بسم الله الرحمن الرحيم

AECENAR
Association for Economical and Technological Cooperation in the Euro-Asian and North-African Region
www.aecenar.com

طاقة الشمال
North Lebanon Alternative Power
www.nlap-lb.com

MEAE - Middle East Alternative Energy Institute
مركز الشرق الاوسط للطاقة البديلة
http://aecenar.com/institutes/meae

Numerical Combustion

برنامج رقمي هدفه حساب ومحاكاة عملية الحرق وديناميكية الموائع الحسابية(CFD)

Last update: Monday, May 13, 2019

Author

Maryam ABDEL-KARIM

Content

مدخل Introduction

استجابة لحاجة المستخدم الكبيرة والتطورات الأخيرة في مجالات ديناميكية السوائل العددية (CFD) ونمذجة عملية الاحتراق (numerical combustion) التي تشتمل 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.

Chapter 1: Basics[1]

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:

$$\boxed{Y_k = \frac{m_k}{m}}$$

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:

$$\boxed{\frac{1}{W} = \sum_{k=1}^{N} \frac{Y_k}{W_k}}$$

The enthalpy is assumed by:

$$h_k = \underbrace{\int_{T_0}^{T} C_{pk} dT}_{\text{sensible}} + \underbrace{\Delta h^o_{f,k}}_{\text{chemical}}$$

T_0=298.15 k (reference temperature).

The formation enthalpies $\Delta h^o_{f,k}$ are the enthalpies needed to form 1 kg of species k at the reference temperature.

[1] "Theoretical and Numerical Combustion" - By Thierry Poinsot, Denis Veynante

Substance	Molecular weight W_k (kg/mole)	Mass formation enthalpy $\Delta h^o_{f,k}$ (kJ/kg)	Molar formation enthalpy $\Delta h^{o,m}_{f,k}$ (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 $T_0 = 298.15\,K$.

The heat capacities at constant pressure of species k (C_{pk}) are

$$C^m_{pk} = 3.5R \quad \text{and} \quad C_{pk} = 3.5R/W_k$$

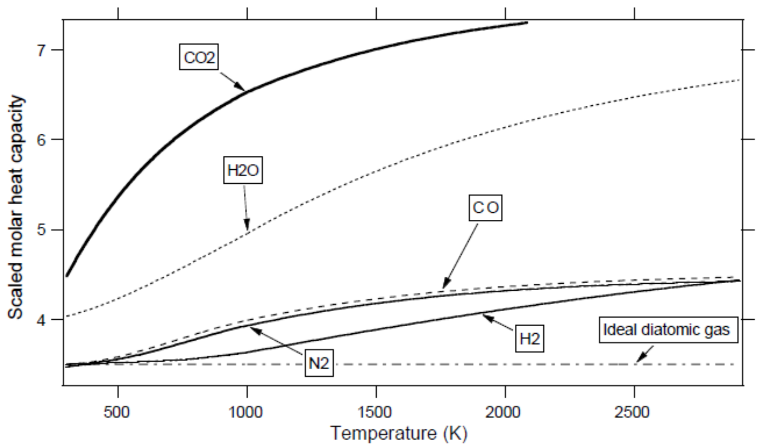


Figure 1.1: Scaled molar heat capacities at constant pressure C^m_{pk}/R of CO_2, CO, H_2O, H_2 and N_2.

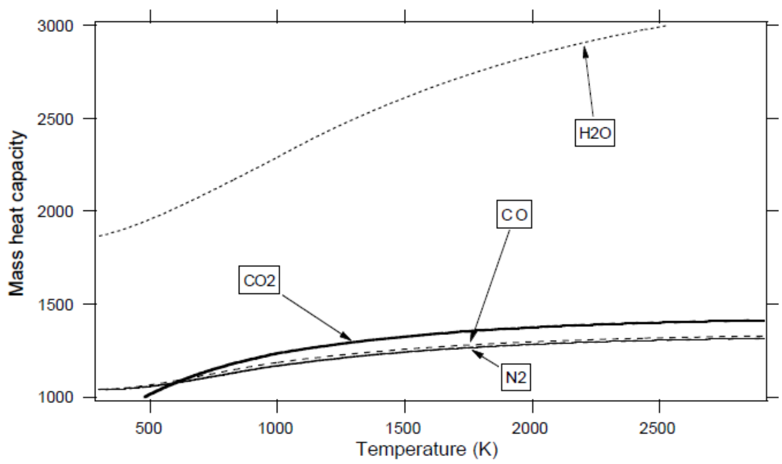


Figure 1.2: Mass heat capacities (J/(kgK)) at constant pressure C_{pk} of CO_2, CO, H_2O and N_2.

The mass heat capacities C_{vk} at constant volume are related to the C_{pk} by:

$$C_{pk} - C_{vk} = R/W_k$$

1.1. 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	$e = h - p/\rho = e_s + \sum_{k=1}^{N} \Delta h_{f,k}^o Y_k$	$h = h_s + \sum_{k=1}^{N} \Delta h_{f,k}^o Y_k$
Total Chemical	$e_t = h_t - p/\rho = e_s + \sum_{k=1}^{N} \Delta h_{f,k}^o Y_k + \frac{1}{2} u_i u_i$	$h_t = h_s + \sum_{k=1}^{N} \Delta h_{f,k}^o Y_k + \frac{1}{2} u_i u_i$
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 $-\sum_{k=1}^{N} \Delta h_{f,k}^o \dot{\omega}_k$. For any energy or enthalpy f: $\rho \frac{Df}{Dt} = \rho(\frac{\partial f}{\partial t} + u_i \frac{\partial f}{\partial x_i}) = \frac{\partial \rho f}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i f)$.

1.2. 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:

$$\sigma_{ij} = \tau_{ij} - p\delta_{ij} = -p\delta_{ij} - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

1.3. Chemical kinetics

Consider a chemical system of N species reacting through M reactions:

$$\sum_{k=1}^{N} \nu'_{kj} \mathcal{M}_k \rightleftharpoons \sum_{k=1}^{N} \nu''_{kj} \mathcal{M}_k \quad \text{for} \quad j = 1, M$$

For simplicity, only mass reaction rates are used. For species k, this rate is the sum of rates produced by all M reactions:

$$\boxed{\dot{\omega}_k = \sum_{j=1}^{M} \dot{\omega}_{kj} = W_k \sum_{j=1}^{M} \nu_{kj} Q_j} \quad \text{with} \quad \frac{\dot{\omega}_{kj}}{W_k \nu_{kj}} = Q_j$$

The progress rate Q_j of reaction j is written:

$$\boxed{Q_j = K_{fj} \Pi_{k=1}^{N} \left(\frac{\rho Y_k}{W_k}\right)^{\nu'_{kj}} - K_{rj} \Pi_{k=1}^{N} \left(\frac{\rho Y_k}{W_k}\right)^{\nu''_{kj}}}$$

Where Kfj and Krj 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 H2-O2 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 Afj in cgs units, βj and Ej in cal/mole. The backwards rates Krj are computed from the forward rates through the equilibrium constants:

$$\boxed{K_{rj} = \frac{K_{fj}}{\left(\frac{p_a}{RT}\right)^{\sum_{k=1}^{N} \nu_{kj}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right)}}$$

```
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).

2. Reacting flow conservation equations

The government conservation equations for reacting flow are shown below:

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: $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i}(\rho(u_i + V_i^c)Y_k) = \frac{\partial}{\partial x_i}(\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k$ and $V_i^c = \sum_{k=1}^N D_k \frac{\partial Y_k}{\partial x_i}$
Momentum $\frac{\partial}{\partial t}\rho u_j + \frac{\partial}{\partial x_i}\rho u_i u_j = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho \sum_{k=1}^N Y_k f_{k,j}$
Energy (sum of sensible and kinetic) $\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j}(\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^N Y_k f_{k,i}(u_i + V_{k,i})$ with $\dot{\omega}_T = -\sum_{k=1}^N \Delta h_{f,k}^o \dot{\omega}_k$ and $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^N h_k Y_k V_{k,i}$

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: $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i}(\rho(u_i + V_i^c)Y_k) = \frac{\partial}{\partial x_i}(\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k$ 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) $\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i}$ 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.

3. 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 Non-reacting	NAVIER STOKES Non-reacting	NAVIER STOKES 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.

3.1. 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 ϒ 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:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial}{\partial x_i}[u_i(\rho E + p)] = -\frac{\partial}{\partial x_i}(q_i) + \frac{\partial}{\partial x_i}(u_i \tau_{ij}) + \dot{\omega}_T$$

$$\frac{\partial (\rho u_j)}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i u_j) + \frac{\partial p}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_i} \quad \text{for} \quad \mathrm{i} = 1,3$$

$$\frac{\partial (\rho Y_k)}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i Y_k) = \frac{\partial}{\partial x_i}(M_{ki}) - \dot{\omega}_k \qquad \text{for} \quad \mathrm{k} = 1, \mathrm{N}$$

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:

$$q_i = -\lambda \frac{\partial T}{\partial x_i} \quad \text{and} \quad M_{ki} = \rho D_k \frac{\partial Y_k}{\partial x_i}$$

3.2. Comparison between NSCBC implementation for Euler and Navier-Stocks

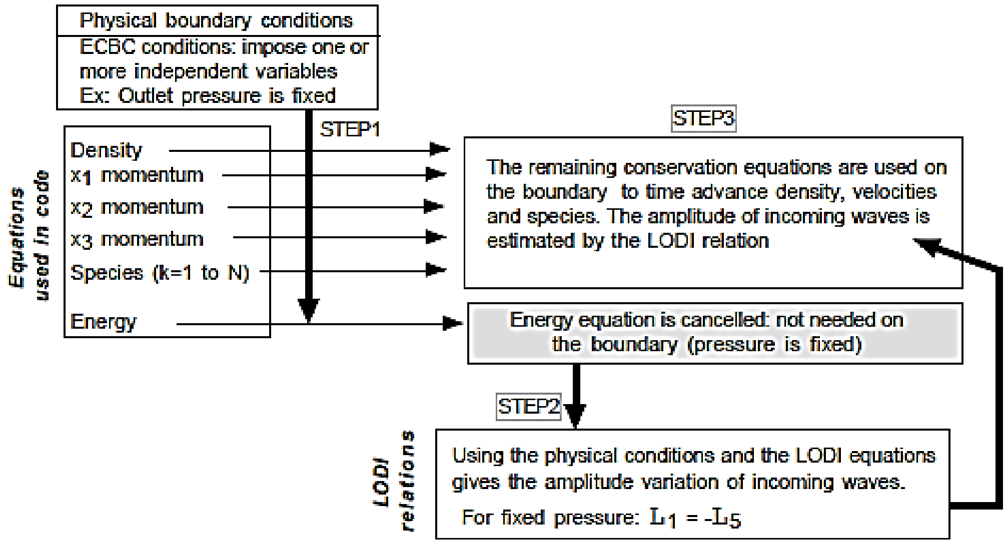


Figure 9.3: NSCBC implementation for Euler equations. Example for a fixed pressure outlet.

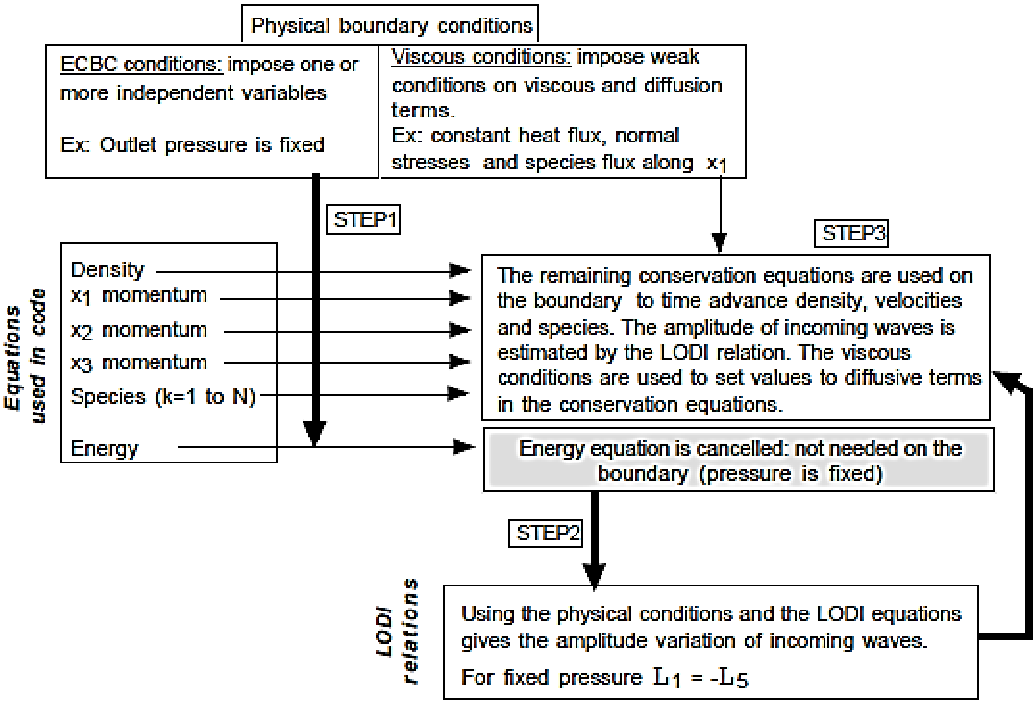


Figure 9.4: NSCBC implementation for Navier-Stokes equations. Example for a fixed pressure outlet.

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	0	0	4+N
SI-2	u_i, ρ, Y_k imposed		u_i, ρ, Y_k imposed	$\frac{\partial \tau_{11}}{\partial x_1} = 0$	No	
		4+N	4+N	1	0	5+N
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	
		4+N	4+N	1	0	5+N
SI-4	No reflected wave		No reflected wave	$\frac{\partial \tau_{11}}{\partial x_1} = 0$	No	
		4+N	4+N	1	0	5+N

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	$\frac{\partial \tau_{12}}{\partial x_1} = 0$ $\frac{\partial \tau_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	
			1	1	3	N	4+N
B3	Partially non reflecting outflow	P infinity imposed		P at infinity imposed	$\frac{\partial \tau_{12}}{\partial x_1} = 0$ $\frac{\partial \tau_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	
			1	1	3	N	4+N
B4	Subsonic reflecting outflow	P outlet imposed		P outlet imposed	$\frac{\partial \tau_{12}}{\partial x_1} = 0$ $\frac{\partial \tau_{13}}{\partial x_1} = 0$ $\frac{\partial q_1}{\partial x_1} = 0$	$\frac{\partial M_{k1}}{\partial x_1} = 0$	
			1	1	3	N	4+N
NSW	Isothermal no slip wall			$u_i = 0$ T imposed		$M_{k1} = 0$	
				4	0	N	4+N
ASW	Adiabatic slip wall			Zero normal velocity	$q_1 = 0$	$M_{k1} = 0$	
				3	1	N	4+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.

3.3. 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)

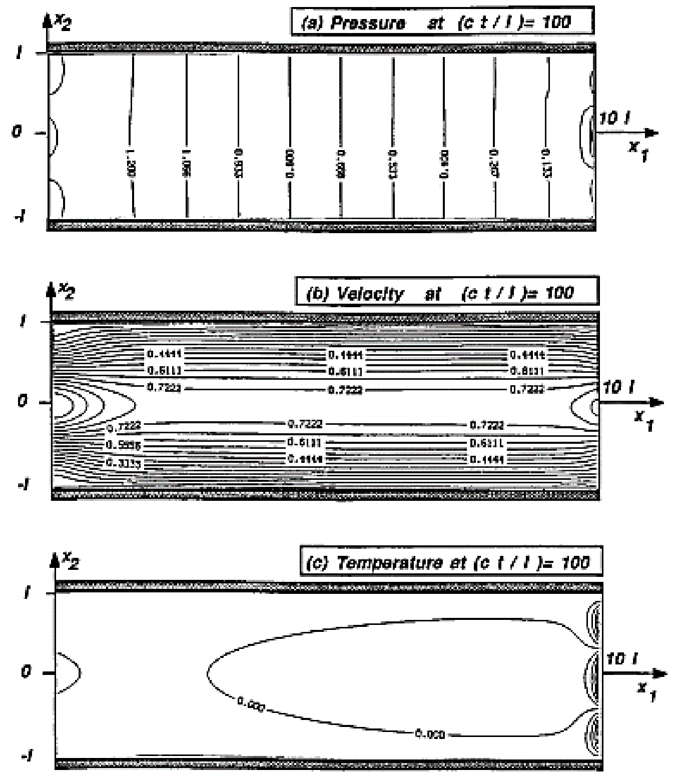


Figure 9.24: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet condition B1.

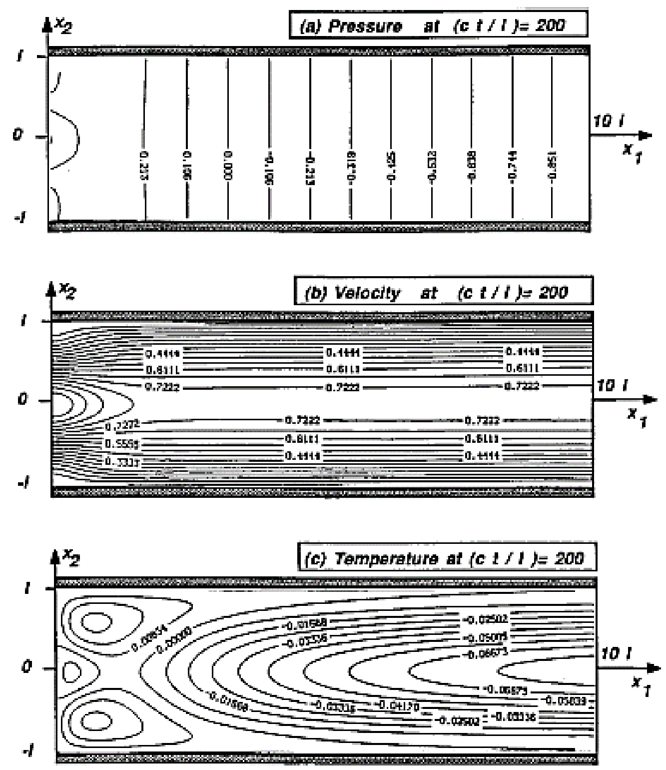


Figure 9.26: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet NSCBC condition B3 ($\sigma = 0.15$).

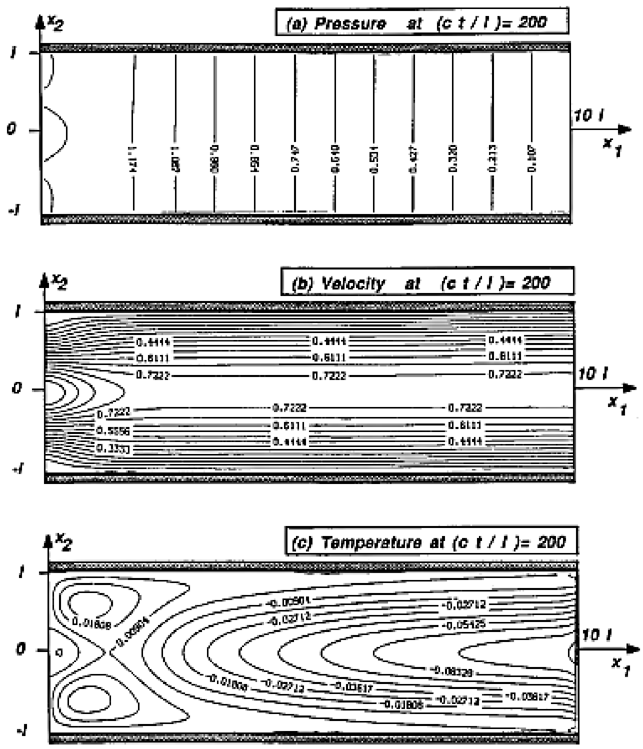


Figure 9.27: Steady state pressure, velocity and temperature fields for the Poiseuille flow with outlet NSCBC condition B4.

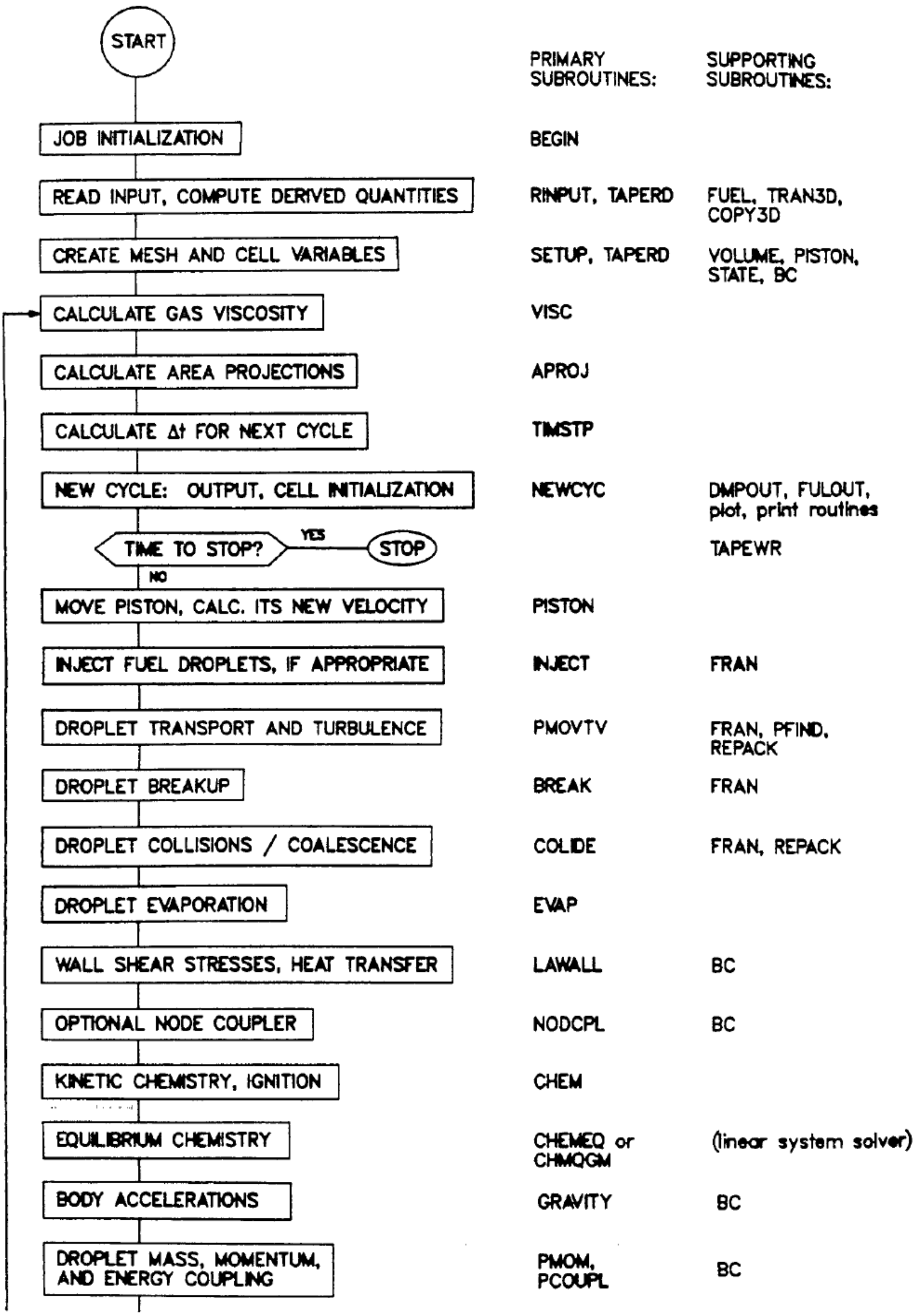
Conservation equation Comparison between KIVAII and Poinsot .4

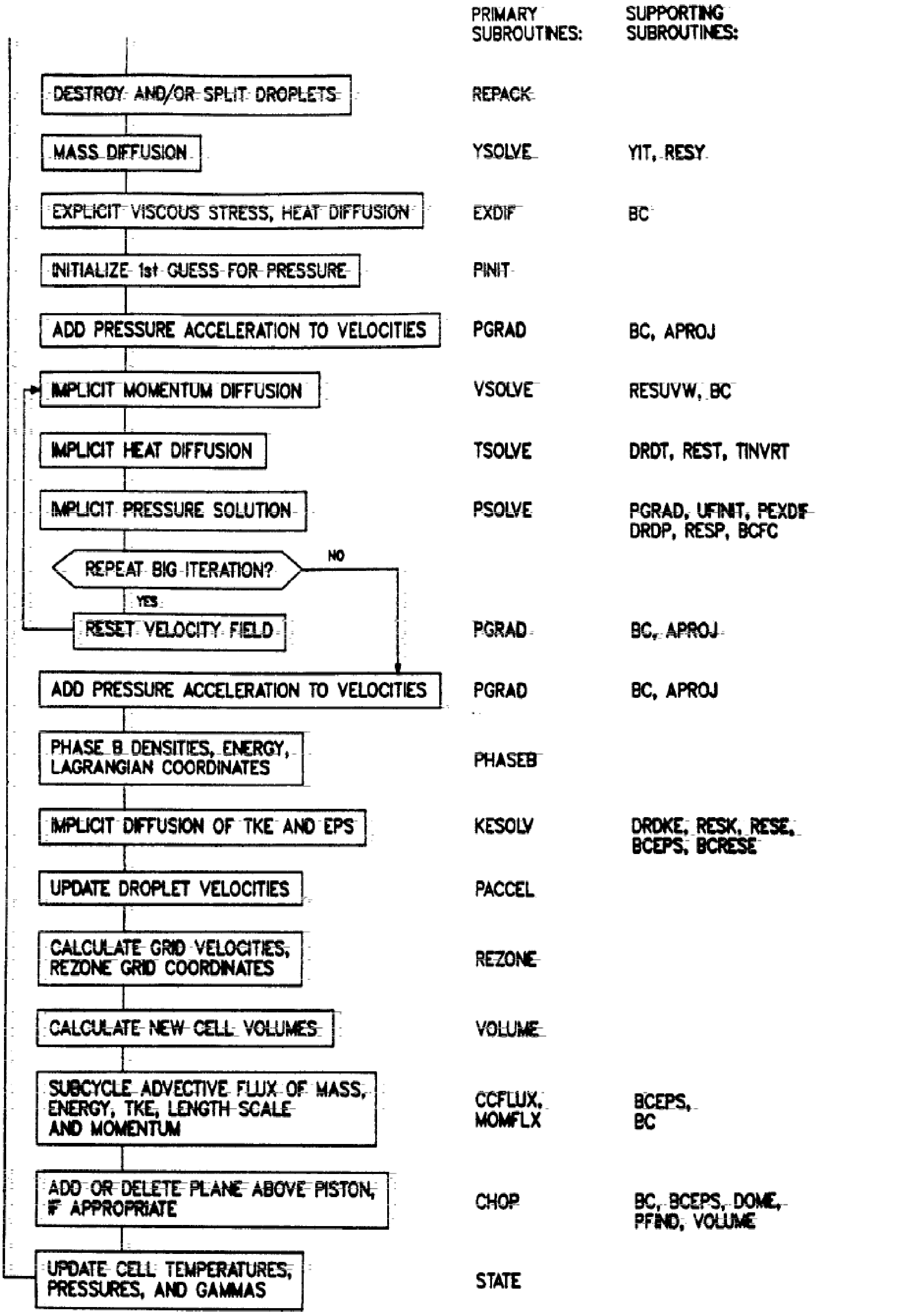
Government equation	Computational fluid dynamics	KIVA II Program	Poinsot
Mass conservation	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0$	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \dot{\rho}^s$	$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0$ 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: $\frac{\partial \rho Y_k}{\partial t} + \frac{\partial}{\partial x_i}(\rho(u_i + V_i^c)Y_k) = \frac{\partial}{\partial x_i}(\rho D_k \frac{\partial Y_k}{\partial x_i}) + \dot{\omega}_k$ and $V_i^c = \sum_{k=1}^{N} D_k \frac{\partial Y_k}{\partial x_i}$
Momentum equation	$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x$ $\frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y$ $\frac{\partial(\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z$	$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\frac{1}{a^2}\nabla p - A_0 \nabla (2/3\, \rho k) + \nabla \cdot \sigma + \mathbf{F}^s + \rho \mathbf{g}$	$\frac{\partial}{\partial t}\rho u_j + \frac{\partial}{\partial x_i}\rho u_i u_j = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho \sum_{k=1}^{N} Y_k f_{k,j}$
Energy equation	$\frac{\partial(\rho e)}{\partial t} + \nabla . (\rho e \vec{V}) = \rho \dot{q} + \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k \frac{\partial T}{\partial z}\right)$ $- p\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right) + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)^2$ $+ \mu\left[2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2\right.$ $\left. + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)^2\right]$	$\frac{\partial(\rho I)}{\partial t} + \nabla \cdot (\rho \mathbf{u} I) = -p \nabla \cdot \mathbf{u} + (1 - A_0)\sigma : \nabla \mathbf{u} - \nabla \cdot \mathbf{J} + A_0 \rho \varepsilon + \dot{Q}^c + \dot{Q}^s$ $\mathbf{J} = -K \nabla T - \rho D \sum_m h_m \nabla (\rho_m/\rho)$	$\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i E) = \dot{\omega}_T - \frac{\partial q_i}{\partial x_i} + \frac{\partial}{\partial x_j}(\sigma_{ij} u_i) + \dot{Q} + \rho \sum_{k=1}^{N} Y_k f_{k,i}(u_i + V_{k,i})$ with $\dot{\omega}_T = -\sum_{k=1}^{N} \Delta h_{f,k}^o \dot{\omega}_k$ and $q_i = -\lambda \frac{\partial T}{\partial x_i} + \rho \sum_{k=1}^{N} h_k Y_k V_{k,i}$
Chemical kinetics		$\dot{\omega}_r = k_{fr} \prod_m (\rho_m/W_m)^{a'_{mr}} - k_{br} \prod_m (\rho_m/W_m)^{b'_{mr}}$	$\dot{\omega}_k = \sum_{j=1}^{M} \dot{\omega}_{kj} = W_k \sum_{j=1}^{M} \nu_{kj} Q_j$ $Q_j = K_{fj} \Pi_{k=1}^{N} \left(\frac{\rho Y_k}{W_k}\right)^{\nu'_{kj}} - K_{rj} \Pi_{k=1}^{N} \left(\frac{\rho Y_k}{W_k}\right)^{\nu''_{kj}}$

Chapter 2: مخطط التدفق العام لبرنامج KIVA II

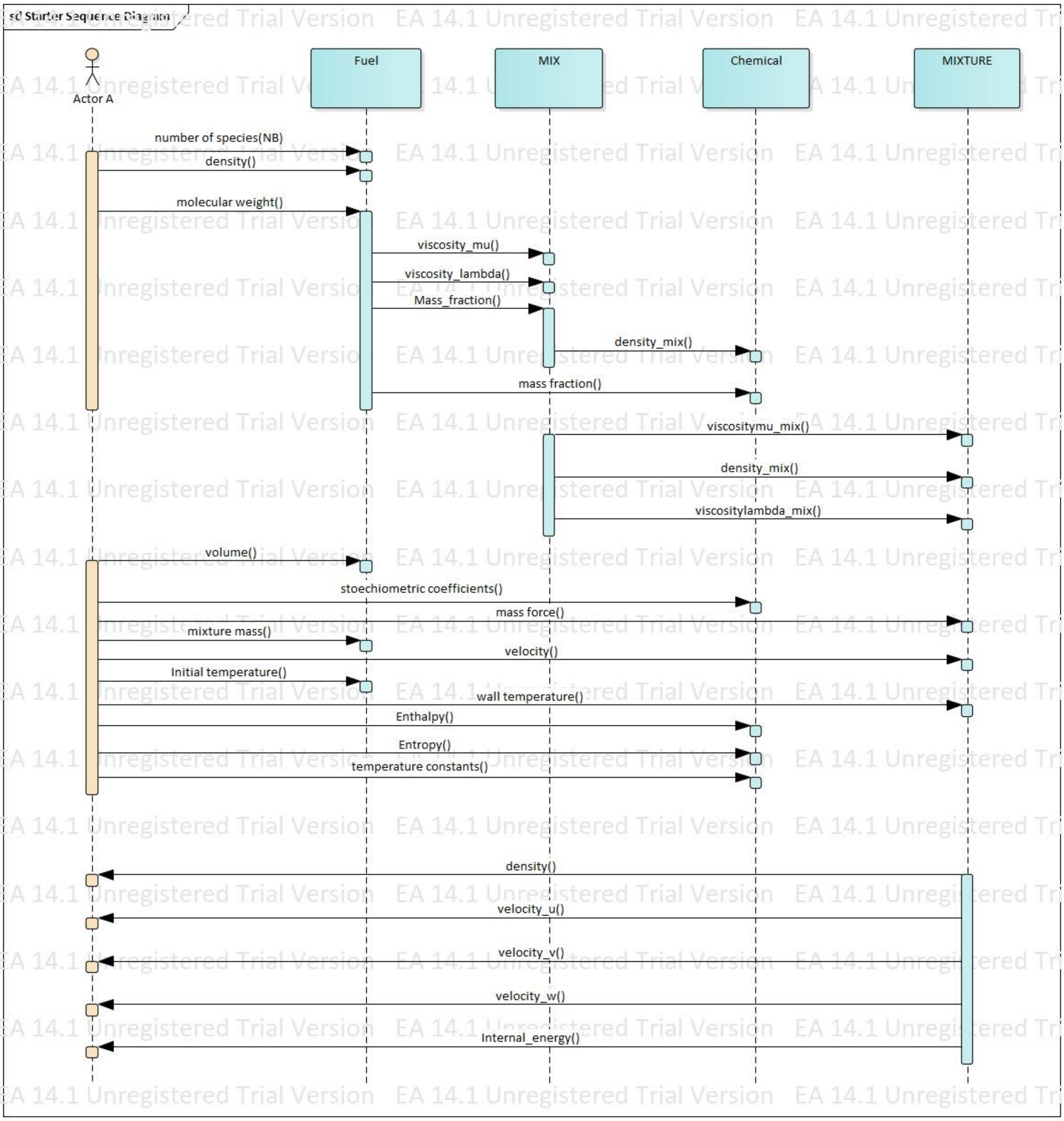
Our diagram is inspired from the KIVA II Diagrams

استوحي النموذج المعتمد في هذا الملف من البرامج التالية: KIVA II

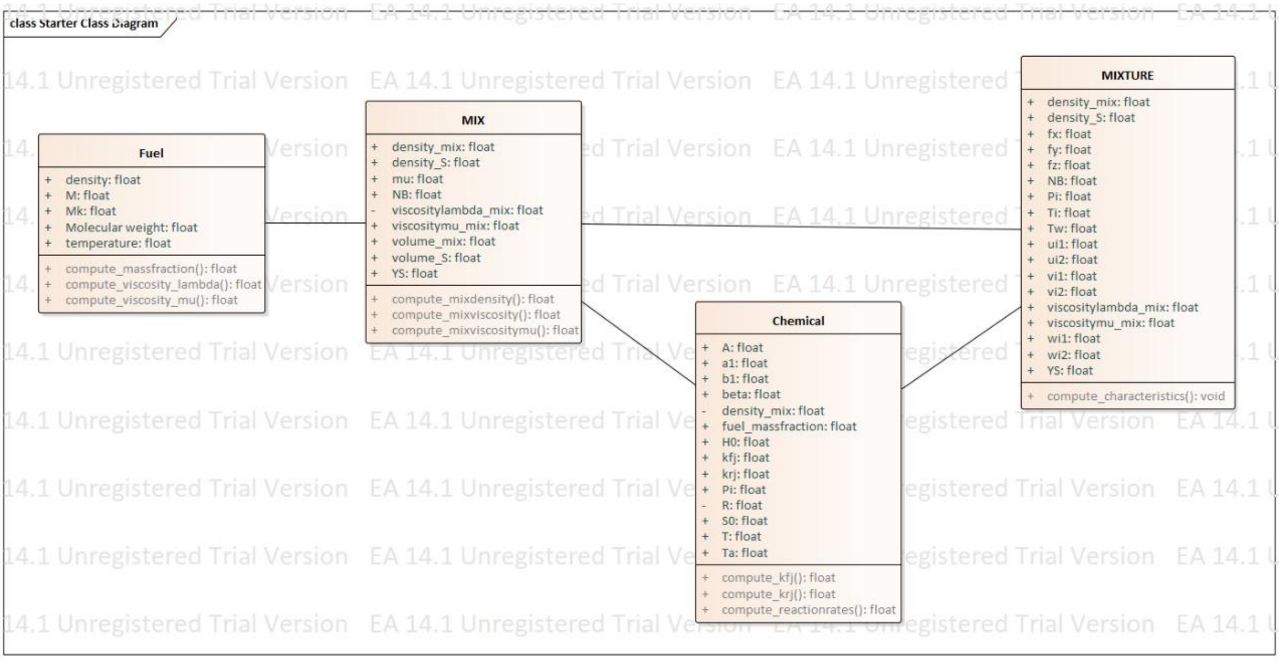



PRIMARY SUBROUTINES:
SUPPORTING SUBROUTINES:
DESTROY AND/OR SPLIT DROPLETS
REPACK
MASS DIFFUSION
YSOLVE
YIT, RESY
EXPLICIT VISCOUS STRESS, HEAT DIFFUSION
EXDIF
BC
INITIALIZE 1st GUESS FOR PRESSURE
PINIT
ADD PRESSURE ACCELERATION TO VELOCITIES
PGRAD
BC, APROJ
IMPLICIT MOMENTUM DIFFUSION
VSOLVE
RESUVW, BC
IMPLICIT HEAT DIFFUSION
TSOLVE
DRDT, REST, TINVRT
IMPLICIT PRESSURE SOLUTION
PSOLVE
PGRAD, UFINIT, PEXDIF
DRDP, RESP, BCFC
REPEAT BIG ITERATION?
NO
YES
RESET VELOCITY FIELD
PGRAD
BC, APROJ
ADD PRESSURE ACCELERATION TO VELOCITIES
PGRAD
BC, APROJ
PHASE B DENSITIES, ENERGY, LAGRANGIAN COORDINATES
PHASEB
IMPLICIT DIFFUSION OF TKE AND EPS
KESOLV
DRDKE, RESK, RESE,
BCEPS, BCRESE
UPDATE DROPLET VELOCITIES
PACCEL
CALCULATE GRID VELOCITIES, REZONE GRID COORDINATES
REZONE
CALCULATE NEW CELL VOLUMES
VOLUME
SUBCYCLE ADVECTIVE FLUX OF MASS, ENERGY, TKE, LENGTH SCALE AND MOMENTUM
CCFLUX,
MOMFLX
BCEPS,
BC
ADD OR DELETE PLANE ABOVE PISTON, IF APPROPRIATE
CHOP
BC, BCEPS, DOME,
PFIND, VOLUME
UPDATE CELL TEMPERATURES, PRESSURES, AND GAMMAS
STATE

Chapter 3: مخطط تسلسل البرنامج (sequence diagram)



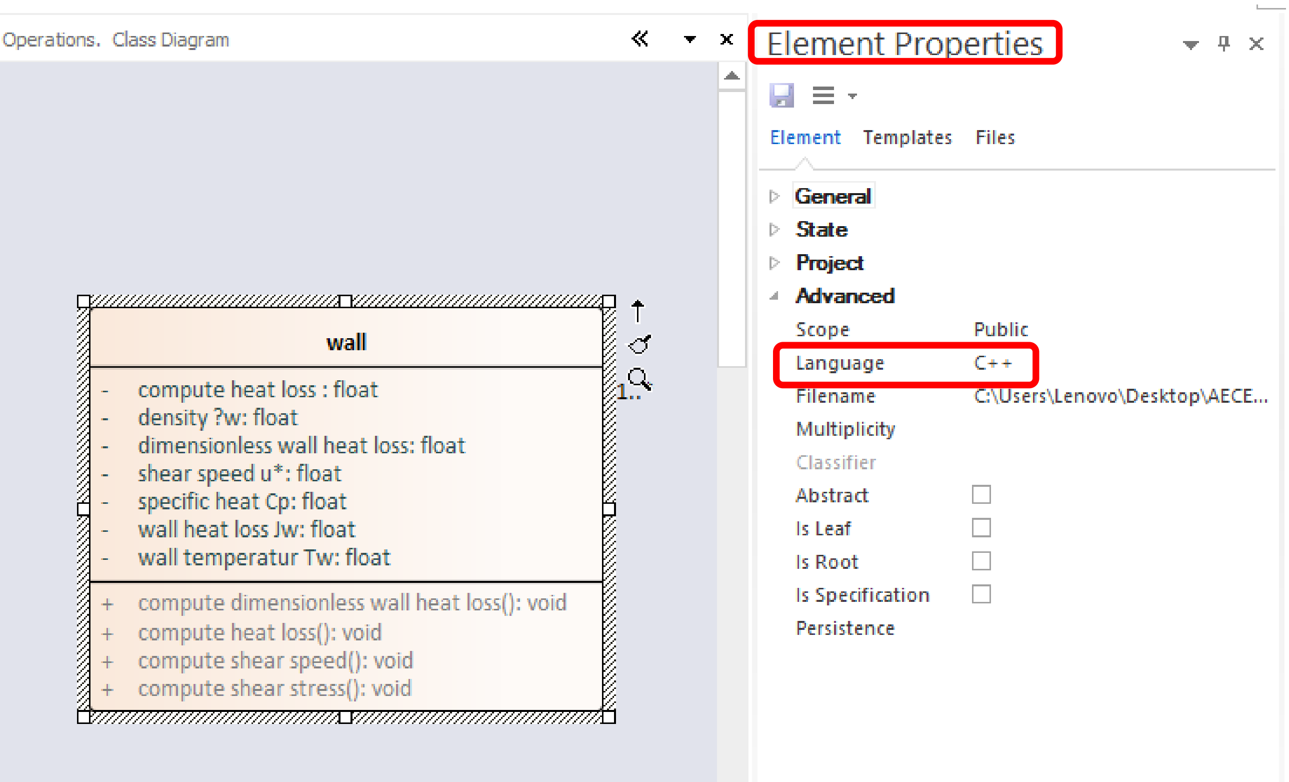
Chapter 4: Class Diagram



Chapter 5: Code generation

To generate your code from your class diagram, follow the steps below:

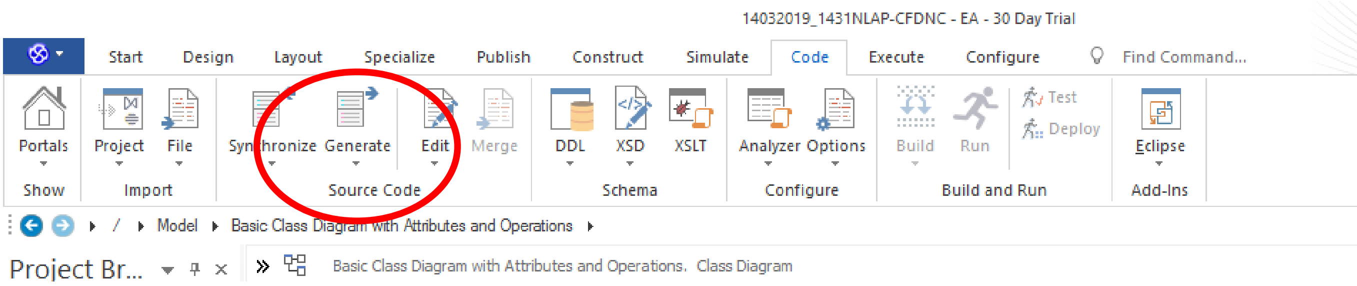
Before starting the code generation make sure to choose the language for your classes, in our case it's C++. First select the class then change the language in: "Element properties → Advanced → Language ". Repeat this on all the classes that you want to generate code from.



Now that all the classes are ready follow these steps.

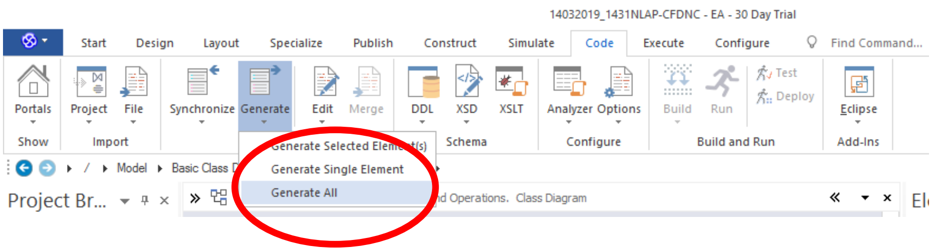
Step 1:

Go to " Code → Source Code → Generate " as shown in the image below



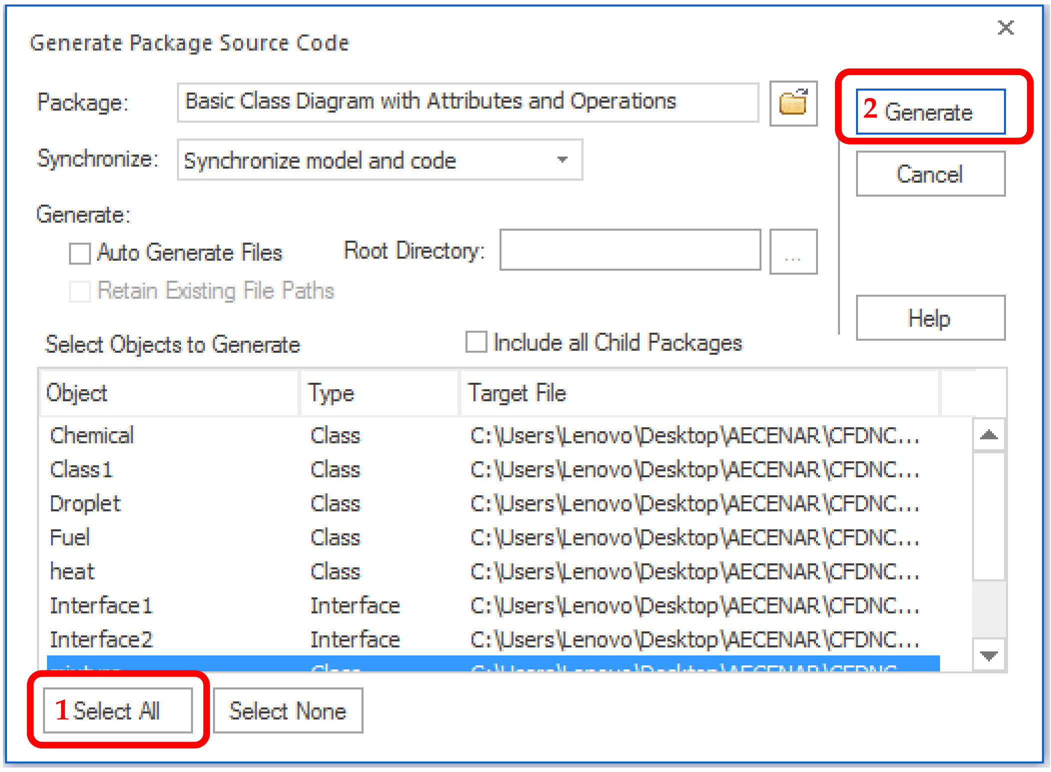
Step 2:

Select the type you want, in our case "Generate all"



Step 3:

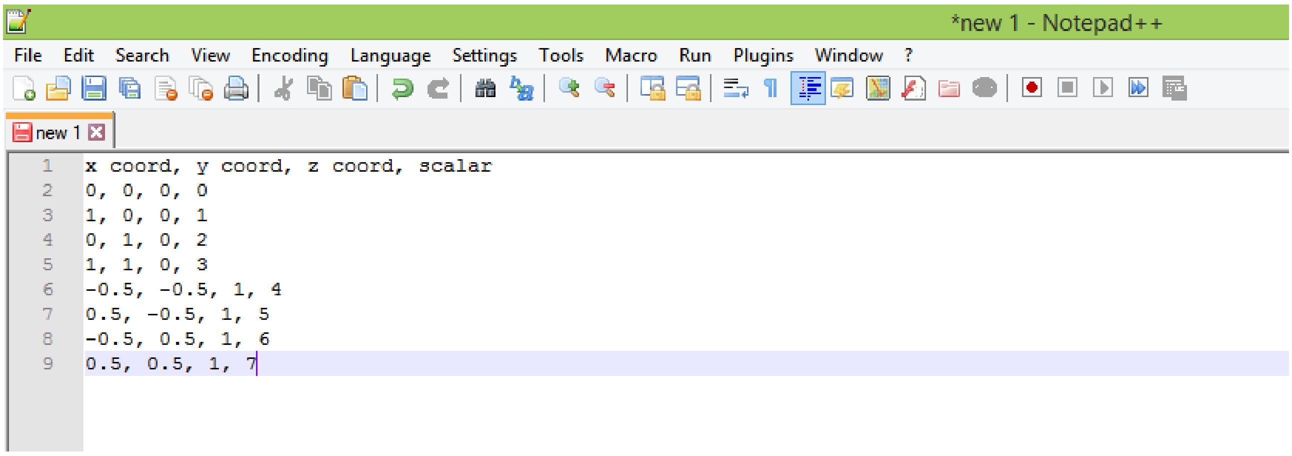
A dialog box will pop up. Choose the Select All button then Generate



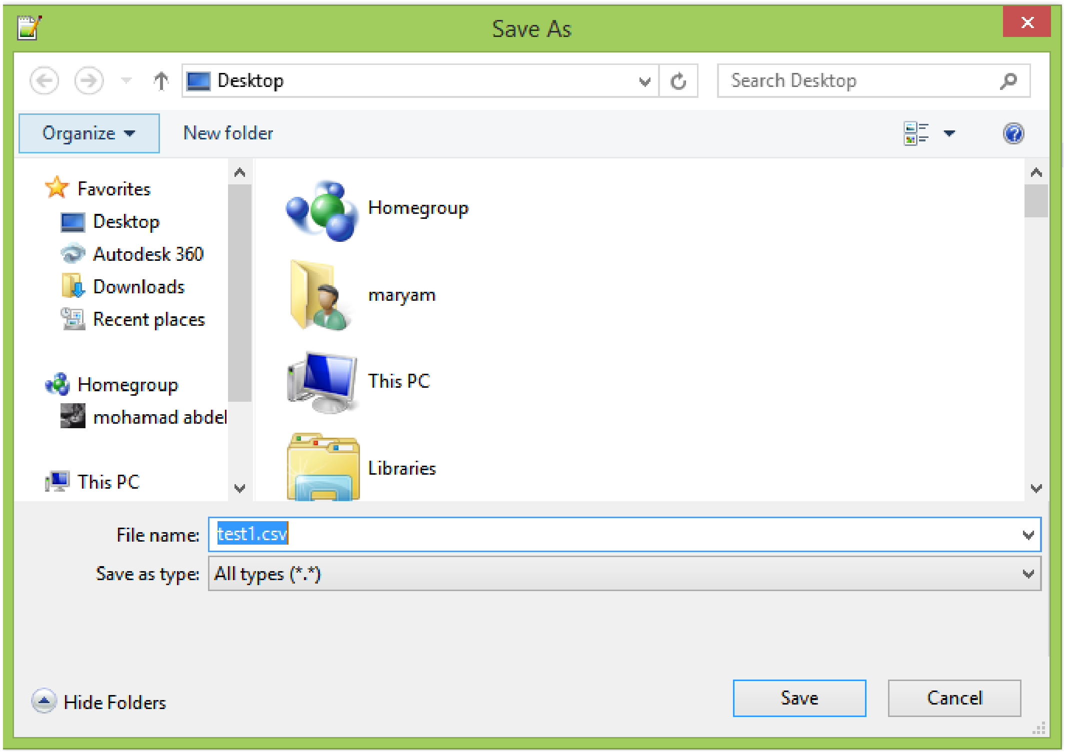
The code will be generated after you choose the designated folder.

Chapter 6: Para view Input files[2]

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.



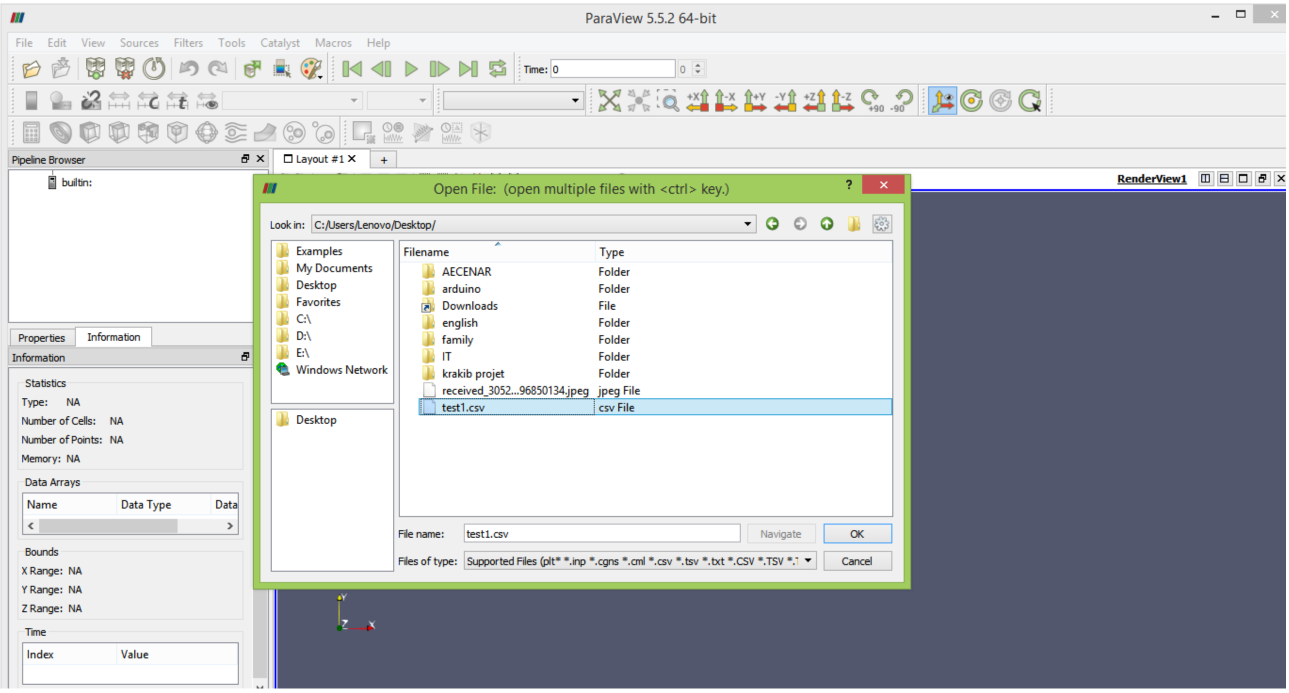
Then save your file as: All types (*. *) as shown in our example test1.csv.



Now it's ready to be opened in Para view.

Select the open button and choose your file .csv.

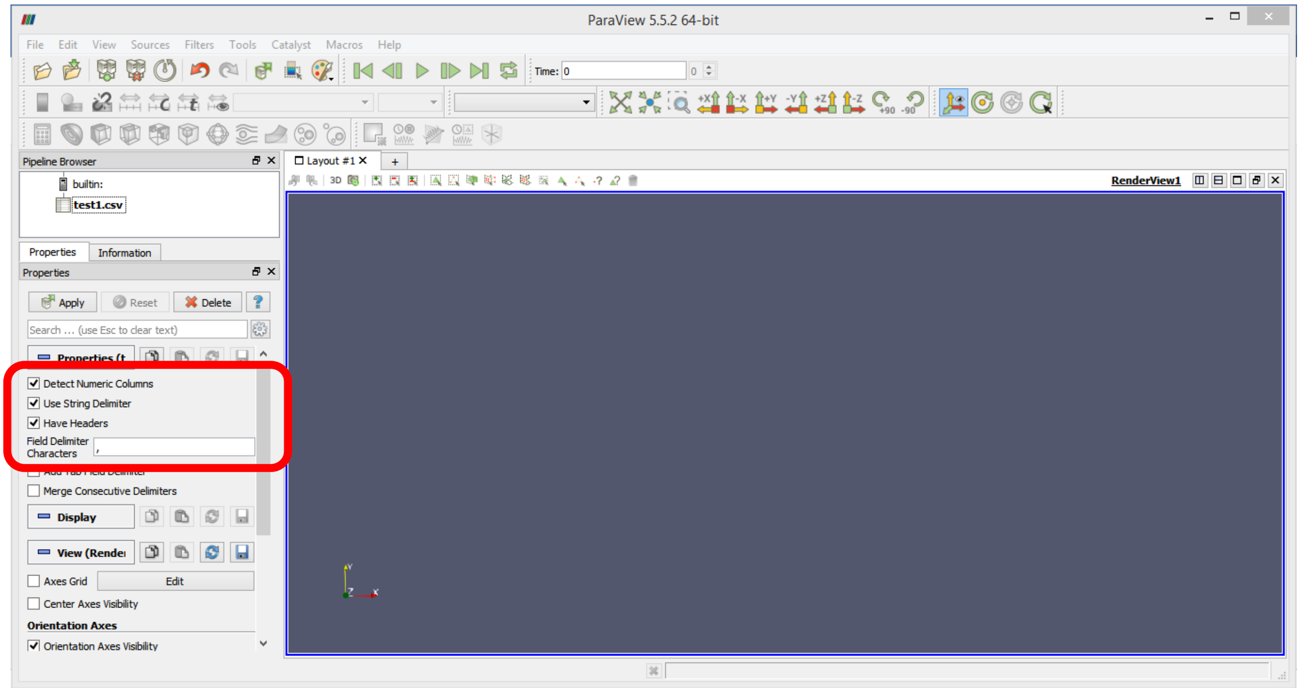
[2] https://www.paraview.org/Wiki/ParaView/Data_formats
https://www.youtube.com/watch?v=mNR2Vn6r0io



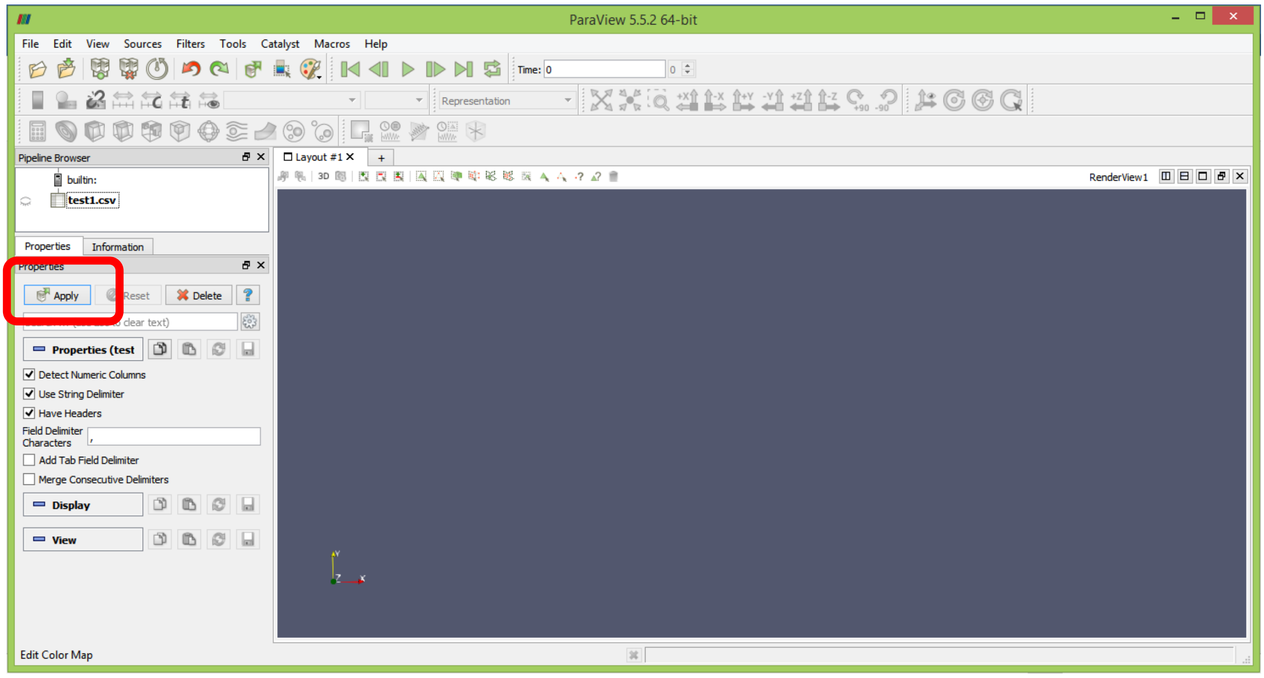
Start Para View, and read in this data. Note that the default settings should be used:

- Detect Numeric Columns ON
- Use String Delimiter ON
- Have Headers ON
- Field Delimiter Characters should be a comma - ','

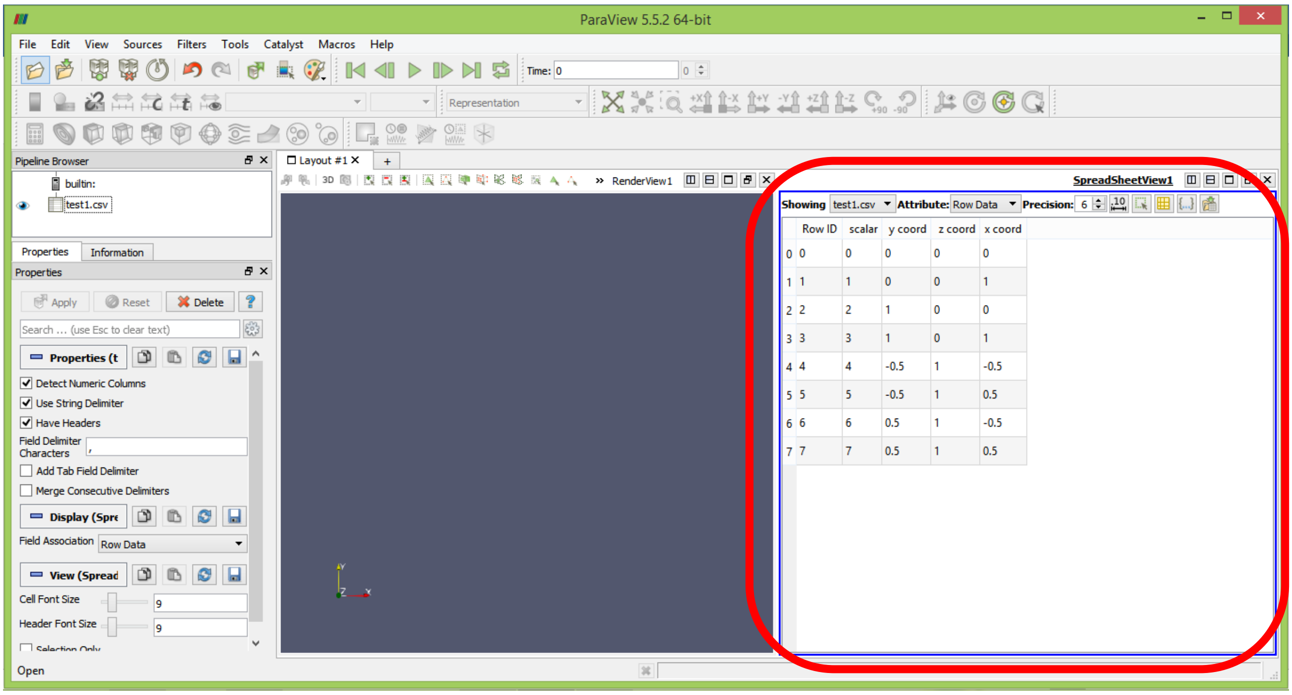
(See figure below)



Then press apply.

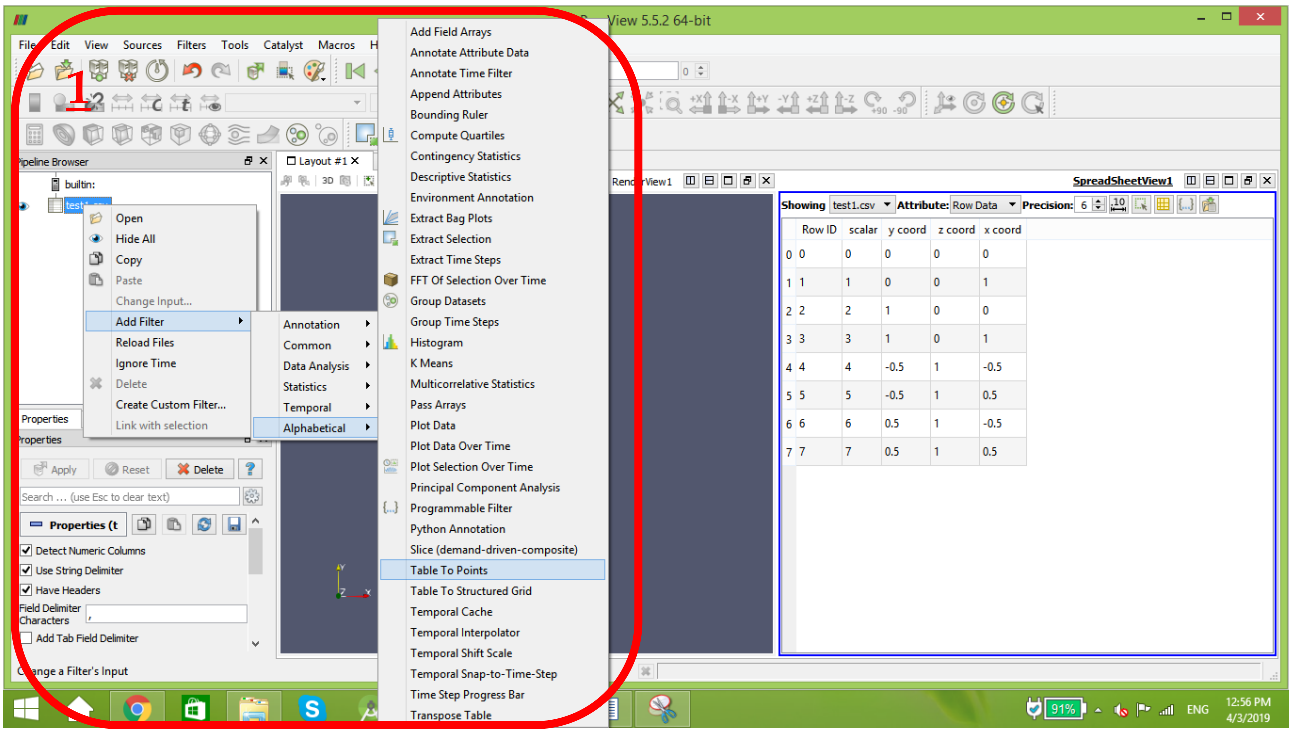


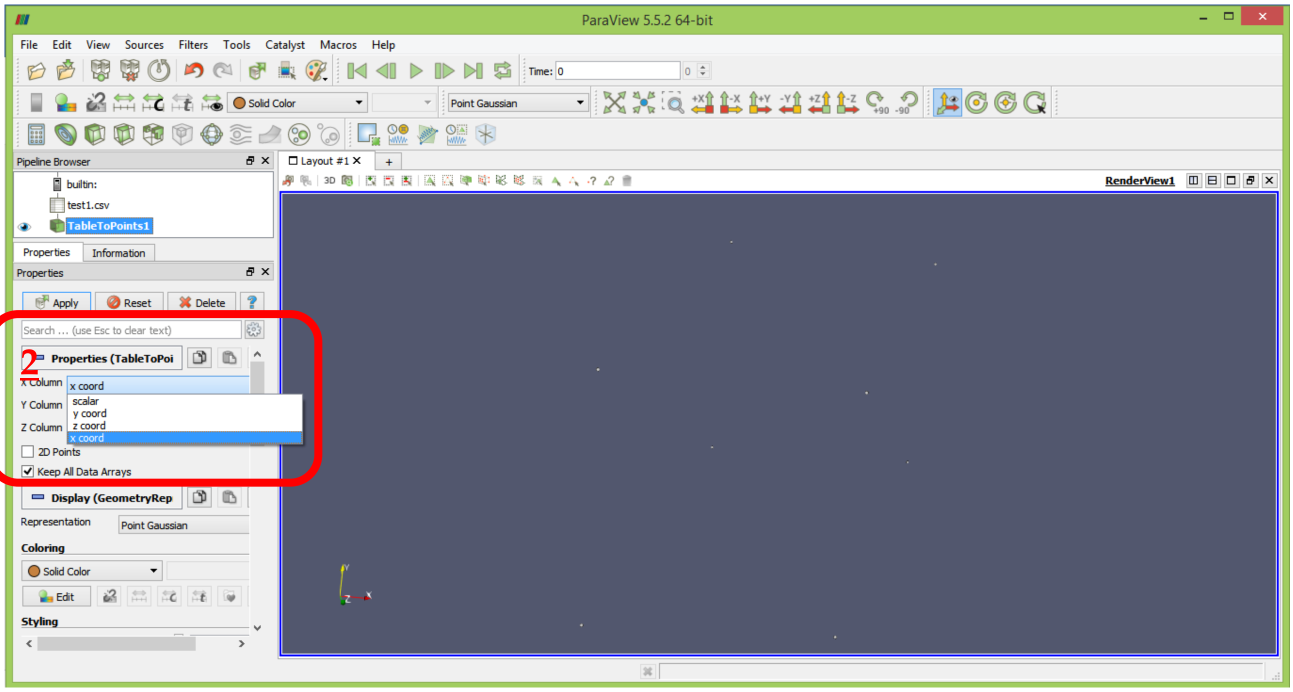
The data should show up as a table.



1. Displaying data as points

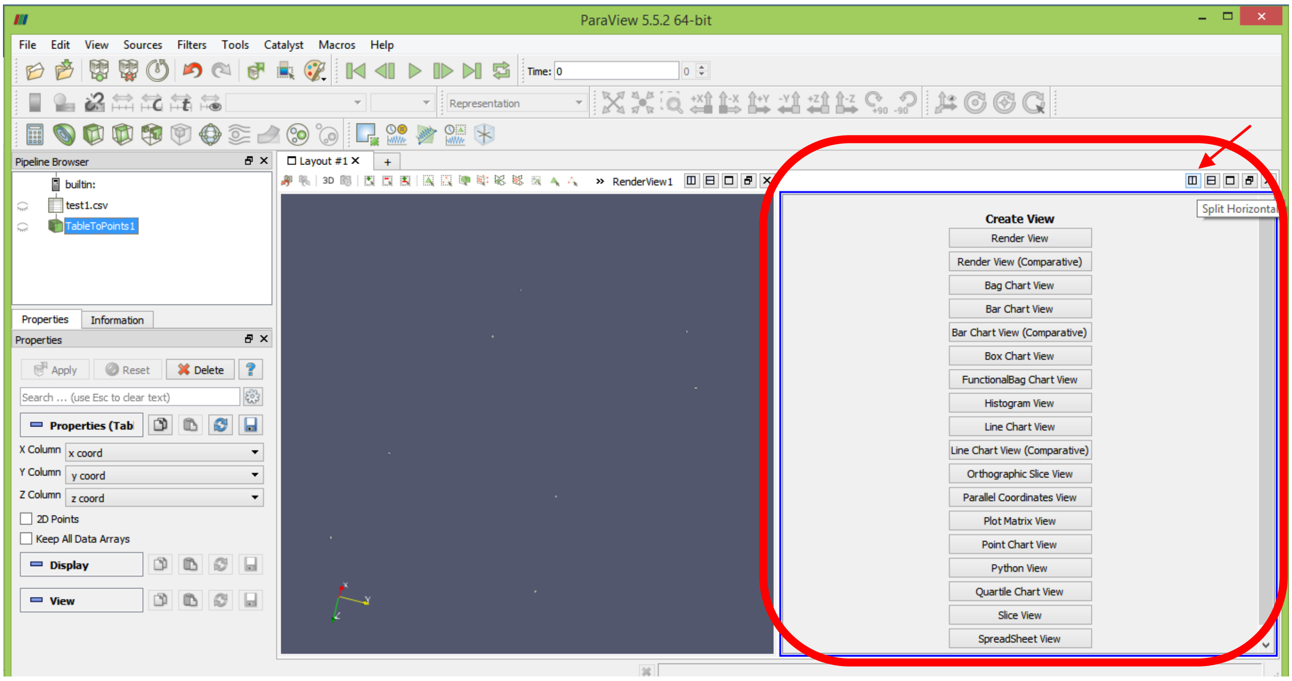
- Run the filter Filters/ Alphabetical/ Table to Points (right click on the table at the left as shown in the figure below).
- Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.





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.

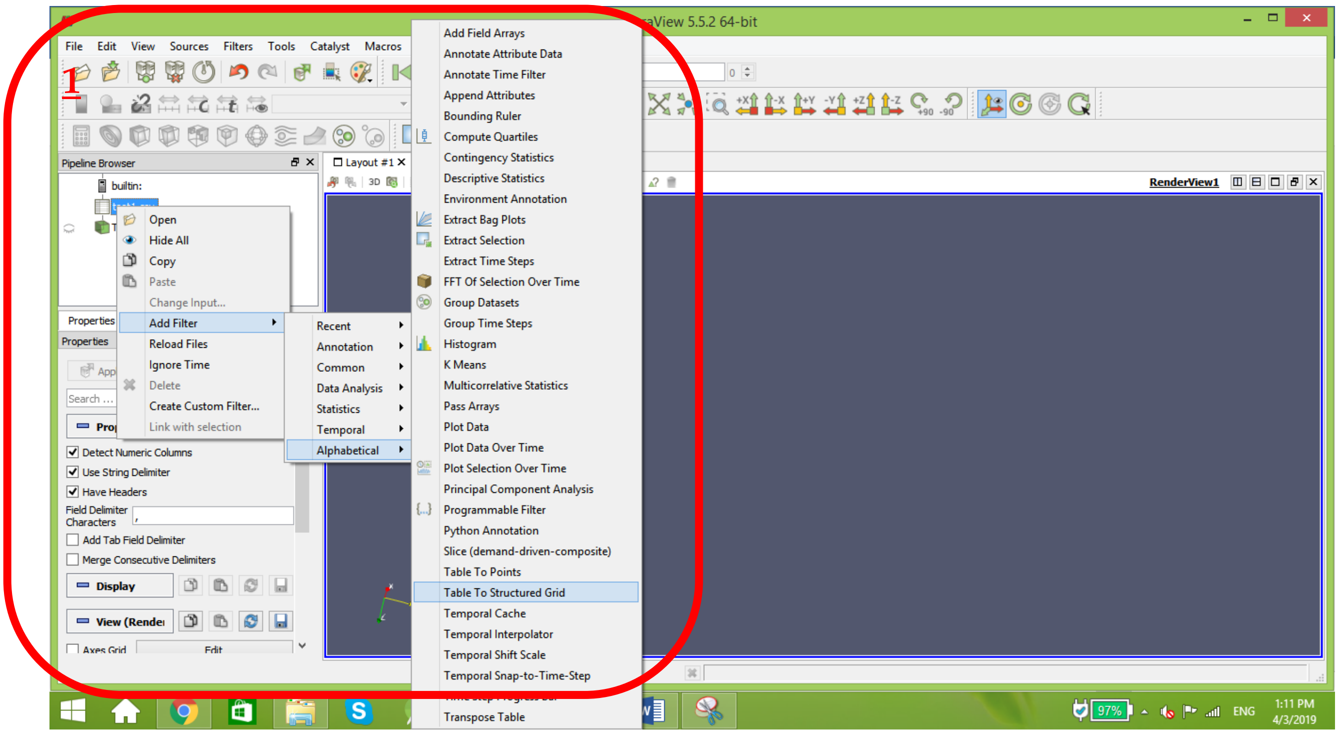


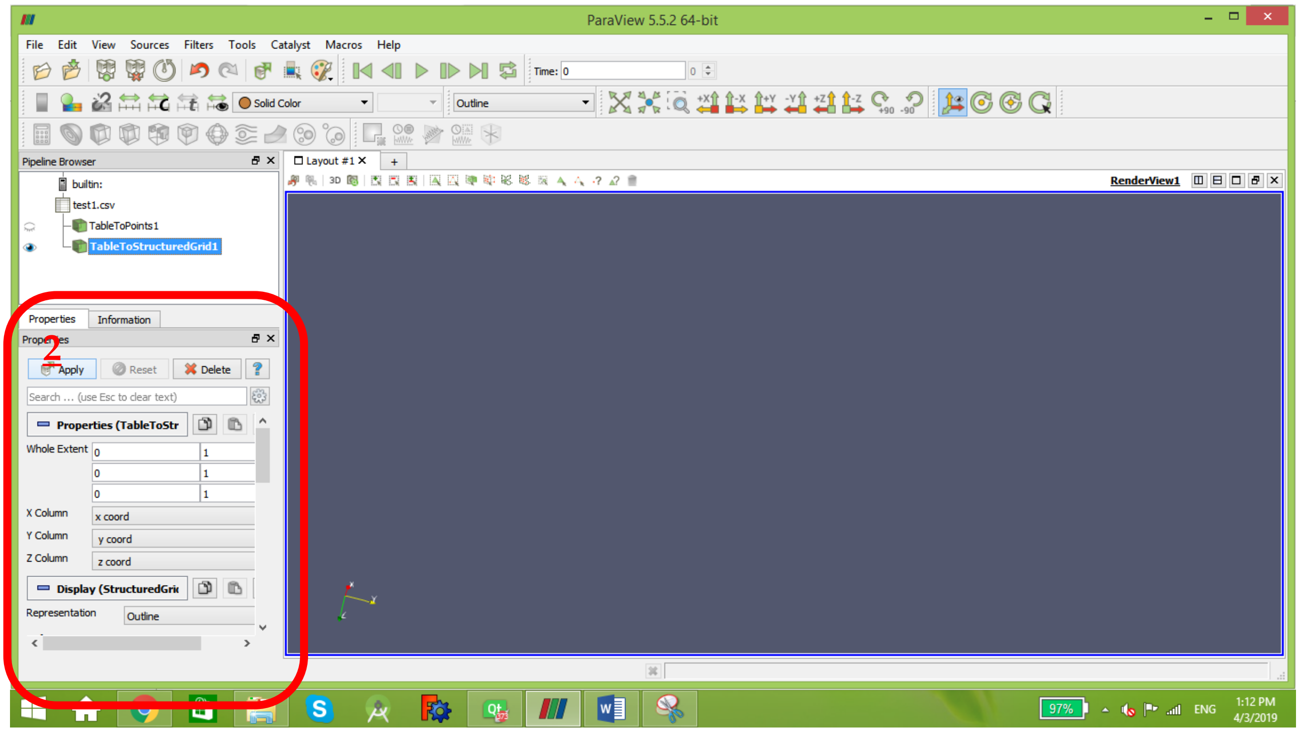
2. Displaying data as structured grid

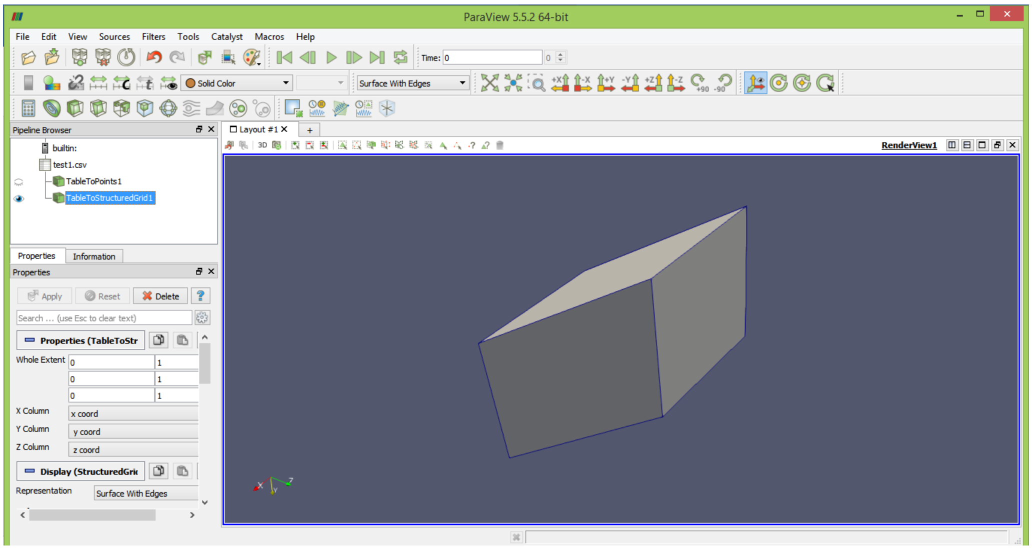
1. Run the filter Filters/ Alphabetical/ Table to Structured Grid.

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 == 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

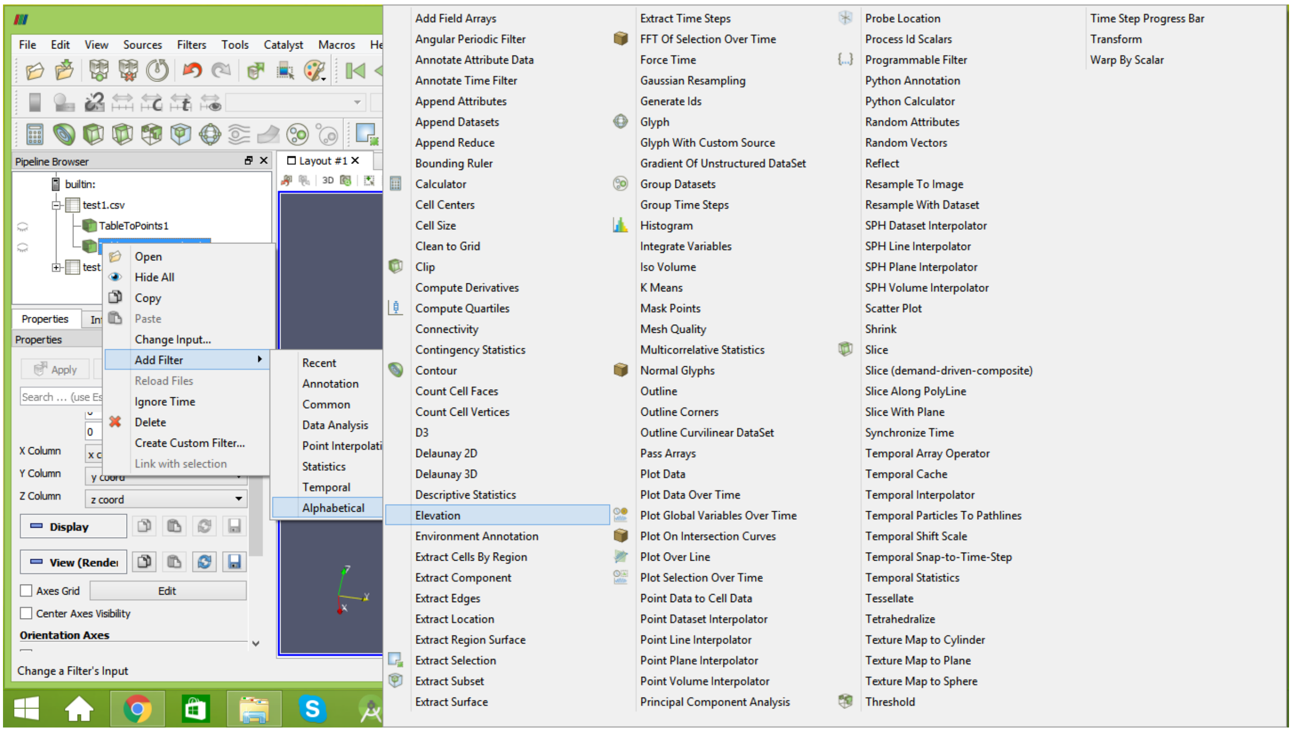
3. Tell Para View what columns are the X, Y and Z coordinate. Be sure to not skip this step. Apply.


1
ParaView 5.5.2 64-bit
File Edit View Sources Filters Tools Catalyst Macros
Pipeline Browser
builtin:
Open
Hide All
Copy
Paste
Change Input...
Add Filter
Reload Files
Ignore Time
Delete
Create Custom Filter...
Link with selection
Recent
Annotation
Common
Data Analysis
Statistics
Temporal
Alphabetical
Add Field Arrays
Annotate Attribute Data
Annotate Time Filter
Append Attributes
Bounding Ruler
Compute Quartiles
Contingency Statistics
Descriptive Statistics
Environment Annotation
Extract Bag Plots
Extract Selection
Extract Time Steps
FFT Of Selection Over Time
Group Datasets
Group Time Steps
Histogram
K Means
Multicorrelative Statistics
Pass Arrays
Plot Data
Plot Data Over Time
Plot Selection Over Time
Principal Component Analysis
Programmable Filter
Python Annotation
Slice (demand-driven-composite)
Table To Points
Table To Structured Grid
Temporal Cache
Temporal Interpolator
Temporal Shift Scale
Temporal Snap-to-Time-Step
Transpose Table
Detect Numeric Columns
Use String Delimiter
Have Headers
Field Delimiter Characters
Add Tab Field Delimiter
Merge Consecutive Delimiters
Display
RenderView1
Layout #1

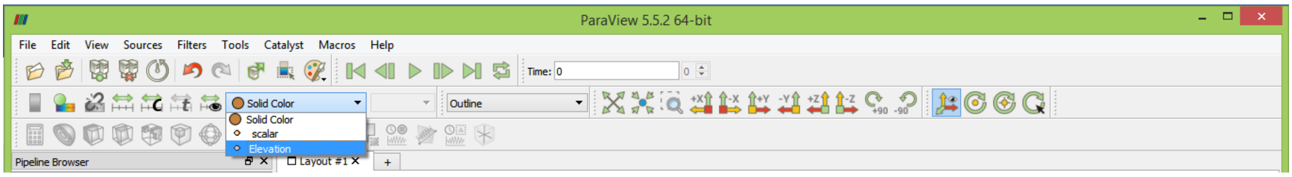

ParaView 5.5.2 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Time: 0
Solid Color
Outline
Pipeline Browser
builtin:
test1.csv
TableToPoints1
TableToStructuredGrid1
Layout #1
RenderView1
Properties Information
2
Apply Reset Delete
Search ... (use Esc to clear text)
Properties (TableToStr
Whole Extent 0 1 0 1 0 1
X Column x coord
Y Column y coord
Z Column z coord
Display (StructuredGrid
Representation Outline



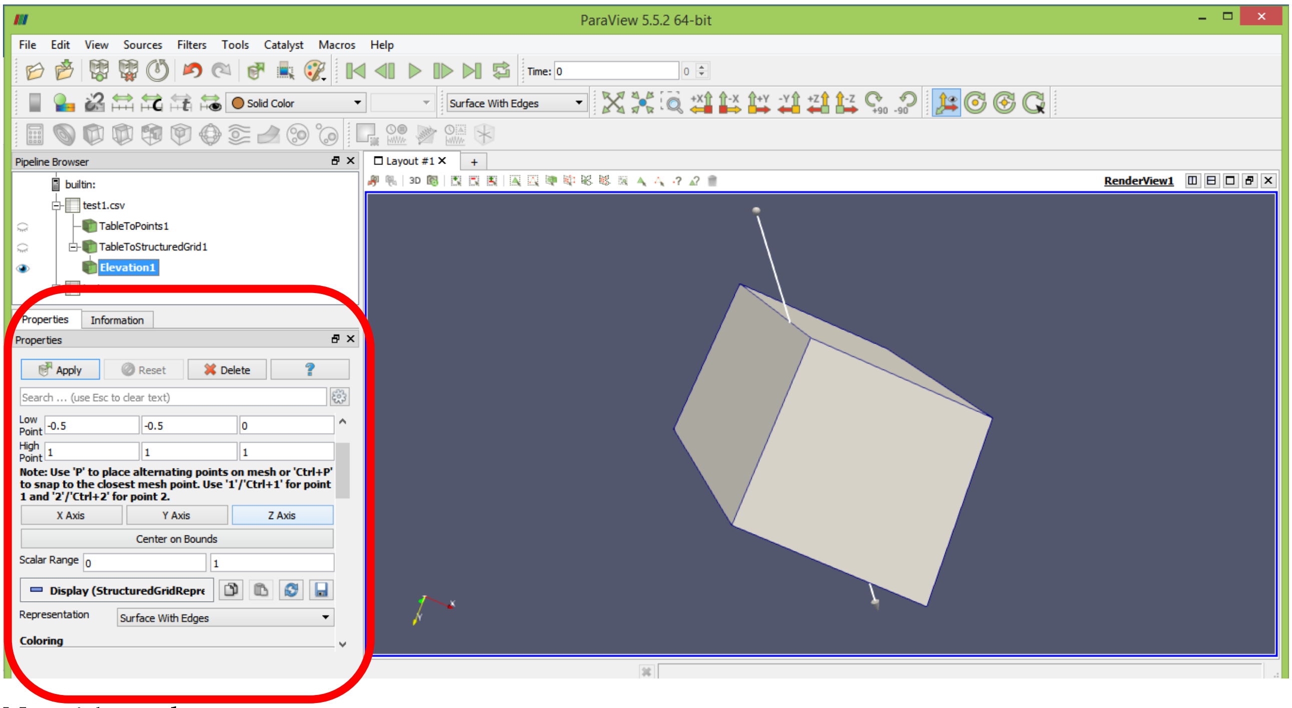
Now to represent yours results with colors, right click on "table to structure"→ add filter → Alphabetic→elevation.



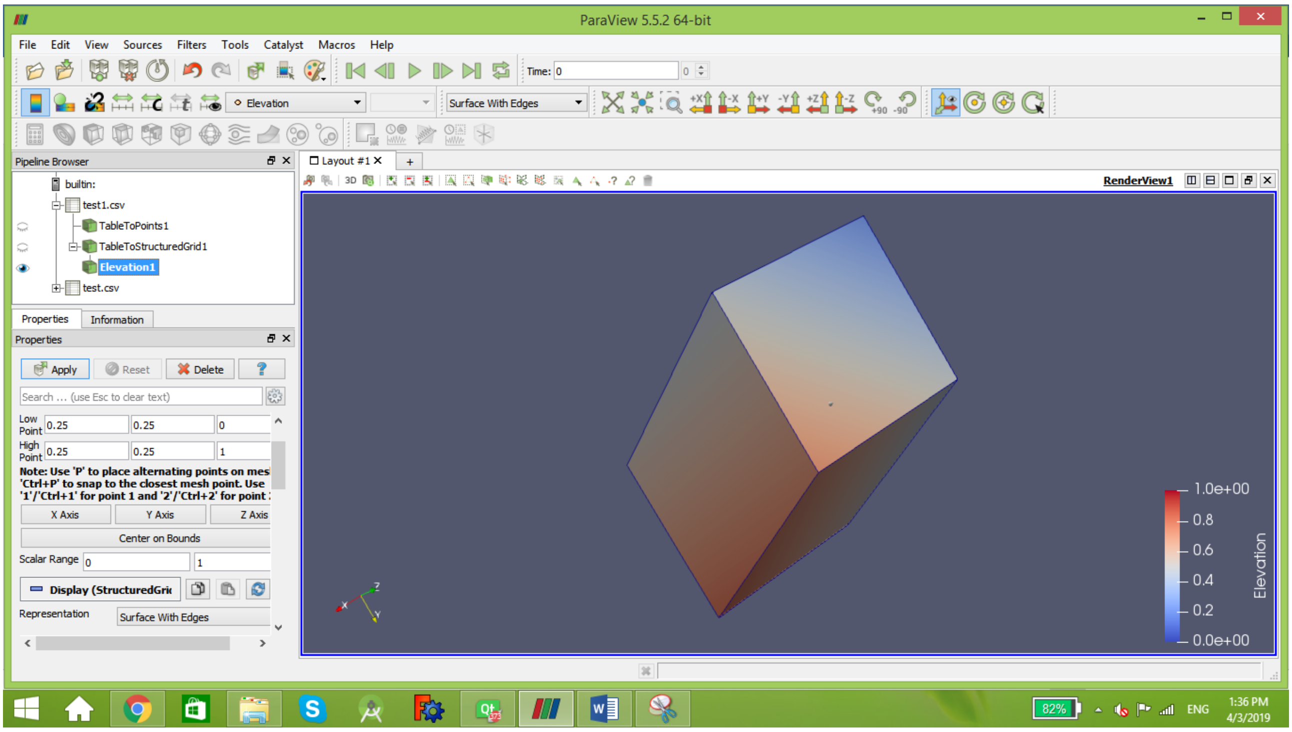
Make sure that you select the elevation button.



Finally choose the desired axis, then apply.



Now it’s ready.



Chapter 7: Discretization of partial differential equations

1. The continuity equation (mass conservation)

$$\frac{\partial\rho}{\partial t}+\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}+\frac{\partial(\rho w)}{\partial z}=0$$

$$\frac{\partial\rho}{\partial t}+\rho\frac{\partial u}{\partial x}+u\frac{\partial\rho}{\partial x}+\rho\frac{\partial v}{\partial y}+v\frac{\partial\rho}{\partial y}+\rho\frac{\partial w}{\partial z}+w\frac{\partial\rho}{\partial z}=0$$

$$\begin{aligned}\frac{\rho_{ijk}^{t+1}-\rho_{ijk}^{t-1}}{2\Delta t}=&-\rho_{ijk}^{t}\left(\frac{u_{i+1jk}^{t}-u_{i-1jk}^{t}}{2\Delta x}\right)-u_{ijk}^{t}\left(\frac{\rho_{i+1jk}^{t}-\rho_{i-1jk}^{t}}{2\Delta x}\right)-\rho_{ijk}^{t}\left(\frac{v_{ij+1k}^{t}-v_{ij-1k}^{t}}{2\Delta y}\right)\\&-v_{ijk}^{t}\left(\frac{\rho_{ij+1k}^{t}-\rho_{ij-1k}^{t}}{2\Delta y}\right)-\rho_{ijk}^{t}\left(\frac{w_{ijk+1}^{t}-w_{ijk-1}^{t}}{2\Delta z}\right)-w_{ijk}^{t}\left(\frac{\rho_{ijk+1}^{t}-\rho_{ijk-1}^{t}}{2\Delta z}\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial^2\rho}{\partial t^2}=&-\frac{\partial\rho}{\partial t}\frac{\partial u}{\partial x}-\rho\frac{\partial^2 u}{\partial x\partial t}-\frac{\partial u}{\partial t}\frac{\partial\rho}{\partial x}-u\frac{\partial^2\rho}{\partial x\partial t}-\frac{\partial\rho}{\partial t}\frac{\partial v}{\partial y}-\rho\frac{\partial^2 v}{\partial t\partial y}-\frac{\partial v}{\partial t}\frac{\partial\rho}{\partial y}-v\frac{\partial^2\rho}{\partial t\partial y}-\frac{\partial\rho}{\partial t}\frac{\partial w}{\partial z}\\&-\rho\frac{\partial^2 w}{\partial t\partial z}-\frac{\partial w}{\partial t}\frac{\partial\rho}{\partial z}-w\frac{\partial^2\rho}{\partial t\partial z}\end{aligned}$$

$$\begin{aligned}\frac{\rho_{ijk}^{t+1}-2\rho_{ijk}^{t}+\rho_{ijk}^{t-1}}{\Delta t^2}\\=&-\frac{\rho_{ijk}^{t+1}-\rho_{ijk}^{t-1}}{2\Delta t}\times\frac{u_{i+1jk}^{t}-u_{i-1jk}^{t}}{2\Delta x}-\rho_{ijk}^{t}\frac{u_{i+1jk}^{t+1}-u_{i-1jk}^{t+1}-u_{i+1jk}^{t-1}+u_{i-1jk}^{t-1}}{4\Delta x\Delta t}\\&-\frac{u_{ijk}^{t+1}-u_{ijk}^{t-1}}{2\Delta t}\times\frac{\rho_{i+1jk}^{t}-\rho_{i-1jk}^{t}}{2\Delta x}-u_{ijk}^{t}\frac{\rho_{i+1jk}^{t+1}-\rho_{i-1jk}^{t+1}-\rho_{i+1jk}^{t-1}+\rho_{i-1jk}^{t-1}}{4\Delta x\Delta t}\\&-\frac{\rho_{ijk}^{t+1}-\rho_{ijk}^{t-1}}{2\Delta t}\times\frac{v_{ij+1k}^{t}-v_{ij-1k}^{t}}{2\Delta y}-\rho_{ijk}^{t}\frac{v_{ij+1k}^{t+1}-v_{ij-1k}^{t+1}-v_{ij+1k}^{t-1}+v_{ij-1k}^{t-1}}{4\Delta y\Delta t}\\&-\frac{v_{ijk}^{t+1}-v_{ijk}^{t-1}}{2\Delta t}\times\frac{\rho_{ij+1k}^{t}-\rho_{ij-1k}^{t}}{2\Delta y}-v_{ijk}^{t}\frac{\rho_{ij+1k}^{t+1}-\rho_{ij-1k}^{t+1}-\rho_{ij+1k}^{t-1}+\rho_{ij-1k}^{t-1}}{4\Delta y\Delta t}\\&-\frac{\rho_{ijk}^{t+1}-\rho_{ijk}^{t-1}}{2\Delta t}\times\frac{w_{ijk+1}^{t}-w_{ijk-1}^{t}}{2\Delta z}-\rho_{ijk}^{t}\frac{w_{ijk+1}^{t+1}-w_{ijk-1}^{t+1}-w_{ijk+1}^{t-1}+w_{ijk-1}^{t-1}}{4\Delta x\Delta t}\\&-\frac{w_{ijk}^{t+1}-w_{ijk}^{t-1}}{2\Delta t}\times\frac{\rho_{ijk+1}^{t}-\rho_{ijk-1}^{t}}{2\Delta z}-w_{ijk}^{t}\frac{\rho_{ijk+1}^{t+1}-\rho_{ijk-1}^{t+1}-\rho_{ijk+1}^{t-1}+\rho_{ijk-1}^{t-1}}{4\Delta z\Delta t}\end{aligned}$$

Species k continuity equation

$$\frac{\partial\rho Y_k}{\partial t}+\frac{\partial\rho u_i Y_k}{\partial x_i}=\frac{\partial}{\partial x_i}\left(\rho D_k\frac{\partial Y_k}{\partial x_i}\right)+\dot{\omega}_k$$

$$Y_k \frac{\partial \rho}{\partial t} + \rho \frac{\partial Y_k}{\partial t} + \frac{\partial \rho u Y_k}{\partial x} + \frac{\partial \rho v Y_k}{\partial y} + \frac{\partial \rho w Y_k}{\partial z}$$
$$= \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2}\right) + D_k \frac{\partial Y_k}{\partial x}\frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2}\right) + D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2}\right)$$
$$+ D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \dot{\omega}_k$$

$$Y_k \frac{\partial \rho}{\partial t} + \rho \frac{\partial Y_k}{\partial t} + \frac{\partial \rho u Y_k}{\partial x} + \frac{\partial \rho v Y_k}{\partial y} + \frac{\partial \rho w Y_k}{\partial z}$$
$$= \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2}\right) + D_k \frac{\partial Y_k}{\partial x}\frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2}\right) + D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2}\right)$$
$$+ D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \dot{\omega}_k$$

$$\rho \frac{\partial Y_k}{\partial t} = -Y_k \frac{\partial \rho}{\partial t} - \rho u \frac{\partial Y_k}{\partial x} - u Y_k \frac{\partial \rho}{\partial x} - \rho Y_k \frac{\partial u}{\partial x} - \rho v \frac{\partial Y_k}{\partial y} - v Y_k \frac{\partial \rho}{\partial y} - \rho Y_k \frac{\partial v}{\partial y} - \rho w \frac{\partial Y_k}{\partial z} - w Y_k \frac{\partial \rho}{\partial z}$$
$$- \rho Y_k \frac{\partial w}{\partial z} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial x^2}\right) + D_k \frac{\partial Y_k}{\partial x}\frac{\partial \rho}{\partial x} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial y^2}\right) + D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \rho D_k \left(\frac{\partial^2 Y_k}{\partial z^2}\right)$$
$$+ D_k \frac{\partial Y_k}{\partial y}\frac{\partial \rho}{\partial y} + \dot{\omega}_k$$

$$\rho \frac{\partial Y_k}{\partial t} = -Y_k \frac{\partial \rho}{\partial t} - \rho u \frac{{Y_k}^t_{i+1jk} - {Y_k}^t_{i-1jk}}{2\Delta x} - u Y_k \frac{\rho^t_{i+1jk} - \rho^t_{i-1jk}}{2\Delta x} - \rho Y_k \frac{u^t_{i+1jk} - u^t_{i-1jk}}{2\Delta x}$$
$$- \rho v \frac{{Y_k}^t_{ij+1k} - {Y_k}^t_{ij-1k}}{2\Delta y} - v Y_k \frac{\rho^t_{ij+1k} - \rho^t_{ij-1k}}{2\Delta y} - \rho Y_k \frac{v^t_{ij+1k} - v^t_{ij-1k}}{2\Delta y}$$
$$- \rho w \frac{{Y_k}^t_{ijk+1} - {Y_k}^t_{ijk-1}}{2\Delta z} - w Y_k \frac{\rho^t_{ijk+1} - \rho^t_{ijk-1}}{2\Delta z} - \rho Y_k \frac{w^t_{ijk+1} - w^t_{ijk-1}}{2\Delta z}$$
$$+ \rho D_k \left(\frac{{Y_k}^t_{i+1jk} - {Y_k}^t_{ijk} + {Y_k}^t_{i-1jk}}{\Delta x^2}\right) + D_k \frac{{Y_k}^t_{i+1jk} - {Y_k}^t_{i-1jk}}{2\Delta x}\frac{\rho^t_{i+1jk} - \rho^t_{i-1jk}}{2\Delta x}$$
$$+ \rho D_k \left(\frac{{Y_k}^t_{ij+1k} - {Y_k}^t_{ijk} + {Y_k}^t_{ij-1k}}{\Delta y^2}\right) + D_k \frac{{Y_k}^t_{ij+1k} - {Y_k}^t_{ij-1k}}{2\Delta y}\frac{\rho^t_{ij+1k} - \rho^t_{ij-1k}}{2\Delta y}$$
$$+ \rho D_k \left(\frac{{Y_k}^t_{ijk+1} - {Y_k}^t_{ijk} + {Y_k}^t_{ijk-1}}{\Delta z^2}\right) + D_k \frac{{Y_k}^t_{ijk+1} - {Y_k}^t_{ijk-1}}{2\Delta z}\frac{\rho^t_{ijk+1} - \rho^t_{ijk-1}}{2\Delta z} + \dot{\omega}_k$$

2. <u>The momentum equation for fluid mixture (momentum conservation)</u>

- **<u>Velocity u:</u>**

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial(\rho u^2)}{\partial x}+\frac{\partial(\rho uv)}{\partial y}+\frac{\partial(\rho uw)}{\partial z}$$
$$=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\lambda\nabla.\vec{V}+2\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right]+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]$$
$$+\rho\sum_{k=1}^{N}Y_k f_x$$

$$\rho\frac{\partial u}{\partial t}+u\frac{\partial \rho}{\partial t}+2\rho u\frac{\partial u}{\partial x}+u^2\frac{\partial \rho}{\partial x}+\rho u\frac{\partial v}{\partial y}+\rho v\frac{\partial u}{\partial y}+uv\frac{\partial \rho}{\partial y}+\rho u\frac{\partial w}{\partial z}+\rho w\frac{\partial u}{\partial z}+uw\frac{\partial \rho}{\partial z}$$
$$=-\frac{\partial p}{\partial x}+\lambda\frac{\partial^2 u}{\partial x^2}+\lambda\frac{\partial^2 v}{\partial x\partial y}+\lambda\frac{\partial^2 w}{\partial x\partial z}+2\mu\frac{\partial^2 u}{\partial^2 x}+\mu\frac{\partial^2 v}{\partial y\partial x}+\mu\frac{\partial^2 u}{\partial y^2}+\mu\frac{\partial^2 u}{\partial z^2}+\mu\frac{\partial^2 w}{\partial z\partial x}$$
$$+\rho\sum_{k=1}^{N}Y_k f_x$$

$$\rho\frac{\partial u}{\partial t}=-u\frac{\partial \rho}{\partial t}-2\rho u\frac{u_{i+1jk}^t-u_{i-1jk}^t}{2\Delta x}-u^2\frac{\rho_{i+1jk}^t-\rho_{i-1jk}^t}{2\Delta x}-\rho u\frac{v_{ij+1k}^t-v_{ij-1k}^t}{2\Delta y}-\rho v\frac{u_{ij+1k}^t-u_{ij-1k}^t}{2\Delta y}$$
$$-uv\frac{\rho_{ij+1k}^t-\rho_{ij-1k}^t}{2\Delta y}-\rho u\frac{w_{ijk+1}^t-w_{ijk-1}^t}{2\Delta z}-\rho w\frac{u_{ijk+1}^t-u_{ijk-1}^t}{2\Delta z}-uw\frac{\rho_{ijk+1}^t-\rho_{ijk-1}^t}{2\Delta z}$$
$$-\frac{p_{i+1jk}^t-p_{i-1jk}^t}{2\Delta x}+\lambda\frac{u_{i+1jk}^t-u_{ijk}^t+u_{i-1jk}^t}{\Delta x^2}$$
$$+\lambda\frac{v_{i+1j+1k}^t-v_{i+1j-1k}^t-v_{i-1j+1k}^t+v_{i-1j-1k}^t}{4\Delta x\Delta y}$$
$$+\lambda\frac{w_{i+1jk+1}^t-w_{i+1jk-1}^t-w_{i-1jk+1}^t+w_{i-1jk-1}^t}{4\Delta x\Delta z}+2\mu\frac{u_{i+1jk}^t-u_{ijk}^t+u_{i-1jk}^t}{\Delta x^2}$$
$$+\mu\frac{v_{i+1j+1k}^t-v_{i+1j-1k}^t-v_{i-1j+1k}^t+v_{i-1j-1k}^t}{4\Delta x\Delta y}+\mu\frac{u_{ij+1k}^t-u_{ijk}^t+u_{ij-1k}^t}{\Delta y^2}$$
$$+\mu\frac{u_{ijk+1}^t-u_{ijk}^t+u_{ijk-1}^t}{\Delta z^2}+\mu\frac{w_{i+1jk+1}^t-w_{i+1jk-1}^t-w_{i-1jk+1}^t+w_{i-1jk-1}^t}{4\Delta x\Delta z}+\rho\sum_{k=1}^{N}Y_k f_x$$

- **Velocity v:**

$$\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho vu)}{\partial x}+\frac{\partial(\rho v^2)}{\partial y}+\frac{\partial(\rho vw)}{\partial z}$$
$$=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right]+\frac{\partial}{\partial y}\left(\lambda\nabla.\vec{V}+2\mu\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right]$$
$$+\rho\sum_{k=1}^{N}Y_k f_{yk}$$

$$\rho\frac{\partial v}{\partial t}+v\frac{\partial\rho}{\partial t}+\rho u\frac{\partial v}{\partial x}+\rho v\frac{\partial u}{\partial x}+uv\frac{\partial\rho}{\partial x}+2\rho v\frac{\partial v}{\partial y}+v^2\frac{\partial\rho}{\partial y}+\rho v\frac{\partial w}{\partial z}+\rho w\frac{\partial v}{\partial z}+vw\frac{\partial\rho}{\partial z}$$
$$=-\frac{\partial p}{\partial y}+\mu\frac{\partial^2 v}{\partial x^2}+\mu\frac{\partial^2 u}{\partial x\partial y}+\lambda\frac{\partial^2 u}{\partial y\partial x}+\lambda\frac{\partial^2 v}{\partial^2 y}+\lambda\frac{\partial^2 w}{\partial y\partial z}+2\mu\frac{\partial^2 v}{\partial^2 y}+\mu\frac{\partial^2 v}{\partial z^2}+\mu\frac{\partial^2 w}{\partial z\partial y}$$
$$+\rho\sum_{k=1}^{N}Y_k f_{yk}$$

$$\rho\frac{\partial v}{\partial t}=-v\frac{\partial\rho}{\partial t}-\rho u\frac{v_{i+1jk}^t-v_{i-1jk}^t}{2\Delta x}-\rho v\frac{u_{i+1jk}^t-u_{i-1jk}^t}{2\Delta x}-uv\frac{\rho_{i+1jk}^t-\rho_{i-1jk}^t}{2\Delta x}-2\rho v\frac{v_{ij+1k}^t-v_{ij-1k}^t}{2\Delta y}$$
$$-v^2\frac{\rho_{ij+1k}^t-\rho_{ij-1k}^t}{2\Delta y}-\rho v\frac{w_{ijk+1}^t-w_{ijk-1}^t}{2\Delta z}-\rho w\frac{v_{ijk+1}^t-v_{ijk-1}^t}{2\Delta z}-vw\frac{\rho_{ijk+1}^t-\rho_{ijk-1}^t}{2\Delta z}$$
$$-\frac{p_{ij+1k}^t-p_{ij-1k}^t}{2\Delta y}+\mu\frac{v_{i+1jk}^t-v_{ijk}^t+v_{i-1jk}^t}{\Delta x^2}$$
$$+\mu\frac{u_{i+1j+1k}^t-u_{i+1j-1k}^t-u_{i-1j+1k}^t+u_{i-1j-1k}^t}{4\Delta x\Delta y}$$
$$+\lambda\frac{u_{i+1j+1k}^t-u_{i+1j-1k}^t-u_{i-1j+1k}^t+u_{i-1j-1k}^t}{4\Delta x\Delta y}+\lambda\frac{v_{ij+1k}^t-v_{ijk}^t+v_{ij-1k}^t}{\Delta y^2}$$
$$+\lambda\frac{w_{ij+1k+1}^t-w_{ij+1k-1}^t-w_{ij-1k+1}^t+w_{ij-1k-1}^t}{4\Delta y\Delta z}+2\mu\frac{v_{ij+1k}^t-v_{ijk}^t+v_{ij-1k}^t}{\Delta y^2}$$
$$+\mu\frac{v_{ijk+1}^t-v_{ijk}^t+v_{ijk-1}^t}{\Delta z^2}+\mu\frac{w_{ij+1k+1}^t-w_{ij+1k-1}^t-w_{ij-1k+1}^t+w_{ij-1k-1}^t}{4\Delta y\Delta z}+\rho\sum_{k=1}^{N}Y_k f_{yk}$$

- **Velocity w:**

$$\frac{\partial(\rho w)}{\partial t}+\frac{\partial(\rho wu)}{\partial x}+\frac{\partial(\rho vw)}{\partial y}+\frac{\partial(\rho w^2)}{\partial z}$$
$$=-\frac{\partial p}{\partial z}+\frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right]+\frac{\partial}{\partial z}\left(\lambda\nabla.\vec{V}+2\mu\frac{\partial w}{\partial z}\right)$$
$$+\rho\sum_{k=1}^{N}Y_k f_{zk}$$

$$\rho\frac{\partial w}{\partial t}+w\frac{\partial\rho}{\partial t}+\rho u\frac{\partial w}{\partial x}+\rho w\frac{\partial u}{\partial x}+uw\frac{\partial\rho}{\partial x}+\rho v\frac{\partial w}{\partial y}+\rho w\frac{\partial v}{\partial y}+vw\frac{\partial\rho}{\partial y}+2\rho w\frac{\partial w}{\partial z}+w^2\frac{\partial\rho}{\partial z}$$
$$=-\frac{\partial p}{\partial z}+\mu\frac{\partial^2 u}{\partial x\partial z}+\mu\frac{\partial^2 w}{\partial x^2}+\mu\frac{\partial^2 v}{\partial y\partial z}+\mu\frac{\partial^2 w}{\partial y^2}+\lambda\frac{\partial^2 u}{\partial z\partial x}+\lambda\frac{\partial^2 v}{\partial z\partial y}+\lambda\frac{\partial^2 w}{\partial^2 z}+2\mu\frac{\partial^2 w}{\partial^2 z}$$
$$+\rho\sum_{k=1}^{N}Y_k f_{zk}$$

$$\rho\frac{w}{\partial t}=-w\frac{\partial\rho}{\partial t}-\rho u\frac{w_{i+1jk}^{t}-w_{i-1jk}^{t}}{2\Delta x}-\rho w\frac{u_{i+1jk}^{t}-u_{i-1jk}^{t}}{2\Delta x}-uw\frac{\rho_{i+1jk}^{t}-\rho_{i-1jk}^{t}}{2\Delta x}-\rho v\frac{w_{ij+1k}^{t}-w_{ij-1k}^{t}}{2\Delta y}$$
$$-\rho w\frac{v_{ij+1k}^{t}-v_{ij-1k}^{t}}{2\Delta y}-wv\frac{\rho_{ij+1k}^{t}-\rho_{ij-1k}^{t}}{2\Delta y}-2\rho w\frac{w_{ijk+1}^{t}-w_{ijk-1}^{t}}{2\Delta z}-w^{2}\frac{\rho_{ijk+1}^{t}-\rho_{ijk-1}^{t}}{2\Delta z}$$
$$-\frac{p_{ijk+1}^{t}-p_{ijk-1}^{t}}{2\Delta z}+\mu\frac{u_{i+1jk+1}^{t}-u_{i+1jk-1}^{t}-u_{i-1jk+1}^{t}+u_{i-1jk-1}^{t}}{4\Delta x\Delta z}$$
$$+\mu\frac{w_{i+1jk}^{t}-w_{ijk}^{t}+w_{i-1jk}^{t}}{\Delta x^{2}}+\mu\frac{v_{ij+1k+1}^{t}-v_{ij+1k-1}^{t}-v_{ij-1k+1}^{t}+v_{ij-1k-1}^{t}}{4\Delta y\Delta z}$$
$$+\mu\frac{w_{ij+1k}^{t}-w_{ijk}^{t}+w_{ij-1k}^{t}}{\Delta y^{2}}+\lambda\frac{u_{i+1jk+1}^{t}-u_{i+1jk-1}^{t}-u_{i-1jk+1}^{t}+u_{i-1jk-1}^{t}}{4\Delta x\Delta z}$$
$$+\lambda\frac{v_{ij+1k+1}^{t}-v_{ij+1k-1}^{t}-v_{ij-1k+1}^{t}+v_{ij-1k-1}^{t}}{4\Delta y\Delta z}+\lambda\frac{w_{ijk+1}^{t}-w_{ijk}^{t}+w_{ijk-1}^{t}}{\Delta z^{2}}$$
$$+2\mu\frac{w_{ijk+1}^{t}-w_{ijk}^{t}+w_{ijk-1}^{t}}{\Delta z^{2}}+\rho\sum_{k=1}^{N}Y_{k}f_{zk}$$

3. The internal energy equation (energy conservation)

$$\frac{\partial(\rho e)}{\partial t}+\nabla.\left(\rho e\vec{V}\right)=\dot{w}_{T}-\frac{\partial q_{i}}{\partial x_{i}}+\frac{\partial}{\partial x_{i}}\left(\sigma_{ij}u_{i}\right)+\dot{Q}+\rho\sum_{k=1}^{N}Y_{k}f_{ki}(u_{i}+V_{ki})$$

$$\frac{\partial(\rho e)}{\partial t}+\frac{\partial\rho eu}{\partial x}+\frac{\partial\rho ev}{\partial y}+\frac{\partial\rho ew}{\partial z}$$
$$=-\sum_{k=1}^{N}\Delta h_{fk}^{0}\,\dot{w}_{k}-\frac{\partial q}{\partial x}-\frac{\partial q}{\partial y}-\frac{\partial q}{\partial z}+\frac{\partial\left((\tau_{ix}-p)u\right)}{\partial x}+\frac{\partial\left((\tau_{iy}-p)v\right)}{\partial y}+\frac{\partial\left((\tau_{iz}-p)w\right)}{\partial z}$$
$$+\dot{Q}+\rho\sum_{k=1}^{N}Y_{k}f_{kx}(u+V_{ki})$$

$$\frac{\partial(\rho e)}{\partial t}+\frac{\partial\rho eu}{\partial x}+\frac{\partial\rho ev}{\partial y}+\frac{\partial\rho ew}{\partial z}$$
$$=-\sum_{k=1}^{N}\Delta h_{fk}^{0}\,\dot{w}_{k}-\frac{\partial q}{\partial x}-\frac{\partial q}{\partial y}-\frac{\partial q}{\partial z}+\tau_{xx}\frac{\partial u}{\partial x}+\tau_{yx}\frac{\partial u}{\partial y}+\tau_{zx}\frac{\partial u}{\partial z}-p\frac{\partial u}{\partial x}+\tau_{xy}\frac{\partial v}{\partial x}+\tau_{yy}\frac{\partial v}{\partial y}$$
$$+\tau_{zy}\frac{\partial v}{\partial z}-p\frac{\partial v}{\partial y}+\tau_{xz}\frac{\partial w}{\partial x}+\tau_{yz}\frac{\partial w}{\partial y}+\tau_{zz}\frac{\partial w}{\partial z}-p\frac{\partial w}{\partial z}+\dot{Q}+\rho\sum_{k=1}^{N}Y_{k}f_{ki}(u+V_{ki})$$

$$
\begin{aligned}
&\frac{\partial(\rho e)}{\partial t}+\frac{\partial \rho e u}{\partial x}+\frac{\partial \rho e v}{\partial y}+\frac{\partial \rho e w}{\partial z} \\
&\qquad=-\sum_{k=1}^{N}\Delta h_{fk}^{0}\,\dot{w}_k-\frac{\partial\left(-\lambda\frac{\partial T}{\partial x}+\sum_{k=1}^{N}h_kY_kV_{k,x}\right)}{\partial x}-\frac{\partial\left(-\lambda\frac{\partial T}{\partial y}+\sum_{k=1}^{N}h_kY_kV_{k,y}\right)}{\partial y} \\
&\qquad-\frac{\partial\left(-\lambda\frac{\partial T}{\partial z}+\sum_{k=1}^{N}h_kY_kV_{k,z}\right)}{\partial z}+\left(-\frac{2}{3}\mu\frac{\partial u_k}{\partial x}+\mu\left(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial x}\right)\right)\frac{\partial u}{\partial x}+\left(\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)\frac{\partial u}{\partial y} \\
&\qquad+\left(\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right)\frac{\partial u}{\partial z}+\left(\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)\frac{\partial v}{\partial x}+\left(-\frac{2}{3}\mu\frac{\partial v_k}{\partial y}+\mu\left(2\frac{\partial v}{\partial y}\right)\right)\frac{\partial v}{\partial y} \\
&\qquad+\left(\mu\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right)\frac{\partial v}{\partial z}+\left(\mu\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right)\frac{\partial w}{\partial x}+\left(\mu\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)\right)\frac{\partial w}{\partial y} \\
&\qquad+\left(-\frac{2}{3}\mu\frac{\partial w_k}{\partial z}+\mu\left(2\frac{\partial w}{\partial z}\right)\right)\frac{\partial w}{\partial z}-p\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+\dot{Q}+\rho\sum_{k=1}^{N}Y_kf_{ki}(u+V_{ki})
\end{aligned}
$$

$$
\begin{aligned}
&\rho\frac{\partial e}{\partial t}+e\frac{\partial \rho}{\partial t}+\rho e\frac{\partial u}{\partial x}+eu\frac{\partial \rho}{\partial x}+\rho u\frac{\partial e}{\partial x}+\rho e\frac{\partial v}{\partial y}+ev\frac{\partial \rho}{\partial y}+\rho v\frac{\partial e}{\partial y}+\rho e\frac{\partial w}{\partial z}+ew\frac{\partial \rho}{\partial z}+\rho w\frac{\partial e}{\partial z}= \\
&\qquad=-\sum_{k=1}^{N}\Delta h_{fk}^{0}\,\dot{w}_k-\lambda\frac{\partial^2 T}{\partial x^2}-\frac{\partial}{\partial x}\sum_{k=1}^{N}h_kY_kV_{k,x}-\lambda\frac{\partial^2 T}{\partial y^2}+\frac{\partial}{\partial y}\sum_{k=1}^{N}h_kY_kV_{k,y}-\lambda\frac{\partial^2 T}{\partial z^2} \\
&\qquad+\frac{\partial}{\partial z}\sum_{k=1}^{N}h_kY_kV_{k,z}-\frac{2}{3}\mu\frac{\partial^2 u}{\partial x^2}+2\mu\frac{\partial^2 u}{\partial x^2}+\mu\frac{\partial^2 u}{\partial y^2}+\mu\frac{\partial v}{\partial x}\frac{\partial u}{\partial y}+\mu\frac{\partial^2 u}{\partial z^2}+\mu\frac{\partial w}{\partial x}\frac{\partial u}{\partial z}+\mu\frac{\partial^2 v}{\partial x^2} \\
&\qquad+\mu\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}-\frac{2}{3}\mu\frac{\partial^2 v}{\partial y^2}+2\mu\frac{\partial^2 v}{\partial y^2}+\mu\frac{\partial^2 v}{\partial z^2}+\mu\frac{\partial w}{\partial y}\frac{\partial v}{\partial z}+\mu\frac{\partial^2 w}{\partial x^2}+\mu\frac{\partial u}{\partial z}\frac{\partial w}{\partial x}+\mu\frac{\partial^2 w}{\partial y^2} \\
&\qquad+\mu\frac{\partial v}{\partial z}\frac{\partial w}{\partial y}-\frac{2}{3}\mu\frac{\partial^2 w}{\partial z^2}+\mu 2\frac{\partial^2 w}{\partial z^2}-p\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)+\dot{Q}+\rho\sum_{k=1}^{N}Y_kf_{kx}u \\
&\qquad+\rho\sum_{k=1}^{N}Y_kf_{kx}V_{kx}+\rho\sum_{k=1}^{N}Y_kf_{ky}v+\rho\sum_{k=1}^{N}Y_kf_{ky}V_{ky}+\rho\sum_{k=1}^{N}Y_kf_{kz}w+\rho\sum_{k=1}^{N}Y_kf_{kz}V_{kz}
\end{aligned}
$$

$$
V_{k,i}Y_k=-D_k\frac{\partial Y_k}{\partial x_i}
$$

$$
\begin{aligned}
&\rho\frac{\partial e}{\partial t}+e\frac{\partial \rho}{\partial t}+\rho e\frac{\partial u}{\partial x}+eu\frac{\partial \rho}{\partial x}+\rho u\frac{\partial e}{\partial x}+\rho e\frac{\partial v}{\partial y}+ev\frac{\partial \rho}{\partial y}+\rho v\frac{\partial e}{\partial y}+\rho e\frac{\partial w}{\partial z}+ew\frac{\partial \rho}{\partial z}+\rho w\frac{\partial e}{\partial z}\\
&\qquad=-\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}\\
&\qquad-D_k\sum_{k=1}^{N}h_k\frac{\partial^2 Y_k}{\partial z^2}-\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}\\
&\qquad+\mu\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}-\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}\\
&\qquad+\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}-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}\\
&\qquad+\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}+\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}-2{Y_k}_{ijk}^{t}+{Y_k}_{i-1jk}^{t}}{\Delta x^2}-D_k\sum_{k=1}^{N}h_k\frac{{Y_k}_{ij+1k}^{t}-2{Y_k}_{ijk}^{t}+{Y_k}_{ij-1k}^{t}}{\Delta y^2} \\
& -D_k\sum_{k=1}^{N}h_k\frac{{Y_k}_{ijk+1}^{t}-2{Y_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}
$$

4. Chemical relations

Mass reaction rate:

$$
\boxed{\dot{\omega}_k=\sum_{j=1}^{M}\dot{\omega}_{kj}=W_k\sum_{j=1}^{M}\nu_{kj}\mathcal{Q}_j}
$$

Rate of progress of reaction j:

$$
\boxed{\mathcal{Q}_j=K_{fj}\Pi_{k=1}^{N}\left(\frac{\rho Y_k}{W_k}\right)^{\nu'_{kj}}-K_{rj}\Pi_{k=1}^{N}\left(\frac{\rho Y_k}{W_k}\right)^{\nu''_{kj}}}
$$

Rate constants:

$$\boxed{K_{fj} = A_{fj} T^{\beta_j} \exp\left(-\frac{E_j}{RT}\right) = A_{fj} T^{\beta_j} \exp\left(-\frac{T_{aj}}{T}\right)}$$

$$\boxed{K_{rj} = \frac{K_{fj}}{\left(\frac{p_a}{RT}\right)^{\sum_{k=1}^{N} \nu_{kj}} \exp\left(\frac{\Delta S_j^0}{R} - \frac{\Delta H_j^0}{RT}\right)}}$$

Mixture density:[3]

$\rho_m = (\rho_1 v_1 + \rho_2 v_2 + .. + \rho_n v_n) / (v_1 + v_2 + .. + v_n)$

Mixture viscosity: [4]

$$\mu_{ga} = \frac{\sum_{i=1}^{N} y_i \mu_i \sqrt{M_{gi}}}{\sum_{i=1}^{N} y_i \sqrt{M_{gi}}},$$

-

Species viscosity:

$$\mu_g = K_1 \exp\left(X\rho^Y\right), \ldots$$

$$K_1 = \frac{\left(0.00094 + 2\times 10^{-6} M_g\right) T^{1.5}}{\left(209 + 19 M_g + T\right)}$$

$$X = 3.5 + \frac{986}{T} + 0.01 M_g$$

- $Y = 2.4 - 0.2X$
- μ_g = gas viscosity, cp
- ρ =gas density, g/cm^3
- p = pressure, psia
- T = temperature °R
- M_g = gas molecular weight = 28.967 γ_g

Pressure:

$$\boxed{p = \rho \frac{R}{W} T} \quad \boxed{\frac{1}{W} = \sum_{k=1}^{N} \frac{Y_k}{W_k}}$$

[3] https://www.engineeringtoolbox.com/gas-mixture-properties-d_586.html

[4] https://petrowiki.org/Gas_viscosity

Chapter 8: Application

1. Hydrogen combustion characteristics

The global reaction for hydrogen combustion is as follow:

$$2H_2 + O_2 \rightarrow 2H_2O$$

3 species are presented: H_2, O_2 and H_2O. Their characteristics are presented below.

1.1. Hydrogen characteristics

Values are considered at T=325 K

Density: ρ=0.07603 kg/m^3

Molecular weight: M=2.016 g/mol

Mass: m=1 kg

Volume: V=1 m^3

Reaction coefficients: a=2, b=0

1.2. Oxygen characteristics

Values are considered at T=325 K

Density: ρ=1.2068 kg/m^3

Molecular weight: M=32 g/mol

Mass: m=1 kg

Volume: V=1 m^3

Reaction coefficients: a=1, b=0

1.3. Water characteristics

Values are considered at T=325 K

Density: ρ=0.6794 kg/m^3

Molecular weight: M=18 g/mol

Mass: m=1 kg

Volume: V=1 m^3

Reaction coefficients: a=0, b=2

1.4. Mixture characteristics

Entropy differences: ΔS^0=-89 j/mol. k

Enthalpy differences: ΔH^0=-484000 j/mol

Activation energy: E=199911.52 j/mol

Chemical constant: A=1.7 E+13

Temperature constant: β=0

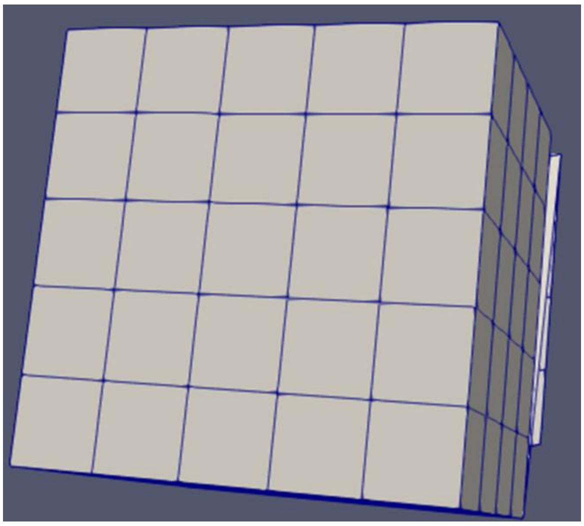
2. Program code

(the program used here isn't completely correct, an error analysis is needed)

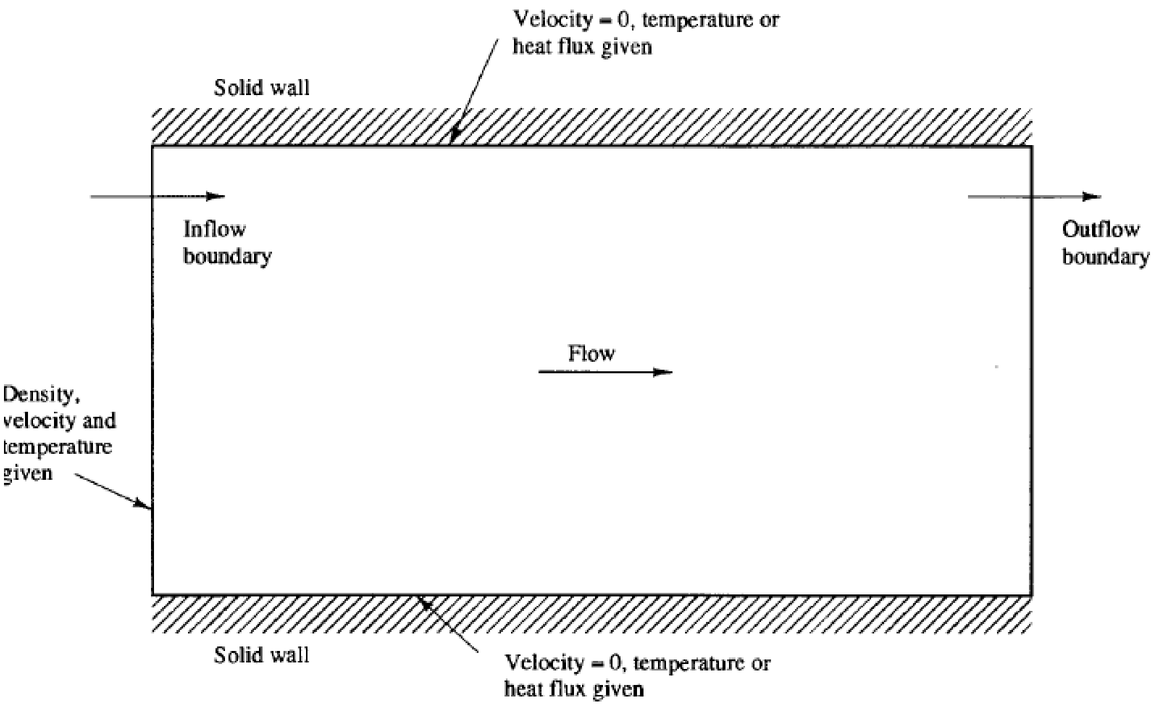
See Annex A.

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.

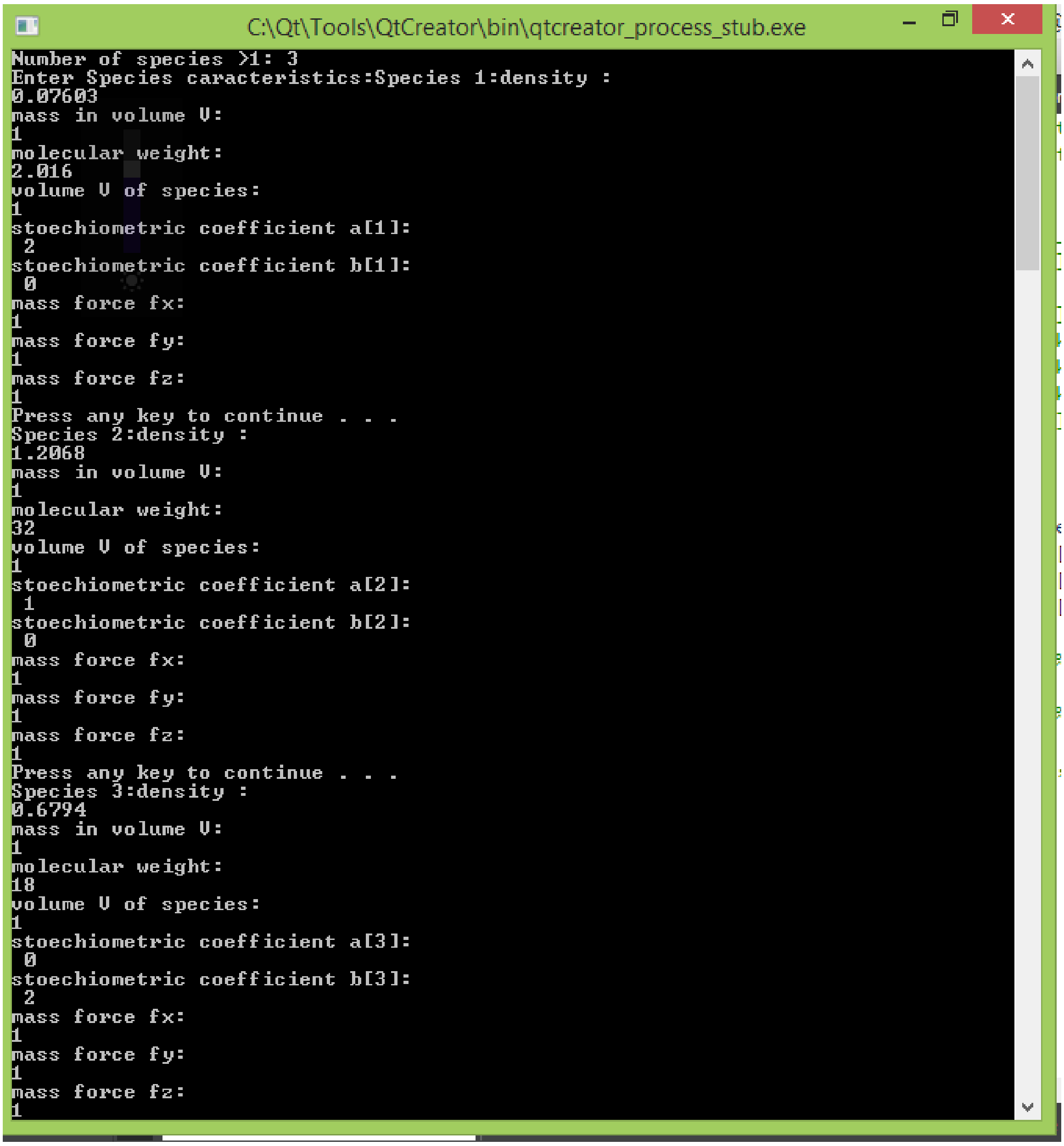


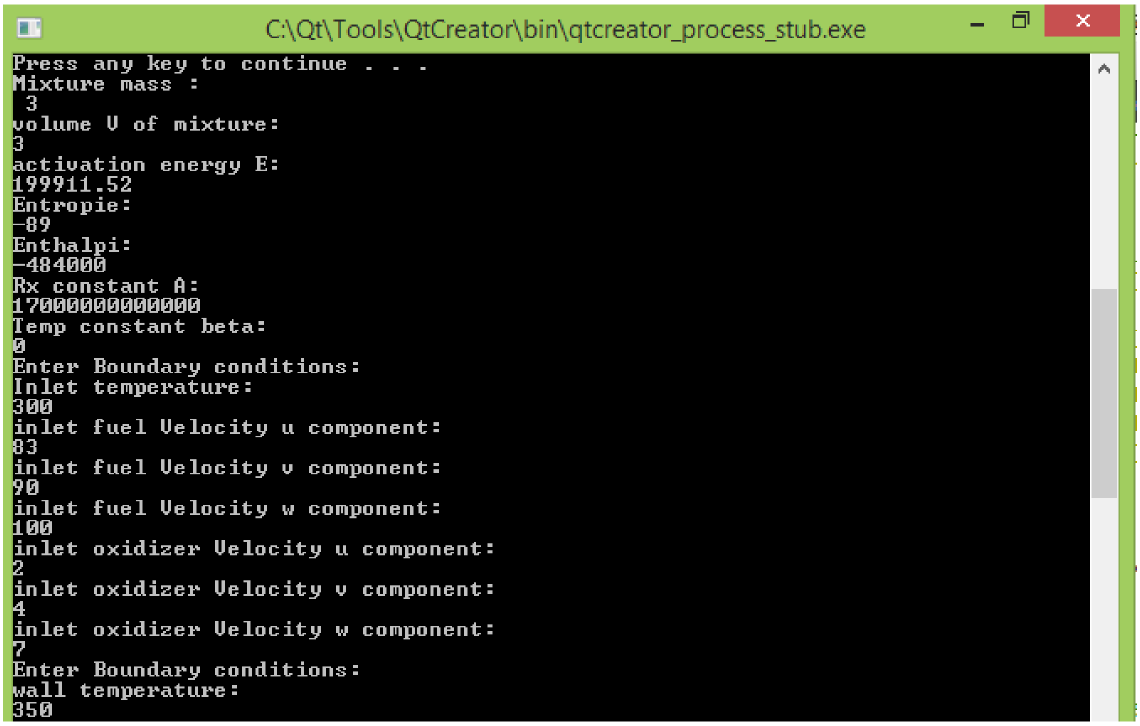
The boundary conditions are imposed as shown in the figure below[5]



[5] ''An introduction to computational fluid dynamics'' H. K. VERSTEEG & W. MALALASEKERA

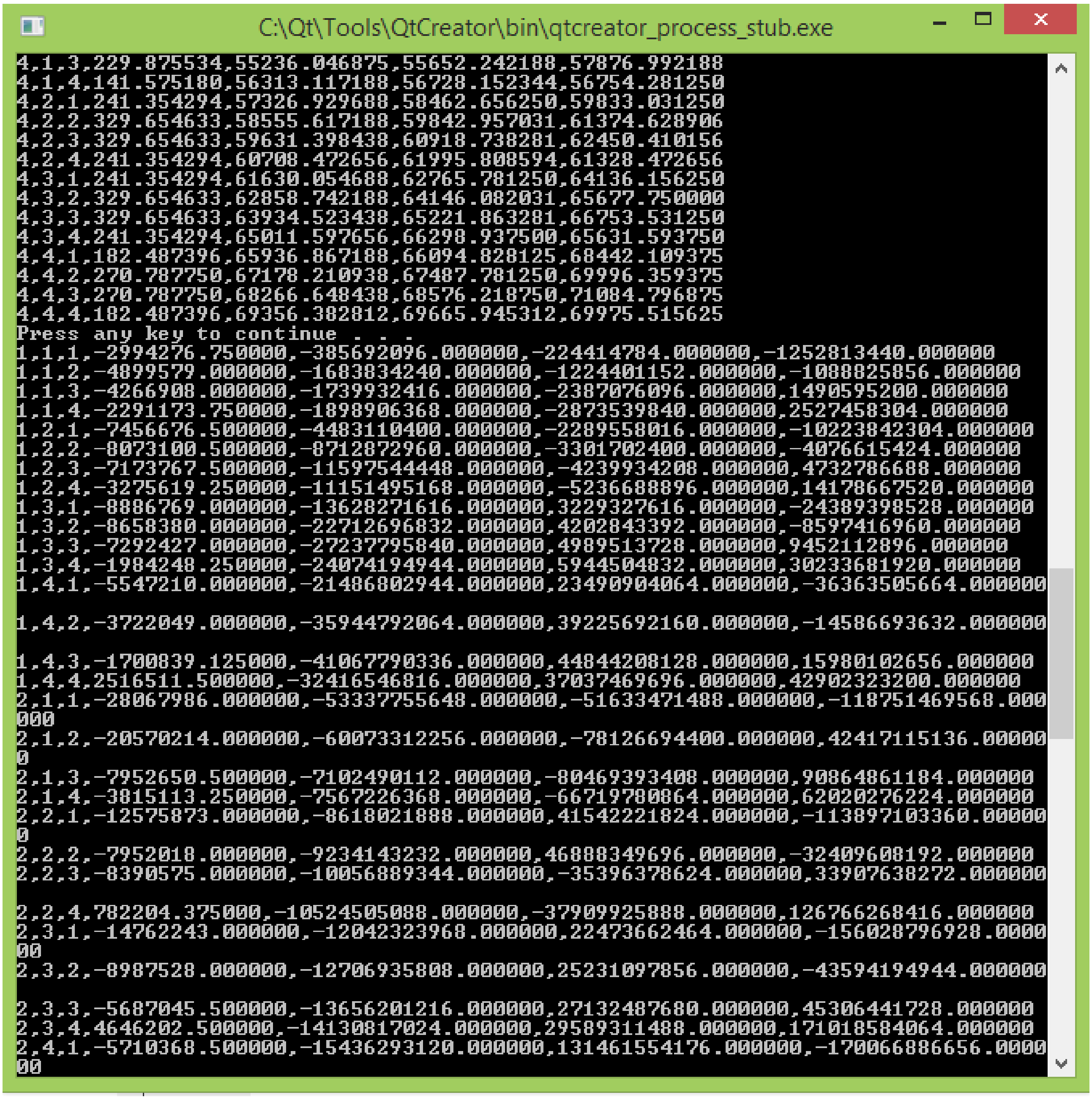
3. Program Input



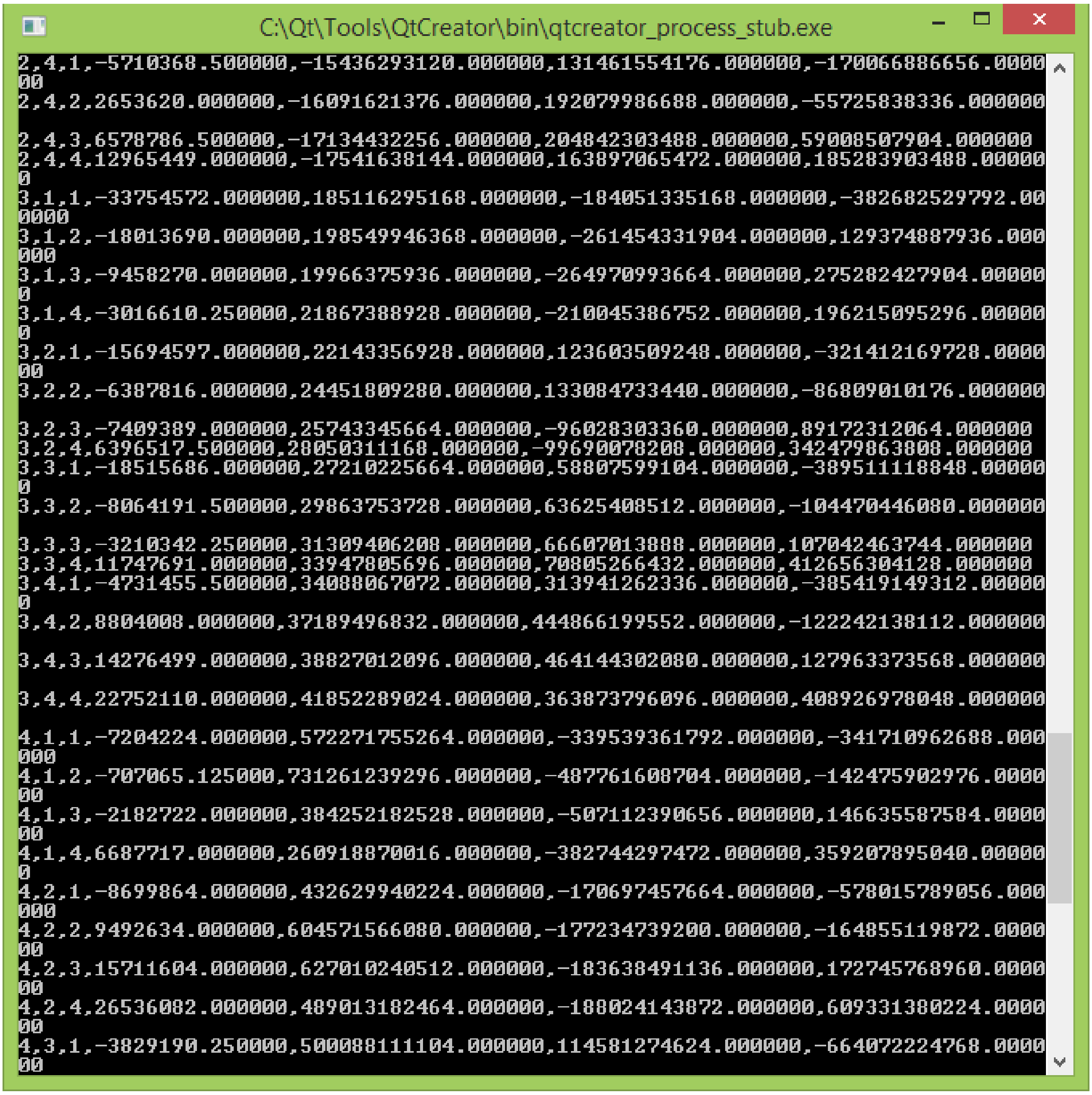

C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Press any key to continue . . .
Mixture mass :
3
volume V of mixture:
3
activation energy E:
199911.52
Entropie:
-89
Enthalpi:
-484000
Rx constant A:
17000000000000
Temp constant beta:
0
Enter Boundary conditions:
Inlet temperature:
300
inlet fuel 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

4. Program Results

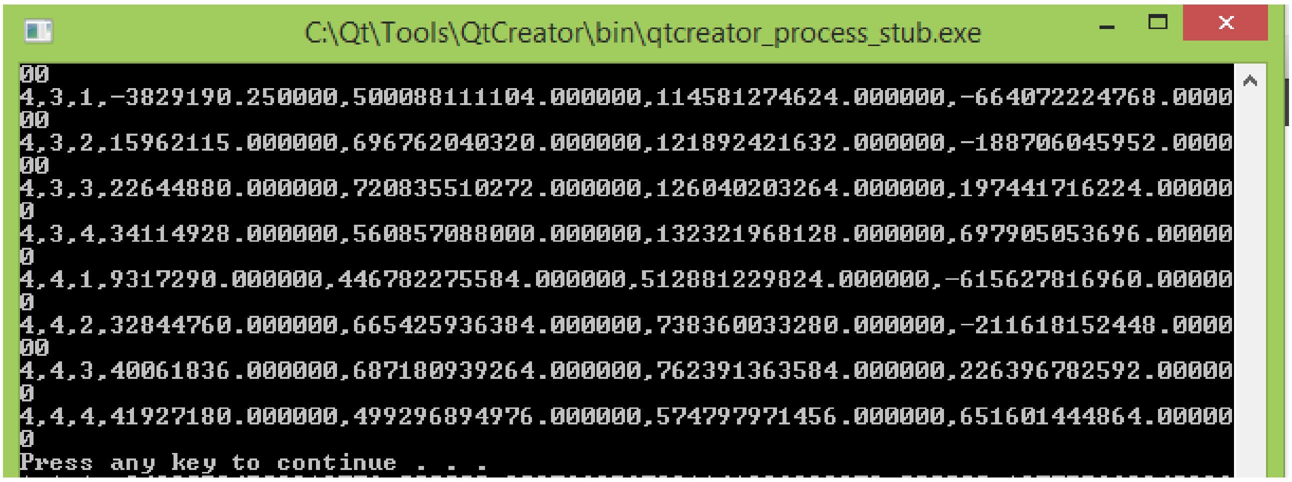
```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Press any key to continue . . .
x,y,z,density,velocity_u,velocity_v,velocity_w
1,1,1,141.575180,1334.039673,1304.178711,3319.248047
1,1,2,229.875534,2565.744385,2687.626221,4863.949707
1,1,3,229.875534,3649.269043,3770.892578,5943.686035
1,1,4,141.575180,4729.168945,4850.921875,4824.661621
1,2,1,241.354294,5749.621094,6589.827637,7909.019531
1,2,2,329.654633,6981.762207,7974.874023,9454.932617
1,2,3,329.654633,8062.289551,9055.401367,10535.459961
1,2,4,241.354294,9142.159180,10136.561523,9417.179688
1,3,1,241.354294,10069.490234,10909.696289,12228.888672
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
2,3,3,359.088104,29732.865234,30775.869141,32307.541016
2,3,4,270.787750,30816.267578,31859.269531,31191.931641
2,4,1,211.920837,31745.041016,31658.662109,34005.945312
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
3,3,4,270.787750,48043.429688,49086.433594,48419.097656
3,4,1,211.920837,48965.875000,48879.496094,51226.781250
3,4,2,300.221191,50194.562500,50259.796875,52768.378906
3,4,3,300.221191,51270.343750,51335.578125,53844.160156
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
```




```
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,-118751469568.000
000
2,1,2,-20570214.000000,-60073312256.000000,-78126694400.000000,42417115136.00000
0
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,-113897103360.00000
0
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,-156028796928.0000
00
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,-170066886656.0000
00
```



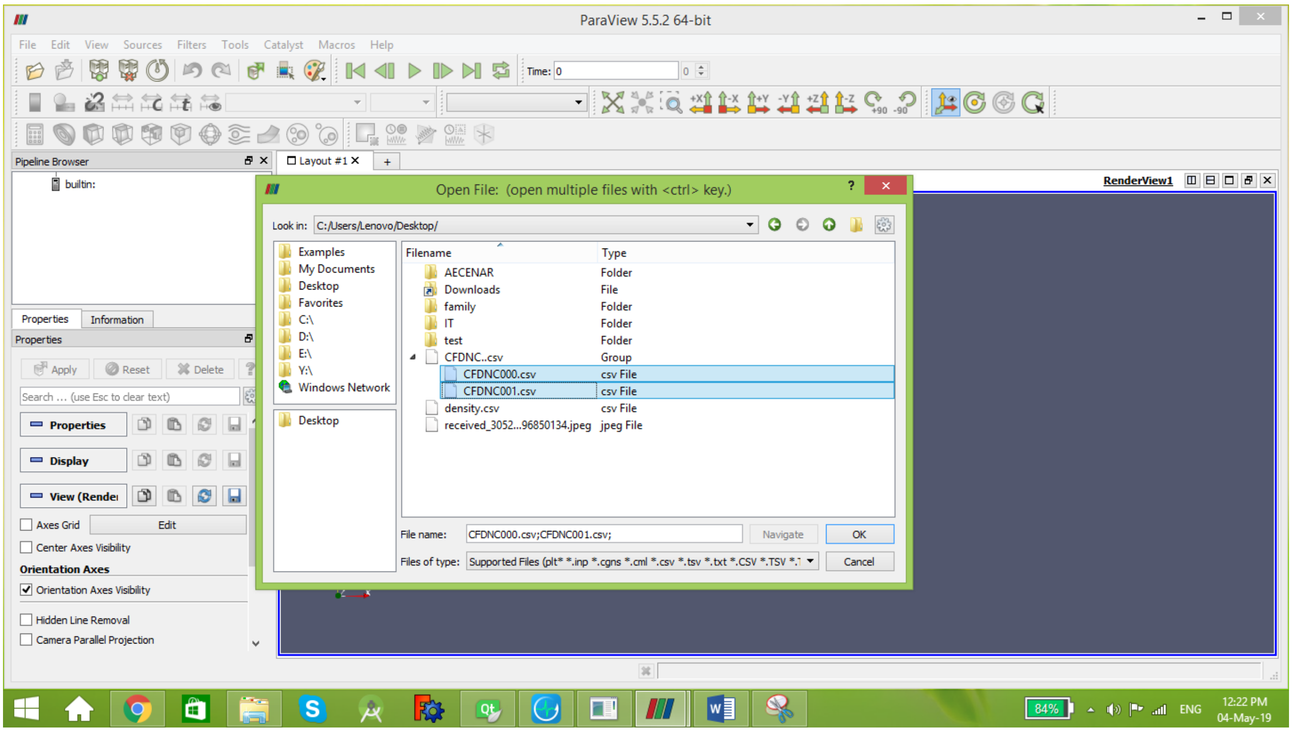
```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
2,4,1,-5710368.500000,-15436293120.000000,131461554176.000000,-170066886656.0000
00
2,4,2,2653620.000000,-16091621376.000000,192079986688.000000,-55725838336.000000

2,4,3,6578786.500000,-17134432256.000000,204842303488.000000,59008507904.000000
2,4,4,12965449.000000,-17541638144.000000,163897065472.000000,185283903488.00000
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,-264970993664.000000,275282427904.00000
0
3,1,4,-3016610.250000,21867388928.000000,-210045386752.000000,196215095296.00000
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,-389511118848.00000
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,-385419149312.00000
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,-382744297472.000000,359207895040.00000
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
```



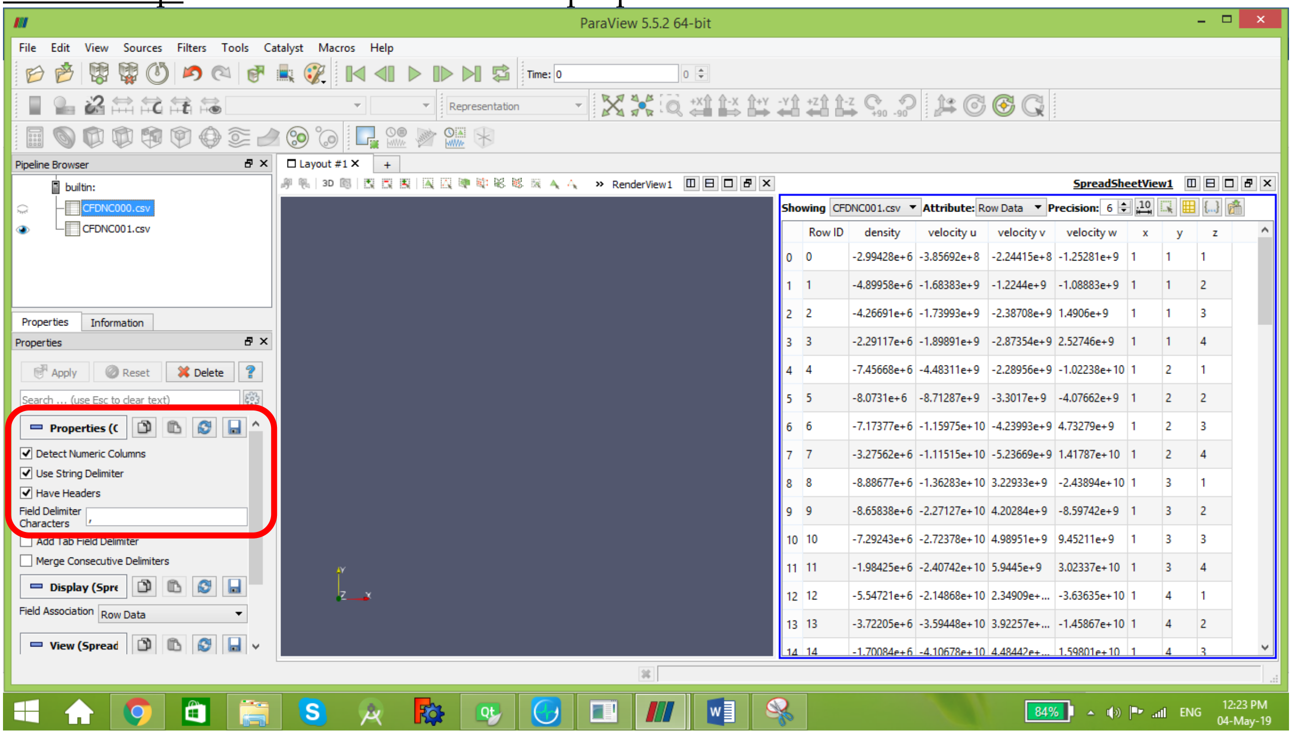
```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
00
4,3,1,-3829190.250000,500088111104.000000,114581274624.000000,-664072224768.0000
00
4,3,2,15962115.000000,696762040320.000000,121892421632.000000,-188706045952.0000
00
4,3,3,22644880.000000,720835510272.000000,126040203264.000000,197441716224.00000
0
4,3,4,34114928.000000,560857088000.000000,132321968128.000000,697905053696.00000
0
4,4,1,9317290.000000,446782275584.000000,512881229824.000000,-615627816960.00000
0
4,4,2,32844760.000000,665425936384.000000,738360033280.000000,-211618152448.0000
00
4,4,3,40061836.000000,687180939264.000000,762391363584.000000,226396782592.00000
0
4,4,4,41927180.000000,499296894976.000000,574797971456.000000,651601444864.00000
0
Press any key to continue . . .
```

5. Results viewed on Paraview

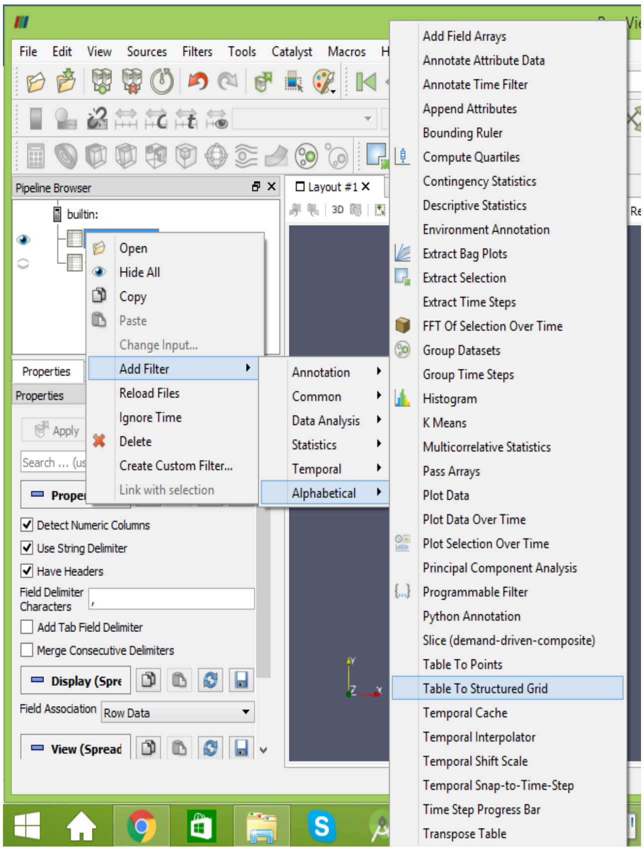
First step: open your .csv files

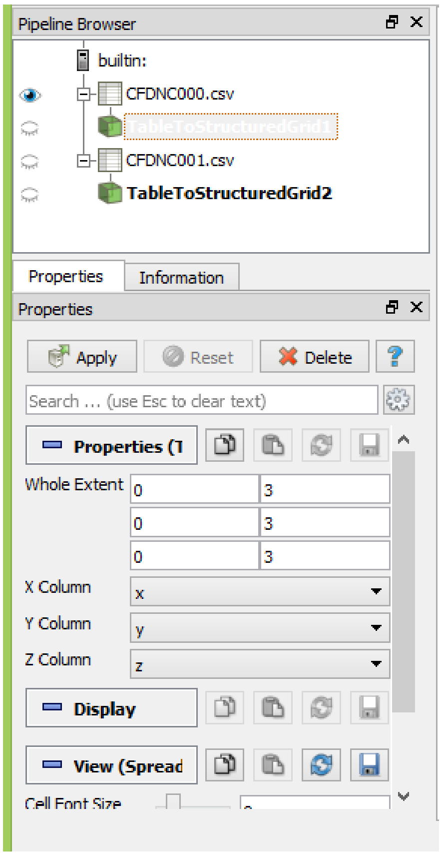


Second Step: make sure that the correct proprieties are enabled.

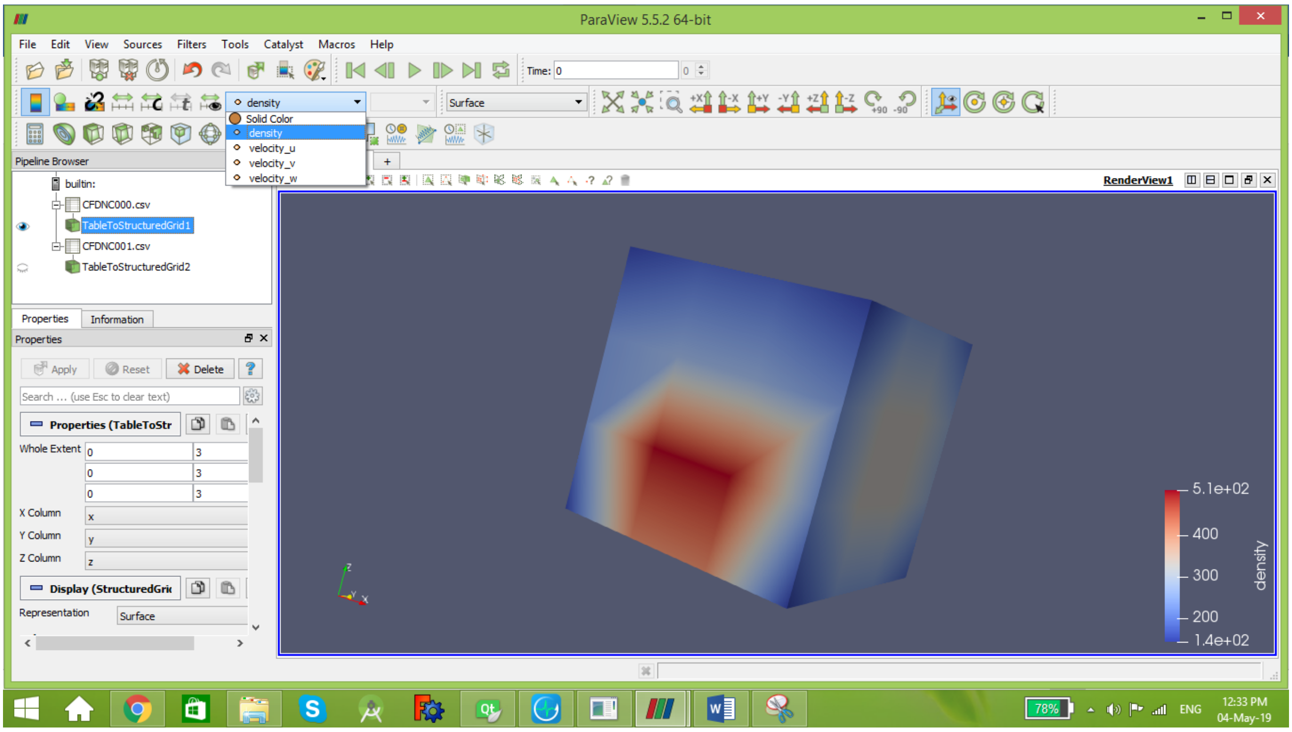


Third Step: Add filter for each file and fill with the right parameters: whole extent (0,3) for each (in our example the X, Y and Z axis are divided to 4 points), don't forget to define which columns of your data are the X, Y and Z columns.





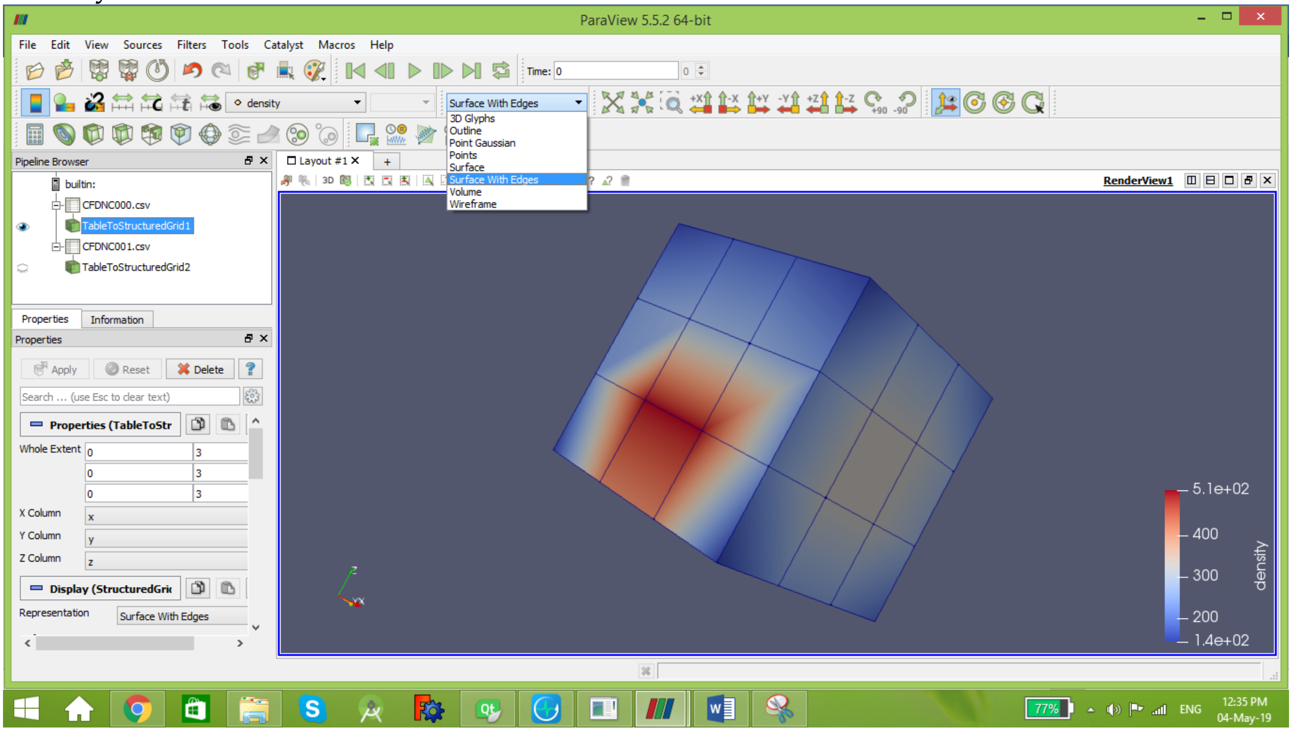
Fourth Step: Choose your variable and the desired view



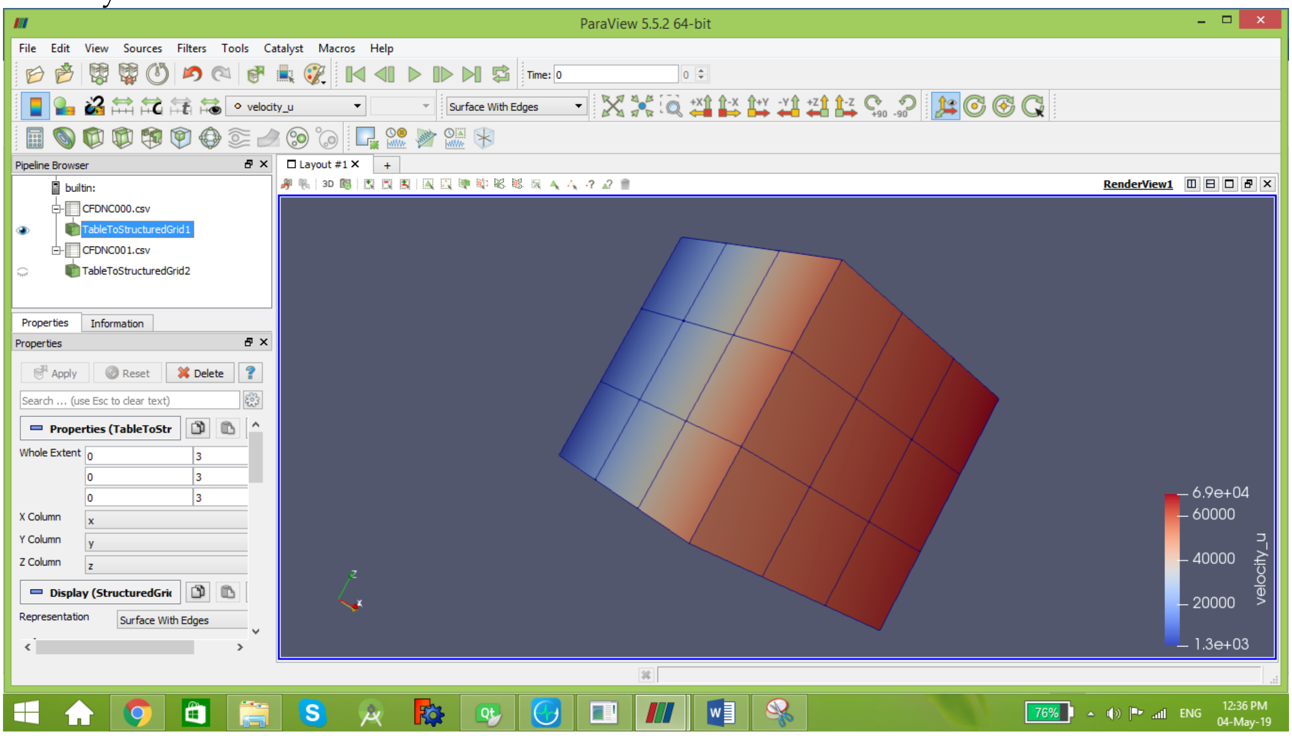
You can see your meshing by choosing the "surface with edges" option.

First time step:

Density:

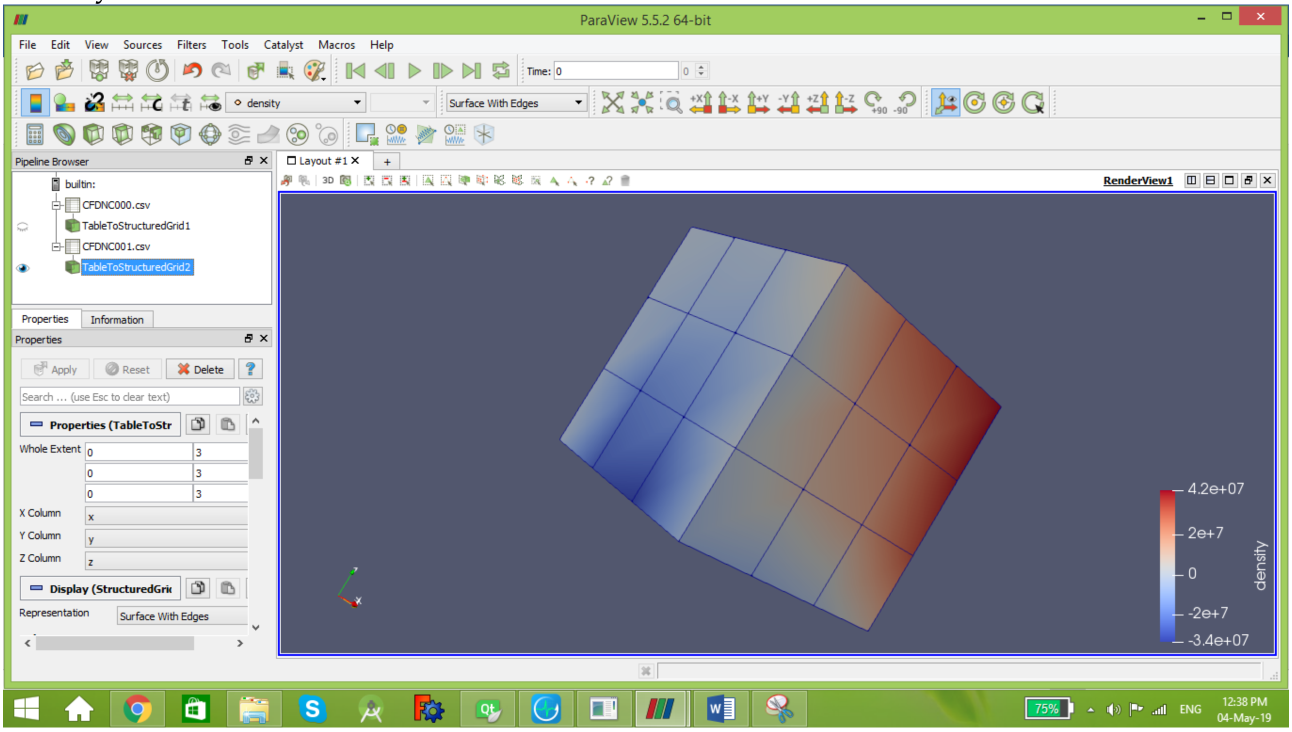


Velocity u:

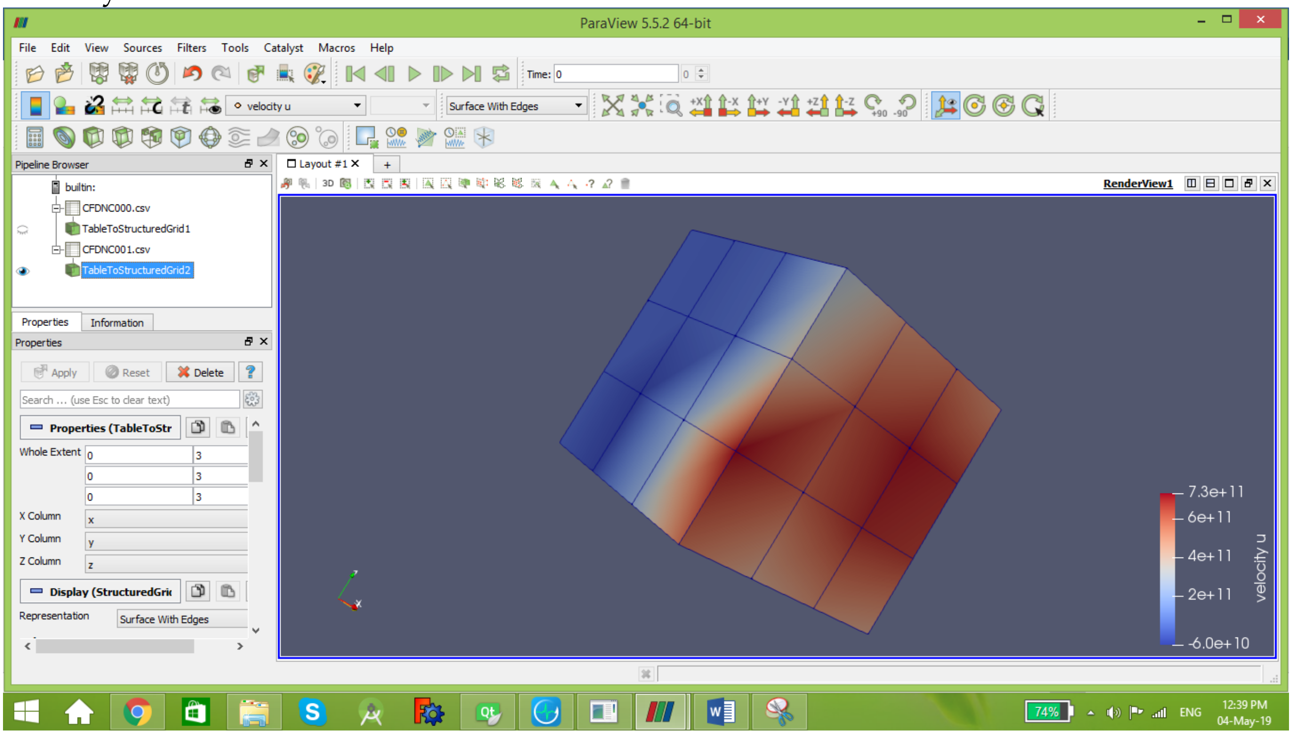


Second time step:

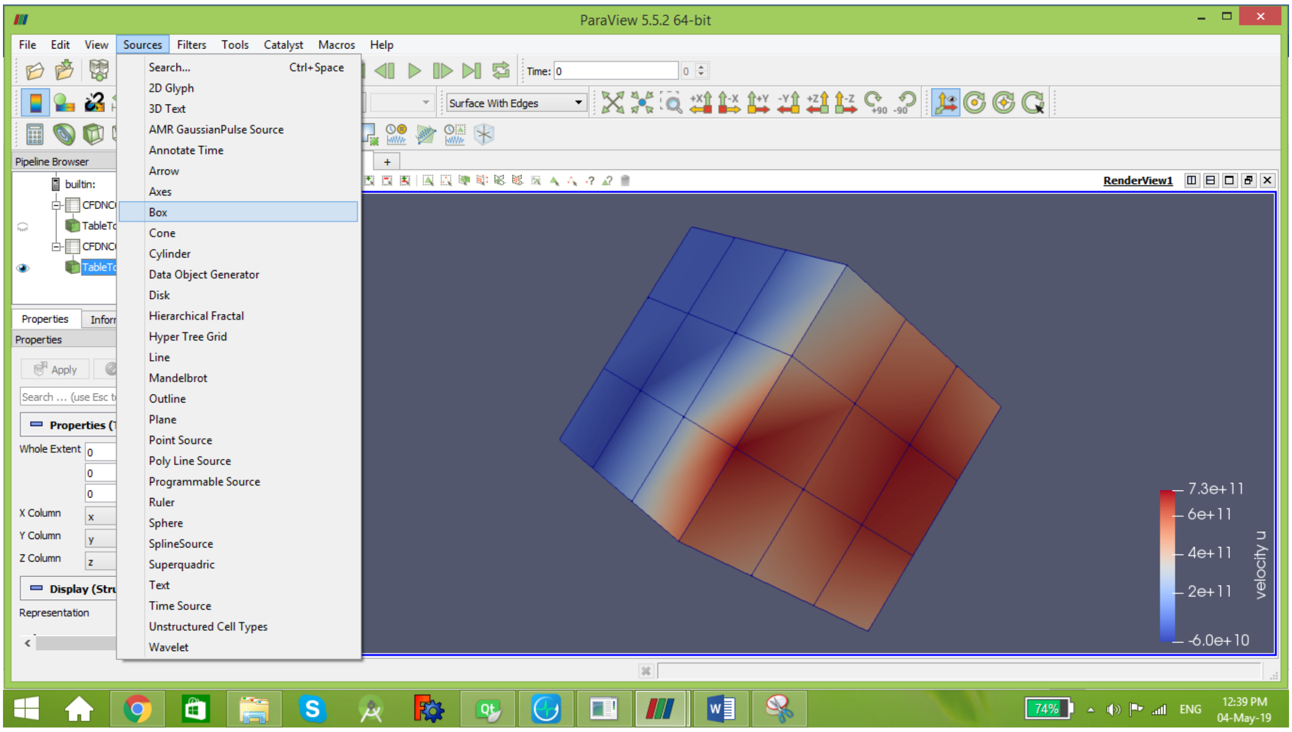
Density:



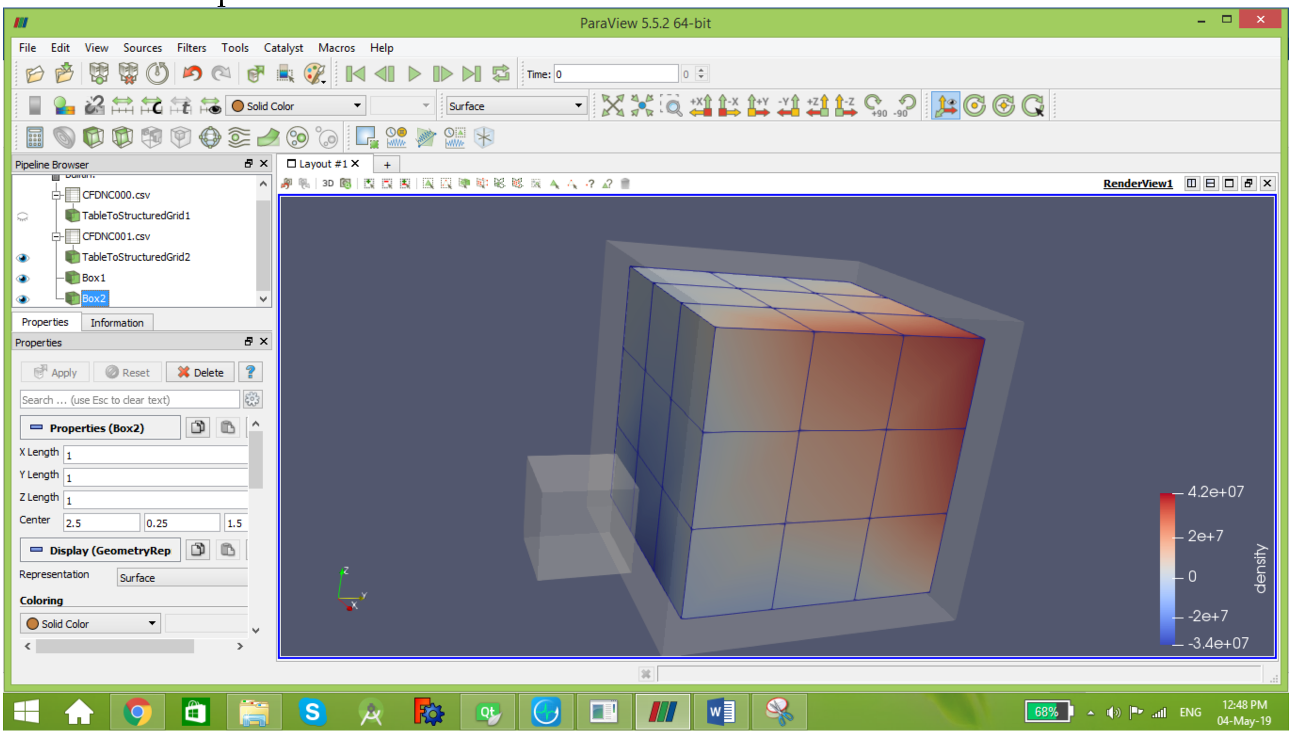
Velocity u:



<u>Fifth Step(Optional): </u>You can also add outline shape for your simulation



Choose the shape then its dimensions

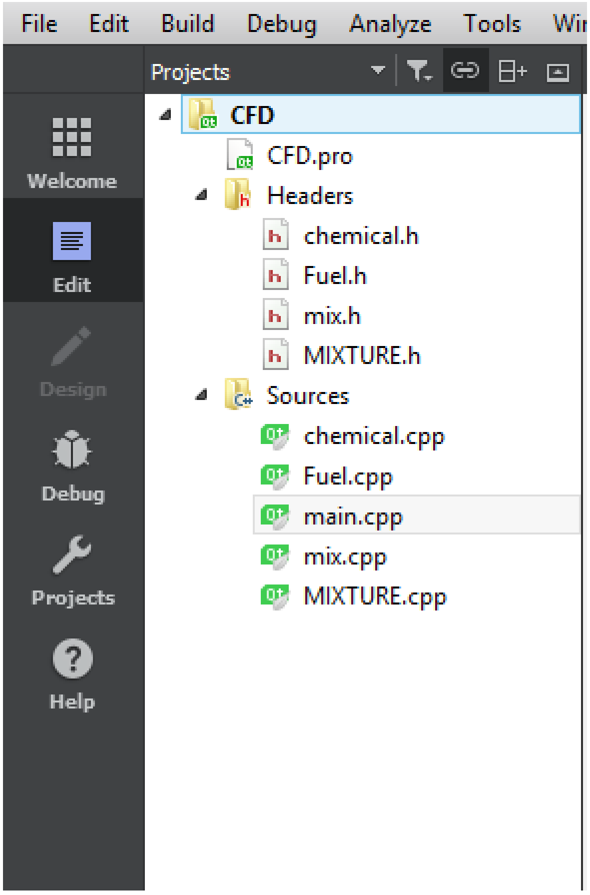


Annex A

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.



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;
}
```

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;
    }
```

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;
    }
```

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_caracteristic(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_caracteristic(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)+Fx;


                     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");
     }
}
```

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 caracteristics:");
     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("stoechiometric coefficient a[%d]:\n ",i);
        scanf("%f",&a1[i]);
        printf("stoechiometric 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 caracteristics

    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 caracteristics

    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 stoechiometric 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_caracteristic(viscositymu_mix,viscositylambda_mix,density_mix,densit
y_S,ui1,vi1,wi1,ui2,vi2,wi2,Ti,Pi,Tw,YS,NB,fx,fy,fz);

    return 0;
}
```