[i][j+1][k][t]-Rhomix[i][j-
1][k][t])/(2*delta_y)\
-Rhomix[i][j][k][t]*(velocity_w[i][j][k+1][t]-
velocity_w[i][j][k-1][t])/(2*delta_z)\
-velocity_w[i][j][k][t]*(Rhomix[i][j][k+1][t]-Rhomix[i][j][k-
1][t])/(2*delta_z));

  dudt[i][j][k][t]=-
(velocity_u[i][j][k][t]/Rhomix[i][j][k][t])*dRhomixdt[i][j][k][t]-
2*velocity_u[i][j][k][t]*((velocity_u[i+1][j][k][t]-velocity_u[i-
1][j][k][t])/(2*delta_x))\
-
velocity_u[i][j][k][t]*velocity_u[i][j][k][t]*((Rhomix[i+1][j][k][t]-Rhomix[i-
1][j][k][t])/(2*delta_x))-velocity_u[i][j][k][t]*((velocity_v[i][j+1][k][t]-
velocity_v[i][j-1][k][t])/(2*delta_y))\
-velocity_v[i][j][k][t]*((velocity_u[i][j+1][k][t]-
velocity_u[i][j-1][k][t])/(2*delta_y))-
(velocity_v[i][j][k][t]*velocity_u[i][j][k][t]/Rhomix[i][j][k][t])* (Rhomix[i][j+1][k][t]
]-Rhomix[i][j-1][k][t])/(2*delta_y)\

```

```

-velocity_u[i][j][k][t]*((velocity_w[i][j][k+1][t]-
velocity_w[i][j][k-1][t])/(2*delta_z))-
velocity_w[i][j][k][t]*((velocity_u[i][j][k+1][t]-velocity_u[i][j][k-
1][t])/(2*delta_z))\
-
((velocity_u[i][j][k][t]*velocity_w[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i][j][k+1][
t]-Rhomix[i][j][k-1][t])/(2*delta_z)))\
+
(viscosity_lambda/Rhomix[i][j][k][t])*((velocity_u[i+1][j][k][t]-
velocity_u[i][j][k][t]+velocity_u[i-1][j][k][t])/(delta_x*delta_x))\
+
((viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*((velocity_v[i+1][j+1][k][t]-
velocity_v[i+1][j-1][k][t]-velocity_v[i-1][j+1][k][t]+velocity_v[i-1][j-
1][k][t])/(4*delta_x*delta_y))+\
((viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*((velocity_w[i+1][j][k+1][t]-
velocity_w[i+1][j][k-1][t]-velocity_w[i-1][j][k+1][t]+velocity_w[i-1][j][k-
1][t])/(4*delta_x*delta_z))\
+
(2*viscosity_mu/Rhomix[i][j][k][t])*((velocity_u[i+1][j][k][t]-
velocity_u[i][j][k][t]+velocity_u[i-1][j][k][t])/(delta_x*delta_x))\
+
(viscosity_mu/Rhomix[i][j][k][t])*((velocity_u[i][j+1][k][t]-
velocity_u[i][j][k][t]+velocity_u[i][j-
1][k][t])/(delta_y*delta_y)+(viscosity_mu/Rhomix[i][j][k][t])*((velocity_u[i][j][k+1][t]-
velocity_u[i][j][k][t]+velocity_u[i][j][k-1][t])/(delta_z*delta_z)+F_x;

dvdt[i][j][k][t]=-
(velocity_v[i][j][k][t]/Rhomix[i][j][k][t])*dRhomixdt[i][j][k][t]-
2*velocity_v[i][j][k][t]*((velocity_v[i][j+1][k][t]-velocity_v[i][j-
1][k][t])/(2*delta_y))\
-
velocity_v[i][j][k][t]*velocity_v[i][j][k][t]*((Rhomix[i][j+1][k][t]-Rhomix[i][j-
1][k][t])/(2*delta_y))-velocity_v[i][j][k][t]*((velocity_w[i][j][k+1][t]-
velocity_w[i][j][k-1][t])/(2*delta_z))\
-velocity_v[i][j][k][t]*((velocity_u[i+1][j][k][t]-
velocity_u[i-1][j][k][t])/(2*delta_x))-
(velocity_v[i][j][k][t]*velocity_u[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i+1][j][k][t]-
Rhomix[i-1][j][k][t])/(2*delta_x))-velocity_u[i][j][k][t]*((velocity_v[i+1][j][k][t]-
velocity_v[i-1][j][k][t])/(2*delta_x))\
-velocity_w[i][j][k][t]*((velocity_v[i][j][k+1][t]-
velocity_v[i][j][k-1][t])/(2*delta_z))\
-
((velocity_v[i][j][k][t]*velocity_w[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i][j][k+1][
t]-Rhomix[i][j][k-1][t])/(2*delta_z)))\
+
(viscosity_mu/Rhomix[i][j][k][t])*((velocity_v[i+1][j][k][t]-
velocity_v[i][j][k][t]+velocity_v[i-1][j][k][t])/(delta_x*delta_x))\
+
((viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*((velocity_u[i+1][j+1][k][t]-
velocity_u[i+1][j-1][k][t]-velocity_u[i-1][j+1][k][t]+velocity_u[i-1][j-
1][k][t])/(4*delta_x*delta_y))+\
((viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*((velocity_w[i][j+1][k+1][t]-
velocity_w[i][j+1][k-1][t]-velocity_w[i][j-1][k+1][t]+velocity_w[i][j-1][k-
1][t])/(4*delta_y*delta_z))\
+
(2*viscosity_mu/Rhomix[i][j][k][t])*((velocity_v[i][j+1][k][t]-
velocity_v[i][j][k][t]+velocity_v[i][j-1][k][t])/(delta_y*delta_y))\
+
(viscosity_lambda/Rhomix[i][j][k][t])*((velocity_v[i][j+1][k][t]-
velocity_v[i][j][k][t]+velocity_v[i][j-

```

```

1][k][t])/(delta_y*delta_y)+(viscosity_mu/Rhomix[i][j][k][t])*(velocity_v[i][j][k+1][t]
-velocity_v[i][j][k][t]+velocity_v[i][j][k-1][t])/(delta_z*delta_z)+Fy;

                dwdt[i][j][k][t]=
(velocity_w[i][j][k][t]/Rhomix[i][j][k][t])*dRhomixdt[i][j][k][t]-
2*velocity_w[i][j][k][t]*((velocity_w[i][j][k+1][t]-velocity_w[i][j][k-
1][t])/(2*delta_z))\
-
velocity_w[i][j][k][t]*velocity_w[i][j][k][t]*((Rhomix[i][j][k+1][t]-Rhomix[i][j][k-
1][t])/(2*delta_z))-velocity_v[i][j][k][t]*((velocity_w[i][j+1][k][t]-velocity_w[i][j-
1][k][t])/(2*delta_y))\
-velocity_w[i][j][k][t]*((velocity_u[i+1][j][k][t]-
velocity_u[i-1][j][k][t])/(2*delta_x))-
(velocity_w[i][j][k][t]*velocity_u[i][j][k][t]/Rhomix[i][j][k][t])*(Rhomix[i+1][j][k][t]
-Rhomix[i-1][j][k][t])/(2*delta_x)-velocity_u[i][j][k][t]*(velocity_w[i+1][j][k][t]-
velocity_w[i-1][j][k][t])/(2*delta_x)\
-velocity_w[i][j][k][t]*((velocity_v[i][j+1][k][t]-
velocity_v[i][j-1][k][t])/(2*delta_y))\
-
((velocity_v[i][j][k][t]*velocity_w[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i][j+1][k]
[t]-Rhomix[i][j-1][k][t])/(2*delta_y)))\
+
(viscosity_mu/Rhomix[i][j][k][t])*(velocity_w[i+1][j][k][t]-
velocity_w[i][j][k][t]+velocity_w[i-1][j][k][t])/(delta_x*delta_x)\
+
((viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*(velocity_u[i+1][j][k+1][t]-
velocity_u[i+1][j][k-1][t]-velocity_u[i-1][j][k+1][t]+velocity_u[i-1][j][k-
1][t])/(4*delta_x*delta_z)+\
(
(viscosity_lambda+viscosity_mu)/Rhomix[i][j][k][t])*(velocity_v[i][j+1][k+1][t]-
velocity_v[i][j+1][k-1][t]-velocity_v[i][j-1][k+1][t]+velocity_v[i][j-1][k-
1][t])/(4*delta_y*delta_z)\
+
(2*viscosity_mu/Rhomix[i][j][k][t])*(velocity_w[i][j][k+1][t]-
velocity_w[i][j][k][t]+velocity_w[i][j][k-1][t])/(delta_z*delta_z)\
+
(viscosity_lambda/Rhomix[i][j][k][t])*(velocity_w[i][j+1][k][t]-
velocity_w[i][j][k][t]+velocity_w[i][j-
1][k][t])/(delta_y*delta_y)+(viscosity_mu/Rhomix[i][j][k][t])*(velocity_w[i][j][k+1][t]
-velocity_w[i][j][k][t]+velocity_w[i][j][k-1][t])/(delta_z*delta_z)+Fz;

dIdt[i][j][k][t]=(-I[i][j][k][t]/Rhomix[i][j][k][t])*dRhomixdt[i][j][k][t]-
I[i][j][k][t]*((velocity_u[i+1][j][k][t]-velocity_u[i-1][j][k][t])/(2*delta_x)) -
(I[i][j][k][t]*velocity_u[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i+1][j][k][t]-
Rhomix[i-1][j][k][t])/(2*delta_x))-velocity_u[i][j][k][t]*((I[i+1][j][k][t]-I[i-
1][j][k][t])/(2*delta_x))-I[i][j][k][t]*((velocity_v[i][j+1][k][t]-velocity_v[i][j-
1][k][t])/(2*delta_y))-
(I[i][j][k][t]*velocity_v[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i][j+1][k][t]-
Rhomix[i][j-1][k][t])/(2*delta_y)) -velocity_v[i][j][k][t]*((I[i][j+1][k][t]-I[i][j-
1][k][t])/(2*delta_y))-I[i][j][k][t]*((velocity_w[i][j][k+1][t]-velocity_w[i][j][k-
1][t])/(2*delta_z))-
(I[i][j][k][t]*velocity_w[i][j][k][t]/Rhomix[i][j][k][t])*((Rhomix[i][j][k+1][t]-
Rhomix[i][j][k-1][t])/(2*delta_z)) -velocity_w[i][j][k][t]*((I[i][j][k+1][t]-I[i][j][k-
1][t])/(2*delta_z))-(viscosity_lambda/Rhomix[i][j][k][t])*((T[i+1][j][k][t]-
2*T[i][j][k][t]+T[i-1][j][k][t])/(delta_x*delta_x)-(T[i][j+1][k][t]-
2*T[i][j][k][t]+T[i][j-1][k][t])/(delta_y*delta_y)-(T[i][j][k+1][t]-
2*T[i][j][k][t]+T[i][j][k-1][t])/(2*delta_z)) -
((4/3)*viscosity_mu/Rhomix[i][j][k][t])*((velocity_u[i+1][j][k][t]-
2*velocity_u[i][j][k][t]+velocity_u[i-
1][j][k][t])/(delta_x*delta_x)+(velocity_v[i][j+1][k][t]-
2*velocity_v[i][j][k][t]+velocity_v[i][j-

```

```

1][k][t))/(delta_y*delta_y)+(velocity_w[i][j][k+1][t]-
2*velocity_w[i][j][k][t]+velocity_w[i][j][k-1][t))/(delta_z*delta_z))\

+(viscosity_mu/Rhomix[i][j][k][t])*(velocity_u[i][j+1][k][t]-
2*velocity_u[i][j][k][t]+velocity_u[i][j-
1][k][t))/(delta_y*delta_y)+(2*viscosity_mu/Rhomix[i][j][k][t])*((velocity_v[i+1][j][k][t]-
velocity_v[i-1][j][k][t))/(2*delta_x))*((velocity_u[i][j+1][k][t]-velocity_u[i][j-
1][k][t))/(2*delta_y))\

+(viscosity_mu/Rhomix[i][j][k][t])*(velocity_u[i][j][k+1][t]-
2*velocity_u[i][j][k][t]+velocity_u[i][j][k-
1][t))/(delta_z*delta_z)+(2*viscosity_mu/Rhomix[i][j][k][t])*((velocity_w[i+1][j][k][t]-
velocity_w[i-1][j][k][t))/(2*delta_x))*((velocity_u[i][j][k+1][t]-velocity_u[i][j][k-
1][t))/(2*delta_z))\

+(viscosity_mu/Rhomix[i][j][k][t])*((velocity_v[i+1][j][k][t]-
2*velocity_v[i][j][k][t]+velocity_v[i-
1][j][k][t))/(delta_x*delta_x)+(viscosity_mu/Rhomix[i][j][k][t])*((velocity_v[i][j][k+
1][t]-2*velocity_v[i][j][k][t]+velocity_v[i][j][k-1][t))/(delta_z*delta_z))\

+(2*viscosity_mu/Rhomix[i][j][k][t])*((velocity_w[i][j+1][k][t]-velocity_w[i][j-
1][k][t))/(2*delta_y))*((velocity_v[i][j][k+1][t]-velocity_v[i][j][k-
1][t))/(2*delta_z))+ (viscosity_mu/Rhomix[i][j][k][t])*((velocity_w[i+1][j][k][t]-
2*velocity_w[i][j][k][t]+velocity_w[i-1][j][k][t))/(delta_x*delta_x))\

+(viscosity_mu/Rhomix[i][j][k][t])*((velocity_w[i][j+1][k][t]-
2*velocity_w[i][j][k][t]+velocity_w[i][j-1][k][t))/(delta_y*delta_y))-
(P[i][j][k][t]/Rhomix[i][j][k][t])*((velocity_u[i+1][j][k][t]-velocity_u[i-
1][j][k][t))/(2*delta_x)+(velocity_v[i][j+1][k][t]-velocity_v[i][j-
1][k][t))/(2*delta_y)+(velocity_w[i][j][k+1][t]-velocity_w[i][j][k-1][t))/(2*delta_z))
+factx+facty+factz;

I[i][j][k][t+delta_t]=I[i][j][k][t]+dIdt[i][j][k][t]*delta_t;
Rhomix[i][j][k][t+delta_t]=Rhomix[i][j][k][t]+dRhomixdt[i][j][k][t]*delta_t;
velocity_u[i][j][k][t+delta_t]=velocity_u[i][j][k][t]+dudt[i][j][k][t]*delta_t;
velocity_v[i][j][k][t+delta_t]=velocity_v[i][j][k][t]+dvdt[i][j][k][t]*delta_t;
velocity_w[i][j][k][t+delta_t]=velocity_w[i][j][k][t]+dwdt[i][j][k][t]*delta_t;

printf("%d,%d,%d,%f,%f,%f,%f\n",i,j,k,Rhomix[i][j][k][t+delta_t],velocity_u[i][j][k][t+
delta_t],velocity_v[i][j][k][t+delta_t],velocity_w[i][j][k][t+delta_t]);
    }
    }
}
system("pause");
}
}

```

40.5 A.5. Main program:

```

#include <QCoreApplication>
#include<iostream>
#include<fstream>
#include<string>
#include<math.h>
#include "Fuel.h"
#include "mix.h"
#include "MIXTURE.h"
#include "chemical.h"

```

```

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    float NB; // nb of species
    float density_S[100]; // density for each species
    float density_mix; // mix density
    float volume_S[100]; // volume of species S
    float volume_mix; // mixture volume
    float wS[100]; // molecular weight for each species
    float w; // mix molecular weight
    float Ti; // initial fuel temp
    float ui1; // initial fuel velocity
    float vi1; // initial fuel velocity
    float wi1; // initial fuel velocity
    float ui2; // initial fuel velocity
    float vi2; // initial fuel velocity
    float wi2; // initial fuel velocity
    float Tw; // wall temp
    float mS[100]; // mass of each species
    float m; // total mass
    float R;
    float Pi; // initial pressur
    float w1; // ys/ws
    float a1[100], b1[100]; // stoeciometric coefficient
    float A; // chemical constant
    float beta; // temp. constant
    float Ta; // activation temperature
    float E; // activation energy
    float S0, H0; // entropie, enthalpie
    float mu[100]; // viscosity of species
    float viscositymu_mix;
    float lambda[100]; // viscosity of species
    float viscositylambda_mix;
    float YS[100]; // mass fraction
    float fx[100], fy[100], fz[100]; // mass forces

    /*.....
    .....
    .....
    .....*/

    //Input

    printf("Number of species >1: ");
    scanf("%f", &NB);
    printf("Enter Species characteristics:");
    for (int i=1; i<=NB; i++) {
        printf("Species %d:", i);
        printf("density : \n");
        scanf("%f", &density_S[i]);
        printf("mass in volume V:\n");
        scanf("%f", &mS[i]);
        printf("molecular weight:\n");
        scanf("%f", &wS[i]);
        printf("volume V of species:\n");
        scanf("%f", &volume_S[i]);
        printf("stoichiometric coefficient a[%d]:\n ", i);
        scanf("%f", &a1[i]);
        printf("stoichiometric coefficient b[%d]:\n ", i);
        scanf("%f", &b1[i]);
        printf("mass force fx:\n");
        scanf("%f", &fx[i]);

```

```

    printf("mass force fy:\n");
    scanf("%f",&fy[i]);
    printf("mass force fz:\n");
    scanf("%f",&fz[i]);
    system("pause");
}

printf("Mixture mass :\n ");
scanf("%f",&m);
printf("volume V of mixture:\n");
scanf("%f",&volume_mix);
printf("activation energy E:\n");
scanf("%f",&E);
printf("Entropie:\n");
scanf("%f",&S0);
printf("Enthalpi:\n");
scanf("%f",&H0);
printf("Rx constant A:\n");
scanf("%f",&A);
printf("Temp constant beta:\n");
scanf("%f",&beta);

// Boundary conditions
printf("Enter Boundary conditions:\n");
printf("Inlet temperature:\n");
scanf("%f",&Ti);
printf("inlet fuel Velocity u component:\n");
scanf("%f",&ui1);
printf("inlet fuel Velocity v component:\n");
scanf("%f",&vi1);
printf("inlet fuel Velocity w component:\n");
scanf("%f",&wi1);
printf("inlet oxidizer Velocity u component:\n");
scanf("%f",&ui2);
printf("inlet oxidizer Velocity v component:\n");
scanf("%f",&vi2);
printf("inlet oxidizer Velocity w component:\n");
scanf("%f",&wi2);
printf("Enter Boundary conditions:\n");
printf("wall temperature:\n");
scanf("%f",&Tw);
/*.....*/
//Species characteristics

Fuel fuel[100];
R=8.314;//j/mole.K

for (int i=1;i<=NB;i++) {
    mu[i] = fuel[i].compute_viscosity_mu(density_S[i],wS[i],Ti);
    printf("viscosity mu[%d]: %f\n",i,mu[i]);
    lambda[i]= fuel[i].compute_viscosity_lambda(0.067,mu[i]);
    printf("viscosity lambda[%d]: %f\n",i,lambda[i]);
    YS[i]=fuel[i].compute_massfraction(mS[i],m);
    printf("Species %d mass fraction : %f\n",i,YS[i]);
}
system("pause");
/*.....*/

```

```

//mixture characteristics

MIX mix1;

density_mix=mix1.compute_mixdensity(density_S,volume_S,volume_mix,NB);
viscositymu_mix=mix1.compute_mixviscositymu(mu,YS,wS,NB);
viscositylambda_mix=mix1.compute_mixviscositylambda(viscositymu_mix,0.067);
w1=0;
for (int i=1;i<=NB;i++) {
    w1=w1+(YS[i]/wS[i]);
}
w=1/w1;
// Pi=density_mix*R/(w*Ti);
Pi=1;

/*.....*/
// Chemical kinetics

Chemical chem;
float Wk[100];//mass reaction rate
float kfj;//forward rate
float krj;//reverse rate
float Qj;//rate of progress
float c1;//molar stoichiometric coefficients
float f1=1;//chemical factor
float f2=1;//chemical factor
float c=0;//chemical factor

for (int i=1;i<=NB;i++) {
    f1=f1*pow((density_mix*YS[i]/w),a1[i]);
    f2=f2*pow((density_mix*YS[i]/w),b1[i]);
    c=c+(b1[i]-a1[i]);
}

Ta=E/R;

for (int i=1;i<=NB;i++) {

    kfj=chem.compute_Kfj(A,Ti,beta,Ta);
    krj=chem.compute_krj(kfj,R,Ti,S0,H0,c,Pi);
    Qj=chem.compute_reaction_rates(f1,f2,a1[i],b1[i],wS[i],kfj,krj);
    c1=b1[i]-a1[i];
    Wk[i]=wS[i]*c1*Qj;
}

/*.....*/
//mixture variables u,v,w,density,energy

MIXTURE MIX;
printf("x,y,z,density,velocity_u,velocity_v,velocity_w\n");
MIX.compute_characteristic(viscositymu_mix,viscositylambda_mix,density_mix,density_S,ui1
,vi1,wil,ui2,vi2,wi2,Ti,Pi,Tw,YS,NB,fx,fy,fz);

return 0;
}

```

TYPE II SUPERNOVA

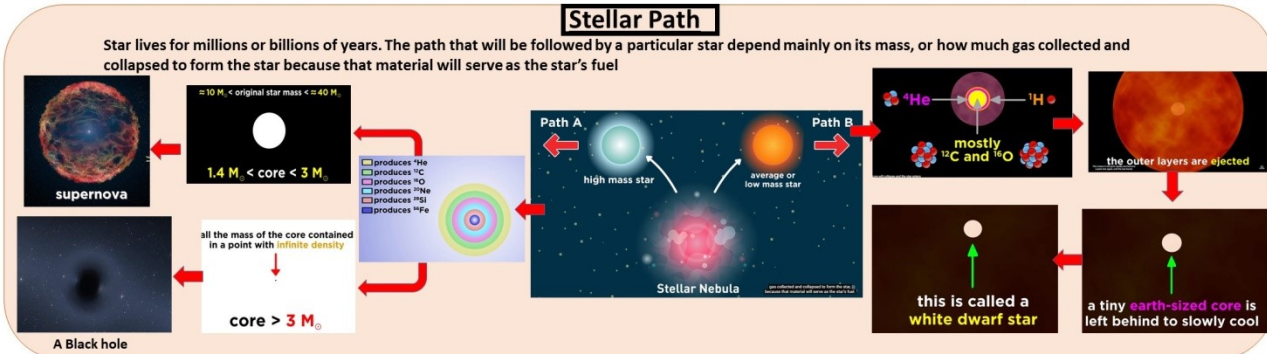
Author:

Maryam ABDEL-KARIM

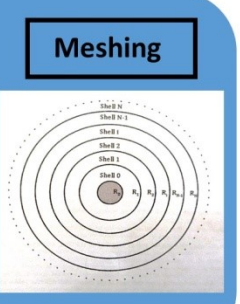
Last updated: 01.06.2019

**برنامج رقمي هدفه حساب و محاكاة (CFD) ظاهرة السوبرنوفا النوع الثاني.
Framework + Graphical User Interface (GUI), Supernova Type-II**

مدخل: الهدف هو نمذجة الديناميكا المائية لنجم سوبر نوفا من النوع الثاني (Supernova type-II) باستخدام C++. لم يلاحظ انفجار المستعر الأعظم النموذجي بسبب عدم ثبات الكود، ولكن تم إجراء اختبار على الغلاف الجوي الشبيه بالأرض، ويمكن الاستنتاج أن الديناميكا المائية لنموذج السوبرنوفا تعمل بشكل صحيح.



- The star is divided into a number of spherical shells.
- These shells contain stellar material with a certain density, temperature, pressure and internal energy.
- The dynamics of the star can be simulated by studying the movement of these spherical shells. By looking at the way in which the radial position of each shell changes due to gravity and pressure, gives an understanding of the dynamics of the star as a whole.



The parameters studied in this program are the pressure, Energy and temperature.

Equations

- The pressure:
$$P = \frac{\rho}{3} T^4 + \frac{\rho k_B T}{\mu}$$
- The total energy is the sum of:
 - The internal energy:
$$U = \frac{2}{3} k_B N T + a V T^4$$
 - The kinetic energy:
$$E_{kin} = \frac{1}{2} m_{shell} \left(\frac{v_{i+1} + v_i}{2} \right)^2$$
 - The gravitational potential energy:
$$E_{grav} = -\frac{G M_{enc} m_{shell}}{r^2}$$
 - The temperature:
$$dT = -\frac{\left(\frac{\rho k T}{\mu} + \frac{a T^4}{3} + a T^4 \right) dV}{\frac{3}{2} k \frac{m_{shell}}{\mu} + 4 a V T^3}$$

Program

Our program is composed of 4 classes:

- Initial functions
- Dynamics
- Energy
- Parameter

```

    SN
    ├── SN.pro
    ├── Headers
    │   ├── dynamics.h
    │   ├── energy.h
    │   ├── initial_functions.h
    │   └── parameter.h
    └── Sources
        ├── dynamics.cpp
        ├── energy.cpp
        ├── initial_functions.cpp
        ├── main.cpp
        └── parameter.cpp
    
```

- The first class Initial functions calculate the initial values of the pressure, temperature, density and the H constant.
- The second class Dynamics calculate the acceleration, pressure generation and the temperature distribution
- The third class Energy calculate the kinetic energy, internal energy and the gravitational energy.
- The fourth class Parameter define the number of shell desired.
- The initial conditions are specified in the main program.

Results النتائج

The results are shown for different time steps in the order shown below. Each circle represents a shell (i.e. a total of 100 circle).

Contents

517 **مدخل** INTRODUCTION**518** **CHAPTER 1: BASICS**

1. INTRODUCTION	518
2. STAR EVOLUTION	518
3. STELLAR COMPOSITION	518
4. ASSUMPTION	519
5. NUMERICAL MODEL	519
6. PHYSICS OF SUPERNOVA	520
6.1. <i>Pressure</i>	520
6.2. <i>Energy</i>	520
6.3. <i>Temperature</i>	521

522 **CHAPTER 2: PROGRAM**

1. CLASSES	522
2. RESULTS SAVING	660

524 **CHAPTER 3: RESULTS**

1. INPUT	524
2. RESULTS	524
3. RESULTS VIEWED ON PARAVIEW	525

527 **ANNEX A**

41 Introduction مدخل

The code presented here is inspired from the report: << "Modeling a Type-II supernova"- F.S. Nobels, J. Ubink, H.W. de vries, Guided by: Prof. Dr. O. Scholten. January 21,2015 >>. The aim was to model the hydrodynamics of a type-II core collapse supernova using C++. code was created for modelling. The typical supernova explosion was not observed due to instability of the code, but a test on the earth-like atmosphere was done, it can be concluded that the hydrodynamics of the supernova model probably work correctly.

البرنامج المطروح في هذا العمل مستوحى من التقرير:

<< "Modeling a Type-II supernova"- F.S. Nobels, J. Ubink, H.W. de vries, Guided by: Prof. Dr. O. Scholten. January 21,2015 >>.

الهدف هو نمذجة الديناميكا المائية لنجم سوبر نوبا من النوع الثاني (Supernova type-II) باستخدام C++. لم يُلاحظ انفجار المستعر الأعظم النموذجي بسبب عدم ثبات الكود، ولكن تم إجراء اختبار على الغلاف الجوي الشبيه بالأرض، ويمكن الاستنتاج أن الديناميكا المائية لنموذج السوبرنوبا تعمل بشكل صحيح.

42 Chapter 1: Basics

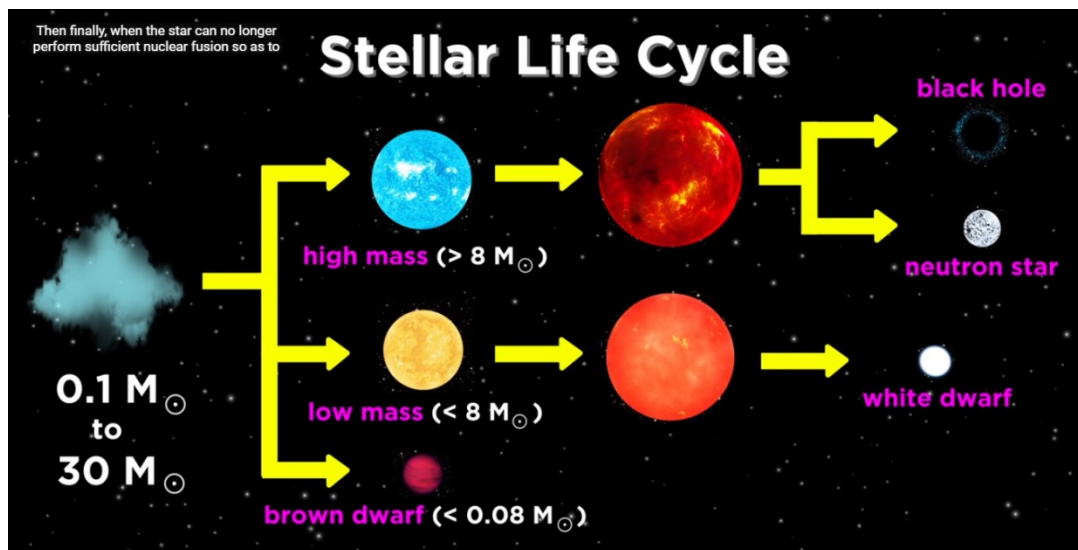
1. Introduction

A star death may be accompanied by a tremendous explosion called supernova. Being the brightest events in the entire universe, a typical supernova can outshine an entire galaxy, emitting a typical amount of 10^{24} J of energy as radiations.

The following code does not take into consideration the luminosity, Decay processes, Opacity and Blast waves.

2. Star evolution

In order for a supernova to occur, a star has to meet certain criteria, of which the most important one for mode will be its mass. If a star is too light, it will not be capable of showcasing a type-II supernova. One can illustrate this criterion by having a look at the influence of mass on the fusion processes that occur in the star. (see figure below)



The Star types are:

- Main sequence:
 - Blue stars: big, hot, bright (up to 200 solar masses)
 - Yellow stars: in between (close to 1 solar mass)
 - Red stars: small, cool, dim (down to 0.1 solar masses)
- Red Giant: red and cool (0.3-8 solar masses)
- White dwarf: tiny and hot (0.2- 1.3 solar masses)

3. Stellar composition

Two different aspect of stellar composition are considered:

- A neutron star core
- Red giant star

3.1. Neutron star core model: The star contains only the element hydrogen, and a neutron star at its core. The fact that there is a neutron star at the center of the atmosphere, comes from the collapsing

of the iron core in the pre-supernova situation. In the simulations, the model starts at the moment at which the atmosphere rebounds from the neutron star core.

3.2. Red giant star model: The stellar material is assumed to consist of approximately 90% hydrogen and a stellar core of iron, the stellar atmosphere is divided into layers of different elements.

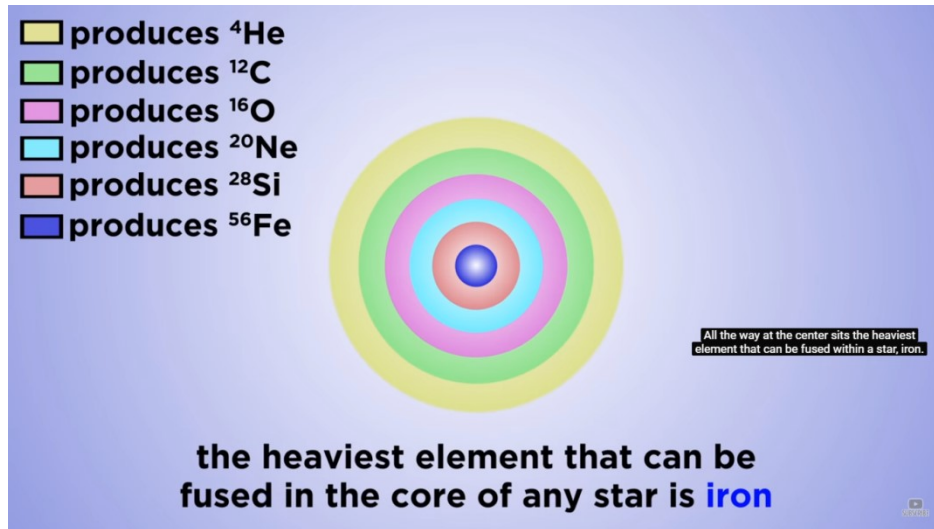


Table 1: Stellar composition of a $20 M_{\odot}$ star, at the end of its life

Element	Main isotope	Total mass	Mean molecular mass (in u)
Iron/neutrons	N.A.	$1.4 M_{\odot}$	N.A.
Silicon	$^{28}_{14}\text{Si}$	$0.12 M_{\odot}$	$\mu = \frac{28}{18} \approx 1.87$
Oxygen	$^{16}_8\text{O}$	$0.12 M_{\odot}$	$\mu = \frac{16}{9} \approx 1.78$
Neon	$^{20}_{10}\text{Ne}$	$0.12 M_{\odot}$	$\mu = \frac{20}{11} \approx 1.82$
Carbon	$^{12}_6\text{C}$	$0.12 M_{\odot}$	$\mu = \frac{12}{7} \approx 1.71$
Helium	^4_2He	$0.12 M_{\odot}$	$\mu = \frac{4}{3} \approx 1.33$
Hydrogen	^1_1H	$18 M_{\odot}$	$\mu = 0.6$ (assumption, from [24])

Table 2: Density and temperature of a $20 M_{\odot}$ star at particular shell boundaries[28]

Shell edge	H-He	He-C	C-Ne	Ne-O	O-Si	Si-core
Density (10^3kg m^{-3})	4.53	$0.968 \cdot 10^3$	$1.70 \cdot 10^5$	$3.10 \cdot 10^6$	$5.55 \cdot 10^6$	$4.26 \cdot 10^7$
Temperature (10^7 K)	3.69	18.8	87.0	157	199	334
Calculated radius (km)	389066	69476	26395	21738	11020	12

4. Assumption

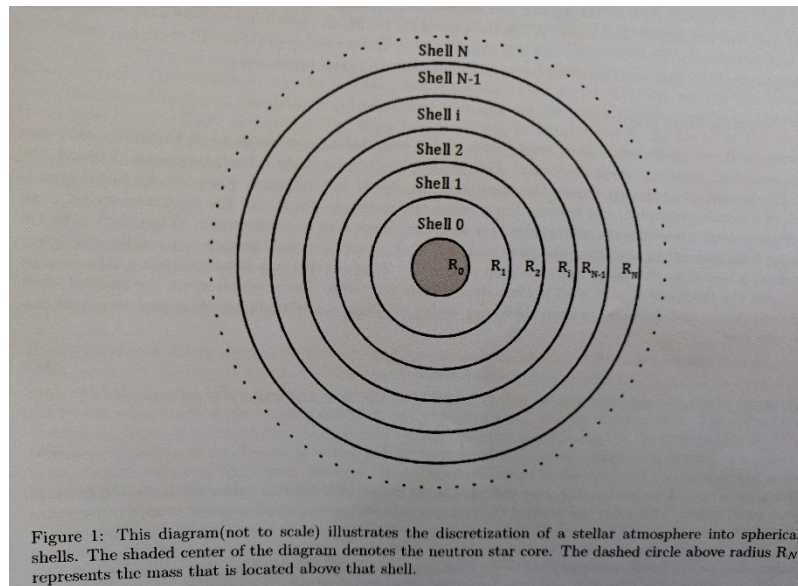
Basics Assumptions:

- Spherical symmetry (The star's properties are assumed to have no angular dependence)
- Gas composition (it is assumed that the star consists of a mixture of photon gas and ideal classical gas)
- Homogeneous distributions (the same value for the density, pressure, and temperature hold at each radial position inside a certain shell. In the model, these quantities are evaluated in the middle of shells)

5. Numerical model

In this model, the star is divided into a number of spherical shells.

These shells contain stellar material with a certain density, temperature, pressure and internal energy.



The matter within a shell is regarded as being isolated from the matter within other shells. The shell may change in volume but the same mass remains inside of it.

The dynamics of the star can be simulated by studying the movement of these spherical shells. By looking at the way in which the radial position of each shell changes due to gravity and pressure, gives an understanding of the dynamics of the star as a whole.

6. Physics of supernova

6.1. Pressure

The star consists of a mixture of photon gas and ideal classical gas. The pressure of the stellar gas is then given by:

$$P = \frac{a}{3} T^4 + \frac{\rho k_B T}{\mu}$$

Where the two terms are respectively the partial pressures of the photon gas and the classical gas.

μ : The mean molecular mass of the classical gas.

k_B : The Boltzmann constant.

T: The temperature.

6.2. Energy

The temperature of a stellar shell changes due to changes in its energy. It is therefore of importance to first find the expressions for the total energy of a shell.

- The first part of the total energy is the internal energy. The internal energy of the mixture is given by the sum of the constituents' internal energies.

$$U = \frac{2}{3} k_B N T + a V T^4$$

Where the first contribution comes from the classical gas and the second from the photon gas.

N: number of particles.

A: radiation constant.

For high temperatures, the photon gas will become dominant over the classical gas.

- The second energy term is the kinetic energy, which is given by:

$$E_{kin} = \frac{1}{2} m_{shell} \left(\frac{v_{i+1} + v_i}{2} \right)^2$$

Where the average velocity of the two surfaces (i.e. the two shell radii) enclosing a shell of stellar material are taken to represent the velocity of the matter inside the shell.

- The third and final term is the gravitational potential energy, and it has negative contribution to the total energy. It is given by:

$$E_{grav} = - \frac{GM_{encl}m_{shell}}{r^2}$$

So combining the above equations gives the following expression for the total energy of a stellar shell:

$$U + E_{kin} + E_{grav} = \frac{2}{3} k_B N T + a V T^4 + \frac{1}{2} m_{shell} \left(\frac{v_{i+1} + v_i}{2} \right)^2 - \frac{GM_{encl}m_{shell}}{r^2}$$

6.3. Temperature

When one assumes that the shell expands and shrinks adiabatically, its temperature is related to its internal energy. Therefore, one can determine the change in temperature of a shell by means of the change in energy.

The internal energy of a shell can change in two ways: by exchanging heat and by performing work:

$$dU = d\omega + dq$$

The stellar gas is assumed to expand adiabatically so dq is set to zero.

$$dU = d\omega = -PdV$$

By replacing the internal energy and the pressure by their values (with $N = \frac{m_{shell}}{\mu}$) we'll have the following expression in terms of the change in temperature as:

$$dT = \frac{- \left(\frac{\rho k T}{\mu} + \frac{a T^4}{3} + a T^4 \right) dV}{\frac{3}{2} k \frac{m_{shell}}{\mu} + 4 a V T^3}$$

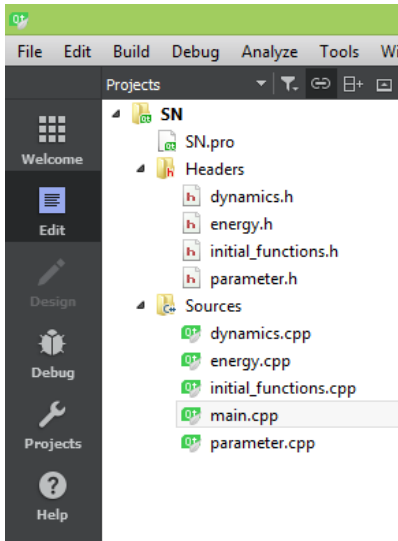
It describes how the temperature of a shell of stellar matter changes due to changes in its volume.

43 Chapter 2: Program code description

1. Classes

Our program is composed of 4 classes:

- Initial functions
- Dynamics
- Energy
- Parameter



The first class Initial_functions calculates the initial values of the pressure, temperature, density and the H constant.

The second class Dynamics calculate the acceleration, pressure generation and the temperature distribution

The third class Energy calculate the kinetic energy, internal energy and the gravitational energy.

The fourth class Parameter define the number of shell desired.

The initial conditions are specified in the main program.

For the full program see Annex A.

2. Results saving

A simple code has been added to the program to save the results automatically into files.csv.

```
#include "iostream"
#include "fstream"
#include "sstream"

string filename;

ofstream myfile;

stringstream b;
b<<i;//number of iteration
  filename= "SN_"+ b.str();
  filename+= ".csv";
```



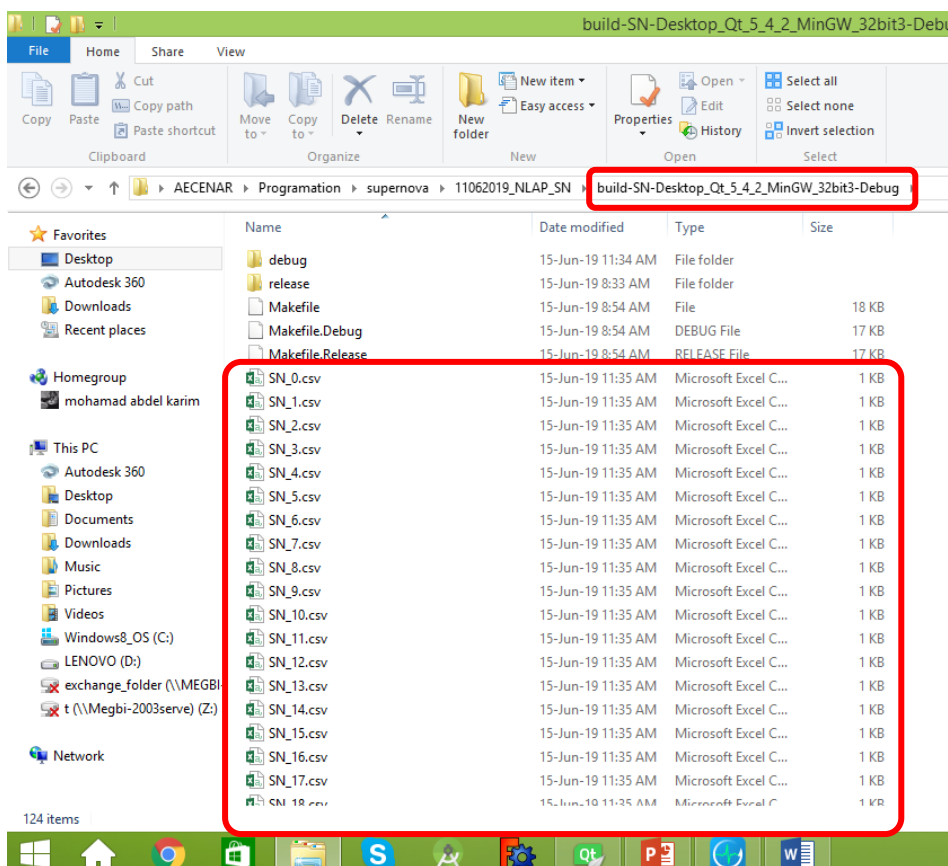
```

myfile.open(filename.c_str());

for(int k=0;k<d;k++){
    rold[k]=r[k]; //store the old radii
    accel[k]=dyn_acc(mass[k],r[k],rho[k],dp_over_dr[k],G);
    v[k]=v[k]*damping+accel[k] * dt;
    r[k]=r[k]+v[k] * dt+accel[k] * dt * dt;
    y=sqrt(r[k]*r[k]-(k+1)*(k+1));
    if(r[k]<0){
        r[k]=0;
    }
    myfile<<"<<r[k]<<","<<0,"<<r[k]<<","<<T[k]<<","<<dt<<","<<pressure[k]<<"\n";
}
myfile.close();

```

This code will create a file named SN000.csv for each iteration. They will be saved in the build folder.



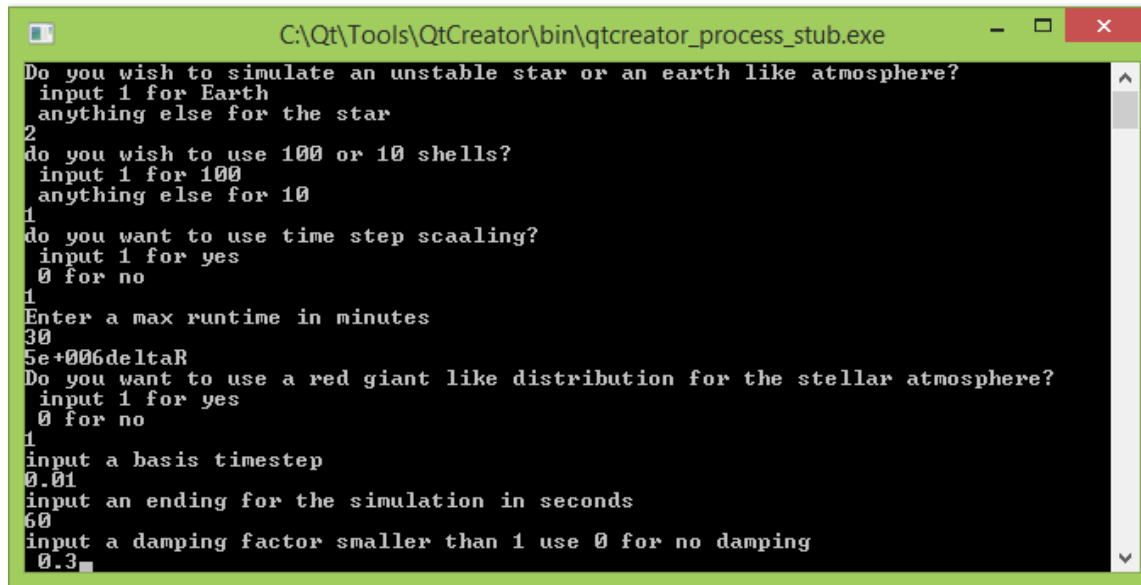
44 Chapter 3: Results

1. Input

The user first specifies if he wants to simulate an unstable star or an earth like atmosphere, then the number of shells desired (10 or 100 shells).

The program presents the option to have a time step scaling. The user should present a max runtime in minutes, a basis time step, an ending for the simulation and finally the damping factor.

The user should also choose whether he wants a neutron core or a red giant distribution.



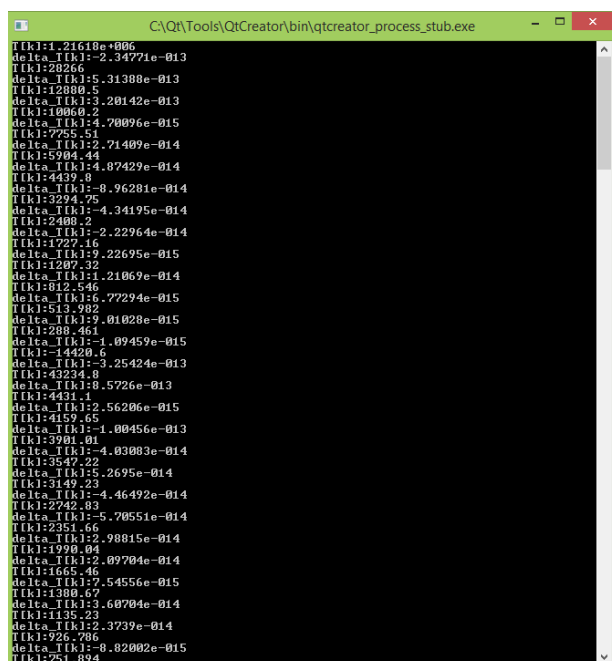
```

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Do you wish to simulate an unstable star or an earth like atmosphere?
input 1 for Earth
anything else for the star
2
do you wish to use 100 or 10 shells?
input 1 for 100
anything else for 10
1
do you want to use time step scaling?
input 1 for yes
0 for no
1
Enter a max runtime in minutes
30
5e+006deltaR
Do you want to use a red giant like distribution for the stellar atmosphere?
input 1 for yes
0 for no
1
input a basis timestep
0.01
input an ending for the simulation in seconds
60
input a damping factor smaller than 1 use 0 for no damping
0.3

```

2. Results

The results start to appear on the console and result files are created for each time step in the directory



```

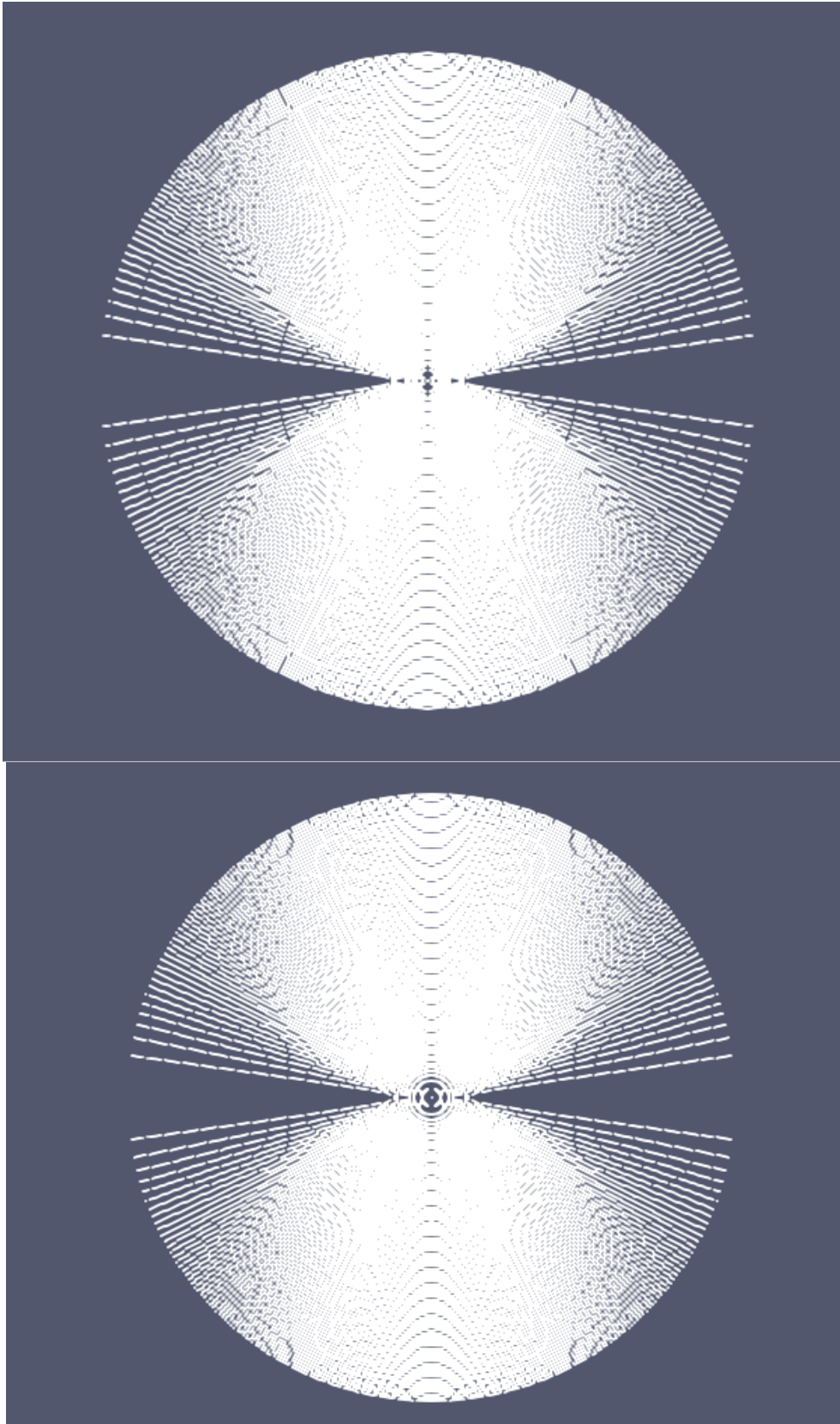
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
T(k):1.21618e+006
delta_T(k):-2.34771e-013
T(k):20266
delta_T(k):5.31388e-013
T(k):12080.5
delta_T(k):3.20142e-013
T(k):10060.2
delta_T(k):4.70096e-015
T(k):7755.51
delta_T(k):2.71409e-014
T(k):5904.44
delta_T(k):4.87429e-014
T(k):4439.0
delta_T(k):-0.96281e-014
T(k):3294.75
delta_T(k):-4.34195e-014
T(k):2400.2
delta_T(k):-2.22964e-014
T(k):1227.46
delta_T(k):9.22695e-015
T(k):1207.32
delta_T(k):1.21069e-014
T(k):812.546
delta_T(k):6.77294e-015
T(k):513.982
delta_T(k):9.01028e-015
T(k):288.461
delta_T(k):-1.09459e-015
T(k):14220.6
delta_T(k):-3.25424e-013
T(k):43234.8
delta_T(k):0.5726e-013
T(k):4431.1
delta_T(k):2.56286e-015
T(k):4159.65
delta_T(k):-1.00456e-013
T(k):3901.01
delta_T(k):-4.03003e-014
T(k):3547.22
delta_T(k):5.2695e-014
T(k):3149.23
delta_T(k):-4.46492e-014
T(k):2742.83
delta_T(k):-5.70551e-014
T(k):2351.66
delta_T(k):2.98815e-014
T(k):1990.04
delta_T(k):2.09704e-014
T(k):1665.46
delta_T(k):7.54556e-015
T(k):1300.67
delta_T(k):3.60704e-014
T(k):1135.23
delta_T(k):2.3739e-014
T(k):926.786
delta_T(k):-8.82002e-015
T(k):751.074

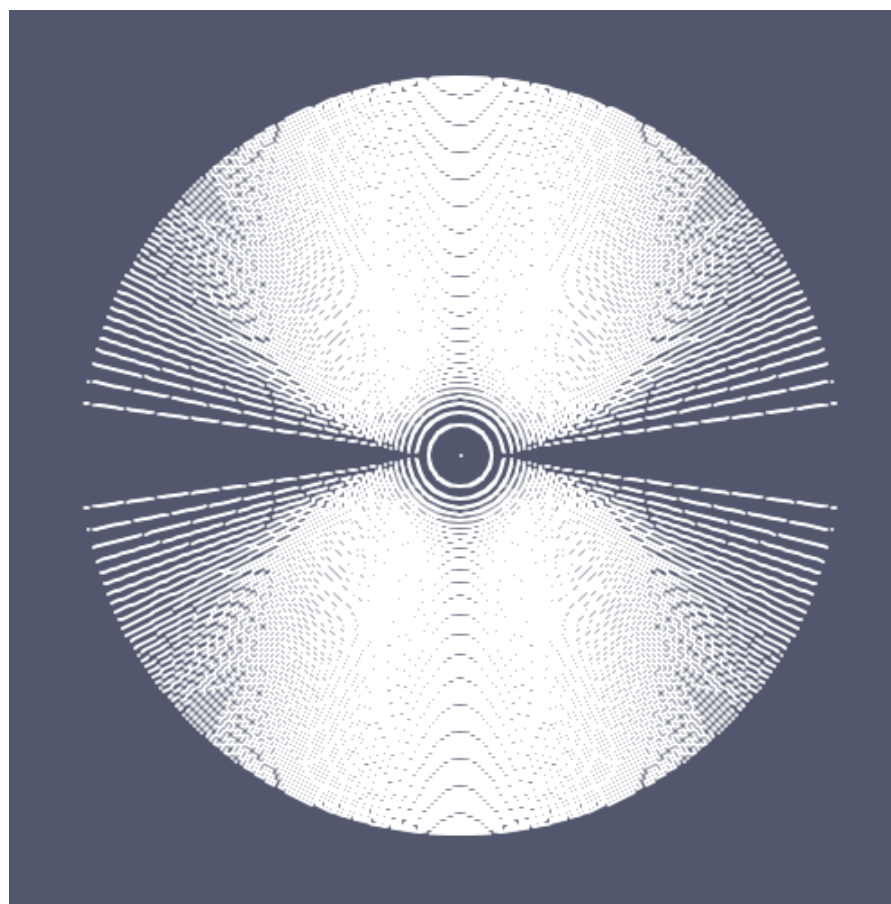
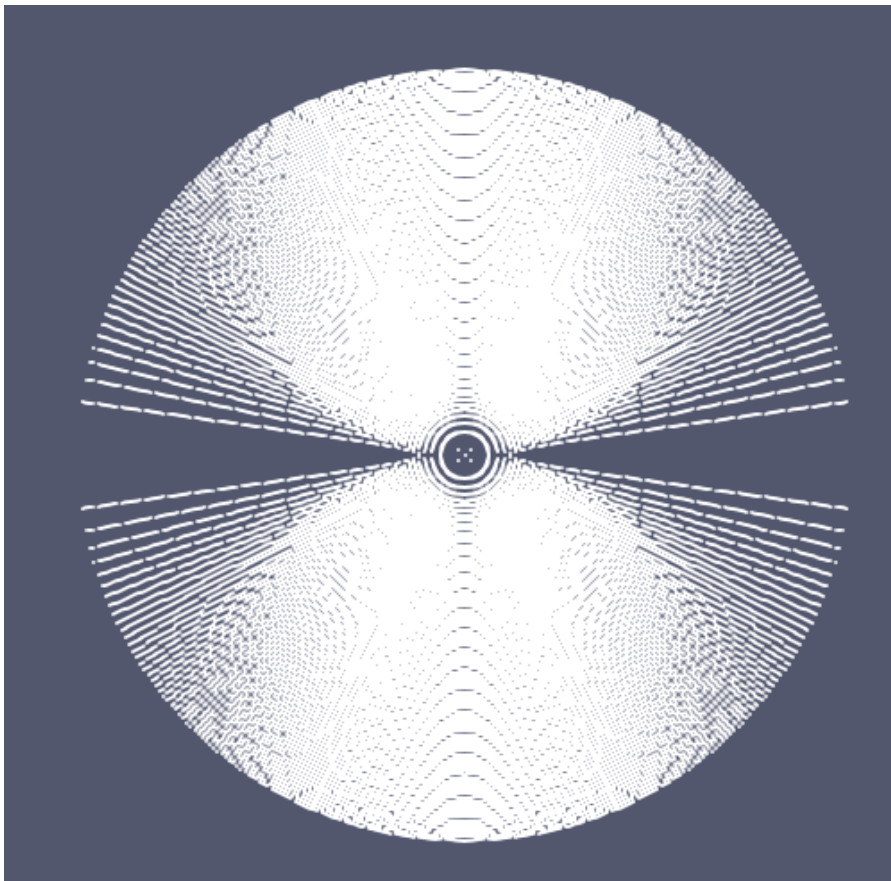
```

3. Results viewed in Paraview

The results shown in this section are for a mesh composed of 100 shells.

For 100 shells

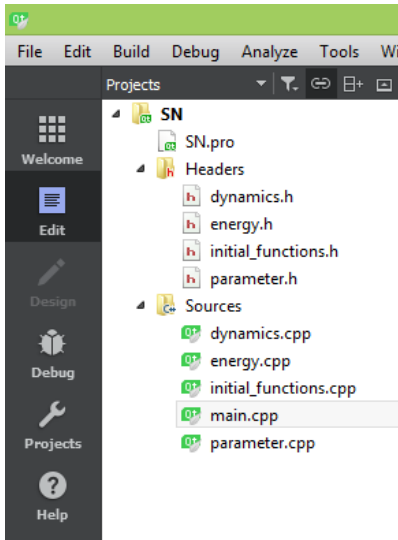




45 Annex A: Code listing

Our program is composed of 4 classes:

- Initial functions
- Dynamics
- Energy
- Parameter



A.1. Class Initial functions

Initial functions.h

```
#ifndef INITIAL_FUNCTIONS_H
#define INITIAL_FUNCTIONS_H
```

```
class Initial_Functions
{
public:
    Initial_Functions () {

    }

    double rhof(double beginrho,double exponentfactrho,double r,double Rearth,double
deltaR, double H_constant);
    double temp(double begintemp);
    double pressure(double beginpres,double exponentfactpres,double r,double
Rearth,double deltaR, double H_constant);
    double H_constant(double molecular_mass, double grav_acc,double avogadro,double T,
double gascons);
    double der(double ai,double ai_1,double bi,double bi_1);

};

#endif // INITIAL_FUNCTIONS_H
```

Initial functions.cpp

```
#include "initial_functions.h"
#include "math.h"

//initial density, temperature, and pressure functions
// density distribution at t=0
```

```
double Initial_Functions::rhof(double beginrho,double exponentfactrho,double r,double
Rearth,double deltaR, double H_constant){
double rhof;
    rhof=beginrho*exp(-exponentfactrho*(r-Rearth-deltaR)/H_constant);

return rhof;
}

// temp distribution at t=0
double Initial_Functions::temp(double begintemp){
    return begintemp;
}

// pressure distribution at t=0
double Initial_Functions::pressure(double beginpres,double exponentfactpres,double
r,double Rearth,double deltaR,double H_constant){
double P;
    P=beginpres*exp(-exponentfactpres*(r-Rearth-deltaR)/H_constant);

    return P;
}

// function for H_constant
double Initial_Functions::H_constant( double molecular_mass, double grav_acc,double
avogadro,double T,double gascon){
    double H;

    H=gascon*T/(grav_acc*molecular_mass*avogadro);

    return H;
}

//derivative function
double Initial_Functions::der(double ai,double ai_1,double bi,double bi_1){

    double k;

    k=(ai-ai_1)/(bi-bi_1);

return k;
}
```

A.2. Class dynamics

Dynamics.h

```
#ifndef DYNAMICS_H
#define DYNAMICS_H
```

```
class dynamics
{
public:
    dynamics () {

    }
```

```
double acc(double m_encl, double r, double rho, double dp_over_dr, double G);
double pressure_gen(double T, double rho, double molecular_mass,double a,double boltz);
double tempdistr(double r, double T1, double T2,double R1,double R2);
```

```
};

#endif // DYNAMICS_H
```

Dynamics.cpp

```
#include "dynamics.h"
#include "math.h"

//acceleration
double dynamics::acc(double m_encl, double r, double rho, double dp_over_dr, double G){
double acc;
if(r<=0){
return 0;
}
else{
acc=-G*m_encl/(r*r)-(1/rho)*dp_over_dr;
return acc;
}
}

//Pressure
double dynamics::pressure_gen(double T, double rho, double molecular_mass,double
a,double boltz){
double press;
press=(a/3)*pow(T,4)+rho*boltz*T/molecular_mass;
return press;
}

//this function is used for interpolating the temperature between element layers
//in the situation of stellar distribution with elements
double dynamics::tempdistr(double r, double T1, double T2,double R1,double R2){
double tempdist;
tempdist=T1*exp(-(1/(R2-R1))*log(T1/T2)*r);
return tempdist;
}
}
```

A.3. Class energy

Energy.h

```
#ifndef ENERGY_H
#define ENERGY_H

class Energy
{
public:
Energy(){

}

double gravenergy(double mass, double massencl,double r, double G);
double internalenergy(double volume,double temperature,double mass, double
molecularmass,double enercon,double boltz);
double kineticenergy(double mass, double velocity);
};

#endif // ENERGY_H
```

Energy.cpp

```
#include "energy.h"
```

```
#include "math.h"

//gravitational energy
double Energy::gravenergy(double mass, double massencl, double r, double G){
    return mass*massencl*G/r;
}

//internal energy
double Energy::internalenergy(double volume, double temperature, double mass, double
molecularmass, double enercon, double boltz){

    return
enercon*volume*pow(temperature,4)+(3/2)*boltz*mass/molecularmass*temperature;
}

//kinetic energy
double Energy::kineticenergy(double mass, double velocity){

    return 0.5*mass*pow(velocity,2);
}
}
```

A.4. Class parameter

Parameter.h

```
#ifndef PARAMETER_H
#define PARAMETER_H

class parameter
{
public:
    parameter(){

    }
int shellnumber(int wanthundred);
};

#endif // PARAMETER_H
```

Parameter.cpp

```
#include "parameter.h"

int parameter::shellnumber(int wanthundred){
    int d;
    if(wanthyundred==1){
        d=100;
    }
    if(wanthyundred!=1){
        d=10;
    }
    return d;
}
}
```

A.5. Main program

```
#include <QCoreApplication>
#include "math.h"
```



```

#include "dynamics.h"
#include "energy.h"
#include "parameter.h"
#include "initial_functions.h"
#include "stdio.h"
#include "ctime"
#include "algorithm"
#include "iostream"
#include "fstream"
#include "sstream"
#include <QtCore/QString>
#include <QtCore/QFile>
#include <QtCore/QDebug>
#include <QtCore/QTextStream>

using namespace std;

int main(int argc, char **argv)
{
int atmassim;
int wanthundred;
int wantdtscaling;
int wantdensity;
double tmax;
double shellnumber;
double deltaR;
double Rearth;
double beginrho;
double beginpress;
double begintemp;
double exponentfactorrho;
double exponentfactorpress;
double Ratm;
double Mstar;
double mol;
double dt;
double t_final;
double dampfactor;
double pi;
double boltz;
double gascon;
double avogadro;
double c;
double sig;
double a;
double hbar;
double enercon;
double G;
double grav_acc;

/*//////////////////////////////////////
//      Initial conditions      //
////////////////////////////////////*/

double Msolar=1.99e30;//mass of the sun in kg
double Munit=1.660538921e-27;//atomic mass unit in kg
double Rsolar=6.955e8;//Radius of the sun in m

//user input
cout<< "Do you wish to simulate an unstable star or an earth like atmosphere?\n input 1
for Earth\n anything else for the star\n";
cin >> atmassim;

```

```

cout<< "do you wish to use 100 or 10 shells?\n input 1 for 100\n anything else for
10\n";
cin>> wanthundred;
cout<<"do you want to use time step scaling?\n input 1 for yes\n 0 for no\n";
cin>> wantdtscaling;
cout<<"Enter a max runtime in minutes\n";
cin>> tmax;

if(atmossim==1){//use earth conditions
  //set the shell number
  shellnumber=1;
  if(wanthundred==1){
    shellnumber=0.1;
  }
  //set the shell thickness
  deltaR=5000 * shellnumber;
  //Radius of Earth
  Rearth=6400 * pow(10,3) - deltaR;
  //set parameters for use in the initial distribution functions
  beginrho=1.251;
  beginpress=1e5/beginrho;
  begintemp=293;//293000*5.55 stability point
  //set a factor that is used to adjust the density distr.
  //it is 1 here because here no adjustment is necessary
  exponentfactorrho=1;
  exponentfactorpress=exponentfactorrho;
  //set the title of the output graph
  //graphtitle='numerical simulation of the atmosphere of the earth';
  //radius of the atmosphere
  Ratm=50000+Rearth;
  //set the mass of the earth (the name Mstar is still used for this variable
  Mstar=5.9721986 * pow(10,24);
  //set that the division in element layers as used in the star simulation is not
used
  wantdensity=0;
  //this variable is the molecular mass of nitrogen
  mol=4.65173 * pow(10,-26);
}

else{// use star conditions
  //initially set that the element didtribution is not used
  wantdensity=0;
  //adjust the number of shells
  shellnumber=1;
  if(wanthundred==1){
    shellnumber=0.1;
  }
  deltaR=50000000 * shellnumber;
  cout<<deltaR<<"deltaR \n";
  //set the radius of the star core (the term Rearth is a remnant of earlier version
of the model
  //and has been kept in. The variable does in fact denote the neutron star core's
radius
  Rearth=12000-deltaR;
  //set the radius of the stellar atmosphere
  Ratm=500000000+Rearth;
  //set the parameters for initial distribution
  exponentfactorrho=0.0001;
  exponentfactorpress=5;
  beginrho=1.251e5;
  begintemp=237000;
  beginpress=1e33;
  //set the mass of the stellar core

```

```

Mstar=Mstellar * 1.4;
//set the title of the output graph
// graphtitle='numerical simulation of a supernova explosion';
if(wanthundred==1){//check whether the user has chosen 100 shells
    //Ask whether the user wants to use the stellar atmosphere distribution
    //where it is divided in a number of layers containing certain elements
    cout<<"Do you want to use a red giant like distribution for the stellar
atmosphere?\n input 1 for yes\n 0 for no\n";
    cin>>wantdensity;

}
//this variable is the molecular mass of hydrogen
mol=3.34745e-27;
}

//ask the user to input some time variables
cout<<"input a basis timestep\n";
cin>> dt;
cout<<"input an ending for the simulation in seconds\n";
cin>> t_final;
cout<< "input a damping factor smaller than 1 use 0 for no damping\n ";
cin>> dampfactor;

//define a number of constants. All physical constants are in SI units
//Pi
pi=3.141592653;
//the boltzmann constant
boltz=1.38064852e-23;
//the gas constant
gascon=8.3144;
//Avogadro's number
avogadro=6.02214129e23;
//speed of light
c=299792458;
//radiation constants
sig=5.67e-8;
a=(4 * sig)/c;
//reduced Planck's constant
hbar=6.62607015e-34;

//define an energy constant.
// This combination of constants occurs multiple times in the program and is therefore
defined for convenience.

enercon=(pow(pi,2) * pow(boltz,4))/(15 * pow(c,3) * pow(hbar,3));
//gravitation constant
G=6.67408e-11;
//gravitational acceleration on earth
grav_acc=9.807;

/*//////////////////////////////////////
// Calculation of the initial Arrays //
//////////////////////////////////////*/

//set the array containing the shell radii
parameter par;
int d;//array length

d=par.shellnumber(wanthundred);
double r[d];

for(int k=0;k<d;k++){
    r[0]=Rearth+deltaR;

```

```

    r[k+1]=r[k]+deltaR;

}
//initialize an array to be used later
double rold[d];
//create an array giving the mean molecular mass of the material inside shells
double molecularmass[d];
for(int i=0;i<d;i++){
    rold[i]=0;
    molecularmass[i]=mol;
}

//calculate the density between radii (loop_rho) and radii themselves (rho)
double loop_rho[d];
double rho[d];
double H_constant=0;
double temp;
Initial_Functions INIT;

for(int i=0;i<d;i++){
    temp=INIT.temp(begintemp);
    H_constant=INIT.H_constant(molecularmass[i],grav_acc,avogadro,temp,gascon);

loop_rho[i]=INIT.rhof(beginrho,exponentfactorrho,r[i]+0.5*deltaR,Rearth,deltaR,H_constant);
rho[i]=INIT.rhof(beginrho,exponentfactorrho,r[i],Rearth,deltaR,H_constant);
}

//initial some arrays to be used later
double mass[d];
double mass_shell[d-1];
for(int i=0;i<d;i++){
    mass[i]=Mstar;
}
for(int i=1;i<d-1;i++){
mass_shell[i]=0;
}

//calculation of the temperature array
double T[d];
double delta_T[d];
double delta_L[d];
for(int i=0;i<d;i++){
    T[i]=INIT.temp(begintemp);
    delta_T[i]=0;
    delta_L[i]=0;
}

dynamics dyn;
if(wantdensity==1){//check if the user wants to use the red giant-like element layer
division
    //set the boundary radii of the elements layers
    r[0]=12e3;
    r[10]=11020e3;
    r[20]=21738e3;
    r[25]=26395e3;
    r[40]=69476e3;
    r[60]=389066e3;
    r[99]=1000*Rsolar;
    //set the temperatures at the boundaries
    T[0]=334e7;
    T[10]=199e7;
    T[20]=157e7;
    T[25]=87e7;

```

```

T[40]=18.8e7;
T[60]=3.69e7;
T[99]=3500;

//define the radii of shells in between layers and interpolate the temperature

for(int k=1;k<10;k++){
  r[k]=r[0]+(r[10]-r[0])/10 * k;
  T[k]=dyn.tempdistr(r[k],T[0],T[10],r[0],r[10]);
  molecuarmass[k]=28/15 * Munit;
}

for(int k=11;k<20;k++){
  r[k]=r[10]+(r[20]-r[10])/10 * (k-10);
  T[k]=dyn.tempdistr(r[k],T[10],T[20],r[10],r[20]);
  molecuarmass[k]=16/9 * Munit;
}

for(int k=21;k<25;k++){
  r[k]=r[20]+(r[25]-r[20])/5 * (k-20);
  T[k]=dyn.tempdistr(r[k],T[20],T[25],r[20],r[25]);
  molecuarmass[k]=20/11 * Munit;
}

for(int k=26;k<40;k++){
  r[k]=r[25]+(r[40]-r[25])/15 * (k-25);
  T[k]=dyn.tempdistr(r[k],T[25],T[40],r[25],r[40]);
  molecuarmass[k]=12/7 * Munit;
}

for(int k=41;k<60;k++){
  r[k]=r[40]+(r[60]-r[40])/20*(k-40);
  T[k]=dyn.tempdistr(r[k],T[40],T[60],r[40],r[60]);
  molecuarmass[k]=4/3 * Munit;
}

for(int k=61;k<100;k++){
  r[k]=r[60]+(r[99]-r[60])/40*(k-60);
  T[k]=dyn.tempdistr(r[k],T[60],T[99],r[60],r[99]);
  molecuarmass[k]=0.6 * Munit;
}

//define the mass of the matter inside each shell
for(int k=0;k<21;k++){
  mass_shell[k]=0.12 * Msolar/10;
}

for(int k=21;k<26;k++){
  mass_shell[k]=0.12 * Msolar/5;
}

for(int k=26;k<41;k++){
  mass_shell[k]=0.12 * Msolar/15;
}

for(int k=41;k<61;k++){
  mass_shell[k]=0.12 * Msolar/20;
}

for(int k=61;k<99;k++){
  mass_shell[k]=18 * Msolar/40;
}

//create the array containing enclosed masses
for(int i=0;i<d-1;i++){
  mass[i+1]=mass[i]+mass_shell[i];
}

```

```

    }
}

//define the mass arrays when the red giant-like element layer division is not used
if(wantdensity!=1){
    for(int i=0;i<d-1;i++){
        mass_shell[i]=4 * pi/3 * (pow(r[i+1],3)-pow(r[i],3)) *
INIT.rhof(beginrho,exponentfactorrho,r[i]+0.5 * deltaR,Rearth,deltaR,H_constant);
        mass[i+1]=mass[i]+mass_shell[i];
        //akkk=mass_shell[35]*9
        //mass_shell[35] 10*mass_shell[35]
        //for i in range(35,r_length-1):
        //mass[i+1]=mass[i+1]+akkk
    }
}
//calculate the densities of the stellar matter in case of the red giant-like element
layer division
if(wantdensity==1){
    for(int k=0;k<d-1;k++){
        loop_rho[k]=mass_shell[k]/(4 * pi/3 * (pow(r[k+1],3)-pow(r[k],3)));
    }
    for(int k=1;k<d-1;k++){
        rho[k]=(loop_rho[k-1]+loop_rho[k])/2;
        rho[d-1]=rho[d-2]/10;
        cout<<"rho " <<rho[k]<<"\n";
    }
}

//initialise the energy array
double energybegin[d];
for(int i=0;i<d;i++){
    energybegin[i]=0;
}
//calculate the pressure inside each shell
double pressure[d];
for(int i=0;i<d;i++){
    pressure[i]=dyn.pressure_gen(T[i],rho[i],molecularmass[i],a,boltz);
}
//calculation of the dp over dr array
//rvooder contains the positions of the center of each shell
double rvooder[d];
for(int i=1;i<d;i++){
    rvooder[i]=1;
}
double dp_over_dr[d];
for (int k = 0; k < d-1; ++k) {
    rvooder[k]=r[k]+(r[k+1]-r[k])/2;
    rvooder[d-1]=r[d-1]+0.5*deltaR;
    dp_over_dr[k]=INIT.der(pressure[k],pressure[k-1],rvooder[k],rvooder[k-1]);
}
//creating the intial speed array, we assume ths is 1m/s at all radial coordinates
double v[d];
for(int i=0;i<d;i++){
    v[i]=1;
}
//initialise the acceleration array
double accel[d];
for(int i=0;i<d;i++){
    accel[i]=0;
}
//Data storage
double v_total[100][d];
double r_total[100][d];

```

```

double rho_total[100][d];
double T_total[100][d];
double dp_over_dr_total[100][d];
double accel_total[100][d];
double P_total[100][d];
double rho_new[d-1];

for(int i=0;i<d;i++){
    v_total[0][i]=v[i];
    r_total[0][i]=r[i];
    rho_total[0][i]=rho[i];
    T_total[0][i]=T[i];
    dp_over_dr_total[0][i]=dp_over_dr[i];
    accel_total[0][i]=accel[i];
    P_total[0][i]=pressure[i];
}
for(int i=0;i<d-1;i++){
    rho_new[i]=0;
}

//define error as 0 , meaning no shell overlap has occurred yet
int error;
error=0;

//initialise time storage
vector<double> totime;
totime.push_back(0);

//initialise the number of iterations of the program
int iterations;
iterations=0;
//save the old time step for use in adjusting it later on
double dtold;
dtold=dt;
//set the first element of the array containing the values of the radii at a previous
time step
rold[0]=r[0];
//initialise some other time variables
double timevar;
time_t t1;
timevar=dt;
time(&t1);
tmax=tmax * 60;

//set the initial two shell distances for the red giant like element layer division
situation
double deltaR1=0;
double deltaR2=0;
if(wantdensity==1){
    deltaR1=r[15]-r[14];
    deltaR2=r[35]-r[34];
}
Energy Energ;
//calculate the initial total energy
for(int k=0;k<d-1;k++){
    energybegin[k]=Energ.internalenergy(((4/3) * pi * (pow(r[k+1],3)-
pow(r[k],3))),T[k],mass_shell[k],molecularmass[k],enercon,boltz)\
    -Energ.gravenergy(mass_shell[k],mass[k],r[k],G)\
    +Energ.kineticenergy(mass_shell[k],(v[k+1]+v[k])/2);
}

//initialise the total energy array that will be changed over the course of the runtime
double energytotal[100][d];

```

```

for(int k=0;k<d-1;k++){
    energytotal[0][k]=energybegin[k];
}
if(atmossim!=1 && wanthundred!=1){
    T[0]=1e9;
    T[1]=8.5e8;
    T[2]=7e8;
    T[3]=6e8;
    T[4]=5e8;
    T[5]=3e8;
    T[6]=1e8;
    T[7]=1e7;
    T[8]=1e6;
    T[9]=1e5;
    rho[0]=10 * rho[0];
    rho[1]=9 * rho[1];
    rho[2]=8 * rho[2];
    rho[3]=7.5 * rho[3];
    rho[4]=8 * rho[4];
    rho[5]=8 * rho[5];
    rho[6]=8 * rho[6];
    rho[7]=8 * rho[7];
    rho[8]=2 * rho[8];

    for(int k=0;k<10;k++){
        cout<<"T : "<<T[k]<<" et rho:" << rho[k]<<"\n";
    }
}

//start the time loop in which all quantities are evaluated step by step
int i=0;
int l=1;
double damping;
double x;
double DeltaV,A1,B1,C1,D1,E1;
double energy[d-1];
double y;
string filename;
while(timevar<=t_final && error==0){//keep running while max. time has not been
exceeded and no overlap error has occurred yet
    totime.push_back(timevar); //update the time array
    dtold=dt;
    //calculation of the new r and v array
    cout<<"dt " <<dt<<"\n";
    damping =1-dampfactor * dt;//calculate the damping
    cout<<"damp " <<damping<<"\n";
    ofstream myfile;

    stringstream b;
    b<<i;
    filename= "SN_"+ b.str();
    filename+= ".csv";
    myfile.open(filename.c_str());
    myfile<<"x,y,z,T\n";
    double stp,x1,y1;
    // cout<<"x,y,z,r,v,dt,accel\n";
    for(int k=0;k<d;k++){
        x1=r[k];
        rold[k]=r[k];//store the old radii
        accel[k]=dyn.acc(mass[k],r[k],rho[k],dp_over_dr[k],G);
        v[k]=v[k]*damping+accel[k] * dt;
        r[k]=r[k]+v[k] * dt+accel[k] * dt * dt;
        if(r[k]<0){
            r[k]=0;

```



```

}
stp=r[k]/100;
for(int i=0;i<100;i++){
    x1=x1-stp;
    y1=sqrt(abs(pow(r[k],2)-pow(x1,2)));
myfile<<x1<<","<<y1<<","0,"<<T[k]<<"\n";
myfile<<x1<<","<<-y1<<","0,"<<T[k]<<"\n";
myfile<<-x1<<","<<y1<<","0,"<<T[k]<<"\n";
myfile<<-x1<<","<<-y1<<","0,"<<T[k]<<"\n";
}

}
myfile.close();
//check whether shells overlap
for(int k=1;k<d;k++){
if(r[k]<r[k-1]){
    cout<<"overlap error\n";
    error=1;
    cout<< "overlap shell"<<k<<"\n";
    break;
}
}
time_t t2;
time(&t2);
if((t2-t1)>=tmax){
    error=1; //here error is used to end the loop when max time has been exceeded
    //if does not mean that there is an error in the program
    cout<< "max run time exceeded\n";
}

//storage of v,r,T, rho and dp_over-dr
for(int j=0;j<d;j++){
    v_total[1][j]=v[j];
    r_total[1][j]=r[j];
    rho_total[1][j]=rho[j];
    T_total[1][j]=T[j];
    dp_over_dr_total[1][j]=dp_over_dr[j];
    accel_total[1][j]=accel[j];
    P_total[1][j]=pressure[j];
}

//update the temperature and pressure
for(int k=0;k<d-1;k++){
    x=mass_shell[k]/(4 * pi/3 * (pow(r[k+1],3)-pow(r[k],3))); //temporarily store
the density in the variable x

    //calculate the terms used in the deltaT equation
    DeltaV=(4/3) * pi * ((pow(r[k+1],3)-pow(r[k],3))- (pow(roid[k+1],3)-
pow(roid[k],3)));
    A1=x * boltz * T[k]/(molecularmass[k]);
    B1=a * pow(T[k],4);
    C1=a/3 * pow(T[k],4);
    D1=(3/2) * boltz * mass_shell[k]/molecularmass[k];
    E1=4 * a * pi * (4/3) * (pow(r[k+1],3)-pow(r[k],3)) * pow(T[k],3);
    delta_T[k]=(-1 * (A1+B1+C1) * DeltaV)/(D1+E1); //caculate delta T
    loop_rho[k]=x; //set the density equal to the temporary variable x
}

}

for(int k=1;k<d-1;k++){
    delta_L[k]=4 * sig * 4 * pi * (pow(T[k-1],4) * pow(r[k],2)+T[k+1] *
pow(r[k+1],2)-T[k] * (pow(r[k],2)+pow(r[k+1],2)));
}
//update the temperature array

```

```
for(int k=1;k<=d;k++){
    T[k]=T[k]+delta_T[k];
    cout<<"T[k]:"<<T[k]<<"\n";
    cout<<"delta_T[k]:"<<delta_T[k]<<"\n";
}

for(int k=0;k<d-1;k++){//update the pressure
    pressure[k]=dyn.pressure_gen(T[k],loop_rho[k],molecularmass[k],a,boltz);
    if(k==1 && i%200 ==0 ){
        cout<<"timevar"<<timevar<<"\n";//print the time step to show the user how far
the program is
    }
}

//initialise the energy array
//calculate the energy of each shell
for(int k=0;k<d-1;k++){
    energy[k]=Energ.internalenergy(((4/3) * pi * (pow(r[k+1],3)-
pow(r[k],3))),T[k],mass_shell[k],molecularmass[k],enercon,boltz)-
Energ.gravenergy(mass_shell[k],mass[k],r[k],G)+Energ.kineticenergy(mass_shell[k],(v[k+1
]+v[k])/2);
}
//update the energy storage
for(int k=0;k<d;k++){
    energytotal[1][k]=energy[k];
}

//calculate the density at shell radii
for(int k=1;k<d-1;k++){
    rho[k]=(loop_rho[k-1]+loop_rho[k])/2;
}
//create the array containing the positions of the centers of the shells
for(int k=0;k<d-1;k++){
    rvooder[k]=r[k]+(r[k+1]-r[k])/2;
    rvooder[d-1]=r[d-1]+0.5 * deltaR;
}
//calculate the pressure gradients
for(int k=1;k<d;k++){
    dp_over_dr[k]=INIT.der(pressure[k],pressure[k-1],rvooder[k],rvooder[k-1]);
}
//adjust the time step size
double rshift[d];
double vshift[d];
double newdeltaR[d];
double rcheck;
double vcheck;
double vavg;
double deltaRavg;
double dtfactor;

if(wantdtscaling==1 && wantdensity!=1){

    for(int k=0;k<d;k++){
        rshift[k]=r[k+1];
        vshift[k]=v[k+1];
    }

    for(int k=0;k<d;k++){
        newdeltaR[k]=rshift[k]-r[k];
    }
}
```

```

double min1,max1;
min1=newdeltaR[0];
max1=v[0];
for(int i=0 ; i<d ; i++){
    if(min1>newdeltaR[i]){
        min1=newdeltaR[i];
    }

    if(max1<v[i]){
        max1=v[i];
    }
}

rcheck=min1;
vcheck=abs(max1);
double sum1=0,sum2=0;
for(int i=0;i<d;i++){
    sum1+=v[i];
    sum2+=newdeltaR[i];
}
vavg=abs(sum1/d);
deltaRavg=sum2/d;
dtfactor=pow((rcheck/deltaR),2);

if( dtfactor>0.000001){//check if the time step size is larger than the allowed
minimum
dt=dtold * dtfactor;
}
else{
    cout<<"dt reached minimum\n";//tell the user that the time step has reached
its lowest value
    //indicating that the program is highly unstable and liable to have shell
overlap
}

}

if(wantdensity==1 && wantdtscaling==1){//adjust the time step size when the red
giant-like element layer is used
for(int k=0;k<d;k++){
    rshift[k]=r[k+1];
    newdeltaR[k]=rshift[k]-r[k];
}

double rcheck1=0,rcheck2=0,min1,min2,dtfactor1=0,dtfactor2=0;
min1=newdeltaR[0];
min2=newdeltaR[25];
for(int i=0;i<25;i++){

    if(min1>newdeltaR[i]){
        min1=newdeltaR[i];
    }
}
for(int i=25;i<99;i++){

    if(min2>newdeltaR[i]){
        min2=newdeltaR[i];
    }
}
rcheck1=min1;
rcheck2=min2;
for(int k=1;k<26;k++){
    dtfactor1=pow((rcheck1/deltaR1),2);
}

```

```
        for(int k=26;k<99;k++){
            dtfactor2=pow((rcheck2/deltaR2),2);
        }
        if(dtfactor1<dtfactor2){
            dtfactor=dtfactor1;
        }
        else{
            dtfactor=dtfactor2;
        }
        cout<<"dt factor "<<dtfactor<<"\n";
        if(dtfactor>0.00001){
            dt=dtold * dtfactor;
        }
        else{
            cout<<"dt reached minimum1:\n";
        }
    }
    //increase the total time by dt
    timevar=timevar+dt;
    iterations=iterations+1;
    i+=1;
    l+=1;
    cout<<i<<"\n";
}

//print the initial and final energies so the user can compare them
cout<<"Energybegin,EnergyEnd\n ";
for(int i=0;i<d;i++){
    cout<<energybegin[i]<<","<<energy[i]<<"\n";
}

cout<<"number of iterations: "<<iterations<<" \n";

}
```

IAP-PSC (Plasma Simulation Code) (2019)



IAP-PSC
Plasma Simulation Code
(Particle-in-Cell Code)

Authors:
Maryam ABDEL-KARIM
Editor: Samir Mourad

Last Update: 23.08.2019

D:\AECENAR\IAP\IAP-TheoreticalAstrophysics\IAP-PSC\230819_14IAP-PSC.docx

Content

ERROR! BOOKMARK NOT DEFINED. **PREFACE**

547 BASICS 1

- 1.1 LASER-MATTER INTERACTION 547
 - 1.1.1 *Plasma definition* 547
 - 1.1.2 *Laser-cutting* 548

550 PLASMA WAVES KINETIC THEORY 2

- 2.1 GOVERNING EQUATIONS, VLASOV-BOLTZMANN EQUATION 550
 - 2.1.1 *Transport equations, Maxwell Equations* 550
 - 2.1.1.1 The Boltzmann collision operator 551

553 NUMERICAL MODEL 3

- 3.1 MESHING 553

554 SOME PUBLIC CODES / RESEARCH GROUPS 4

- 4.1 THE PLASMA THEORY AND SIMULATION GROUP PTSG UNIV. OF CALIFORNIA, BERKLEY 554
 - 4.1.1 *People of PTSG* 555
 - 4.1.2 *Projects* 555
 - 4.1.2.1 Current Projects in PTSG 555
 - 4.1.2.2 Recently Completed Projects in PTSG 555
 - 4.1.3 *PTSG Software* 555
 - *General Information* 555
 - *Acknowledgments* 555
 - *Description* 556
 - *Distribution* 556
 - Codes that require xgrafx* 556
 - Codes with own version of xgrafx* 556
 - Python/Matlab based codes* 556
 - Code Consulting and Maintenance* 557
 - 4.1.4 *Publications Using PTSG Codes, Worldwide* 557
 - 4.1.5 *Workshops* 557
 - 4.1.5.1 Plasma Device Workshop 2009 (PDW2009) 557

558 COMPARISON BETWEEN ACTUAL IAP-PCS AND XPDP2 5

- 5.1 DESCRIPTION OF XPDP2 558
 - 5.1.1 *Abstract of [1]:* 558
 - 5.1.2 *Abstract of [2]:* 558
 - 5.1.3 *xpdp2 Code Description* 559

563 IAP-PSC PROGRAM 6

- 6.1 DESCRIPTION 563
- 6.2 THE CODE 564
 - 6.2.1 *Class 1: parameters* 564
 - 6.2.2 *Class 2: Lorentz* 566
 - 6.2.3 *Class 3: pos_veloct* 566
 - 6.2.4 *Class 4: Dens_current* 567
 - 6.2.5 *Class 5: Maxwell_equat* 567

6.2.6 *Main program* 569
6.3 RESULTS 575

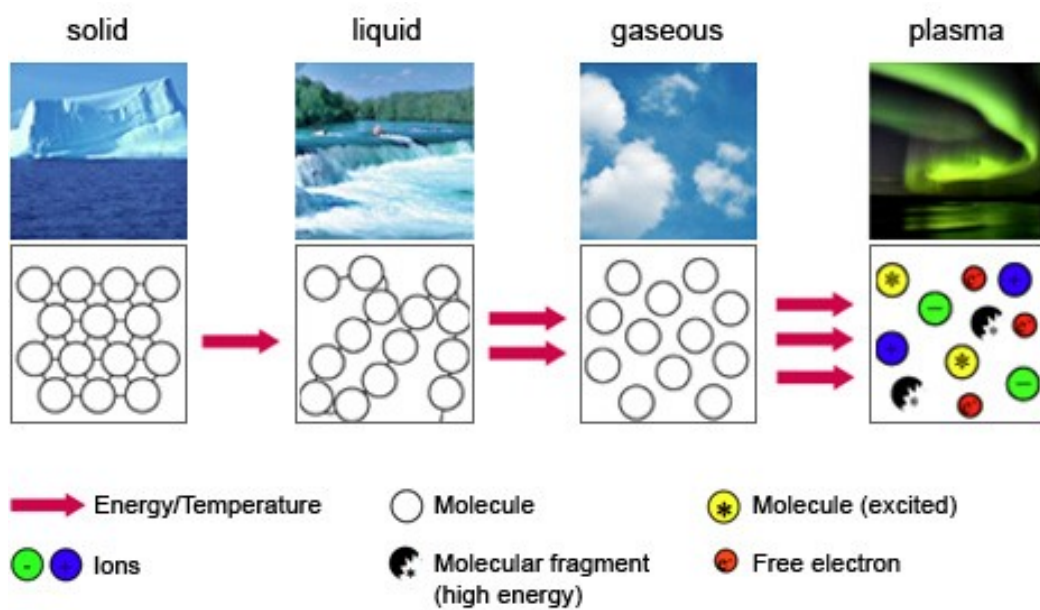
577 LITERATURE

46 Basics

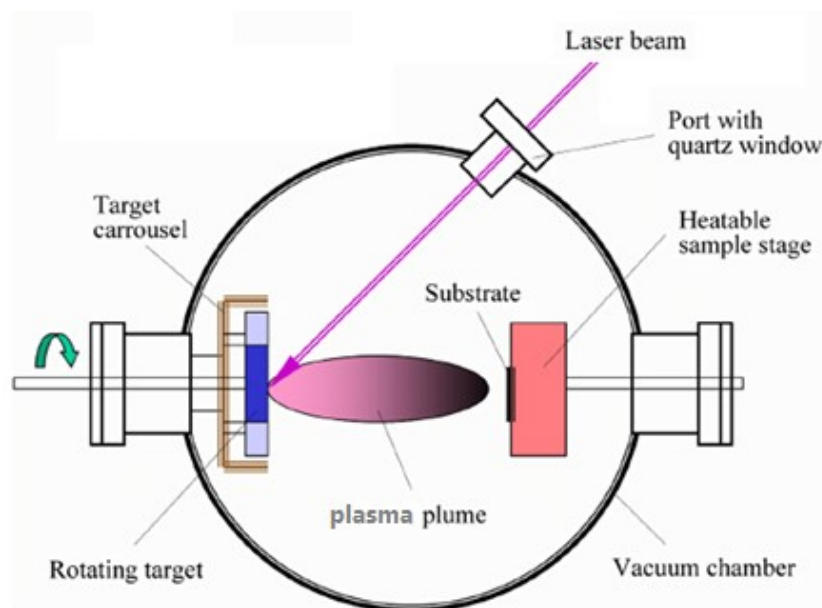
46.1 Laser-matter interaction

46.1.1 Plasma definition

Plasma is considered the fourth state of matter. The three other states are solid, liquid, and gas. Plasma is a cloud of protons, neutrons and electrons where all the electrons have come loose from their respective molecules and atoms, giving the plasma the ability to act as a whole rather than as a bunch of atoms. A plasma is more like a gas than any of the other states of matter because the atoms are not in constant contact with each other, but it behaves differently from a gas. It has what scientists call collective behavior. This means that the plasma can flow like a liquid or it can contain areas that are like clumps of atoms sticking together.

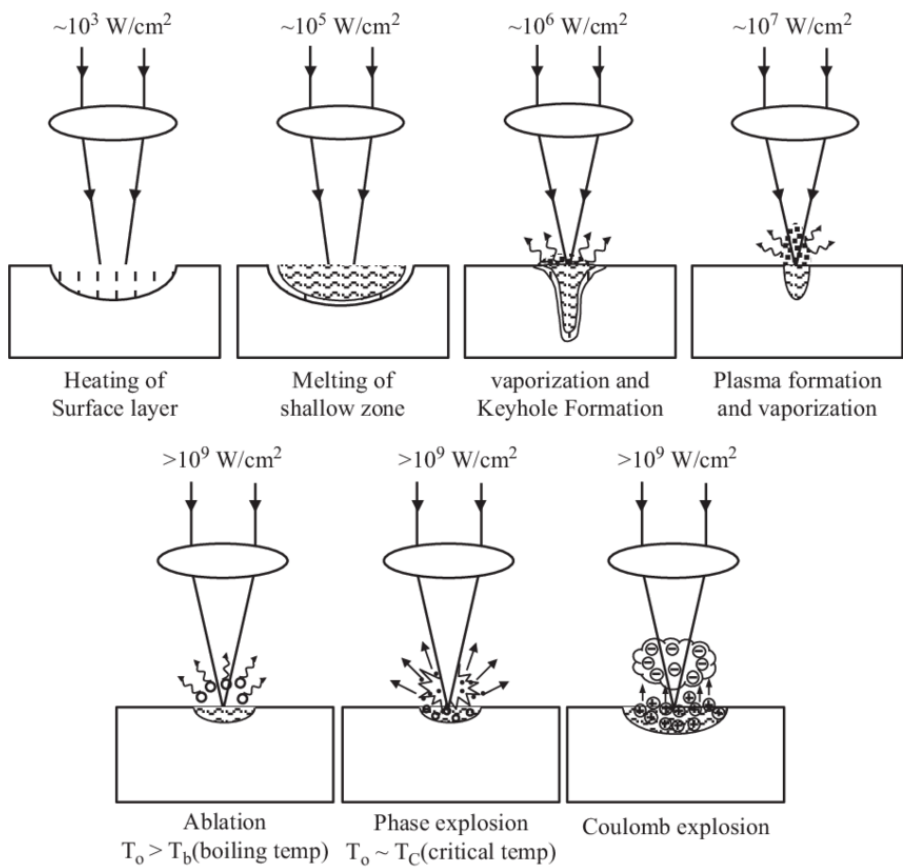


Plasma results from the interaction between a laser beam and a metal target (see figure below).



The power of the laser used affect the phenomena that will occur.

laser used affect the phenomena that will occur.



46.1.2 Laser-cutting

There are many different methods in cutting using lasers, with different types used to cut different material. Some of the methods are vaporization, melt and blow, etc.

- Vaporization:
- Laser heats surface to vaporization
- Forms keyhole
- Now light highly absorbed in hole (light reflects until absorbed)
- Vapor from boiling stabilizes molten walls
- Material ejected from hole can form Dross at bottom and top
- In materials that do not melt, just vapor escapes

e.g. Wood, carbon, some plastics

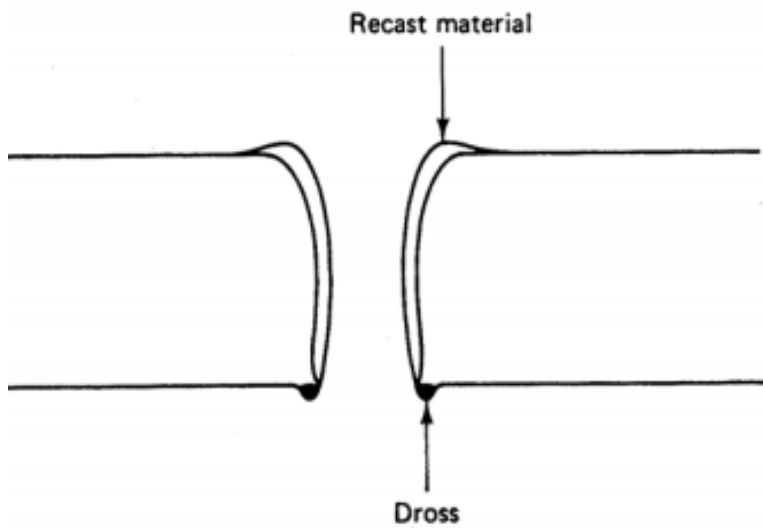
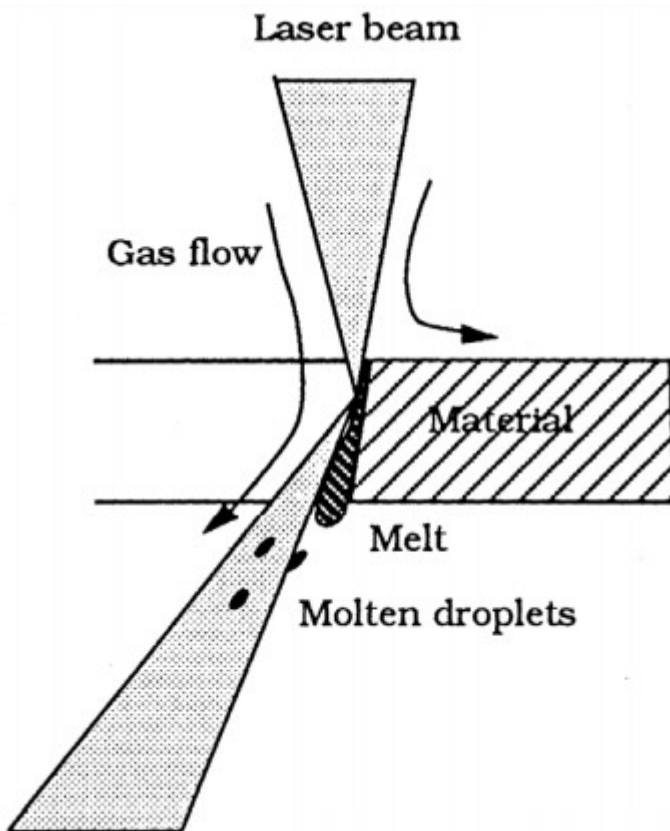


Figure 13-21 Sketch of laser-drilled hole.

Melt and Blow

Once melt is formed use gas flow to blow away materials
Do not need to vaporize, thus power reduced by factor of about 10

e.g. Metals.



47 Plasma Waves Kinetic Theory

47.1 Governing Equations, Vlasov-Boltzmann equation

Intense laser radiation interacting with matter at relativistic intensities ($a > 1$) is capable of generating plasma far away from equilibrium. A promising approach to describe the nonlinear, kinetic nature of the interaction are plasma models based on the fully relativistic Vlasov-Boltzmann equations combined with Maxwell's equations.

So, the governing equations of intense laser plasma interaction, are the Vlasov-Boltzmann equations combined with Maxwell's equations in three spatial and momentum dimensions.

The Vlasov equation is a differential equation describing time evolution of the distribution function of plasma consisting of charged particles with long-range interaction.

We consider a plasma consisting of electrons and ions, which are represented by distribution functions f_k (\vec{x}, \vec{p}, t). The distribution functions f_k give the probability of finding particles of sort k in a given volume of phase space. We assume that the electrons and ions in the plasma under consideration interact via electromagnetic radiation and binary collisions. Hence, an appropriate description of the plasma is based on the following set of transport equations, which for brevity are stated in covariant form.

47.1.1 Transport equations, Maxwell Equations

$$p_k^\mu \frac{\partial f_k}{\partial x^\mu} + m_k F_k^\mu \frac{\partial f_k}{\partial p_k^\mu} = \sum_{l=n,e,i} \int \frac{d^3 p_l}{p_l^0} F_{kl} \int d\Omega_\psi \sigma^{kl}(s, \psi) (f_k' f_l' - f_k f_l)$$

The quantity $d\Omega_\psi$ is an element of solid angle between \vec{p}'_k and \vec{p}_k and $\sigma^{kl}(s, \psi)$ denotes the invariant cross section. The relative velocity between particles k and l is given by

$$v_{kl} = \frac{c F_{kl}}{p_k^0 p_l^0} = \sqrt{(\vec{v}_k - \vec{v}_l)^2 - \frac{1}{c^2} (\vec{v}_k \times \vec{v}_l)^2}.$$

The force on the charged particles in the plasma is the Lorentz force given by

$$F_k^\mu = \frac{q_k}{m_k c} F^{\mu\nu} p_{k\nu}.$$

Maxwell's equations are represented as:

$$\frac{\partial}{\partial x^\mu} F^{\mu\nu} = \frac{j^\nu}{c\epsilon_0}, \quad \frac{\partial}{\partial x^\mu} \tilde{F}^{\mu\nu} = 0$$

$$j^\nu = \sum_{k=n,e,i} q_k \int d^4 p \, 2\Theta(p_0) \delta(p^2 - m_k^2 c^2) c p^\nu f_k$$

Equivalently, Eqn. (3.1) can be rewritten in three vector notation

$$\begin{aligned} & \left(\partial_t + \vec{v}_k \partial_{\vec{x}} + q_k \left[\vec{E} + \vec{v}_k \times \vec{B} \right] \partial_{\vec{p}_k} \right) f_k \\ & = \sum_{l=n,e,i} \int d^3 p_l v_{kl} \int d\Omega_\psi \sigma^{kl}(s, \psi) \left(f'_k f'_l - f_k f_l \right). \end{aligned}$$

Maxwell equations become

$$\begin{aligned} \partial_t \vec{E} &= c^2 \vec{\nabla} \times \vec{B} - \vec{j} / \epsilon_0, \\ \partial_t \vec{B} &= -\vec{\nabla} \times \vec{E}, \\ \partial_t \rho &= -\vec{\nabla} \cdot \vec{j}. \end{aligned}$$

The charge and current densities in three notations are given by

$$\begin{aligned} \rho &= q_e \int d^3 p_e f_e + q_i \int d^3 p_i f_i, \\ \vec{j} &= q_e \int d^3 p_e \vec{v}_e f_e + q_i \int d^3 p_i \vec{v}_i f_i. \end{aligned}$$

47.1.1.1 The Boltzmann collision operator

$$\begin{aligned} C_{kl} &= \int \frac{d^3 p_l}{p_l^0} \int \frac{d^3 p'_k}{p_k^0} \int \frac{d^3 p'_l}{p_l^0} \left(W_{klk'l'} f'_k f'_l - W_{k'l'kl} f_k f_l \right) \\ W_{k'l'kl} &= \frac{e_k^2 e_l^2 m_k^2 m_l^2 c^2}{4\pi^2 \epsilon_0^2} |M_{k'l'kl}|^2 \delta^4 \left(p'_k + p'_l - p_k - p_l \right) \\ &= s \sigma^{kl}(s, \psi) \delta^4 \left(p'_k + p'_l - p_k - p_l \right) \end{aligned}$$

with the invariant cross section

$$\sigma^{kl}(s, \psi) = \frac{e_k^2 e_l^2 m_k^2 m_l^2 c^2}{4\pi^2 \epsilon_0^2 s} |M_{k'l'kl}|^2$$

where $s = p_t^2$ and $p_t = p_k + p_l$. The quantities p_k and p_l are the pre-collisional momenta whereas p'_k and p'_l denotes the post-collisional momenta. The binary transition matrix elements are denoted by $|M_{k'l'kl}|$.

Integrating over \vec{p}'_l we obtain $\vec{p}'_t = \vec{p}_t - \vec{p}'_k$. Since the final momentum p'_l has to be on the mass shell we find $(p_t - p'_k)^2 = m_l^2 c^2$ from which we obtain $s + (m_k^2 - m_l^2) c^2 = 2 p'_k \cdot p_t$. The quantities m_k and m_l are the pre-collision rest masses of the colliding particles. We find

$$\begin{aligned} \delta \left(p_k^0 + p_l^0 - p_k^0 - p_l^0 \right) &= \frac{p_k^0 p_l^0}{p_t^0 |\vec{p}'_k| - |\vec{p}_t| p_k^0 \cos \psi} \delta \left(|\vec{p}'_k| - \mathcal{F}_{kl} \right) \quad \text{where} \\ \mathcal{F}_{kl} &= \frac{\mathcal{I} |\vec{p}_t| \cos \psi}{2 (s + \vec{p}_t^2 \sin^2 \psi)} + \sqrt{\left(\frac{\mathcal{I} |\vec{p}_t| \cos \psi}{2 (s + \vec{p}_t^2 \sin^2 \psi)} \right)^2 + \frac{\mathcal{I}^2 - 4 m_k^2 c^2 p_t^{02}}{4 (s + \vec{p}_t^2 \sin^2 \psi)}} \end{aligned}$$

with $\mathcal{I} = s + (m_k^2 - m_l^2) c^2$. The angle ψ denotes the angle between \vec{p}_t and \vec{p}'_k . Making use of Equation (3.14) we obtain

$$C_{kl} = \frac{e_k^2 e_l^2 m_k^2 m_l^2 c^2}{4\pi^2 \epsilon_0^2} \int \frac{d^3 p_l}{p_l^0} \int d\Omega_\psi \frac{|\vec{p}_k'|^2}{p_l^0 |\vec{p}_k'| - |\vec{p}_t| p_k^0 \cos \psi} \times |M_{k'l'kl}|^2 (f_k' f_l' - f_k f_l),$$

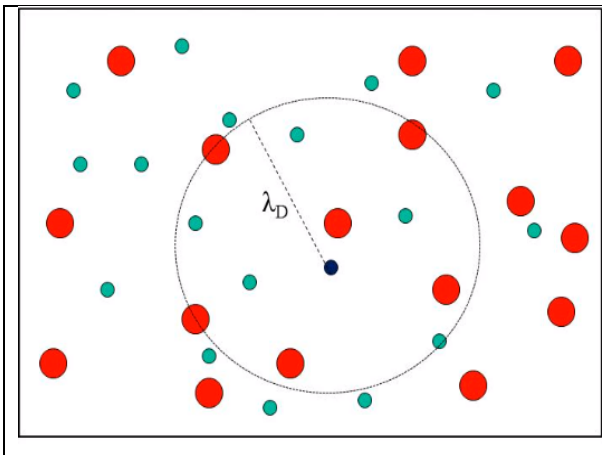
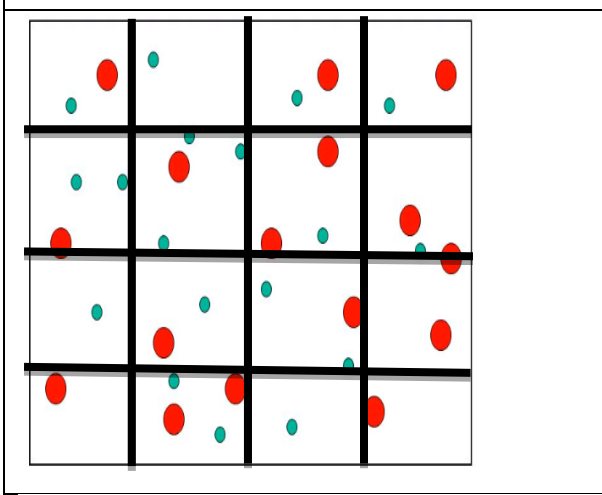
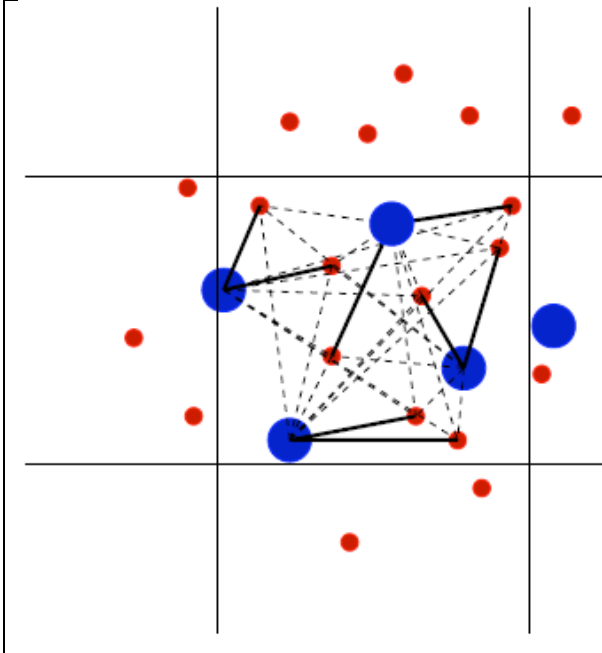
where $|\vec{p}_k'| = F_{kl}$ holds. The quantity $d\Omega_\psi$ denotes an element of solid angle between \vec{p}_t and \vec{p}_k' . As an example the transition matrix elements for elastic binary collisions are given

$$\begin{aligned} |M_{k'l'kl}|^2 &= \frac{1}{4} \sum_{s_k' s_l' s_k s_l} \left| \bar{u}_k' \gamma_\mu u_k \frac{1}{t + i\epsilon} \bar{u}_l' \gamma^\mu u_l \right|^2 \\ &= \frac{(p_l' \cdot p_k')(p_l \cdot p_k) + (p_l' \cdot p_k)(p_l \cdot p_k')}{2m_k^2 m_l^2 c^4 t^2} \\ &\quad - \frac{m_k^2 c^2 (p_l' \cdot p_l) + m_l^2 c^2 (p_k' \cdot p_k) - 2m_k^2 m_l^2 c^4}{2m_k^2 m_l^2 c^4 t^2}, \end{aligned}$$

where $t = (p_k - p_k')^2$ holds. The pre- and post-collision masses are the same. To perform the angle integration we introduce a coordinate system whose polar axis is parallel to \vec{p}_t .

48 Numerical model

48.1 Meshing

 A diagram showing a central blue particle surrounded by a dashed circle representing the Debye length, labeled λ_D . The sphere contains several other particles, some red and some teal, illustrating the concept of a Debye sphere in a plasma.	<p>Each particle in the plasma has the probability to collide with other particles in a considered volume.</p>
 A diagram showing a 4x4 grid of cells. Each cell contains a few particles, some red and some teal, illustrating the discretization of the plasma into a cubic grid.	<p>The grid considered in this program is cubic.</p>
 A diagram showing a 3x3 grid of cells. The central cell contains a mesh of blue particles connected by solid lines, with dashed lines extending to other particles in the surrounding cells, illustrating the application of equations to each cell.	<p>The equations are applied to each cell.</p>

49 Some public codes / research groups

<http://ptsg.egr.msu.edu>

Public codes

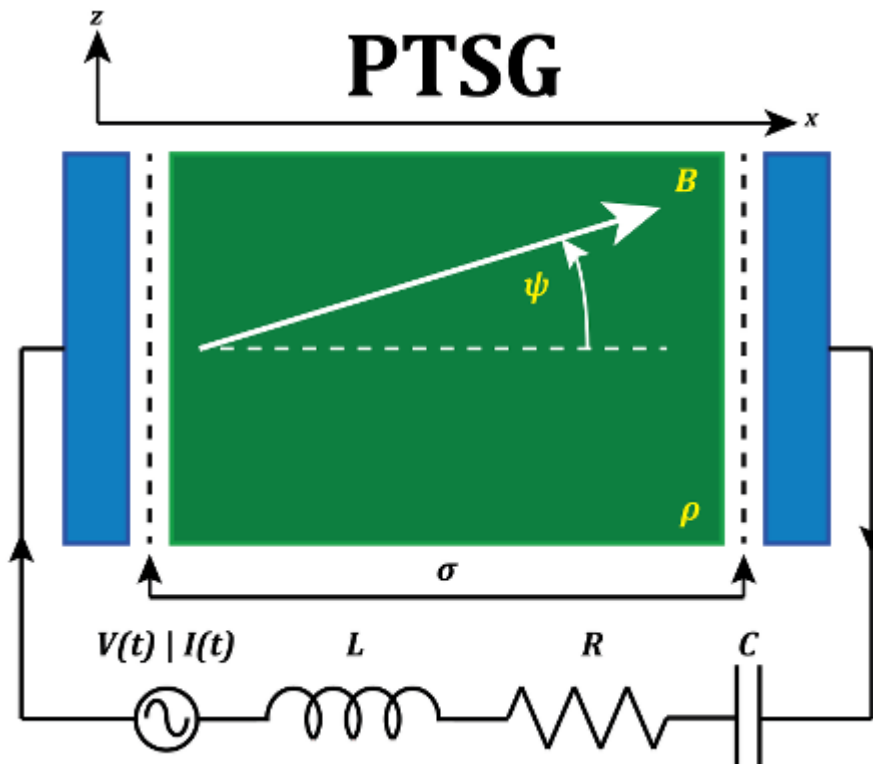
<http://ptsg.egr.msu.edu/>

Download the software now.

- [XES1](#)
- [XPDP1](#)
- [XPDC1](#)
- [XPDS1](#)
- [XPDP2](#)
- [XOOPIC](#)
- [XIBC](#)
- [XGRAFIX](#)

Our most recent, popular and well kept up codes are on bounded plasma, plasma device codes XPDP1, XPDC1, XPDS1, and XPDP2. The P, C, and S mean planar, cylindrical, or spherical bounding electrodes; the 1 means 1d 3v and the 2 means 2d 3v. These are electrostatic, may have an applied magnetic field, use many particles (like hundreds to millions), particle-in-cell (PIC), and allow for collisions between the charged particles (electrons and ions, + or -) and the background neutrals (PCC-MCC). The electrodes are connected by an external series R, L, C, source circuit, solved by Kirchhoff's laws simultaneously with the internal plasma solution (Poisson's equation). The source may be $V(t)$ or $I(t)$, may include a ramp-up (in time). XPDP2 is planar in x , periodic in y or fully bounded in (x,y) , driven by one or two sources. (For detailed information, [click here](#))

49.1 The Plasma Theory and Simulation Group PTSG Univ. of California, Berkley



49.1.1 People of PTSG

1 Professor, 2 Postdoctoral Researchers, 2 Graduate Students

49.1.2 Projects

49.1.2.1 Current Projects in PTSG

AFOSR: Consortium on Cathodes and Breakdown for High Power Microwave Devices

AFOSR/Calabazas Creek Research: High Voltage Computer Laboratory

DOE/LLNL: Hydrocarbon Transport in the Diverter Sheath

DOE/Calabazas Creek Research: A Finite Element PIC Model

49.1.2.2 Recently Completed Projects in PTSG

DOE: Modeling and Simulation of Plasma Edge Behavior: Formation, Stability and Heating

ONR: Plasma Boundaries, Neutral & Non-neutral Plasma, Plasma Devices

ONR/NRL: Simulation of Low Frequency Noise in a Coupled Cavity Traveling Wave Tube

TECH-X: Object-Oriented PIC Code With Upgraded Physics And Platform Independent GUI

Hitachi (Japan): Modeling and Simulation of Plasma Display Panels (PDP's)

Sumitomo Metals (Japan): Electromagnetic Modeling and Simulation of Sumitomo Surface Wave Plasma (SWP) Device

LLNL (Livermore): Plasma bulk (fluids), plasma sheath (particles); bounding wall (profiler); an attempt to construct seamless boundaries at the fluid /particle/ wall boundaries, and to run at fluid code speeds

49.1.3 PTSG Software

- **General Information**

Our practice has been to make all software developed by PTSG freely available to anyone. If you use our codes or our graphics (both are copyrighted), then please acknowledge PTSG in your publications and send us a copy of your journal articles or reports (send to Prof. John P. Verboncoeur). We would also appreciate receiving copies of your input files (representative) and a copy of code changes which you have made.

- **Acknowledgments**

For our plasma device codes, **XPDP1**, **XPDP2**, and **XPDS1**, please acknowledge:

Verboncoeur, J.P., M.V. Alves, V. Vahedi, and C.K. Birdsall, "Simultaneous Potential and Circuit Solution for 1d bounded Plasma Particle Simulation Codes," J. Comp. Physics, 104, pp. 321-328, February 1993.

For **XPDP2**, please acknowledge:

Vahedi, V., C.K. Birdsall, M.A. Lieberman, G. DiPeso, and T.D. Rognlien, "Verification of frequency scaling laws for capacitive radio-frequency discharges using two-dimensional simulations," Phys. Fluids B 5 (7), pp. 2719-2729, July 1993.

V. Vahedi and G. DiPeso, "Simultaneous Potential and Circuit Solution for Two-Dimensional Bounded Plasma Simulation Codes," J. Comp. Phys. 131, pp. 149-163, 1997. (Work begun at U.C. Berkeley)

For **XOOPIC** please acknowledge:

J.P. Verboncoeur, A.B. Langdon and N.T. Gladd, "An Object-Oriented Electromagnetic PIC Code", Comp. Phys. Comm., 87, May11, 1995, pp. 199-211.

- **Description**

Our most recent, popular and well kept up codes are on bounded plasma, plasma device codes **XPDP1**, **XPDC1**, **XPDS1**, and **XPDP2**. The P, C, and S mean planar, cylindrical, or spherical bounding electrodes; the 1 means 1d 3v and the 2 means 2d 3v. These are electrostatic, may have an applied magnetic field, use many particles (like hundreds to millions), particle-in-cell (PIC), and allow for collisions between the charged particles (electrons and ions, + or -) and the background neutrals (PCC-MCC). The electrodes are connected by an external series R, L, C, source circuit, solved by Kirchhoff's laws simultaneously with the internal plasma solution (Poisson's equation), The source may be V(t) or I(t), may include a ramp-up (in time). XPDP2 is planar in x, periodic in y or fully bounded in (x,y), driven by one or two sources (for detailed information, [click here](#)). Our older, far less well kept up codes are: **XES1 and XIBC**

- **Distribution**

All PTSG software is distributed as source codes in form of tarball file. To run the code, one have to have an access to computer with development environment (C/C++ compilers, header files) to be able to compile the code in their system. Source code are available for free download from our [website \(ptsg.egr.msu.edu\)](http://ptsg.egr.msu.edu). If you would require older version of one of the codes, please contact [PTSG Support](#) for further info on how to acquire it.

Codes that require xgrafx	Codes with own version of xgrafx	Python/Matlab based codes
Following codes require system-wide installed xgrafx (usually in /usr/local/lib). Before compiling codes, please download, compile and install xgrafx . xoopsic oopd1 xpdc1	Following codes do not work with current version of xgrafx and have version of xgrafx packed into tarball. xem1 xes1 xibc	Following codes require working Python environment (python 2.7/iython) or MatLab. pypd1 : Python based code, based on oopd1 (library version of oopd1 is already included in distribution)

Some public codes / research groups

xpdc2 xpdp1 xpdp2 xpds1		s_parnos : MatLab code (requires MatLab)
--	--	---

Code Consulting and Maintenance

PTSG may answer short inquiries on our various codes, as we are primarily committed to university type plasma device research and development. However, more detailed inquiries for help should be addressed to Research Institute for Science and Engineering, a small private firm well skilled in PTSG codes: rise@ieee.org.

If you find bugs while using our program, please report them to us. We appreciate these information. Click here for [the bug report form](#).

49.1.4 Publications Using PTSG Codes, Worldwide

Many authors, using our codes, have made acknowledgments to PTSG and sent us copies of their reports, journal articles, letters, and MS, Ph.D. theses. We appreciate both.

Since 1991, we have listed the publications brought to our attention using our codes XPDP1, XPDC1, XPDS1, XPDP2, and XOOPIE (by us and others). The total that we are aware of now exceeds 300, including over 60 by us. We are delighted to have been helpful to these authors. We are even more delighted to learn of their excellent and ingenious applications and modifications of our codes (go to the [List of the publications](#)).

49.1.5 Workshops

49.1.5.1 Plasma Device Workshop 2009 (PDW2009)

We have given workshops for users of our plasma device codes. These codes are particle-in-cell (PIC) codes with Monte Carlo collision (MCC) models in 1D and 2D. Cartesian, cylindrical, and spherical coordinates are modeled, in both electrostatic and electromagnetic regimes. The models include flexible boundary conditions, external driving circuits, and user-customizable diagnostics. The PTSG has been at the forefront of algorithm development for almost 50 years, and continues to develop new models for physics, computer science, and applied mathematics.

Inquiries are welcome (address to [Prof. John P. Verboncoeur](#)).

Copyright © 2012-2016 Plasma Theory and Simulation Group

Electrical Computer Engineering Department

Michigan State University

Copyright © 2008 Plasma Theory and Simulation Group, Nuclear Engineering Department, University of California, Berkeley

50 Comparison between actual IAP-PCS and XPDP2

 xgrafix.tar.gz	 xpdp2.tar.gz	 IAP-PSC_220819_src+exe+output.zip
---	---	--

50.1 Description of XPDP2

For **XPDP2**, please acknowledge:

[1] Vahedi, V., C.K. Birdsall, M.A. Lieberman, G. DiPeso, and T.D. Rognlien, "Verification of frequency scaling laws for capacitive radio-frequency discharges using two-dimensional simulations," *Phys. Fluids B* 5 (7), pp. 2719-2729, July 1993.

[2] V. Vahedi and G. DiPeso, "Simultaneous Potential and Circuit Solution for Two-Dimensional Bounded Plasma Simulation Codes," *J. Comp. Phys.* 131, pp. 149-163, 1997. (Work begun at U.C. Berkeley)

50.1.1 Abstract of [1]:

Weakly ionized processing plasmas are studied in two dimensions using a bounded particle-in-cell (PIC) simulation code with a Monte Carlo collision (MCC) package. The MCC package models the collisions between charged and neutral particles, which are needed to obtain a self-sustained plasma and the proper electron and ion energy loss mechanisms. A two-dimensional capacitive radio-frequency (rf) discharge is investigated in detail. Simple frequency scaling laws for predicting the behavior of some plasma parameters are derived and then compared with simulation results, finding good agreements. It is found that as the drive frequency increases, the sheath width decreases, and the bulk plasma becomes more uniform, leading to a reduction of the ion angular spread at the target and an improvement of ion dose uniformity at the driven electrode.

50.1.2 Abstract of [2]:
















An algorithm for coupling external circuit elements to be bounded two-dimensional electrostatic plasma simulation codes is developed. In general, the external circuit equations provide a mixture of Dirichlet and Neumann boundary conditions for the Poisson equation, which is solved each time step for the internal plasma potential. We rewrite the coupling between the plasma and the external circuit parameters as an algebraic or ordinary differential equation for the potential on the boundary. This scheme allows decomposition of the field solve into a Laplace solver with boundary conditions (e.g., applied potentials) and a Poisson solver with zero boundary conditions. We present the details of the external circuit coupling to an explicit electrostatic planar two-dimensional particle-in-cell code called PDP2, and discuss briefly how the coupling can be done in an implicit electrostatic code. The decomposition replaces the iterative coupling with a direct coupling and reduces the amount of computational time spent in the field solver. We use PDP2 to simulate a dually excited capacitively coupled RF discharge and show how such a system can be used as a plasma processing tool with separate control over ion flux and ion bombarding energy.

50.1.3 xpdp2 Code Description

inp	19.08.2019 21:33	Dateiordner	
argonmcc.c	13.09.2016 23:55	C-Datei	16 KB
boundary.c	13.09.2016 23:55	C-Datei	14 KB
dadi.c	13.09.2016 23:55	C-Datei	20 KB
def.h	13.09.2016 23:55	H-Datei	2 KB
fft.c	13.09.2016 23:55	C-Datei	4 KB
field.c	13.09.2016 23:55	C-Datei	16 KB
field2.c	13.09.2016 23:55	C-Datei	9 KB
gather.c	13.09.2016 23:55	C-Datei	7 KB
history.c	13.09.2016 23:55	C-Datei	12 KB
initwin.c	13.09.2016 23:55	C-Datei	31 KB
input_file_doc	13.09.2016 23:55	Datei	9 KB
load.c	13.09.2016 23:55	C-Datei	4 KB
makefile	13.09.2016 23:55	Datei	2 KB
maxwellv.c	13.09.2016 23:55	C-Datei	3 KB
move.c	13.09.2016 23:55	C-Datei	5 KB
pdp2.c	13.09.2016 23:55	C-Datei	3 KB
pdp2.h	13.09.2016 23:55	H-Datei	5 KB
README_V2.3	13.09.2016 23:55	3-Datei	4 KB
start.c	13.09.2016 23:55	C-Datei	30 KB
uraniummcc.c	13.09.2016 23:55	C-Datei	12 KB

50.2 Actual IAP-PSC code

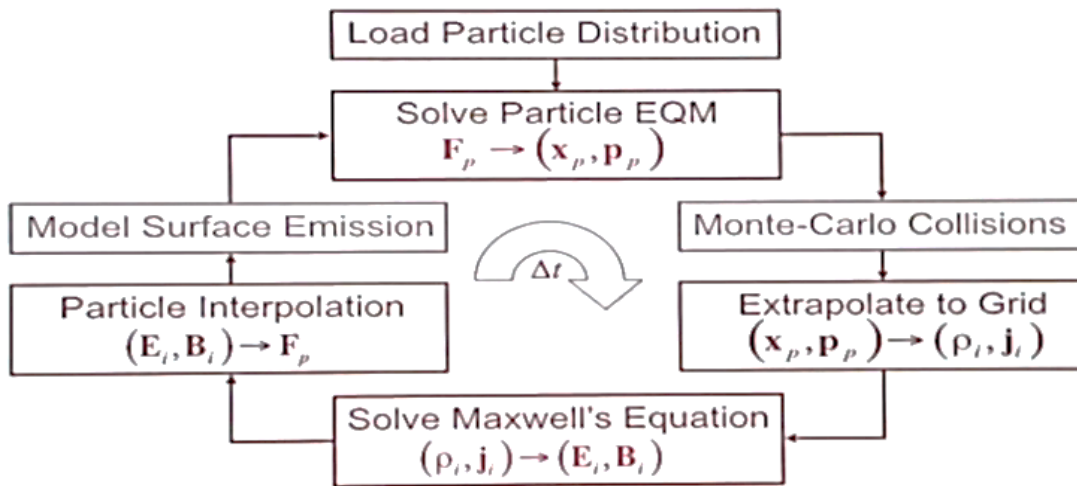
IAP-PSC program is composed from:

-  PSC_PIC
 -  PSC_PIC.pro
 -  Headers
 -  dens_currnt.h
 -  lorentz.h
 -  maxwell_equat.h
 -  parameter.h
 -  pos_veloct.h
 -  Sources
 -  dens_currnt.cpp
 -  lorentz.cpp
 -  main.cpp
 -  maxwell_equat.cpp
 -  parameter.cpp
 -  pos_veloct.cpp

The mesh is a cube divided into 5 parts on each side.

It contains 1000 particles, 500 electrons and 500 ions.

The sequence followed is:



These equations are applied:

- Lorentz-Force: $\mathbf{F}_p = q\mathbf{E}_p + \frac{q}{m}(\mathbf{p}_p \times \mathbf{B}_p)$

Position and velocity:

$$\frac{d\vec{v}_j}{dt} = \frac{q_j}{m_j} \left(\vec{E} + \frac{\vec{v}_j \times \vec{B}}{c} \right)$$

$$\frac{d\vec{x}}{dt} = \vec{v}$$

Density and current:

$$\rho(\vec{x}) = \sum_j q_j \delta(\vec{x} - \vec{x}_j)$$

$$\vec{j}(\vec{x}) = \sum_j q_j \vec{v}_j \delta(\vec{x} - \vec{x}_j)$$

Maxwell equations to calculate E and B

$$\nabla \times \vec{E} = -\frac{1}{c} \frac{\partial \vec{B}}{\partial t}$$

$$\nabla \cdot \vec{B} = 0, \quad \nabla \cdot \vec{E} = 4\pi\rho$$

$$\nabla \times \vec{B} = \frac{4\pi\vec{j}}{c} + \frac{1}{c} \frac{\partial \vec{E}}{\partial t} \quad \underline{15}$$

Numerical curl – Advancing B

- The advance of B from step (n-1/2) to (n+1/2) is done via central differences using E at step n
- The x-component of curl-E is:

$$\frac{\partial \vec{E}_z / \partial y - \partial \vec{E}_y / \partial z}{\Delta y} = -\frac{\partial \vec{B}_x / \partial t}{\Delta t}$$

$$\frac{\frac{\vec{E}_{z,n}^{i,j+1,k} - \vec{E}_{z,n}^{i,j,k}}{\Delta y} - \frac{\vec{E}_{y,n}^{i,j,k+1} - \vec{E}_{y,n}^{i,j,k}}{\Delta z}}{\Delta y} = -\frac{1}{c} \frac{\vec{B}_{x,n+1/2}^{i,j,k} - \vec{B}_{x,n-1/2}^{i,j,k}}{\Delta t}$$

Numerical curl – Advancing E

- The advance of E from step (n) to (n+1) is done similarly using B at step (n+1/2).
- The xz-component of curl-B is:

$$\frac{\partial \vec{B}_z / \partial y - \partial \vec{B}_y / \partial z}{\Delta y} = \frac{4\pi}{c} \vec{j}_x + \frac{1}{c} \frac{\partial \vec{E}_x / \partial t}{\Delta t}$$

$$\frac{\frac{\vec{B}_{z,n+1/2}^{i,j,k} - \vec{B}_{z,n+1/2}^{i,j-1,k}}{\Delta y} - \frac{\vec{B}_{y,n+1/2}^{i,j,k} - \vec{B}_{y,n+1/2}^{i,j,k-1}}{\Delta z}}{\Delta y} = \frac{4\pi}{c} \vec{j}_{x,n+1/2}^{i,j,k} + \frac{1}{c} \frac{\vec{E}_{x,n+1}^{i,j,k} - \vec{E}_{x,n}^{i,j,k}}{\Delta t}$$

50.3 Comparing information content of files of IAP-PSC and XPDP2

¹⁵ <https://www.youtube.com/watch?v=I09OeVDoEZY>

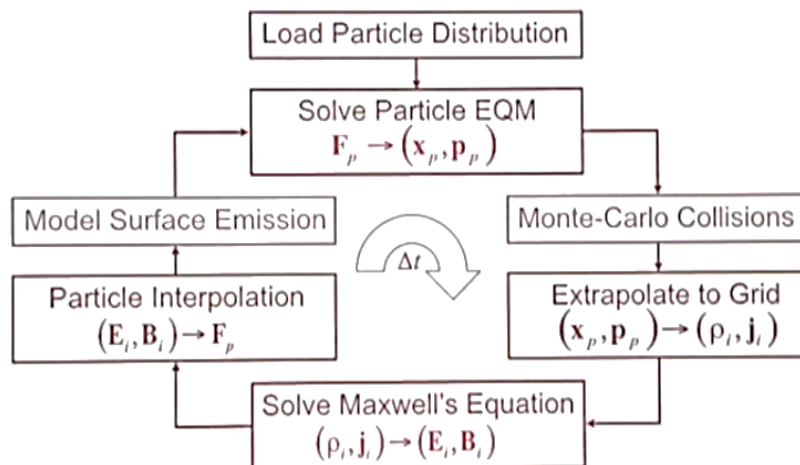
XPDP2 file	Description	IAP-PSC file	Description
inp/argon.inp	argon.inp: Argon RF discharge -nsp---ncx---ncy---nc2p---dt[s]---xlength[m]---ylength[m]---zlength[m]---epsilon- 2 80 160 1e6 3.600894e-11 0.03 0.06 0.1 1.0 ELECTRONS (Species 1) ---q[C]-----m[Kg]-----k-----max-np---rel_weight-- -1.602e-19 9.11e-31 1 200000 1	Main.cpp	Call all the functions , advance their values to the next time step
inp/maxwell.inp			
maxwellv.c	void maxwellv(SCALAR *vx, SCALAR *vy, SCALAR *vz, SCALAR vth, SCALAR Rv, SCALAR Rphi, SCALAR Rthe)	maxwell_equat.h maxwell_equat.cpp	$\nabla \times \vec{E} = -\frac{1}{c} \frac{\partial \vec{B}}{\partial t}$ $\nabla \cdot \vec{B} = 0, \quad \nabla \cdot \vec{E} = 4\pi\rho$ $\nabla \times \vec{B} = \frac{4\pi\vec{j}}{c} + \frac{1}{c} \frac{\partial \vec{E}}{\partial t}$
Move.c		Pos_veloct.h Pos_veloct.c	$\frac{d\vec{v}_j}{dt} = \frac{q_j}{m_j} (\vec{E} + \frac{\vec{v}_j \times \vec{B}}{c})$ $\frac{d\vec{x}}{dt} = \vec{v}$
		Lorentz.h, Lorentz.c	$\mathbf{F}_p = q\mathbf{E}_p + \frac{q}{m} (\mathbf{p}_p \times \mathbf{B}_p)$
		Parameter.h, Parameter.c	Initial and boundary conditions
		Dens_currnt.h Dens_currnt.c	$\rho(\vec{x}) = \sum_j q_j \delta(\vec{x} - \vec{x}_j)$ $\vec{j}(\vec{x}) = \sum_j q_j \vec{v}_j \delta(\vec{x} - \vec{x}_j)$

51 IAP-PSC Program

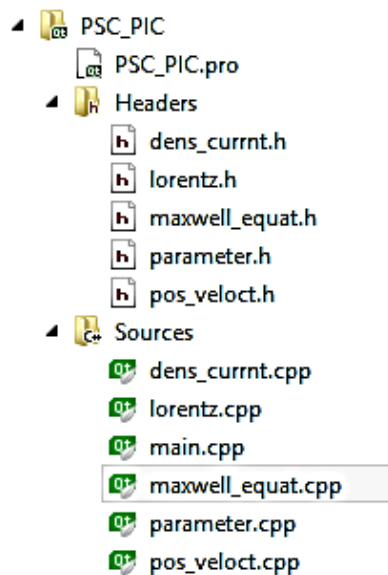
51.1 Description

Our program is written with C++ on Qt creator and the results were presented on Paraview.

The program study the compartment of a plasma composed of 1000 particles (500 electrons and 500 ions) placed randomly in a box under the reaction of electric and magnetic field. The program follows the chart below:



It is composed from 5 classes and the main program.



After initializing the parameter and the initial condition using the class << **parameter.cpp** >>, the Lorentz force on each particle is calculated using the class << **Lorentz.cpp** >> then the new position and the velocity is calculated using the class << **pos_veloct.cpp** >>. The density and the current on the grid are calculated using the class << **dens_currnt.cpp** >>. Using these values and the Maxwell equations, the electric and magnetic field are calculated on each cell using the class << **Maxwell_equat.cpp** >>. Now the Lorentz force is recalculated and a new time step starts.

Our program repeats this chain for 10 time steps.

51.2 The Code

51.2.1 Class 1: parameters

Parameter.h

```

#ifndef PARAMETER_H
#define PARAMETER_H

class parameter
{
public:
    parameter() {

        void init_elect_param(double xe[500][10],double ye[500][10],double
ze[500][10],double vex[500][10],double vey[500][10],double vez[500][10],double
Fex[500][10],double Fey[500][10],double Fez[500][10],double Eex[500][10],double
Eey[500][10],double Eez[500][10],double Bex[500][10],double Bey[500][10],double
Bez[500][10]);

        void init_ion_param(double xi[500][10],double yi[500][10],double zi[500][10],double
vix[500][10],double viy[500][10],double viz[500][10],double Fix[500][10],double
Fiy[500][10],double Fiz[500][10],double Eix[500][10],double Eiy[500][10],double
Eiz[500][10],double Bix[500][10],double Biy[500][10],double Biz[500][10]);

        void init_EB(double Ex[5][5][5][10],double Ey[5][5][5][10],double
Ez[5][5][5][10],double Bx[5][5][5][10],double By[5][5][5][10],double
Bz[5][5][5][10],double rhox[5][5][5][10],double rhox[5][5][5][10],double
rhoz[5][5][5][10],double Jx[5][5][5][10],double Jy[5][5][5][10],double
Jz[5][5][5][10]);

};

#endif // PARAMETER_H

```

Parameter.cpp

```

#include "parameter.h"
#include <cstdlib>
void parameter::init_elect_param(double xe[500][10],double ye[500][10],double
ze[500][10],double vex[500][10],double vey[500][10],double vez[500][10],double
Fex[500][10],double Fey[500][10],double Fez[500][10],double Eex[500][10],double
Eey[500][10],double Eez[500][10],double Bex[500][10],double Bey[500][10],double
Bez[500][10]) {

int a=0, b=0, c=0;
    for(int l=0;l<500;l++){
        a=rand()%1000;
        b=rand()%1000;
        c=rand()%1000;

        xe[l][0]=a;

        ye[l][0]=b;

        ze[l][0]=c;

    }

    for(int i=0;i<500;i++){

```

```

    vex[i][0]=2.42;//m/s
    vey[i][0]=2.42;
    vez[i][0]=2.42;
    Fex[i][0]=0;
    Fey[i][0]=0;
    Fez[i][0]=0;
    Eex[i][0]=2000;
    Eey[i][0]=2000;
    Eez[i][0]=2000;
    Bex[i][0]=47*10e-6;
    Bey[i][0]=47*10e-6;
    Bez[i][0]=47*10e-6;
}
}
void parameter::init_ion_param(double xi[500][10],double yi[500][10],double
zi[500][10],double vix[500][10],double viy[500][10],double viz[500][10],double
Fix[500][10],double Fiy[500][10],double Fiz[500][10],double Eix[500][10],double
Eiy[500][10],double Eiz[500][10],double Bix[500][10],double Biy[500][10],double
Biz[500][10]){
    int a=0, b=0, c=0;
        for(int l=0;l<500;l++){
            a=rand()%1000;
            b=rand()%1000;
            c=rand()%1000;

            xi[l][0]=a;
            yi[l][0]=b;
            zi[l][0]=c;

        }

        for(int i=0;i<500;i++){
            vix[i][0]=1;
            viy[i][0]=1;
            viz[i][0]=1;
            Fix[i][0]=0;
            Fiy[i][0]=0;
            Fiz[i][0]=0;
            Eix[i][0]=2000;
            Eiy[i][0]=2000;
            Eiz[i][0]=2000;
            Bix[i][0]=47*10e-6;
            Biy[i][0]=47*10e-6;
            Biz[i][0]=47*10e-6;
        }
}
}
void parameter::init_EB(double Ex[5][5][5][10],double Ey[5][5][5][10],double
Ez[5][5][5][10],double Bx[5][5][5][10],double By[5][5][5][10],double
Bz[5][5][5][10],double rhox[5][5][5][10],double rhoy[5][5][5][10],double
rhoz[5][5][5][10],double Jx[5][5][5][10],double Jy[5][5][5][10],double
Jz[5][5][5][10]){
    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
            for(int k=0;k<5;k++){
                Ex[i][0][0][0]=10e5;
                Ey[0][j][0][0]=10e5;
                Ez[0][0][k][0]=10e5;
                Bx[0][j][k][0]=1*10e-6;

```

```
    By[i][0][k][0]=1*10e-6;
    Bz[i][j][0][0]=1*10e-6;
    rhox[i][j][k][0]=0;
    rhoy[i][j][k][0]=0;
    rhoz[i][j][k][0]=0;
    Jx[i][j][k][0]=0;
    Jy[i][j][k][0]=0;
    Jz[i][j][k][0]=0;
  }
}
```

51.2.2 Class 2: Lorentz

Lorentz.h

```
#ifndef LORENTZ_H
#define LORENTZ_H

class lorentz
{
public:
    lorentz() {

    }

    double Force (double q,double Ep,double m,double p,double Bp);
};

#endif // LORENTZ_H
```

Lorentz.cpp

```
#include "lorentz.h"
#include "math.h"

double lorentz::Force (double q,double Ep,double m,double p,double Bp)
{
double F;
F= q*Ep+q/m*(p*Bp);

return F;
}
```

51.2.3 Class 3: pos_veloct

Pos_veloct.h

```
#ifndef POS_VELOCT_H
#define POS_VELOCT_H

class Pos_veloct
{
public:
    Pos_veloct() {

    }
}
```

```
    double velocity(double F,double m);  
};  
#endif // POS_VELOCT_H
```

Pos_veloct.cpp

```
#include "pos_veloct.h"  
  
double Pos_veloct::velocity(double F,double m){  
    double dvdt;  
    dvdt=F/m;  
    return dvdt;  
}
```

51.2.4 Class 4: Dens_current

Dens_current.h:

```
#ifndef DENS_CURRNT_H  
#define DENS_CURRNT_H  
  
class dens_currnt  
{  
public:  
    dens_currnt(){  
  
    }  
  
    double dens(double q,double i,double x,double delta);  
    double curr(double q,double i,double x,double delta, double v);  
};  
#endif// DENS_CURRNT_H
```

Dens_currnt.cpp

```
#include "dens_currnt.h"  
  
double dens_currnt::dens(double q,double i,double x,double delta){  
    double dens;  
    dens=q*delta*(i-x);  
    return dens;  
}  
double dens_currnt::curr(double q,double i,double x,double delta, double v){  
    double curr;  
    curr=q*v*delta*(i-x);  
    return curr;  
}
```

51.2.5 Class 5: Maxwell_equat

Maxwell_equat.h

```
#ifndef MAXWELL_EQUAT_H  
#define MAXWELL_EQUAT_H
```

```

class maxwell_equat
{
public:
    maxwell_equat() {

    }

    double electx(double ex,double c,int delta_y,int delta_z,int delta_t,double
Bz,double bz, double By,double by,double J,double PI );
    double electy(double ey,double c,int delta_x,int delta_z,int delta_t,double
Bz,double bz, double Bx,double bx,double J,double PI );
    double electz(double ez,double c,int delta_y,int delta_x,int delta_t,double
Bx,double bx, double By,double by,double J,double PI );

    double magntx( double bx, double c,double Ez,double ez,double Ey,double ey,int
delta_y,int delta_z,int delta_t);
    double magnty( double by, double c,double Ez,double ez,double Ex,double ex,int
delta_x,int delta_z,int delta_t);
    double magntz( double bz, double c,double Ex,double ex,double Ey,double ey,int
delta_x,int delta_y,int delta_t);

};

#endif // MAXWELL_EQUAT_H

```

Maxwell_equat.cpp

```

#include "maxwell_equat.h"
#include "math.h"

double maxwell_equat::electx(double ex,double c,int delta_y,int delta_z,int
delta_t,double Bz,double bz, double By,double by,double J,double PI ){
    double Ex;
    Ex=ex-delta_t*4*PI*J+(delta_t*c/delta_y)*(Bz-bz)-(delta_t*c/delta_z)*(By-by);
return Ex;
}

double maxwell_equat::electy(double ey,double c,int delta_x,int delta_z,int
delta_t,double Bz,double bz, double Bx,double bx,double J,double PI ){
double Ey;
Ey=ey-delta_t*4*PI*J+(delta_t*c/delta_x)*(Bz-bz)-(delta_t*c/delta_z)*(Bx-bx);
return Ey;
}

double maxwell_equat::electz(double ez,double c,int delta_y,int delta_x,int
delta_t,double Bx,double bx, double By,double by,double J,double PI ){
double Ez;
Ez=ez-delta_t*4*PI*J+(delta_t*c/delta_y)*(Bx-bx)-(delta_t*c/delta_x)*(By-by);
return Ez;
}

double maxwell_equat::magntx( double bx, double c,double Ez,double ez,double Ey,double
ey,int delta_y,int delta_z,int delta_t){
    // x component
    double Bx;
    Bx=bx-(c*delta_t)*((Ez-ez)/delta_y)+(c*delta_t)*((Ey-ey)/(delta_z));
return Bx;
}

double maxwell_equat::magnty( double by, double c,double Ez,double ez,double Ex,double
ex,int delta_x,int delta_z,int delta_t){
double By;
// y component
By=by-(c*delta_t)*((Ez-ez)/delta_x)+(c*delta_t)*((Ex-ex)/(delta_z));
return By;
}

```

```

double maxwell_equat::magntz( double bz, double c, double Ex, double ex, double Ey, double
ey, int delta_x, int delta_y, int delta_t) {
double Bz;
    // z component
    Bz=bz-(c*delta_t)*((Ex-ex)/delta_y)+(c*delta_t)*((Ey-ey)/(delta_x));
return Bz;
}

//      Bx[i][j][k][t+1]=Bx[i][j][k][t]-(c*delta_t)*((Ez[i][j+1][k][t]-
Ez[i][j][k][t])/delta_y)+(c*delta_t)*((Ey[i][j][k+1][t]-Ey[i][j][k][t])/(delta_z));
//      By[i][j][k][t+1]=By[i][j][k][t]-(c*delta_t)*((Ez[i+1][j][k][t]-
Ez[i][j][k][t])/delta_x)+(c*delta_t)*((Ex[i][j][k+1][t]-Ex[i][j][k][t])/(delta_z));
//      Bz[i][j][k][t+1]=Bz[i][j][k][t]-(c*delta_t)*((Ex[i][j+1][k][t]-
Ex[i][j][k][t])/delta_y)+(c*delta_t)*((Ey[i+1][j][k][t]-Ey[i][j][k][t])/(delta_x));
// Ex[i][j][k][t+1]=Ex[i][j][k][t]-
delta_t*4*PI*Jx[i][j][k][t]+(delta_t*c/delta_y)*(Bz[i][j][k][t+1]-Bz[i][j-1][k][t+1])-
(delta_t*c/delta_z)*(By[i][j][k][t+1]-By[i][j][k-1][t+1]);
// Ey[i][j][k][t+1]=Ey[i][j][k][t]-
delta_t*4*PI*Jy[i][j][k][t]+(delta_t*c/delta_x)*(Bz[i][j][k][t+1]-Bz[i-1][j][k][t+1])-
(delta_t*c/delta_z)*(Bx[i][j][k][t+1]-Bx[i][j][k-1][t+1]);
// Ez[i][j][k][t+1]=Ez[i][j][k][t]-
delta_t*4*PI*Jz[i][j][k][t]+(delta_t*c/delta_y)*(Bx[i][j][k][t+1]-Bx[i][j-1][k][t+1])-
(delta_t*c/delta_x)*(By[i][j][k][t+1]-By[i-1][j][k][t+1]);

```

51.2.6 Main program

Main.cpp

```

#include <QCoreApplication>
#include<iostream>
#include<fstream>
#include<string>
#include "parameter.h"
#include "lorentz.h"
#include "pos_veloct.h"
#include "dens_currnt.h"
#include "maxwell_equat.h"
#include "math.h"
#include "algorithm"
#include "iostream"
#include "fstream"
#include "sstream"
#include "stdio.h"
#include <QtCore/QString>
#include <QtCore/QFile>
#include <QtCore/QDebug>
#include <QtCore/QTextStream>
#include <cstdlib>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    double xe[500][10];
    double ye[500][10];
    double ze[500][10];
    double vex[500][10];
    double vey[500][10];
    double vez[500][10];
    double dvex[500][10];
    double dvey[500][10];
    double dvez[500][10];
    double Fex[500][10];

```

```
double Fey[500][10];
double Fez[500][10];
double xi[500][10];
double yi[500][10];
double zi[500][10];
double vix[500][10];
double viy[500][10];
double viz[500][10];
double dvix[500][10];
double dviy[500][10];
double dviz[500][10];
double Fix[500][10];
double Fiy[500][10];
double Fiz[500][10];
double mi=1.67*10e-27,me=9.11*10e-28,qi=1.602*10e-19,qe=-1.602*10e-19;
double pex[500][10];
double pey[500][10];
double pez[500][10];
double pix[500][10];
double piy[500][10];
double piz[500][10];
double Ex[5][5][5][10];
double Ey[5][5][5][10];
double Ez[5][5][5][10];
double Eix[500][10],Eex[500][10];
double Eiy[500][10],Eey[500][10];
double Eiz[500][10],Eez[500][10];
double Bix[500][10],Bex[500][10];
double Biy[500][10],Bey[500][10];
double Biz[500][10],Bez[500][10];
double Bx[5][5][5][10];
double By[5][5][5][10];
double Bz[5][5][5][10];
double rhox[5][5][5][10];
double rhoz[5][5][5][10];
double rhoy[5][5][5][10];
double Jx[5][5][5][10];
double Jy[5][5][5][10];
double Jz[5][5][5][10];
double delta=0;
double c=3*10e8,PI=3.14;
int delta_x=1, delta_y=1, delta_z=1, delta_t=1;

cout<<"The program has start\n";

// parameter initialization

parameter param;
param.init_elect_param(xe, ye, ze, vex, vey, vez, Fex, Fey, Fez, Eex, Eey, Eez, Bex, Bey, Bez);
param.init_ion_param( xi, yi, zi, vix, viy, viz, Fix, Fiy, Fiz, Eix, Eiy, Eiz, Bix, Biy, Biz);
param.init_EB( Ex, Ey, Ez, Bx, By, Bz, rhox, rhoy, rhoz, Jx, Jy, Jz);

//force de lorentz
for(int t=0;t<10;t++){
    lorentz lort;
    for(int i=0;i<500;i++){
        Fex[i][t]=lort.Force(qe,Eex[i][t],me,pex[i][t],Bex[i][t]);
        Fey[i][t]=lort.Force(qe,Eey[i][t],me,pey[i][t],Bey[i][t]);
        Fez[i][t]=lort.Force(qe,Eez[i][t],me,pez[i][t],Bez[i][t]);
        Fix[i][t]=lort.Force(qi,Eix[i][t],mi,pix[i][t],Bix[i][t]);
        Fiy[i][t]=lort.Force(qi,Eiy[i][t],mi,piy[i][t],Biy[i][t]);
        Fiz[i][t]=lort.Force(qi,Eiz[i][t],mi,piz[i][t],Biz[i][t]);
    }
}
```



```

//position and velocity
Pos_veloct vel;
for (int i=0;i<500;i++){
    dvex[i][t]=vel.velocity(Fex[i][t],me);
    dvey[i][t]=vel.velocity(Fey[i][t],me);
    dvez[i][t]=vel.velocity(Fez[i][t],me);
    dvix[i][t]=vel.velocity(Fix[i][t],mi);
    dviy[i][t]=vel.velocity(Fiy[i][t],mi);
    dviz[i][t]=vel.velocity(Fiz[i][t],mi);
}
for(int i=0;i<500;i++){
    vex[i][t+delta_t]= vex[i][t]+((2*delta_t) * dvex[i][t]);
    vey[i][t+delta_t]= vey[i][t]+((2*delta_t) * dvey[i][t]);
    vez[i][t+delta_t]= vez[i][t]+((2*delta_t) * dvez[i][t]);
    vix[i][t+delta_t]= vix[i][t]+((2*delta_t) * dvix[i][t]);
    viy[i][t+delta_t]= viy[i][t]+((2*delta_t) * dviy[i][t]);
    viz[i][t+delta_t]= viz[i][t]+((2*delta_t) * dviz[i][t]);
}
for(int i=0;i<500;i++){
    xe[i][t+delta_t]= xe[i][t]+((2*delta_t) * vex[i][t]);
    ye[i][t+delta_t]= ye[i][t]+((2*delta_t) * vey[i][t]);
    ze[i][t+delta_t]= ze[i][t]+((2*delta_t) * vez[i][t]);
    xi[i][t+delta_t]= xi[i][t]+((2*delta_t) * vix[i][t]);
    yi[i][t+delta_t]= yi[i][t]+((2*delta_t) * viy[i][t]);
    zi[i][t+delta_t]= zi[i][t]+((2*delta_t) * viz[i][t]);
}
//the particles should not go out the box(boundaries)
for(int i=0;i<500;i++){
    if(xe[i][t+delta_t]>1000 || xe[i][t+delta_t]<0){
        xe[i][t+delta_t]= rand()%1000;}
    if(ye[i][t+delta_t]>1000 || ye[i][t+delta_t]<0){
        ye[i][t+delta_t]= rand()%1000;}
    if(ze[i][t+delta_t]>1000 || ze[i][t+delta_t]<0){
        ze[i][t+delta_t]= rand()%1000;}
    if(xi[i][t+delta_t]>1000 || xi[i][t+delta_t]<0){
        xi[i][t+delta_t]= rand()%1000;}
    if(yi[i][t+delta_t]>1000 || yi[i][t+delta_t]<0){
        yi[i][t+delta_t]= rand()%1000;}
    if(zi[i][t+delta_t]>1000 || zi[i][t+delta_t]<0 ){
        zi[i][t+delta_t]= rand()%1000;}
}
// system("pause");
// density and current
dens_currnt dens;
double k1=0,k2=0,k3=0,k4=0,k5=0,k6=0;
double l1=0,l2=0,l3=0,l4=0,l5=0,l6=0;
for(int i=0;i<5;i++){
    for (int j=0;j<5;j++){
        for (int k=0;k<5;k++){
            for (int l=0;l<500;l++) {
                k1+=dens.dens(qe,i,xe[l][t],delta);
                k2+=dens.dens(qe,j,ye[l][t],delta);
                k3+=dens.dens(qe,k,ze[l][t],delta);
                k4+=dens.curr(qe,i,xe[l][t],delta,vex[l][t]);
                k5+=dens.curr(qe,j,ye[l][t],delta,vey[l][t]);
            }
        }
    }
}

```

```

        k6+=dens.curr(qe,k,ze[l][t],delta,vez[l][t]);
    //}
//if(xi[l][t+1]<i+1 && xi[l][t+1]>i && yi[l][t+1]<j+1 && yi[l][t+1]>j && zi[l][t+1]<k+1
&& zi[l][t+1]>k){
    l1+=dens.dens(qi,i,xi[l][t],delta);
    l2+=dens.dens(qi,j,yi[l][t],delta);
    l3+=dens.dens(qi,k,zi[l][t],delta);
    l4+=dens.curr(qi,i,xi[l][t],delta,vix[l][t]);
    l5+=dens.curr(qi,i,yi[l][t],delta,viy[l][t]);
    l6+=dens.curr(qi,i,zi[l][t],delta,viz[l][t]);
    // }
}

rhox[i][j][k][t]=(k1+l1);
rhoy[i][j][k][t]=(k2+l2);
rhoz[i][j][k][t]=(k3+l3);
Jx[i][j][k][t]=(k4+l4);
Jy[i][j][k][t]=(k5+l5);
Jz[i][j][k][t]=(k6+l6);
}
}

// electric and magnetic field

maxwell_equat max;

for(int i=0;i<5;i++){
    for (int j=0;j<5;j++){
        for (int k=0;k<5;k++){

Bx[i][j][k][t+1]=max.magntx(Bx[i][j][k][t],c,Ez[i][j+1][k][t],Ez[i][j][k][t],Ey[i][j][k
+1][t],Ey[i][j][k][t],delta_y,delta_z,delta_t);

By[i][j][k][t+1]=max.magnty(By[i][j][k][t],c,Ez[i+1][j][k][t],Ez[i][j][k][t],Ex[i][j][k
+1][t],Ex[i][j][k][t],delta_x,delta_z,delta_t);

Bz[i][j][k][t+1]=max.magntz(Bz[i][j][k][t],c,Ex[i][j+1][k][t],Ex[i][j][k][t],Ey[i+1][j]
[k][t],Ey[i][j][k][t],delta_x,delta_y,delta_t);

        }
    }
}

for(int i=0;i<5;i++){
    for (int j=0;j<5;j++){
        for (int k=0;k<5;k++){

Ex[i][j][k][t+1]=max.electx(Ex[i][j][k][t],c,delta_y,delta_z,delta_t,Bz[i][j][k][t+1],B
z[i][j-1][k][t+1],By[i][j][k][t+1],By[i][j][k-1][t+1],Jx[i][j][k][t],PI);

Ey[i][j][k][t+1]=max.electy(Ey[i][j][k][t],c,delta_x,delta_z,delta_t,Bz[i][j][k][t+1],B
z[i-1][j][k][t+1],Bx[i][j][k][t+1],Bx[i][j][k-1][t+1],Jy[i][j][k][t],PI);

Ez[i][j][k][t+1]=max.electz(Ez[i][j][k][t],c,delta_y,delta_x,delta_t,Bx[i][j][k][t+1],B
x[i][j-1][k][t+1],By[i][j][k][t+1],By[i-1][j][k][t+1],Jz[i][j][k][t],PI);

        }
    }
}
double s4=0,s1=0,s2=0,s3=0; //sum
double m1=0,m2=0,m3=0,m4=0;

```

IAP-PSC Program

```
double n1=0,n2=0,n3=0,n4=0;

//particle interpolation
//x component
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

s4+=Ex[i][j][k][t+1];
s2+=Bx[i][j][k][t+1];

}
    }
    Eex[l][t+1]=s4;
    Bex[l][t+1]=s2;
}
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

s1+=Ex[i][j][k][t+1];
s3+=Bx[i][j][k][t+1];

}
    }
    Eix[l][t+1]=s1;
    Bix[l][t+1]=s3;
}

//y component
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

m1+=Ey[i][j][k][t+1];
m2+=By[i][j][k][t+1];

}
    }
    Eey[l][t+1]=m1;
    Bey[l][t+1]=m2;
}
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

m3+=Ey[i][j][k][t+1];
m4+=By[i][j][k][t+1];

}
    }
    Eiy[l][t+1]=m3;
```

```

    Biy[l][t+1]=m4;
}

//z component
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

n1+=Ez[i][j][k][t+1];
n2+=Bz[i][j][k][t+1];

            }
        }
        Eez[l][t+1]=n1;
        Bez[l][t+1]=n2;
    }
for(int l=0;l<500;l++){
    for(int i=0;i<5;i++){
        for (int j=0;j<5;j++){
            for (int k=0;k<5;k++){

n3+=Ez[i][j][k][t+1];
n4+=Bz[i][j][k][t+1];

            }
        }
        Eiz[l][t+1]=n3;
        Biz[l][t+1]=n4;
    }
string filename;
ofstream myfile;

stringstream d;
d<<t;
filename= "PSC_"+ d.str();
filename+= ".csv";
myfile.open(filename.c_str());
myfile<<"x,y,z,v,B,E\n";
cout<<"the results are :\n";
for(int i=0;i<500;i++){

myfile<<xe[i][t]<<","<<ye[i][t]<<","<<ze[i][t]<<","<<sqrt(pow(vex[i][t],2)+pow(vey[i][t],2)+pow(vez[i][t],2))<<","<<sqrt(pow(Bex[i][t],2)+pow(Bey[i][t],2)+pow(Bez[i][t],2))<<","<<sqrt(pow(Eex[i][t],2)+pow(Eey[i][t],2)+pow(Eez[i][t],2))<<"\n";

myfile<<xi[i][t]<<","<<yi[i][t]<<","<<zi[i][t]<<","<<sqrt(pow(vix[i][t],2)+pow(viy[i][t],2)+pow(viz[i][t],2))<<","<<sqrt(pow(Bex[i][t],2)+pow(Bey[i][t],2)+pow(Bez[i][t],2))<<","<<sqrt(pow(Eix[i][t],2)+pow(Eiy[i][t],2)+pow(Eiz[i][t],2))<<"\n";

}

myfile.close();
system("pause");
// new time step

}

```











```

    return a.exec();
}

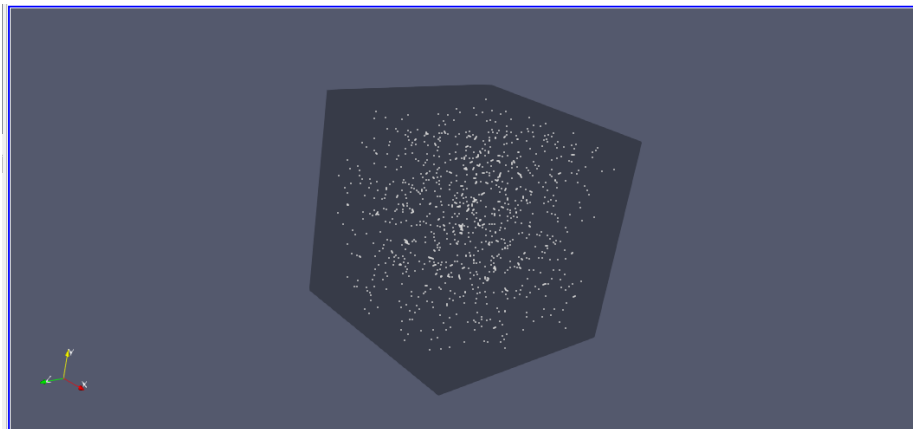
```

51.3 Results

The results are saved in 10 excel sheets and presented on Paraview.

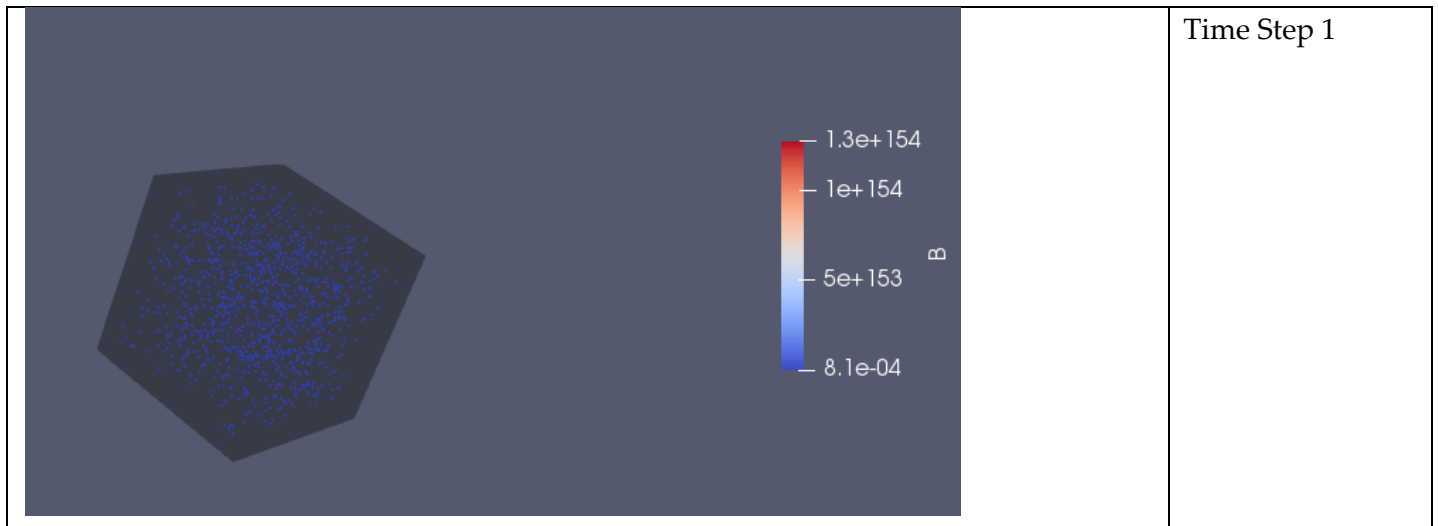
 PSC_0.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	39 KB
 PSC_1.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	55 KB
 PSC_2.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	49 KB
 PSC_3.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	51 KB
 PSC_4.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	51 KB
 PSC_5.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	51 KB
 PSC_6.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	51 KB
 PSC_7.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	51 KB
 PSC_8.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	27 KB
 PSC_9.csv	21-Aug-19 9:25 PM	Microsoft Excel C...	25 KB


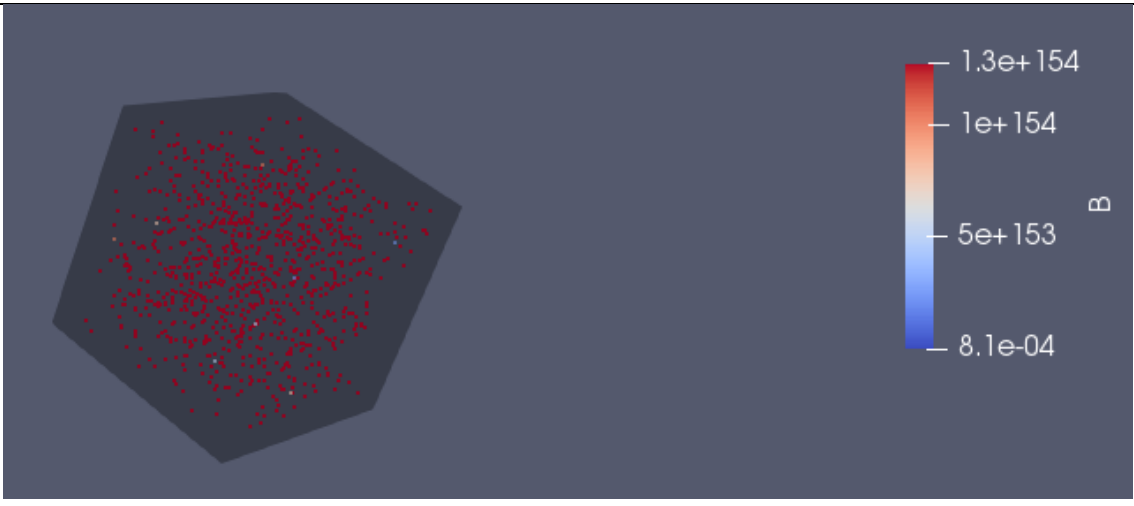
The picture below presents the particles arbitrarily distributed in the box at $t=0$.



we can present any variable we want: velocity, electric field or magnetic field.

The pictures below present the variation of the **magnetic field** with time. The magnetic field varies from time step to another as we can see in the pictures.



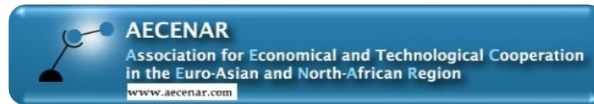
	Time Step 5
	Time Step 9

Literature

<http://ptsg.egr.msu.edu>

INT-LMIC (Laser Matter Interaction Code) (2020)

Based on Master Thesis of Abdurrahman Ibrahim, M.Sc.



Laser Based Flue Gas Detection

Integration and tuning of a laser system for flue gas detection and measurement

Master Thesis

Author:

Abdul Rahman Ibrahim

Supervisors:

Dr. Hassan Amour

Dr. Samir Mourad

Last Update: 26.10.2020

Submitted in fulfilment of the requirements for the degree of
Master of **Physics of the Radiation-Matter Interaction**

Department of Physics-Faculty of Science

Lebanese University

2020

Contents

		583	ACKNOWLEDGEMENT	
		584	ABSTRACT	
		585	INTRODUCTION	1
1.1	PROJECT ENVIRONMENT AND RESEARCH STARTING POINT		ERROR! BOOKMARK NOT DEFINED.	
1.1.1	<i>System Design</i>		Error! Bookmark not defined.	
1.2	TASK OF MASTER THESIS	587		
1.2.1	<i>Modeling and Visualization of Laser-gas interaction</i>	587		
		588	BASICS	2
2.1	PLASMA MODELS FOR LASER-PLASMA INTERACTIONS INCLUDING ULTRA-SHORT LASER PULSES AND HIGH ENERGY CIRCUMSTANCES (RELATIVISTIC MODEL)	588		
2.1.1	<i>Static model</i>		Error! Bookmark not defined.	
2.1.2	<i>Fluid model</i>		Error! Bookmark not defined.	
2.1.3	<i>Kinetic model</i>		Error! Bookmark not defined.	
2.2	LASER-GAS INTERACTION AND LASER-PLASMA INTERACTION	589		
2.2.1	<i>Single Electron Dynamics and Radiation Friction</i>	590		
2.2.2	<i>Motion in Plane Wave Fields</i>		Error! Bookmark not defined.	
2.2.3	<i>Ponderomotive Force</i>	590		
2.2.4	<i>Radiation Friction (Reaction)</i>	592		
2.2.5	<i>Kinetic and Fluid Equations</i>		Error! Bookmark not defined.	
2.3	SIMULATION OF INTERACTION LASER-PLASMA (NUMERICAL ASPECT)	594		
2.3.1	<i>Summary of numerical method to treat the kinetic model of plasma using Particle-in-Cell (PIC) method with Smilei code</i>	594		
2.3.2	<i>The Maxwell-Vlasov model</i>		Error! Bookmark not defined.	
2.3.3	<i>Reference units</i>	594		
2.3.4	<i>Quasi-particles and the PIC method</i>		Error! Bookmark not defined.	
2.3.5	<i>Time- and space-centered discretization</i>		Error! Bookmark not defined.	
2.3.6	<i>Initialization of the simulation</i>		Error! Bookmark not defined.	
2.3.7	<i>The PIC loop</i>	599		
2.3.8	<i>Boundary conditions</i>	603		
		608	CONTRIBUTION: LASER-MATTER INTERACTION IN IAP-PSC CODE	3
3.1	IAP-PSC CODE WITHOUT LASER-MATTER INTERACTION		ERROR! BOOKMARK NOT DEFINED.	
3.1.1	<i>Discussion</i>		Error! Bookmark not defined.	
3.1.2	<i>To be improved in IAP-PSC code without laser-matter interaction</i>		Error! Bookmark not defined.	
3.2	CREATING (YEE) GRID		ERROR! BOOKMARK NOT DEFINED.	
3.3	DISCRETIZATION OF MAXWELL'S EQUATIONS ON THE YEE GRID		ERROR! BOOKMARK NOT DEFINED.	
3.3.1	<i>Finite-difference approximations of Maxwell's equations on a Yee grid</i>		Error! Bookmark not defined.	
3.4	SOLVING THE EQUATIONS FOR THE VARIABLES		ERROR! BOOKMARK NOT DEFINED.	
3.5	VISUALIZATION		ERROR! BOOKMARK NOT DEFINED.	
3.5.1	<i>Paraview Input files</i>		Error! Bookmark not defined.	
3.5.2	<i>Displaying data as points</i>		Error! Bookmark not defined.	
3.5.3	<i>Displaying data as structured grid</i>		Error! Bookmark not defined.	
3.6	SAVING RESULTS		ERROR! BOOKMARK NOT DEFINED.	
			ERROR! BOOKMARK NOT DEFINED. REALIZATION WITH C++	4

Error! No text of specified style in document.

- 4.1 ALGORITHM **ERROR! BOOKMARK NOT DEFINED.**
- 4.2 CLASS DIAGRAM **ERROR! BOOKMARK NOT DEFINED.**

663 TEST RESULTS 5

- 5.1 PULS LASER - CO GAS INTERACTION **ERROR! BOOKMARK NOT DEFINED.**
- 5.2 PULS LASER - NO GAS INTERACTION **ERROR! BOOKMARK NOT DEFINED.**
- 5.3 PULS LASER - SO2 GAS INTERACTION **ERROR! BOOKMARK NOT DEFINED.**
- 5.4 PULS LASER - HF GAS INTERACTION **ERROR! BOOKMARK NOT DEFINED.**
- 5.5 PULS LASER - HCL GAS INTERACTION **ERROR! BOOKMARK NOT DEFINED.**

ERROR! BOOKMARK NOT DEFINED. ANNEX 6

- 6.1 LASER - MATTER INTERACTION C++ CODE **ERROR! BOOKMARK NOT DEFINED.**

ERROR! BOOKMARK NOT DEFINED. REFERENCES

Acknowledgement

First, I want to thank Allah the Almighty for all.

Then I cannot express enough thanks to my committee for their continued support and encouragement: Dr. Adnan Naja; Dr. Samir Mourad; Dr. Hassan Amoud; I offer my sincere appreciation for the jury.

The concept for the measurement environment, which is the base of this work, was done by Siham Aisha and Mariam Abdelkarim. I would like to thank them very much for this.

My completion of this project could not have been accomplished without the support of my dad in the first place, my family, and my classmates. Thank you, Mum, you have been my primary supporter until I completed my higher education.

Finally, to my caring, loving, and supportive wife, Shaymaa: my deepest gratitude. Your encouragement when the times got rough are much appreciated and duly noted. My heartfelt thanks.

Abstract

The general objective of this work is to numerically simulate the behavior of plasma as part of a static model, without or with an external electromagnetic field. The large number of particles brings us back to using the Particle in Cell (PIC) model. It turns out that even with a PIC approach the calculation is still expensive in terms of numerical resources in 3d. However, the case of a one-dimensional plasma gas is successfully processed numerically and results will be presented here where we used both Python and C ++ languages to perform the simulation.

52 Introduction

The project consists of multiple sections, and one of these sections is a device for measuring the emissions of some gases by laser and its interaction with these gases (figure 1.2).

This step is divided into two parts: the practical side and the theoretical side. The theoretical side, is the study of simulation of the interaction of a laser with the gases emitted that we want to measure. This last side is what we will try to achieve in the Master.



Figure 1.1: Real picture of a waste incineration power generation device

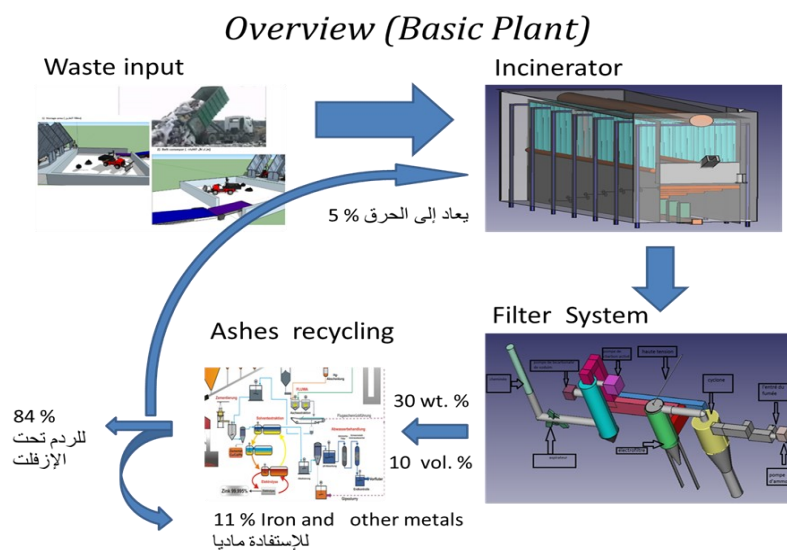


Figure 1.2: basic plant of a waste incineration power generation device.

52.1 Project Environment and Research Starting Point

AECENAR (Association for Economical and Technological Cooperation in the Euro-Asian and North-African Region) has built an incinerator and its filter system. As it is known, this device has many burning residues that will be treated in different ways according to the type of these remains. For the continuation of the waste purification process through filtration systems, it should be measured by the necessary devices.

The Gases are one of the incinerator emissions that must be treated, and to treat it we need to always measure it, so we need a device to measure these gases, to be proved that the concentration of the flue gas is under the norms, i.e. environmentally friendly.

The research committee of AECENAR was decided to use Laser based flue gas measurement technology. To further improve the flue gas measurement procedure, it shall be investigated through simulation, if it is sustainable to make the measurement through laser pulse rather than through continuous wave laser.

The objective of this work is the development of a suitable code of simulating laser-matter interaction, with short pulses, in order to quantify the amount of the flue gas.

52.2 Overview and Task of master thesis

First, we began by identifying the appropriate model to understand the interaction laser-plasma, and we confirm, that the Kinetic model of plasma is the most appropriate model for this idea.

After that, we used certain approaches which can simplify the model in order to facilitate the simulation process. The approach used in this project is Vlasov-Maxwell approach. After this step, we reached the stage where we need the numerical method that can achieve these simulations according to the previously selected model and approaches, which was (PIC). In conclusion, we have performed successfully a simulation for a 1d plasma gases by solving Maxwell's equations. In the 3d cases we have written a C++ code based on the Yee grid.

However, we can not compile this code due to a buffer overflow problem. A super computer will be necessary to overcome this problem.

In chapter 2, we briefly mentioned some theoretical information about the plasma, and then we identified the necessary model to achieve the simulations (Kinetic model) with the necessary approaches (the Maxwell-Vlasov) to the appropriate numerical method (PIC).

Chapter 3 describes the proposed numerical model for the interaction laser-plasma. This model is written in Python. In this chapter, we will display the simulation results and we discuss these results.

In chapter 4, we will present the problems of the code, used by the ACENAR for the simulation of the interaction plasma-LASER. And then, we will discuss the code realized in this project to simulate the solution of Maxwell's equations in the material (plasma). After that, we will present the results with a simplified explanation; using actual computer science methods as class diagrams (for static view of the code) and sequence diagrams (for dynamic view of the code).

52.2.1 Task of master thesis

The master thesis has the following task:

- Modelling and Visualization of LASER gas interaction for the four gases (CO, NO, SO₂, HF, HCl) based on the IAP PSC program.

52.2.2 Modeling and Visualization of Laser-gas interaction

To further improve the flue gas measurement procedure, it shall be investigated through simulation, if it is sustainable to make the make measurement through laser pulse rather than through continuous wave laser.

Steps:

1. Write Algorithm of relativistic model
2. Implement in C++
3. Set the initial value for energy to 3 mW - 5 mW.

53 Basics: Physics of Laser-plasma interaction

In this chapter some theoretical background is provided to describe laser (pulse) radiation-material interaction.

53.1 Introduction

Plasma is defined like a group of charged particles, but the electric aspect (macroscopically) is neutral, this means equal number of positive charge particles and negative charge particles. So, at equilibrium, the plasma therefore contains n_e electrons, n_i ions and n_0 neutral per unit volume. All bodies are transformed into plasma when the temperature and/or density are high enough.

About the interaction between the particles, the dominant force is the Coulomb interaction. Unlike gases where the interactions are short range, in the plasma the interactions are long range, this implies a collective behavior of the particles.

53.1.1 Kinetic-Correlated plasmas

The plasma is correlated at low temperature or high density, and is kinetic at high temperature or low density. The distinction between kinetic and correlated plasma is done by comparing the Coulomb interaction energy with the kinetic energy:

- Kinetic energy: $U_K = \frac{3}{2}k_b T$ (2.1)

- Coulomb interaction energy: $U_{int} = \frac{Z^2 e^2}{4\pi\epsilon_0 d}$ (2.2)

Where d is the distance between two particles, e elementary charge, ϵ_0 electric constant
 Z Charge number, k_b Boltzmann's constant and T temperature.

So when

- $U_K \gg U_{int}$ kinetic plasma, ideal gas behavior.
- $U_K \ll U_{int}$ correlated plasma, the electrostatic forces modify the behavior of charged particles.

Another distinction can be done using the length of Landau r_0 which is defined as follow:

Approach length of two particles of energy $k_b T$.

$$r_0 = \frac{z^2 e^2}{4\pi\epsilon_0 k_b T} \quad (2.3)$$

So when:

- $r_0 \ll d$: the plasma is in kinetic status.
- $r_0 \gg d$: the plasma is in correlated status.

53.2 Plasma models for laser-plasma interactions including ultra-short laser pulses and high energy circumstances¹⁶.

When we want to describe a phenomenon physically, we must use a model that takes into account the influencing factors, and must also respect the physical laws.

On the other hand, in choosing the model, we must keep in mind the objective of the study and the results we want to obtain. These same conditions must be applied when choosing a model to explain the interaction laser-plasma.

There are three models of plasma that can explain this interaction in the high-intensity (10^{18} W/cm²) short pulse (1 *pico second*):

I. Static model

This approach is not concerned with the dynamics of the plasma particles, so it is not a good option to explain the interaction laser-plasma. A static model can be used for describing the laser propagation.

II. Fluid model

This model is developed to describe specific situations when a particle-particle collision is the dominant factor.

III. Kinetic model

This model specifies the particle distributions self-consistently. It is commonly used in laser-plasma interaction simulations, and the mostly used numerical method for solving this model is PIC (Particle-in-cell). It follows the evolution of the laser pulse on the short timescale associated with the laser period and simulates motion of charged particles, or plasma accordingly [1].

53.3 Laser-gas interaction and laser-plasma interaction¹⁷

Exposure of the material to an electromagnetic (EM) field with a power of 10^{21} W, focused on a spot (within one micrometre for sub-picosecond systems), leads to intense ionization of the material, that is, its transformation into a plasma. The freed electrons oscillate with $m_e c$ energy (where m_e is the electron mass and c is the speed of light).

¹⁶ [Modelling of a laser-plasma injector for multi-stage acceleration. introduction]

¹⁷ [Laser-Driven Sources of High Energy Particles and Radiation. Chap2]

The nonlinear dynamics of such **relativistic plasma** in a **super-strong EM field** is the basis of advanced schemes of laser-plasma sources of high energy electrons, ions and photons which are characterized by high brilliance and ultrashort duration.

53.3.1 Single Electron Dynamics and Radiation Friction

53.3.2 Single Electron Dynamics

We started by talking about the dynamics of one electron, meaning about a charged particle, to simplify the explanation of a system consisting of a large number of charged particles (plasma) for example.

The motion of an electron in a *given* EM field is described by the equations:

$$\frac{d\mathbf{P}}{dt} = -e \left(\mathbf{E} + \frac{\mathbf{v}}{c} \times \mathbf{B} \right); \quad \frac{d\mathbf{r}}{dt} = \mathbf{v}; \quad (2.4)$$

Where $\mathbf{P}=\mathbf{P}(t)$ is the momentum, $\mathbf{r}=\mathbf{r}(t)$ the position, $\mathbf{v}=\mathbf{v}(t)=\mathbf{P}/m_e\gamma$ the velocity, and the fields are evaluated at the electron position, i.e $\mathbf{E}=\mathbf{E}(\mathbf{r}(t), t)$ and $\mathbf{B}=\mathbf{B}(\mathbf{r}(t), t)$. By *given* fields we mean that we neglect their self-consistent modification by the motion of the electron.

53.3.3 Kinetic and Fluid Equations

After we have chosen the most appropriate model for understanding and simulating the laser-plasma interaction (kinetic model), we will begin by trying to understand and simplify this model, and then choose the best numeric model to convert it into a code that simulates this interaction.

To explain plasma dynamics more comprehensively, it is necessary to know the distribution function $f_a=f_a(r, p, t)$, which gives the density of particles in the phase space (r, p) for all species a ($a = e, i$) where i for a single ion distribution.

Note that we will not take into account the binary collisions, and for simplicity we neglect any process which may create or annihilation particles. This means that the number of particles of each species (electrons and ions) is conserved, and the distribution function satisfies a continuity equation in the phase space (the Vlasov equation) [2]:

$$\frac{\partial f_a}{\partial t} + \frac{\partial}{\partial r} (\dot{r}_a f_a) + \frac{\partial}{\partial p} (\dot{p}_a f_a) = 0 \quad (2.5)$$

Where

$$\dot{r}_a = v = \frac{pc}{(p^2 + m_a^2 c^2)^{\frac{1}{2}}} \quad \dot{p}_a = q_a \left(E + \frac{v}{c} \times B \right) \quad (2.6)$$

And m_a is mass of species a ($a = e, i$), i for a single ion distribution, q_a charge of each species, v is the velocity, p momentum and \dot{p}_a is the Lorentz force.

The coupling with Maxwell equations for the EM fields $\mathbf{E} = \mathbf{E}(\mathbf{r}, t)$ and $\mathbf{B} = \mathbf{B}(\mathbf{r}, t)$ occurs via the charge and current densities obtained from f_a :

$$\rho(r, t) = \sum_a q_a \int f_a d^3p, \quad J(r, t) = \sum_a q_a \int v f_a d^3p \quad (2.7)$$

Now that we have combined Vlasov's equation with Maxwell's equations (Vlasov-Maxwell), and then we have obtained the basic system on which the kinetic model of laser-plasma interaction is built. In most cases, the PIC method is used to perform this simulation [3].

53.3.4 Ponderomotive Force

The motion in a plane wave is an useful reference case, but in most cases we have to deal with more complex field distributions, such as a laser pulse with a finite extension in space and time. At least we may assume the field to be *quasi-monochromatic*, i.e. to be described by

$$\mathbf{A}(\mathbf{r}, t) = \text{Re}[\tilde{\mathbf{A}}(\mathbf{r}, t)e^{-i\omega t}] \text{ with } \langle \mathbf{A}(\mathbf{r}, t) \rangle \sim 0 \text{ and } \langle \tilde{\mathbf{A}}(\mathbf{r}, t) \rangle \sim \tilde{\mathbf{A}}(\mathbf{r}, t),$$

i.e. the envelope function $\tilde{\mathbf{A}}(\mathbf{r}, t)$ describes the temporal variation of the field on a scale slower than the oscillation at frequency ω . The idea is to separate these different scales by writing for the position $\mathbf{r}(t) \equiv \mathbf{r}_s(t) + \mathbf{r}_o(t)$ where $\langle \mathbf{r}_s(t) \rangle \sim \mathbf{r}_s(t)$ and $\langle \mathbf{r}_o(t) \rangle \sim 0$, i.e. $\mathbf{r}_o(t)$ describes the fast oscillation around the slowly-moving center $\mathbf{r}_s(t)$. In the non-relativistic case, one obtains equations for the “slow” motion as

$$m_e \frac{d\mathbf{v}_s}{dt} = -\frac{e^2}{2m_e\omega^2} \nabla \langle E^2(\mathbf{r}_s(t), t) \rangle \equiv \mathbf{F}_p, \quad \frac{d\mathbf{r}_s}{dt} = \mathbf{v}_s, \quad (2.5)$$

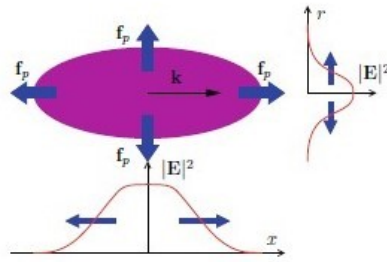
where \mathbf{F}_p is named the *ponderomotive* force (PF). Equation (2.5) is based on a perturbative approach where magnetic effects are taken into account up to first order in v/c , and the spatial variation of the fields over a wavelength is small ($|\lambda \nabla E| \ll E$).

According to (2.5) the electrons are pushed out of the regions where the field is higher. Thus, if a laser pulse propagates through a tenuous plasma (Fig. 2.2), electrons will be pushed in the forward (propagation) direction on the leading edge of the pulse, and in the backward direction on the trailing edge: in proper conditions, this effect generates wake waves in the plasma. The PF associated to the intensity gradient in the radial direction tends to pile electrons at the edge of the laser beam and create a low-density channel along the propagation path, which can cause a self-guiding effect.

An extension of the PF to the relativistic regime is not straightforward. For a quasi-transverse, quasi-plane wave field one may follow the hint that the non-relativistic PF (2.5) is the gradient of the average oscillation energy (“ponderomotive potential”).

Assuming $\mathbf{p}_\perp \sim e\mathbf{A}/c$ and $\gamma \sim (1 + \mathbf{p}_\perp^2/m_e^2c^2)^{1/2}$, one can write the oscillation energy in the relativistic case as $m_e c^2 (\gamma - 1)$ and replace the potential in (2.5). However,

Fig. 2.2 Ponderomotive scattering of electrons by the ponderomotive force (2.5) of a laser pulse having finite length and width



one has also to take into account that the oscillatory motion yields relativistic inertia. One may thus write

$$\frac{d}{dt} (m_{\text{eff}} \mathbf{v}_s) \simeq -\nabla (m_{\text{eff}} c^2), \quad m_{\text{eff}} \equiv m_e (1 + \langle \mathbf{a}^2 \rangle (\mathbf{r}_s, t))^{1/2}, \quad (2.6)$$

(where $\mathbf{a} = e\mathbf{A}/m_e c^2$) with m_{eff} acting as an effective, position- and time-dependent mass. We remark that this expression is limited to a “semi-relativistic case”, in which the average velocity $|\mathbf{v}_s| \ll c$, and for smooth field profiles where transverse components are much larger than longitudinal ones (e.g. a loosely focused laser beam).

53.3.5 Radiation Friction (Reaction)

While an electron is accelerated by an EM field, it also radiates EM waves when accelerated. But the “standard” equations of motion (2.1) do not account for the energy and momentum carried away by the radiation. For example, according to (2.1) an electron in an uniform and constant magnetic field performs a circular orbit at constant energy; but since the electron experiences a centripetal acceleration, it will radiate and lose energy, so that we expect the trajectory to become a spiral as if the electron was experiencing a friction force. To describe such *radiation friction* (RF) effects, additional terms must be added to the Lorentz force in order that the motion is self-consistent with the radiation emission. The phenomenon can also be described as the back-action of the fields generated by the electron on itself, so it is also named *radiation reaction* (RR).

RR (or RF) is a longstanding and classic problem of classical electrodynamics. In ordinary conditions the effect is either negligible or at least it can be treated perturbatively and phenomenologically, e.g. inserting a simple friction force. The dynamics of the electron becomes strongly affected by the radiation emission when the energy of the emitted radiation is comparable to the work done on the electron by the accelerating fields, which implies field strengths at the frontier of those produced by present-day laser technology. This circumstance has revitalized the debate (and associated controversy) on RR in recent years. However, it is apparent that as long as a classical description is adequate, one can safely use the RR force given in the textbook by Landau and Lifshitz (LL):

$$\mathbf{F}_{\text{RR}} \simeq -\frac{2r_c^2}{3} \left(\gamma^2 \left(\mathbf{L}^2 - \left(\frac{\mathbf{v}}{c} \cdot \mathbf{E} \right)^2 \right) \frac{\mathbf{v}}{c} - \mathbf{L} \times \mathbf{B} - \left(\frac{\mathbf{v}}{c} \cdot \mathbf{E} \right) \cdot \mathbf{E} \right), \quad (2.7)$$

where $\mathbf{L} \equiv \mathbf{E} + \mathbf{v} \times \mathbf{B}/c$, $r_c = e^2/m_e c^2$ is the classical electron radius, and small terms containing the temporal derivatives of the fields have been dropped down .

It may be interesting to notice that for an electron which is instantaneously at rest ($\mathbf{v} = 0$) the force reduces to

$$\mathbf{F}_{\text{RR}} \simeq \frac{2r_c^2}{3} \mathbf{E} \times \mathbf{B} = \sigma_T \frac{\mathbf{S}}{c}, \quad (2.8)$$

where $\sigma_T = 8\pi r_c^2/3$ is the Thomson cross section for the scattering of an EM wave, and $\mathbf{S} = c\mathbf{E} \times \mathbf{B}/4\pi$ is the Poynting vector giving the energy flux of the wave (the intensity $I = |\mathbf{S}|$): thus, in this limit the RR force is a drag force which describes the absorption of an amount of EM momentum proportional to the amount of EM energy subtracted from the wave and then radiated away.

An exact solution for the motion in a plane EM wave exists also when the RR force (2.7) is included. The modification of the trajectory is shown in Fig. 2.1, for the same initial conditions yielding the closed “figure of eight” when neglecting RR: if the latter is included, the trajectory opens up with the electron gaining energy and accelerating along the propagation direction. Of course, a friction force sounds as unable to accelerate anything, but actually the effect of friction is to change the relative phase between the fields and the electron velocity. This yields $\langle \mathbf{v} \cdot \mathbf{E} \rangle \neq 0$, so that the electron gains energy from the wave, and $\langle \mathbf{v} \times \mathbf{B} \rangle \neq 0$, so that the electron is accelerated along $\hat{\mathbf{x}}$.

The classical theory predicts that the spectrum of the radiation scattered from a relativistic electron peaks at frequencies $\omega_{\text{rad}} \sim \gamma^3 \omega_i$, where ω_i is the frequency of the incident radiation ($\omega_{\text{rad}} = \omega_i$ in the linear non-relativistic regime).

Thus, with increasing γ eventually the energy of a single photon $\hbar \omega_{\text{rad}} > m_e c^2 \gamma$, the electron energy, so that the recoil from the photon emission is not negligible and a quantum electrodynamics (QED) description becomes necessary. This is reminiscent of the well-known Compton scattering, but here the relevant regime involves the sequential absorption of very many low-frequency photons and the emission of several high-frequency photons. A QED theory of RR is still an open issue and is the subject of current research.

53.4 Simulation of Interaction LASER-Plasma (numerical aspect)¹⁸

53.4.1 Summary of numerical method to treat the kinetic model of plasma using Particle-in-Cell (PIC) method with Smilei code¹⁹

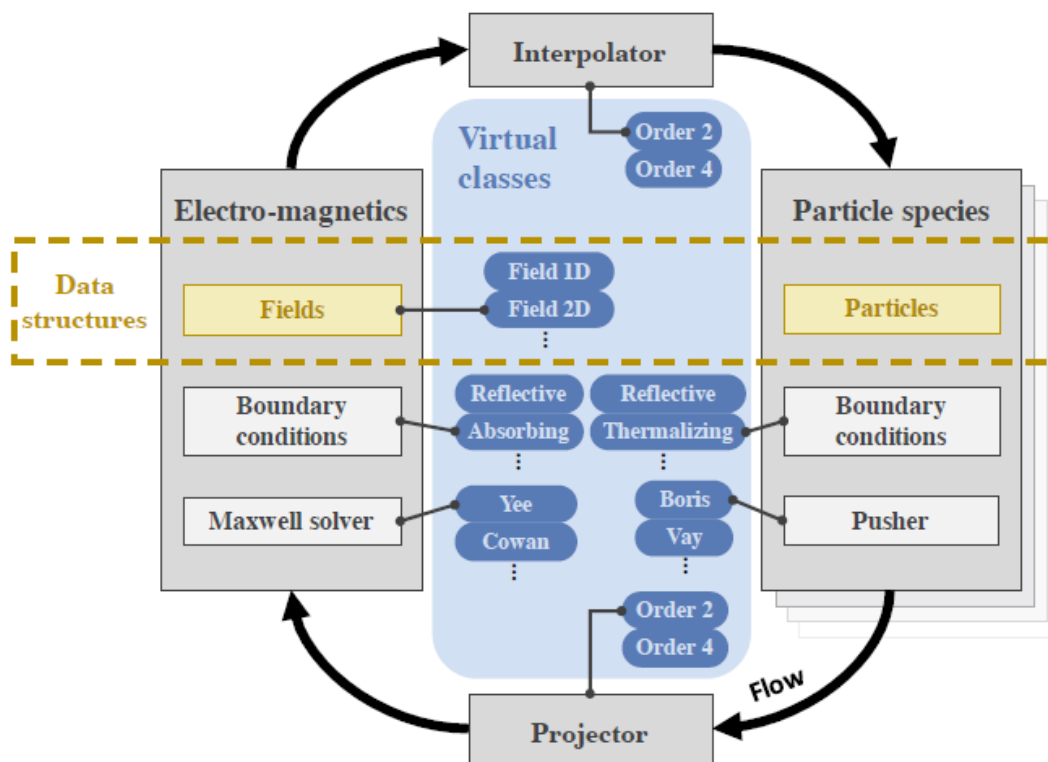


Figure 2: C++ flow, classes and data structure in SMILEI.

PIC is a model developed for fluid dynamics studies, and its approach has many of advantages, like as conceptual simplicity and efficient implementation on massively parallel computers etc..

Smilei is a programme that is used in several range of physics studies, like a relativistic Laser-plasma interaction and astrophysical plasmas etc..

As for the source, Smilei is open-source, object-oriented (C++) particle-in-cell (PIC) code, and its core program is C++ object-oriented programming provides an efficient way of structuring the code.

53.4.1.1 The Maxwell-Vlasov model

After we talked about the best plasma model to explain the laser-plasma interaction in the paragraph 2.3.III. And, we referred to the approximations that we took into account as like as

¹⁸ Reference?

¹⁹ Reference? [Smilei: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation]

neglect any forces that lead to create or destroy the particles, and we considered that the collision between particles is negligible. Then we can now apply the model of Vlasov-Maxwell. In talking about the latter, we note that the different species constituting the plasma are described by their respective distribution function $f_s(t; \mathbf{X}; \mathbf{P})$, where s denotes a given species consisting of particles with charge q_s and mass m_s , and \mathbf{X} and \mathbf{P} denote the position and momentum of a phase-space element. The distribution f_s satisfies Vlasov's equation:

$$\left(\partial_t + \frac{\mathbf{P}}{m_s \gamma} \cdot \nabla + \mathbf{F}_L \cdot \nabla_P \right) f_s = 0 \quad (2.8)$$

Where $\gamma = \sqrt{1 + \frac{\mathbf{P}^2}{(m_s c)^2}}$ is the (relativistic) Lorentz factor, c is the speed of light in vacuum, and

$$\mathbf{F}_L = q_s (\mathbf{E} + \mathbf{v} \wedge \mathbf{B}) \quad (2.9)$$

is the Lorentz force acting on a particle with velocity $\mathbf{v} = \mathbf{P} / (m_s \gamma)$.

This force follows from the existence, in the plasma, of collective electric [$\mathbf{E}(t; \mathbf{x})$] and magnetic [$\mathbf{B}(t; \mathbf{x})$] fields satisfying Maxwell's equations:

$$\nabla \cdot \mathbf{B} = 0 \quad (2.10 \text{ a})$$

$$\nabla \cdot \mathbf{E} = \rho / \epsilon_0 \quad (2.10 \text{ b})$$

$$\nabla \times \mathbf{B} = \mu_0 \epsilon_0 \partial_t \mathbf{E} \quad (2.10 \text{ c})$$

$$\nabla \times \mathbf{E} = -\partial_t \mathbf{B} \quad (2.10 \text{ d})$$

Where ϵ_0 and μ_0 are the vacuum permittivity and permeability, respectively.

The Vlasov-Maxwell system of Eqs. 2.8 -2.10 describes the self-consistent dynamics of the plasma which constituents are subject to the Lorentz force, and in turn modify the collective electric and magnetic fields through their charge and current densities as follow:

$$\rho(r, t) = \sum_s q_s \int f_s d^3p(t, X, P),$$

$$J(r, t) = \sum_s q_s \int v f_s d^3p(t, X, P).$$

Where $\rho(r, t)$ is the charge and $J(r, t)$ is the current densities.

53.4.1.2 Quasi-particles and the PIC method

The "Particle-In-Cell" method owes its name to the discretization of the distribution function f_s as a sum of N_s "quasi-particles" (also referred to as "super-particles" or "macro-particles"):

$$f_s(t, X, P) = \sum_{p=1}^{N_s} \omega_p S(X - X_p(t)) \delta(P - P_p(t)) \quad (2.11)$$

Where ω_p is a quasi-particle "weight", x_p is its position, \mathbf{P}_p is its momentum, δ is the Dirac distribution, and $S(x)$ is the shape-function of all quasi-particles. The properties of the shape-function used in Smilei are given in reference [4].

In PIC codes, Vlasov's Eq. 2.8 is integrated along the continuous trajectories of these quasi-particles, while Maxwell's Eqs. 2.10 are solved on a discrete spatial grid. The spaces between consecutive grid points being referred to as "cells". Injecting the discrete distribution function of Eq. 2.11 in Vlasov's Eq. 2.8, multiplying the result by \mathbf{p} and integrating over all \mathbf{p} leads to:

$$\sum_{p=1}^{N_s} \omega_p \mathbf{P}_p \cdot [\partial_{x_p} S(x - x_p) + \partial_x S(x - x_p)] \mathbf{v}_p + \sum_{p=1}^{N_s} \omega_p S(x - x_p) [\partial_t \mathbf{P}_p - q_s (\mathbf{E} + \mathbf{v}_p \times \mathbf{B})] = 0 \quad (2.12)$$

Where we have introduced $\mathbf{v}_p = \mathbf{P}_p / (m_s \gamma_p) = dx_p / dt$ the p^{th} quasi-particle velocity, and $\gamma_p = (1 + P_p^2 / (m_s^2))^{1/2}$ its Lorentz factor. Considering all p quasi-particles independently, and integrating over all (real) space x , the first term in Eq. 2.12 vanishes due to the properties of the shape-function, and one obtains that all quasi-particles satisfy the relativistic equations of motion:

$$\frac{d\mathbf{x}_p}{dt} = \frac{\mathbf{u}_p}{\gamma_p} \quad (2.13)$$

$$\frac{d\mathbf{u}_p}{dt} = r_s \left(\mathbf{E}_p + \frac{\mathbf{u}_p}{\gamma_p} \times \mathbf{B}_p \right) \quad (2.14)$$

Where $r_s = q_s / m_s$ the charge-over-mass ratio (for species s), $\mathbf{u}_p = \mathbf{P}_p / m_s$ the quasi-particle reduced momentum, and the fields interpolated at the particle position:

$$\mathbf{E}_p = \int dx S(x - x_p) \mathbf{E}(x) \quad (2.15)$$

$$\mathbf{B}_p = \int dx S(x - x_p) \mathbf{B}(x) \quad (2.16)$$

Note that, because of the finite (non-zero) spatial extension of the quasi-particles, additional cells (called *ghost cells*) have to be added at the border of the simulation domain to ensure that the full quasi-particle charge and/or current densities are correctly projected onto the simulation grid.

53.4.1.3 Time- and space-centered discretization

Maxwell's equations are solved here using the Finite Difference Time Domain (FDTD) approach [4] as well as refined methods based on this algorithm. In these methods, the electromagnetic fields are discretized onto a staggered grid, the Yee-grid, which allows for spatial-centering of the discretized curl operators in Maxwell's Eqs. (2.10c) and (2.10d). Figure 2.2 summarizes at which points of the Yee-grid the electromagnetic fields, as well as charge and density currents, are

defined. Similarly, the time-centering of the time-derivative in Maxwell's Eqs. (2.10c) and (2.10d) is ensured by considering the electric fields as defined at integer timesteps (n) and magnetic fields at half-integer time-steps ($n + 1/2$). Time-centering of the magnetic fields is however necessary for diagnostic purposes, and most importantly when computing the Lorentz force acting on the quasi-particles. It should also be noted, that a leap-frog scheme is used to advance the particles in time, so that their positions and velocities are defined at integer (n) and half-integer ($n - 1/2$) time-steps, respectively [4].

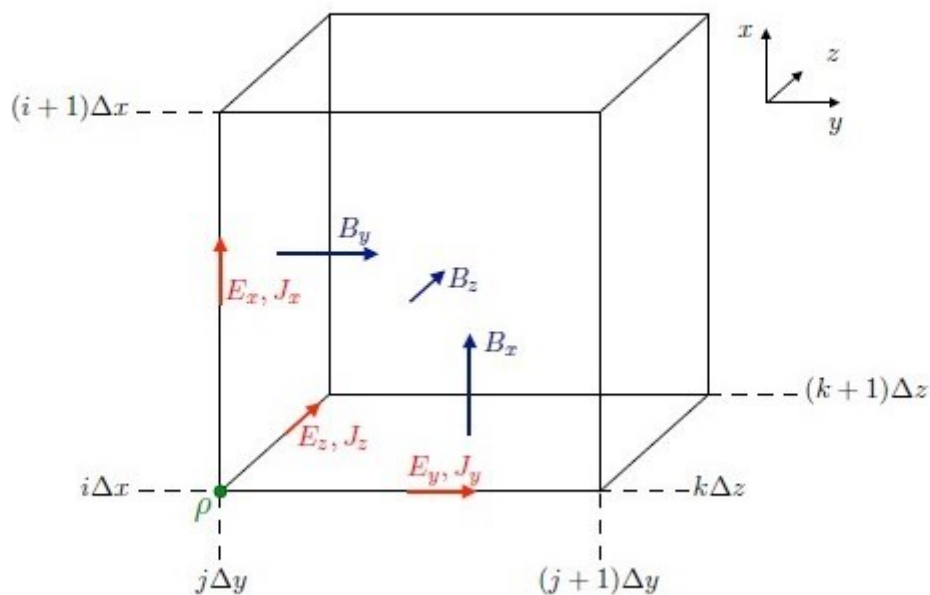


Figure 2.2: Representation of the staggered Yee-grid. The location of all fields and current densities follows from the (rather standard) convention to define charge densities at the cell nodes [4].

53.4.1.4 Initialization of the simulation and the PIC loop

The initialization of a PIC simulation is a three-step process consisting in:

- (i) loading particles,
- (ii) computing the initial total charge and current densities onto the grid,
- (iii) computing the initial electric and magnetic field at the grid points.

At the end of the initialization stage [time-step ($n = 0$)], all quasi-particles in the simulation have been loaded and the electromagnetic fields have been computed over the whole simulation grid.

The PIC loop is then started over N time-steps each consisting in:

- (i) interpolating the electromagnetic fields at the particle positions,

- (ii) computing the new particle velocities and positions,
- (iii) projecting the new charge and current densities on the grid,
- (iv) computing the new electromagnetic fields on the grid. (Tableau 2.1).

Table 2.1: summary of SMILEI's PIC algorithm [4].

Initialization	time step $n = 0$, time $t = 0$
Particle loading	$\forall p$, define $(\mathbf{x}_p)^{n=0}$, $(\mathbf{u}_p)^{n=-\frac{1}{2}}$
Charge projection on grid	$[\forall p, (\mathbf{x}_p)^{n=0}] \rightarrow \rho^{(n=0)}(\mathbf{x})$
Compute initial fields	- solve Poisson on grid: $[\rho^{(n=0)}(\mathbf{x})] \rightarrow \mathbf{E}_{\text{stat}}^{(n=0)}(\mathbf{x})$ - add external fields: $\mathbf{E}^{(n=0)}(\mathbf{x}) = \mathbf{E}_{\text{stat}}^{(n=0)}(\mathbf{x}) + \mathbf{E}_{\text{ext}}^{(n=0)}(\mathbf{x})$ $\mathbf{B}^{(n=\frac{1}{2})}(\mathbf{x}) = \mathbf{B}_{\text{ext}}^{(n=\frac{1}{2})}(\mathbf{x})$
PIC loop: from time step n to $n + 1$, time $t = (n + 1) \Delta t$	
Restart charge & current densities	
Save magnetic fields value (used to center magnetic fields)	
Interpolate fields at particle positions	$\forall p, [\mathbf{x}_p, \mathbf{E}^{(n)}(\mathbf{x}), \mathbf{B}^{(n)}(\mathbf{x})] \rightarrow \mathbf{E}_p^{(n)}, \mathbf{B}_p^{(n)}$
Push particles	- compute new velocity $\forall p, \mathbf{p}_p^{(n-\frac{1}{2})} \left[\mathbf{E}_p^{(n)}, \mathbf{B}_p^{(n)} \right] \mathbf{p}_p^{(n+\frac{1}{2})}$ - compute new position $\forall p, \mathbf{x}_p^{(n)} \left[\mathbf{p}_p^{(n+\frac{1}{2})} \right] \mathbf{x}_p^{(n+1)}$
Project current onto the grid using a charge-conserving scheme	
$[\forall p, \mathbf{x}_p^{(n)}, \mathbf{x}_p^{(n+1)}, \mathbf{p}_p^{(n+\frac{1}{2})}] \rightarrow \mathbf{J}^{(n+\frac{1}{2})}(\mathbf{x})$	
Solve Maxwell's equations	
- solve Maxwell-Faraday: $\mathbf{E}^{(n)}(\mathbf{x}) \left[\mathbf{J}^{(n+\frac{1}{2})}(\mathbf{x}) \right] \mathbf{E}^{(n+1)}(\mathbf{x})$	
- solve Maxwell-Ampère: $\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x}) \left[\mathbf{E}^{(n+1)}(\mathbf{x}) \right] \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x})$	
- center magnetic fields: $\mathbf{B}^{(n+1)}(\mathbf{x}) = \frac{1}{2} \left(\mathbf{B}^{(n+\frac{1}{2})}(\mathbf{x}) + \mathbf{B}^{(n+\frac{3}{2})}(\mathbf{x}) \right)$	

53.4.2 Reference units

Smilei is a fully-relativistic electromagnetic PIC code. As such, it is convenient to normalize all velocities in the code to c . Furthermore, charges and masses are normalized to e and m_e , respectively, with $-e$ the electron charge and m_e its mass. Momenta and energies (and by extension temperatures) are then expressed in units of $m_e c$ and $m_e c^2$, respectively.

The normalization for time and space is not decided a priori. Instead, all the simulation results may be scaled by an arbitrary factor. Denoting the (a priori unknown) time units by ω_r^{-1} , distances are normalized to c/ω_r . Electric and magnetic fields are expressed in units of $m_e c \omega_r / e$ and $m_e \omega_r / e$, respectively. We define the units for number densities as $n_r = \epsilon_0 m_e \omega_r^2 / e^2$, while charge and current densities are in units of $e n_r$ and

$e c n_r$, respectively. Note that this definition of n_r is chosen for best simplification of the Vlasov-Maxwell equations, but does not correspond to the reference distance c/ω_r to the power of -3.

Let us now illustrate by two simple examples this choice of normalization. When dealing with a plasma at constant density n_e , it is convenient to normalize times by introducing the electron plasma frequency $\omega_{pe}=(e^2 n_e/(\epsilon_0 m_e))^{1/2}$. Choosing $\omega_r = \omega_{pe}$,

Units of velocity	c
Units of charge	e
Units of mass	m_e
Units of momentum	$m_e c$
Units of energy, temperature	$m_e c^2$
Units of time	ω_r^{-1}
Units of length	c/ω_r
Units of number density	$n_r = \epsilon_0 m_e \omega_r^2/e^2$
Units of current density	$e c n_r$
Units of pressure	$m_e c^2 n_r$
Units of electric field	$m_e c \omega_r/e$
Units of magnetic field	$m_e \omega_r/e$
Units of Poynting flux	$m_e c^3 n_r/2$

Table 1: List of the most common normalizations used in SMILEI. The value of ω_r is not defined *a priori*, but can be set *a posteriori* as a scaling factor. For simulations requiring the use of ionization and/or collision modules (see Sec. 5), ω_r needs to be defined, in SI units, by the user.

distances are now expressed in units of the electron skin-depth c/ω_{pe} , while number densities are normalized to n_e , and the electric and magnetic fields are in units of $m_e \omega_{pe} c/e$ and $m_e \omega_{pe}/e$, respectively.

In contrast, when considering the irradiation of a plasma by a laser with angular frequency ω_0 , it is convenient to use $\omega_r = \omega_0$. From this choice, it follows that distances are measured in units of $k_0^{-1}= c/\omega_0$, while the electric and magnetic fields are in units of $E_c = m_e c \omega_0/e$ and $m_e \omega_0/e$, respectively. Note that E_c is the Compton field, which is widely used to measure the importance of relativistic effects in laser-plasma interaction. In addition, number densities are expressed in units of $n_c = \epsilon_0 m_e \omega_0^2/e^2$, the well-known critical density delimiting plasmas that are transparent or opaque to an electromagnetic radiation with angular frequency ω_0 .

Table 1 gives a list of the most common normalizations used in Smilei. In what follows (and if not specified otherwise), all quantities will be expressed in normalized units.

53.4.3 The PIC loop in detail

At the end of the initialization stage [time-step ($n = 0$)], all quasi-particles in the simulation have been loaded and the electromagnetic fields have been computed over the whole simulation grid. The PIC loop is then started over N time-steps each consisting in (i) interpolating the electromagnetic fields at the particle positions, (ii) computing the new particle velocities and positions, (iii) projecting the new charge and current densities on the grid, and (iv) computing the new electromagnetic fields on the grid. In this section, we describe these four steps taken to advance from time-step (n) to time-step ($n+1$).

53.4.3.1 Field interpolation at the particle

At the beginning of time-step (n), the particles velocities and positions are known at time-step (n - 1/2) and (n), respectively. For each particle p, the electromagnetic fields [at time-step (n)] are computed at the particle position using a simple interpolation technique:

$$\mathbf{E}_p^{(n)} = \int d\mathbf{x} S(\mathbf{x} - \mathbf{x}_p^{(n)}) \mathbf{E}^{(n)}(\mathbf{x}), \quad (13)$$

$$\mathbf{B}_p^{(n)} = \int d\mathbf{x} S(\mathbf{x} - \mathbf{x}_p^{(n)}) \mathbf{B}^{(n)}(\mathbf{x}), \quad (14)$$

9

where we have used the time-centered magnetic fields $\mathbf{B}^{(n)} = 1/2 [\mathbf{B}^{(n+1/2)} + \mathbf{B}^{(n-1/2)}]$. Additional information on the field interpolation are given in [Appendix A].

53.4.3.2 Particle pusher

Knowing, for each quasi-particle, the electromagnetic fields at its position, the new particle momentum and position are computed using a (second order) leap-frog integrator. In Smilei, two different schemes have been implemented, the well-known Boris pusher [19] and the one developed by J.-L. Vay [20]. Both schemes compute the new particle momentum according to:

$$\mathbf{u}_p^{(n+1/2)} = \mathbf{u}_p^{(n-1/2)} + r_s \Delta t \left[E_p^{(n)} + \frac{\mathbf{v}_p^{(n+1/2)} + \mathbf{v}_p^{(n-1/2)}}{2} \times B_p^{(n)} \right], \quad (15)$$

as well as the new particle position:

$$\mathbf{x}_p^{(n+1)} = \mathbf{x}_p^{(n)} + \Delta t \frac{\mathbf{u}_p^{(n+1/2)}}{\gamma_p}, \quad (16)$$

where Δt denotes the duration of a time-step.

The Boris pusher is a widely-used second-order leap-frog solver. However, Ref. [20] shows that it introduces errors when calculating the orbits of relativistic particles in special electromagnetic field configurations (e.g. when the electric and magnetic contributions cancel each other in the Lorentz force). Vay's solver proposes an alternative formulation of the leap-frog solver that prevents such problems with an additional (albeit not large) computational cost.

53.4.3.3 Charge conserving current deposition

Charge deposition (i.e. charge and current density projection onto the grid) is then

performed using the charge-conserving algorithm proposed by Esirkepov [21]. The current densities in the dimensions of the grid (i.e., the x-direction for 1-dimensional simulations, both x- and y-directions for 2-dimensional simulations, and all three x-, y- and z-directions for 3-dimensional simulations) are computed from the charge flux through the cell borders (hence ensuring charge conservation) while the current densities along the other dimensions are performed using a simple projection. To illustrate this point, we take the example of current deposition in a 2-dimensional simulation. The current densities in the x- and y-directions associated to a particle with charge q are computed as:

$$(J_{x,p})_{i+\frac{1}{2},j}^{(n+\frac{1}{2})} = (J_{x,p})_{i-\frac{1}{2},j}^{(n+\frac{1}{2})} + q w_p \frac{\Delta x}{\Delta t} (W_x)_{i+\frac{1}{2},j}^{(n+\frac{1}{2})} \quad (17)$$

$$(J_{y,p})_{i,j+\frac{1}{2}}^{(n+\frac{1}{2})} = (J_{y,p})_{i,j-\frac{1}{2}}^{(n+\frac{1}{2})} + q w_p \frac{\Delta y}{\Delta t} (W_y)_{j,i+\frac{1}{2}}^{(n+\frac{1}{2})} \quad (18)$$

where $(W_x)^{(n+1/2)}$ and $(W_y)^{(n+1/2)}$ are computed from the particle present and former positions $x_p^{(n+1)}$ and $x_p^{(n)}$, respectively, using the method developed by Esirkepov. The particle current in the z-direction (not a dimension of the grid) is, in this geometry, computed using the direct projection technique described in [Appendix A]:

$$(J_{z,p})_{i,j} = q w_p \mathbf{v}_p \int d\mathbf{x} S(\mathbf{x} - \mathbf{x}_p) P_D(\mathbf{x} - \mathbf{x}_{i,j}). \quad (19)$$

The charge density deposited by the particle can be obtained, if required e.g. for diagnostic purpose, using a similar direct projection.

The total charge and current densities henceforth gather the contributions of all quasiparticles of all species. It is worth noting that, within a charge-conserving framework, charge densities are only projected on the grid for diagnostics purposes (as we will see in next paragraph, it is not used to advance the electromagnetic fields).

53.4.3.4 Maxwell solvers

Now that the currents are known at time-step $(n+ 1/2)$, the electromagnetic fields can be advanced solving Maxwell's Eqs. (3). First, Maxwell-Ampere Eq. (3c) is solved, giving the advanced electric fields:

$$\mathbf{E}^{(n+1)} = \mathbf{E}^{(n)} + \Delta t \left[(\nabla \times \mathbf{B})^{(n+\frac{1}{2})} - \mathbf{J}^{(n+\frac{1}{2})} \right]. \quad (20)$$

Then, Maxwell-Faraday Eq. (3d) is computed, leading to the advanced magnetic fields:

$$\mathbf{B}^{(n+\frac{3}{2})} = \mathbf{B}^{(n+\frac{1}{2})} - \Delta t (\nabla \times \mathbf{E})^{(n+1)}. \quad (21)$$

Before discussing the discretization of the curl-operator in more details, it is worth noting that solving Eqs. (3c) and (3d) is sufficient to get a complete description of the new electromagnetic fields. Indeed, it can be shown that this conserves a divergence-free magnetic field if Gauss' Eq. (3a) is satisfied at time $t = 0$. Similarly, Poisson's Eq. (3b) is verified as long as it is satisfied at time $t = 0$ as long as the charge deposition algorithm fulfills the charge conservation equation:

$$\partial_t \rho + \nabla \cdot \mathbf{J} = 0 \quad (22)$$

This motivated the use of Esirkepov's projection scheme discussed in the previous paragraph. We conclude this Section by discussing in more details the discretization of the curl operators in Eqs. (3c) and (3d). To do so, let us focus on the equations for the electric and magnetic fields E_x and B_x discretized on the (staggered) Yee-grid:

$$\frac{(E_x)_{i+\frac{1}{2},j,k}^{(n+1)} - (E_x)_{i+\frac{1}{2},j,k}^{(n)}}{\Delta t} = (J_x)_{i+\frac{1}{2},j,k}^{n+\frac{1}{2}} + (\partial_y B_z)_{i+\frac{1}{2},j,k}^{(n+\frac{1}{2})} - (\partial_z B_y)_{i+\frac{1}{2},j,k}^{(n+\frac{1}{2})}, \quad (23)$$

$$\frac{(B_x)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+\frac{3}{2})} - (B_x)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+\frac{1}{2})}}{\Delta t} = (\partial_z^* E_y)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+1)} - (\partial_y^* B_z)_{i,j+\frac{1}{2},k+\frac{1}{2}}^{(n+\frac{1}{2})}. \quad (24)$$

The partial derivatives in space in both equations are discretized as follows. In the Maxwell-Ampere equation, the partial derivative in x (similarly in y and z) reads:

$$(\partial_x F)_{i,j,k} = \frac{F_{i+\frac{1}{2},j,k} - F_{i-\frac{1}{2},j,k}}{\Delta x}, \quad (25)$$

and corresponds to the usual curl-operator discretization used in the FDTD method. In the Maxwell-Faraday equation, the partial derivatives can be modified using an extended stencil (see Ref. [15] for a comparative study of different solvers). The spatial derivative in the x-direction (similarly in the y and z directions) reads:

$$\begin{aligned} (\partial_x^* F)_{i,j,k} &= \alpha_x \frac{F_{i+\frac{1}{2},j,k} - F_{i-\frac{1}{2},j,k}}{\Delta x} + \eta_x \frac{F_{i+\frac{3}{2},j,k} - F_{i-\frac{3}{2},j,k}}{\Delta x} \\ &+ \beta_{xy} \left[\frac{F_{i+\frac{1}{2},j+1,k} - F_{i-\frac{1}{2},j+1,k}}{\Delta x} + \frac{F_{i+\frac{1}{2},j-1,k} - F_{i-\frac{1}{2},j-1,k}}{\Delta x} \right] \\ &+ \beta_{xz} \left[\frac{F_{i+\frac{1}{2},j,k+1} - F_{i-\frac{1}{2},j,k+1}}{\Delta x} + \frac{F_{i+\frac{1}{2},j,k-1} - F_{i-\frac{1}{2},j,k-1}}{\Delta x} \right], \end{aligned} \quad (26)$$

the set of parameters α_x , η_x , β_{xy} and β_{xz} depending of the type of solver used [15], and the standard FDTD solver is recovered for $\alpha_x = 1$; $\eta_x = \beta_{xy} = \beta_{xz} = 0$.

Note that the FDTD solvers are subject to a Courant-Friedrich-Lewy (CFL) condition.

For the standard solver, the CFL condition requires the time-step to be smaller than:

$$\Delta t_{\text{CFL}} = \sum_{\mu} (\Delta \mu^{-2})^{-\frac{1}{2}}, \quad (27)$$

$\mu = (x; y; z)$ standing for the different spatial directions resolved in the simulation.

53.4.4 Boundary conditions

After having computed new quasi-particle positions and velocities, boundary conditions (BCs) are applied to each quasi-particle that may be located in a ghost cell, i.e. outside of the 'real' grid. Quasi-particle species may have a different BC for each boundary of the simulation box: the quasi-particles can either loop around the box (periodic), be stopped (momentum set to zero), suppressed (removed from memory), reflected (momentum and position follow specular reflection rules) or thermalized. In the latter case, the quasi-particle is set back inside the simulation box, and its new momentum is randomly sampled in a Maxwellian distribution [22] with a given temperature and drift velocity, both specified by the user.

BCs are applied to the electromagnetic fields after Maxwell's equations have been solved. Each boundary of the simulation box can feature a different BC. First, injecting/absorbing BCs inspired from the "Silver-Müller" BC [23] are able to inject an electromagnetic wave (e.g. a laser) and/or to absorb outgoing electromagnetic waves. In contrast, the reflective electromagnetic BC will reflect any outgoing electromagnetic wave reaching the simulation boundary. Lastly, periodic BCs are also available.

53.5 Quantum Electrodynamics QED and Quantum Chromodynamics QCD

53.5.1 QED MODEL IN PARTICLE-IN-CELL SIMULATIONS²⁰

To simulate photon emission, we use a simple numerical model based on a Monte-Carlo method. For each charged particle moving in an electromagnetic field, we calculate the total probability of photon emission, W_{ph} , and the radiant intensity, I_{ph} , at each time step. The probability of photon emission and the radiant intensity can be calculated using a QED approach as

²⁰ [Energy partition, γ -ray emission, and radiation reaction in the near-quantum electro-dynamical regime of laser-plasma interaction]

$$W_{ph} = \frac{\alpha m_e^2 c^4}{3\sqrt{3}\pi\hbar\epsilon} \int_0^\infty dx \frac{5x^2 + 7x + 5}{(1+x)^3} K_{2/3}\left(\frac{2x}{3\chi}\right), \quad (1)$$

$$I_{ph} = \frac{e^2 m_e c^4}{3\sqrt{3}4\pi^2 \hbar^2} \int_0^\infty dx \frac{x(4x^2 + 5x + 4)}{(1+x)^4} K_{2/3}\left(\frac{2x}{3\chi}\right), \quad (2)$$

where e and m_e are the electron charge and mass, respectively, c is the speed of light, \hbar is Planck's constant, $\alpha=e^2/\hbar c$ is the fine-structure constant, γ is the gamma-factor of the electron, and $K_{2/3}(x)=2^{-1/2}3^{1/3}x^{-2/3} \int_z^\infty \sin[3(x/2)^{2/3} z+z^3]dz$ is the McDonald function. It follows from Eqs. (1) and (2) that the probability and intensity depend on the QED parameter

$$\chi = \frac{e\hbar}{m_e^3 c^4} \sqrt{\left(\frac{e\mathbf{E}}{c} + \mathbf{p} \times \mathbf{H}\right)^2 - (\mathbf{p} \cdot \mathbf{E})^2}, \quad (3)$$

where \mathbf{E} and \mathbf{H} are the components of the electric and magnetic fields and \mathbf{p} is the electron momentum. In the limit, where $\chi \ll 1$, photon emission can be treated classically. For $\chi > 1$, QED effects such as electron spin effect and recoil must be taken into account. The peak value of χ can be estimated as $\chi_m \approx \gamma a_0 (r_e/\lambda_0)(2\pi/\alpha)$ from Eq. (3), where $r_e = e^2/m_e c^2$ is the classical electron radius, $a_0 = eE_0/m_e \omega_0 c$ is the normalized laser amplitude (E_0 and ω_0 are the laser electric field and frequency, respectively) and $\lambda_0 = 0.8 \mu\text{m}$ is the laser wavelength. In our simulations, $\gamma \approx 1000$ for $a_0 \approx 200\text{--}1000$, giving $\chi_m \approx 0.6\text{--}3$, leading to a significant energy transfer to γ -photons. As such, a QED method is necessary to model the RR and γ -photon emission. The time moment of the photon emission is calculated according to an event generator based on the probability defined by Eq. (1). The photon energy is chosen to be the mean photon energy introduced as the ratio of the intensity of radiation to the total probability of the photon emission, according to

$$\langle \hbar\omega \rangle = \frac{I_{ph}}{W_{ph}}. \quad (4)$$

The photon energy is subtracted from the energy of the charged particle emitting the photon. Implementing this energy conservation allows recoil effects to be accounted for.

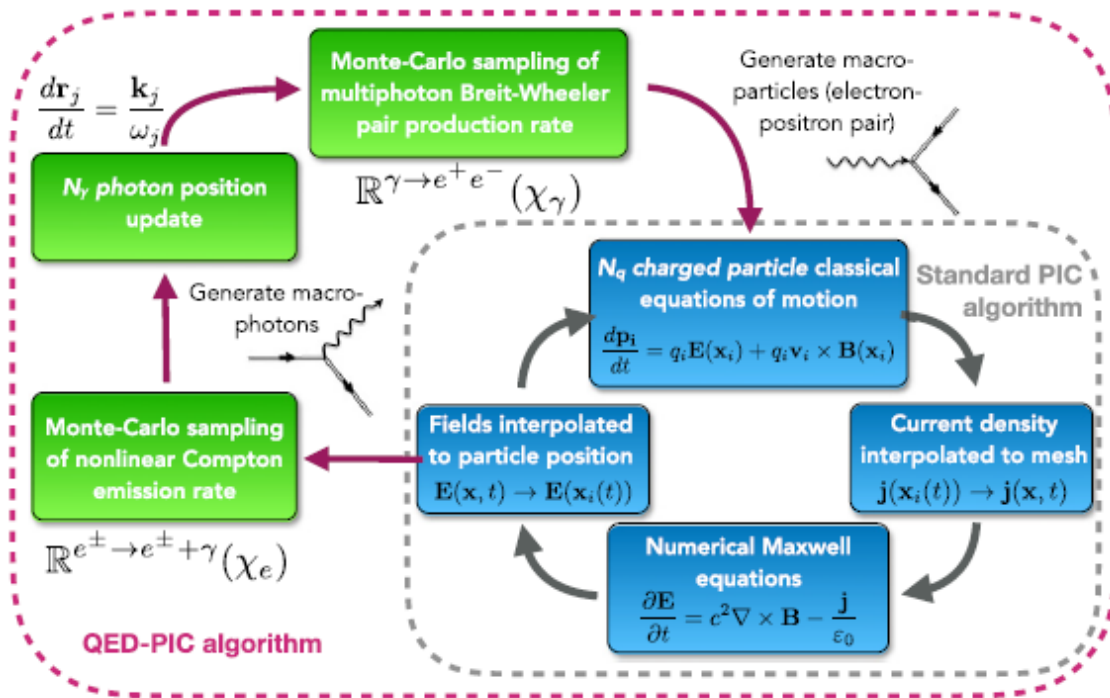


FIG. 6. Framework of QED PIC.

53.5.2 Quantum Chromodynamics QCD²¹

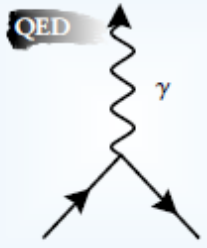
Quantum chromodynamics, familiarly called QCD, is the modern theory of the strong interaction. Historically its roots are in nuclear physics and the description of ordinary matter—understanding what protons and neutrons are and how they interact. Nowadays QCD is used to describe most of what goes on at high-energy accelerators. Twenty or even fifteen years ago, this activity was commonly called “testing QCD.” Such is the success of the theory, that we now speak instead of “calculating QCD backgrounds” for the investigation of more speculative phenomena. For example, discovery of the heavy W and Z bosons that mediate the weak interaction, or of the top quark, would have been a much more difficult and uncertain affair if one did not have a precise, reliable understanding of the more common processes governed by QCD. With regard to things still to be found, search strategies for the Higgs particle and for manifestations of supersymmetry depend on detailed understanding of production mechanisms and backgrounds calculated by means of QCD. Quantum chromodynamics is a precise and beautiful theory. One reflection of this elegance is that the essence of QCD can be portrayed, without severe distortion, in the few simple pictures at the bottom of the box on the next page. But first, for comparison, let me remind you that the essence of quantum electrodynamics (QED), which is a generation older than QCD, can be portrayed by the single picture at the top of the box, which represents the interaction vertex at which a photon responds to the presence or motion of electric charge.² This is not just a metaphor.

²¹ [QCD Made Simple]

Quite definite and precise algorithms for calculating physical processes are attached to the Feynman graphs of QED, constructed by connecting just such interaction vertices.

In the same pictorial language, QCD appears as an expanded version of QED. Whereas in QED there is just one kind of charge, QCD has three different kinds of charge, labeled by “color.” Avoiding chauvinism, we might choose red, green, and blue. But, of course, the color charges of QCD have nothing to do with physical colors. Rather, they have properties analogous to electric charge. In particular, the color charges are conserved in all physical processes, and there are photon-like massless particles, called color gluons, that respond in appropriate ways to the presence or motion of color charge, very similar to the way photons respond to electric charge.

QED and QCD in Pictures.



QED


The physical content of quantum electrodynamics is summarized in the algorithm that associates a probability amplitude with each of its Feynman graphs, depicting a possible process in spacetime. The Feynman graphs are constructed by linking together interaction vertices of the type at left, which represents a point charged particle (lepton or quark) radiating a photon. To get the amplitude, one multiplies together a kinematic “propagator” factor for each line and an interaction factor for each vertex. Reversing a line’s direction is equivalent to replacing a particle by its antiparticle.

Quantum chromodynamics can be similarly summarized, but with a more elaborate set of ingredients and vertices, as shown below. Quarks (antiquarks) carry one positive (negative) unit of color charge. Linear superpositions of the 9 possible combinations of gluon colors shown below form an SU(3) octet of 8 physical gluon types.

A qualitatively new feature of QCD is that there are vertices describing direct interactions of color gluons with one another. Photons, by contrast, couple only to electric charge, of which they carry none themselves.


QCD

Quarks

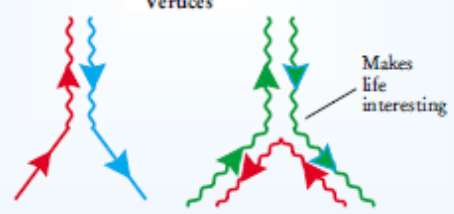


3 colors
6 flavors
(u, d, s, c, b, t)

Gluons



Vertices



Makes life interesting

53.6 Monte Carlo Techniques²²

Monte Carlo (MC) technique is a numerical method that makes use of random numbers to solve mathematical problems for which an analytical solution is not known. The first article, “The Monte Carlo Method” by Metropolis and Ulam, has appeared for the first time in 1949, even though well before that certain statistical problems were solved using random numbers. Since the simulation of random numbers is very time consuming, MC has become practical only with the advent of computers.

53.6.1 A simple MC simulation is the determination of π

Suppose we have a circle with radius $r = 1$ inscribed within a square. Then the area ratio is:

$$\frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \quad (1)$$

To determine we randomly select n points, $p_i = (x_i, y_i)$ for $i = 1 \dots n$, in the square and approximate the area ratio by:

$$\frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{m}{n} \quad (2)$$

where m is the number of random points in the circle, they must satisfy $x_i^2 + y_i^2 \leq 1$. Hence $\pi = 4 \times m/n$.

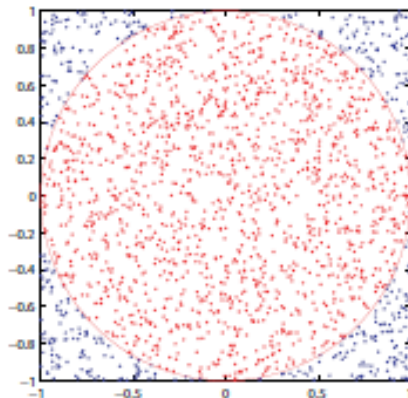


Figure 1: Compute π

This method is called hit-or-miss Monte Carlo since the estimate is computed as the actual ratio of hits to random tries. It is the least efficient MC method.

²² [Tutorial on Monte Carlo Techniques]

54 Contribution: The Particle-In-Cell simulation of plasmas: 1D

54.1 Introduction

In this chapter we will present the programs that we tried to use to study the simulation of a laser-plasma interaction. In the beginning, we preferred the C++ language. Unfortunately, the problem was that the programs operating in the C++ language need a very high-performance computer that was not available in our center. In the previous chapter (section 2.5), we presented a PIC model implemented in a C++ program (Smilei). Unfortunately, I could not install and compile this numerical program on our computers. A special environment is required for this task. Therefore, we decided to move to other languages, and the option was the ESPIC code written in Python language. The choice for this program in this language was because it saves the time of calculation and does not consume computer memory. In the following paragraph, I will present the program that I chose to display the PIC results. To achieve this goal, I learned the Python language and downloaded the necessary programs to run the code.

54.2 ESPIC

In this section we present the Python code ESPIC that we use to simulate the behaviour of plasma in a finite one-dimensional grid. First, we present the general structure of this code as well as the important parameter that should be initialised by the user. Next, we present some results obtained by ESPIC with a brief discussion.

I. Core routines

ESPIC is a simple 1D1V electrostatic PIC code [5]. The code is written in the object-oriented language Python. The necessary libraries to run this code are:

- **NumPy [6]**: provides a basic tool to manipulate arrays.
- **PyLab [7]**: allows to plot results during the simulation.

Both, *NumPy* and *PyLab* are open-source Python libraries and can be installed easily on any operating system.

The main routines of ESPIC are:

- **loadx**: This routine allows to initialize the positions of the particles onto a finite one-dimensional grid. In addition, *loadx* supports two type of boundaries: Reflective boundaries and periodic boundaries.

- **Loadv:** This routine is used to initialize the particle velocities v_i . By default, the plasma is considered as a cold plasma:

$$v_i=0. \quad (3.1)$$

The user can also set up a thermal distribution. In this case, the velocities are chosen in term of a random variable θ_i such that:

$$v_i = v_0 \sqrt{-2 \log \left(\frac{i+0,5}{N_{part}} \right)} \sin(\theta_i) \quad \text{For } i = 0, 1, \dots, N_{part}-1 \quad (3.2)$$

Where N_{part} number of particles and v_0 is an input parameter.

- **Density:** the electron density is obtained by the positions of electrons on the grid. The ion density is considered to be fixed during the simulation. This approximation is justified by the big difference between the mass of ions and electrons. Compared to electrons, the ions can be considered at rest.
- **Field:** the electrostatic field is calculated by this “field” routine and the magnetic field is neglected in this code. In this case, the electric field is given by the integration over the well-known Gauss’s law:

$$\nabla \cdot E = \frac{dE}{dx} = \frac{\rho}{\epsilon_0} \quad (3.3)$$

The integration is performed by using the trapezoidal approximation:

$$\int_a^b f(x) dx \approx \frac{(b-a)}{2} (f(b) - f(a)) \quad (3.4)$$

Which provides a reasonable accuracy if $|b - a| \ll 1$.

- **Push:** Once the electric field is computed on the grid, the routine “push” interpolates this field from grid to particle, and then the new velocities of particles are obtained by solving the equation:

$$V_i = v_i + \frac{q}{m} E_{interp} \Delta t, \quad (3.5)$$

With V_i the new velocity, v_i the precedent velocity, q and m are the charge and the mass of particles respectively, E_{interp} is the field after interpolation and Δt is the time step of the simulation. Next, the positions x_i are updated by

$$X_i = x_i + \Delta t V_i. \quad (3.6)$$

- **Particle_bc:** This routine is used after every iteration in order to verify that the boundaries conditions are not violated.
- **Diagnostics:** in this part, the electric field, the density, the distribution function ($f(v)$) and the phase space ($v(x)$) are plotted every time step. These plots are printed and saved (in *png* extension). Obviously, the number of plots can be chosen easily by the user as well as the time step and the duration of calculation.
- **Histories:** in the last routine, we check if the energy conservation is violated during the simulation. In fact, the kinetic energy

$$K = \frac{1}{2}m \sum_i v_i^2 , \quad (3.7)$$

and the potential energy

$$U = \frac{1}{2}\Delta x \sum_j E_j^2 , \quad (3.8)$$

as well as the total energy

$$E = K + U , \quad (3.9)$$

are saved in the file energies.out and are plotted in energies.png.

II. Loop over time of simulation

The routines explained above are called in a closed loop over the time of simulation as follows:

- Call **loadx** to load particles onto grid (to initialize x).
- Call **loadv** to define velocity distribution (to initialize v).
- Centre positions for the first leap-frog step:

$$x \leftarrow x + \frac{1}{2} dtv. \quad (3.10)$$

- Call **particle_bc** to apply the boundary conditions.
- Call **field** to compute the electric field.
- Call **diagnostics** to show the results after this initialization.
- Start the main iteration loop
 - Call **push** to update x and v .
 - Call **particle_bc** to check boundaries.
 - Call **density** to compute the new density.
 - Call **field** to find the new field.
 - Call **diagnostics** to plot and save figures of electric field, density, the distribution function and $v(x)$.
- At the end, call histories to check the total energy conservation during the simulation.

III. Important parameters.

Here, we present briefly the principal input (parameters) that should be provided by the user.

- The number of particles: *npart*.
- The number of grid points: *ngrid*.
- The number of steps (over time of simulation): *nsteps*.
- The length of the grid: *grid_length*.
- The time step: *dt*.
- The type of the field boundary conditions: *bc_field* = 1 for periodic boundaries and *bc_field* = 2 for reflective boundaries.
- The type of the particle boundary conditions: *bc_particle* (similarly to *bc_field*).

54.3 Results

54.3.1 Results without **B**

In this section, we present and discuss some results obtained for a cold plasma without a magnetic field **B**, with the initial parameters as follows:

- *npart* = 10000.
- *ngrid* = 100.
- *nsteps* = 100.
- *grid_length* = 2p.
- *dt* = 0.05.
- *bc_field* = *bc_particle* = 1.

The initialization is illustrated in the (figure 3.1). We show in this figure the density as a function of *x*. The positive maximum indicates the centre of distribution of ions, and the negative minimum shows the centre of distribution of electrons. As we can see, in the two cases (ions and electrons), the particles are distributed around the corresponding centre of concentration with a normal (Gaussian) distribution. The user can easily modify this choice of distribution by modifying the routine *loadx*. We can see also a small fluctuation which has been added also in *loadx*. We show also the electric field in function of *x*. As expected, when the density is positive, the field is decreasing and when it is negative, the field is increasing. This observation is predictable since we know that the derivation of *E* is proportional to $-\rho$. We see also that $E(x = 0) = E(x = 2p)$, which corresponds to the periodic boundaries choice. Since the plasma was taken as cold plasma, the velocities are very small. Finally, we have also plotted the distribution function $f(v)$.

In order to show the evolution of plasma simulation, we plot the results for $t_{33} = 33dt$, $t_{66} = 66dt$ and $t_{99} = 99dt$ in figure 3.2, figure 3.3 and figure 3.4, respectively. The quality of the curve of $f(v)$ could be ameliorated by increasing the number of grid points.

In the figure 3.5, we show the evolution of potential, kinetic and total energies during the simulation. Of course, the maximum of kinetic energy corresponds to a minimum of potential energy and vice-versa.

However, the total energy is not perfectly a horizontal straight line. An important reason for these fluctuations is the integration of Gauss theorem with finite development up to the first term.

Nevertheless, the fluctuation of the total energy is relatively small and the global accuracy of the model is very reasonable, especially if we consider the low computational cost needed to run the code.

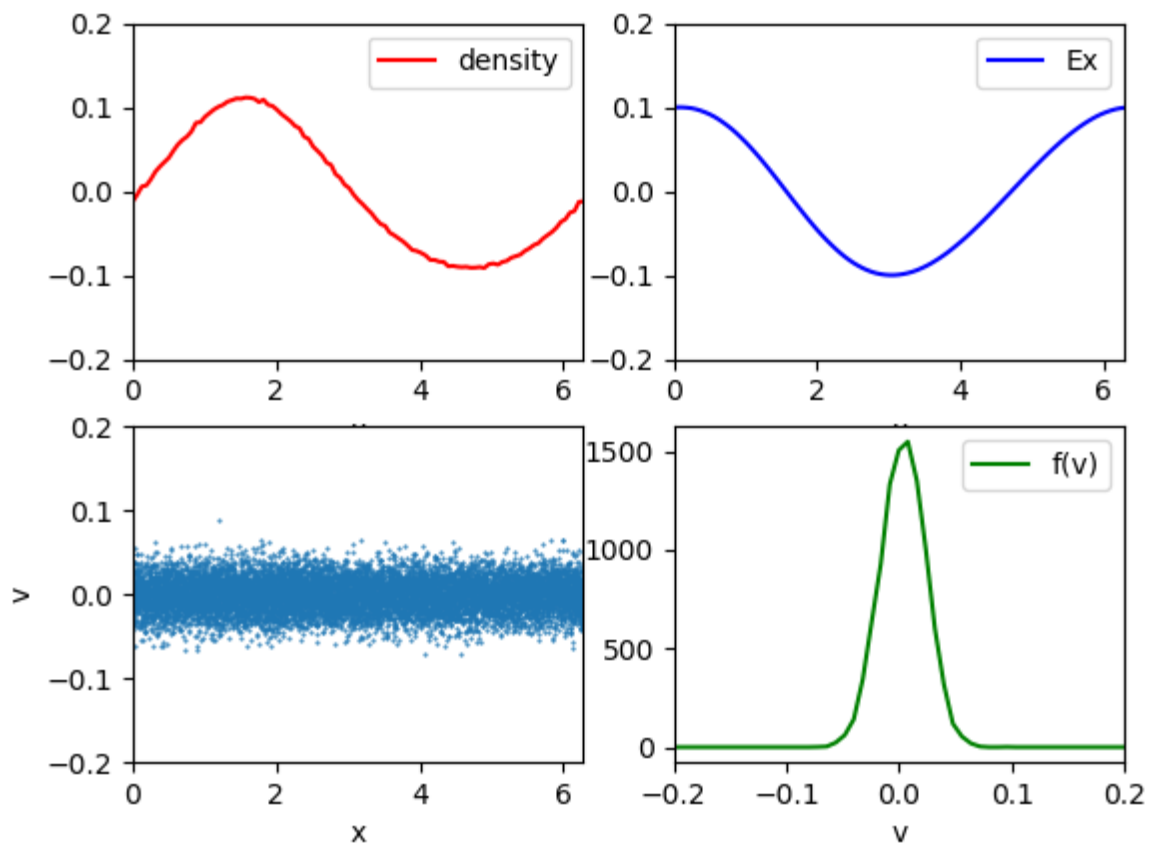


FIGURE 3.1: Results obtained at t_0 . Top left: the density of charges as a function of x (m). Top right: the electric field E_x (v/m). Bottom left: the phase space v_x (m/s). Bottom right: the distribution function $f(v)$.

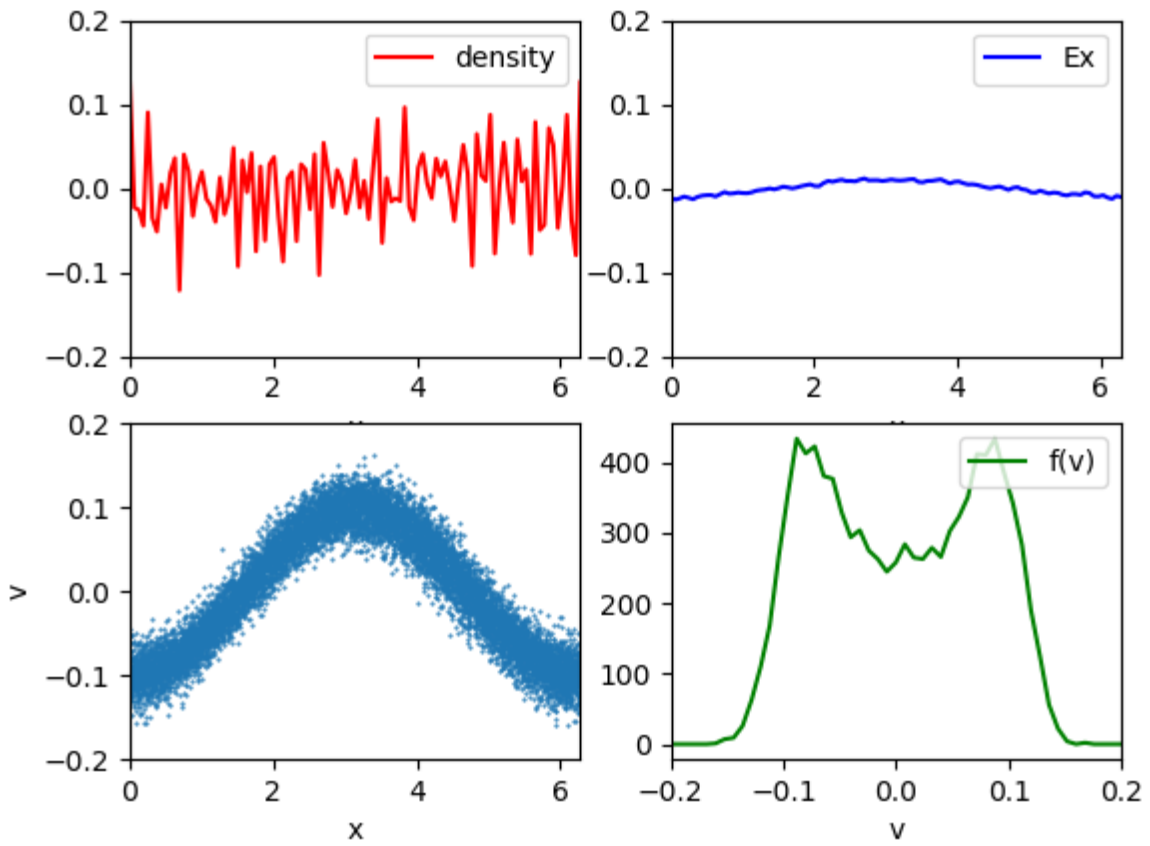


FIGURE 3.2: Similarly to figure 3.1 for $t_{33} = 33dt$.

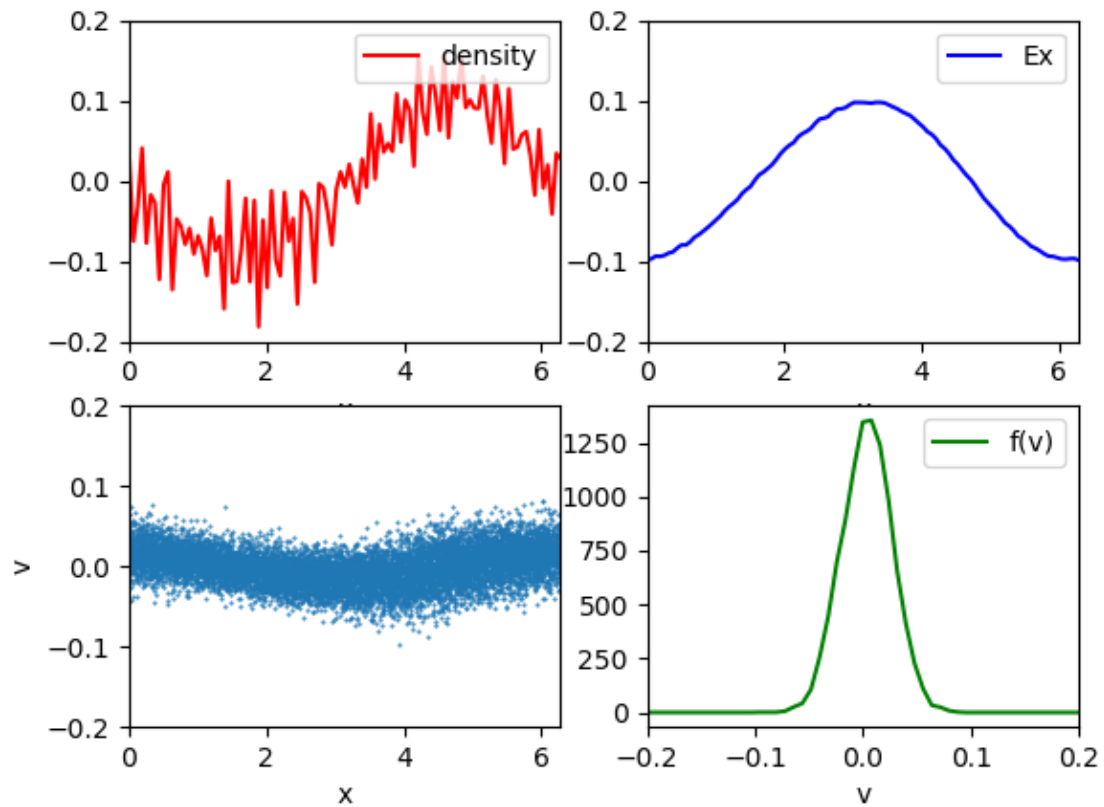


FIGURE 3.3: Similarly to figure 1 for $t_{66} = 99dt$.

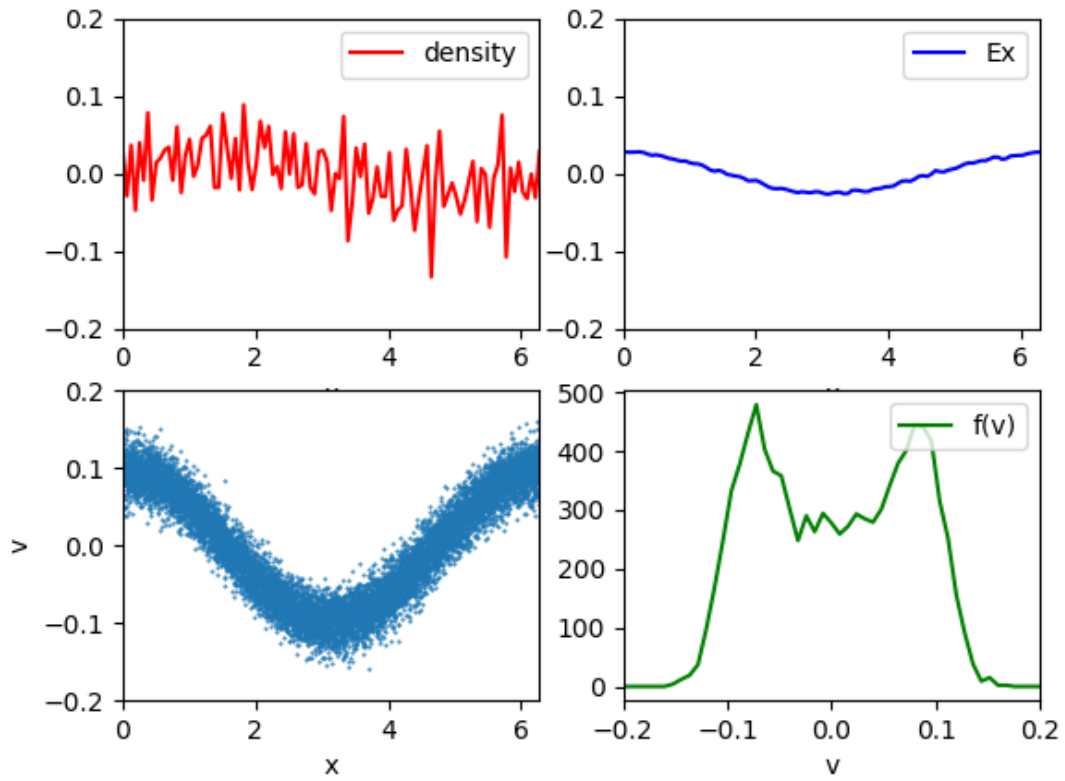


FIGURE 3.4: Similarly to figure 1 for $t_{99} = 99dt$ (end of the simulation).

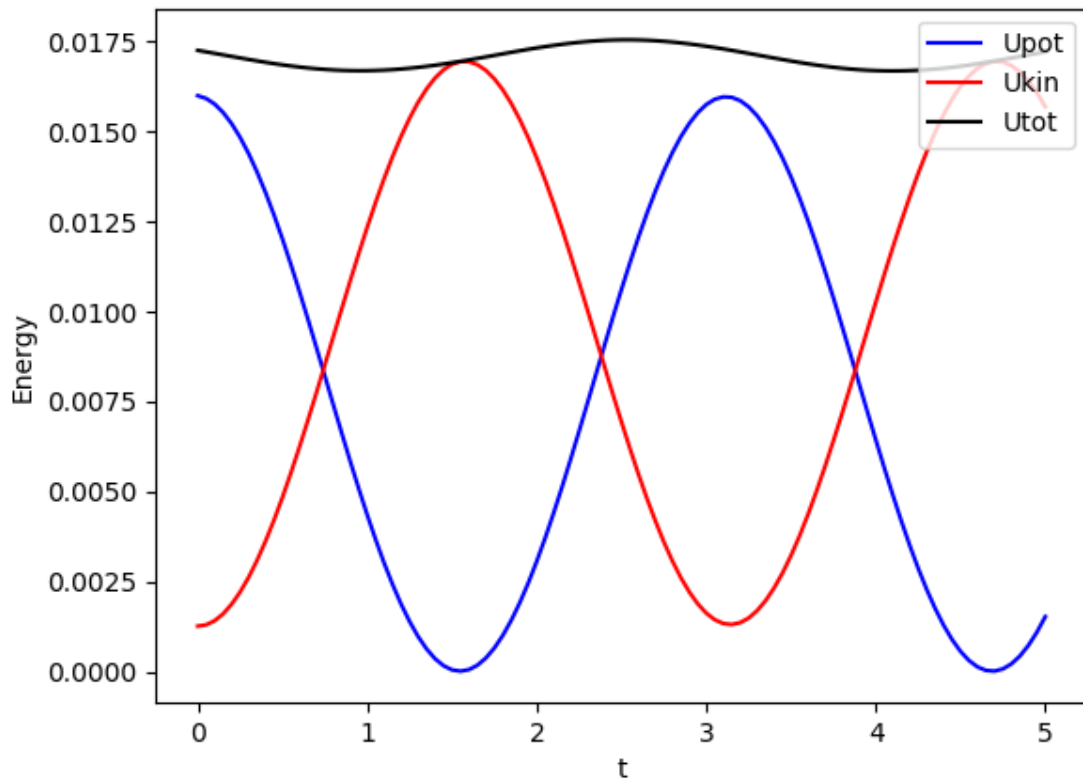


FIGURE 3.5: The evolution of potential (U_{pot} (J)), kinetic (U_{kin} (J)) and total (U_{tot} (J)) energies during the simulation.

54.3.2 Results with B

The existence of the magnetic field \mathbf{B} will cause some changes in the results: for example, because of the particular shape of the magnetic field function, and the relation between \mathbf{E} and \mathbf{B} equ (2.10c), we expect that the modification in the electric field will be approximately a simple shift in the amplitude, and some perturbations will occur in the distribution of particles. On the other hand, the addition of an external source of energy will increase the total energy of the plasma gases.

We have made some modifications to the code and added an external magnetic field, and displayed the results as in the (figure 3.6) and (figure 3.7). To achieve this step, I assumed the existence of a magnetic field with the following function:

$$\mathbf{B} = B_0 \mu_0 \varepsilon_0 \gamma \vec{u}_z \quad (3.11)$$

Where \mathbf{B} External magnetic field, and $B_0=0.05$ v/m.s.

The first thing that we can notice after adding the magnetic field is the big change in energy. The energy is no longer conserved as it appears in the picture (figure 3.5). The energy has become increasing (figure 3.7). This is because the plasma is exposed to an external energy source, which is the magnetic field.

As for the electric field E_x , it appears that it was subjected to a shifting proportional to the value of B_0 . (Figure 3.6).

With regard to the density and the distribution function, we see that it has changed somewhat due to the interaction of the magnetic field with the charged particles, and this result was expected in (figure 3.6).

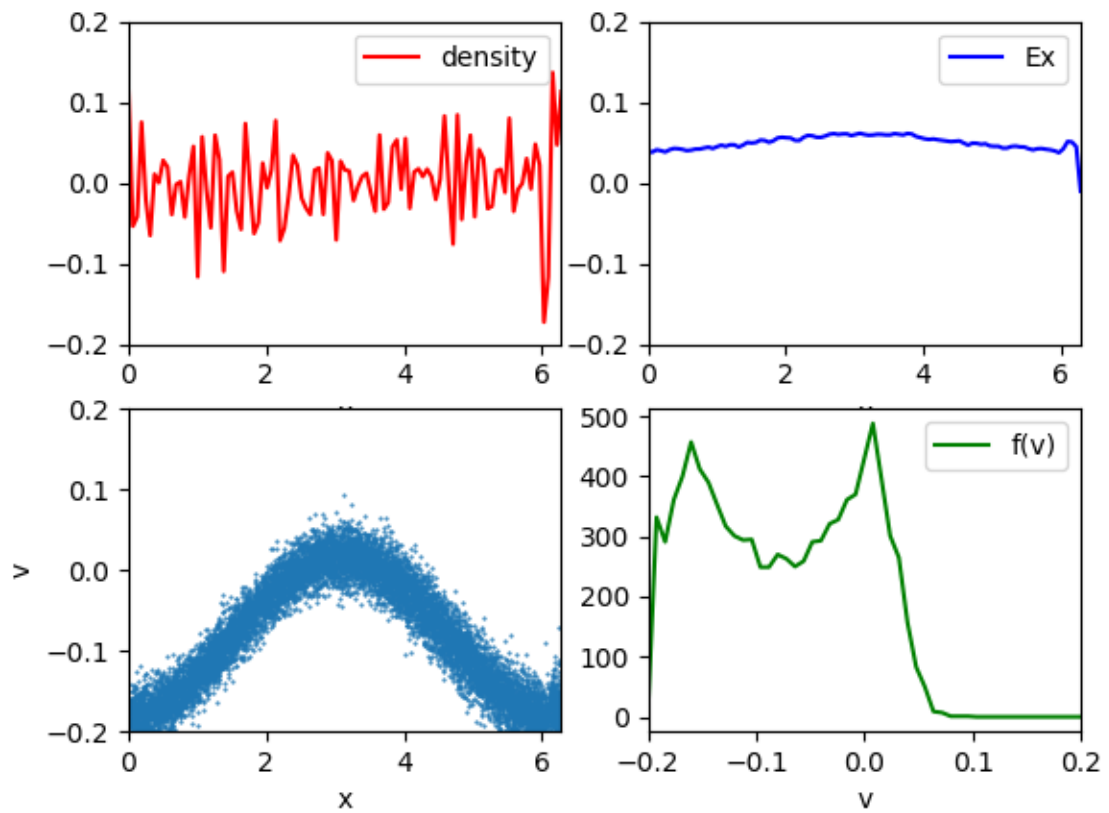


FIGURE 3.6: Similarly to figure 3.2 for $t_{33} = 33dt$, after the plasma is exposed to an external magnetic field.

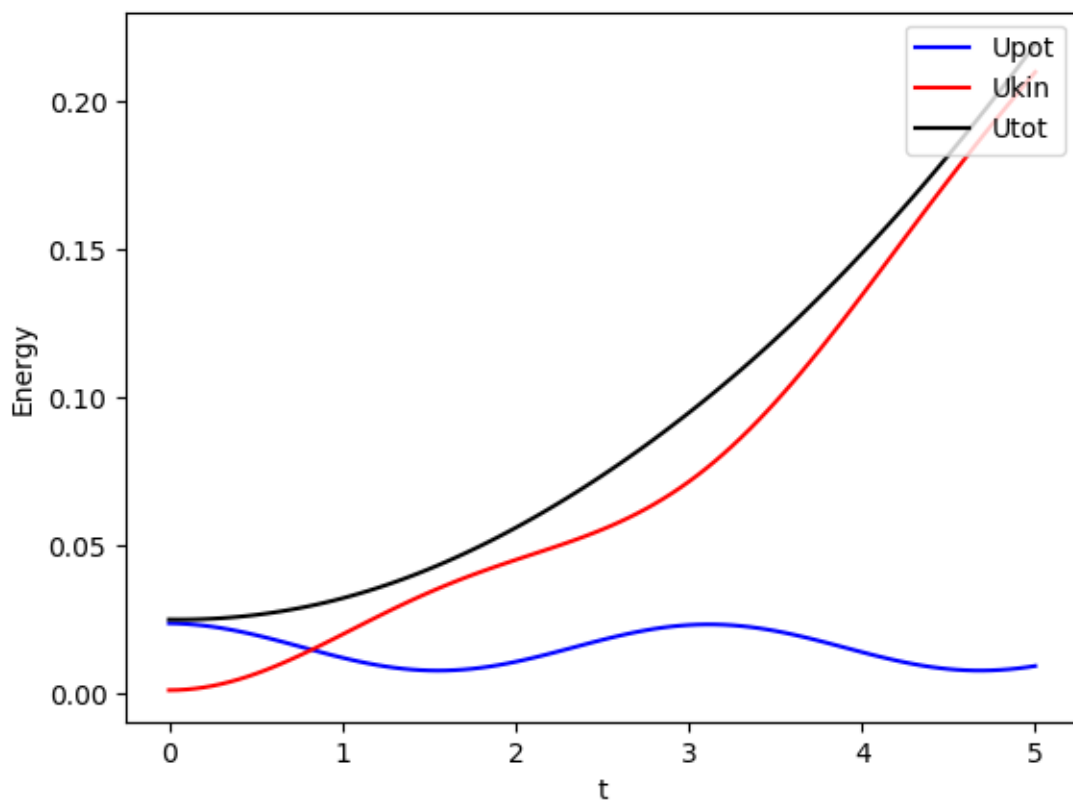


FIGURE 3.7: The evolution of potential (U_{pot} (J)), kinetic (U_{kin} (J)) and total (U_{tot} (J)) energies during the simulation t (s), after the plasma is exposed to an external magnetic field.

54.4 Conclusion

In this chapter, we present the structure of a Particle in Cell code in one dimension. We present few results obtained to illustrate the performance of this code. Also, we added a modification to the code to simulate the interaction of an external magnetic field with the plasma, the results were as we expected. Then we presented the results of this interaction and we analyzed and compared them with the results of simulating the plasma without an external magnetic field.

The advantage of this code is the possibility to simulate the behavior of a big number of particles with a very low computationally cost. And, on the other hand, in this code the magnetic field is neglected and the discretization of electric field equation is performed up to first order.

In the next chapter, we will present a more general and more realistic code that simulates a plasma in three dimensions and takes into account the magnetic field resulting from the movement of charged particles.

55 Contribution: Laser-Matter Interaction in IAP-PSC code

This chapter is a correction and development of the previous program created in AECENAR center, which is a simulation code for the interaction of plasma particles. As for correcting and improving the code, this development was based on the result of the study that we explained in the previous chapters of this thesis.

55.1 Improvement in IAP-PSC code without laser-matter interaction

This master thesis is a project development for IAP-PSC Plasma Simulation Code (Particle-in-Cell Code). The Plasma Simulation Code (PSC) is developed as an open source software. It is the intent of the project to provide a state-of-the-art simulation code for education and research in the field of intense laser-matter interaction [8].

In this project, write a code was supposed to accomplish tasks similar to that of the PSC code (simulations of the interaction of plasma's particles). However, the existing code had some errors that appeared in the results (fig 4.1). This matter is expected because the code should processes 6^{1000} equations (example used) and in addition, it contains many other errors.

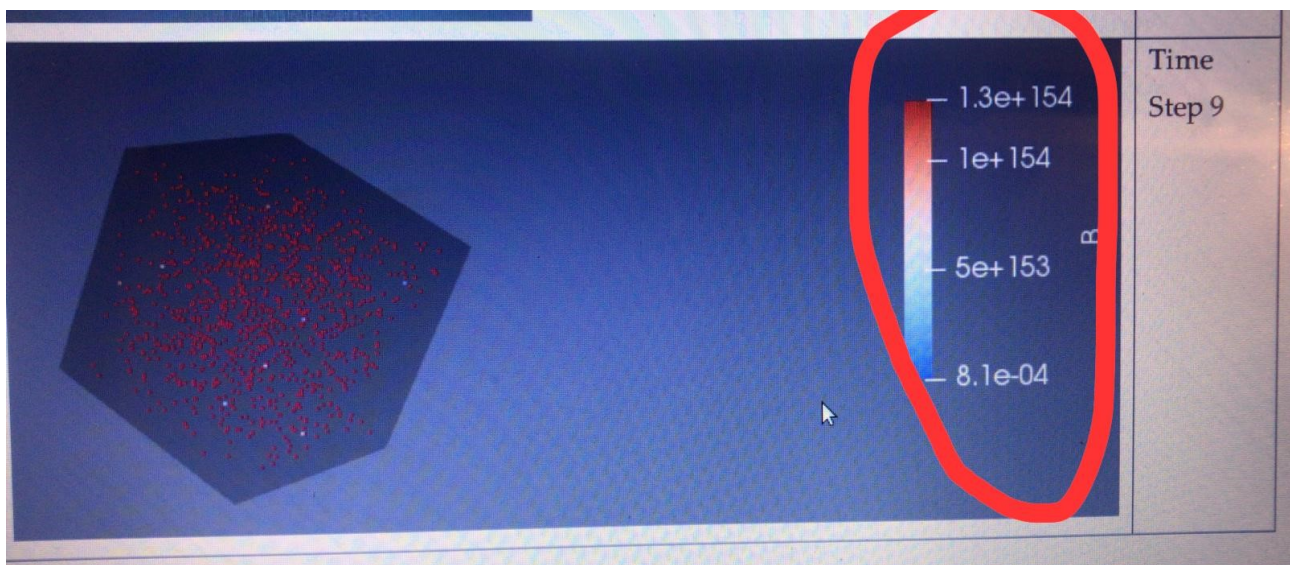


Figure 4.1: The magnetic field in order of 10^{154} ! [10]

Before starting to study the interaction laser-plasma with relativistic approach, the program of IAP PSC code must be corrected, and this is what we will achieve in this chapter.

The improvement will be a series of consecutive steps

1-Gridding

2-write the equations (Maxwell-Vlasov) after that we will discrete the equations.

3- Solve the equations and we will develop at this point the initialization parameters and boundary conditions

4-Visualization.

Upon reaching the third stage we will have to get the correct results, then we move to the fourth stage.

55.2 Modified code

We have reformed nearly 95% of the code, and optimized it to produce acceptable results.

I will briefly mention these reforms in the form of points.

Noting that the class names remain the same, out of respect for those who started writing the code before me. However, the content of the new classes written in the code presented in this chapter is completely different, in addition, the results are different.

- New algorithm: initialization (X, Y, Z, V_x, V_y, V_z) of Elec and ions=>Densities ρ ,
 $\rho = \rho_e + \rho_i \Rightarrow$ fields (E and B) \Rightarrow (new X, Y, Z, V_x, V_y, V_z).
- Corrected:
 - Initialization.
 - Grid (Yee grid).
 - Over floating.
 - Respect equation of continuity.
 - Respect boundary condition (periodic boundary).
- Results:
 - The disappearance of divergence.
 - Reasonable results (1D).

55.3 UML Diagrams

IAP-PSC is a C++ code of 4 classes, which solves the Maxwell equations in the presence of a gas of charged particles (plasma).

As shown in class diagram 4.2, these four classes are:

1. *parameter*,
2. *dens_currnt*,
3. *maxwell_equat*,
4. *pos_veloct*.

The class “parameter” is used for the initialization of parameters. In the class “maxwell_equat”, we solve Maxwell's equations to get the magnetic and the electric field. We use the class “dens_currnt” to obtain the density and the current. The class “pos_veloct” allows to compute the position and the velocity of particles. An illustration of the code and the classes is given in the sequence diagram 4.3

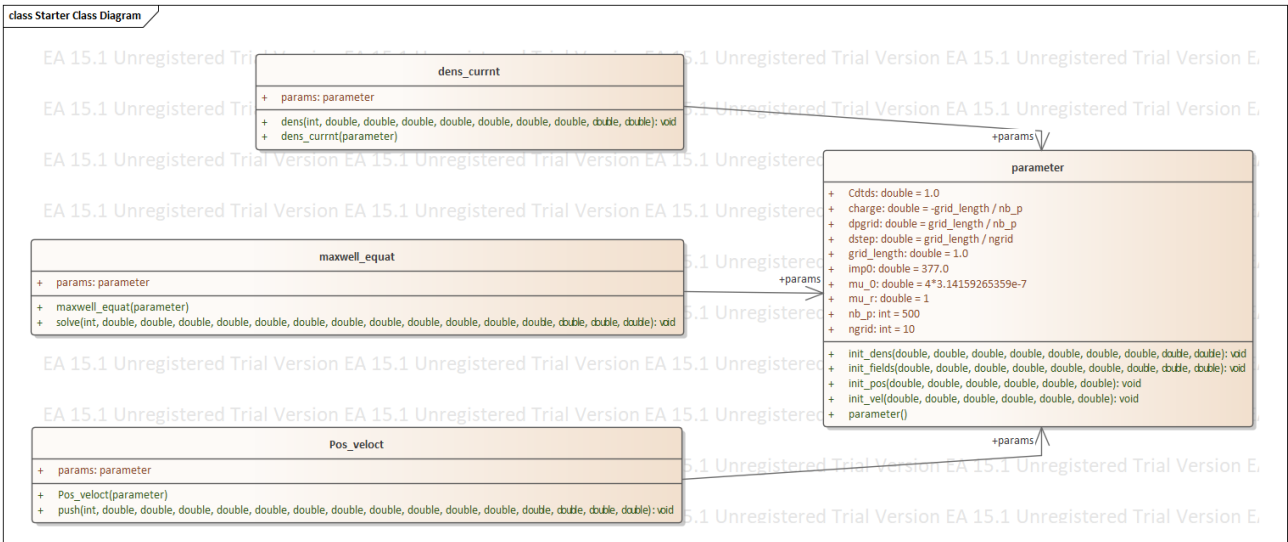


Diagram 4.2: Class diagram

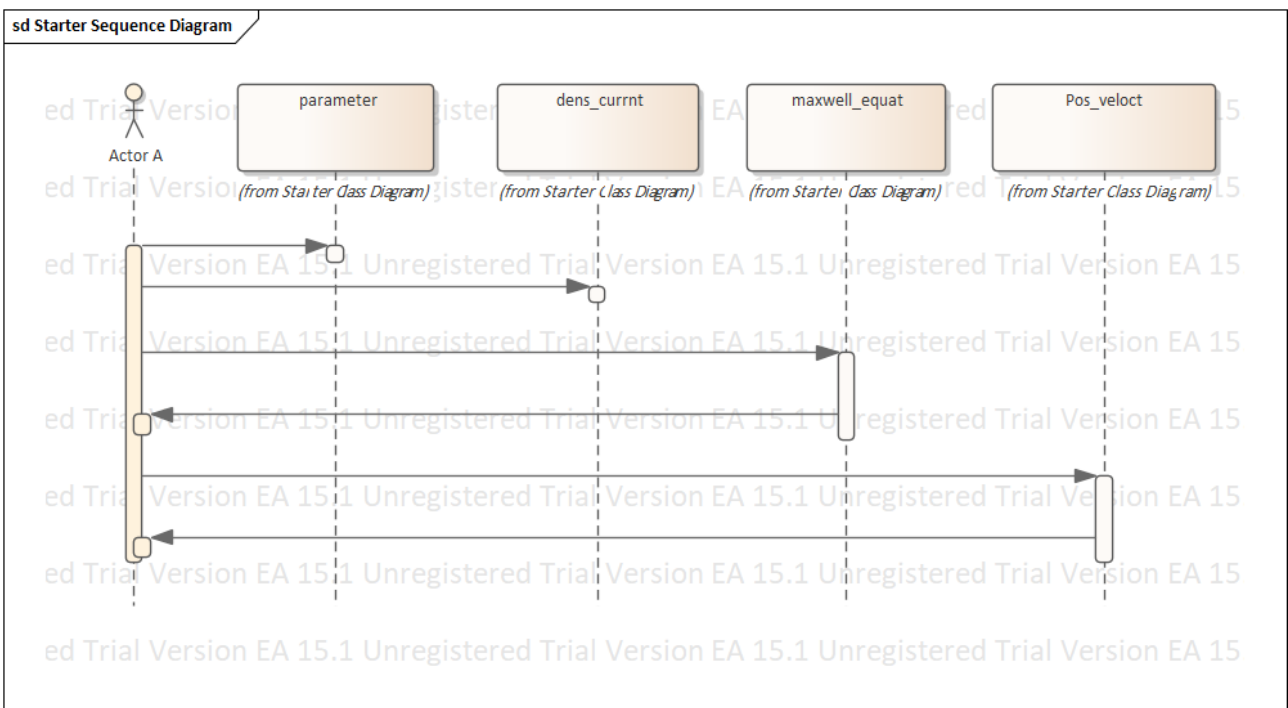


Diagram 4.3: Sequence diagram

55.4 Simulation in one dimension

We are not able to run our code in the case of 3d simulation because of the huge memory needed to save the fields E and B. However, In order to make a test for our code, we consider the 1d case and we performed a simulation similar to that discussed in the chapter 3 of Schneider [9].

The simulation parameters are fixed as those chosen in the program 3.1 page 41 (Schneider...) [9] except the size of the distance box of simulation (100 spatial steps rather than 200).

We should emphasis again that the goal here is only to validate the code in the case of 1d, since we do not have the require cluster to compile the code in the 3d case.

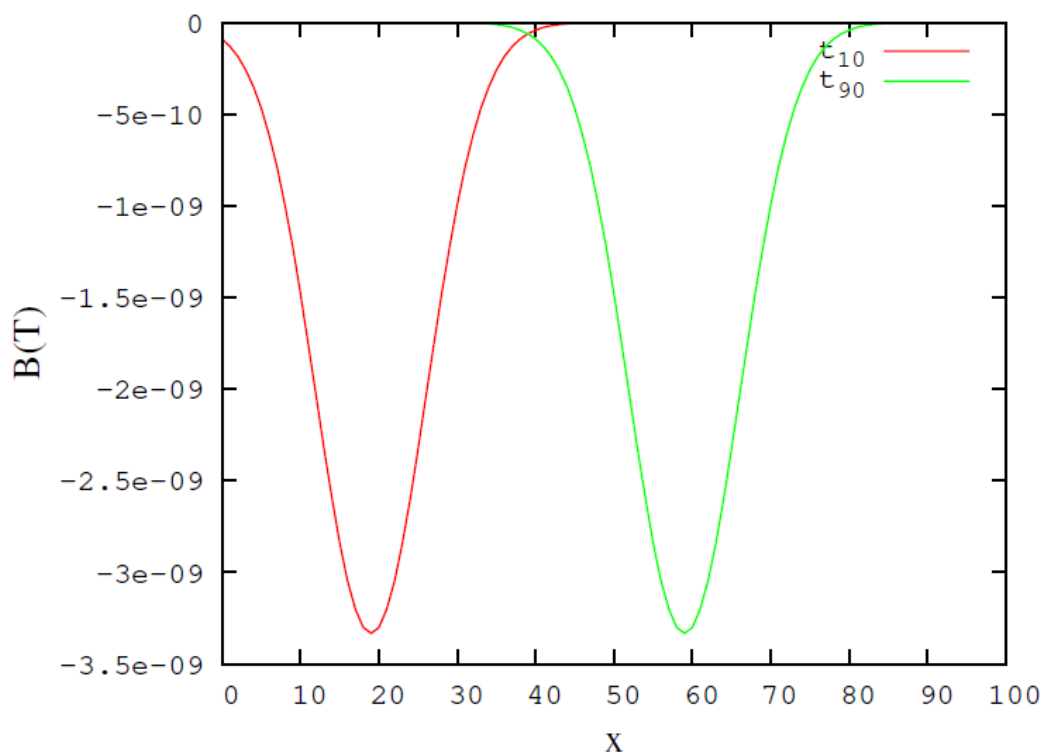


Figure 4.4: Results obtained at t_{10} and t_{90} . Red line the magnetic field $B(x)$ at t_{10} . Green line the magnetic field $B(x)$ at t_{90} . It's generated by Bare-bones one-dimensional simulation with a hard source.

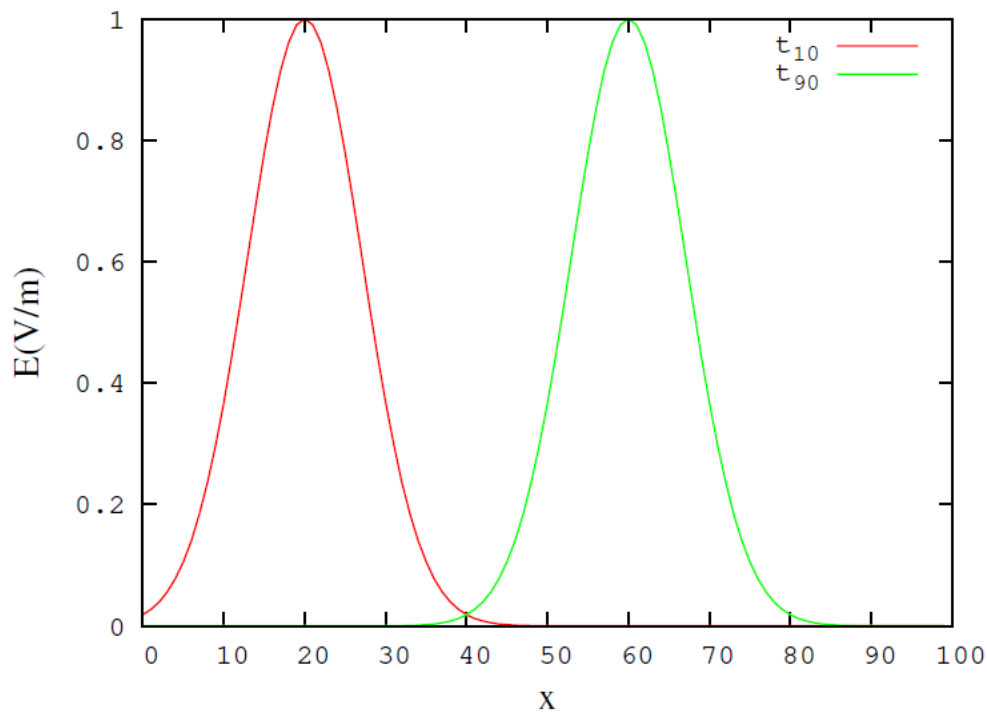


Figure 4.5: Results obtained at t_{10} and t_{90} . Red line the electric field $E(x)$ at t_{10} . Green line the electric field $E(x)$ at t_{90} . It's generated by Bare-bones one-dimensional simulation with a hard source.

55.5 Simulation in three dimensions

In the case of the three dimensions, the matter was different, the memory is full due to the increased number of processed equations, which forced us to reduce the number of points in the discretisation. Hence the results were not as suspected. To solve this problem we need a supercomputer, which is not available. So we can not display the results in the three dimensions.

55.6 Conclusion

In this chapter we began by presenting the old code's problems, and its problems' results. Then we wrote a C++ code that can simulate the solution of Maxwell's equations in the material in an easy model. Then, we took into account the magnetic field inside the charged gas (plasma). And we explained in a simple way the equations used in the code. And we presented the results in the case of one dimension, then mentioned the problem in the case of the three dimensions.

56 BIBLIOGRAPHY

- [1] Lee, Patrick. Modelling of a laser-plasma injector for multi-stage acceleration. Diss. 2017.
- [2] Lontano, Maurizio, et al. "A kinetic model for the one-dimensional electromagnetic solitons in an isothermal plasma." *Physics of Plasmas* 9.6 (2002): 2562-2568.
- [3] Gizzi, Leonida Antonio. "Laser-Driven Sources of High Energy Particles and Radiation." *Laser-Driven Sources of High Energy Particles and Radiation*. Springer, Cham, 2019. 1-24.
- [4] Derouillat, Julien, et al. "Smilei: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation." *Computer Physics Communications* 222 (2018): 351-373.
- [5] P. Gibbon, KU-Leuven and FZ-Juelich, November 2013.
- [6] McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. "O'Reilly Media, Inc.", 2012.
- [7] Ranjani, J., A. Sheela, and K. Pandi Meena. "Combination of NumPy, SciPy and Matplotlib/PyLab-a good alternative methodology to MATLAB-A Comparative analysis." 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT). IEEE, 2019.
- [8] Ruhl, Hartmut. "Classical Particle Simulations with the PSC code." Ruhr-Universität Bochum (2005).
- [9] John B. Schneider, *Understanding the Finite-Difference Time-Domain Method*, sect 3, July 6, 2020.
- [10] Mariam Abdel-Karim, Editor: Samir Mourad, *IAP-PSC Plasma Simulation Code (Particle-in-Cell Code)*, 2019, <http://aecenar.com/index.php/downloads/send/10-iap/496-iap-psc-plasma-simulation-code-pdf>

57 LIST OF SYMBOLS

Electron rest mass	m_e	$9.109 \times 10^{-31} \text{ kg}$
Electronic charge	e	$1.6022 \times 10^{-19} \text{ C}$
Speed of light in free space	c	$2.9979 \times 10^8 \text{ m s}^{-1}$
Permeability of free space	μ_0	$4\pi \times 10^{-7} \text{ H m}^{-1}$
Permittivity of free space	ϵ_0	$8.854 \times 10^{-12} \text{ F m}^{-1}$
Boltzmann's constant	k_B	$1.3807 \times 10^{-23} \text{ J K}^{-1}$


```

//
double rhox[10+1][10];
double rhoy[10+1][10];
double rhoz[10+1][10];
//
double gridx[10] ;
double gridy[10] ;
double gridz[10] ;
//
cout<<"The program has start\n";
//
// field's grid (uniform)
//
    for(int mm = 0 ; mm < param.ngrid ; mm++){
        gridx[mm] = mm*param.dstep ;
        gridy[mm] = mm*param.dstep ;
        gridz[mm] = mm*param.dstep ;
    }
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// parameter initialization
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
param.init_pos(xe, ye, ze, xi, yi, zi);
param.init_vel(vex, vey, vez, vix, viy, viz);
param.init_dens(rhox, rhoy, rhoz, xe, ye, ze, xi, yi, zi);
double upd_Ex[10][10][10], upd_Ey[10][10][10], upd_Ez[10][10][10] ;
double upd_Bx[10][10][10], upd_By[10][10][10], upd_Bz[10][10][10] ;
double Ex_save[10][10][10], Ey_save[10][10][10], Ez_save[10][10][10] ;
double Bx_save[10][10][10], By_save[10][10][10], Bz_save[10][10][10] ;
param.init_fields(upd_Ex, upd_Ey, upd_Ez, upd_Bx, upd_By, upd_Bz, rhox,
rhoy, rhoz);
//
// save fields
//
for(int mm = 0 ; mm < param.ngrid ; mm++){
    for(int nn = 0 ; nn < param.ngrid ; nn++){
        for(int pp=0 ; pp < param.ngrid ; pp++){
            Ex_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Ey_save[mm][nn][pp] = upd_Ey[mm][nn][pp] ;
            Ez_save[mm][nn][pp] = upd_Ez[mm][nn][pp] ;
            Bx_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            By_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Bz_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
        }
    }
}
//
// total electric field
//
for(int mm = 0 ; mm < param.ngrid ; mm++){
    for(int nn = 0 ; nn < param.ngrid ; nn++){
        for(int pp=0 ; pp < param.ngrid ; pp++){
            E[mm][nn][pp] = upd_Ex[mm][nn][pp] + upd_Ey[mm][nn][pp] +
upd_Ez[mm][nn][pp] ;
        }
    }
}
//

```

```

// total magnetix field
//
    for(int mm = 0 ; mm < param.ngrid ; mm++){
        for(int nn = 0 ; nn < param.ngrid ; nn++){
            for(int pp = 0 ; pp < param.ngrid ; pp++){
                B[mm][nn][pp] = upd_Bx[mm][nn][pp] + upd_By[mm][nn][pp] +
upd_Bz[mm][nn][pp] ;
            }
        }
    }
//
// output results
//
string filename = "PSC_E_0.csv" ;
ofstream myfile ;
// fields E and B on the grid
myfile.open(filename.c_str());
myfile<<"x,y,z,E,B\n";
    for(int mm = 0 ; mm < param.ngrid ; mm++){
        for(int nn = 0 ; nn < param.ngrid ; nn++){
            for(int pp = 0 ; pp < param.ngrid ; pp++){
                myfile << gridx[mm] << "," << gridy[nn] << "," << gridz[pp] <<
", "
                << E[mm][nn][pp] << "," << B[mm][nn][pp] << "\n" ;
            }
        }
    }
myfile.close() ;
// distributions
double vv ;
filename= "PSC_dist_0.csv" ;
myfile.open(filename.c_str());
myfile<<"x,y,z,v\n";
for(int i = 0 ; i < param.nb_p ; i++){
    // electrons
    vv = sqrt( vex[i][0]*vex[i][0] + vey[i][0]*vey[i][0] +
vez[i][0]*vez[i][0] ) ;
    myfile << xe[i][0] << "," << ye[i][0] << "," << ze[i][0] << "," << vv <<
"\n";
    // ions
    vv = sqrt( vix[i][0]*vix[i][0] + viy[i][0]*viy[i][0] +
viz[i][0]*viz[i][0] ) ;
    myfile << xi[i][0] << "," << yi[i][0] << "," << zi[i][0] << "," << vv <<
"\n";
}
myfile.close() ;
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Loop over time
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
for(int t = 1 ; t < maxTime ; t++){
    //
    // Push position and velocity
    //
    Pos_veloct vel = Pos_veloct(param) ;
    vel.push(t, xe, ye, ze, vex, vey, vez, xi, yi, zi, vix, viy, viz,
upd_Ex, upd_Ey, upd_Ez) ;
    //

```

```

// update density
//
dens_currnt densities = dens_currnt(param);
densities.dens(t, xe, ye, ze, xi, yi, zi, rhox, rhoy, rhoz) ;
//
// solve maxwell equations
//
for(int mm = 0 ; mm < param.ngrid ; mm++){
    for(int nn = 0 ; nn < param.ngrid ; nn++){
        for(int pp=0 ; pp < param.ngrid ; pp++){
            Ex_save[mm][nn][pp] = Ex_save[mm][nn][pp] + 1000;
        }
    }
}
maxwell_equat max = maxwell_equat(param);
max.solve(t, upd_Ex, upd_Ey, upd_Ez, upd_Bx, upd_By, upd_Bz, rhox, rhoy,
rhoz, Ex_save,Ey_save,Ez_save,Bx_save,By_save,Bz_save) ;
//
// save fields
//
for(int mm = 0 ; mm < param.ngrid ; mm++){
    for(int nn = 0 ; nn < param.ngrid ; nn++){
        for(int pp=0 ; pp < param.ngrid ; pp++){
            Ex_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Ey_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Ez_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Bx_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            By_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
            Bz_save[mm][nn][pp] = upd_Ex[mm][nn][pp] ;
        }
    }
}
//
// total electric field
//
for(int mm=0;mm<param.ngrid;mm++){
    for(int nn=0;nn<param.ngrid;nn++){
        for(int pp=0;pp<param.ngrid;pp++){
            E[mm][nn][pp] = upd_Ex[mm][nn][pp] + upd_Ey[mm][nn][pp] +
upd_Ez[mm][nn][pp] ;
        }
    }
}
//
// total magnetix field
//
for(int mm=0;mm<param.ngrid;mm++){
    for(int nn=0;nn<param.ngrid;nn++){
        for(int pp=0;pp<param.ngrid;pp++){
            B[mm][nn][pp] = upd_Bx[mm][nn][pp] + upd_By[mm][nn][pp] +
upd_Bz[mm][nn][pp] ;
        }
    }
}
//
// output results
//
string filename ;
ofstream myfile ;
stringstream d ;
double vv ;
d<<t;
// fields E and B on the grid

```

```

    filename= "PSC_E_"+ d.str();
    filename+= ".csv";
    myfile.open(filename.c_str());
    myfile<<"x,y,z,E,B\n";
        for(int mm=0;mm<param.ngrid;mm++){
            for(int nn=0;nn<param.ngrid;nn++){
                for(int pp=0;pp<param.ngrid;pp++){
                    myfile << gridx[mm] << "," << gridy[nn] << "," << gridz[pp]
<< ","
                    << E[mm][nn][pp] << "," << B[mm][nn][pp] << "\n" ;
                }
            }
        }
    myfile.close();
    // distributions
    filename= "PSC_dist_"+ d.str();
    filename+= ".csv";
    myfile.open(filename.c_str());
    myfile<<"x,y,z,v\n";
    for(int i=0;i<param.nb_p;i++){
        // electrons
        vv = sqrt( vex[i][t]*vex[i][t] + vey[i][t]*vey[i][t] +
vez[i][t]*vez[i][t] ) ;
        myfile << xe[i][t] << "," << ye[i][t] << "," << ze[i][t] << "," <<
vv << "\n";
        // ions
        vv = sqrt( vix[i][t]*vix[i][t] + viy[i][t]*viy[i][t] +
viz[i][t]*viz[i][t] ) ;
        myfile << xi[i][t] << "," << yi[i][t] << "," << zi[i][t] << "," <<
vv << "\n";
    }
    myfile.close();
    //
    //system("pause");
}
//return a.exec();
cout << "END \n" ;
return 0 ;
// End
}

```

Parameter

Parameter.h::

```

#ifndef PARAMETER_H
#define PARAMETER_H

//int maxTime = 50 ;
//int nb_p = 500 ;
//int ngrid = 50 ;

class parameter{
public:
    parameter(){}
    //
    void init_pos(double xe[500][10], double ye[500][10], double ze[500][10],
        double xi[500][10], double yi[500][10], double zi[500][10]) ;
    //
    void init_vel(double vex[500][10], double vey[500][10], double vez[500][10],
        double vix[500][10], double viy[500][10], double viz[500][10]) ;
    //
}

```



```
// double dpgrid = grid_length / nb_p ; //particle spacing
// double charge = -grid_length / nb_p ; //pseudo-particle charge normalised
to give ncrit=1
// double imp0 = 377.0 ;
// double CdtDs = 1.0 ; // Courant number
// double mu_r=1 ;
// double mu_0 = 4*3.14159265359e-7 ;
//
int b=0 ;
//
for(int l=0; l < nb_p; l++){
//
xe[l][0] = 0 + dpgrid * (l+0.5) ;
xe[l][0] = xe[l][0] + 0.1 * cos(xe[l][0]) ;
// periodic boundaries
if( xe[l][0] < 0 ){
xe[l][0] = xe[l][0] + grid_length ;
} else if( xe[l][0] >= grid_length ){
xe[l][0] = xe[l][0] - grid_length ;
}
//
b=rand()%nb_p ;
ye[l][0] = b * grid_length / nb_p ;
// periodic boundaries
if( ye[l][0] < 0 ){
ye[l][0] = ye[l][0] + grid_length ;
} else if( ye[l][0] >= grid_length ){
ye[l][0] = ye[l][0] - grid_length ;
}
//
b=rand()%nb_p ;
ze[l][0] = b * grid_length / nb_p ;
// periodic boundaries
if( ze[l][0] < 0 ){
ze[l][0] = ze[l][0] + grid_length ;
} else if( ze[l][0] >= grid_length ){
ze[l][0] = ze[l][0] - grid_length ;
}
//
b=rand()%nb_p ;
xi[l][0] = b * grid_length / nb_p ;
// periodic boundaries
if( xi[l][0] < 0 ){
xi[l][0] = xi[l][0] + grid_length ;
} else if( xi[l][0] >= grid_length ){
xi[l][0] = xi[l][0] - grid_length ;
}
//
b=rand()%nb_p ;
yi[l][0] = b * grid_length / nb_p ;
// periodic boundaries
if( ye[l][0] < 0 ){
yi[l][0] = yi[l][0] + grid_length ;
} else if( yi[l][0] >= grid_length ){
yi[l][0] = yi[l][0] - grid_length ;
}
//
b=rand()%nb_p ;
zi[l][0] = b * grid_length / nb_p ;
// periodic boundaries
if( ze[l][0] < 0 ){
zi[l][0] = zi[l][0] + grid_length ;
} else if( zi[l][0] >= grid_length ){
```

```

        zi[l][0] = zi[l][0] - grid_length ;
    }
}
// end void
}
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// load velocity (cold plasma as example but it can be modified)
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
void parameter::init_vel(double vex[500][10], double vey[500][10], double
vez[500][10],
double vix[500][10], double viy[500][10], double viz[500][10]){
//
// int nb_p = 500 ;
// int ngrid = 10 ;
// double grid_length = 1.0 ;
// double dstep = grid_length / ngrid ;
// double dpgrid = grid_length / nb_p ; //particle spacing
// double charge = -grid_length / nb_p ; //pseudo-particle charge normalised
to give ncrit=1
// double imp0 = 377.0 ;
// double Ctds = 1.0 ; // Courant number
// double mu_r=1 ;
// double mu_0 = 4*3.14159265359e-7 ;
//
for(int l=0; l < nb_p; l++) {
//
vex[l][0] = 1e-5 ;
vey[l][0] = 1e-5 ;
vez[l][0] = 1e-5 ;
//
// ion is heavy
vix[l][0] = 1e-10 ;
viy[l][0] = 1e-10 ;
viz[l][0] = 1e-10 ;
}
// end void
}
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Compute densities
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
void parameter::init_dens(double rhox[10+1][10], double rhoy[10+1][10],double
rhoz[10+1][10],
double xe[500][10], double ye[500][10], double ze[500][10],
double xi[500][10], double yi[500][10], double zi[500][10]){
//
// int nb_p = 500 ;
// int ngrid = 10 ;
// double grid_length = 1.0 ;
// double dstep = grid_length / ngrid ;
// double dpgrid = grid_length / nb_p ; //particle spacing

```

```

// double charge = -grid_length / nb_p ; //pseudo-particle charge normalised
to give ncrit=1
// double imp0 = 377.0 ;
// double CdtDs = 1.0 ; // Courant number
// double mu_r=1 ;
// double mu_0 = 4*3.14159265359e-7 ;
//
double re = charge / dstep ; //charge weighting factor
double rhoe[10+1], rhoi[10+1] ;
for(int l=0;l<ngrid+1;l++){
    rhoe[l] = 0.0 ; //background electron density
    rhoi[l] = 1.0 ; //background ion density
}
double xa, f1, f2 ;
int j1, j2 ;
// map charges onto grid
for(int l=0; l < ngrid; l++){
    xa = xe[l][0] / dstep ;
    j1 = int(xa) ;
    j2 = j1 + 1 ;
    f2 = xa - j1 ;
    f1 = 1.0 - f2 ;
    rhoe[j1] = rhoe[j1] + re*f1 ;
    rhoe[j2] = rhoe[j2] + re*f2 ;
}
// periodic boundaries
rhoe[0] = rhoe[0] + rhoe[ngrid] ;
rhoe[ngrid] = rhoe[0] ;
//
for(int l=0; l < ngrid+1; l++){
    rhox[l][0] = rhoe[l] + rhoi[l] ;
    rhox[l][0] = 0.0 ;
    rhoz[l][0] = 0.0 ;
}
// end void
}
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// solve maxwell equations (Ampere's and Faraday's laws)
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
void parameter::init_fields(double upd_Ex[10][10][10], double
upd_Ey[10][10][10], double upd_Ez[10][10][10],
double upd_Bx[10][10][10], double upd_By[10][10][10], double
upd_Bz[10][10][10],
double rhox[10+1][10], double rhox[10+1][10],double rhoz[10+1][10]){
//
// int ngrid = 10 ;
// double grid_length = 1.0 ;
// double dstep = grid_length / ngrid ;
// double imp0 = 377.0 ;
// double CdtDs = 1.0 ; // Courant number
// double mu_r=1 , mu_0 = 4*3.14159265359e-7 ;
//
// double Ex_0[10][10][10], Ey_0[10][10][10], Ez_0[10][10][10] ;
// double Hx_0[10][10][10], Hy_0[10][10][10], Hz_0[10][10][10] ;
//
double s_ex=0, s_ey=0, s_ez=0 ; //need this for consistency with charge
conservation

```



```

//
for (int mm = 0; mm < ngrid ; mm++){
  for (int nn = 0; nn < ngrid; nn++){
    for (int pp = 0; pp < ngrid; pp++){
      //
      Ex_0[mm][nn][pp] = 0.0 ;
      Ey_0[mm][nn][pp] = 0.0 ;
      Ez_0[mm][nn][pp] = 0.0 ;
      // =
      Hx_0[mm][nn][pp] = 0.0 ;
      Hy_0[mm][nn][pp] = 0.0 ;
      Hz_0[mm][nn][pp] = 0.0 ;
      //
    }
  }
}
//
// Electric field
//
for (int mm = 0; mm < ngrid - 1; mm++){
  for (int nn = 1; nn < ngrid - 1; nn++){
    for (int pp = 1; pp < ngrid - 1; pp++){
      Ex_0[mm][nn][pp] = 1.0 * Ex_0[mm][nn][pp] + (Cdt ds / imp0) *
      ((Hz_0[mm][nn][pp] -
        Hz_0[mm][nn-1][pp]) - (Hy_0[mm][nn][pp] - Hy_0[mm][nn][pp-1])) ;
      Ex_0[mm][nn][pp] = Ex_0[mm][nn][pp] - 0.5*( rhox[mm][0] +
      rhox[mm+1][0] ) * dstep ; //Additive Source
      s_ex = s_ex + Ex_0[mm][nn][pp] ; //need this for consistency with
      charge conservation
    }
  }
}
//
for (int mm = 1; mm < ngrid - 1; mm++){
  for (int nn = 0; nn < ngrid - 1; nn++){
    for (int pp = 1; pp < ngrid - 1; pp++){
      Ey_0[mm][nn][pp] = 1.0 * Ey_0[mm][nn][pp] + (Cdt ds / imp0) *
      ((Hx_0[mm][nn][pp] -
        Hx_0[mm][nn][pp-1]) - (Hz_0[mm][nn][pp] - Hz_0[mm-1][nn][pp])) ;
      Ey_0[mm][nn][pp] = Ey_0[mm][nn][pp] - 0.5*( rho y[mm][0] +
      rho y[mm+1][0] ) * dstep ; //Additive Source
      s_ey = s_ey + Ey_0[mm][nn][pp] ; //need this for consistency with
      charge conservation
    }
  }
}
//
for (int mm = 1; mm < ngrid - 1; mm++){
  for (int nn = 1; nn < ngrid - 1; nn++){
    for (int pp = 0; pp < ngrid - 1; pp++){
      Ez_0[mm][nn][pp] = 1.0 * Ez_0[mm][nn][pp] + (Cdt ds / imp0) *
      ((Hy_0[mm][nn][pp] -
        Hy_0[mm-1][nn][pp]) - (Hx_0[mm][nn][pp] - Hx_0[mm][nn-1][pp])) ;
      Ez_0[mm][nn][pp] = Ez_0[mm][nn][pp] - 0.5*( rho z[mm][0] +
      rho z[mm+1][0] ) * dstep ; //Additive Source
      s_ez = s_ez + Ez_0[mm][nn][pp] ; //need this for consistency with
      charge conservation
    }
  }
}
//
// Magnetic field
//

```

```

    for (int mm = 0; mm < ngrid; mm++){
        for (int nn = 0; nn < ngrid - 1; nn++){
            for (int pp = 0; pp < ngrid - 1; pp++){
                Hx_0[mm][nn][pp] = (Cdt ds / imp0) * Hx_0[mm][nn][pp] + 1.0 *
                ((Ey_0[mm][nn][pp+1] -
                 Ey_0[mm][nn][pp]) - (Ez_0[mm][nn+1][pp] - Ez_0[mm][nn][pp])) ;
            }
        }
    }
    //
    for (int mm = 0; mm < ngrid - 1; mm++){
        for (int nn = 0; nn < ngrid; nn++){
            for (int pp = 0; pp < ngrid - 1; pp++){
                Hy_0[mm][nn][pp] = (Cdt ds / imp0) * Hy_0[mm][nn][pp] + 1.0 *
                ((Ez_0[mm+1][nn][pp] -
                 Ez_0[mm][nn][pp]) - (Ex_0[mm][nn][pp+1] - Ex_0[mm][nn][pp])) ;
            }
        }
    }
    //
    for (int mm = 0; mm < ngrid - 1; mm++){
        for (int nn = 0; nn < ngrid - 1; nn++){
            for (int pp = 0; pp < ngrid; pp++){
                Hz_0[mm][nn][pp] = (Cdt ds / imp0) * Hz_0[mm][nn][pp] + 1.0 *
                ((Ex_0[mm][nn+1][pp] -
                 Ex_0[mm][nn][pp]) - (Ey_0[mm+1][nn][pp] - Ey_0[mm][nn][pp])) ;
            }
        }
    }
    //
    // return Electric fields
    //
    for (int mm = 0; mm < ngrid ; mm++){
        for (int nn = 0; nn < ngrid ; nn++){
            for (int pp = 0; pp < ngrid ; pp++){
                upd_Ex[mm][nn][pp] = Ex_0[mm][nn][pp] - s_ex / ngrid ;
            }
        }
    }
    upd_Ex[ngrid-1][ngrid-1][ngrid-1] = upd_Ex[0][ngrid-1][ngrid-1] ; //
    periodic boundaries
    //
    for (int mm = 0; mm < ngrid ; mm++){
        for (int nn = 0; nn < ngrid ; nn++){
            for (int pp = 0; pp < ngrid ; pp++){
                upd_Ey[mm][nn][pp] = Ey_0[mm][nn][pp] - s_ey / ngrid ;
            }
        }
    }
    upd_Ey[ngrid-1][ngrid-1][ngrid-1] = upd_Ey[ngrid-1][0][ngrid-1] ; //
    periodic boundaries
    //
    for (int mm = 0; mm < ngrid ; mm++){
        for (int nn = 0; nn < ngrid ; nn++){
            for (int pp = 0; pp < ngrid ; pp++){
                upd_Ez[mm][nn][pp] = Ez_0[mm][nn][pp] - s_ez / ngrid ;
            }
        }
    }
    upd_Ez[ngrid-1][ngrid-1][ngrid-1] = upd_Ez[ngrid-1][ngrid-1][0] ; //
    periodic boundaries
    //
    // return Magnetic field : B = mu_r * mu_0 * H

```

```

//
for (int mm = 0; mm < ngrid ; mm++){
  for (int nn = 0; nn < ngrid ; nn++){
    for (int pp = 0; pp < ngrid ; pp++){
      upd_Bx[mm][nn][pp] = mu_r * mu_0 * Hx_0[mm][nn][pp] ;
    }
  }
}
//
for (int mm = 0; mm < ngrid ; mm++){
  for (int nn = 0; nn < ngrid ; nn++){
    for (int pp = 0; pp < ngrid ; pp++){
      upd_By[mm][nn][pp] = mu_r * mu_0 * Hy_0[mm][nn][pp] ;
    }
  }
}
//
for (int mm = 0; mm < ngrid ; mm++){
  for (int nn = 0; nn < ngrid ; nn++){
    for (int pp = 0; pp < ngrid ; pp++){
      upd_Bz[mm][nn][pp] = mu_r * mu_0 * Hz_0[mm][nn][pp] ;
    }
  }
}
}
//end void
}

```

Density

Dens_currnt.h:

```

#ifndef DENS_CURRNT_H
#define DENS_CURRNT_H
#include "parameter.h"
class dens_currnt
{
public :
  parameter params;
  dens_currnt(parameter p){
    this->params = p;
  }
  void dens(int t, double xe[500][10], double ye[500][10], double ze[500][10],
            double xi[500][10] , double yi[500][10] , double zi[500][10],
            double rhox[10+1][10], double rhoy[10+1][10],double
rhoz[10+1][10]) ;
};

#endif // DENS_CURRNT_H

```

Dens_currnt.cpp::

```

//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// useful libraries
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#include "dens_currnt.h"
#include "parameter.h"
//

```

```

//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Global parameters
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
//int maxTime = 50 ;
//int nb_p = 500 ;
//int ngrid = 50 ;
//
//double pi = 3.14159265359 ;
//
//double grid_length = 2*pi ;
//double dstep = grid_length / ngrid ;
//double dpgrid = grid_length / nb_p ; //particle spacing
//double charge = -grid_length / nb_p ; //pseudo-particle charge normalised to
give ncrit=1
//double mass = grid_length / nb_p ; //pseudo-particle mass
//double imp0 = 377.0 ;
//double mu_r=1, mu_0 = 4*3.14159265359e-7 ;
//double q_over_me=-1.0 ;
//double dt = 0.05 ; //normalised timestep
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
parameter paraml;
void dens_currnt::dens(int t, double xe[500][10], double ye[500][10], double
ze[500][10],
double xi[500][10], double yi[500][10], double zi[500][10],
double rhox[10+1][10], double rhoy[10+1][10], double
rhoz[10+1][10]){
//
// int nb_p = 500 ;
// int ngrid = 10 ;
// //
// double grid_length = 1.0 ;
// double dstep = grid_length / ngrid ;
// double charge = -grid_length / nb_p ;
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Compute densities
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
double re = paraml.charge / paraml.dstep ; //charge weighting factor
double rhoe[10+1], rhoi[10+1] ;
for(int l = 0 ; l < paraml.ngrid + 1 ; l++){
rhoe[l] = 0.0 ;
rhoi[l] = 1.0 ; //background ion density
}
double xa, f1, f2 ;
int j1, j2 ;
//

```

```

//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// map charges onto grid
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
for(int l=0;l<param1.ngrid;l++){
    xa = xe[l][t] / param1.dstep ;
    j1 = int(xa) ;
    j2 = j1 + 1 ;
    f2 = xa - j1 ;
    f1 = 1.0 - f2 ;
    rhoe[j1] = rhoe[j1] + re*f1 ;
    rhoe[j2] = rhoe[j2] + re*f2 ;
}
// periodic boundaries
rhoe[0] = rhoe[0] + rhoe[param1.ngrid] ;
rhoe[param1.ngrid] = rhoe[0] ;
//
for(int l=0;l<param1.ngrid+1;l++){
    rhox[l][t] = rhoe[l] + rhoi[l] ;
    rhox[l][t] = 0.0 ;
    rhoz[l][t] = 0.0 ;
}
// end void
}

```

Maxwell equation

Maxwell_equat.h::

```

#ifndef MAXWELL_EQUAT_H
#define MAXWELL_EQUAT_H

//int maxTime = 50 ;
//int nb_p = 500 ;
//int ngrid = 50 ;
#include "parameter.h"
class maxwell_equat
{
public:
    parameter params;
    maxwell_equat(parameter p){
        this->params = p;
    }
//
    void solve(int t, double Ex[10][10][10], double Ey[10][10][10], double
Ez[10][10][10],
        double Bx[10][10][10], double By[10][10][10], double Bz[10][10][10],
        double rhox[10+1][10], double rhoz[10+1][10],double rhoe[10+1][10],
        double Ex_save[10][10][10], double Ey_save[10][10][10], double
Ez_save[10][10][10],
        double Bx_save[10][10][10], double By_save[10][10][10], double
Bz_save[10][10][10]);
};

```

```
#endif // MAXWELL_EQUAT_H
```

Maxwell_equat.cpp::

```
//
```

```

//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// useful libraries
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
#include "maxwell_equat.h"
#include "parameter.h"
#include "math.h"
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Global parameters
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
parameter param;
void maxwell_equat::solve(int t, double Ex[10][10][10], double
Ey[10][10][10], double Ez[10][10][10],
double Bx[10][10][10], double By[10][10][10], double Bz[10][10][10],
double rhox[10+1][10], double rhoy[10+1][10], double rhoz[10+1][10],
double Ex_save[10][10][10], double Ey_save[10][10][10], double
Ez_save[10][10][10],
double Bx_save[10][10][10], double By_save[10][10][10], double
Bz_save[10][10][10]){
//
// int ngrid = 10 ;
// double grid_length = 1.0 ;
// double dstep = grid_length / ngrid ;
// double mu_0 = 4*3.14159265359e-7 ;
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// solve maxwell equations (Ampere's and Faraday's laws)
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
double s_ex=0, s_ey=0, s_ez=0 ; //need this for consistency with charge
conservation
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Magnetic field
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
for (int mm = 0; mm < param.ngrid ; mm++){
for (int nn = 0; nn < param.ngrid ; nn++){

```

```

        for (int pp = 0; pp < param.ngrid ; pp++){
            Bx[mm][nn][pp] = Bx_save[mm][nn][pp] / param.mu_0 ;
            By[mm][nn][pp] = By_save[mm][nn][pp] / param.mu_0 ;
            Bz[mm][nn][pp] = Bz_save[mm][nn][pp] / param.mu_0 ;
        }
    }
}
//
for (int mm = 0; mm < param.ngrid; mm++){
    for (int nn = 0; nn < param.ngrid - 1; nn++){
        for (int pp = 0; pp < param.ngrid - 1; pp++){
            Bx[mm][nn][pp] = (1.0 / 377.0) * Bx_save[mm][nn][pp] + 1.0 *
(Ey_save[mm][nn][pp+1] -
            Ey_save[mm][nn][pp]) - (Ez_save[mm][nn+1][pp] -
Ez_save[mm][nn][pp]) ;
        }
    }
}
//
for (int mm = 0; mm < param.ngrid - 1; mm++){
    for (int nn = 0; nn < param.ngrid; nn++){
        for (int pp = 0; pp < param.ngrid - 1; pp++){
            By[mm][nn][pp] = (1.0 / 377.0) * By_save[mm][nn][pp] + 1.0 *
((Ez_save[mm+1][nn][pp] -
            Ez_save[mm][nn][pp]) - (Ex_save[mm][nn][pp+1] -
Ex_save[mm][nn][pp])) ;
        }
    }
}
//
for (int mm = 0; mm < param.ngrid - 1; mm++){
    for (int nn = 0; nn < param.ngrid - 1; nn++){
        for (int pp = 0; pp < param.ngrid; pp++){
            Bz[mm][nn][pp] = (1.0 / 377.0) * Bz_save[mm][nn][pp] + 1.0 *
((Ex_save[mm][nn+1][pp]
            - Ex_save[mm][nn][pp]) - (Ey_save[mm+1][nn][pp] -
Ey_save[mm][nn][pp])) ;
        }
    }
}
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Electric field
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
for (int mm = 0; mm < param.ngrid - 1; mm++){
    for (int nn = 1; nn < param.ngrid - 1; nn++){
        for (int pp = 1; pp < param.ngrid - 1; pp++){
            Ex[mm][nn][pp] = 1.0 * Ex_save[mm][nn][pp] + (1.0 / 377.0) *
((Bz_save[mm][nn][pp] -
            Bz_save[mm][nn-1][pp]) - (By_save[mm][nn][pp] -
By_save[mm][nn][pp-1])) ;
            Ex[mm][nn][pp] = Ex[mm][nn][pp] - 0.5*( rhox[mm][t] +
rhox[mm+1][t] )*param.dstep ; //Additive Source
            s_ex = s_ex + Ex[mm][nn][pp] ; //need this for consistency with
charge conservation
        }
    }
}
}

```

```

//
for (int mm = 1; mm < param.ngrid - 1; mm++){
  for (int nn = 0; nn < param.ngrid - 1; nn++){
    for (int pp = 1; pp < param.ngrid - 1; pp++){
      Ey[mm][nn][pp] = 1.0 * Ey_save[mm][nn][pp] + (1.0 / 377.0) *
((Bx_save[mm][nn][pp] -
      Bx_save[mm][nn][pp-1]) - (Bz_save[mm][nn][pp] - Bz_save[mm-
1][nn][pp])) ;
      Ey[mm][nn][pp] = Ey[mm][nn][pp] - 0.5*( rhoy[mm][t] +
rhoy[mm+1][t] )*param.dstep ; //Additive Source
      s_ey = s_ey + Ey[mm][nn][pp] ; //need this for consistency with
charge conservation
    }
  }
}
//
for (int mm = 1; mm < param.ngrid - 1; mm++){
  for (int nn = 1; nn < param.ngrid - 1; nn++){
    for (int pp = 0; pp < param.ngrid - 1; pp++){
      Ez[mm][nn][pp] = 1.0 * Ez_save[mm][nn][pp] + (1.0 / 377.0) *
((By_save[mm][nn][pp] -
      By_save[mm-1][nn][pp]) - (Bx_save[mm][nn][pp] -
Bx_save[mm][nn-1][pp])) ;
      Ez[mm][nn][pp] = Ez[mm][nn][pp] - 0.5*( rhoz[mm][t] +
rhoz[mm+1][t] )*param.dstep ; //Additive Source
      s_ez = s_ez + Ez[mm][nn][pp] ; //need this for consistency with
charge conservation
    }
  }
}
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// return Electric fields
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
for (int mm = 0; mm < param.ngrid ; mm++){
  for (int nn = 0; nn < param.ngrid ; nn++){
    for (int pp = 0; pp < param.ngrid ; pp++){
      Ex[mm][nn][pp] = Ex[mm][nn][pp] - s_ex / param.ngrid ;
      Ey[mm][nn][pp] = Ey[mm][nn][pp] - s_ey / param.ngrid ;
      Ez[mm][nn][pp] = Ez[mm][nn][pp] - s_ez / param.ngrid ;
    }
  }
}
Ex[param.ngrid-1][param.ngrid-1][param.ngrid-1] = Ex[0][param.ngrid-
1][param.ngrid-1] ; // periodic boundaries
Ey[param.ngrid-1][param.ngrid-1][param.ngrid-1] = Ey[param.ngrid-
1][0][param.ngrid-1] ; // periodic boundaries
Ez[param.ngrid-1][param.ngrid-1][param.ngrid-1] = Ez[param.ngrid-
1][param.ngrid-1][0] ; // periodic boundaries
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// return Magnetic field : B = mu_0 * H
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//

```



```

    for (int mm = 0; mm < param.ngrid ; mm++){
        for (int nn = 0; nn < param.ngrid ; nn++){
            for (int pp = 0; pp < param.ngrid ; pp++){
                Bx[mm][nn][pp] = param.mu_0 * Bx[mm][nn][pp] ;
                By[mm][nn][pp] = param.mu_0 * By[mm][nn][pp] ;
                Bz[mm][nn][pp] = param.mu_0 * Bz[mm][nn][pp] ;
            }
        }
    }
}
//end void
}

```

Position and velocity

Pos_veloct.h

```

#ifndef POS_VELOCT_H
#define POS_VELOCT_H
#include "parameter.h"
class Pos_veloct
{
public:
    parameter params;
    Pos_veloct(parameter p){
        this->params = p;
    }
    void push(int t, double xe[500][10], double ye[500][10], double ze[500][10],
              double vex[500][10], double vey[500][10], double vez[500][10],
              double xi[500][10] , double yi[500][10] , double zi[500][10],
              double vix[500][10], double viy[500][10], double viz[500][10],
              double Ex[10][10][10], double Ey[10][10][10], double
Ez[10][10][10]);
};

#endif // POS_VELOCT_H

```

Pos_veloct.cpp::

```

//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// useful libraries
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
#include "pos_veloct.h"
#include "parameter.h"
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Global parameters
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
//int maxTime = 50 ;
//int nb_p = 500 ;
//int ngrid = 50 ;
//
//double pi = 3.14159265359 ;
//

```

```

//double grid_length = 2*pi ;
//double dstep = grid_length / ngrid ;
//double dpgrid = grid_length / nb_p ; //particle spacing
//double charge = -grid_length / nb_p ; //pseudo-particle charge normalised to
give ncrit=1
//double mass = grid_length / nb_p ; //pseudo-particle mass
//double imp0 = 377.0 ;
//double Cdt ds = 1.0 ; // Courant number
//double mu_r=1, mu_0 = 4*3.14159265359e-7 ;
//double q_over_me=-1.0 ;
//double dt = 0.05 ; //normalised timestep
//
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
parameter param2;
void Pos_veloct::push(int t, double xe[500][10], double ye[500][10], double
ze[500][10],
                double vex[500][10], double vey[500][10], double vez[500][10],
                double xi[500][10] , double yi[500][10] , double zi[500][10],
                double vix[500][10], double viy[500][10], double viz[500][10],
                double Ex[10][10][10], double Ey[10][10][10], double
Ez[10][10][10]){
    //
//    int nb_p = 500 ;
//    int ngrid = 10 ;
//    double grid_length = 1.0 ;
//    double dstep = grid_length / ngrid ;
    double q_over_me=-1.0 ;
    double dt = 0.05 ; //normalised timestep
    //
    //
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//    interpolate field E from grid to particle (B<<E)
//
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//
    double xa, b1, b2, exi ;
    int j1, j2 ;
    //
    for(int l = 0 ; l < param2.nb_p ; l++){
        //
        exi = 0 ;
        xa = xe[l][t-1] / param2.dstep ;
        j1 = int(xa) ;
        j2 = j1 + 1 ;
        b2 = xa - j1 ;
        b1 = 1.0 - b2 ;
        for(int nn=0;l<param2.ngrid;l++){
            for(int pp=0;l<param2.ngrid;l++){
                if( (j1 <= param2.ngrid-2)&&(j2 <= param2.ngrid-2) ){
                    exi = exi + b1*Ex[j1][nn][pp] + b2*Ex[j2][nn][pp] ;
                } else if( (j1 > param2.ngrid-2)&&(j2 <= param2.ngrid-2) ){
                    exi = exi + b1*Ex[j1-(param2.ngrid-2)][nn][pp] +
b2*Ex[j2][nn][pp] ;
                } else if( (j1 <= param2.ngrid-2)&&(j2 > param2.ngrid-2) ){

```

```

                exi = exi + b1*Ex[j1][nn][pp] + b2*Ex[j2-(param2.ngrid-
2)][nn][pp] ;
            } else if( (j1 > param2.ngrid-2)&&(j2 > param2.ngrid-2) ){
                exi = exi + b1*Ex[j1-(param2.ngrid-2)][nn][pp] + b2*Ex[j2-
(param2.ngrid-2)][nn][pp] ;
            }
        }
    }
    // update velocities
    vex[l][t] = vex[l][t-1] + q_over_me * dt * exi ;
    vey[l][t] = vey[l][t-1] ;
    vez[l][t] = vez[l][t-1] ;
    // update positions
    xe[l][t] = xe[l][t-1] + vex[l][t] * dt ;
    ye[l][t] = ye[l][t-1] + vey[l][t] * dt ;
    ze[l][t] = ze[l][t-1] + vez[l][t] * dt ;
    // freeze approximation for ions
    vix[l][t] = vix[l][t-1] ;
    viy[l][t] = viy[l][t-1] ;
    viz[l][t] = viz[l][t-1] ;
    xi[l][t] = xi[l][t-1] + vix[l][t] * dt ;
    yi[l][t] = yi[l][t-1] + viy[l][t] * dt ;
    zi[l][t] = zi[l][t-1] + viz[l][t] * dt ;
    //
    // periodic boundaries
    //
    if( xe[l][t] < 0 ){
        xe[l][t] = xe[l][t] + param2.grid_length ;
    } else if( xe[l][t] >= param2.grid_length ){
        xe[l][t] = xe[l][t] - param2.grid_length ;
    }
    //
    if( ye[l][t] < 0 ){
        ye[l][t] = ye[l][t] + param2.grid_length ;
    } else if( ye[l][t] >= param2.grid_length ){
        ye[l][t] = ye[l][t] - param2.grid_length ;
    }
    //
    if( ze[l][t] < 0 ){
        ze[l][t] = ze[l][t] + param2.grid_length ;
    } else if( ze[l][t] >= param2.grid_length ){
        ze[l][t] = ze[l][t] - param2.grid_length ;
    }
    //
    if( xi[l][t] < 0 ){
        xi[l][t] = xi[l][t] + param2.grid_length ;
    } else if( xi[l][t] >= param2.grid_length ){
        xi[l][t] = xi[l][t] - param2.grid_length ;
    }
    //
    if( yi[l][t] < 0 ){
        yi[l][t] = yi[l][t] + param2.grid_length ;
    } else if( yi[l][t] >= param2.grid_length ){
        yi[l][t] = yi[l][t] - param2.grid_length ;
    }
    //
    if( zi[l][t] < 0 ){
        zi[l][t] = zi[l][t] + param2.grid_length ;
    } else if( zi[l][t] >= param2.grid_length ){
        zi[l][t] = zi[l][t] - param2.grid_length ;
    }
}
// end void

```

}

Annex B:: code python

-*- coding: utf-8 -*-

```
""" ESPIC: a simple 1D1V electrostatic PIC code
    Based on 'espic.c' developed for 2000 Heraeus School, Jena

    (c) P. Gibbon, KU-Leuven & FZ-Juelich, November 2013

"""

import numpy as np  ## numeric routines; arrays
import pylab as plt  ## plotting

""" core routines
"""

#=====
# Particle loading - positions
#=====

def loadx(bc_particle):
    global dx, grid_length, rho0, npart, q_over_me, a0
    global charge, mass, wall_left, wall_right
    print("Load particles")

    # set up particle limits
    if (bc_particle >= 2):
        # reflective boundaries
        # place particle walls half a mesh spacing inside field boundaries
        wall_left = dx/2.
        wall_right = grid_length-3*dx/2.
        plasma_start = wall_left
        plasma_end = wall_right  # actually want min(end,wr) */

    else:
        # periodic boundaries
        plasma_start = 0.
        plasma_end = grid_length
        wall_left = 0.
        wall_right = grid_length

    xload = plasma_end - plasma_start  # length for particle loading */
    dpdx = xload/npart  # particle spacing */
    charge = -rho0*dpdx  # pseudo-particle charge normalised to give
    ncrit=1 (rhoc=-1)
    mass = charge/q_over_me  # pseudo-particle mass (need for kinetic energy
    diagnostic)

    for i in range(npart):
        x[i] = plasma_start + dpdx*(i+0.5)  # Python ndarrays start at index 0
        x[i] += a0*np.cos(x[i])  # Include small perturbation

    return True

#=====
# Particle loading - velocities
#=====
```

```

def loadv(idist,vte):
    global npart,v,grid_length,v0
    print("Set up velocity distribution")
    iseed = 76523 # random number seeds
    idum1 = 137

    if ( idist == 1 ):
        # >10 = set up thermal distribution
        for i in range(npart):
            vm = vte*np.sqrt( (-2.*np.log((i+0.5)/npart)) ) # inverted 2v-
distribution - amplitude */
            rs = np.random.random_sample() # random angle */
            theta = 2*np.pi*rs
            v[i] = vm*np.sin(theta) # x-component of v

        # scramble particle indicies to remove correlations between x and v
        np.random.shuffle(v)

    else:
        # Default is cold plasma */
        v[1:npart] = 0.

# add perturbation
v += v0*np.sin(2*np.pi*x/grid_length)
return True

=====
# Compute densities
=====

def density(bc_field,qe):
    global x,rhoe,rhoi,dx,npart,ngrid,wall_left,wall_right
    j1=np.dtype(np.int32)
    j2=np.dtype(np.int32)

    re = qe/dx # charge weighting factor
    rhoe=np.zeros(ngrid+1) # electron density
    # map charges onto grid
    for i in range(npart):
        xa = x[i]/dx
        j1 = int(xa)
        j2 = j1 + 1
        f2 = xa - j1
        f1 = 1.0 - f2
        rhoe[j1] = rhoe[j1] + re*f1
        rhoe[j2] = rhoe[j2] + re*f2

    if (bc_field == 1):
        # periodic boundaries */
        rhoe[0] += rhoe[ngrid]
        rhoe[ngrid] = rhoe[0]

    elif (bc_field == 2):
        # reflective - 1st and last (ghost) cells folded back onto physical grid
        */
        iw1 = wall_left/dx
        rhoe[iw1+1] += rhoe[iw1]
        rhoe[iw1] = 0.0
        iwr = wall_right/dx
        rhoe[iwr] += rhoe[iwr+1]
        rhoe[iwr+1] = rhoe[iwr]
    else:

```

```

    print("Invalid value for bc_field:", bc_field)

# Add neutral ion density
rhoi = rho0
# print(rhoe[0:ngrid+1])
return True

=====
# Compute electrostatic field
=====

def field():
    global rhoe,rhoi,ex,dx,ngrid

    rhot=rhoe+rhoi # net charge density on grid

    # integrate div.E=rho directly (trapezium approx)
    # end point - ex=0 mirror at right wall

    Ex[ngrid]=0. # zero electric field
    edc = 0.0

    for j in range(ngrid-1,-1,-1):
        Ex[j] = Ex[j+1] - 0.5*( rhot[j] + rhot[j+1] )*dx
        edc = edc + Ex[j]

    if (bc_field == 1):
        # periodic fields: subtract off DC component */
        # -- need this for consistency with charge conservation */
        Ex[0:ngrid] -= edc/ngrid
        Ex[ngrid] = Ex[0]

    return True

=====
# Particle pusher
=====

def push():
    global x,v,Ex,dt,dx,npart,q_over_me

    for i in range(npart):

        # interpolate field Ex from grid to particle */
        xa = x[i]/dx
        j1 = int(xa)
        j2 = j1 + 1
        b2 = xa - j1
        b1 = 1.0 - b2
        exi = b1*Ex[j1] + b2*Ex[j2]
        v[i] = v[i] + q_over_me*dt*exi # update velocities */

    x += dt*v # update positions (2nd half of leap-frog)

    return True

=====
# check particle boundary conditions
=====

```

```

def particle_bc(bc_particle,xl):
    global x
    # int iseed1 = 28631;          /* random number seed */
    # int iseed2 = 1631;          /* random number seed */

    # loop over all particles to see if any have
    # left simulation region: if so, we put them back again
    # according to the switch 'bc_particle' **/

    if (bc_particle == 1):
        # periodic
        for i in range(npart):
            if ( x[i] < 0.0 ):
                x[i] += xl
            elif ( x[i] >= xl ):
                x[i] -= xl

    elif (bc_particle == 2):
        # reflective
        print("Reflective boundaries not implemented yet.")

    elif (bc_particle == 3):
        # reflective
        print("Thermal boundaries not implemented yet.")

    else:
        print("Invalid value for bc_particle:", bc_particle)

    return True

=====
# Diagnostic outputs for fields and particles
=====

def diagnostics():
    global rhoe,Ex,ngrid,itime,grid_length,rho0,a0
    global ukin, upot, utot, udrift, utherm, emax,fv,fm
    global iout,igraph,iphase,ivdist
    xgrid=dx*np.arange(ngrid+1)
    if (itime==0):
        plt.figure('fields')
        plt.clf()
    if (igraph > 0):
        if (np.fmod(itime,igraph)==0): # plots every igrph steps
    # Net density
        plt.subplot(2, 2, 1)
        if (itime > 0 ): plt.cla()
        plt.plot(xgrid, -(rhoe+rho0), 'r', label='density')
        plt.xlabel('x')
        plt.xlim(0,grid_length)
        plt.ylim(-2*a0,2*a0)
        plt.legend(loc=1)
    # Electric field
        plt.subplot(2, 2, 2)
        if (itime > 0 ): plt.cla()
        plt.plot(xgrid, Ex, 'b', label='Ex')
        plt.xlabel('x')
        plt.ylim(-2*a0,2*a0)
        plt.xlim(0,grid_length)

        plt.legend(loc=1)

```

```

    if (iphase > 0):
        if (np.fmod(itime,iphase)==0):
# Phase space plots every iphase steps
        axScatter = plt.subplot(2, 2, 3)
        if (itime >0 ): plt.cla()
        axScatter.scatter(x,v,marker='.',s=1)
#
        axScatter.set_aspect(1.)
        axScatter.set_xlim(0,grid_length)
        axScatter.set_ylim(-vmax,vmax)
        axScatter.set_xlabel('x')
        axScatter.set_ylabel('v')

    if (ivdist > 0):
        if (np.fmod(itime,ivdist)==0):
# Distribution function plots every ivdist steps
        fv=np.zeros(nvbin+1) # zero distn fn
        dv = 2*vmax/nvbin # bin separation */
        for i in range(npart):
            vax= ( v[i] + vmax )/dv # norm. velocity */
            iv = int(vax)+1 # bin index */
            if (iv <= nvbin and iv > 0): fv[iv] +=1 # /* increment dist. fn
if within range

        plt.subplot(2, 2, 4)
        if (itime >0 ): plt.cla()
        vgrid=dv*np.arange(nvbin+1)-vmax
        plt.plot(vgrid, fv, 'g', label='f(v)')
        plt.xlabel('v')
        plt.xlim(-vmax,vmax)
#
        plt.ylim(-2*a0,2*a0)
        plt.legend(loc=1)
        fn_vdist = 'vdist_%0*d'%(5, itime)

        np.savetxt(fn_vdist,
np.column_stack((vgrid,fv)),fmt=('%1.4e','%1.4e')) # write to file

        plt.pause(0.0001)
        plt.draw()
        filename = 'fields_%0*d'%(5, itime)
        if (iout > 0):
            if (np.fmod(itime,iout)==0): # printed plots every iout steps
                plt.savefig(filename+'.png')

# total kinetic energy
v2=v**2
vdrift=sum(v)/npart
ukin[itime] = 0.5*mass*sum(v2)
udrift[itime] = 0.5*mass*vdrift*vdrift*npart
utherm[itime] = ukin[itime] - udrift[itime]

# potential energy
e2=Ex**2
upot[itime] = 0.5*dx*sum(e2)
emax = max(Ex) # max field for instability analysis */

# total energy
utot[itime] = upot[itime] + ukin[itime]

return True

```



```

=====
#   Plot time-histories
=====

def histories():

#FILE *history_file;      /* file for writing out time histories */

    global ukin, upot, utot, udrift, utherm
    xgrid=dt*np.arange(nsteps+1)
#   plt.clf()
    plt.figure('Energies')
#   plt.subplot(2, 2, 1)
    plt.plot(xgrid, upot, 'b', label='Upot')
    plt.plot(xgrid, ukin, 'r', label='Ukin')
    plt.plot(xgrid, utot, 'black', label='Utot')
#   plt.plot(xgrid, udrift, 'g', label='Udrift')
    plt.xlabel('t')
    plt.ylabel('Energy')

#   plt.xlim(0,grid_length)
#   plt.ylim(-2*a0,2*a0)
    plt.legend(loc=1)
    plt.savefig('energies.png')

#   write energies out to file */
    np.savetxt('energies.out',
np.column_stack((xgrid,upot,ukin,utot)),fmt=('%1.4e','%1.4e','%1.4e','%1.4e'))
#   x,y,z equal sized 1D arrays
#   fohist = open("energies.data","w")
#   str.format("{0:<10.5f}", 3.14159265)
#   if (itime==1) {fprintf(history_file, " t, U_drift, U_therm, U_field,
U_total, Emax\n");}
#   fprintf( history_file, "%f %e %e %e %e %e\n", itime*dt, udrift, utherm,
upot, utot, emax );

=====
#   Main program
=====

npart=10000          # particles
ngrid=100           # grid points
nsteps=100          # timesteps

# particle arrays
x = np.zeros(npart) # positions
v = np.zeros(npart) # velocities#   particle_bc()

# grid arrays
rhoe=np.zeros(ngrid+1) # electron density
rhoi=np.zeros(ngrid+1) # ion density
Ex=np.zeros(ngrid+1) # electric field
phi=np.zeros(ngrid+1) # potential
# time histories
ukin=np.zeros(nsteps+1)
upot=np.zeros(nsteps+1)
utherm=np.zeros(nsteps+1)
udrift=np.zeros(nsteps+1)
utot=np.zeros(nsteps+1)

```

```

# Define main variables and defaults
# -----

#ni = 0;          /* # ions (fixed) */
grid_length = 2.0*np.pi # size of spatial grid
#grid_length = 16 # size of spatial grid
plasma_start = 0. # LH plasma edge
plasma_end = grid_length # RH plasma edge
dx = grid_length/ngrid
dt = 0.05 # normalised timestep
q_over_me=-1.0 # electron charge:mass ratio
rho0 = 1.0 # background ion density
vte = 0.02 # thermal velocity
nvbin=50 # bins for f(v) plot
a0 = 0.1 # perturbation amplitude
vmax = 0.2 # max velocity for f(v) plot
v0=0.0 # velocity perturbation
wall_left=0.
wall_right=1.
bc_field = 1 # field boundary conditions: 1 = periodic
# # 2 = reflective
bc_particle = 1 # particle BCs: 1 = periodic
# # 2 = reflective
# # 3 = thermal
profile = 1 # density profile switch
distribution = 1 # velocity distribution switch: 0 = cold, 1=thermal
# # 2 = 2-stream
ihist = 5 # frequency of time-history output
igraph = int(np.pi/dt/16) # freq. of graphical snapshots
iphase = igragh
ivdist = -igraph
iout = igragh*1 # freq. of saved graphics files
itime = 0 # initialise time counter

# Setup initial particle distribution and fields
# -----

loadx(bc_particle) # load particles onto grid
loadv(distribution,vte) # define velocity distribution
x += 0.5*dt*v # centre positions for 1st leap-frog step
particle_bc(bc_particle,grid_length)
density(bc_field,charge) # compute initial density from particles
field() # compute initial electric field
diagnostics() # output initial conditions
print('resolution dx/\lambda_D=',dx/vte)

# Main iteration loop
# -----

for itime in range(1,nsteps+1):
    print('timestep ',itime)
    push() # Push particles
    particle_bc(bc_particle,grid_length) # enforce particle boundary conditions
    density(bc_field,charge) # compute density
    field() # compute electric field (Poisson)
    diagnostics() # output snapshots and time-histories

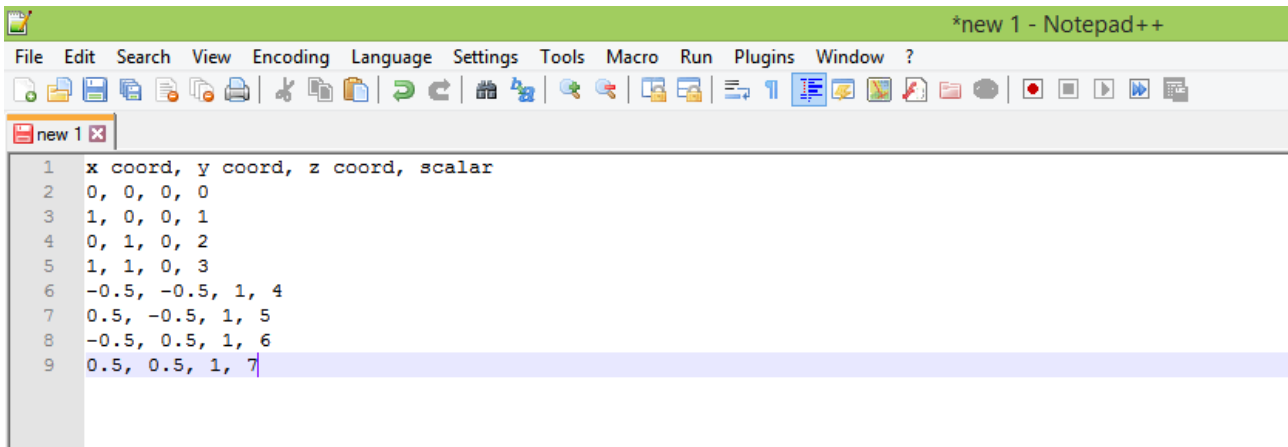
histories() # Produce time-history plots

#raw_input("Press key to finish")
plt.close()
print('done')

```

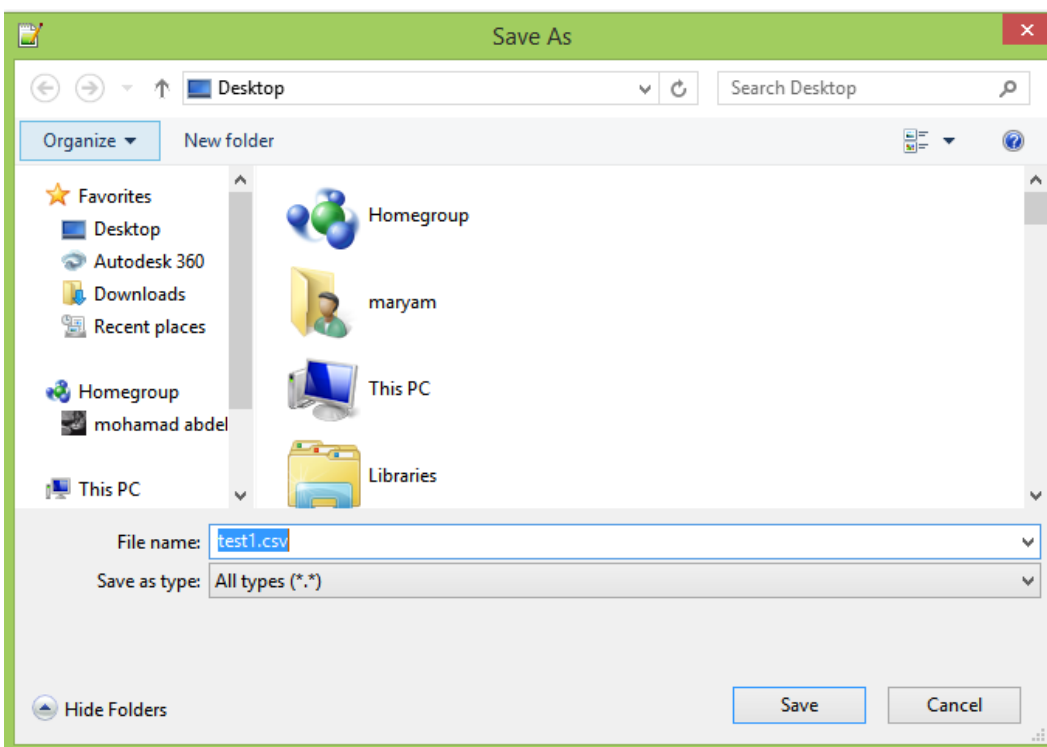
58.2 Paraview Input files

The type of files we are using to read our solution via Para view is the .csv files (comma separated variables). In this section we'll show a simple example (8 points). First start with defining the .csv file using notepad++ as shown in the figure bellows.



```
*new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1 x
1 x coord, y coord, z coord, scalar
2 0, 0, 0, 0
3 1, 0, 0, 1
4 0, 1, 0, 2
5 1, 1, 0, 3
6 -0.5, -0.5, 1, 4
7 0.5, -0.5, 1, 5
8 -0.5, 0.5, 1, 6
9 0.5, 0.5, 1, 7
```

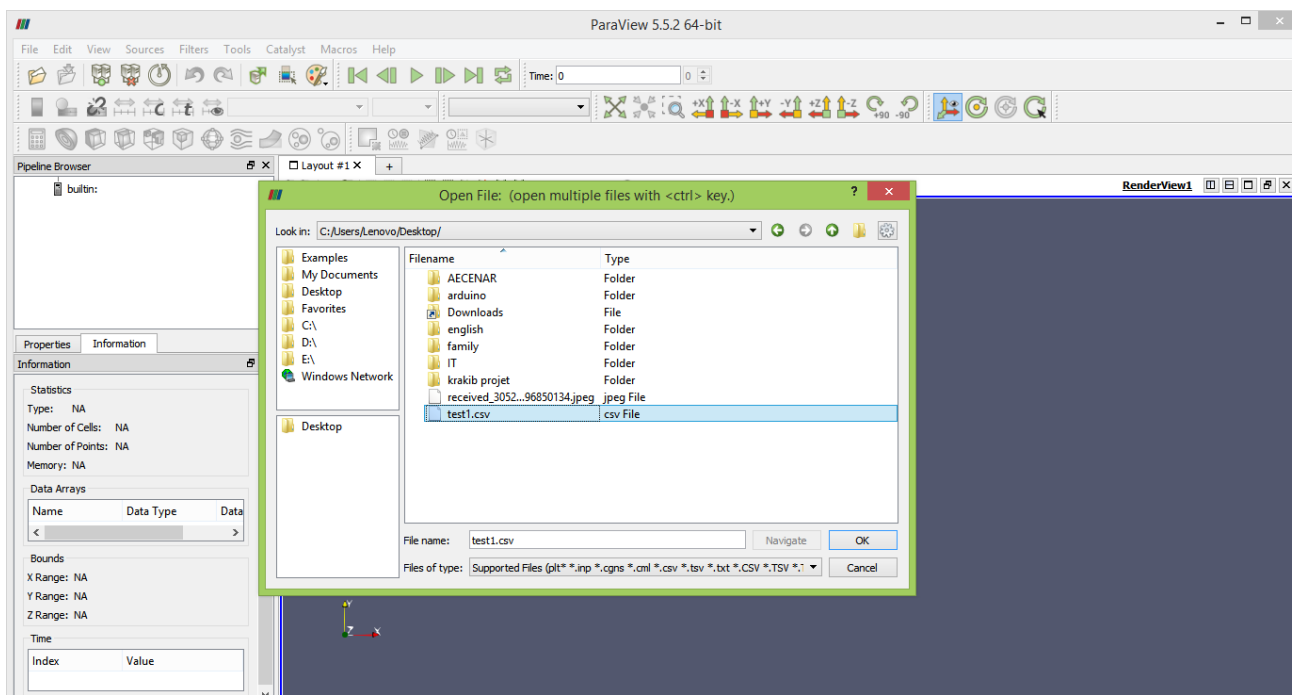
Then save your file as: All types (*.*) as shown in our example test1.csv.



Now it's ready to be opened in Para view.

Select the open button and choose your file .csv.

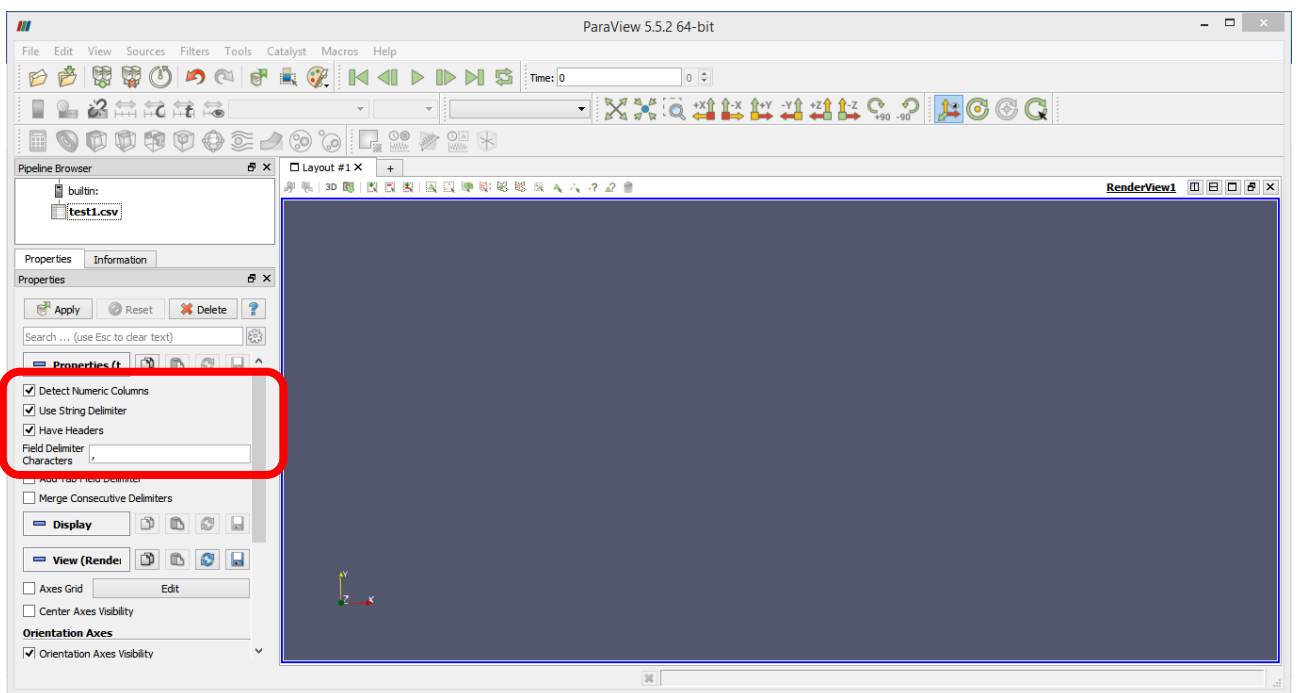
INT-LMIC (Laser Matter Interaction Code) (2020)



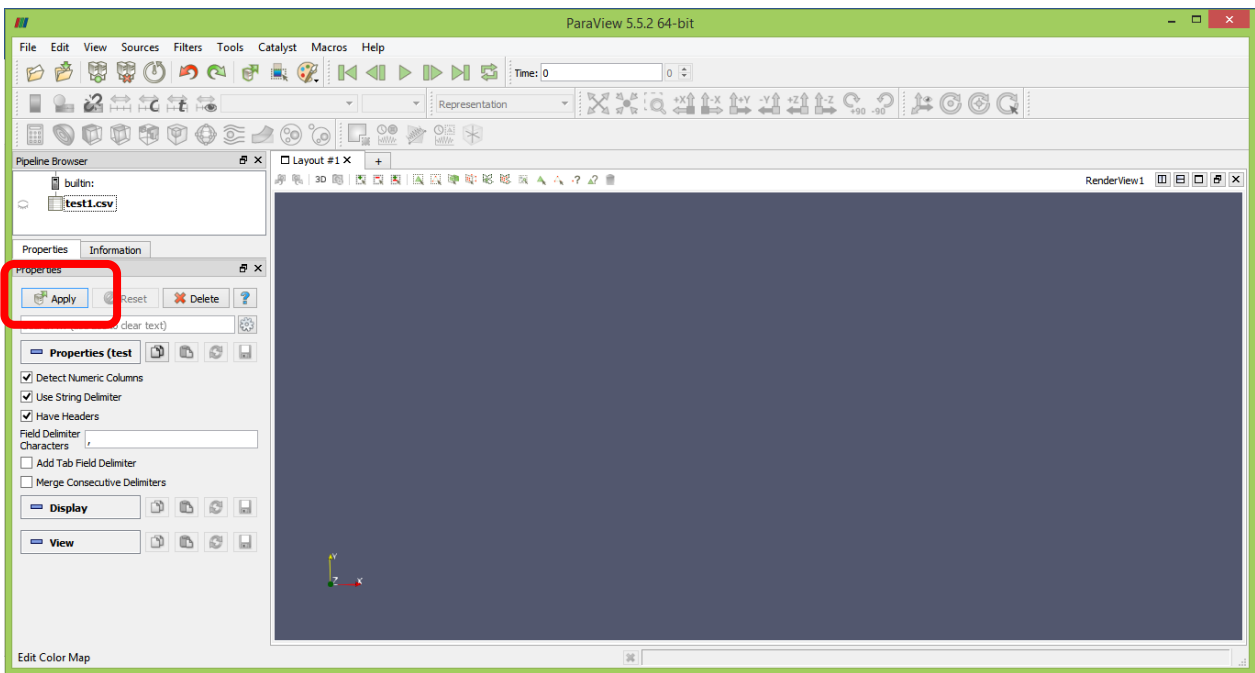
Start Para View, and read in this data. Note that the default settings should be used:

- Detect Numeric Columns ON
- Use String Delimiter ON
- Have Headers ON
- Field Delimiter Characters should be a comma - ','

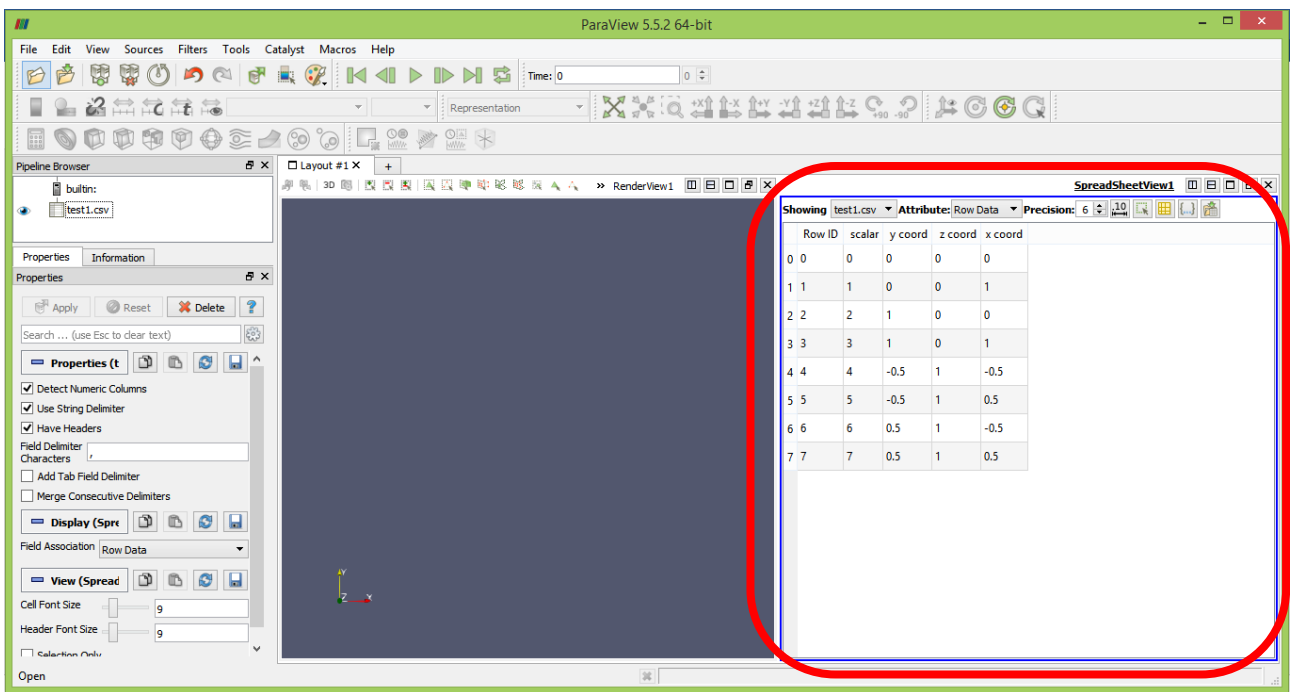
(See figure below)



Then press apply.



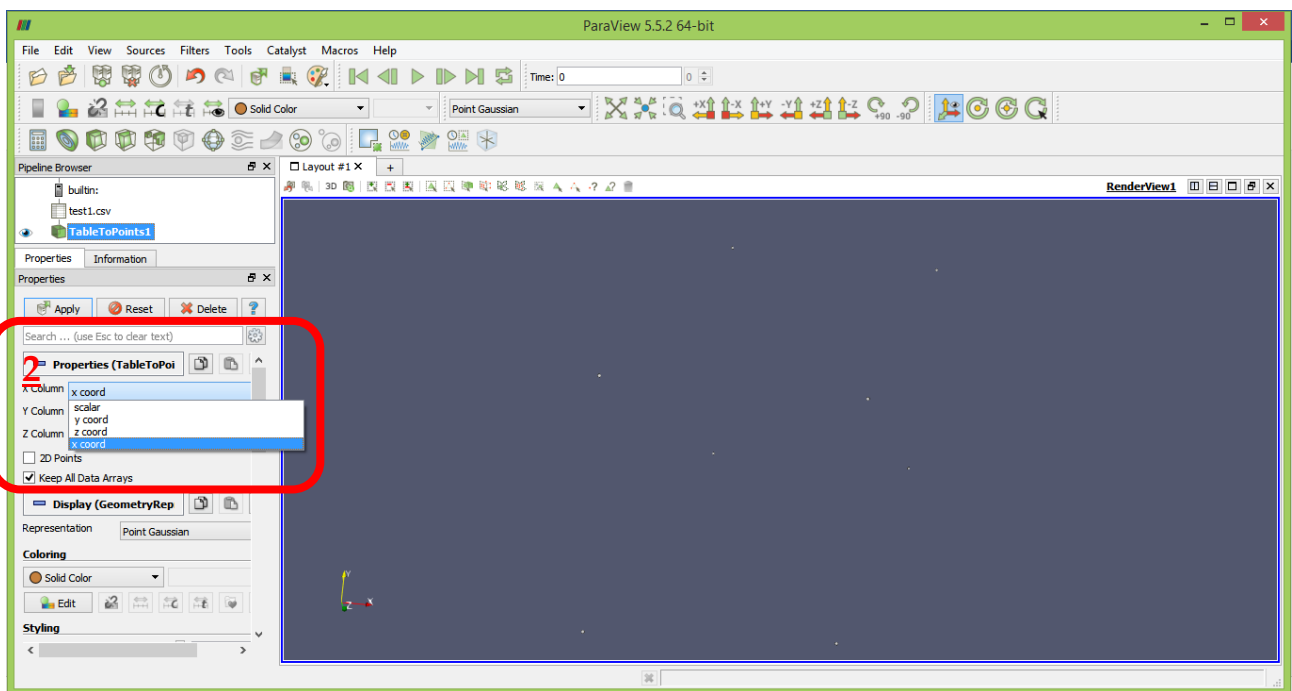
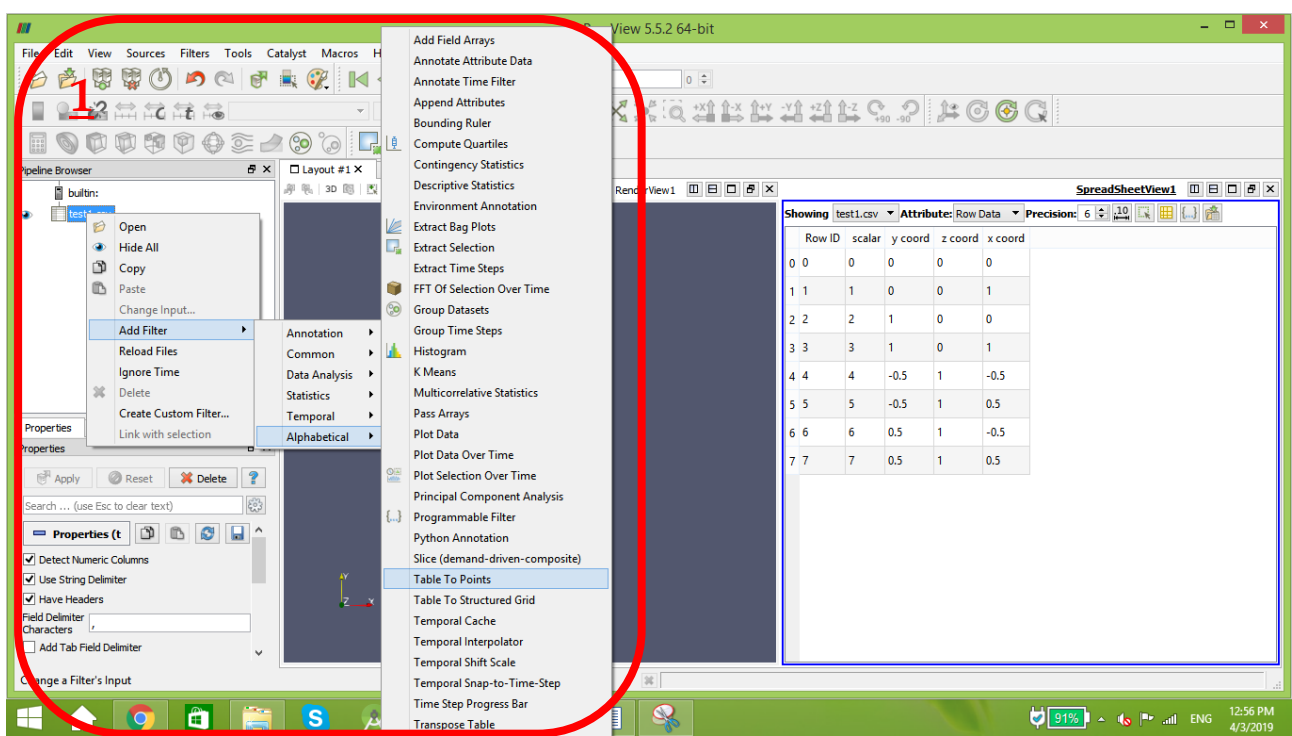
The data should show up as a table.



58.2.1 Displaying data as points

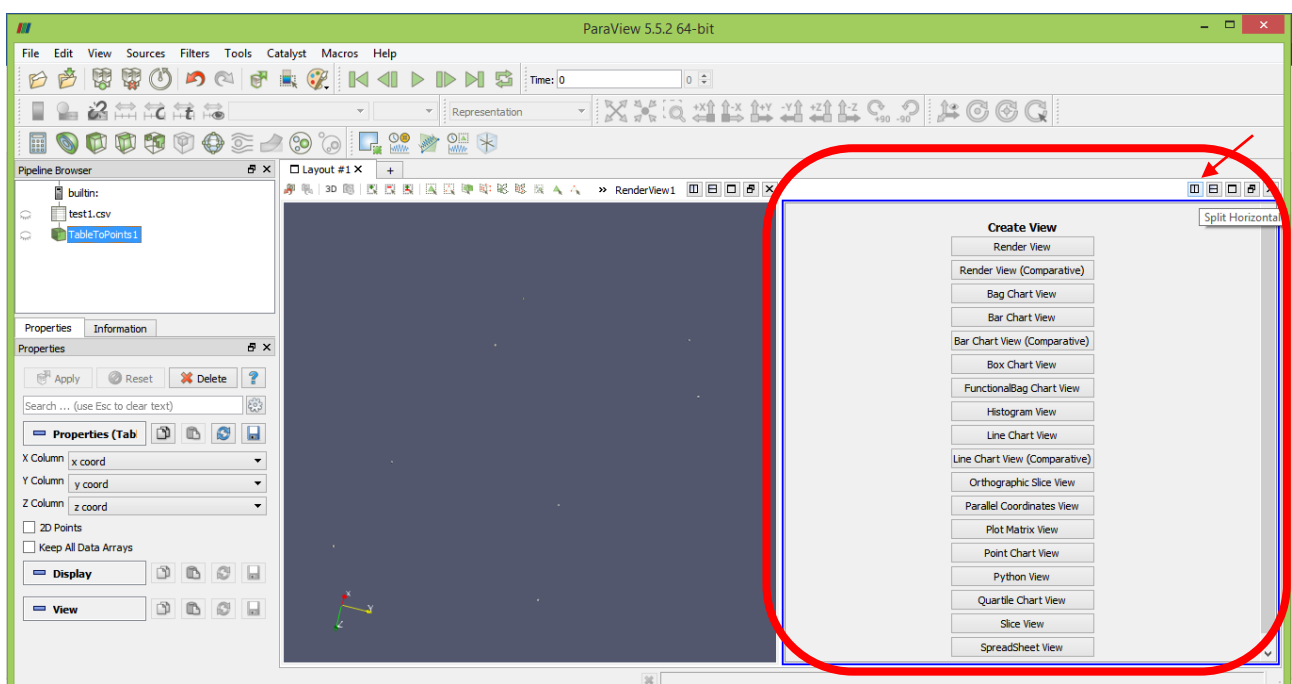
- Run the filter Filters/ Alphabetical/ Table to Points (right click on the table at the left as shown in the figure below).
- Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.

INT-LMIC (Laser Matter Interaction Code) (2020)



Press apply and the points are visible now.

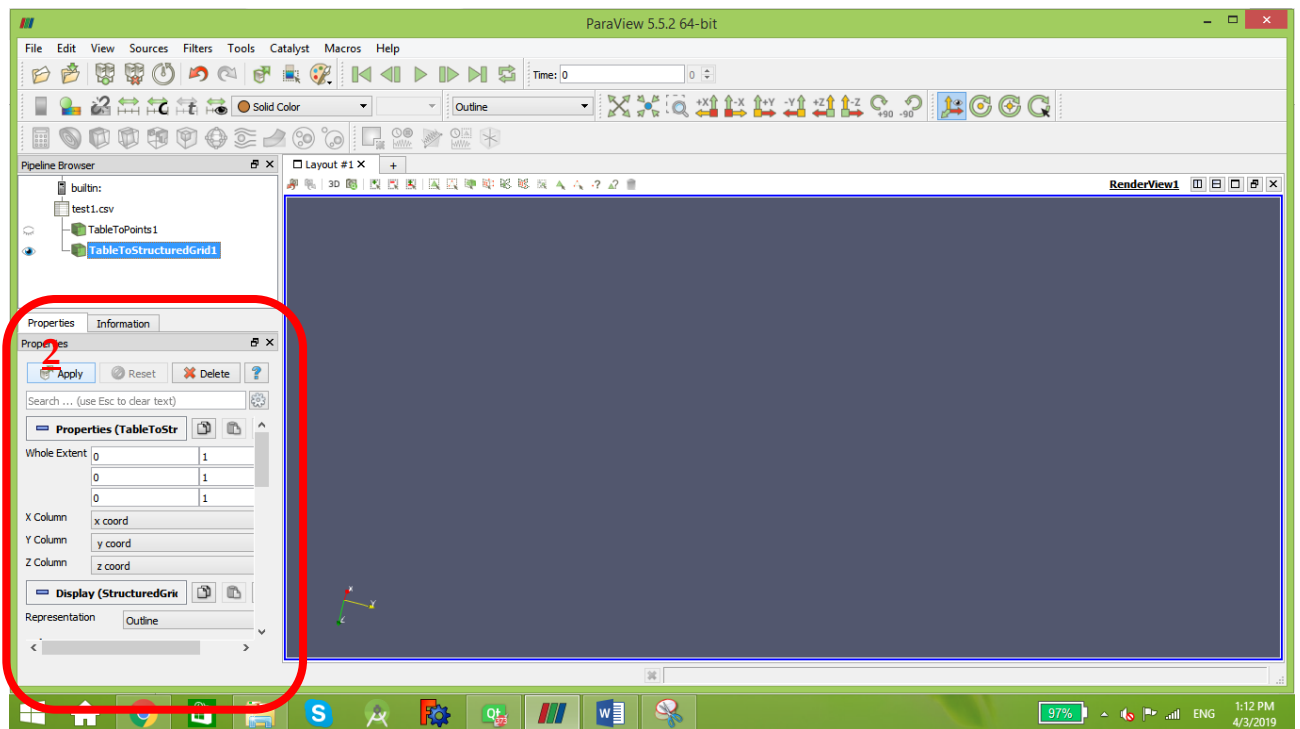
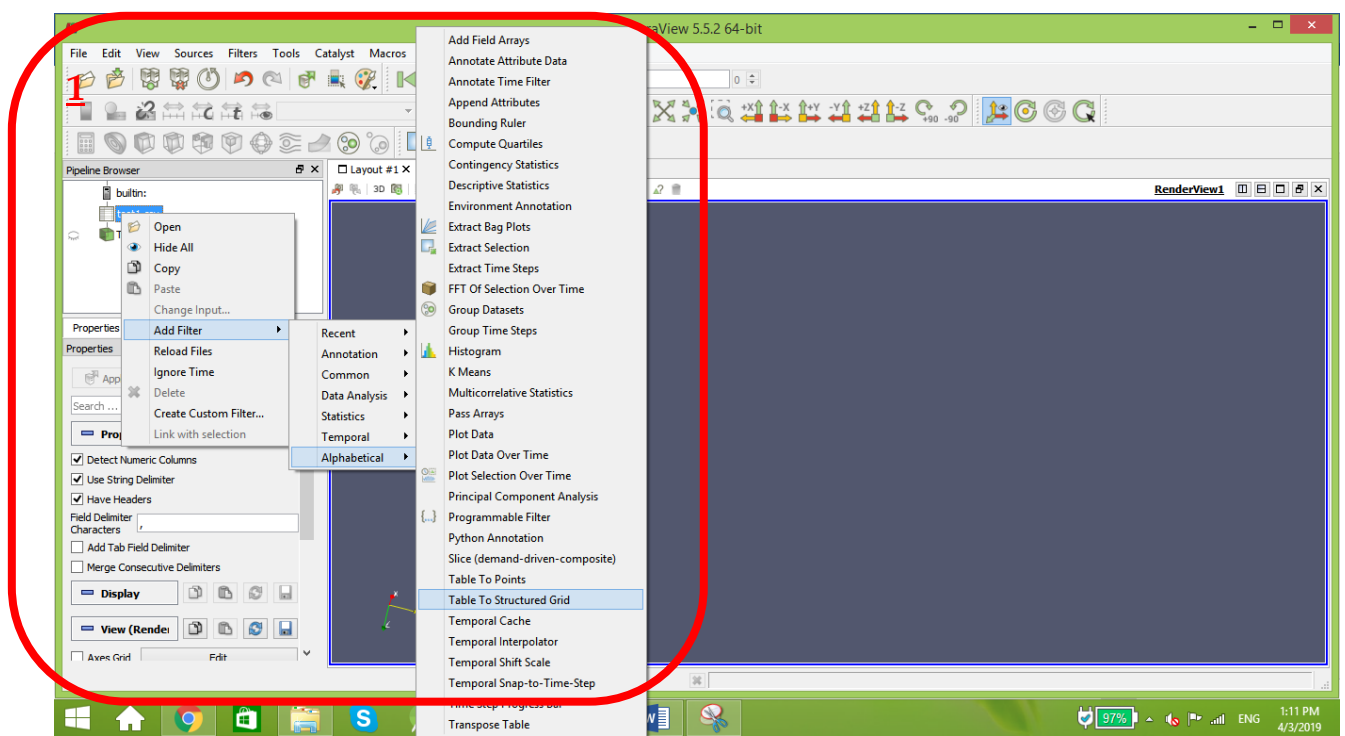
If your points didn't show up press on "split horizontal" button. And choose the desired view.



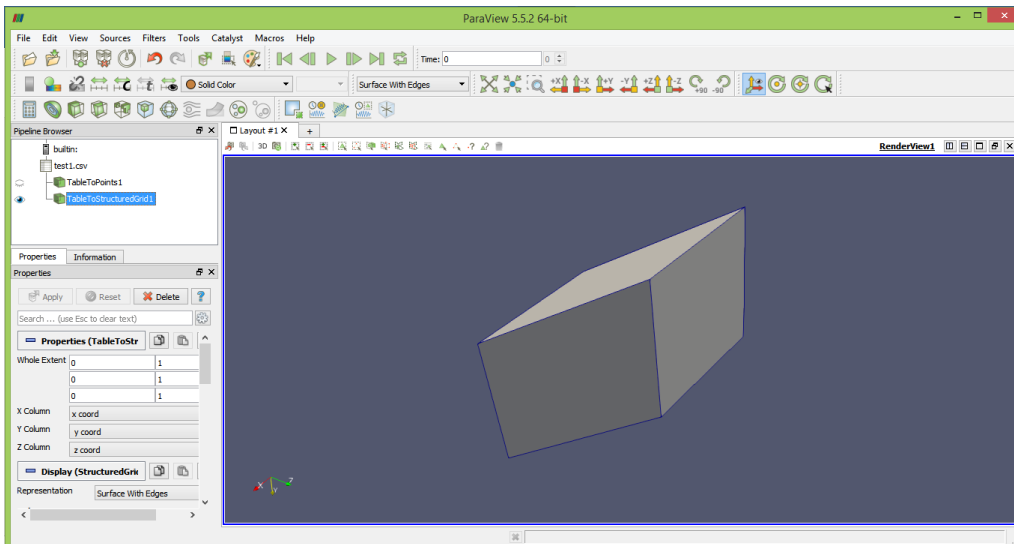
58.2.2 Displaying data as structured grid

4. Run the filter Filters/ Alphabetical/ Table to Structured Grid.
5. Tell Para View what extent, or array sizes, your data is in. For instance, the data above has 8 points, forming a leaning cube. Points arrays are in X == size 2, Y == size 2, and Z == size 2. In this example we will use C indexing for the arrays, thus they go from 0 to 1 (2 entries).
 - Whole extent is as follows:
 - 0 1
 - 0 1
 - 0 1
6. Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.

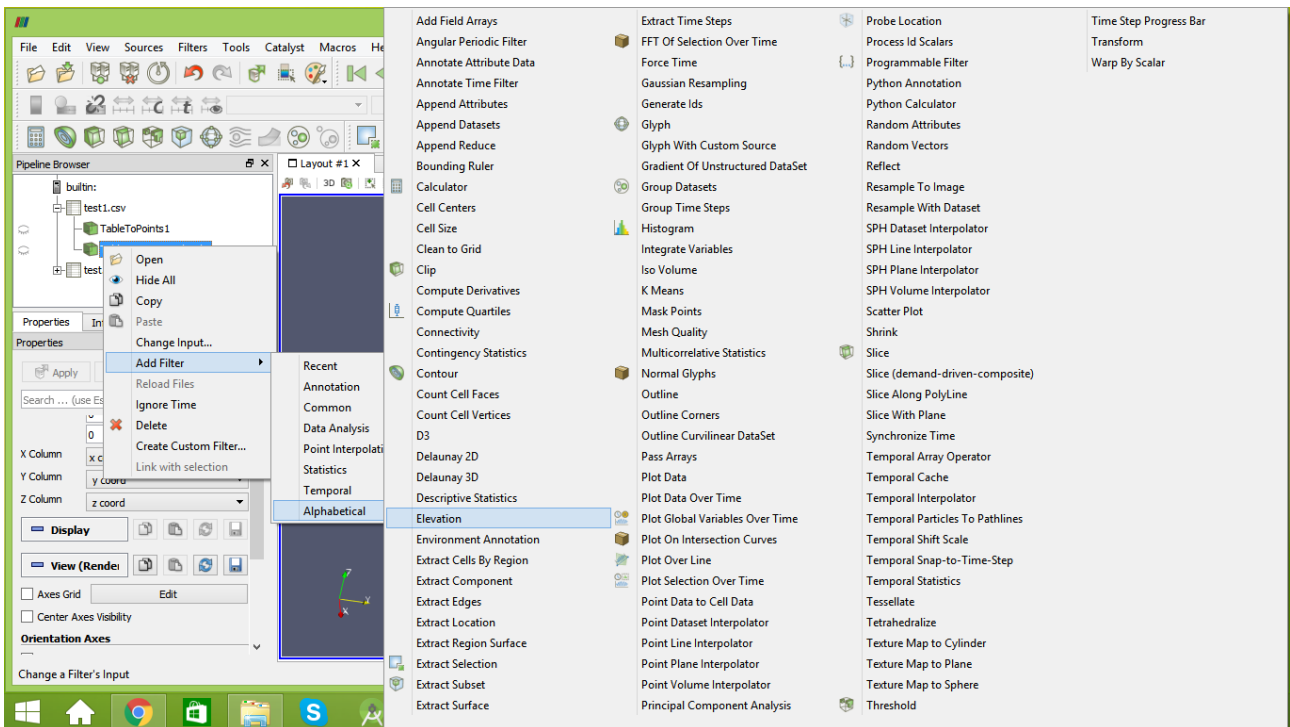
INT-LMIC (Laser Matter Interaction Code) (2020)



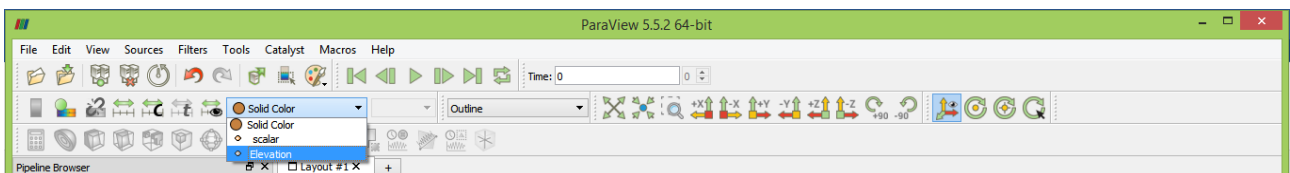
INT-LMIC (Laser Matter Interaction Code) (2020)



Now to represent your results with colors, right click on “table to structure” → add filter → Alphabetic → elevation.

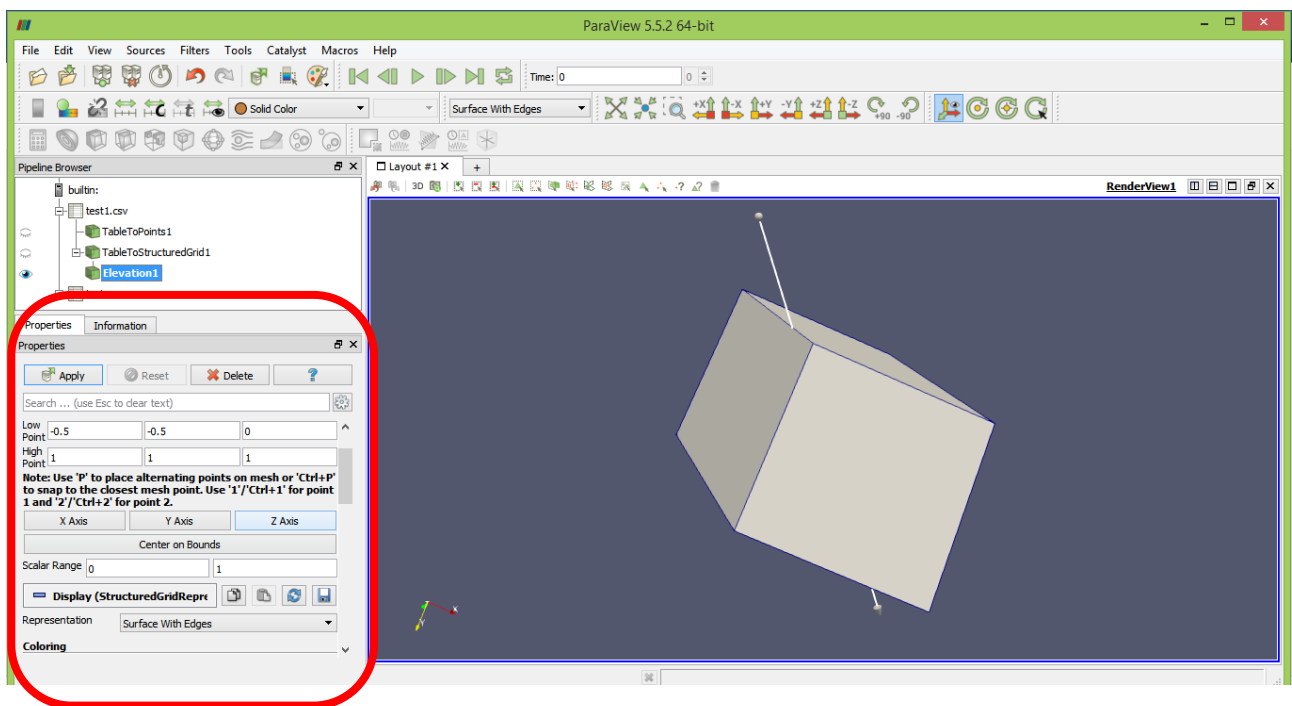


Make sure that you select the elevation button.

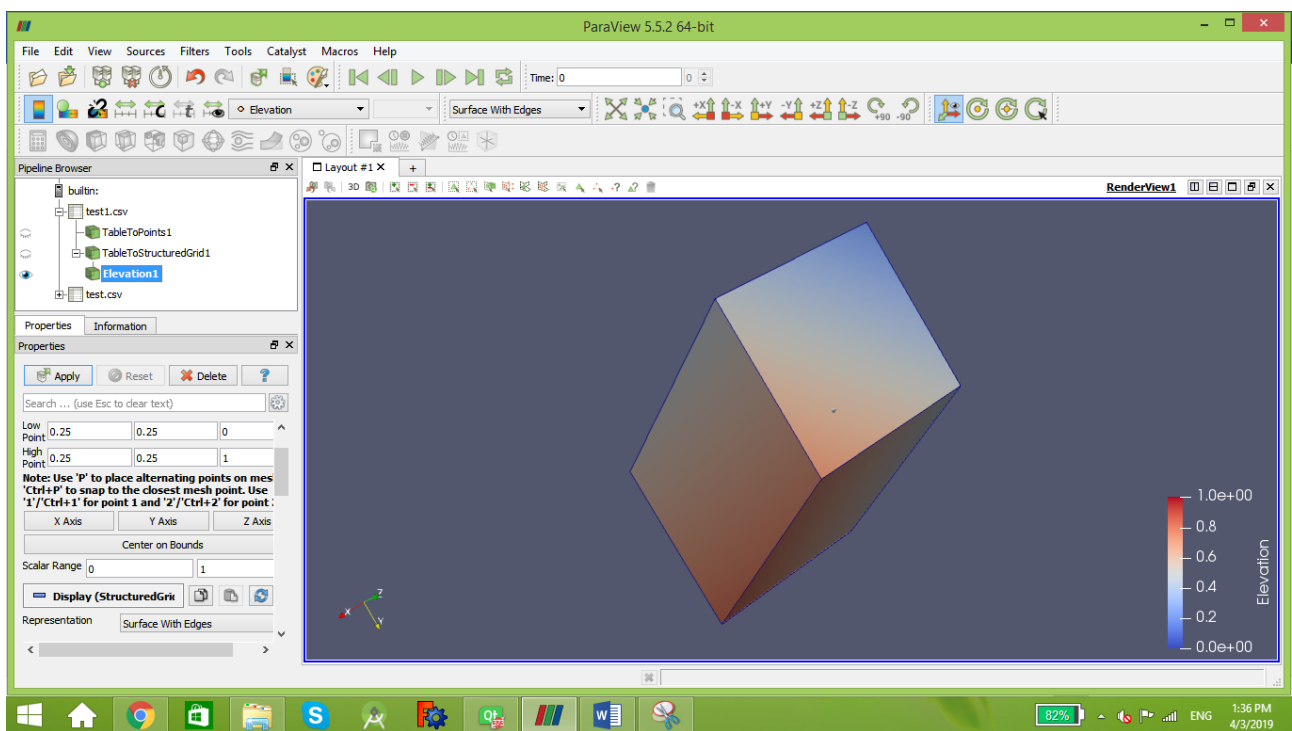


Finally choose the desired axis, then apply.

INT-LMIC (Laser Matter Interaction Code) (2020)



Now it's ready.



58.2.3 Saving Results

A simple code has been added to the program to save the results automatically into files.csv.

```
#include "iostream"  
#include "fstream"  
#include "sstream"
```

```

string filename;

ofstream myfile;

stringstream b;
b<<i;//number of iteration
  filename= "SN_"+ b.str();
  filename+= ".csv";
  myfile.open(filename.c_str());

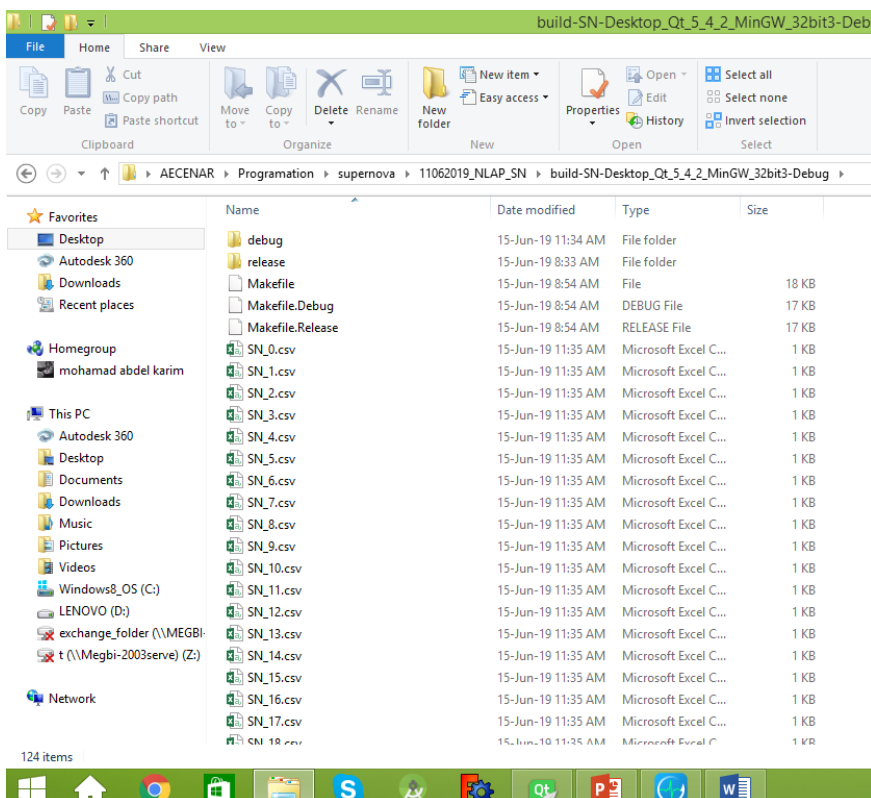
  for(int k=0;k<d;k++){
    rold[k]=r[k];//store the old radii
    accel[k]=dyn.acc(mass[k],r[k],rho[k],dp_over_dr[k],G);
    v[k]=v[k]*damping+accel[k] * dt;
    r[k]=r[k]+v[k] * dt+accel[k] * dt * dt;
    y=sqrt(r[k]*r[k]-(k+1)*(k+1));
    if(r[k]<0){
      r[k]=0;
    }

myfile<<0<<","<<r[k]<<","0,"<<r[k]<<","<<T[k]<<","<<dt<<","<<pressure[k]<<"\n";

  }
myfile.close();

```

This code will create a file named SN000.csv for each iteration. They will be saved in the build folder.

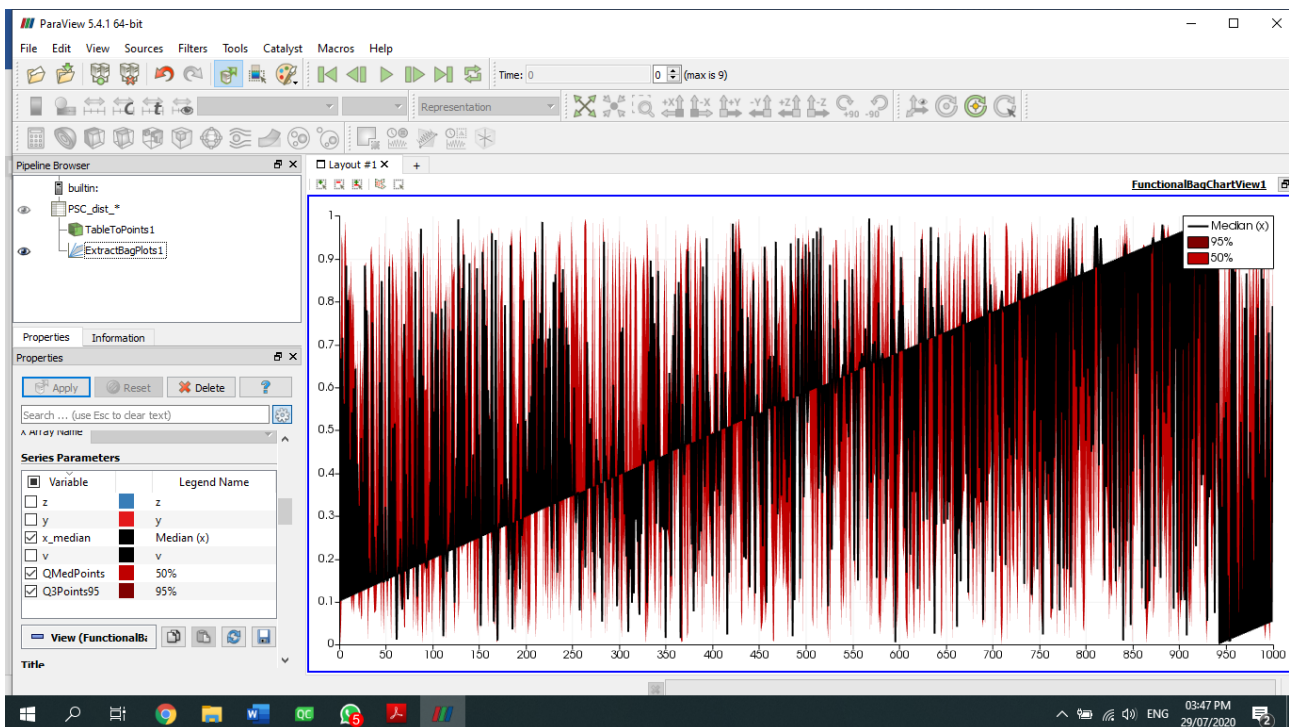
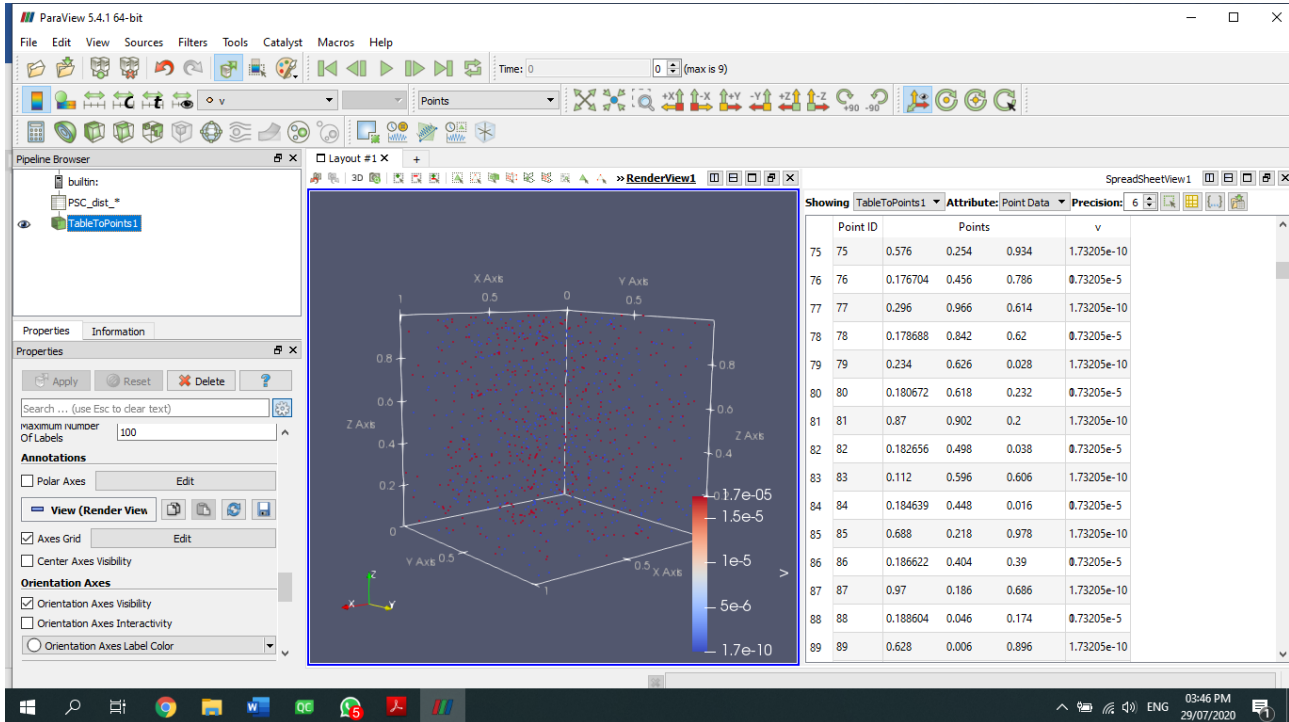


58.3 First Test results

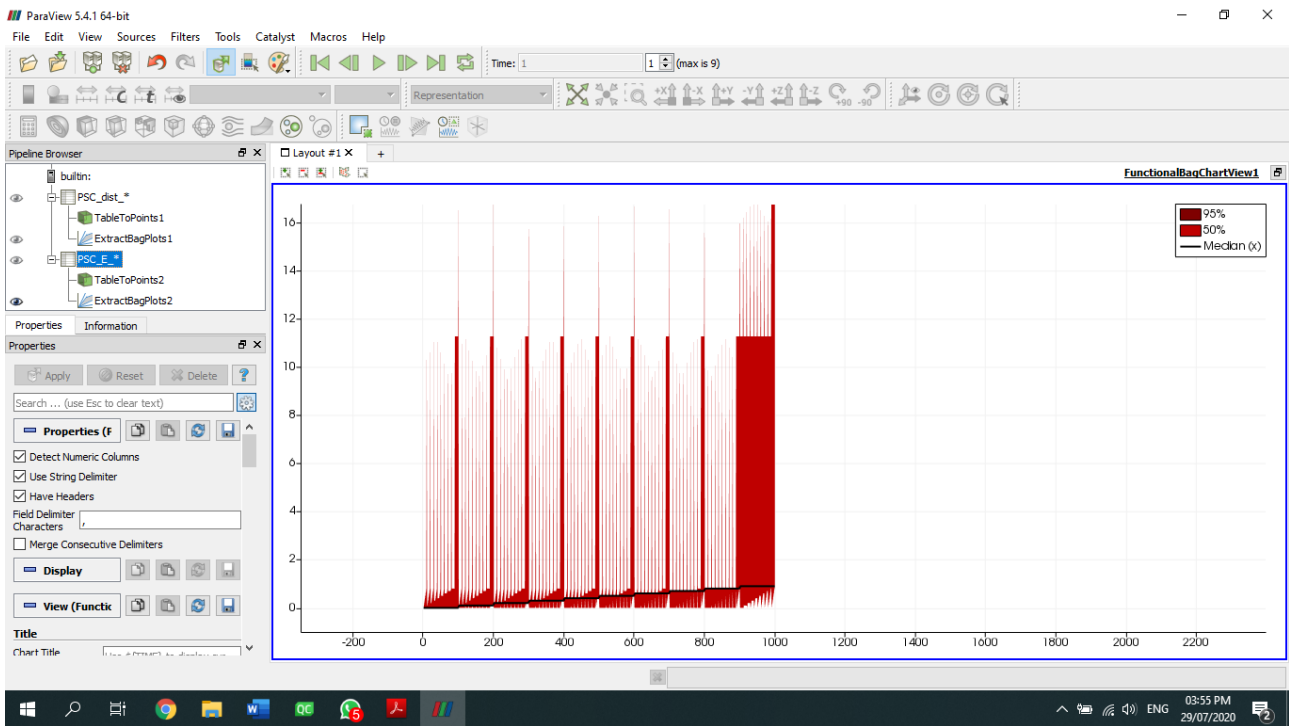
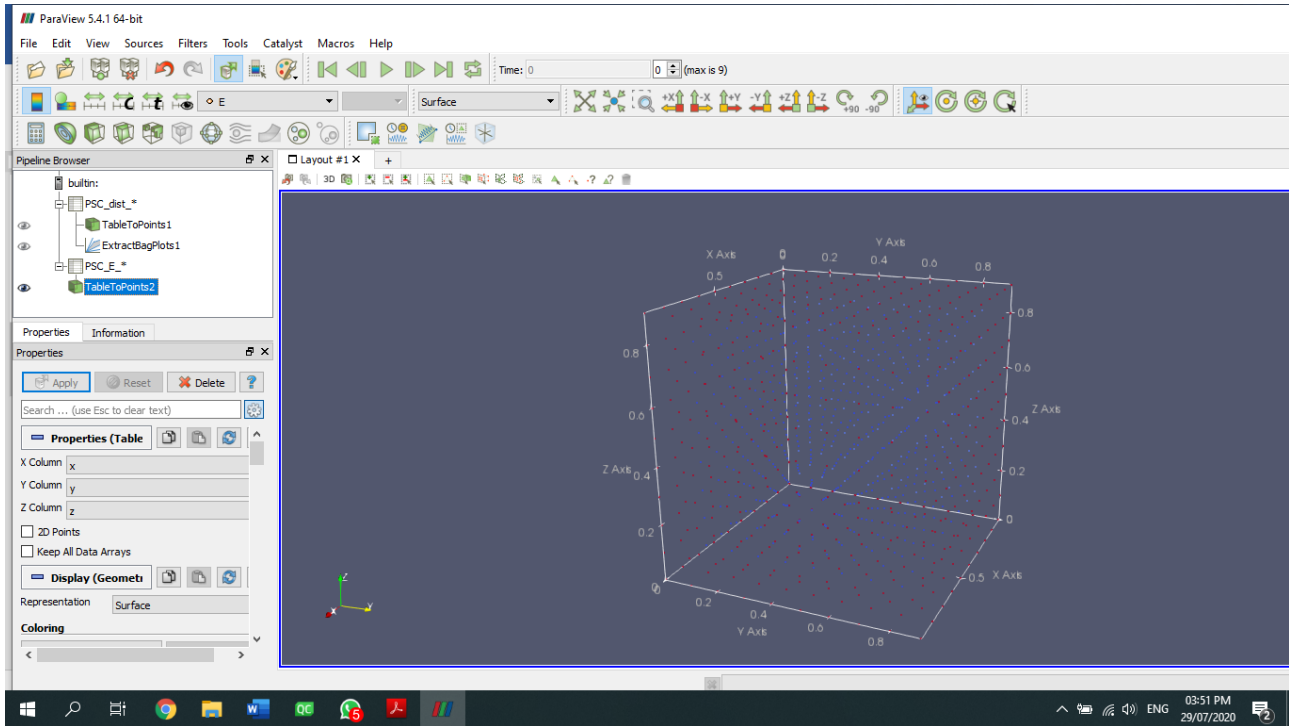
To be done: after implementation of the new code (laser-matter interaction) has to be run and applied with a laser pulse to the four gases

58.3.1 IAP_PSC

DISTRIBUTION DE CHARGE



ELECTRIQUE FIELDS



MAGNETIC FIELD

