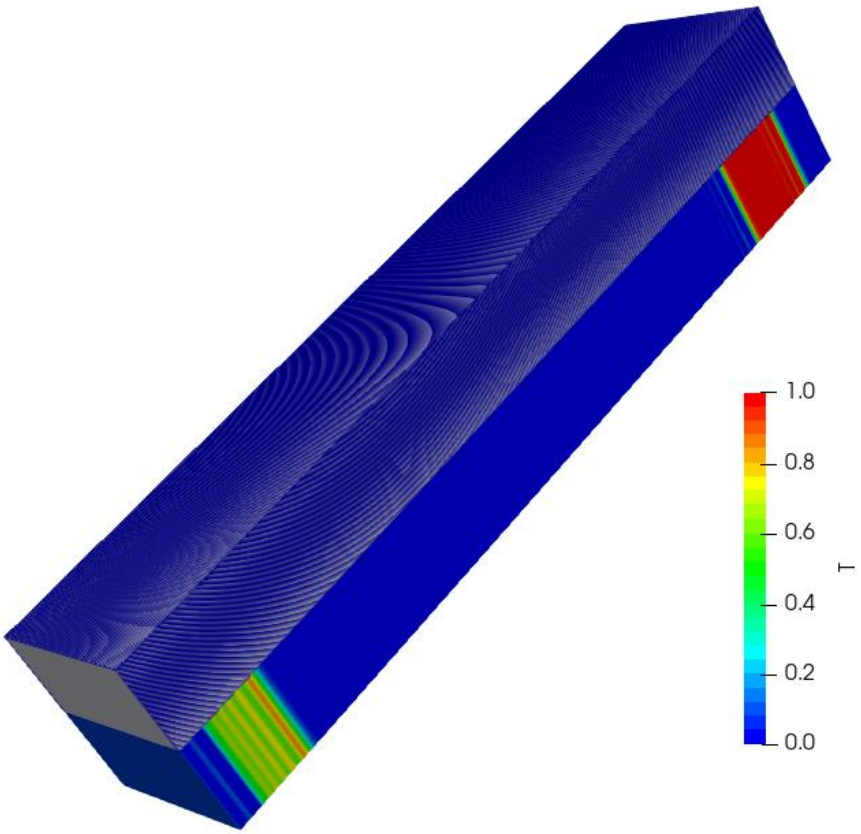
Tutorial Four

Discretization – Part 1



Bahram Haddadi

ENGINEERING FLUID DYNAMICS

7th edition, March 2025

Contributors:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek
- Clemens Gößnitzer
- Sylvia Zibuschka
- Yitong Chen
- Vikram Natarajan
- Jozsef Nagy

TU WIEN ICEBE IMAGINEERING NATURE

Technische Universität Wien

Institute of Chemical, Environmental

& Bioscience Engineering

Available from: www.fluiddynamics.at

Background

1. Discretizing general transport equation terms

Understanding the process of discretization is essential in Computational Fluid Dynamics (CFD). Discretization involves breaking down continuous differential equations into algebraic equations that can be solved numerically. In OpenFOAM®, various discretization schemes are used to approximate different terms in the transport equation, which describes how physical quantities (e.g., velocity, temperature, or concentration) change over space and time. Below is a detailed explanation of how each term in the transport equation is discretized.

1.1. Time derivative

The time derivative term represents how a variable evolves over time. This term is crucial for transient simulations, where the solution changes over time. Discretization of the time derivative such as $\frac{\partial \rho\varphi}{\partial t}$ of the transport equation is performed by integrating it over the control volume of a grid cell. Here, the Euler implicit time differencing scheme is explained. It is unconditionally stable, but only first order accurate in time. Assuming linear variation of φ within a time step gives:

$$\int_V \frac{\partial \rho\varphi}{\partial t} dV \approx \frac{\rho_P^n \varphi_P^n - \rho_P^0 \varphi_P^0}{\Delta t} V_P$$

Where $\varphi^n \equiv \varphi\ (t + \Delta t)$ stands for the new value at the time step we are solving for and $\varphi^0 \equiv \varphi(t)$ denotes old values from the previous time step.

Higher-order schemes, such as Crank-Nicolson, offer improved accuracy but may introduce oscillations if not applied carefully.

1.2. Convection term

The convection term describes the transport of a property due to the motion of the fluid. Convection plays a significant role in CFD since it governs how momentum, heat, and mass are transported within the fluid domain.

Discretization of convection terms is performed by integrating over a control volume and transforming the volume integral into a surface integral using the Gauss's theorem as follows:

$$\int_A \boldsymbol{n} \cdot (\rho\varphi\boldsymbol{u})\, dA \approx \sum_f \boldsymbol{n} \cdot (A\rho\boldsymbol{u})_f \varphi_f = \sum_f F\varphi_f$$

Where F is the mass flux through the face f defined as $F = \boldsymbol{n} \cdot (A\rho\boldsymbol{u})_f$. The value φ_f on face f can be evaluated in a variety of ways, which will be covered later in section 2. The subscript f refers to a given face.

Choosing the right numerical scheme is essential for balancing accuracy and stability in convection-dominated problems.

1.3. Diffusion term

The diffusion term represents the spread of the property due to molecular effects such as viscosity or heat conduction. The diffusion term is a second-order derivative term that requires careful discretization. Discretization of diffusion terms is done in a similar way to the convection terms. After integration over the control volume, the term is converted into a surface integral:

$$\int_A \boldsymbol{n} \cdot (\Gamma \nabla \varphi)\, dA = \sum_f \Gamma_f (\boldsymbol{n} \cdot \nabla_f \varphi) A_f$$

Note that the above approximation is only valid if Γ is a scalar. Here, $\nabla_f \varphi$ denotes the gradient at the face A denotes the surface area of the control volume and A_f denotes the area of a face for the control volume. However, it does not imply a specific discretization technique. The face normal gradient can be approximated using the scheme:

$$\boldsymbol{n} \cdot \nabla_f \varphi = \frac{\varphi_N - \varphi_P}{|\boldsymbol{d}|}$$

This approximation is second order accurate when the vector $\boldsymbol{d}$ between the center of the cell of interest P and the center of a neighboring cell N is orthogonal to the face plane, i.e. parallel to A. In the case of non-orthogonal meshes, a correction term could be introduced which is evaluated by interpolating cell centered gradients obtained from Gauss integration.

1.4. Source term

Source terms, such as $S_\varphi$of the transport equation, can be a general function of φ. Before discretization, the term is linearized:

$$S_\varphi = \varphi S_I + S_E$$

where S_E and S_I may depend on φ. The term is then integrated over a control volume as follows:

$$\int_V S_\varphi dV = S_I V_P \varphi_P + S_E V_P$$

There is some freedom on exactly how a particular source term is linearized. When deciding on the form of discretization (e.g. linear, upwind), its interaction with other terms in the equation and its influence on boundedness and accuracy should be examined.

2. Discretization Schemes

Discretization schemes determine how values are interpolated between cell centers and faces to compute fluxes accurately. The choice of scheme affects solution accuracy, numerical diffusion, and computational stability. Below are commonly used schemes and their respective advantages and limitations.

In general, interpolation needs a flux F through a general face f, and in some cases, one or more parameters γ. The face value φ_f can be evaluated from the

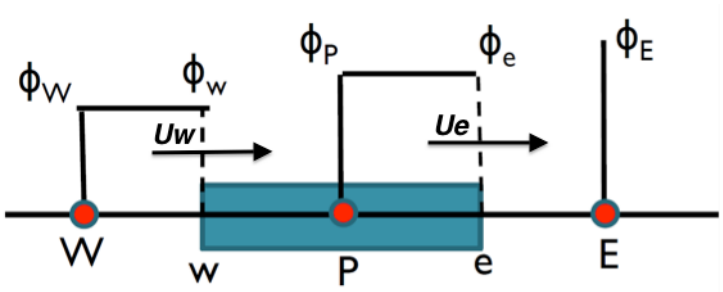
values in the neighboring cells using a variety of schemes. The flux satisfies continuity constraints, which is prerequisite to obtaining the results.

2.1. First Order Upwind Scheme

In first order upwind scheme we define φ as follows:

Note: Here we define two faces, e and w. To obtain flux through faces e and w, we need to look its neighbouring values at P/E and W/P respectively. The subscripts denote the face at which the face value φ or the flux F is located at.

$$\varphi_e = \varphi_P \qquad if, F_e > 0$$
$$\varphi_e = \varphi_E \qquad if, F_e < 0$$



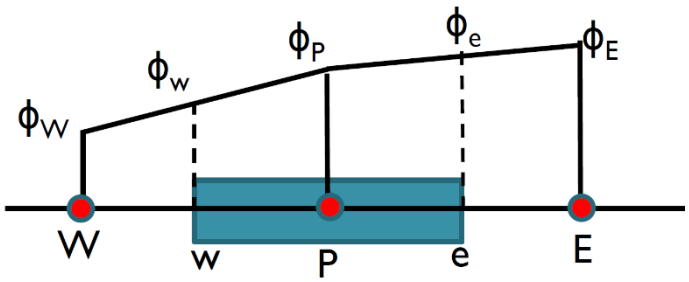
First Order Upwind Scheme

φ_w is also defined similarly (Positive direction is from W to E).

2.2. Central Differencing Scheme

Here, we use linear interpolation for computing the cell face values.

$$\varphi_e = \frac{\varphi_E + \varphi_P}{2}, \qquad \varphi_w = \frac{\varphi_P + \varphi_W}{2}$$



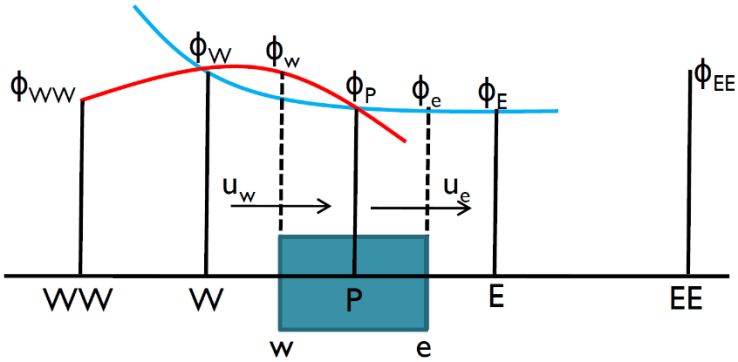
Central Differencing Scheme

2.3. QUICK

QUICK stands for Quadratic Upwind Interpolation for Convective Kinetics. In the QUICK scheme 3 point upstream-weighted quadratic interpolation are used for cell face values.

$$When\ F_e > 0, \qquad \varphi_e = \frac{6}{8}\varphi_P + \frac{3}{8}\varphi_E - \frac{1}{8}\varphi_W$$

$When\ F_w > 0,\qquad \varphi_w = \frac{6}{8}\varphi_W + \frac{3}{8}\varphi_P - \frac{1}{8}\varphi_{WW}$



QUICK scheme

Similar expressions can be obtained for $F_e < 0$ and $F_w < 0$.

Now that you know a bit more about discretization schemes, we can move on to the tutorial. In this tutorial, the scalarTransportFoam solver is used. More explanation of this solver can be found below.

3. functions solver

Among *foamRun* solver modules *functions* solver, which is specifically designed to execute function objects as defined in the *system/controlDict* or *system/functions* files. Function objects are utilities within OpenFOAM that facilitate workflow configurations and enhance simulations by generating additional data during runtime or post-processing. By utilizing the *functions* solver module with *foamRun*, users can automate the execution of these function objects, streamlining processes such as data logging, field calculations, and custom analyses without the need to run a full simulation. This approach optimizes computational resources and simplifies the integration of auxiliary calculations into the simulation workflow.

One of these functions is scalarTransport which resolves a transport equation for a passive scalar. The velocity field and boundary condition need to be provided by the user. It works by setting the source term in the transport equation to zero (see equation below), and then solving the equation.

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla\cdot(\rho\varphi\boldsymbol{u}) - \nabla\cdot(\Gamma\nabla\varphi) = 0$$

functions Solver – shockTube

Tutorial outline

Use the functions solver, simulate 5 s of flow inside a shock tube, with 1D mesh of 1000 cells (10 m long geometry from -5 m to 5 m). Patch with a scalar of 1 from -0.5 to 0.5. Simulate following cases:

- Set U to uniform (0 0 0). Vary diffusion coefficient (low, medium and high value).
- Set the diffusion coefficient to zero and also U to (1 0 0) and run the simulation in the case of pure advection using following discretization schemes:
  - upwind
  - linear
  - linearUpwind
  - QUICK
  - cubic

Objectives

- Understanding different discretization schemes.

Data processing

Import your simulation into ParaView, and plot temperature along tube length.

1. Pre-processing

1.1. Compile tutorial

Create a folder in your working directory:

```
>mkdir shockTube
```

Copy the following case to the created directory:

```
$FOAM_TUTORIALS/fluid/shockTube
```

In the 0 directory, create a copy of T.orig and U.orig and rename them to T and U respectively. In the constant directory delete *physicalProperties* file, and in the system directory delete all the files except for *blockMeshDict* and *setFieldsDict* files.

From the following case:

```
$FOAM_TUTORIALS/incompressibleFluid/pitzDailyScalarTransp
ort
```

Copy *physicalProperties* file to the constant folder in the newly created case constant folder. Copy *controlDict*, *fvSchemes, fvSolution* and *functions* files from the above case system directory to the created case system directory.

1.2. system directory

Edit the *setFieldsDict*, to patch the T field to 1.0 between -0.5 m and 0.5 m and to set the U to (0 0 0) for the whole domain. For setting U in the whole domain to (1 0 0), just change `(0 0 0)` to `(1 0 0)`:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
defaultFieldValues
(
        volVectorFieldValue U ( 0 0 0 )
        volScalarFieldValue T 0.0
);
regions
(
        boxToCell
        {
                box ( -0.5 -1 -1 ) ( 0.5 1 1 );

                fieldValues
                (
                        volScalarFieldValue T 1.0
                );
        }
);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
```

In the *controlDict*, update the `endTime` to 5 for 5s of simulation. As it was mentioned before, the discretization scheme for each operator of the governing equations can be set in *fvSchemes*.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,T)      Gauss linearUpwind grad(T);
}

laplacianSchemes
{
    default         none;
    laplacian(DT,T) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * *//
```

For each type of operation a default scheme can be set (e.g. for `divSchemes` is set to `none`, it means no default scheme is set). Also a special type of discretization for each element can be assigned (e.g. `div(phi,T)` it is set to `linearUpwind`). For each element, where a discretization method has not been set, the default method will be applied. If the default setting is `none`, no scheme is set for that element and the simulation will crash.

Note: *In fvSchemes, the schemes for the time term of the general transport equation are set in `ddtSchemes` sub-dictionary. `divSchemes` are responsible for the advection term schemes and `laplacianSchemes` set the diffusion term schemes.*

Note: *`divSchemes` should be applied like this: Gauss + scheme. The `Gauss` keyword specifies the standard finite volume discretization of Gaussian integration which requires the interpolation of values from cell centers to face centers. Therefore, the `Gauss` entry must be followed by the choice of interpolation scheme (www.openfoam.org).*

In the *fvSolution* file add pressure reference cell number and value to the PIMPLE sub-dictionary, it should look like the following:

```
PIMPLE
{
    nNonOrthogonalCorrectors 0;
    pRefCell 0;
    pRefValue 0;
}
```

Note: pRefCell and pRefValue are dummy values that solver can start the calculations, since there is no pressure field available.

In the *functions* file, just keep the line for activating the scalar transport function. In the functions file, different functions can be called, in this case the scalar transport function is called with using "T" as the property (scalar) to be solved, it uses a constant diffusivity model and set the value of it by setting D (in this case it is 0.01).

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#includeFunc scalarTransport(T, diffusivity=constant, D = 0.01)
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Note: By setting the diffusion coefficient "D" to zero, the case will be switched to a pure advection simulation with no diffusion.

For part two:

- Set the diffusivity to 0, by setting the `D` in the *functions* file
- Set the velocity field to (1 0 0), either by using setFields utility or simply in the *0/U* file change the `internalField` to `(1 0 0)`
- Set different schemes in the *fvSchemes* file, for the `div(phi, T)`

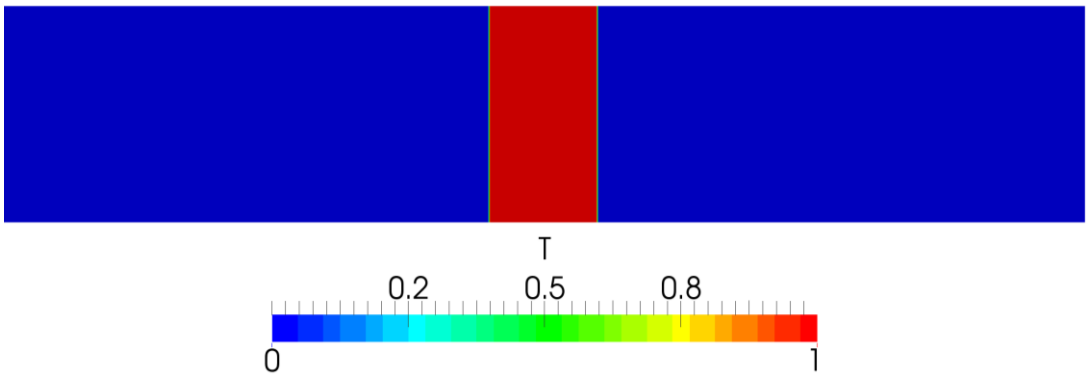
2. Running simulation

```
>blockMesh

>setFields

>foamRun -solver functions
```

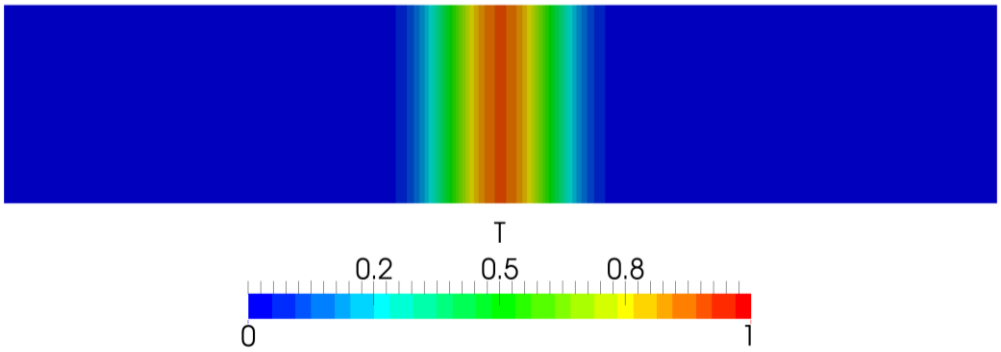
3. Post-processing

The simulation results are as follows.

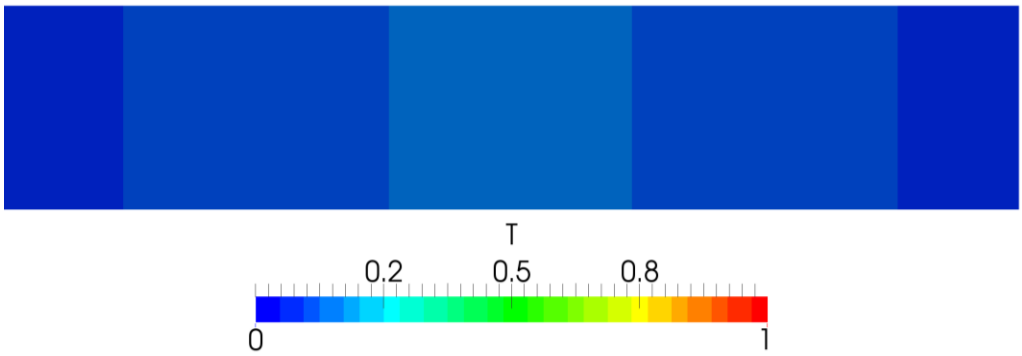
A. Case with zero velocity (pure diffusion):



Pure diffusion with low diffusivity (0.00001) at t = 5 s

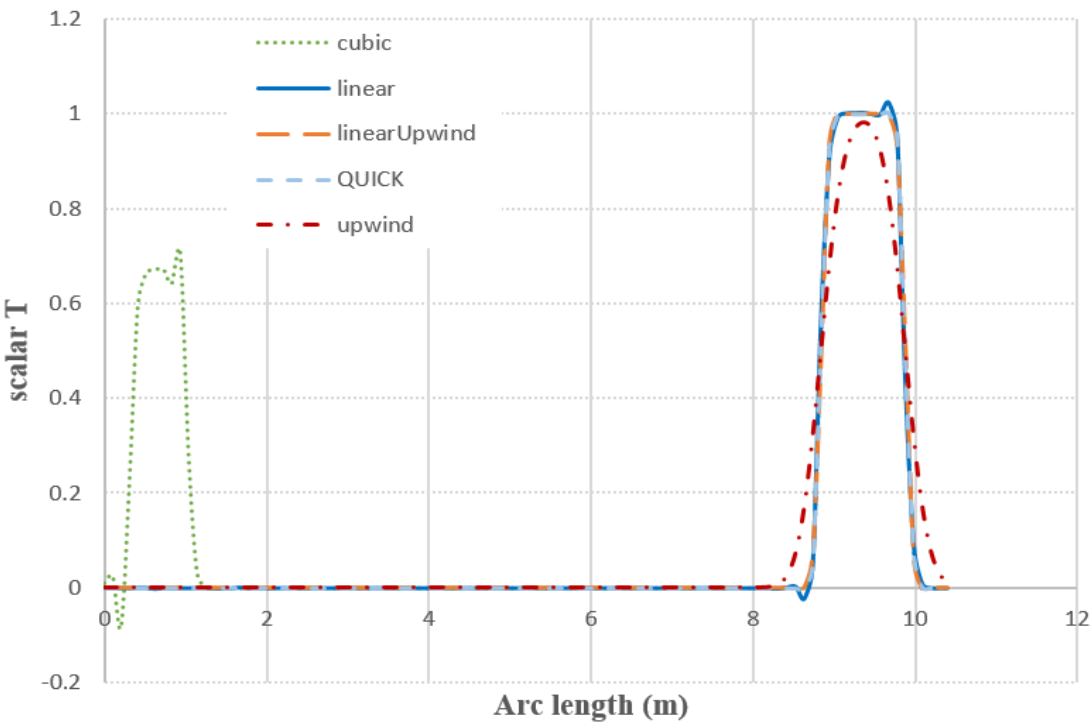


Pure diffusion with medium diffusivity (0.01) at t = 5 s



Pure diffusion with high diffusivity (1) at t = 5 s

B. Case with pure advection (diffusion coefficient = 0):



Scalar T along tube at t = 4 s

The cubic scheme predicted an unexpected rise in temperature between around 0 to 1 m, which differs hugely from the other schemes. This can be explained by looking at the numerical behavior of the cubic scheme. It is operated in fourth order accuracy with unbounded solutions, which caused another false root solution to be found. Therefore, higher order accuracy does not always generate better results!