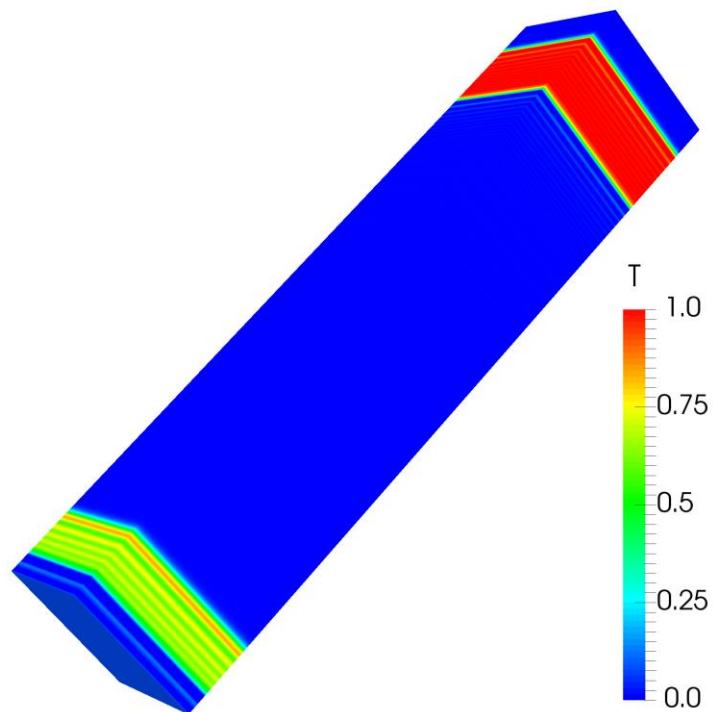


Tutorial Four


Discretization – Part 1



4th edition, Jan. 2018



This offering is not approved or endorsed by ESI® Group, ESI-OpenCFD® or the OpenFOAM® Foundation, the producer of the OpenFOAM® software and owner of the OpenFOAM® trademark.

 Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Editorial board:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek

Compatibility:

- OpenFOAM® 5.0
- OpenFOAM® v1712

Contributors:

- Bahram Haddadi
- Clemens Gößnitzer
- Jozsef Nagy
- Vikram Natarajan
- Sylvia Zibuschka
- Yitong Chen

Cover picture from:

- Bahram Haddadi

 Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Attribution–NonCommercial–ShareAlike 3.0 Unported (CC BY–NC–SA 3.0)

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:

- Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Noncommercial — You may not use this work for commercial purposes.
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.
- Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- Other Rights — In no way are any of the following rights affected by the license:
- Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
- Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

For more tutorials visit: www.cfd.at

Background

1. Discretizing general transport equation terms

Tutorial Four aims to help the users understand the different discretization schemes in OpenFOAM®. But before that, it is important to understand the exact mathematical procedures involved in discretization. Below is a detailed explanation of how each term of the transport equation is discretized.

1.1. Time derivative

Discretization of the time derivative such as $\frac{\partial \rho \phi}{\partial t}$ 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:

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

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

1.2. Convection term

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:

$$\int_A \mathbf{n} \cdot (\rho \phi \mathbf{u}) dA \approx \sum_f \mathbf{n} \cdot (A \rho \mathbf{u})_f \phi_f = \sum_f F \phi_f$$

Where F is the mass flux through the face f defined as $F = \mathbf{n} \cdot (A \rho \mathbf{u})_f$. 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.

1.3. Diffusion term

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:

$$\int_A \mathbf{n} \cdot (\Gamma \nabla \phi) dA = \sum_f \Gamma_f (\mathbf{n} \cdot \nabla_f \phi) A_f$$

Note that the above approximation is only valid if Γ is a scalar. Here, $\nabla_f \phi$ 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. It does not, however, imply a specific

discretization technique. The face normal gradient can be approximated using the scheme:

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

This approximation is second order accurate when the vector \mathbf{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 \mathbf{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_φ 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

Since the results of CFD simulations are typically stored at the cell centers, it is important to interpolate the results from cell centers to the face centers, to obtain the fluxes for the surface integrals in the transport equation. For each term of the transport equation, there is a variety of discretization/interpolation schemes available.

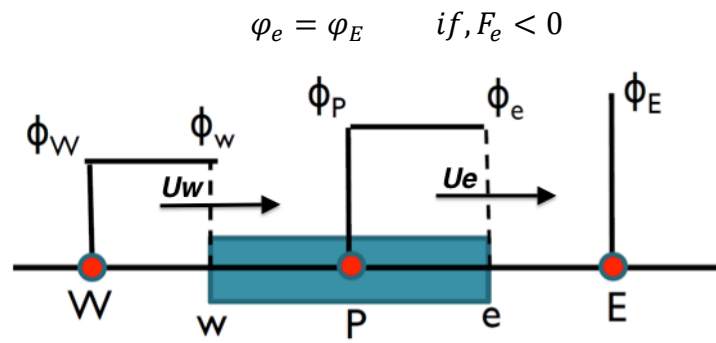
In general, interpolation needs a flux F through a general face f , and in some cases, one or more parameters γ . The face value φ_f can be evaluated from the values in the neighboring cells using a variety of schemes. The flux satisfies continuity constraints, which is prerequisite to obtaining the 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.

$$\varphi_e = \varphi_P \quad \text{if } F_e > 0$$



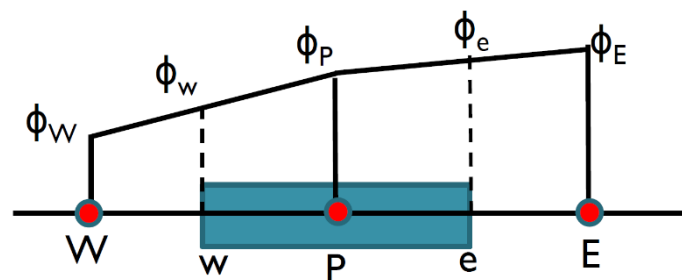
First Order Upwind Scheme

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.

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



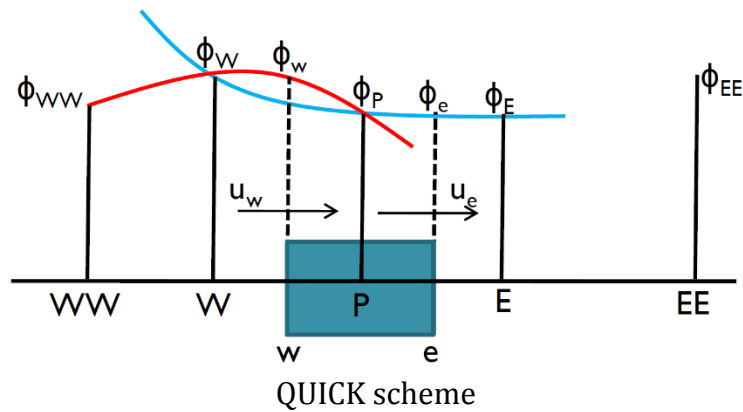
Central Differencing Scheme

2.3. QUICK

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

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

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



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.

4. *scalarTransportFoam* solver

scalarTransportFoam is a basic solver 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.

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

scalarTransportFoam – shockTube

Simulation

Use the scalarTransportFoam 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/compressible/sonicFoam/laminar/shockTube
```

In the 0 and 0.orig directories, delete *p* files. In the constant directory delete the *thermophysicalProperties* and *turbulenceProperties* files, and in the system directory delete *controlDict*, *fvSchemes* and *fvSolution* files.

From the following case:

```
$FOAM_TUTORIALS/basic/scalarTransportFoam/pitzDaily
```

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

1.2. constant directory

The diffusion coefficient can be set in the *transportProperties* file. For a low value try 0.00001, for a medium value use 0.01 and for a high value use 1:

```
DT          DT [ 0 2 -1 0 0 0 0] 0.01;
```

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

OpenFOAM® v1712: Just DT and its value are listed – no dimensions!

1.3. system directory

Edit the *setFieldsDict*, to patch the T field from -0.5 m to 0.5 m and also 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
        );
    }
);
```



```

    }
);
// * * * * *
* * * * *//

```

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 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).

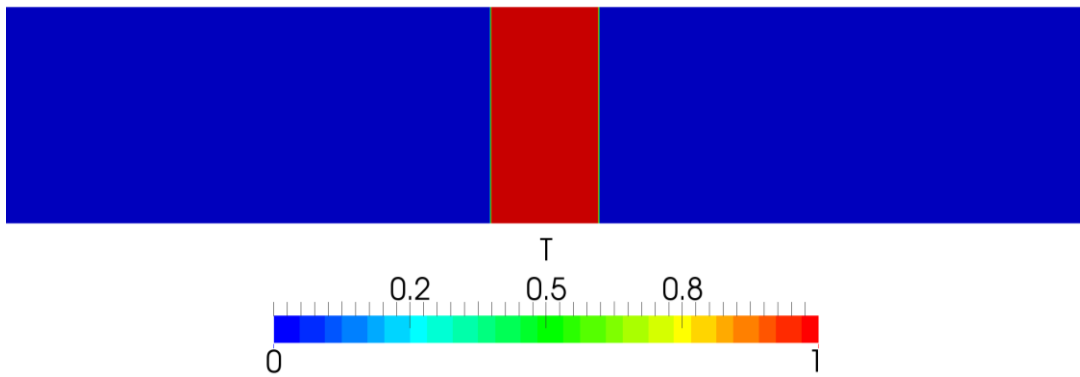
2. Running simulation

```
>blockMesh
>setFields
>scalarTransportFoam
```

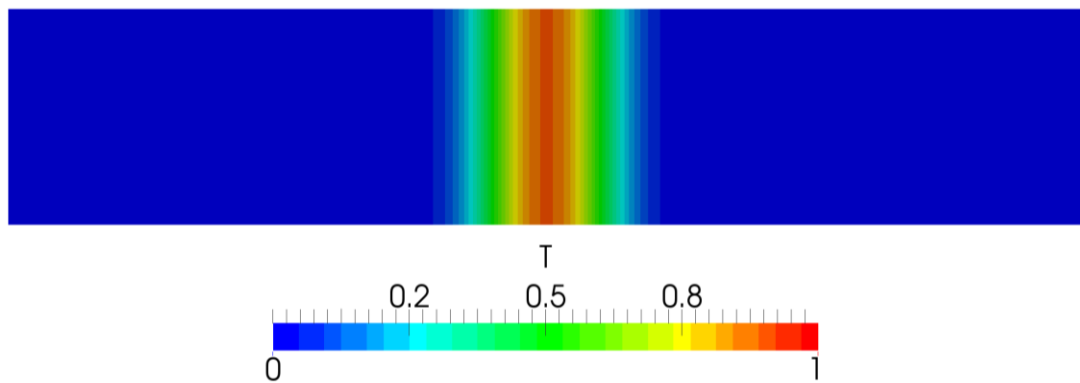
3. Post-processing

The simulation results are as follows.

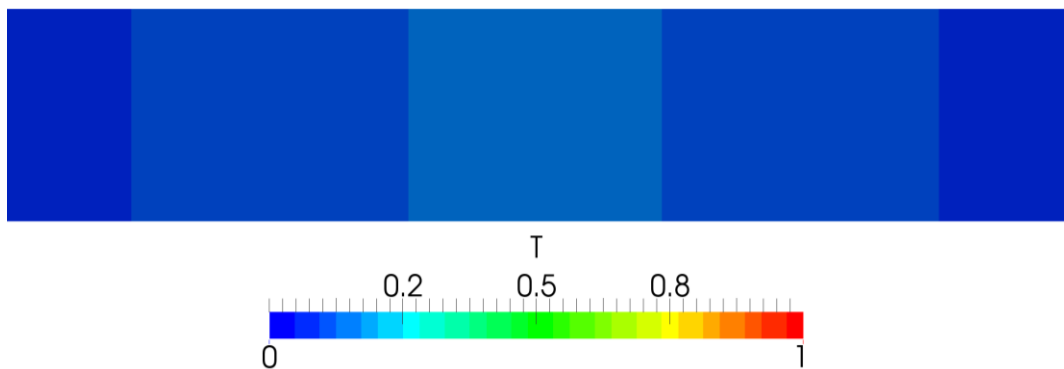
A) Case with zero velocity (pure diffusion):



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

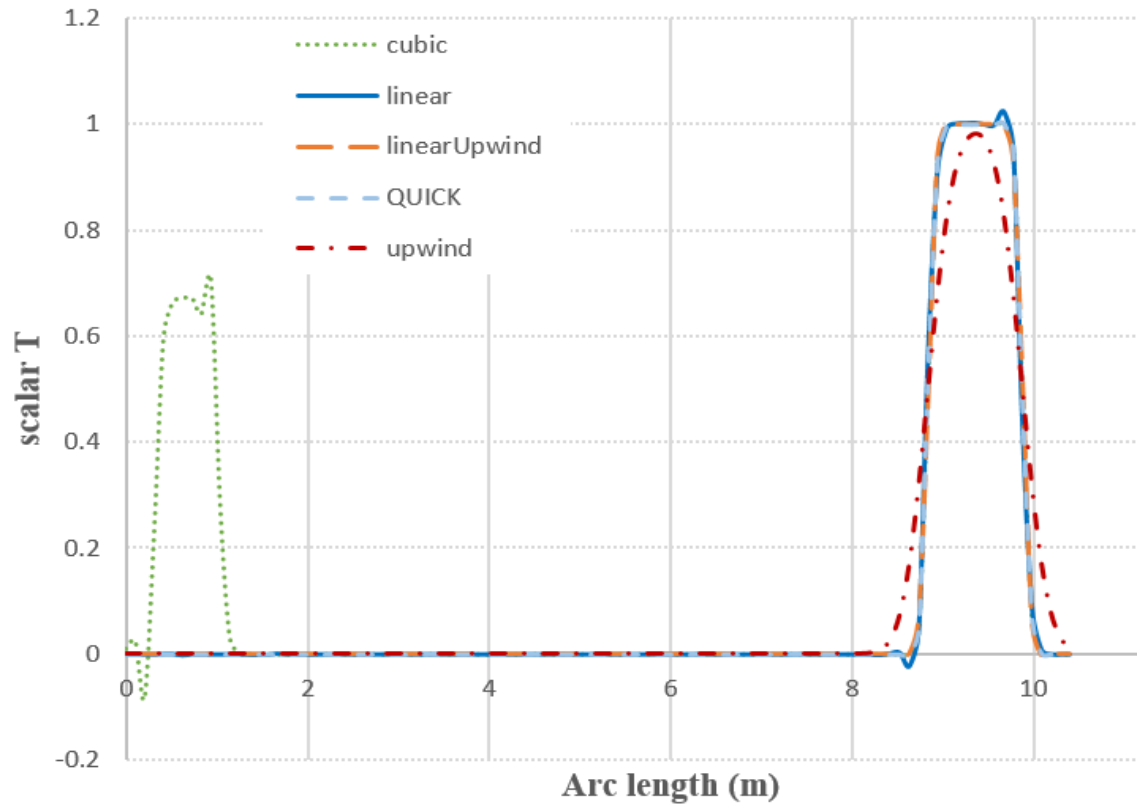


Pure diffusion with medium diffusivity (0.01) at $t = 5$ s



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. So, higher order accuracy does not always generate better results!