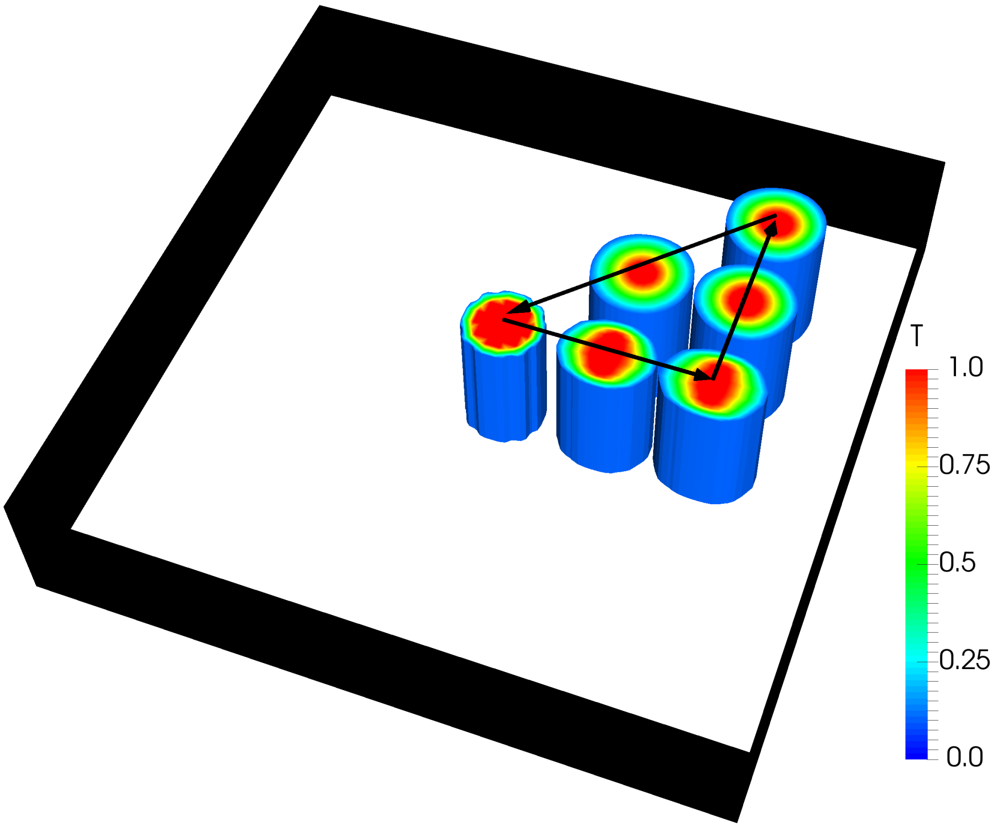
Tutorial Five

Discretization – Part 2



5th edition, Sep. 2019

Compatibility:
- OpenFOAM® 7
- OpenFOAM® v1906

Cover picture from:
- Bahram Haddadi

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

ISBN 978-3-903337-00-8

Publisher: chemical-engineering.at

For more tutorials visit: www.cfd.at

Background

1. Properties of discretization schemes

Let's explore some fundamental properties of discretization schemes. These properties are required for our numerical results to be physically realistic. An understanding of these properties will help the users to choose the appropriate discretization schemes for their model.

1.1. Conservativeness

Integration of the convection–diffusion equation over a finite number of control volumes yields a set of discretized conservation equations involving fluxes of the transported property φ through control volume faces. To ensure conservation of φ for the whole solution domain the flux of φ leaving a control volume across a certain face must be equal to the flux of φ entering the adjacent control volume through the same face. To achieve this flux through a common face must be represented in a consistent manner – by one and the same expression – in adjacent control volumes of each face.

1.2. Boundedness

Normally we use iterative numerical techniques to solve discretized equations at each nodal point. The methods start with a guessed distribution of the initial conditions of the variable φ and perform successive updates until a converged solution is obtained.

The sufficient condition for a converged solution is:

$$\frac{\sum|a_{nb}|}{|a'_P|} \begin{cases} \le 1 \; at\ all\ nodes \\ < 1 \; at\ one\ node\ at\ least \end{cases}$$

Here a'_P is the net coefficient of the central node P (i.e. $a'_P - S_P$), a_{nb} are the coefficient of the neighbouring nodes. If the condition is satisfied, the resulting matrix of coefficients is diagonally dominant. We need the net coefficients to be as large as possible, this means that S_P should be always negative. If this is the case, S_P becomes positive due to the modulus sign and adds to a_P.

1.3. Transportiveness

To understand transportiveness, one should look at a dimensionless number called the Peclet number, Pe. It measures the relative strengths of convection, N_{conv} and diffusion, N_{diff}.

$$Pe = \frac{N_{conv}}{N_{diff}} = \frac{LU}{D}$$

Note: L is a characteristic length scale, U is the velocity magnitude, D is a characteristic diffusion coefficient.

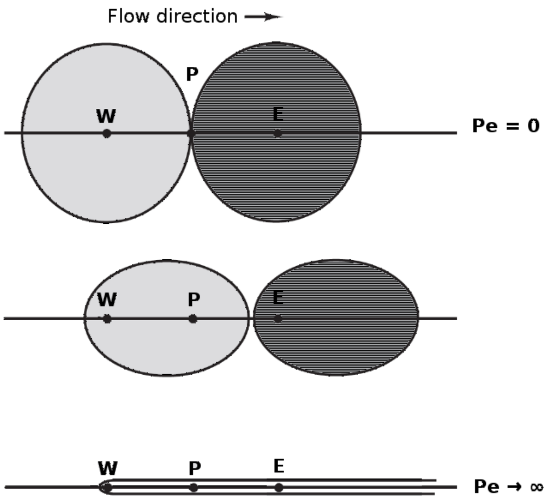
The primary goal is to ensure that the transportiveness is borne out of the discretization scheme.

Let's consider the effect at a point P due to two constant sources of φ at nearby points *W* and *E* on either side, in three cases.

1) When Pe = 0 (pure diffusion), the countours of φ are circles, as φ is spread out evenly in all directions

2) As Pe increases, the contours become elliptical, as the values of φ are influenced by convection

3) When Pe→∞, the countours become straight lines, since φ are stretched out completely and affected only by upstream conditions



Transportiveness property

2. Assessing the general discretization schemes

It is useful to compare the different types of general discretization schemes covered in Tutorial Four based on their conservativeness, boundedness and transportiveness properties.

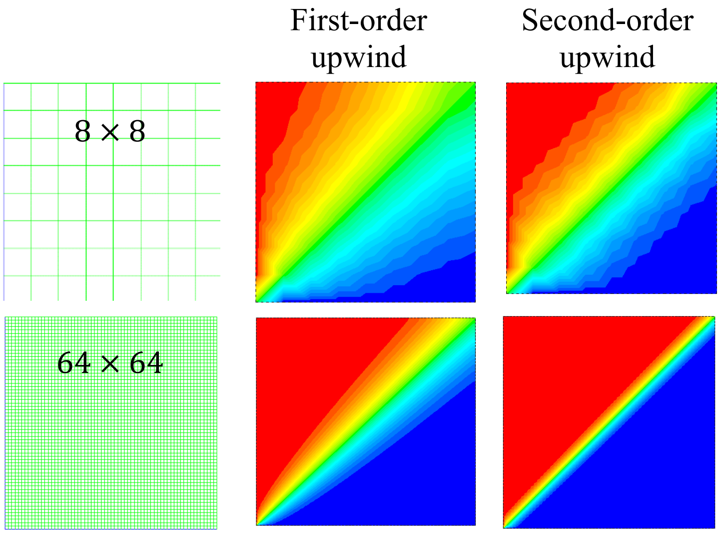
Different discretizing schemes assessment

Scheme	Conser -vative	Bounded	Accuracy	Trans-portive	Remarks
Upwind	Yes	Unconditionally bounded	First order	Yes	Include false diffusion if the velocity vector is not parallel to one of the coordinate directions
Central Differencing	Yes	Conditionally bounded*	Second order	No	Unrealistic solutions at large Pe number
QUICK	Yes	Unconditionally bounded	Third order	Yes	Less computationally stable. Can give small undershoots and overshoots

****Pe should be less than 2.***

3. Numerical (false) diffusion

Numerical diffusion is a multidimensional phenomenon and it occurs when the flow is not perpendicular to the grid lines. It is a numerically introduced diffusion and arises in convection dominated flows, i.e. high Pe number flows.



Numerical diffusion

4. Numerical behavior of OpenFOAM® discretization schemes

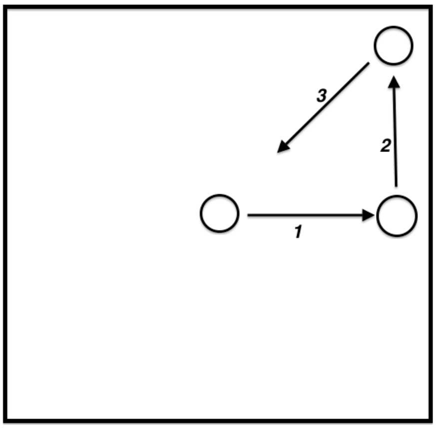
The choice of discretization scheme for this tutorial should depend critically on the numerical behaviour of the scheme. Using higher order schemes, numerical diffusion errors can be reduced, however it requires higher computational efforts.

Scheme	Numerical behaviour
upwind	First order, bounded
linear	Second order, unbounded
linearUpwind	First/second order, bounded
QUICK	Second order or higher, bounded
cubic	Fourth order, unbounded

scalarTransportFoam – circle

Simulation

Use the scalarTransportFoam solver, do simulate the movement of a circular scalar spot region (radius = 1 m) at the middle of a 100 × 100 cell mesh (10 m × 10 m), then move it to the right (3 m), to the top (3 m) and diagonally.



Schematic sketch of the problem

Objectives

- Choosing the best discretization scheme.

Data processing

Examine your simulation in ParaView.

1. Pre-processing

1.1. Compile tutorial

Create the new case in your working directory like in tutorial four.

1.2. 0 directory

To move the circle to right change the `internalField` to `(1 0 0)` in the U file for setting the velocity field towards the right. Modify U at suitable times, to obtain a velocity field which will move the circle up and also diagonally.

1.3. constant directory

In the *transportProperties,* set `DT` to zero (no diffusion!).

1.4. system directory

Modify the *blockMeshDict* for creating a 2D geometry with 100 × 100 cells mesh.

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
convertToMeters 1;

vertices
(
    (-5 -5 -0.01)
    (5 -5 -0.01)
    (5 5 -0.01)
    (-5 5 -0.01)
    (-5 -5 0.01)
    (5 -5 0.01)
    (5 5 0.01)
    (-5 5 0.01)
);
blocks
(
    hex (0 1 2 3 4 5 6 7) (100 100 1) simpleGrading (1 1 1)
);
edges
(
);
boundary
(
    sides
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (0 4 7 3)
            (3 7 6 2)
            (0 1 5 4)
        );
    }
    empty
    {
        type empty;
        faces
        (
            (5 6 7 4)
            (0 3 2 1)
        );
    }
);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
```

Choose a discretization scheme based on the results from the previous example and set it in the *fvSchemes*.

In the *setFieldsDict* patch a circle to the middle of the geometry using the following lines.

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

defaultFieldValues (volScalarFieldValue T 0 );

regions
(
       cylinderToCell
       {
              p1  ( 0 0  -1 );
              p2  ( 0 0   1 );
              radius 0.5;
              fieldValues
              (
                     volScalarFieldValue T 1
              ) ;
       }
);

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

`cylinderToCell` command is used to patch a cylinder to the region, `p1` and `p2` show the two ends of cylinder center line, in the `radius` the radius is set.

Check *controlDict*, in the first part of simulation, where the circle should move to the right set the `startFrom` to `startTime` and `startTime` to `0`. By a simple calculation it can be seen that the `endTime` should be `3` s (to move the circle from center to the right side). Similar calculations need to be done for the two other parts, except the `startTime` is set to the `endTime` of previous part, and new `endTime` should be that part "simulation time" plus `endTime` of the previous part.

2. Running Simulation

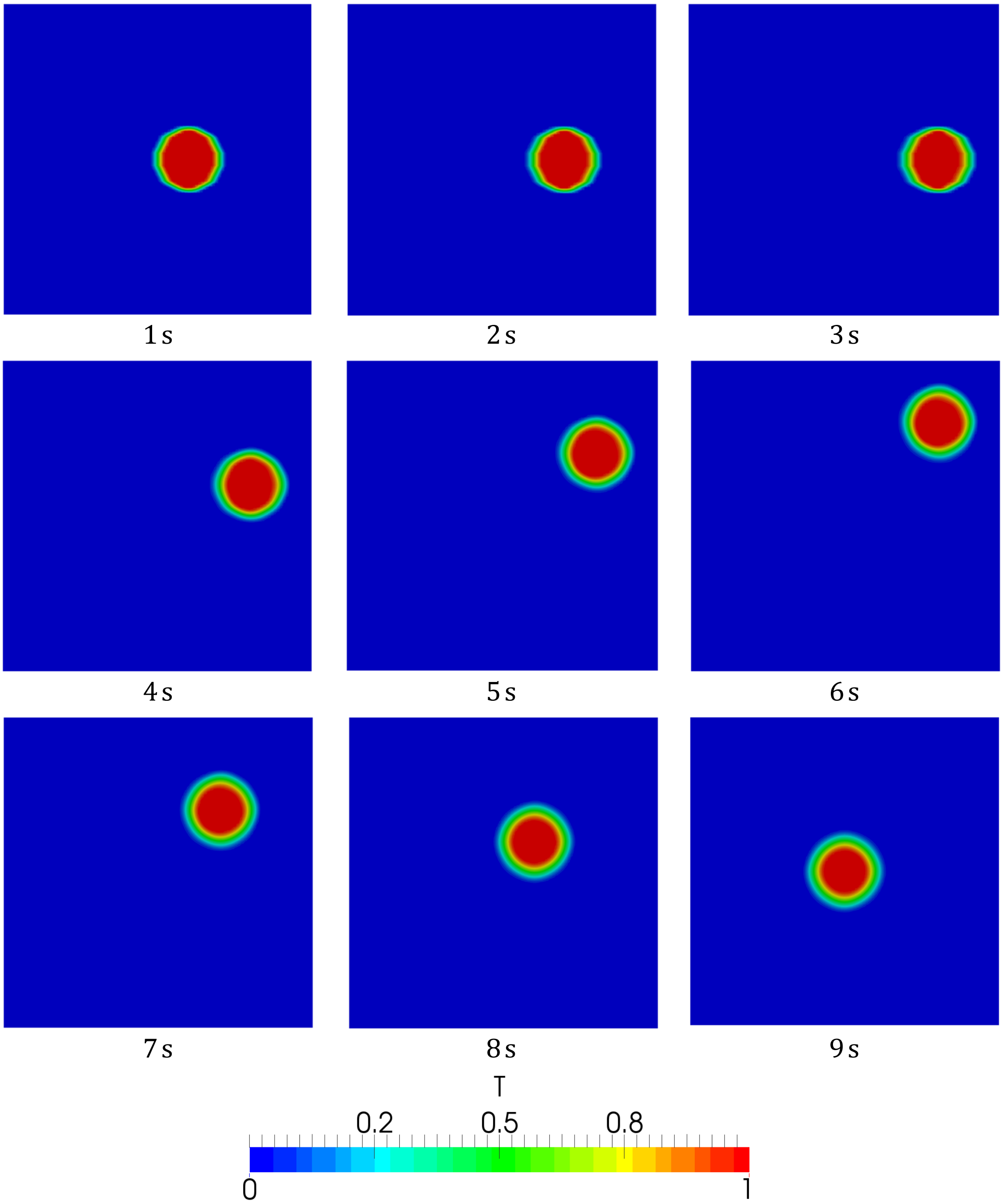
```
>blockMesh
>setFields
>scalarTransportFoam
```

For running the further parts (moving the circle to top, and then diagonally) change the velocity field in the last time step directory, i.e. change the velocity in the time step directory 3 to (0 1 0) so the circle moves up, further change the velocity in the directory 6 to (-1 -1 0) to move the circle diagonally back to the original position.

After moving the circle to the right and changing the velocity field, the simulation is resumed. It can be seen that the circle does not go up but moves to the right. This occurs due to the fact that OpenFOAM® used the previous time step fluxes (phi) to do the calculations. We can solve this problem by deleting phi file from the latest time step (of the previous part of simulation, e.g. 3). In this way, OpenFOAM® creates new fluxes based on the new velocity field that we just updated. So, easily delete phi and enjoy!

3. Post-processing

The simulation results are as follows:



Position of the circle at different time steps

ISBN 978-3-903337-00-8

9 783903 337008