



Lab Migration Report

On

**Modelling And Simulation Of Passive Scalar As Reactive
Chemical Species Undergoing First Order Chemical
Reaction In An Ideal Plug Flow Reactor**

Proposed By:

Prof. Dhiraj Garg

Solution By:

John Pinto

Manjil Sitoula

Table of Contents

1. Governing Equations and Models	3
1.1 Problem Statements.....	3
1.2 Governing Equations.....	3
1.3 Geometry and Mesh.	5
1.4 Solver Setup	6
1.4.1 Fluid Properties	6
1.5 Case Setup	6
1.6 Initial and Boundary Conditions	10
1.6.1 Flow Field	10
1.6.2 Scalar.....	10
2. Results and Discussions.....	11

List of Figures

Fig 1: Schematic Diagram of Computational Domain	5
Fig 2: Mesh Generation using BlockMesh	5
Fig 3 Tree diagram of the main folder	6
Fig 4 Detailed Tree diagram of the directories	7
Fig 5 Scalar Concentration along the tube.....	11
Fig 6 Scalar concentration at 0.375 m at 20 seconds.....	Error! Bookmark not defined.
Fig 7 Scalar concentration at 0.375 m after steady State	Error! Bookmark not defined.
Fig 8 Scalar concentration along a line at the cross section of Outlet after steady state	Error! Bookmark not defined.
Fig 9 Scalar Concentration along various sections of the tube.....	Error! Bookmark not defined.
Fig 10 Scalar concentration vs time at the Outlet	Error! Bookmark not defined.

List of Tables

Table 1-1 Boundary Conditions for U	10
Table 1-2 Boundary Conditions for p	10
Table 1-3 Boundary Conditions for scalar.....	10

1. Governing Equations and Models

1.1 Problem Statements

To model **passive scalar** as **reactive** chemical species following first order **chemical reaction** and simulate the reaction in an **ideal plug flow reactor**.

Objective – The purpose is to understand how a passive scalar can be used as reactive chemical species (with or without diffusion) with flow condition. To do this we need a model for chemical reactions involving desired chemical species. The chemical reaction term is incorporated as **source term** in the passive scalar transport equation. It is this source term which differentiates whether the chemical species is reactive or non-reactive. Besides, numerical value of the source term is negative for reactants, which means consumption and positive for the products which means generation. Simulating the domain as ideal plug flow reactor and comparing the results with known results is important to verify the correctness of the model of the reaction rate implemented along with the flow conditions in the code. How to simulate the domain with moving flow as an ideal plug flow reactor is equally important, that the user will learn here.

1.2 Governing Equations

The continuity and momentum conservation equations are solved for the calculation of flow parameters like velocity and pressure. The conservation equations are expressed below:

Continuity Equation:

$$\nabla \cdot \mathbf{u} = 0 \quad (1)$$

Momentum Equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} (\nabla p) + \nu \nabla^2 \mathbf{u} \quad (2)$$

where ν is the kinematic viscosity, ρ is the density, \mathbf{u} is the velocity vector and p is the pressure.

A concentration transport equation is incorporated for the calculation of passive scalar concentration using modified solver.

Equation to model the transport of Passive Scalars (S1 & S2):

$$\frac{\partial S}{\partial t} + \mathbf{u} \cdot \nabla S = \nabla \cdot (\Gamma \nabla S) + S_c \quad (3)$$

Where S (in our case S1 & S2) is the passive scalar, Γ is the diffusion coefficient and S_c is the source term of respective passive scalar. The source term would be required to model chemical reaction related to S1 and S2.

Let the chemical reaction be $A \rightarrow B$ and assume the rate of reaction is of first order in a batch reactor, i.e.

$$-r_A = -\frac{dC_A}{dt} = kC_A = r_B = \frac{dC_B}{dt} \quad (4)$$

The analytical solution for this equation is

$$C_A = C_{A0} \exp(-kt) \quad (5a)$$

where C_{A0} is initial concentration of A. Similarly, for B, the solution would be

$$C_B = C_{A0}[1 - \exp(-kt)] \quad (5b)$$

where the initial concentration of B would be zero. The above solution can be obtained by integrating $-r_A = -\frac{dC_A}{dt} = \frac{dC_B}{dt} = r_B \rightarrow -dC_A = dC_B$ from 0 to t.

Let S1 represents the concentration of chemical species A and S2 represents concentration of chemical species B. As we can see, A is reactant and will be consumed in the reaction whereas B is product and thus will be produced during the reaction. So, the source term for S1 would be

$$S_{C1} = -r_A = -kC_A \quad (6a)$$

and for S2, it would be

$$S_{C2} = r_B = kC_A \quad (6b)$$

So the transport equation for C_A can be written as

$$\frac{\partial C_A}{\partial t} + \mathbf{u} \cdot \nabla C_A = \nabla(\Gamma \nabla C_A) + (-r_A) = \nabla(\Gamma \nabla C_A) - kC_A \quad (7)$$

The ideal plug flow reactor in context of straight cylindrical tube means to have flat radial velocity profile from inlet to outlet with complete radial mixing. This will ensure that every particle which is entering at any point of the inlet will spend same residence time inside the tube irrespective of its location on entry on inlet. Thus, the time spent by any particle inside the tube is dependent on the distance from inlet only. So average and maximum velocities are the same and equal. Thus,

$$t = \frac{l}{u_{max}} = \frac{l}{\bar{u}} \quad (8)$$

So equation (5) will become

$$C_A = C_{A0} \exp\left(-k\left(\frac{l}{\bar{u}}\right)\right) \quad (9a)$$

$$C_B = C_{A0} \left[1 - \exp\left(-k\left(\frac{l}{\bar{u}}\right)\right)\right] \quad (9b)$$

The purpose of this exercise is to check whether we are getting correct results from the simulation for clubbing reaction scheme with flow conditions by comparing the results against the analytical solution shown in eqn. (9). So the plot will be along the length in the direction of flow instead of time. This process of comparing the results to check the correctness of the model is called verification. This does not require large number of meshes in straight cylindrical tube as velocity profile will be flat. The diffusion coefficient of scalars can be made zero and non-zero to check

whether it has any impact on the simulations. We want to observe whether the concentration is decreasing because of reaction alone (as the case should be) or due to flow also (convection and diffusion) – which should not be the case. Any deviation in space or time will indicate the potential issue either with model or simulation. This will enable the user to identify the problem and rectify it before fully implementing the reaction scheme along with flow in actual domain.

For more information regarding the source term, you can refer to this document [Theory](#).

1.3 Geometry and Mesh.

The domain is a straight cylindrical tube with length of 0.5 m and diameter of the cross section as 0.01 m. The geometry is 3D. The geometry is long enough for the flow to fully develop. The scalar S1 will be injected at the inlet.

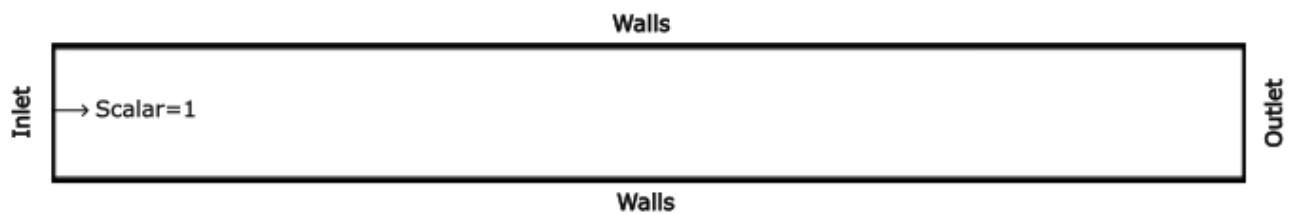


Fig 1: Schematic Diagram of Computational Domain

The meshing is done using blockMesh utility, openFOAM's built-in tool. The geometry is divided into 5 blocks and 18 vertices and the number of cells is 672000 for this case. The user is free to choose different types of meshing and numbers to get the similar results. For the detailed meshing process, one can go through the openFOAM spoken tutorial number 6. [spoken tutorial](#)

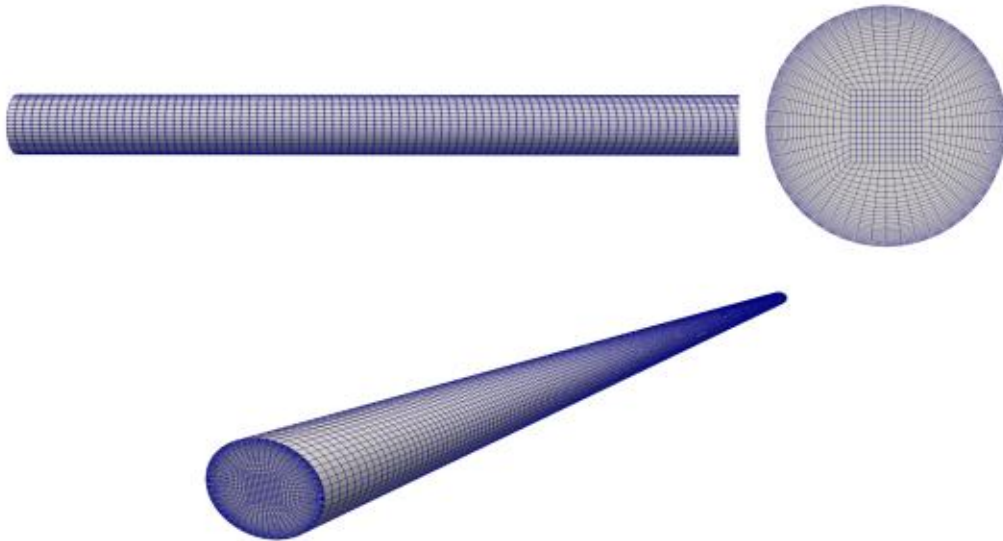


Fig 2: Mesh Generation using BlockMesh

1.4 Solver Setup

1.4.1 Fluid Properties

Water at room temperature is used as the fluid and the thermo physical properties of water are taken for calculations. The kinematic viscosity of water at room temperature (300 K) is $8.58 \times 10^{-7} \text{ m}^2/\text{s}$ and the average velocity of the flow is 0.1 m/s , which is half the maximum velocity. The Reynolds number is expressed as:

$$Re = \frac{U_{avg} * D}{\nu}$$

Where U_{avg} is the average velocity, D is the diameter of the tube and ν is the kinematic viscosity. For the above input flow parameters, the Reynolds number of the flow is 1165 which is in the laminar regime (<2100). So, a laminar model is used for the simulation.

Flow Parameters	Value
Max. Velocity (U_{max})	0.2 m/s
Average Velocity (U_{avg})	0.1 m/s
Density (ρ)	1000 kg/m ³
kinematic viscosity (ν)	$8.58 \times 10^{-7} \text{ m}^2/\text{s}$
Reynolds No.	1165
Scalar Diffusivity constant (DS1)	0 m ² /s
Scalar Kinetic rate coefficient(kS1)	1 s ⁻¹

1.4.2 Case Setup

The case files for the current session are available in this [link](#). Download and extract these files into your run directory. A general overview of the setup is explained below:

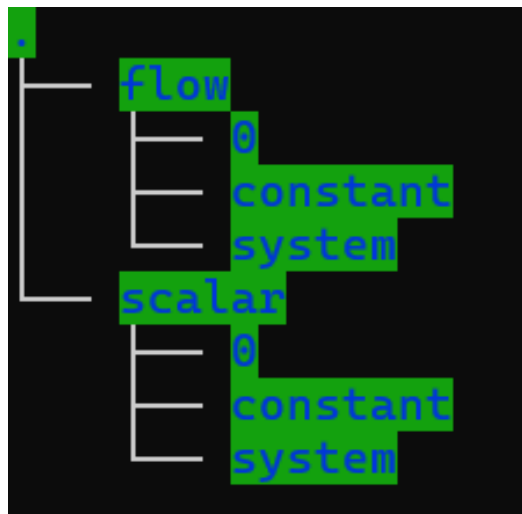


Fig 3 Tree diagram of the main folder

The main folder consists of flow and scalar folder as in Fig 3. The flow folder consists of files to solve the flow field. The scalar folder has the necessary files to simulate the scalar. Since, newScalarTransportFoam1S, does not solve for velocity, we need to first simulate the flow field. So, initially, the steady state velocity field is solved using flow folder. This flow field is then used by the scalar to move along the flow through local velocity. This is explained more in section 1.6. The folders can be further expanded along this tree:

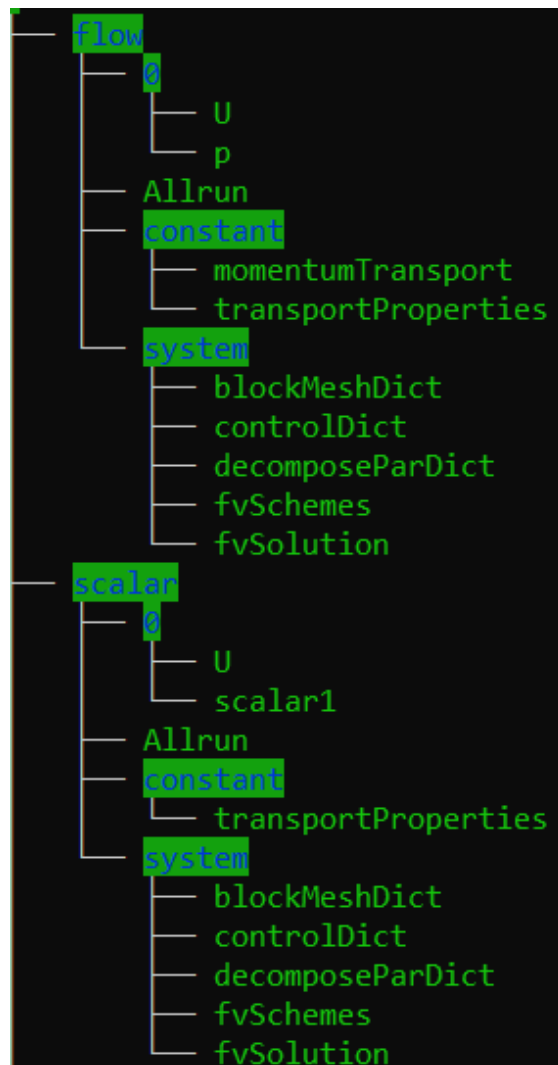


Fig 4 Detailed Tree diagram of the directories

The 0 directory of scalar folder consists of initial and boundary conditions for scalar1. Scalar 1 is used to model one of the chemical species which after reaction is converted to another species.

Flow case setup:

The initial and boundary conditions for velocity and pressure are provided in the U and p files of 0 directory. You can see the boundary conditions by accessing these files. The boundary conditions are further explained in the next section.

- The kinematic viscosity of fluid is provided in **constant/transportProperties**.

```
transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 8.58E-07;
```

- Similarly, in momentumTransport dictionary, type of model for the simulation is provided. In our case, we have used the laminar model.

```
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      momentumTransport;
}

simulationType laminar;
```

- The blockMeshDict consists of mesh information and the controlDict dictionary consists of case controls like timing, write information etc.
- System/decomposeParDict dictionary is used for parallel computing.

The steps for the simulation are provided below:

1. First, you need to navigate to the flow folder in your run directory.

```
cd $FOAM_RUN
```

```
cd plugFlow_New/flow
```

2. The Allrun file consists of necessary commands to run the simulation. Type **./Allrun** and press enter.

The Allrun file consists of following commands:

```
blockMesh
decomposePar
mpirun -np 6 simpleFoam -parallel
reconstructPar
```

The blockMesh command is used to generate the mesh. Command decomposePar decomposes the domain into subdomains and assigns the number of processors to these subdomains based on the method like simple, scotch etc. In this case, 6 processors are used in parallel and simpleFoam solver is used. At last, reconstructPar command is used to reconstruct a single domain from the processor sub-domains.

Scalar Case setup:

Solver modification and compilation

To simulate the scalar, we have modified the scalarTransportFoam solver and named it newScalarTransportFoam. To make executables for this solver we will need to compile it first. To do that follow the steps given below:

1. Open your terminal and navigate to the run directory.
cd \$FOAM_RUN
2. Navigate to the solver folder by typing the following command.
cd newScalarTransportFoam
3. Compile the solver by typing the following command and press enter.
wclean
wmake

After this the following steps are required.

1. First, you need to navigate to the flow folder.

cd plugFlow_New /scalar

2. The Allrun file consists of necessary commands to run the simulation. Type **./Allrun** and press enter.

The Allrun file consists of the following commands:

```
blockMesh
decomposePar
mpirun -np 4 newScalarTransportFoam -parallel
reconstructPar
```

These commands are explained in the previous section.

Transport Properties:

```
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}

DS1            [0 2 -1 0 0 0 0] 0;
DS2            [0 2 -1 0 0 0 0] 0;
kS             [0 0 -1 0 0 0 0] 1;
```

Here, DS1 is the diffusivity and kS1 is the kinetic rate coefficient. For this case, there is no diffusion of both scalars and since it models chemical reaction the kinetic rate coefficient is set to 1.

1.4.3 Initial and Boundary Conditions

The initial and boundary conditions for the flow and scalar simulation are used separately. The scalar term represents the concentration of the passive scalar in the simulation. At first, the flow is simulated as steady state to obtain a velocity field through which scalar moves. The flow field conditions for the scalar simulation are provided by the flow simulation and a new solver is implemented to simulate the concentration of passive scalar. For the flow simulation, simple Foam, a steady state solver is used whereas newScalarTransportFoam, an incompressible transient solver is used for the scalar. The initial and boundary conditions are provided in the 0 directory which contains U and Scalar1 dictionary.

1.4.3.1 Flow Field

The initial conditions are set to zero and the necessary boundary conditions for the flow are tabulated below:

Table 1-1 Boundary Conditions for U

Patch	Condition
Inlet	fixedValue(uniform 0.1)
Outlet	zeroGradient
Walls	fixedValue(uniform 0.1)

Table 1-2 Boundary Conditions for p

Patch	Condition
Inlet	zeroGradient
Outlet	fixedValue(uniform 0)
Walls	zeroGradient

Moving wall Boundary condition instead of no-slip at walls is to be implemented for ensuring flat velocity profile throughout the flow.

A steady state simulation is done to calculate the steady velocity profile in the tube which is used as an input to the scalar which is carried out as transient simulation.

1.4.3.2 Scalar1

1.4.3.2.1 Initial Conditions

The steady state velocity profile obtained from the flow simulation is used as a local flow field for the scalar to trace the path of flow. The U file of last time step from the flow simulation is kept in the 0 directory of the scalar1. Scalar1 is injected at the inlet cross section with concentration of 1.

1.4.3.2.2 Boundary conditions

The flow field is used from the flow simulation and the boundary conditions for the scalar1 is tabulated below:

Table 1-3 Boundary Conditions for scalar1

Patch	Condition
Inlet	fixedValue(uniform 1)
Outlet	zeroGradient
Walls	zeroGradient

1.4.3.2.3 Source term

The scalar S1 is consumed during reaction which is modeled by newScalarTransportFoam solver through source term.

```

while (simple.correctNonOrthogonal())
{
    fvScalarMatrix scalar1Eqn
    (
        fvm::ddt(scalar1)
        + fvm::div(phi, scalar1)
        - fvm::laplacian(DS1, scalar1)
        ==
        -kS1*scalar1 ← Source
    );

    scalar1Eqn.relax();
    fvConstraints.constrain(scalar1Eqn);
    scalar1Eqn.solve();
    fvConstraints.constrain(scalar1);
}

```

1.4.3.3 Steady State Study of Scalar

The scalar steady case is to see the steady state of the simulation. If one is interested in the final results or when the solution doesnot change with time and don't want to trace the scalar with time, once can navigate to this folder **cd plugFlow_New /scalar_steady** and type **./Allrun** and press enter. This only provides the steady state solution and observe the final state of the scalar.

2. Results and Discussions

The scalar is injected from the inlet with concentration 1 and gradually moves along the tube with time.

The scalar concentration along the tube is shown below at $t = 10s$ in Fig 5 which is convected by the flow field. Since the velocity field is uniform (flat profile) and also moving velocity of 0.1 m/s at the wall, the scalar moves uniformly and has a uniform concentration along the direction perpendicular to the flow. As it moves, it is consumed as in the reactant and decays exponentially along the length of the tube in the flow direction.

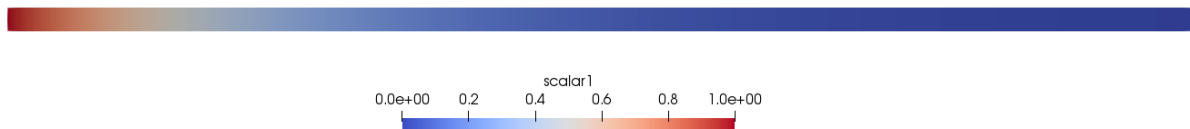


Fig 5 Scalar Concentration along the tube at $t=10s$

A line is drawn from inlet to outlet in the flow direction and the scalar concentration along the length is plotted shown in **Error! Reference source not found..** The exponential decrease of the scalar is observed which is modeled by newScalarTransportFoam solver. This is analogous to the concentration vs time for a batch reactor. If we divide the length (x axis) by velocity (0.1m/s) and plot the concentration, it provides the concentration vs time graph as in the case of a batch reactor. This signifies that each plug acts as a batch reactor or a batch reactor at different time being placed along the length.

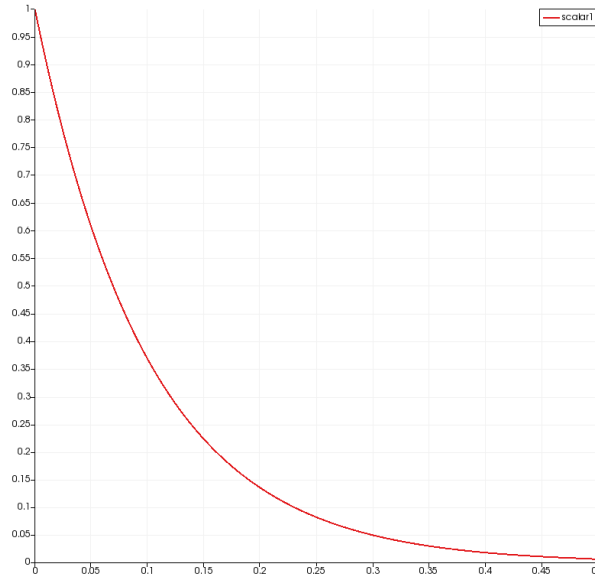


Fig 6 Scalar Concentration along the length at $t=10s$

The user can change the value of kinetic rate coefficient and observe the effect on the decay as to how fast and how slow it is happening. The user can also implement higher order reaction rate given by

$$-r_A = \frac{dC_A}{dt} = kC_A^n \quad (10)$$

where n is the order of the reaction and can have any value from 0 onwards. The user can obtain an analytical solution corresponding to this and compare the results obtained by the simulation. Similarly, the user can obtain the concentration profile for S2.

Another important aspect about CFD is that it is based on law of conservation of mass and not moles. Moles are conserved only when there is no reaction. All rate equations are based on moles instead of mass as the units of concentration terms are in mol/vol and reaction rate is defined as mol/vol/time. The units of kinetic rate coefficient k can be obtained accordingly. So, the rate equation has to be made dimensionless in terms of moles. This is achieved by following method:

Applying following transformation in eqn. (10),

$$C'_A = \frac{C_A}{C_{A0}} \quad (11)$$

This will make initial concentration as 1 at inlet which is the case here also.

$$\text{we have } -r'_A = -\frac{dC'_A}{dt} = k \cdot C_{A0}^{n-1} C_A'^n = k' C_A'^n \quad (12)$$

$$\text{Where } k' = k \cdot C_{A0}^{n-1} \quad (13)$$

This will work for all values of n . This transformation will also achieve the purpose of normalization thus reducing the numerical errors also.