Lab Migration

Report

On

Modelling and simulation of two unmixed passive scalars as non-diffusive reactive chemical species with second order chemical reaction

Proposed By:

Prof. Dhiraj Garg

Solution By:

John Pinto

Manjil Sitoula

Table of Contents

List of Figures

List of Tables

1. Governing Equations and Models

1.1 Problem Statements

To model **passive scalars** as unmixed **reactive** chemical species **without** diffusion following **second** order **chemical reaction** and simulate the reaction in a laminar **flow reactor**.

Objective – The purpose is to understand how passive scalars can be used as reactive chemical species **without** diffusion under **flow** conditions. To do this we need a model for chemical reactions involving desired chemical species which is taken to be second order here. The chemical reaction term is incorporated as **source term** in the passive scalar transport equation. The effect of flow process (convection only) and chemical reaction on concentration of chemical species in both upper and lower half along the flow is to be observed.

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:

$$\nabla . \mathbf{u} = 0 \qquad (1)$$

Momentum Equation:

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

where ν is the kinematic viscocity, ρ is the density, **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):

$$\frac{\partial S}{\partial t} + \mathbf{u}. \nabla S = \nabla(\Gamma \nabla S) + \mathbf{S_C} \qquad \mathbf{(3)}$$

Where S (in our case S1, S2 & S3) 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, S2 and S3.

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

$$-r_A = -\frac{dC_A}{dt} = kC_A C_B = -r_B = -\frac{dC_B}{dt} = r_C = \frac{dC_C}{dt} \qquad (4)$$

C_{A0}, C_{B0} are the initial concentration of A and B, while the initial concentration of C would be zero.

Let S1, S2 and S3 represent the concentration of chemical species A, B and C respectively. As we can see, A and B are reactants and will be consumed in the reaction whereas C is product and thus will be produced during the reaction. So, the source term for S1 would be

$$S_{C1} = -r_A = -kC_AC_B \quad (5a)$$

for S2, it would be

$$S_{C2} = -r_B = -kC_AC_B \quad (5b)$$

and for S3, it would be

$$S_{C3} = r_C = kC_AC_B \quad (5c)$$

So the transport equation for C_A can be written as

$$\frac{\partial \mathrm{C_A}}{\partial t} + \mathbf{u}.\nabla \mathrm{C_A} = \nabla(\Gamma\nabla C_A) + (-\boldsymbol{r_A}) = \nabla(\Gamma\nabla C_A) - \boldsymbol{kC_A} \quad \mathbf{(6)}$$

The ideal laminar flow reactor in context of straight cylindrical tube means to have parabolic radial velocity profile from inlet to outlet with **no** radial mixing. The only way a particle can move radially would be through diffusion only. Both the reactants A and B are injected in separate halves of the cylinder thus ensuring unmixed condition. So, they must mix to cause chemical reaction. Since the diffusion is zero in the current case, theoretically, the chemical species should not diffuse to other half of the cylinder while moving along the flow in their respective halves. So, we should not observe any chemical reaction along the flow till the outlet. The purpose of this exercise is to check whether we are getting physically correct results from the simulation for no diffusion along with flow conditions. If any diffusion in the lower half of the cylinder is observed, it will cause chemical reaction which can be confirmed by the change in the concentrations of A, B and C at the cross-section especially at the interface. This diffusion, thus, must be coming from numerical diffusion. So, solver settings, higher order discretization schemes etc. have to be modified to minimize or eliminate it. Else we won't be able to observe the correct effect of diffusion on the reaction and flow.

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 as shown in Fig. 1. The geometry is 3D. The geometry is long enough for the flow to fully develop. The chemical species S1 will be injected at upper half and S2 will be injected at lower half of the inlet.

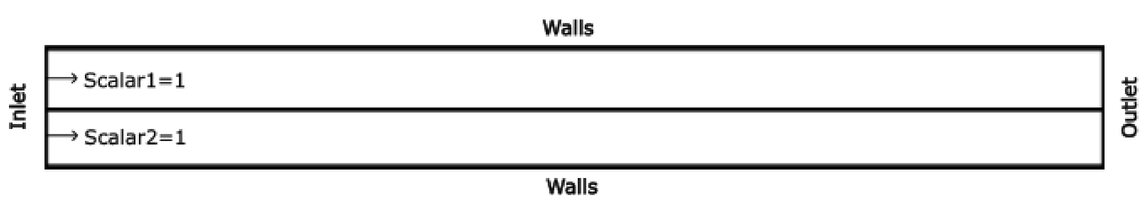

Fig 1: Schematic Diagram of Computational Domain

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

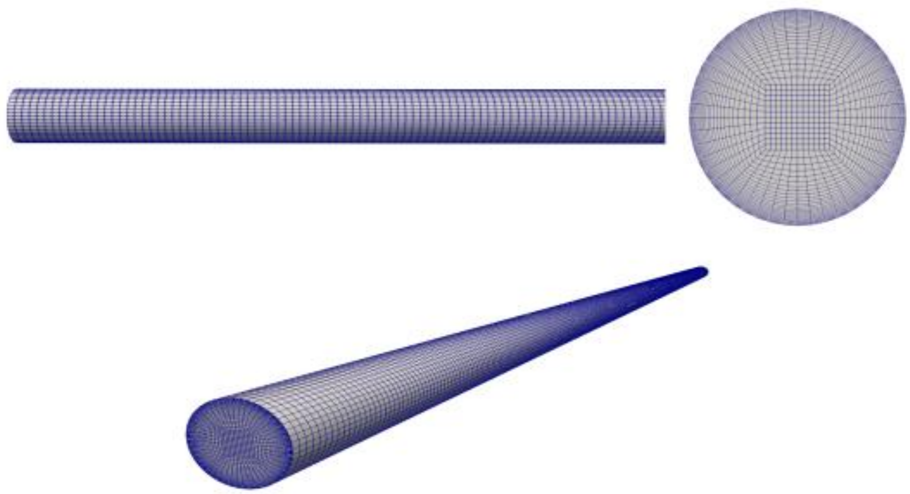

Fig 2: Mesh Generation using BlockMesh

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 $8.58 \times 10^{-7}\ m^2/s$ 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)	$8.58 \times 10^{-7}\ m^2/s$
Reynolds No.	1165
Scalar Diffusivity constant (DS1, DS2, DS3)	0 m^2/s
Scalar Kinetic rate coefficient(kS)	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:

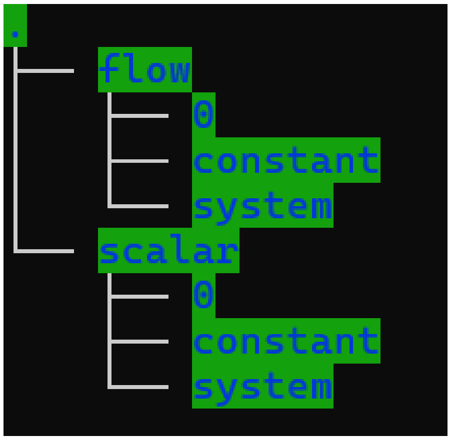

Fig 3 Tree diagram of the main folder

The main folder consists of flow and scalar folder as in Fig 3. The flow folder consists of files to solve the flow field, and the scalar folder has the necessary files to simulate the scalar. The velocity field solved using flow folder is used by the scalar to move along the tube and trace the path of the flow. Since, **newScalarTransportFoam2S**, does not solve for velocity, we need to simulate for the flow field and provide as a path for the scalar. The scalar folder is used in simulating the scalar change depicting the chemical reaction. The folders can be further expanded along this tree:

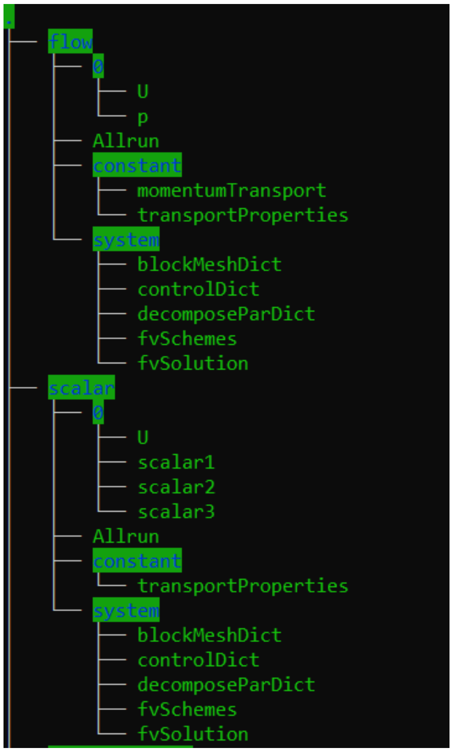

Fig 4 Detailed Tree diagram of the directories

The 0 directory of scalar folder consists of initial and boundary conditions for scalar1, scalar2 and scalar3. Scalar1 and Scalar2 are used to model the chemical species which after reaction are converted to another species, modeled by scalar3.

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 viscocity 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 HalfBore_0D_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 newScalarTransportFoam2S. 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 newScalarTransportFoam2S
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 2HalfBore_0D_Source/scalar

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

```
blockMesh
decomposePar
mpirun -np 6 newScalarTransportFoam2S -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;
DS3              [0 2 -1 0 0 0 0] 0;
kS              [0 0 -1 0 0 0 0] 1;
```

Here, DS1, DS2 and DS3 are the diffusivities of Scalar1, Scalar2 and Scalar3 respectively and kS is the kinetic rate coefficient. For this case, there is no diffusion of all scalars. Since, it models chemical reactions, the kinetic rate coefficient is set to 1. This can be changed to different values.

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 newScalarTransportFoam2S, an incompressible transient solver is used for the scalar. The initial and boundary conditions are provided in the 0 directory which contains U, Scalar1, Scalar2 and Scalar3 dictionary.

```
scalar
├── 0
    ├── U
    ├── scalar1
    └── scalar2
```

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	codedFixedValue
Outlet	zeroGradient
Walls	noslip

Table 1-2 Boundary Conditions for p

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

Since, the velocity at the inlet is parabolic and the cylinder is 3D, velocity at various locations (x, y) is calculated using a code which can be accessed in U file of 0 folder.

```
inlet
{
    type            codedFixedValue;
    value           uniform (0 0 0);

    name parabolicVelocity;
    code
    #{
            const vectorField& Cf = patch().Cf();

            vectorField& field = *this;
            const scalar R = 0.005;
            const scalar c = 0;
            const scalar Umax = 0.2;

            forAll(Cf, faceI)
            {
                    const scalar x = Cf[faceI][0];
                    const scalar y = Cf[faceI][1];

                    field[faceI] = vector(0,0,Umax*(1-((pow((y-c)/R,2))+(pow((x-c)/R,2)))));
            }
    #};
}
```

The velocity at the inlet is computed using the coded boundary condition.

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 Scalar (Scalar1, Scalar2 & Scalar3)

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 scalar. Firstly, the scalar1 and scalar2 are injected at the upper half and lower half of the inlet cross section respectively with concentration of 1. Then, Scalar1 and Scalar2, using their non-zero values in a given mesh, would react to produce Scalar3.

1.4.3.2.2 Boundary conditions

The flow field is used from the flow simulation and the boundary conditions for the scalar1, scalar2 and scalar3 are tabulated below:

Table 1-3 Boundary Conditions for Scalar1

Patch	Condition
Inlet	codedFixedValue
Outlet	zeroGradient
Walls	zeroGradient

codedFixedValue for Scalar1:

```
inlet
{
    type            codedFixedValue;
    value           uniform 1;

    name halfBore;
    code
    #{
            const vectorField& Cf = patch().Cf();

            scalarField& field = *this;

            forAll(Cf, faceI)
            {
                    const scalar r = Cf[faceI][1];
                    if (r>=0)
                    {
                    field[faceI] = 1;
                    }

                    else
                    {
                    field[faceI] = 0;
                    }
            }
    #};

}
```

Here, r takes the y coordinates of the inlet face and if r is greater than zero i.e. the upper half, it assigns the value of scalar field at inlet to 1, otherwise 0.

codedFixedValue for Scalar2:

```
inlet
{
    type            codedFixedValue;
    value           uniform 1;

    name halfBore1;
    code
    #{
            const vectorField& Cf = patch().Cf();

            scalarField& field = *this;

            forAll(Cf, faceI)
            {
                    const scalar r = Cf[faceI][1];
                    if (r>=0)
                    {
                    field[faceI] = 0;
                    }

                    else
                    {
                    field[faceI] = 1;
                    }
            }
    #};

}
```

Here, r takes the y coordinates of the inlet face and if r is less than zero i.e. the lower half, it assigns the value of scalar field at inlet to 1, otherwise 0.

Boundary condition for Scalar2:

Table 1-4 Boundary Conditions for Scalar2

Patch	Condition
Inlet	codedFixedValue
Outlet	zeroGradient
Walls	zeroGradient

The boundary condition at the inlet is set to codedFixedValue for both scalar1 and scalar2 which after reaction converts to scalar3 modeled in newScalarTransportFoam2S solver with necessary formulations.

Boundary condition for Scalar3:

Scalar3 is set to zero at the inlet and zeroGradient at the outlet and walls.

```
FoamFile
{
    format      ascii;
    class       volScalarField;
    object      scalar3;
}

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    outlet
    {
        type            zeroGradient;
    }

    wall
    {
        type            zeroGradient;
    }
}
```

1.4.3.2.3 Source term

The scalar S1 and S2 are consumed during reaction which is modeled by newScalarTransportFoam2S solver through source term.

```
while (simple.correctNonOrthogonal())
{
    fvScalarMatrix scalar1Eqn
    (
        fvm::ddt(scalar1)
      + fvm::div(phi, scalar1)
      - fvm::laplacian(DS1, scalar1)
     ==
        -kS*scalar1*scalar2
    );

    fvScalarMatrix scalar2Eqn
    (
        fvm::ddt(scalar2)
      + fvm::div(phi, scalar2)
      - fvm::laplacian(DS2, scalar2)
     ==
        -kS*scalar1*scalar2
    );

    fvScalarMatrix scalar3Eqn
    (
        fvm::ddt(scalar3)
      + fvm::div(phi, scalar3)
      - fvm::laplacian(DS3, scalar3)
     ==
        kS*scalar1*scalar2
    );
```

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 2HalfBore_0D_Source /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 scalar1 and scalar2 are injected from the upper and lower cross section respectively of the inlet each with the concentration 1. Since there is no diffusivity of both the scalars, both the scalars will remain in their respective halves and will not mix with each other in the laminar flow. Hence there is no chemical reaction in the bulk. If at all there is some reaction, it will occur at the interface between the two halves first. This may happen if the interface is passing through the mesh thus causing the presence of both the scalars in it. Or it may happen due to numerical diffusion.

The scalar concentration along the tube is shown below at steady state in Fig 5. The scalar1 and scalar2 react to form scalar3 as the flow proceeds.

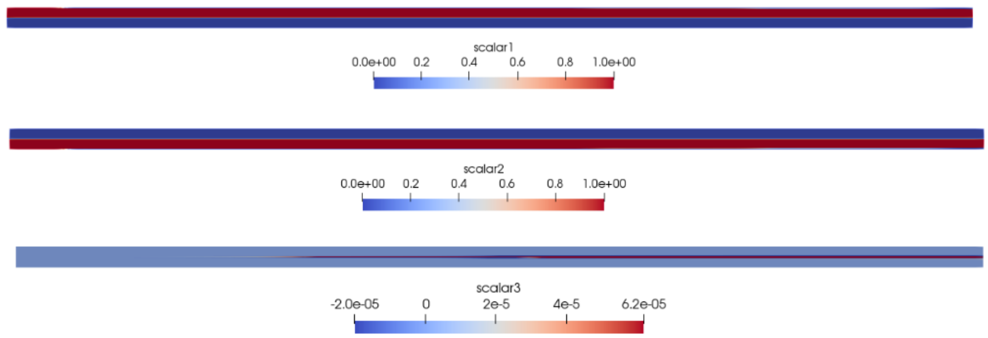

Fig 5 Scalar Concentration along the tube

The scalar concentration at the cross section of the tube at outlet is shown in **Error! Reference source not found.**. Since there is no diffusion, the flow does not cross the center to the lower half. Scalar3 is formed only at the interface of scalar1 and scalar2 where reaction takes place.

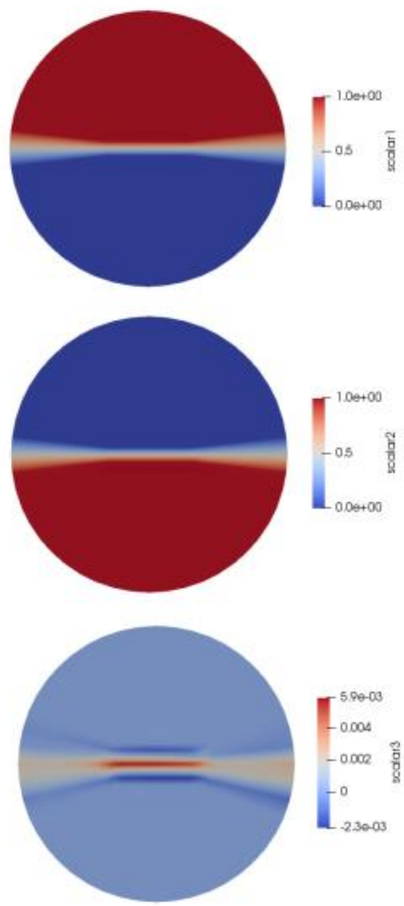

Fig 6 Scalar concentration at outlet after steady State

At the cross section of outlet, a vertical line along the cross section in y direction is drawn and plotted, the scalar concentration can be seen in Fig 7.

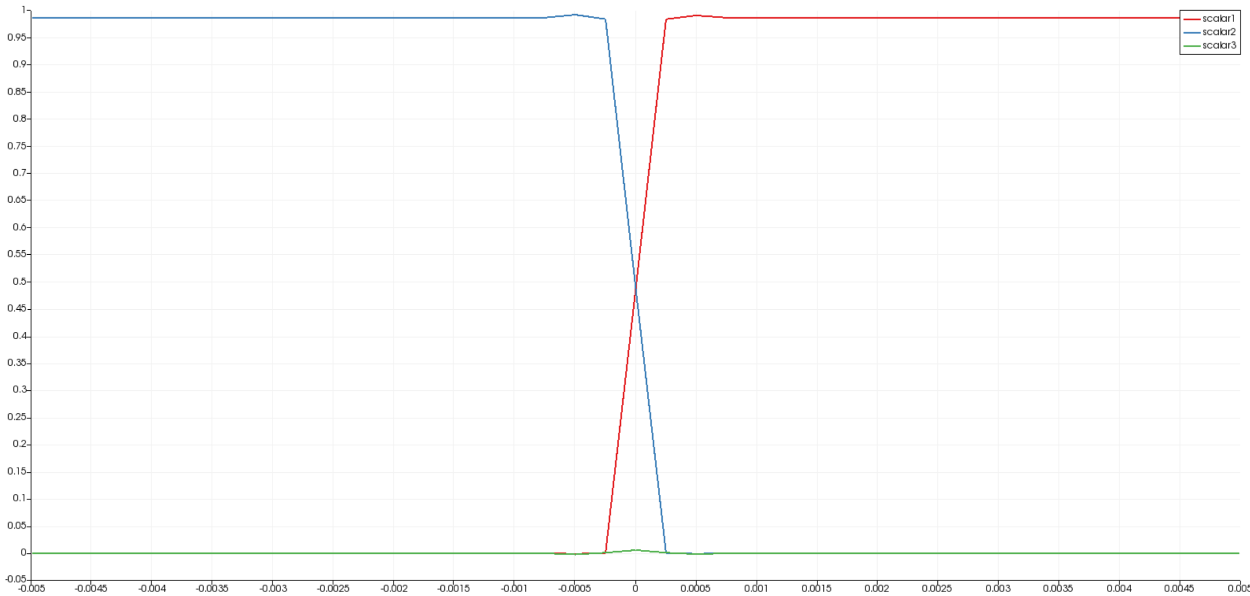

Fig 7 Scalar concentration along a vertical line at the cross section of Outlet after steady state

After the flow reaches steady state, the area averaged concentration of scalar3 is calculated at various sections of the tube (0.125 m,0.25 m,0.375m) and cup average concentration is calculated at the outlet (0.5m).

The area average concentration can be calculated as:

$$S_{avg} = \frac{\int S.dA}{A_{cross-section}}$$

The cup average concentration can be calculated as:

$$S_{avg} = \frac{\int u.S.dA}{U_{avg} * A_{outlet}}$$

Where u is the velocity and S is the scalar concentration at area dA.

The scalar3 concentration is plotted along the tube at various cross sections in Fig 8. We can see that there is although very small but still some formation of the scalar3 taking place. This is be due to the numerical diffusion which should be further minimized or eliminated.

The user can check the simulations with different values of kinetic rate coefficient and still get the same results. If the result changes, it means there is some issue. Identify the issue and rectify it.

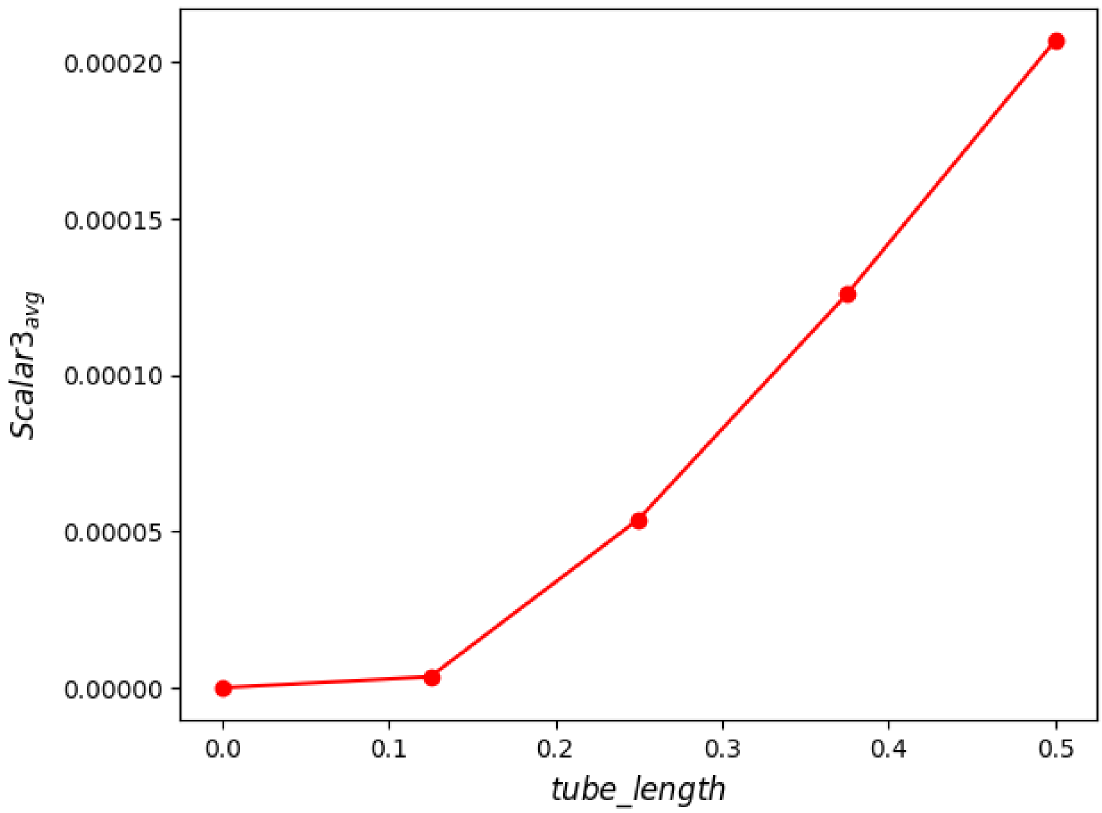

Fig 8 Scalar3 Concentration along various sections of the tube