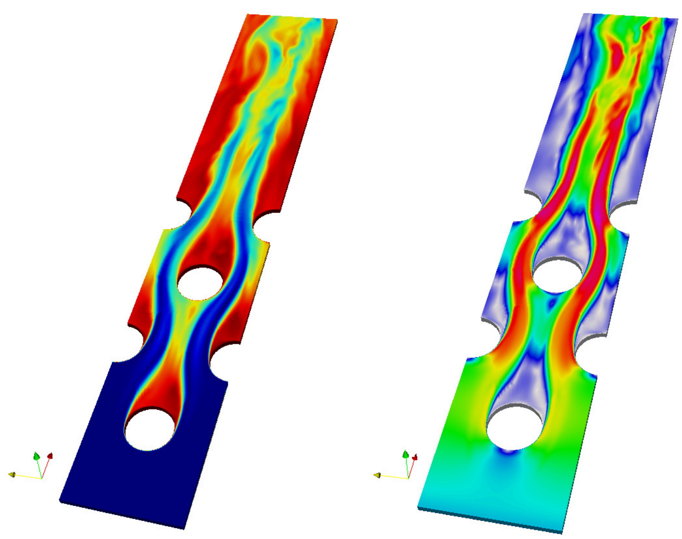
# Numerical Techniques for Process Modelling
# Part I:
# Finite Volume Method
# and
# Computational Fluid Dynamics

KEB-45250
Numerical Techniques for Process Modelling,
Prosessien numeerinen mallinnus
5 credits

Antti Mikkonen
antti.mikkonen@kapsi.fi
Tampere University of Technology

Updated on:

January 5, 2018



# Contents

# Preface

This booklet is meant for students on the course Numerical Techniques for Process Modelling, arranged on Tampere University of Technology. The course consist of three parts: computational fluid dynamics, general use of numerical methods in energy related industrial problems, and reaction modeling. This booklet considers the computational fluid dynamics part.

This booklet is a custom fitted to contain the material considered on the course. If you prefer a full text book, see the introductory level text book by Versteeg and Malalasekera, An introduction to Computational Fluid Dynamics [1]. More detailed pointers to relevant section will be given in the text as well as other outside sources.

This is the first time this booklet is used and the booklet will be written as the course progresses. Errors are inevitable. Feel free to email the author: antti.mikkonen@kapsi.fi.

## Acronyms and translations

| | | |
|---|---|---|
| CFD | Computational fluid dynamics | Virtauslaskenta |
| FEM | Finite element method | Elementtimenetelmä |
| RANS | Reynolds-averaged Navier-Stokes (equations) | |
| NS | Navier-Stokes (equations) | |
| FVM | Finite volume method | Tilavuusmenetelmä |

## Naming conventions and translations

The following mean roughly the same thing. Be vary of small differences!

| | | |
|---|---|---|
| Heat diffusion | Heat conduction | Lämmön johtuminen/diffuusio |

## Plan

| Week | | | Chap. | Time |
|---|---|---|---|---|
| 1 | Introduction | Antti/Kaj/ Niko | | |
| 2 | | | | |
| 3 | Linear differential equations | Kaj | | |
| 4 | “” | Kaj | | |
| 5 | Non-linear differential equations | Kaj | | |
| 6 | “” | Kaj | | |
| 7 | Finite Volume Method | Antti | 1 | |
| 8 | ANSYS | Medeso | | |
| 9 | Diffusion-advection | Antti | | |
| 10 | Navier-Stokes, pressure coupling, compressibility | Antti | | |
| 11 | Turbulence | Antti | | |
| 12 | Mesh | Antti | | |
| 13 | Reaction models | Niko | | |
| 14 | Reaction models + | Niko | | |

# 1 Finite Volume Method

In this section we study the general properties of Finite Volume Method (FVM) using one-dimensional heat conduction as an example. The necessary steps are very similar in more complicated cases. For the same information in more elaborated format see the introduction level text book by Versteeg and Malalasekera, An introduction to Computational Fluid Dynamics. For this section see chapter four.

The goal of FVM, and other similar numerical methods, is to divide a continuous problem into small discrete pieces. An approximate solution of the original problem is expressed with a large number of simple arithmetic equations. This results are readily collected in to a matrix and solved with a computer.

The main advantages of numerical solution over analytical solution is the ease of solving problems in complicated geometries and boundary conditions or with varying fluid properties. For such problems analytical solutions often don't exists or are very time consuming to implement. A word of warning is, however in order: it is very easy to produce false results with numerical tools! Especially with commercial software. Therefore it is important to always verify the numerical method with a similar analytical solution if such is available.

During this cource we become familiar with Finite Volume Method (FVM). Other very similar numerical methods are Finite Element Method (FEM) and Finite Difference Method (FDM). In general, numerical solution of fluid related problems are often called Computational Fluid Dynamics (CFD). FVM method is the preferred method for fluid flow problems because of it's simplicity and conservativeness (mass balance etc.).

# 2 Heat Conduction (diffusion)

*The material discussed in this section may be found in a more elaborated format from Versteeg and Malalasekara 2007, Chapter 5. [1]*

In this section we will learn to formulate and solve linear diffusion problems. We use heat conduction as an example.

## 2.1 Developing the equations for 1D problem

*See Versteeg and Malalasekare [1] page 114, Section 4.1-4.3.*

The key concept for nearly all numerical solutions is discretization of the problem. In Finite Volume Method we divide the solution domain in small control volumes very similar to the ones used in analytical modeling, see Fig. 1. The difference is that in analytical modeling we allow to control volume to become infinitesimally small. In <u>Finite</u> Volume Method we use some suitably small size instead.

We use letter $P$ to refer to the central point (node) of the control volume we are currently interested in and $W$ and $E$ to refer to neighboring nodes as shown in Fig. 1.

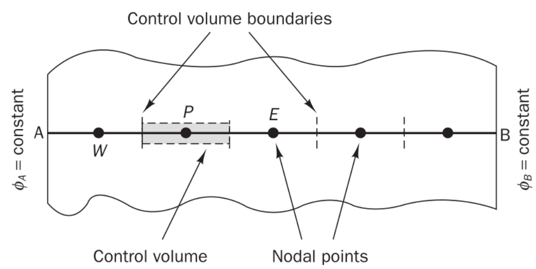



Figure 1: Control volumes and nodes [1]

In addition to control volumes and nodes we also need to refer to the faces at control volume boundaries. We call these faces $w$ and $e$, see Fig. 2. The cell size, or distance from one face to another is called $\Delta x$ as shown in Fig. 2.

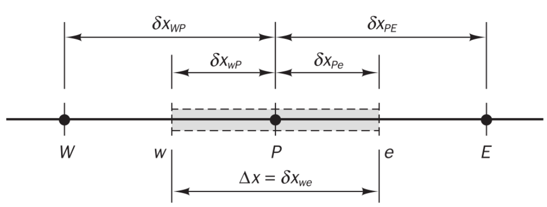



Figure 2: Faces and distances [1]

As a practical example, our domain could be a 100mm long metal rod and we could divide it in 100 control volumes, each 1mm long.

Let us concentrate on one control volume, the one around node $P$, for a while. If there is a heat flux density $q = -k\frac{\mathrm{d}T}{\mathrm{d}x}$ from neighboring cell $W$, across the face $A_w$ at the boundary between nodes $W$ and $P$, to cell $P$, the heat flux is $(qA)_w = -\left(kA\frac{\mathrm{d}T}{\mathrm{d}x}\right)_w$. We use subscripts for locations, see Fig. 3. Similarly for the east face $(qA)_e = -\left(kA\frac{\mathrm{d}T}{\mathrm{d}x}\right)_e$.

If there is a mean volumetric heat source $\overline{S}$ inside the control volume, the total heat power is $S = \overline{S}\Delta V$. A practical example of such a source is electric heating.

Similarly to the control volume used in analytical methods, we can now collect the terms in a balance equation as

$$
\begin{aligned}
(qA)_w - (qA)_e + \overline{S}\Delta V = 0 \\
\Leftrightarrow \left(kA\frac{\mathrm{d}T}{\mathrm{d}x}\right)_e - \left(kA\frac{\mathrm{d}T}{\mathrm{d}x}\right)_w + \overline{S}\Delta V = 0
\end{aligned}
\tag{1}
$$

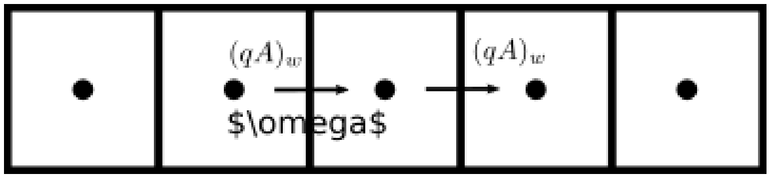



Figure 3: Heat conduction

The Eq. 1 contains the essence of Finite Volume Method. Using a familiar control volume approach we have discretized our problem for heat fluxes.

What remains is to discretize the temperature gradients $\left(\frac{\mathrm{d}T}{\mathrm{d}x}\right)_w$ and $\left(\frac{\mathrm{d}T}{\mathrm{d}x}\right)_e$ at cell faces. If thermal conductivity $k$ and/or cross-section area $A$ varies, they must be evaluated also.

The most common way to discretize gradient term is to use central differencing. The basic idea is to assume linear temperature profile between two nodes as shown in Fig. 4. In essence central differencing is the same thing as linear interpolation.

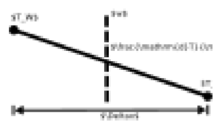


Figure 4: Central differencing

With uniform grid linear interpolation gives at cell faces

$$\boxed{\begin{aligned} k_w &= \frac{k_W + k_P}{2} \\ k_e &= \frac{k_E + k_P}{2} \\ A_w &= \frac{A_W + A_P}{2} \\ A_e &= \frac{A_E + A_P}{2} \end{aligned}} \tag{2}$$

If we have constant thermal conductivity $k$ and/or constant cross-section area $A$, we can ignore the interpolation in Eq. 2.

With uniform grid central differencing scheme gives for a temperature gradients

$$\boxed{\begin{aligned}\left(\frac{\mathrm{d}T}{\mathrm{d}x}\right)_w &\approx \frac{T_P - T_W}{\Delta x}\\ \left(\frac{\mathrm{d}T}{\mathrm{d}x}\right)_e &\approx \frac{T_E - T_P}{\Delta x}\end{aligned}} \tag{3}$$

Note that for non-uniform grid where the $\Delta x$ is not constant the form is slightly more complex.

Substituting Eq. 3 into Eq. 1

$$k_e A_e \frac{T_E - T_P}{\Delta x} - k_w A_w \frac{T_P - T_W}{\Delta x} + \overline{S}\Delta V = 0 \tag{4}$$

Now we have a fully discretized equation for heat conduction!

It is often useful to rearrange Eq. 4 as

$$\begin{aligned}&\frac{k_e A_e}{\Delta x} T_E - \frac{k_e A_e}{\Delta x} T_P - \frac{k_w A_w}{\Delta x} T_P + \frac{k_w A_w}{\Delta x} T_W + \overline{S}\Delta V = 0\\ &\Leftrightarrow \underbrace{\left(\frac{k_e A_e}{\Delta x} + \frac{k_w A_w}{\Delta x}\right)}_{a_P = a_W + a_E} T_P = \underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_w} T_W + \underbrace{\left(\frac{k_e A_e}{\Delta x}\right)}_{a_E} T_E + \underbrace{\overline{S}\Delta V}_{S_u}\end{aligned} \tag{5}$$

$$\Leftrightarrow \boxed{a_P T_P = a_W T_W + a_E T_E + S_u} \tag{6}$$

At this point our equation is in a ready-for-a-computer form but lets make a sanity check first.

From experience we know that thermal conductivity $k$, cross-section area $A$ and cell size $\Delta x$ are positive numbers. Therefore $a = \frac{kA}{\Delta x}$ is also a positive number. Now, lets imagine that the temperature in the neighboring cell $T_W$ rises. What happens to the temperature in the studied cell $T_P$? According to Eq. 6 it rises because $a_W$ is positive. This agrees with our experience. And what if we increase thermal conductivity? Multiplier $a_W$ increases and $T_P$ reacts more to changes in $T_W$. Again, this agrees with our experience. Running a similar though experiment for $a_E$ and $S_u$ gives similar results and we can conclude that there is no obvious errors in our formulation.

### 2.1.1 Same thing in math

For one-dimensional steady heat conduction (diffusion) with a volumetric source term we have

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(k\frac{\mathrm{dT}}{dx}\right) + S = 0 \tag{7}$$

where $T$ is the temperature and $s$ is the volumetric source term. Integrating over a control volume gives

$$\int_{CV} \frac{\mathrm{d}}{\mathrm{d}x}\left(k\frac{\mathrm{dT}}{dx}\right)\mathrm{d}V + \int_{CV} S\mathrm{d}V = 0 \tag{8}$$

Using the Gauss divergence theorem we get

$$\begin{aligned} \int_A \boldsymbol{n}\cdot k\frac{\mathrm{dT}}{dx}\mathrm{d}A + \int_{CV} S\mathrm{dV} = 0 \\ \Leftrightarrow \left(kA\frac{\mathrm{dT}}{dx}\right)_e - \left(kA\frac{\mathrm{dT}}{dx}\right)_w + \overline{S}\Delta V = 0 \end{aligned} \tag{9}$$

With central differencing, see Eqs. 2 and 3

$$k_e A_e \frac{T_E - T_P}{\Delta x} - k_w A_w \frac{T_P - T_W}{\Delta x} + \overline{S}\Delta V = 0 \tag{10}$$

which is the same equations as Eq. 4. Rearranging Eq. 10 results in Eq. 5.

$$\underbrace{\left(\frac{k_e A_e}{\Delta x} + \frac{k_w A_w}{\Delta x}\right)}_{a_P = a_W + a_E} T_P = \underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_w} T_W + \underbrace{\left(\frac{k_e A_e}{\Delta x}\right)}_{a_E} T_E + \underbrace{\overline{S}\Delta V}_{S_u} \tag{11}$$

$$\Leftrightarrow \boxed{a_P T_P = a_W T_W + a_E T_E + S_u} \tag{12}$$

## 2.2 Boundary conditions

In Section 2.1 we developed the discretized equations for 1D heat transfer inside the domain. In order to use the equation 5 for anything real we now need to give it boundary conditions.

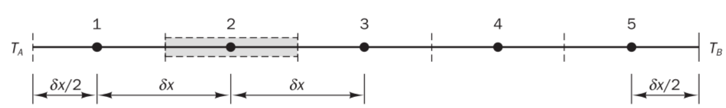



Figure 5: Boundary conditions 1D

The cells that touch boundaries are called boundary cells and require special attention. Let us concentrate on cell 1 in Fig. 5 for a while.

### 2.2.1 Constant temperature boundary

*See Versteeg and Malalasekare [1] page 118, Example 4.1.*

We define a **constant temperature $T_A$ for left boundary**. We note that the distance from node 1 to face A is $\Delta x/2$ (not all FVM meshes are build this way but the ones we use are). Using the same interpolation and central differencing schemes as before, Eqs. 2 and 3, we get for the west side face in Eq. 1

$$\left(kA\frac{\mathrm{d}T}{\mathrm{d}x}\right)_w \approx k_A A_A \frac{T_P - T_A}{\Delta x/2} \tag{13}$$

Substituting this in Eq. 1 and threating the inside face $w$ in the same way as before results in

$$k_e A_e \frac{T_E - T_P}{\Delta x} - k_A A_A \frac{T_P - T_A}{\Delta x/2} + \overline{S}\Delta V = 0 \tag{14}$$

Rearranging Eq. 14 gives

$$\begin{gathered}\underbrace{\left(\frac{k_e A_e}{\Delta x} + \overbrace{\frac{k_A A_A}{\Delta x/2}}^{S_P}\right)}_{a_P = a_E + S_P} T_P = \underbrace{\left(\frac{k_e A_e}{\Delta x}\right)}_{a_E} T_E + \underbrace{\left(\frac{k_A A_A}{\Delta x/2}\right) T_A + \overline{S}\Delta V}_{S_u} \\ \Leftrightarrow a_P T_P = a_E T_E + S_u\end{gathered} \tag{15}$$

Comparing Eqs. 5 and 15 we notice a remarkable similarity. The *east* side coefficient $a_E$ is unaffected. *West* side face doesn't exist and is replaced with source terms $S_P$ and $\left(\frac{k_A A_A}{\Delta x/2}\right) T_A$. The source term $\overline{S}\Delta V$ stays unchanged. This is typical to boundary conditions are simplifies matrix assembly. We can first assemble the inner face coefficients and volumetric sources and worry about boundaries later. More about matrix assembly later in Sec. 2.3.

If the constant temperature is set on the $B$ boundary as shown in Fig. 5 we follow a similar procedure as for $A$ boundary. Now we replace East side coefficient $a_E$ with suitable source terms and arrive to

$$\begin{gathered}\underbrace{\left(\frac{k_w A_w}{\Delta x} + \overbrace{\frac{k_B A_B}{\Delta x/2}}^{S_P}\right)}_{a_P = a_W + S_P} T_P = \underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_W} T_W + \underbrace{\left(\frac{k_B A_B}{\Delta x/2}\right) T_B + \overline{S}\Delta V}_{S_u} \\ \Leftrightarrow a_P T_P = a_W T_W + S_u\end{gathered} \tag{16}$$

### 2.2.2 Zero gradient (insulated) boundary

*See Versteeg and Malalasekare [1] page 125, Example 4.3.*

Zero gradient boundary for temperature in heat conduction context means insulations as

$$q = -k \underbrace{\frac{\mathrm{d}T}{\mathrm{d}x}}_{0} = 0 \tag{17}$$

To apply the zero gradient boundary condition for East face in Eq. 1 we but the east side coefficient to zero as $a_E = 0$ and from Eq. 5 we get

$$
\begin{aligned}
\underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_P = a_W} T_P &= \underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_w} T_W + \underbrace{\overline{S}\Delta V}_{S_u} \\
\Leftrightarrow a_P T_P &= a_W T_W + S_u
\end{aligned} \tag{18}
$$

No additional source terms are needed.

## 2.3 Matrix assembly

In previous section we have learned how to formulate the equations for our problem. Now we learn how to assemble these equations into a matrix form and feed it to a computer. We use Python for calculations. We want the problem in linear matrix equations form

$$
\boldsymbol{A}\boldsymbol{T} = \boldsymbol{b} \tag{19}
$$

which can be readily solved with almost any programming language. For example in Python, using scipy reads

```
# Solution
T = sp.linalg.solve(A,b)
```

Figure 6: Linear system solution with Scipy

We but all the constant values that do not depend on $T$ in source vector $\boldsymbol{b}$ and all the multiplying coefficients of temperatures in matrix $\boldsymbol{A}$.

Let us consider one-dimensional heat conduction where the left boundary is set at constant temperature $T_B$ and right boundary is insulated, i.e. $\frac{\mathrm{dT}}{\mathrm{d}x} = 0$. We have constant fluid properties and cross-section are $A$. The domain length is $L$ and we divide it in $n$ volumes, each $\Delta x = L/n$ long. We start the cell numbering from left and use uniform mesh. We have uniform volumetric heat generation $S$.

For the inside cells, dividing by the constant area $A$ we have from Eq. 5

$$
\begin{aligned}
\underbrace{\left(\frac{k}{\Delta x} + \frac{k}{\Delta x}\right)}_{a_P = a_W + a_E} T_P &= \underbrace{\left(\frac{k}{\Delta x}\right)}_{a_w} T_W + \underbrace{\left(\frac{k}{\Delta x}\right)}_{a_E} T_E + \underbrace{S\Delta x}_{S_u} \\
\Leftrightarrow a_P T_P &= a_W T_W + a_E T_E + S_u
\end{aligned} \tag{20}
$$

For left boundary, dividing by the constant area $A$ we have from Eq. 15

$$
\underbrace{\left(\frac{k}{\Delta x} + \overbrace{\frac{k}{\Delta x/2}}^{S_P}\right)}_{a_P = a_E + S_P} T_P = \underbrace{\left(\frac{k}{\Delta x}\right)}_{a_E} T_E + \underbrace{\left(\frac{k}{\Delta x/2}\right) T_A + \overline{S}\Delta V}_{S_u} \tag{21}
$$

$$
\Leftrightarrow a_P T_P = a_E T_E + S_u
$$

For right boundary, dividing by the constant area $A$ we have from Eq. 18

$$
\underbrace{\left(\frac{k}{\Delta x}\right)}_{a_P = a_W} T_P = \underbrace{\left(\frac{k}{\Delta x}\right)}_{a_w} T_W + \underbrace{\overline{S}\Delta V}_{S_u} \tag{22}
$$

$$
\Leftrightarrow a_P T_P = a_W T_W + S_u
$$

We may now observe from Eqs. 20-22 that the for all cells that do have a west boundary, the west face coefficient is the same $a_W = \frac{k}{\Delta x}$. The west face coefficient operates on the currently studied cell $T_P$ and the west side neighbor $T_W$.

Adding the west side coefficients into matrix $\boldsymbol{A}$ gives

$$
\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_W & a_W & 0 & 0 & 0 \\ 0 & -a_W & a_W & 0 & 0 \\ 0 & 0 & -a_W & a_W & 0 \\ 0 & 0 & 0 & -a_W & a_W \end{bmatrix} \tag{23}
$$

With python this can be done as

```python
# Coefficient matrix
A = sp.zeros((n,n))

# Add aW to matrix A
for k in range(1,n):
    A[k,k]   += aW
    A[k,k-1] += -aW
```

Similarly, for all east faces that exist the east face coefficient $a_E = \frac{k}{\Delta x}$ and the coefficient operates on the current cell and the east neighbor. Adding the east coefficients to already build matrix in Eq. 23 gives

$$
\boldsymbol{A} = \begin{bmatrix} a_E & -a_E & 0 & 0 & 0 \\ -a_W & a_W + a_E & -a_E & 0 & 0 \\ 0 & -a_W & a_W + a_E & -a_E & 0 \\ 0 & 0 & -a_W & a_W + a_E & -a_E \\ 0 & 0 & 0 & -a_W & a_W \end{bmatrix} \tag{24}
$$

With python

```
# Add aE to matrix A
for k in range(n-1):
    A[k,k]   += aE
    A[k,k+1] += -aE
```

Now we need to add the boundary conditions to our matrix $\boldsymbol{A}$ and source vector $\boldsymbol{b}$.

As can be seen from Eq. 21 the constant temperature boundary condition on the left contributes terms $S_P = \frac{k}{\Delta x/2}$ to the matrix $\boldsymbol{A}$ first cell diagonal. Adding that term gives

$$
\boldsymbol{A} = \begin{bmatrix} a_E + \frac{k}{\Delta x/2} & -a_E & 0 & 0 & 0 \\ -a_W & a_W + a_E & -a_E & 0 & 0 \\ 0 & -a_W & a_W + a_E & -a_E & 0 \\ 0 & 0 & -a_W & a_W + a_E & -a_E \\ 0 & 0 & 0 & -a_W & a_W \end{bmatrix} \tag{25}
$$

The constant boundary condition also contributes a source term $\left(\frac{k}{\Delta x/2}\right) T_A$ to the first cell source vector. The resulting source vector

$$
\boldsymbol{b} = \begin{bmatrix} \left(\frac{k}{\Delta x/2}\right) T_B & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{26}
$$

With python the constant boundary condition is added as

```
# Source vector
b = sp.zeros(n)

# Boundary  on the left
A[0,0] += k/(dx/2)
b[0]   += k/(dx/2)*T_B
```

The insulated boundary on the right produces no coefficients as no heat is transfered through the boundary. Nothing needs to be done.

As final step we need to add the volumetric source to our source vector $\boldsymbol{b}$. The volumetric source is the same in all Eqs. 20-22, $S_u = S\Delta x$. By adding it to the source vector we get

$$\boldsymbol{b} = \begin{bmatrix} \left(\frac{k}{\Delta x/2}\right) T_B + S\Delta x & S\Delta x & S\Delta x & S\Delta x & S\Delta x \end{bmatrix}^T \tag{27}$$

With python this can be done with vectorized statement as

```
# Add heat generation
b += S*dx
```

Now our linear system is ready and we can solve as shown in Fig. 6.

It's a good idea to check your code in as small increments as possible. Print out your coefficients and matrices during development and compare them to known values. Debugging of small pieces is easier that large ones. Software houses have teams of designated testers working in their companies with the sole purpose of finding bugs. Do it early.

You can plot the solutions with python as

```
# Plot numerical and exact solution
x = sp.linspace(dx/2,L-dx/2,n)
plt.plot(x, T, "k--o", label="numerical")
plt.legend()
```

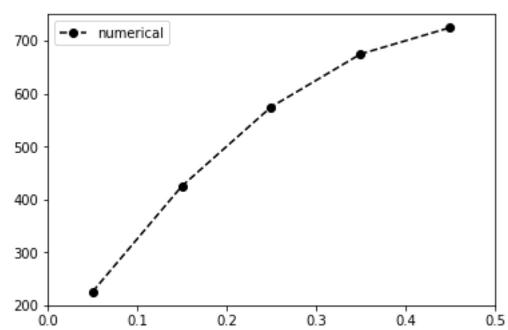



Figure 7: Plotting

## 2.4 Unsteady

*See Versteeg and Malalasekare [1] page 243, Chapter 8.*

There are many different methods to tackle the time derivative term in one-dimensional heat conduction

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + S \tag{28}$$

were $\rho$ is density, $c$ is specific heat capacity. The partial derivate $\frac{\partial T}{\partial t}$ mean that we now have two different derivatives, $\frac{\partial T}{\partial t}$ and $\frac{\partial T}{\partial x}$.

We will try to avoid mathematics as much as possible, for a more formal approach see *Versteeg and Malalasekare [1]*.

Let us first consider the left hand side of Eq. 28, $\rho c \frac{\partial T}{\partial t}$ . We may discretized it as

$$\rho c \frac{\partial T}{\partial t} \approx \rho c \frac{T - T^0}{\Delta t} \tag{29}$$

where $\Delta t$ is time step and $T^0$ is temperature from previous time step. If we now substitute Eq. 29 into Eq. 28 we get

$$\begin{aligned} \rho c \frac{\partial T}{\partial t} &= \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + S \\ \Leftrightarrow \rho c \frac{T - T^0}{\Delta t} &= \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + S \end{aligned} \tag{30}$$

we may now recognize the right hand side of Eq. 30, $\frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + S$ as the governing equation for steady state heat conduction, Eq. 7. We may use the FVM methods described before to discretize it. Using FVM to the left hand side of Eq. 30

$$\int_{CV} \rho c \frac{T - T^0}{\Delta t} dV \approx \rho c A \frac{T - T^0}{\Delta t} \Delta x \tag{31}$$

The remaining choice is at what time to we evaluate the right hand side of Eq. 30. If we choose to evaluate it at the last time step corresponding to $T^0$ we get explicit Euler scheme. Explicit Euler requires very small time steps. We choose to use the new time. This is caller implicit Euler scheme. Implicit Euler is unconditionally stable and easy to use. For more temporal accuracy look in to Runge-Kutta or other higher order schemes.

With FVM and Implicit Euler we get from Eqs. 30, 31, 5.

$$\begin{aligned} \underbrace{\rho c A \frac{\Delta x}{\Delta t}}_{a_P^0}(T_P - T_P^0) + \underbrace{\left(\frac{k_e A_e}{\Delta x} + \frac{k_w A_w}{\Delta x}\right)}_{a_W + a_E} T_P &= \underbrace{\left(\frac{k_w A_w}{\Delta x}\right)}_{a_w} T_W + \underbrace{\left(\frac{k_e A_e}{\Delta x}\right)}_{a_E} T_E + \underbrace{\overline{S}\Delta V}_{S_u} \\ \Leftrightarrow a_P^0 T_P - a_P^0 T_P^0 + a_W T_P + a_E T_P &= a_w T_W + a_E T_E + S_u \\ \Leftrightarrow \boxed{\underbrace{(a_P^0 + a_W + a_E)}_{a_P} T_P = a_w T_W + a_E T_E + a_P^0 T_P^0 + S_u} \end{aligned} \tag{32}$$

From Eq. 32 we see that the solution of one time step in unsteady case is very similar to the steady one. To progress the solution repeat the process in a loop. An example is given in Fig. 8.

```
# Solution
for step in range(1,steps+1):
    b  = bConstant + aP0*T
    T  = sp.linalg.solve(A,b)
    Ts[step] = T
    t += dt
```

Figure 8: Time stepping

## 2.5 2D case

## 2.6 Notations

## 2.7 Extra material

*Extra material. Not needed on this course.*

### 2.7.1 Performance issues with example codes

*Extra material. Not needed on this course.*

All the examples are written with clarity in mind. They usually sacrifice performance for this goal. The most important performance limiting factor is the use of dense matrices.

The multiplying matrix $\boldsymbol{A}$ is mostly empty in most FVM problems, i.e. mostly filled with zeros. It would be a LOT better to use sparse matrices. With large systems sparse matrices are decades faster that dense matrices.

The Python sparse matrix syntax is rather verbose but straight forward to use. Do not be frighten by the long command. You can usually just copy-paste it from your previous codes. See https://docs.scipy.org/doc/scipy/reference/sparse.html for more.

We often use unnecessary loops. Looping is slow in Python and indexing of large matrices is also unnecessary. As rule of a thumb always use vectorized commands if you can.

### 2.7.2 Application to other physics

*Extra material. Not needed on this course.*

Many problems in other fields are very similar to heat conduction. Here are some examples.

#### Moisture transfer

**Electric heating** Electric potential be solved from the well-known Poisson equation (same as heat conduction equation)

$$\nabla \cdot \sigma \nabla \phi = 0 \tag{33}$$

where $\phi$ is electric potential. Electric field $\boldsymbol{E}$ is solved from

$$\boldsymbol{E} = -\nabla\phi \tag{34}$$

and electric current density $\boldsymbol{J}$ from

$$\boldsymbol{J} = \sigma\boldsymbol{E} \tag{35}$$

and finally the volumetric electric heating power $p$

$$p = \frac{\partial P}{\partial V} = \boldsymbol{J}\cdot\boldsymbol{E} = \boldsymbol{J}\cdot\boldsymbol{J}/\boldsymbol{\sigma} = \frac{\mid \boldsymbol{J} \mid^{\mathbf{2}}}{\boldsymbol{\sigma}} \tag{36}$$

Electric conductivity can be calculated from electric resistivity $\rho$ as

$$\sigma = \frac{1}{\rho} \tag{37}$$

**Linear Mechanics** For small deformations

$$-\nabla\cdot\boldsymbol{\sigma} = \boldsymbol{f} \tag{38}$$

$$\boldsymbol{\sigma} = \lambda\nabla\cdot\boldsymbol{u}\mathbf{I} + 2\mu\boldsymbol{D} \tag{39}$$

The resulting equation is similar linear system as for heat conduction and can be solved with same methods.

The stress equation is very similar as that in Navier-Stokes equations. If we remove movement related terms from Navier-Stokes equations, we end up with an equation very similar to Eq. 38.

# 3 Advection-diffusion

# 4 Navier-Stokes

## 4.1 The NS equation

## 4.2 Compressible vs. incompressible

## 4.3 Pressure coupling

## 4.4 Boundary conditions

# 5 Mesh

# 6 Turbulence

# 7 Math

Tähän kappaleeseen kootaan hyödyllistä matematiikkaa, joka ei olisi sopinut prujun muihin kohtiin.

**Gaussin divergenssi lause**

$$\int_V \nabla \cdot \boldsymbol{F} dV = \int_S \boldsymbol{n} \cdot \boldsymbol{F} dA \tag{40}$$

jossa $\boldsymbol{F}$on vektori arvoinen function, esim. lämmönjohtumiselle $k\nabla T$, $\boldsymbol{n}$ on pinnan normaalivektori.

# 8 External Resources

## 8.1 Books etc.

- **Recommended for this course**
  - An Introduction to Computational Fluid Dynamics: The Finite Volume Method, Versteeg, H.K. and Malalasekera, W. 2007
- FVM diffusion in Wikipedia
  - https://en.wikipedia.org/wiki/Finite_volume_method_for_one-dimensional_steady_state_diffusion
  - https://en.wikipedia.org/wiki/Finite_volume_method_for_two_dimensional_diffusion_problem
  - https://en.wikipedia.org/wiki/Finite_volume_method_for_three-dimensional_diffusion_problem
- For advanced studies in OpenFOAM
  - https://www.researchgate.net/profile/Tobias_Holzmann/publication/307546712_Mathematics_Numerics_Derivations_and_OpenFOAMR/links/59c7ffde0f7e9bd2c014693c/Mathematics-Numerics-Derivations-and-OpenFOAMR.pdf

## 8.2 Programs, programming, libraries etc.

- https://anaconda.org/
- http://www.openfoam.org/
- http://www.openfoam.com/
- http://www.ansys.com/
- https://fenicsproject.org/
- https://www.ctcms.nist.gov/fipy/index.html

## Authors notes

- Source code 5xCtrl+, 50% scaling

# References

[1] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education Limited, 2007.