Professur für
Thermofluiddynamik

Notes on

Computational Thermo-Fluid Dynamics

Camilo F. Silva, Ph. D.

Kilian Förner, M. Sc.

Prof. Wolfgang Polifke, Ph. D.

Summer 2016

www.tfd.mw.tum.de

An expert is a man who has made all the mistakes, which can be made, in a very narrow field.

Niels Bohr (1885-1962).

Contents

1 Introduction

References

[1] KUZMIN D. *A guide to numerical methods for transport equations*. Fiedrich-Alexander-Universität, 2010.

[2] POLIFKE, W., AND KOPITZ, J. *Wärmeübertragung. Grundlagen, analytische und numerische Methoden*. Pearson Studium, 2005.

Objectives

- Getting introduced to PDEs
- Coding in Matlab analytical solutions of simple convection, diffusion and convection-diffusion equations.

Contents

1.1 Partial Differential Equations (PDEs)

How to model pollutant dispersal in a river, or the evolving distribution of a harmful gas in a city? How to describe heat conduction in solids or the propagation of sound through the atmosphere? How to understand the flow of air in the vicinity of an airfoil? A general answer to these apparently different queries is given by *Partial Differential Equations* (PDE). Generally, PDEs are derived from first principles, as for example laws of conservation. We will introduce now a general but also simple framework to formulate laws of conservation and derive PDEs from them.

Let ϕ be the concentration per unit mass of a conserved scalar. Examples of this scalar ϕ could be some intrinsic properties of a given substance as, for instance, specific internal energy, specific enthalpy or specific entropy, among others. A given component of the vector of momentum per unit mass can be also seen as an example of a conserved scalar ϕ. Now, if we assume ρ to be the density of the carrier flow, we define $w = \rho\phi$. We are refering then to the concentration of that scalar per unit volume. The whole amount of that scalar within a defined control volume V is given by

$$W = \int_V w(\boldsymbol{x}, t)\, dV = \int_V \rho(\boldsymbol{x}, t)\phi(\boldsymbol{x}, t)\, dV, \tag{1.1}$$

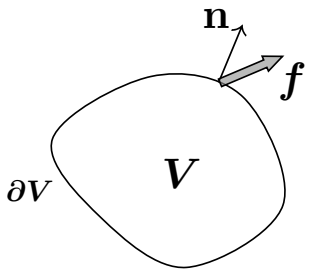


Figure 1.1: An arbitrary control volume V with boundary ∂V, boundary normal vector $\mathbf{n}$ and flux $\boldsymbol{f}$

where $\boldsymbol{x} = (x, y, z)$ is a vector describing space and t denotes time. Any variation of the scalar $w(\boldsymbol{x}, t)$ within the control volume will depend on the rate at which ϕ enters or leave the domain. This is generally expressed as a flux $\boldsymbol{f}$ normal to the surface of the control volume as depicted in Fig. 1.1. This can be mathematically written as

$$\frac{\partial}{\partial t} \int_V \overbrace{\rho(\boldsymbol{x}, t)\phi(\boldsymbol{x}, t)}^{w(\boldsymbol{x},t)} \, dV + \int_{\partial V} \boldsymbol{f} \cdot \mathbf{n} \, dS = 0. \tag{1.2}$$

It is also possible that the scalar $w(\boldsymbol{x}, t)$ is produced or extinguished within the control volume. We define then a source (sink) $s(\boldsymbol{x}, t)$ as a mechanism that produces (annihilates) a given amount of $w(\boldsymbol{x}, t)$ per unit volume and per unit time. The conservation equation reads

$$\frac{\partial}{\partial t} \int_V \overbrace{\rho(\boldsymbol{x}, t)\phi(\boldsymbol{x}, t)}^{w(\boldsymbol{x},t)} \, dV + \int_{\partial V} \boldsymbol{f} \cdot \mathbf{n} \, dS = \int_V s(\boldsymbol{x}, t) \, dV. \tag{1.3}$$

Sources (sinks) are usually placed at the right-hand side (RHS) of the conservation equation. It is then clear that the amount of $w(\boldsymbol{x}, t)$ in a defined control volume at a given time depends not only on how much of this quantity is entering and leaving the domain, but also on how much of this quantity is being produced or consumed inside the domain. Let us now apply the divergence theorem

$$\int_{\partial V} \boldsymbol{f} \cdot \mathbf{n} \, dS = \int_V \nabla \cdot \boldsymbol{f} \, dV, \tag{1.4}$$

where

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right). \tag{1.5}$$

Accordingly, Eq. (1.3) becomes

$$\frac{\partial}{\partial t}\int_V \rho(\boldsymbol{x},t)\phi(\boldsymbol{x},t)\,dV + \int_V \nabla\cdot\,\boldsymbol{f}\,dV = \int_V s(\boldsymbol{x},t)\,dV, \tag{1.6}$$

and reordering

$$\int_V \left[\frac{\partial}{\partial t}\rho(\boldsymbol{x},t)\phi(\boldsymbol{x},t)\;+\nabla\cdot\,\boldsymbol{f}\;-s(\boldsymbol{x},t)\right]dV = 0 \tag{1.7}$$

Since these integrals refers to any control volume (the choice of the control volume is arbitrary), this control volume can be chosen as small as possible (V$\rightarrow$ dV) so that the evolution of $w(\boldsymbol{x},t)$ is then described by a PDE:

$$\frac{\partial}{\partial t}\overbrace{\rho(\boldsymbol{x},t)\phi(\boldsymbol{x},t)}^{w(\boldsymbol{x},t)}+\nabla\cdot\,\boldsymbol{f} = s(\boldsymbol{x},t) \tag{1.8}$$

The only idea missing in the analysis is the evaluation of the flux $\boldsymbol{f}$ of $w(\boldsymbol{x},t)$. A given flux $\boldsymbol{f}$ is composed by a convective $\boldsymbol{f}_{\text{con}}$ and a diffusive part $\boldsymbol{f}_{\text{dif}}$. On the one hand, the convective contribution of the flux is defined as

$$\boldsymbol{f}_{\text{con}} = \boldsymbol{v}(\boldsymbol{x},t)\;w(\boldsymbol{x},t) \tag{1.9}$$

i. e. as function of the velocity of convection $\boldsymbol{v}$, i. e. the velocity of the carrier fluid, in the direction normal to the surface of the control volume. An illustration of convective flow is given in Fig. 1.2(a). On the other hand, the diffusive contribution is written as

$$\boldsymbol{f}_{\text{dif}} = -\mathcal{D}(\boldsymbol{x},t)\;\rho(\boldsymbol{x},t)\nabla\phi(\boldsymbol{x},t), \tag{1.10}$$

which is the simplest but also the most classical form. Principally, it is as a linear function between the gradient of the concentration of ϕ and a *diffusion coefficient* $\mathcal{D}(\boldsymbol{x},t)$. This expression is known as the *Fick's Law* when referring to mass diffusion and as the *Fourier's law* when talking about heat conduction, respectively. An illustration of diffusive flux is given in Fig. 1.2(b). Note that the diffusion coefficient $\mathcal{D}(\boldsymbol{x},t)$ is accompanied by a negative sign. Using this model, diffusion occurs from regions of high concentration to regions of low concentration of w.

1.2 Generalities on partial differential equations (PDEs)

After having introduced the generic form of a transport equation (Eq. (1.8)), it is interesting to perform now a standard classification of the PDEs that are often used in thermo-fluid dynam-

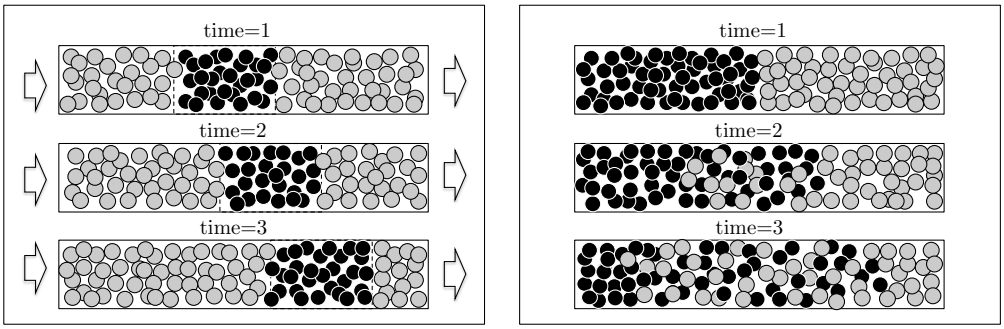


Figure 1.2: Ilustration of two type of fluxes. (a) convective flux $\boldsymbol{f}_{\text{con}}$ (b) diffusive flux $\boldsymbol{f}_{\text{dif}}$.

ics. Let us write now a standard **linear** PDE

$$A\frac{\partial^2\phi(\boldsymbol{x},t)}{\partial t^2}+B\frac{\partial^2\phi(\boldsymbol{x},t)}{\partial t\partial x}+C\frac{\partial^2\phi(\boldsymbol{x},t)}{\partial x^2}+D\frac{\partial\phi(\boldsymbol{x},t)}{\partial t}+E\frac{\partial\phi(\boldsymbol{x},t)}{\partial x}+F\phi(\boldsymbol{x},t)=G \tag{1.11}$$

where the coefficients A, B, C, D, E, F, G may either depend on time and/or space or be constant. Defining now

$$\alpha = B^2-4AC \tag{1.12}$$

it can be shown that, similar to conic sections which are defined by $Ax^2+2Bxt+Cy^2+\cdots=0$, a second order PDE is classified as:

- **Elliptic:** if $\alpha<0$
- **Parabolic:** if $\alpha=0$
- **Hyperbolic** if $\alpha>0$

whereas the PDE is always hyperbolic if A, B and C are zero. The behaviour of analytical and numerical solutions of these equations depend on how the coefficients $(A\cdots F)$ interact with each other. A 'best' numerical method for solving these three types of PDEs does not exist: a given numerical scheme could be appropriate for solving a given PDE, but would fail when intending to solve another. In addition, it is important to retain that, depending on how these coefficients are related to each other, i. e. how a PDE is classified, boundary and/or initial conditions are applied. Assuming that a given physical problem contains both convective and diffusive flux, Eq. (1.8) becomes

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla\cdot\ (\boldsymbol{v}\,\rho\phi) - \nabla\cdot\ (\mathcal{D}\rho\nabla\phi) = s, \tag{1.13}$$

where the dependency on space and time of all variables is implicitly considered. Equation (1.13) is a general transport equation and we will referred to it in this course as the *convection-diffusion-reaction* (CDR) equation. We point out that the equations of conservation of mass and energy, in fluid mechanics, may be written as

- **Mass conservation:** $\phi = 1$. Both diffusive and source terms are equal to zero.
- **Energy conservation:** $\phi = h_t$ where h_t refer to the total enthalpy of the fluid. s may contain several sources of energy as the heat release coming from combustion, for example.

The momentum equation, in contrast, is a well recognized **non**-linear PDE and therefore cannot be classified as previously described. This transport equation is written as

- **Momentum conservation:** $[\phi_1, \phi_2, \phi_3] = \boldsymbol{v}$, $s = -\nabla p$ where $\boldsymbol{v}$ and p are the velocity field and the pressure field, respectively. Note that $\rho\mathcal{D}\nabla\boldsymbol{v}$ is the viscous stress tensor and that $\mathcal{D}$ is associated, in Newtonian fluids, with two scalars known as kinematic viscosity and bulk viscosity.

1.3 Parabolic PDEs

The general transport equation (Eq. (1.13)) is recognized as a *parabolic* PDE. Neglecting the convective flux in particular, Eq. (1.20) becomes

$$\frac{\partial}{\partial t}(\rho\phi) - \nabla\cdot\ (\mathcal{D}\rho\nabla\phi) = s. \tag{1.14}$$

Replacing $\phi = cT$, $\mathcal{D}\rho c = \lambda$, $s = \dot{\omega}$ and assuming constant density and constant c, Eq. (1.14) results in

$$\rho c\frac{\partial T}{\partial t} - \nabla\cdot\ (\lambda\nabla T) = \dot{\omega}, \tag{1.15}$$

where c, λ, and $\dot{\omega}$ stand for the heat capacity of the medium, the thermal conductivity of the medium, and a heat release rate source term, respectively. In this case, the diffusion coefficient $\mathcal{D}$ is called *thermal diffusivity*. Equation (1.15) is known as the *heat equation*, a fundamental PDE in this course.

Parabolic equations always need, in addition to boundary conditions, an initial condition, i. e. an initial distribution of temperature for instance. Moreover, if a physical process is intended to reach a steady state after a given transient, then the time derivative term is expected to vanish and, consequently, the solution given by the corresponding elliptic equation (steady state version) is obtained.

1.4 Hyperbolic PDEs

Equation (1.13) is a general case of a parabolic PDE. In this equation, there are two terms in competition: the convective flux and the diffusive flux of a given scalar. An interesting case can be taken from the energy equation where $\phi = cT$, $\mathcal{D}\rho c = \lambda$, $s = \dot{\omega}$. If λ and the convective velocity field $\boldsymbol{v}$ are constant in time and space, we can write

$$\frac{\partial T}{\partial t} + \boldsymbol{v}\nabla \cdot T - \mathcal{D}\nabla^2 T = \frac{\dot{\omega}}{\rho c}. \tag{1.16}$$

When the convective flux, or simply called convection (or sometimes advection), is of several orders of magnitude larger than the diffusive flux, Eq. (1.16) starts behaving more as an hyperbolic equation than as a classical parabolic equation. The ratio of diffusion terms to convective terms is defined by the *Peclet* number:

$$\mathrm{Pe} = \frac{v_0 L_0}{\mathcal{D}_0}, \tag{1.17}$$

where v_0, L_0, and D_0 stand for a reference velocity, a reference length, and a reference diffusion coefficient, respectively. For significant high Peclet numbers, where $\boldsymbol{v} \gg \mathcal{D}$, Eq. (1.16) can be approximated to

$$\frac{\partial T}{\partial t} + \boldsymbol{v}\nabla \cdot T = \frac{\dot{\omega}}{\rho c}. \tag{1.18}$$

which is a PDE of *hyperbolic* type. An example of a physical mechanism described by Eq. (1.18) would be present in a high speed combustion chamber in which a hot spot released by the turbulent flame is convected downstream at a high velocity. This type of equation, although it may appear simpler than Eq. (1.16), is indeed more difficult to solve. The reason is that hyperbolic conservation laws admit discontinuous solutions!. Sometimes numerical schemes that aim to solve hyperbolic PDEs introduce a so-called artificial viscosity, i. e. a ficticious diffusion coefficient in order to smooth solutions and avoid, therefore, too fine meshes in regions where high gradients, or even discontinuities, of quantities involved are expected.

In a first order hyperbolic PDE, as Eq. (1.18), information travels in the direction of the flow,

i. e. 'downstream', as time evolves. As a result, this PDE needs only boundary conditions in the upstream region, i. e. at the inlet of the system. Imposing boundary conditions downstream at the domain outlet would lead to an ill-posed problem. An example of a second order hyperbolic PDE is the so-called wave equation (not shown here). In fluid dynamics, this equation is obtained by combining mass and momentum equations. This hyperbolic PDE, in contrast with the first order hyperbolic PDE, needs boundary conditions both at the inlet and the outlet of the domain since information, i.e. waves, propagates not only forwards but also backwards as time is evolving. In addition to boundary conditions, hyperbolic PDEs, as in the case for parabolic PDEs, also require an initial condition.

1.5 Elliptic PDEs

A physical process, after a given transient, may reach a steady state. In such a case, the first term of Eq. (1.13) vanishes resulting in

$$\nabla \cdot (v\, \rho\phi) - \nabla \cdot (\mathcal{D}\rho\nabla\phi) = s. \tag{1.19}$$

This equation is classified as *elliptic* as long as $\mathcal{D}$ is defined as strictly positive. Further on, in a flow at rest $v = 0$, we write

$$-\nabla \cdot (\mathcal{D}\rho\nabla\phi) = s. \tag{1.20}$$

Assuming now a constant density as well as a constant diffusion coefficient and reordering, Eq. (1.20) becomes

$$-\nabla^2\phi = s/\mathcal{D}\rho. \tag{1.21}$$

Equation (1.21) is known as the *Poisson equation* and is very useful in incompressible flows. The Poisson equation describes the steady state of heat transfer in a uniform medium, as will be shown in more detail in next chapters. When no source is present, i. e. $s = 0$, then the Poisson equations turns into the *Laplace equation* and reads as

$$\nabla^2\phi = 0. \tag{1.22}$$

Equations (1.19) to (1.22) are elliptic. All of them model a system in which any perturbation somewhere within the domain (a sudden change in the position of the source s, a perturbation on a boundary, etc) is felt immediately all over the whole domain. Accordingly, these equations require boundary conditions at every border, i. e. restrictions in all frontiers of the system. Table

Elliptic	$\nabla \cdot (\boldsymbol{v}\, \rho\phi) - \nabla \cdot (\mathcal{D}\rho\nabla\phi) = s$	Steady	Convection-Diffusion-Reaction
	- $\nabla \cdot (\mathcal{D}\rho\nabla\phi) = s$	Steady	Diffusion-Reaction
Parabolic	$\frac{\partial}{\partial t}(\rho\phi) + \nabla \cdot (\boldsymbol{v}\, \rho\phi) - \nabla \cdot (\mathcal{D}\rho\nabla\phi) = s$	Unsteady	Convection-Diffusion-Reaction
	$\frac{\partial}{\partial t}(\rho\phi) - \nabla \cdot (\mathcal{D}\rho\nabla\phi) = s$	Unsteady	Convection-Diffusion-Reaction
Hyperbolic	$\frac{\partial}{\partial t}(\rho\phi) + \nabla \cdot (\boldsymbol{v}\, \rho\phi) = s$	Unsteady	Convection-Reaction
	$\nabla \cdot (\boldsymbol{v}\, \rho\phi) = s$	Steady	Convection-Reaction

Table 1.1: Summary of PDE's classification. Reaction is described in all equations as long as $s \neq 0$.

1.1 summarizes the classification of the PDEs described in this chapter.

1.6 Boundary conditions

In order to complete the statement of the physical problem, which is modelled through a given CDR equation, we need to establish Boundary Conditions (BC), if elliptic equations are being used, or both initial and BC if parabolic or unsteady hyperbolic equations are considered. Let us consider Ω as a bounded domain with boundaries Γ. These boundaries can be decomposed

$$\Gamma = \Gamma^- + \Gamma^+ + \Gamma^0, \tag{1.23}$$

where Γ^-, Γ^+, and Γ^0 represent inlet, outlet, and walls, respectively. They are defined as

$$\textbf{Inlet} \quad \Gamma^- = \{\boldsymbol{x} \in \Gamma | \boldsymbol{f} \cdot \mathbf{n} < 0\} \tag{1.24}$$

$$\textbf{Outlet} \quad \Gamma^+ = \{\boldsymbol{x} \in \Gamma | \boldsymbol{f} \cdot \mathbf{n} > 0\} \tag{1.25}$$

$$\textbf{Wall} \quad \Gamma^0 = \{\boldsymbol{x} \in \Gamma | \boldsymbol{f} \cdot \mathbf{n} = 0\} \tag{1.26}$$

where $\mathbf{n}$ is the unit normal vector pointing outwards of the domain Ω at the point $\boldsymbol{x} \in \Gamma$, as shown in Fig. 1.3. Boundary conditions need to be applied to all boundaries if parabolic or elliptic PDEs are considered. Only Γ^- and Γ^0 should be accounted for if an hyperbolic PDE of first order is used. Boundary conditions are normally divided in three types as follows.

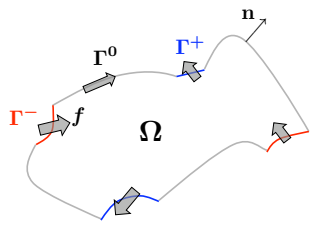


Figure 1.3: An arbitrary domain Ω with boundaries identified as inlets, outlets and walls.

1.6.1 Dirichlet boundary condition

Usually, the scalar $w = \rho\phi$ is known (or can be imposed relatively easily) at inlet and walls of a given system. Modeling requires then that w is fixed at a given value. In such situations

$$w(\boldsymbol{x}, t) = w_D(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_D, \tag{1.27}$$

where Γ_D is the subset of Γ in which *Dirichlet* BC are applied.

1.6.2 Neumann boundary condition

Neumann BC are imposed generally at the outlet, where the flux of w is known. It follows

$$\boldsymbol{f}(\boldsymbol{x}, t) \cdot \mathbf{n} = g(\boldsymbol{x}, t) \qquad \forall \boldsymbol{x} \in \Gamma_N, \tag{1.28}$$

where Γ_N is the subset of Γ in which Neumann BC are applied.

1.6.3 Robin boundary condition

Dirichlet and Neumman BC can be combined in a third type of boundaries known as *Robin* BC. They are defined as

$$w(\boldsymbol{x}, t) + \boldsymbol{f}(\boldsymbol{x}, t) \cdot \mathbf{n} = 0 \qquad \forall \boldsymbol{x} \in \Gamma_R, \tag{1.29}$$

where Γ_R is the subset of Γ on which Robin BC are applied.

1.7 Overview of the course

After visualizing the general structure of a transport equation (Eq. 1.13), it becomes intriguing to know the different ways in which they are aimed to be solved. Naturally, the first possibility that is considered involves analytical methods. Some of them rely on separation of variables, Fourier Series, eigenfunction expansions and Fourier transform techniques, among others. If a given PDE can be solved analytically, there is probably no reason to look for a numerical method. Analytical methods are sometimes preferable since they are exact, crystalline (you can usually look through them) and compact. There are situations though, where implementing an analytical method can be computationally expensive and maybe not exact. A clear example of such a situation are the solutions based on series expansions: in theory a solution is exact as long as a sum includes an infinite number of terms. In addition, analytical solutions are available only for relatively simple PDEs. This simplicity is related usually to the topology of the associated domain: lines, rectangles, circles, cuboids, spheres or a given geometry with some kind of regularity. The definition of 'simple' also considers the PDE structure for most of the cases. A linear PDE with variable coefficients is already too complex to be solved analytically. Moreover, for non-linear PDEs analytical solutions are very difficult to be obtained.

Due to these restrictions of analytical methods, numerical strategies are introduced. Numerical methods, in the framework of PDEs, can be divided in three big groups: Finite Differences (FD), Finite Volumes (FV) and Finite Elements (FEM):

- **Finite Differences (***Chap. 2***)** is a method derived from the definition of both the derivative and the Taylor series. It consists in replacing the differential terms of a PDE by the corresponding finite difference expressions. Although very powerful, since high order numerical schemes (very accurate schemes), can be easily derived, it is suitable only for relatively simple geometries. Problems with complex geometries need mapping strategies (mathematical transforms) to convert the geometry from the physical space to a computational space with suitable coordinates so that the problem becomes tractable with FD methods. Moreover, some times this method leads to numerical schemes that are not conservative.

- **Finite Volumes (***Chap. 3***)** is a numerical method widely spread in models describing fluids, i. e. in *Computational Fluid Dynamics* (CFD). This is due to the corresponding numerical scheme which is based on the conservation law, a fundamental principle in nature. Another advantage is the facility in which the numerical scheme can be adapted to complex topologies. The main drawback of FV is the relative difficulty that exists to increase the order of the associated numerical schemes.

- **Finite Elements (***Chap. 8***)** is the most spread numerical method in structural mechanics. The main reason for this success is the ability of such schemes to adapt to complex geometries in an easy way and, sometimes under particular approaches, to account for

problems with discontinuous solutions. The principal disadvantage of FEM is its complexity in both scheme derivation and numerical implementation.

Finite differences, finite volumes and finite elements are **spatial** discretization methods. Accordingly, they are used either alone when solving elliptic PDEs, or together with temporal schemes when solving **Unsteady problems (***Chap. 4***)**.

When a PDE is linear, the corresponding discretization (by any of the three methods above mentioned) is a linear algebraic system, i.e. a system of the form $A\boldsymbol{x} = \boldsymbol{b}$. Here, A, $\boldsymbol{x}$ and $\boldsymbol{b}$ stand for the system matrix, the vector of unknowns and the source vector, respectively. Solving this linear system means solving the associated PDE. There is no such a thing as 'best' algorithm to solve a linear system. The performace of a numerical algorithm when solving $A\boldsymbol{x} = \boldsymbol{b}$ depends principally both on the structure (how the non-zero elements are distributed within the matrix) and on the size of the matrix A. Consequently, it is fundamental to put some attention in **Sparse Matrices and Linear Solvers (***Chap. 5***)**

As mentioned before, analytical approaches, if available, are sometimes preferable with respect to numerical methods. At this point it is very useful to introduce **Green's functions (***Chap. 6***)**. Green's functions build an analytical framework in which a general solution of a given PDE (for general boundary conditions and source terms) is explicitly constructed as a sum of integral terms. Green functions are another powerful option to solve PDEs, although it is somehow restricted to topologies that can be easily described by either cartesian, polar or spherical coordinates.

What is the next step when a system is understood and well described by a physical model? The next step, at least from an engineering point of view, is to optimize it. Optimization is an enormous field of research and in this course an introduction to this topic will be given. Some **Optimization techniques (***Chap. 7***)** will be reviewed and applied, in addition to some of the numerical methods developed all along the course, so that the shape of a fin is optimized to maximize heat transfer.

1.8 Exercises

In this chapter we introduced partial differential equations (PDEs) in the context of transport equations. We have identified particularly three types of equations in which linear PDEs are classified: parabolic, hyperbolic or elliptic. Now, we propose to use Matlab as the numerical tool to write and visualize some analytical solutions that correspond to some specific cases of these equations.

1.8.1 1D convection equation

The 1D convection equation reads:

$$\frac{\partial \phi}{\partial t} + v_x \frac{\partial \phi}{\partial x} = 0 \tag{1.30}$$

Any function ϕ with argument $x - v_x t$ is a solution of this equation. ($\phi(x,t) = \phi(x - v_x t)$).

Stage 1: Use Matlab symbolic toolbox

We consider the function $\phi(x)$ defined as:

$$\phi(x) = \begin{cases} x & \text{if } 0 < x \leq 1 \\ (2-x) & \text{if } 1 < x \leq 2 \\ 0 & \text{if } 2 < x < L \end{cases} \tag{1.31}$$

This function can describe, for example, the distribution of temperature in a fluid at given moment. This distribution could be observed as an initial condition when solving the convection-diffusion equation. In order to introduce the information given by this function, it is necessary to express it as a Fourier sine expansion as

$$\phi(x) \approx \tilde{\phi}(x) = \sum_{m=1}^{N} a_m \sin \frac{m\pi x}{L} \tag{1.32}$$

where N is a number high enough to satisfy a good approximation $\tilde{\phi}(x)$ (It is recommended $N > 100$)[1]. The period is established as $L = 50$. The Fourier coefficients a_m are found by solving

[1]Note that Eqs. (1.32), (1.35) and (1.37) are aproximations of the exact solution. The exact solution is obtained when $N = \infty$. Here we prefer to stress that such a case is impossible to be achieved in practice and, accordingly, we do not use equalities.

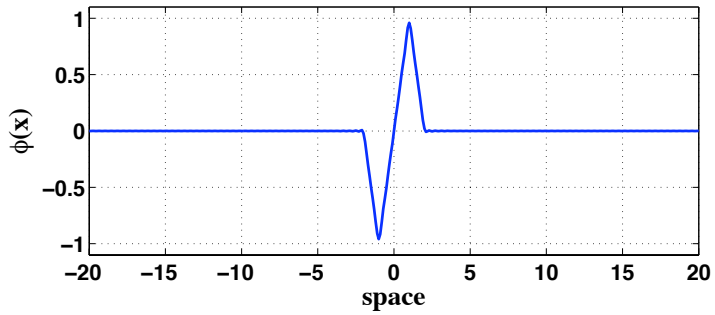


Figure 1.4: The function $\phi(x)$ with $L = 20$ and $N = 100$.

```
if t==0                    drawnow
  im=frame2im(frame)
else
end
imwrite(imind,cm,filename,'gif','WriteMode','append','Delaytime',0)
                                        frame=getframe(gcf)
[imind,cm] = rgb2ind(im,8)
imwrite(imind,cm,filename,'gif','Loopcount',inf)
```

Figure 1.5: Commands to generate a video in gif format. Note that these command are in **disorder!**

$$a_m = \frac{2}{L}\int_0^L \phi(x)\,\sin\frac{m\pi x}{L}\,dx. \tag{1.33}$$

Use matlab to compute Eq. (1.33) analitically.

Stage 2: Ploting $\tilde{\phi}(x)$

- Code the expressions for a_m according to Eq (1.33) . Subsequently, code and plot the function $\tilde{\phi}(x)$ resulting from Eq. (1.32). Try several values of N and see its influence when estimating $\phi(x)$. The function $\tilde{\phi}(x)$ should look like shown in Fig. 1.4.

Stage 3: Making a movie of convection

- Replace $\tilde{\phi}(x)$ by $\tilde{\phi}(x - v_x t)$ (note that coefficients a_m remain the same) for one value of v_x ($v_x = 1$ m/s for example) and several times ($t_{i+1} = t_i + \Delta t$) for small Δt.
- For each time t plot the function $\tilde{\phi}(x - v_x t)$. In order to create the movie (here in gif format) use the functions shown in Fig. 1.5.

1.8.2 1D diffusion equation

The diffusion equation in a one dimensional domain reads

$$\frac{\partial \phi}{\partial t} - \mathcal{D}\frac{\partial^2 \phi}{\partial x^2} = 0. \tag{1.34}$$

The solution of this equation, for homogeneous Dirichlet boundary conditions, is written as

$$\phi(x,t) \approx \tilde{\phi}(x,t) \sum_{m=1}^{N} a_m e^{-\mathcal{D}(m\pi/L)^2 t} \sin\frac{m\pi x}{L}. \tag{1.35}$$

This solution, which is based on a Fourier series expansion, is a classical way to solve the diffusion equation mentioned in numerous books. As an example we can refer to chatper 14 in [3].

Stage 4: Making a movie of diffusion

- For each time t plot the function $\tilde{\phi}(x,t)$. In order to create the movie (here in gif format) use the functions shown in Fig. 1.5.

1.8.3 1D convection diffusion equation

The convection-diffusion equation for 1D reads:

$$\frac{\partial \phi}{\partial t} + u_x\frac{\partial \phi}{\partial x} - \mathcal{D}\frac{\partial^2 \phi}{\partial x^2} = 0. \tag{1.36}$$

Analytical solutions for this equation exist and are very different depending on the boundary conditions used. For a very simple case, in which periodic boundary conditions are used ($\phi(0,t) = \phi(L,t)$), the solution can be written as:

$$\phi(x,t) \approx \tilde{\phi}(x,t) = \sum_{m=1}^{N} a_m e^{-\mathcal{D}(m\pi/L)^2 t} \sin\frac{m\pi(x - v_x t)}{L} \tag{1.37}$$

Stage 5: Making a movie of convection-diffusion

- For each time t plot the function $\tilde{\phi}(x,t)$. In order to create the movie (here in gif format) use the functions shown in Fig. 1.5.

Figure 1.6: Solution of the 1D convection-diffusion equation with $v_x = 3$ m/s and $\mathcal{D} = 0.1$ m^2/s (see Eq. (1.37)) . Left: Snapshot after $t = 1$ s. Right: Snapshot after $t = 5$ s.

1.8.4 More videos. Now in 2D (Optional)

Solutions of the previous equations can be also ploted as animated surfaces. In order to create such plots, it is first required that the vector x and $\tilde{\phi}(x,t)$ (for a given value of t) is mapped to a rectangular grid. Subsequently use the functions shown in Fig. 1.5. Two snapshots, which correspond to a solution of the convection-diffusion equation, are shown in Fig. 1.6. Note that the direction y is chosen arbitrarily.

1.8.5 Useful MATLAB commands

`syms`	Defines the following symbols to be treated as symbols by the symbolic toolbox of MATLAB (e.g. `syms x` defines `x` to be handled as symbol to do symbolic calculations.)

Example:

```
>> syms x
>> int(x)

ans =

x^2/2
```

2

Finite Differences

References

[1] MORTON, K. W., AND MAYERS, D. F. *Numerical solution of partial differential equations*. Cambridge University Press, 2005.

[2] POLIFKE, W., AND KOPITZ, J. *Wärmeübertragung. Grundlagen, analytische und numerische Methoden*. Pearson Studium, 2005.

Objectives

- Learn to derive a finite difference numerical scheme for different orders of accuracy.

Contents

2.1 Computational grid

A computational grid is a discretization of the computational domain, made of elements, faces and nodes. Finite differences is a very efficient method for relative simple topologies, and therefore it is mainly applied on Cartesian grids such as the ones shown in Fig. 2.1.

In Cartesian grids, the faces of the elements are aligned with the axes of a Cartesian frame or reference. The nodes, defined as the intersection of the faces, can be placed either uniformly

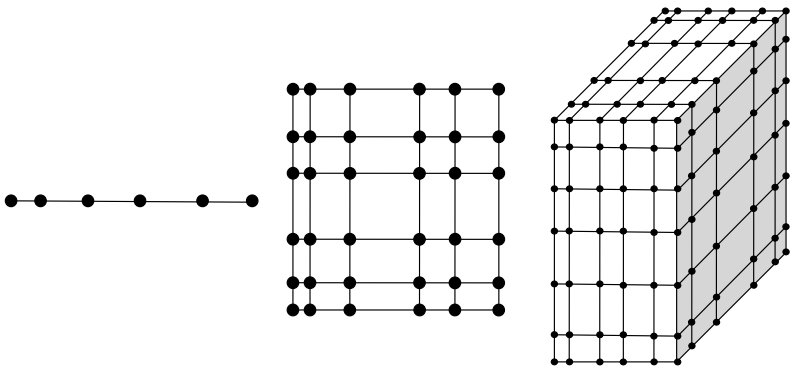


Figure 2.1: Cartesian grids in 1D, 2D and 3D

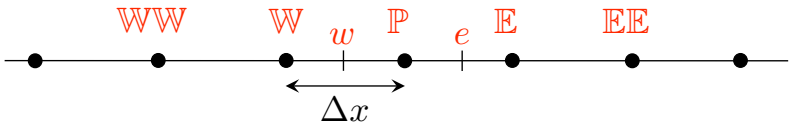


Figure 2.2: Labeling of 1D computational grid.

(equidistant) or non-uniformly. Accordingly, the elements (or cells) can be either lines (1D), rectangles (2D) or cuboids (3D).

2.2 Deriving FD numerical schemes of arbitrary order

As already mentioned at the end of last chapter, the finite differences method consists in replacing derivative expressions of a given quantity at a specific point, by a linear combination of values of that quantity at neighbour points. We could say then that derivative with respect to x of a quantity ϕ at $\mathbb{P}$ is approximated by a linear combination:

$$\left.\frac{d\phi}{dx}\right|_{x_\mathbb{P}} \approx \cdots + a_{\mathbb{WW}}\phi_{\mathbb{WW}} + \cdots + a_\mathbb{P}\phi_\mathbb{P} + a_\mathbb{E}\phi_\mathbb{E} + \cdots , \tag{2.1}$$

where the subindex indicate the position at which ϕ is evaluated. In practice, there are three schemes highly used. The definition of these schemes is referred to a stream that flows from left (upstream or $\mathbb{W}$est) to right (downstream or $\mathbb{E}$ast) as illustrated in Fig. 2.2. These schemes are:

- **Forward Euler (FE):** Linear combination of values of ϕ in the downstream region:

$$\left.\frac{d\phi}{dx}\right|_{x_\mathbb{P}} \approx a_\mathbb{P}\phi_\mathbb{P} + a_\mathbb{E}\phi_\mathbb{E} + a_{\mathbb{EE}}\phi_{\mathbb{EE}} + \cdots \tag{2.2}$$

- **Centered:** Linear combination of values of ϕ in both upstream and downstream regions:

$$\left.\frac{d\phi}{dx}\right|_{x_\mathbb{P}} \approx \cdots + a_\mathbb{W}\phi_\mathbb{W} + a_\mathbb{P}\phi_\mathbb{P} + a_\mathbb{E}\phi_\mathbb{E} + \cdots \tag{2.3}$$

- **Backward Euler (BE):** Linear combination of values of ϕ in the upstream region:

$$\left.\frac{d\phi}{dx}\right|_{x_\mathbb{P}} \approx \cdots + a_{\mathbb{WW}}\phi_{\mathbb{WW}} + a_\mathbb{W}\phi_\mathbb{W} + a_\mathbb{P}\phi_\mathbb{P} \tag{2.4}$$

The question now is how to establish appropriate values for the coefficients a so that a more accurate approximation is obtained. We will see it in the following.

2.2.1 Taylor series and truncation error

The derivation of numerical schemes in the framework of finite differences is strongly related with the definition of the Taylor series. This series expresses a function f evaluated at any point in x as an infinite sum of terms that depend on the function derivatives at a reference point $x_{\mathbb{P}}$. It is written as

$$\begin{aligned} f(x) =& \sum_{k=0}^{\infty} \frac{(x-x_{\mathbb{P}})^k}{k!} \left.\frac{d^k f}{dx^k}\right|_{x_{\mathbb{P}}} \\ =& f(x_{\mathbb{P}}) + (x-x_{\mathbb{P}}) \left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \frac{(x-x_{\mathbb{P}})^2}{2!} \left.\frac{d^2 f}{dx^2}\right|_{x_{\mathbb{P}}} + \frac{(x-x_{\mathbb{P}})^3}{3!} \left.\frac{d^3 f}{dx^3}\right|_{x_{\mathbb{P}}} + \cdots + \frac{(x-x_{\mathbb{P}})^k}{k!} \left.\frac{d^k f}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots . \end{aligned} \tag{2.5}$$

2.2.2 Forward Euler scheme

Let us start considering only two coefficients of the linear combination (Eq. (2.2)). The approximation of $\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}}$ is then given by

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx a_1 f(x_{\mathbb{P}}) + a_2 f(x_{\mathbb{E}}). \tag{2.6}$$

Considering now Eq. (2.5), fixing $x = x_{\mathbb{E}}$ and rearranging

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} = \frac{f(x_{\mathbb{E}}) - f(x_{\mathbb{P}})}{\Delta x} - \frac{\Delta x}{2!} \left.\frac{d^2 f}{dx^2}\right|_{x_{\mathbb{P}}} - \frac{(\Delta x)^2}{3!} \left.\frac{d^3 f}{dx^3}\right|_{x_{\mathbb{P}}} - \cdots - \frac{(\Delta x)^{k-1}}{k!} \left.\frac{d^k f}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots , \tag{2.7}$$

where the notation $\Delta x = x_{\mathbb{E}} - x_{\mathbb{P}}$ has been introduced. In practice, it is not possible to carry out an infinite summation of terms. Instead a truncation of the series is performed in which only the significant terms are considered. Since the derivative terms in the RHS of Eq. (2.7) are usually unknown they are dropped. The expression (2.7) is then written as

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \mathcal{O}(\Delta x) = \frac{f(x_{\mathbb{E}}) - f(x_{\mathbb{P}})}{\Delta x} \quad \text{and therefore} \quad \left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx \overbrace{\left(\frac{-1}{\Delta x}\right)}^{a_1} f(x_{\mathbb{E}}) + \overbrace{\left(\frac{1}{\Delta x}\right)}^{a_2} f(x_{\mathbb{P}}). \tag{2.8}$$

The term $\mathcal{O}(\Delta x)$ represents the **truncation error** introduced in the approximation. In this case

this term ($\mathcal{O}(\Delta x)$) is of *first order* , i. e. the same order of Δx. We say then that the scheme is *first order accurate*. We introduce the definition of order of accuracy:

Order of accuracy:	The accuracy order of a FD discretisation is the power of (Δx) to which the truncation error is approximately proportional.

It means that if the distance between nodes ($\Delta x = x_{\mathbb{E}} - x_{\mathbb{P}}$) is reduced by the half, for instance, the error in the discretisation of $\frac{df}{dx}\Big|_{x_{\mathbb{P}}}$ will also be reduced by approximately the half. In the limit of $\Delta x \to 0$, we recover the definition of the derivative:

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} = \lim_{\Delta x \to 0} \frac{f(x_{\mathbb{E}}) - f(x_{\mathbb{P}})}{\Delta x} \tag{2.9}$$

The truncation error $\mathcal{O}(\Delta x)$ will be very small for very small Δx, but it would represent considerable large number of grid points, i.e large meshes! Let us instead be smart and add another term to the linear combination. It means that we will consider the information given by a third grid point. This leads to

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx a_1 f(x_{\mathbb{P}}) + a_2 f(x_{\mathbb{E}}) + a_3 f(x_{\mathbb{EE}}). \tag{2.10}$$

We now evaluate $f(x_{\mathbb{E}})$ and $f(x_{\mathbb{EE}})$ using the Taylor series definition (Eq. (2.5)):

$$f(x_{\mathbb{E}}) = f(x_{\mathbb{P}}) + \frac{\Delta x}{1!}\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \frac{(\Delta x)^2}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x_{\mathbb{P}}} + \frac{(\Delta x)^3}{3!}\left.\frac{d^3 f}{dx^3}\right|_{x_{\mathbb{P}}} + \cdots + \frac{(\Delta x)^k}{k!}\left.\frac{d^k f}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots \tag{2.11}$$

$$f(x_{\mathbb{EE}}) = f(x_{\mathbb{P}}) + \frac{2\Delta x}{1!}\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \frac{(2\Delta x)^2}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x_{\mathbb{P}}} + \frac{(2\Delta x)^3}{3!}\left.\frac{d^3 f}{dx^3}\right|_{x_{\mathbb{P}}} + \cdots + \frac{(2\Delta x)^k}{k!}\left.\frac{d^k f}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots, \tag{2.12}$$

where a uniform grid is considered and therefore $2\Delta x = x_{\mathbb{EE}} - x_{\mathbb{P}}$. By combining Eqs. (2.11) and (2.12) with Eq. (2.10), we realize that three conditions between the coefficients arise. These conditions are:

$$a_1 + a_2 + a_3 = 0 \qquad \text{since we want} \qquad (a_1 + a_2 + a_3) f(x_{\mathbb{P}}) = 0 \tag{2.13}$$

$$a_2 \Delta x + 2a_3 \Delta x = 1 \qquad \text{since we want} \qquad (a_2 \Delta x + 2a_3 \Delta x) \left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} = \left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \tag{2.14}$$

$$a_2/2 + 2a_3 = 0 \qquad \text{since we want} \qquad (a_2(\Delta x)^2/2! + a_3(2\Delta x)^2/2!) = 0 \tag{2.15}$$

which leads to $a_1 = -3/(2\Delta x)$, $a_2 = 2/\Delta x$, $a_3 = -1/(2\Delta x)$. Plugging them into Eq. (2.10) yields

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \mathcal{O}(\Delta x)^2 = \frac{-3f(x_{\mathbb{P}}) + 4f(x_{\mathbb{E}}) - f(x_{\mathbb{EE}})}{2\Delta x}. \tag{2.16}$$

The approximation of Eq. (2.16) is of order $\mathcal{O}(\Delta x)^2$, i. e. second order accurate. The terms of second order $\mathcal{O}(\Delta x)^2$ of the Taylor series expression were 'designed' to be zero (Eq. (2.29)). Note, nevertheless, that the remaining terms of order $\mathcal{O}(\Delta x)^3$ in the Taylor series are divided by Δx in Eq. (2.16), and that is why this approximation is second order accurate and **not** third order accurate.

One advantage of a second order scheme in comparison with one of first order is that results based on a second order scheme will approach the exact value of $\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}}$ faster as $\Delta x \to 0$. It can be reasoned: if the distance between nodes Δx is reduced by two (increasing the number of elements in the mesh by two), the truncation error in the approximation of $\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}}$ will be approximately four times smaller than previously (when the cells were 2 times larger). For a first order scheme this error would be only approximately two times smaller if reducing Δx by the half. Theoretically, we could increase the order of the approximation by adding more terms to the linear combination (Eq. (2.10)). Accordingly, four terms would lead to an approximation of third order, since terms $\mathcal{O}(\Delta x)$ and $\mathcal{O}(\Delta x)^2$ will be forced to vanish; five terms would produce an approximation of fourth order, and so on. In practice, the length of the stencil, i. e. number of coefficients involved in the approximation, should no be too high since several difficulties can arise. For instance, if a large stencil scheme characterizes a linear PDE, the resulting linear system is not as easy to solve as if a short stencil is considered: the resulting matrix is less sparse and possibly more irregular with large stencils. Another problem could be also related to computational memory issues since eventually too many coefficients would be required to be stored at each node.

2.2.3 Centered scheme

Let us write the most classical centered scheme as a linear combination of two terms:

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx a_1 f(x_{\mathbb{W}}) + a_2 f(x_{\mathbb{E}}) \tag{2.17}$$

We proceed now as done previously. Using Eq. (2.5) the quantities $f(x_{\mathbb{W}})$ and $f(x_{\mathbb{E}})$ are described as

$$f(x_{\mathbb{W}}) = f(x_{\mathbb{P}}) - \frac{\Delta x}{1!}\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \frac{(\Delta x)^2}{2!}\left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} - \frac{(\Delta x)^3}{3!}\left.\frac{d^3f}{dx^3}\right|_{x_{\mathbb{P}}} + \cdots + \frac{(\Delta x)^k}{k!}\left.\frac{d^kf}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots \tag{2.18}$$

$$f(x_{\mathbb{E}}) = f(x_{\mathbb{P}}) + \frac{\Delta x}{1!}\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} + \frac{(\Delta x)^2}{2!}\left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} + \frac{(\Delta x)^3}{3!}\left.\frac{d^3f}{dx^3}\right|_{x_{\mathbb{P}}} + \cdots + \frac{(\Delta x)^k}{k!}\left.\frac{d^kf}{dx^k}\right|_{x_{\mathbb{P}}} + \cdots \tag{2.19}$$

The equations to solve are:

$$a_1 + a_2 = 0 \tag{2.20}$$
$$-a_1\Delta x + a_2\Delta x = 1 \tag{2.21}$$
$$\tag{2.22}$$

which give the solutions $a_1 = -1/(2\Delta x)$ and $a_2 = 1/(2\Delta x)$:

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx \frac{f(x_{\mathbb{E}}) - f(x_{\mathbb{W}})}{2\Delta x}. \tag{2.23}$$

which is actually second order accurate although information from only two grid points is used!. The reason comes up by replacing Eq. (2.18) and Eq. (2.19) into Eq. (2.23): the term containing (Δx) is actually eliminated by the subtraction! As done by the Forward Euler scheme, the centered scheme can have also as many terms as desired, so that accuracy is increased.

2.2.4 Backward Euler scheme

Considering two terms, the Backward Euler (BE) scheme is written:

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx a_1 f(x_{\mathbb{P}}) + a_2 f(x_{\mathbb{W}}) \tag{2.24}$$

After, repeating the procedure previously used, we obtain the approximation

$$\left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} \approx \frac{f(x_{\mathbb{P}}) - f(x_{\mathbb{W}})}{\Delta x}. \tag{2.25}$$

which is an approximation of order $\mathcal{O}(\Delta x)$. Higher accuracy can be obtained, as stated before, by adding more terms to the linear combination of Eq. (2.24).

2.2.5 Second order derivatives

We have already seen how FD schemes are designed for a first order derivative using three approaches: FE, centred and BE. Now, we will study the technique to compute a second order derivative for a defined order of accuracy. We will concentrate on the centered scheme. As before, we started expressing the second order derivative as a linear combination of three terms

$$\left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} \approx a_1 f(x_{\mathbb{W}}) + a_2 f(x_{\mathbb{P}}) + a_3 f(x_{\mathbb{E}}). \tag{2.26}$$

Subsequently, we express $f(x_{\mathbb{W}})$ and $f(x_{\mathbb{E}})$ considering Eq. (2.18) and Eq. (2.19). As a result, three equations are stated:

$$a_1 + a_2 + a_3 = 0 \quad \text{since we want} \quad (a_1 + a_2 + a_3) f(x_i) = 0 \tag{2.27}$$

$$-a_1\Delta x + a_3\Delta x = 0 \quad \text{since we want} \quad (-a_1\Delta x + a_3\Delta x) \left.\frac{df}{dx}\right|_{x_{\mathbb{P}}} = 0 \tag{2.28}$$

$$a_1(\Delta x)^2/2 + a_3(\Delta x)^2/2 = 1 \quad \text{since we want} \quad (a_1(\Delta x)^2/2! + a_2(\Delta x)^2/2!) \left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} = \left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} \tag{2.29}$$

which yield: $a_1 = 1/(\Delta x)^2$, $a_1 = -2/(\Delta x)^2$ and $a_3 = 1/(\Delta x)^2$. Replacing now these values in Eq. (2.26) yields

$$\left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} + \mathcal{O}(\Delta x)^2 = \frac{f(x_{\mathbb{W}}) - 2f(x_{\mathbb{P}}) + f(x_{\mathbb{E}})}{(\Delta x)^2}. \tag{2.30}$$

Naturally, we would expect that this approximation is of first order. Nevertheless, replacing Eq. (2.18) and Eq. (2.19) in Eq. (2.30) turns out that, in addition to the first and second terms in the RHS, also the fourth term cancels out. Consequently, the approximation given by Eq. (2.30) is second order accurate. It should be mentioned, that this relation can be also computed by

combining the definition of first order derivative as follows:

$$\left.\frac{d^2f}{dx^2}\right|_{x_{\mathbb{P}}} \approx \frac{\left.\frac{df}{dx}\right|_{x_e} - \left.\frac{df}{dx}\right|_{x_w}}{(\Delta x)} = \frac{\left[f(x_{\mathbb{E}}) - f(x_{\mathbb{P}})\right]/\Delta x - \left[f(x_{\mathbb{P}}) - f(x_{\mathbb{W}})\right]/\Delta x}{\Delta x}, \tag{2.31}$$

in which eventually the same expression given by Eq. (2.30) is recovered. As mentioned for first order derivatives, the order of accuracy of second order derivatives approximations is also strongly linked with the length of the stencil, i. e. number of coefficients involved in the linear combination: the higher the number of coefficients, the higher the order accuracy.

Finally, a new concept will be introduced:

Consistency:	A numerical scheme is consistent as long as the truncation error tends to zero when Δx tends to zero.

As observed in all previous FD schemes described, decreasing the value of Δx results in a reduction of the truncation error. Therefore, we can conclude that all these schemes are consistent. How large this reduction is, depends on the order of accuracy of the scheme.

2.3 2D steady heat equation

Now, that we have observed how first and second order derivatives are discretized, let us put that in practice. Recall the general form of a diffusion-reaction transport equation

$$\frac{\partial}{\partial t}(\rho\phi) - \nabla\cdot(\mathcal{D}\rho\nabla\phi) = s, \tag{2.32}$$

where ϕ is the concentration of a conserved scalar per unit mass, ρ is the density of the carrier flow and $\mathcal{D}$ is the diffusivity of ϕ. In an incompressible medium (constant density), the specific internal energy u is given by $u = cT$, where c is the heat capacity. Considering then u as the conserved scalar we are interested in, Eq. (2.32) becomes

$$\rho c\frac{\partial T}{\partial t} - \rho c\nabla\cdot(\mathcal{D}\nabla T) = s, \tag{2.33}$$

If now we consider the thermal diffusivity $\mathcal{D} = \lambda/(\rho c)$, where λ is the thermal conductivity of the medium, Eq. (2.33) results in

$$\rho c\frac{\partial T}{\partial t} - \nabla\cdot(\lambda\nabla T) = \dot{\omega}. \tag{2.34}$$

where $\dot{\omega}$ represents a given heat release source term. For two dimensions, Eq. (2.34) is written as

$$\rho c \frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) - \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right) = \dot{\omega}. \tag{2.35}$$

Note that we are considering a thermal conductivity λ that depends on space and, as a result, it cannot be taken outside from the spatial differential operator. In the following we will consider the steady version of the heat equation and, in addition, no source inside the domain is accounted for. The temperature distribution in the domain will only depend on the boundary conditions and on the thermal conductivity of the medium.

2.3.1 Discretizing the 2D steady heat equation by finite differences

In the steady state, and considering, the problem of interest reduces to

$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right) = -\dot{\omega} \tag{2.36}$$

Applying now the approximation suggested by Eq. (2.30) leads to

$$\begin{aligned}\left.\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)\right|_{\mathbb{P}} &\approx \frac{\lambda_e \left(\frac{\partial T}{\partial x}\right)_e - \lambda_w \left(\frac{\partial T}{\partial x}\right)_w}{\Delta x} = \frac{\lambda_e \frac{T_{\mathbb{E}} - T_{\mathbb{P}}}{\Delta x} - \lambda_w \frac{T_{\mathbb{P}} - T_{\mathbb{W}}}{\Delta x}}{\Delta x} \\ &= \frac{\lambda_e T_{\mathbb{E}} - T_{\mathbb{P}}\left(\lambda_e + \lambda_w\right) + \lambda_w T_{\mathbb{W}}}{(\Delta x)^2}\end{aligned} \tag{2.37}$$

and

$$\begin{aligned}\left.\frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right)\right|_{\mathbb{P}} &\approx \frac{\lambda_n \left(\frac{\partial T}{\partial y}\right)_n - \lambda_s \left(\frac{\partial T}{\partial y}\right)_s}{\Delta y} = \frac{\lambda_n \frac{T_{\mathbb{N}} - T_{\mathbb{P}}}{\Delta y} - \lambda_s \frac{T_{\mathbb{P}} - T_{\mathbb{S}}}{\Delta y}}{\Delta y} \\ &= \frac{\lambda_n T_{\mathbb{N}} - T_{\mathbb{P}}\left(\lambda_n + \lambda_s\right) + \lambda_s T_{\mathbb{S}}}{(\Delta y)^2}.\end{aligned} \tag{2.38}$$

where the notation (labeling of nodes) is shown in Fig. 2.3. The approximation of the steady heat equation, which is second order accurate, in a Cartesian grid is given then by

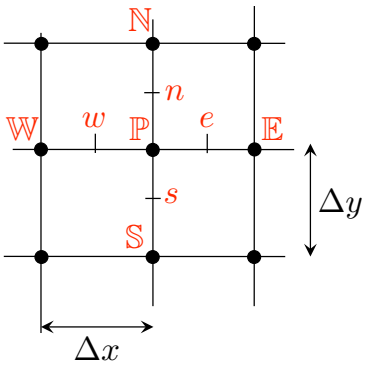


Figure 2.3: Labeling of a 2D computational grid

$$\begin{aligned}&\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)\Big|_{\mathbb{P}}+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)\Big|_{\mathbb{P}}\\ \approx&\frac{\lambda_e T_{\mathbb{E}}+\lambda_w T_{\mathbb{W}}}{(\Delta x)^2}+\frac{\lambda_n T_{\mathbb{N}}+\lambda_s T_{\mathbb{S}}}{(\Delta y)^2}-T_{\mathbb{P}}\left(\frac{\lambda_e+\lambda_w}{(\Delta x)^2}+\frac{\lambda_n+\lambda_s}{(\Delta y)^2}\right)=-\dot{\omega}\end{aligned} \tag{2.39}$$

The values of the thermal conductivity between nodes are computed by interpolation. The discretization above can be rewritten as

$$\frac{\lambda_e}{(\Delta x)^2}T_{\mathbb{E}}+\frac{\lambda_w}{(\Delta x)^2}T_{\mathbb{W}}+\frac{\lambda_n}{(\Delta y)^2}T_{\mathbb{N}}+\frac{\lambda_s}{(\Delta y)^2}T_{\mathbb{S}}+\left(-\frac{\lambda_e}{(\Delta x)^2}-\frac{\lambda_w}{(\Delta x)^2}-\frac{\lambda_n}{(\Delta y)^2}-\frac{\lambda_s}{(\Delta y)^2}\right)T_{\mathbb{P}} \tag{2.40}$$

$$=c_{\mathbb{E}}T_{\mathbb{E}}+c_{\mathbb{W}}T_{\mathbb{W}}+c_{\mathbb{N}}T_{\mathbb{N}}+c_{\mathbb{S}}T_{\mathbb{S}}+c_{\mathbb{P}}T_{\mathbb{P}}=-\dot{\omega}. \tag{2.41}$$

We want to point out that, when $\dot{\omega}=0$, the coefficient for the central node $c_{\mathbb{P}}$ can be computed as the sum of all other coefficients:

$$c_{\mathbb{P}}=-\left(c_{\mathbb{E}}+c_{\mathbb{W}}+c_{\mathbb{N}}+c_{\mathbb{S}}\right), \tag{2.42}$$

or, as will be more evident in chapter 5, that the resulting matrix is digonal dominant.

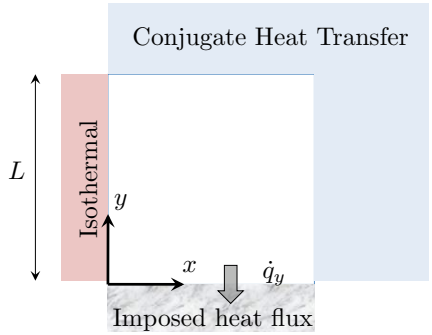


Figure 2.4: Boundary conditions

2.3.2 Boundary conditions

In the introductory chapter, three types of Boundary Conditions (BC) were introduced: Dirichlet, Neumann and Robin. In this section, they are stated in the framework of the heat equation and finite differences. Let us assume that we desire to impose a temperature profile $T_D(y)$ at the West boundary (see Fig. 2.4). The equations at nodes where $x = 0$ and $0 \leq y \leq L$ are given by

$$T_{\mathbb{P}} = T_D(y) \qquad \forall\, \mathbb{P} \in \Gamma_D \qquad \textbf{Dirichlet BC} \text{ (or first kind)} \tag{2.43}$$

where Γ_D is the set of nodes that belong to the West boundary. Let us assume now that we want to apply a given heat flux $\dot{\boldsymbol{q}} \cdot \mathbf{n} = g(\boldsymbol{x})$ at the South boundary. We want then that

$$\dot{q}_y\big|_{\mathbb{P}} = -\lambda_{\mathbb{P}} \left.\frac{\partial T}{\partial y}\right|_{\mathbb{P}} = g(x) \qquad \forall\, \mathbb{P} \in \Gamma_N \qquad \textbf{Neumann BC} \text{ (or second kind)}, \tag{2.44}$$

where Γ_N is the set of nodes that belong to the South boundary. The first idea that naturally comes to our minds is to discretize this BC as

$$-\lambda_{\mathbb{P}} \left.\frac{\partial T}{\partial y}\right|_{\mathbb{P}} = -\lambda_{\mathbb{P}} \frac{T_{\mathbb{N}} - T_{\mathbb{P}}}{\Delta y}. \tag{2.45}$$

Nevertheless, this BC is only first order accurate, as seen previously. In an elliptic equation the information (a given error for instance) propagates everywhere in the domain. Accordingly, even if we use a second order centered scheme to discretize the steady heat equation in at the interior of the domain, the solution would not be anymore second order accurate but only first order accurate due to the low order of accuracy of the downwind scheme used for discretizing the BC. Instead, we can use the second order downwind scheme (Eq. 2.16) so that

$$-\lambda_{\mathbb{P}} \left.\frac{\partial T}{\partial y}\right|_{\mathbb{P}} = -\lambda_{\mathbb{P}} \frac{-3T_{\mathbb{P}} + 4T_{\mathbb{N}} - T_{\mathbb{N}\mathbb{N}}}{2\Delta y} = g(x). \tag{2.46}$$

This is the equation that should be applied to the nodes belonging to Γ_N if we want to conserve the order of accuracy of the solution. Finally, we want to apply a Robin BC at the ℕorth and 𝔼ast boundaries of the domain. For the ℕorth boundary, the BC reads:

$$\alpha(T_{\mathbb{P}} - T_\infty) + \lambda_{\mathbb{P}} \left.\frac{\partial T}{\partial y}\right|_{\mathbb{P}} = 0 \qquad \forall \mathbb{P} \in \Gamma_R \qquad \textbf{Robin BC} \text{ (or third kind)} \tag{2.47}$$

Equation (2.47) is a classical BC for *conjugate heat transfer*, i. e. heat transfer at the interface of a solid and a fluid (see [3]). This boundary condition is widely used when evaluating cooling (or heating) of a solid caused by the thermal interaction with the fluid in which it is immersed. In this case, T_∞ stands for the temperature of the surrounding fluid and α is the *convective heat transfer coefficient*. Using a second order accurate scheme, Eq. (2.47) is discretized as:

$$\alpha(T_{\mathbb{P}} - T_\infty) + \lambda_{\mathbb{P}} \frac{3T_{\mathbb{P}} - 4T_{\mathbb{S}} + T_{\mathbb{S}\mathbb{S}}}{2\Delta y} = 0. \tag{2.48}$$

2.3.3 Assembling the linear system

Let us express Eq. (2.39), as a linear system of equations $R\ T = b_I$ for the inner nodes, and Eqs. (2.43), (2.44) and (2.47) as a linear system of equations $B\ T = b_b$ for the boundary nodes. A final system $A\ T = b$ can be assembled, where A is a matrix composed by the sub matrices R and B and b a vector composed by b_I and b_b. We have consequently a linear system

$$A\ T = b \tag{2.49}$$

to be solved, where A is $N \times N$ matrix and N is the number of nodes of the complete system after discretization. Correspondingly, T and b are vectors of size $N \times 1$.

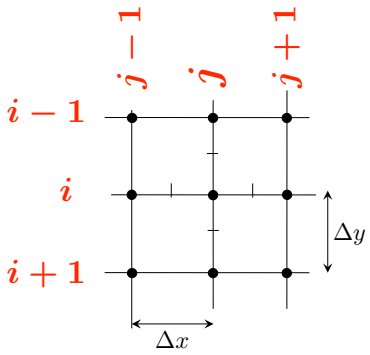


Figure 2.5: Suggested labeling of a 2D computational grid

2.4 Exercises

Discretizing the 2D steady heat equation

Stage 1: Understanding (pen and paper)

The 2D steady heat equation reads:

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)=0.. \tag{2.50}$$

- Derive a FD scheme of second order of accuracy with constant λ and Dirichlet Boundary conditions. The suggested labeling of the grid is illustrated in Fig. 2.5

Stage 2: Coding in Matlab: Dirichlet BC and constant λ

A file 'finite_diff.m' is given as a starting point. Only the heading of the code is explicitly given. It is suggested to start with very small systems (25 nodes at most).

Stage 3: Coding in Matlab: Different BC.

In addition to Dirichlet BC at the West, a Neumann BC will be imposed at the South whereas Robin BC will be imposed at North and East. This is illustrated in Fig. 2.6.

Stage 4: Coding in Matlab: Considering variable thermal conductivity (optional).

Eq. (2.50) should be now discretized by considering λ depending on space.

Stage 5: Coding in Matlab: Considering a pointwise source (optional).

Consider a non-zero source in Eq. (2.50). An illustration of such a field is given in Fig. 2.7

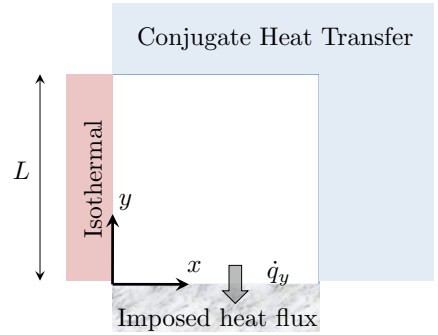


Figure 2.6: Boundary conditions at stage 3.

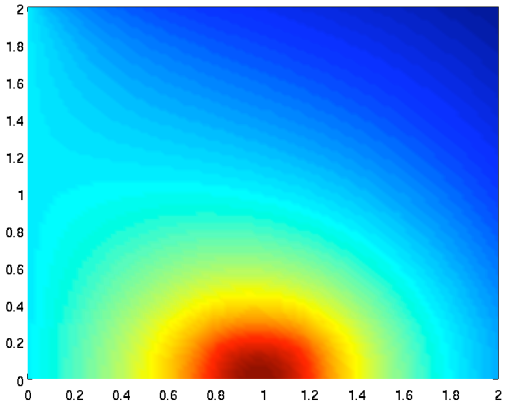


Figure 2.7: Example of temperature field obtained by finite differences.

2.4.1 Useful MATLAB commands

`index = @(ii,jj) ii+(jj-1)*n`

This is a useful definition to address an element of a vector by two coordinates. If a property that is physically in 2D space with dimensions $n \times m$ has to be saved in a 1D vector (dimension $nm \times 1$), all colums of the 2D object are concatenated to form only one column. This functions lets you address an element as if it were still in 2D (e.g. **X(index(2,2)**).

`x=A\b`

Solve the linear system $Ax = b$ for x. The backslash operator (\) selects the best way to to this within matlab.

2.4.2 Tips

by Juan Pablo Garcia (ex-student)

- It is suggested to start with a case in which $\Delta x = \Delta y$ and $L_x = L_y = 1$ in order to avoid a very complex problem from the beginning. In addition, by generating a small mesh and simple values for L_x, L_y, Δx and Δy, the problem will be easier to debug. In order to do that, is good to start with a mesh $11x11$ and having $\Delta x = \Delta y = 0.1$
- Solving the problem will be done by solving one system of equations. It is suggested then to refresh the knowledge on solving equation systems (with matrices).
- Some of the tools learned in the Teaser.m exercises are necessary. For example, generating a vector with all the values of a matrix (if the matrix is 10×10 tje vector will be 100×1).
- Give a number (index) to each node corresponding to the position in the matrix. In order to relate the position in the matrix to the position in the coordinates x, y two things should be considered:
  - As the number of column increases, it also increases the value of the x coordinate. In contrast, as the number of row increases, the value of the y coordinate decreases (due to the notation illustrated in Fig. 2.5).
  - The position in the matrix is given by (row, column), which would correspond to (y, x) instead of (x, y) that we usually use to name the points in the cartesian system of coordinates.

3

Finite Volumes

Objectives

- Learn to derive a finite volume numerical scheme for non-Cartesian domains.

Contents

In Chapter 2, we have seen how the 2D steady heat equation is discretized by the Finite Differences (FD) technique. We resolved the 2D steady heat equation considering a rectangular domain and a thermal conductivity λ dependent on space. In the present chapter we will add an additional difficulty: the evaluation of heat conduction in a **non**-Cartesian grid. The fact that

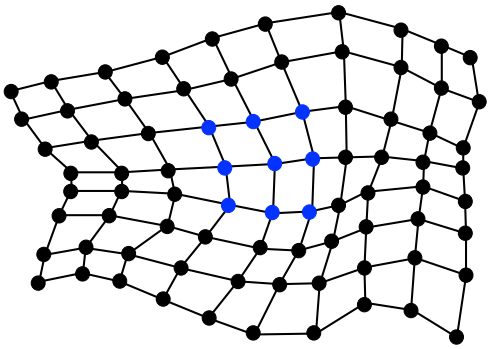


Figure 3.1: Illustration of domain and grid of interest. Nodes in blue are shown with the respective notation in Fig. 3.2

the faces of cells are not longer aligned with the Cartesian frame of reference add an enormous complexity to the problem if FD is the chosen approach. On the contrary, the Finite Volumes method, a completely different approach, 'defeats' complexity. A great advantage of FV is that under pure physical reasoning, a given PDE may be discretized even in complex geometries. Moreover, it respects the laws of conservation under which transport equations are constructed and is not computationally expensive if compared with techniques such as finite elements.

In order not to add superfluous complexity to the derivation of FV, the heat conductivity λ will be considered constant over the whole domain. We are interesting then in solving

$$\lambda \nabla^2 T = 0, \tag{3.1}$$

in a domain discretized with a non-Cartesian grid as the one illustrated in Fig. 3.1

3.1 Derivation of algebraic equations from PDE

Note that the derivation procedure that will be introduced holds for any geometry in 2D (and eventually 3D). The procedure will be described as a sequence of steps.

3.1.1 Applying divergence theorem

We integrate Eq. (3.1) over a control surface[1] $S^{\mathbb{P}}$ centered at the node $\mathbb{P}$. This control surface $S^{\mathbb{P}}$ has vertices sw, se, ne and nw, as shown in Fig. 3.2(b), where s, e, n and w stand for 'south', 'east', 'north' and 'west' with respect to the node $\mathbb{P}$. Note from Fig 3.2 that lowercase letters represent points between nodes, whereas capital letters stand for the actual nodes of the grid (Fig. 3.2(a)).

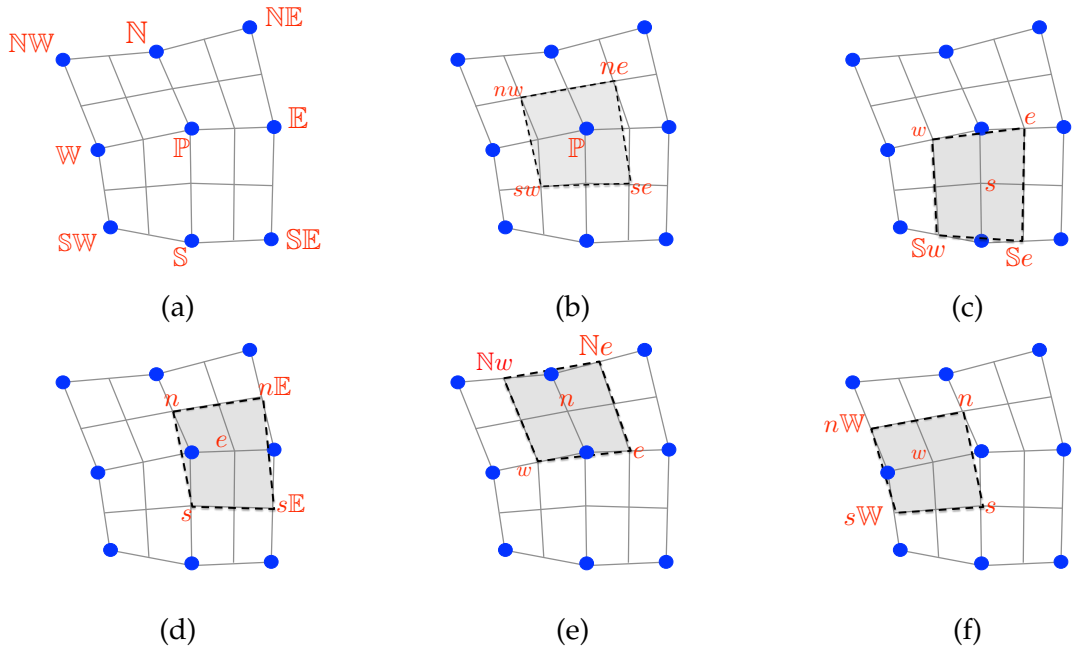


Figure 3.2: Domain of interest. Six different illustrations where nodes and middle point between nodes are labeled by upper cases and lower cases, respectively. Control surfaces are labelled with respect to the center node (or point) of the corresponding surface.

After applying the divergence theorem, this surface integral becomes a line integral. It reads:

$$\int_{S^{\mathbb{P}}} \lambda \nabla^2 T dS = \oint_{\partial S^{\mathbb{P}}} \lambda \nabla T \cdot \mathbf{n} dl = 0. \tag{3.2}$$

where $\mathbf{n}$ is the unit vector normal to the line element dl. It is positive defined if pointing outwards with respect to $S^{\mathbb{P}}$. The value of $\lambda \nabla^2 T$ at the node $\mathbb{P}$ is then given by

$$\lambda \nabla^2 T\big|_{\mathbb{P}} = \frac{1}{S^{\mathbb{P}}} \oint_{\partial S^{\mathbb{P}}} \lambda \nabla T \cdot \mathbf{n} dl. \tag{3.3}$$

Subsequently, we decompose the line integral by the corresponding contributions of the four faces:

$$\oint_{\partial S^{\mathbb{P}}} \lambda \nabla T \cdot \mathbf{n} dS = \int_{l_{sw}^{se}} \lambda \nabla T \cdot \mathbf{n} dl + \int_{l_{se}^{ne}} \lambda \nabla T \cdot \mathbf{n} dl + \int_{l_{ne}^{nw}} \lambda \nabla T \cdot \mathbf{n} dl + \int_{l_{nw}^{sw}} \lambda \nabla T \cdot \mathbf{n} dl. \tag{3.4}$$

[1]Note that we are considering here a two-dimensional case. That is why we treat a computational cell as a control surface and not as a control volume.

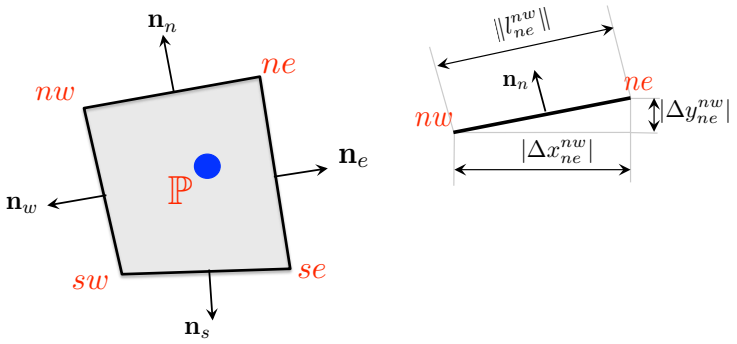


Figure 3.3: Normal vectors belonging to the control surface $S^{\mathbb{P}}$

3.1.2 Defining cell normals

The unit normal vector $\mathbf{n}$ of the four faces belonging to the control surface $S^{\mathbb{P}}$ are defined as:

$$\mathbf{n}_s = \frac{(\Delta y_{sw}^{se}, -\Delta x_{sw}^{se})}{\|l_{sw}^{se}\|} \quad \text{where} \quad \|l_{sw}^{se}\| = \sqrt{(\Delta x_{sw}^{se})^2 + (\Delta y_{sw}^{se})^2}, \tag{3.5}$$

$$\mathbf{n}_e = \frac{(\Delta y_{se}^{ne}, -\Delta x_{se}^{ne})}{\|l_{se}^{ne}\|} \quad \text{where} \quad \|l_{se}^{ne}\| = \sqrt{(\Delta x_{se}^{ne})^2 + (\Delta y_{se}^{ne})^2}, \tag{3.6}$$

$$\mathbf{n}_n = \frac{(\Delta y_{ne}^{nw}, -\Delta x_{ne}^{nw})}{\|l_{ne}^{nw}\|} \quad \text{where} \quad \|l_{ne}^{nw}\| = \sqrt{(\Delta x_{ne}^{nw})^2 + (\Delta y_{ne}^{nw})^2}, \tag{3.7}$$

$$\mathbf{n}_w = \frac{(\Delta y_{nw}^{sw}, -\Delta x_{nw}^{sw})}{\|l_{nw}^{sw}\|} \quad \text{where} \quad \|l_{nw}^{sw}\| = \sqrt{(\Delta x_{nw}^{sw})^2 + (\Delta y_{nw}^{sw})^2}, \tag{3.8}$$

where $\Delta y_a^b = y_b - y_a$ and $\Delta x_a^b = x_b - x_a$. The normal vectors are illustrated in Fig. 3.3. By recalling that $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$, the first term on the RHS of Eq. (3.4) becomes

$$\int_{l_{sw}^{se}} \lambda \nabla T \cdot \mathbf{n} dl = \frac{\lambda}{\|l_{sw}^{se}\|} \int_{l_{sw}^{se}} \left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y} \right) \cdot (\Delta y_{sw}^{se}, -\Delta x_{sw}^{se}) \, dl \tag{3.9}$$

Performing now the inner product operation yields

$$\int_{l_{sw}^{se}} \lambda \nabla T \cdot \mathbf{n} dl = \frac{\lambda \Delta y_{sw}^{se}}{\|l_{sw}^{se}\|} \int_{l_{sw}^{se}} \frac{\partial T}{\partial x} \, dl - \frac{\lambda \Delta x_{sw}^{se}}{\|l_{sw}^{se}\|} \int_{l_{sw}^{se}} \frac{\partial T}{\partial y} \, dl \tag{3.10}$$

3.1.3 Applying an integral rule

Now it is necessary to perform the integration on the resulting terms. Let us define a variable $\heartsuit$ that depends on l and assume we want to solve

$$\int_{l_{sw}^{se}} \heartsuit(l)\, dl. \tag{3.11}$$

There are several techniques available of numerical integration. We show here three generally used:

- **Mid-point rule:**

$$\int_{l_{sw}^{se}} \heartsuit(l)\, dl \approx l_{sw}^{se}\, \heartsuit|_s\, dl = \heartsuit|_s\, \|l_{sw}^{se}\| \tag{3.12}$$

- **Trapezoidal rule:**

$$\int_{l_{sw}^{se}} \heartsuit(l)\, dl \approx \int_{l_{sw}^{se}} \frac{\heartsuit|_{sw} + \heartsuit|_{se}}{2} dl = \frac{\heartsuit|_{sw} + \heartsuit|_{se}}{2} \|l_{sw}^{se}\| \tag{3.13}$$

- **Simpson's rule:**

$$\int_{l_{sw}^{se}} \heartsuit(l)\, dl \approx \int_{l_{sw}^{se}} \frac{\heartsuit|_{sw} + 4\,\heartsuit|_s + \heartsuit|_{se}}{6} dl = \frac{\heartsuit|_{sw} + 4\,\heartsuit|_s + \heartsuit|_{se}}{6} \|l_{sw}^{se}\| \tag{3.14}$$

Applying now the mid point rule to Eq. (3.10) leads to

$$\begin{aligned} \int_{l_{sw}^{se}} \lambda \nabla T \cdot \mathbf{n} dl &= \frac{\lambda \Delta y_{sw}^{se}}{\|l_{sw}^{se}\|} \|l_{sw}^{se}\| \left.\frac{\partial T}{\partial x}\right|_s - \frac{\lambda \Delta y_{sw}^{se}}{\|l_{sw}^{se}\|} \|l_{sw}^{se}\| \left.\frac{\partial T}{\partial y}\right|_s \\ &= \lambda \Delta y_{sw}^{se} \left.\frac{\partial T}{\partial x}\right|_s - \lambda \Delta x_{sw}^{se} \left.\frac{\partial T}{\partial y}\right|_s \end{aligned} \tag{3.15}$$

A similar procedure is performed to the other three terms of the RHS of Eq. (3.4). The resulting expression reads

$$\begin{aligned} \nabla^2 T\big|_P = &\frac{\lambda}{S^P} \oint_{\partial S^P} \nabla T \cdot \mathbf{n} dS \approx \\ &\frac{\lambda}{S^P} \left[\Delta y_{sw}^{se} \left.\frac{\partial T}{\partial x}\right|_s - \Delta x_{sw}^{se} \left.\frac{\partial T}{\partial y}\right|_s + \Delta y_{se}^{ne} \left.\frac{\partial T}{\partial x}\right|_e - \Delta x_{se}^{ne} \left.\frac{\partial T}{\partial y}\right|_e \right. \\ &\left. + \Delta y_{ne}^{nw} \left.\frac{\partial T}{\partial x}\right|_n - \Delta x_{ne}^{nw} \left.\frac{\partial T}{\partial y}\right|_n + \Delta y_{nw}^{sw} \left.\frac{\partial T}{\partial x}\right|_w - \Delta x_{nw}^{sw} \left.\frac{\partial T}{\partial y}\right|_w \right] \end{aligned} \tag{3.16}$$

Note that it has been assumed that s, e, n and w are the midpoints between $sw-se$, $se-ne$, $ne-nw$ and $nw-sw$, respectively. Whereas this assumption is valid for meshes that are not strongly deformed, it should be reexamined if that is not the case. For example, considering e (which is defined as the midpoint between the nodes $\mathbb{P}$ and $\mathbb{E}$) to be also the midpoint between sw and se might be a strong assumption for the grid illustrated in Fig. 3.2. Instead, a true midpoint $e^* = (x_{e^*}, y_{e^*})$ should be accounted for by doing $x_{e^*} = (x_{sw} + x_{se})/2$ and $y_{e^*} = (y_{sw} + y_{se})/2$. This is not done here in order not to add unnecesary complexity in the notation.

3.1.4 Applying Green's theorem

Green's theorem, in its general form, is defined for a line integral so that, for a point s for instance, we have

$$\left.\frac{\partial T}{\partial x}\right|_s = \frac{1}{S^s}\oint_{\partial S^s} T dy \qquad \text{and} \qquad \left.\frac{\partial T}{\partial y}\right|_s = \frac{1}{S^s}\oint_{\partial S^s} -T dx, \tag{3.17}$$

where S^s is the control surface with respect to the point s, as illustrated in Fig. 3.2(c). We decompose now the first line integral of Eq. (3.16) by the contributions of all four faces of the control surface S^s. It yields

$$\left.\frac{\partial T}{\partial x}\right|_s = \frac{1}{S^s}\int_{\mathrm{S}w}^{\mathrm{S}e} T\, dy + \frac{1}{S^s}\int_{\mathrm{S}e}^{e} T\, dy + \frac{1}{S^s}\int_{e}^{w} T\, dy + \frac{1}{S^s}\int_{w}^{\mathrm{S}w} T\, dy \tag{3.18}$$

Applying now the mid-point rule of integration, Eq. (3.18) becomes

$$\left.\frac{\partial T}{\partial x}\right|_s \approx \frac{1}{S^s}\left(\Delta y^{\mathrm{S}e}_{\mathrm{S}w} T_{\mathrm{S}} + \Delta y^{e}_{\mathrm{S}e} T_{se} + \Delta y^{w}_{e} T_P + \Delta y^{\mathrm{S}w}_{w} T_{sw}\right). \tag{3.19}$$

Carrying out the same procedure to the second line integral of Eq. (3.17) yields:

$$\left.\frac{\partial T}{\partial y}\right|_s = \frac{1}{S^s}\oint_{\partial S^s} -T dx \approx \frac{-1}{S^s}\left(\Delta x^{\mathrm{S}e}_{\mathrm{S}w} T_{\mathrm{S}} + \Delta x^{e}_{\mathrm{S}e} T_{se} + \Delta x^{w}_{e} T_P + \Delta x^{\mathrm{S}w}_{w} T_{sw}\right). \tag{3.20}$$

In the same way, we can retrieve expressions for the other spatial derivatives of Eq. (3.16):

$$\left.\frac{\partial T}{\partial x}\right|_e = \frac{1}{S^e}\left(\Delta y_s^{s\mathbb{E}} T_{se} + \Delta y_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta y_{n\mathbb{E}}^{n} T_{ne} + \Delta y_n^s T_{\mathbb{P}}\right), \tag{3.21}$$

$$\left.\frac{\partial T}{\partial y}\right|_e = \frac{-1}{S^e}\left(\Delta x_s^{s\mathbb{E}} T_{se} + \Delta x_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta x_{n\mathbb{E}}^{n} T_{ne} + \Delta x_n^s T_{\mathbb{P}}\right), \tag{3.22}$$

$$\left.\frac{\partial T}{\partial x}\right|_n = \frac{1}{S^n}\left(\Delta y_w^e T_{\mathbb{P}} + \Delta y_e^{\mathbb{N}e} T_{ne} + \Delta y_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}} + \Delta y_{\mathbb{N}w}^{w} T_{nw}\right), \tag{3.23}$$

$$\left.\frac{\partial T}{\partial y}\right|_n = \frac{-1}{S^n}\left(\Delta x_w^e T_{\mathbb{P}} + \Delta x_e^{\mathbb{N}e} T_{ne} + \Delta x_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}} + \Delta x_{\mathbb{N}w}^{w} T_{nw}\right), \tag{3.24}$$

$$\left.\frac{\partial T}{\partial x}\right|_w = \frac{1}{S^w}\left(\Delta y_{s\mathbb{W}}^{s} T_{sw} + \Delta y_s^n T_{\mathbb{P}} + \Delta y_n^{n\mathbb{W}} T_{nw} + \Delta y_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}}\right), \tag{3.25}$$

$$\left.\frac{\partial T}{\partial y}\right|_w = \frac{-1}{S^w}\left(\Delta x_{s\mathbb{W}}^{s} T_{sw} + \Delta x_s^n T_{\mathbb{P}} + \Delta x_n^{n\mathbb{W}} T_{nw} + \Delta x_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}}\right). \tag{3.26}$$

Gathering all the terms, leads to the final expression for $\nabla^2 T$ at the node $\mathbb{P}$ for the geometry of Fig. 3.2. After replacing them in Eq. (3.16), we can write

$$\begin{aligned}
\nabla^2 T\big|_{\mathbb{P}} \approx\ & \\
& \frac{\lambda \Delta y_{sw}^{se}}{S^{\mathbb{P}} S^s}\left(\Delta y_{\mathbb{S}w}^{\mathbb{S}e} T_{\mathbb{S}} + \Delta y_{\mathbb{S}e}^{e} T_{se} + \Delta y_e^w T_P + \Delta y_w^{\mathbb{S}w} T_{sw}\right) \\
+\ & \frac{\lambda \Delta x_{sw}^{se}}{S^{\mathbb{P}} S^s}\left(\Delta x_{\mathbb{S}w}^{\mathbb{S}e} T_{\mathbb{S}} + \Delta x_{\mathbb{S}e}^{e} T_{se} + \Delta x_e^w T_P + \Delta x_w^{\mathbb{S}w} T_{sw}\right) \\
+\ & \frac{\lambda \Delta y_{se}^{ne}}{S^{\mathbb{P}} S^e}\left(\Delta y_s^{s\mathbb{E}} T_{se} + \Delta y_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta y_{n\mathbb{E}}^{n} T_{ne} + \Delta y_n^s T_{\mathbb{P}}\right) \\
+\ & \frac{\lambda \Delta x_{se}^{ne}}{S^{\mathbb{P}} S^e}\left(\Delta x_s^{s\mathbb{E}} T_{se} + \Delta x_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta x_{n\mathbb{E}}^{n} T_{ne} + \Delta x_n^s T_{\mathbb{P}}\right) \\
+\ & \frac{\lambda \Delta y_{ne}^{nw}}{S^{\mathbb{P}} S^n}\left(\Delta y_w^e T_{\mathbb{P}} + \Delta y_e^{\mathbb{N}e} T_{ne} + \Delta y_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}} + \Delta y_{\mathbb{N}w}^{w} T_{nw}\right) \\
+\ & \frac{\lambda \Delta x_{ne}^{nw}}{S^{P} S^n}\left(\Delta x_w^e T_{\mathbb{P}} + \Delta x_e^{\mathbb{N}e} T_{ne} + \Delta x_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}} + \Delta x_{\mathbb{N}w}^{w} T_{nw}\right) \\
+\ & \frac{\lambda \Delta y_{nw}^{sw}}{S^{\mathbb{P}} S^w}\left(\Delta y_{s\mathbb{W}}^{s} T_{sw} + \Delta y_s^n T_{\mathbb{P}} + \Delta y_n^{n\mathbb{W}} T_{nw} + \Delta y_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}}\right) \\
+\ & \frac{\lambda \Delta x_{nw}^{sw}}{S^{\mathbb{P}} S^w}\left(\Delta x_{s\mathbb{W}}^{s} T_{sw} + \Delta x_s^n T_{\mathbb{P}} + \Delta x_n^{n\mathbb{W}} T_{nw} + \Delta x_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}}\right)
\end{aligned} \tag{3.27}$$

The areas of the control surfaces can be computed by applying the Gaussian trapezoidal for-

mula as follows

$$S^P = \frac{1}{2}\left|(x_{ne}y_{se} - x_{se}y_{ne}) + (x_{se}y_{sw} - x_{sw}y_{se}) + (x_{sw}y_{nw} - x_{nw}y_{sw}) + (x_{nw}y_{ne} - x_{ne}y_{nw})\right| \quad (3.28)$$

$$S^s = \frac{1}{2}\left|(x_e y_{\mathbb{S}e} - x_{\mathbb{S}e}y_e) + (x_{\mathbb{S}e}y_{\mathbb{S}w} - x_{\mathbb{S}w}y_{\mathbb{S}e}) + (x_{\mathbb{S}w}y_w - x_w y_{\mathbb{S}w}) + (x_w y_e - x_e y_w)\right| \quad (3.29)$$

$$S^e = \frac{1}{2}\left|(x_{n\mathbb{E}}y_{s\mathbb{E}} - x_{s\mathbb{E}}y_{n\mathbb{E}}) + (x_{s\mathbb{E}}y_s - x_s y_{s\mathbb{E}}) + (x_s y_n - x_n y_s) + (x_n y_{n\mathbb{E}} - x_{n\mathbb{E}}y_n)\right| \quad (3.30)$$

$$S^n = \frac{1}{2}\left|(x_{\mathbb{N}e}y_e - x_e y_{\mathbb{N}e}) + (x_e y_w - x_w y_e) + (x_w y_{\mathbb{N}w} - x_{\mathbb{N}w}y_w) + (x_{\mathbb{N}w}y_{\mathbb{N}e} - x_{\mathbb{N}e}y_{\mathbb{N}w})\right| \quad (3.31)$$

$$S^w = \frac{1}{2}\left|(x_n y_s - x_s y_n) + (x_s y_{s\mathbb{W}} - x_{s\mathbb{W}}y_s) + (x_{s\mathbb{W}}y_{n\mathbb{W}} - x_{n\mathbb{W}}y_{s\mathbb{W}}) + (x_{n\mathbb{W}}y_n - x_n y_{n\mathbb{W}})\right| \quad (3.32)$$

whereas the values at points within nodes are obtained by interpolation:

$$T_{sw} = \frac{T_{\mathbb{S}\mathbb{W}} + T_{\mathbb{S}} + T_P + T_{\mathbb{W}}}{4}, \quad (3.33)$$

$$T_{se} = \frac{T_{\mathbb{S}} + T_{\mathbb{S}\mathbb{E}} + T_{\mathbb{E}} + T_P}{4}, \quad (3.34)$$

$$T_{ne} = \frac{T_P + T_{\mathbb{E}} + T_{\mathbb{N}\mathbb{E}} + T_{\mathbb{N}}}{4}, \quad (3.35)$$

$$T_{nw} = \frac{T_{\mathbb{W}} + T_P + T_{\mathbb{N}} + T_{\mathbb{N}\mathbb{W}}}{4}. \quad (3.36)$$

Using this procedure to derive a FV scheme over an uniform rectangular domain allows several simplifications, explained in detail in appendix A. The resulting algebraic expression is nothing but the same derived from a FD scheme. The treatment of nodes at boundaries follows the same derivation explained in this chapter. The corresponding derivation is also shown in appendix A.

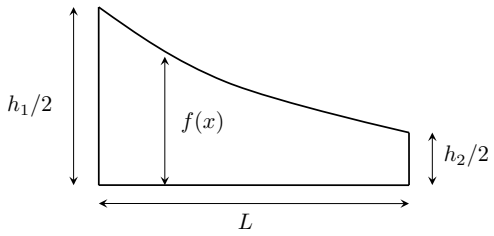


Figure 3.4: Illustration of domain of interest (note that $df/dx < 0$).

3.2 Exercises Part 1

Stage 1: Pen and paper

- Understand the derivation of Eq. (3.27).

Stage 2: Building the Grid

As starting point five matlab files are given:

1. **FVM_main.m:** This is the main file of the program. It calls the file **InitFVM** and the routines **setUpMesh** and **solveFVM**.
2. **InitFVM:** Initial parameters are given. Nothing need to be done.
3. **solveFVM:** subroutine to set up the matrix A and the vector B. In particular the vector B needs to be filled.
4. **stamp:** routine to fill the elements of matrix A. This routine is given practically empty.
5. **generate_stencil_innernode.m:** routine to buid the stencil for the inner node.

The first task consist in:

- **setUpMesh:** This routine must be written. It should take into accout the formfunction defined in InitFVM.

Stage 3: Just do it

This stage is anything but complete. Almost everything must be done

- complete routines solveFVM and stamp.

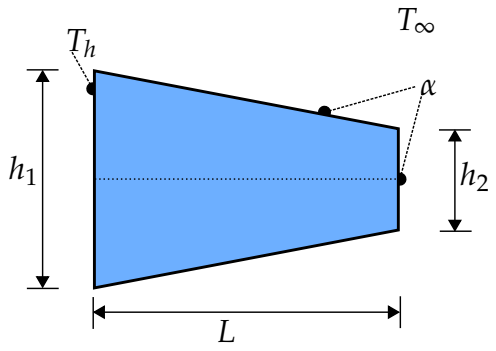


Figure 3.5: Sketch of a cooling fin.

3.3 Exercises Part 2

In the exercises, we examine the stationary temperature distribution in a cooling fin. The fin is 'fat', so the well-known quasi-1D approximation is not applicable. The configuration of the fin is sketched in Fig. 3.5.

On the west surface, we assume a constant hot temperature T_h. The other surfaces[2] are cooled by convection with an ambient temperature T_∞. We apply then the conjugated heat transfer formula

$$\dot{q}|_W = \alpha(T_W - T_\infty), \tag{3.37}$$

with the heat-transfer coefficient α. Since the fin is symmetric, we can simulate only a half fin to reduce the computational effort. Consequently, we set zero heat flux at the symmetry axis, i. e.

$$\dot{q}|_S = 0. \tag{3.38}$$

In Fig. 3.6, the computed temperature distribution is plotted (with $h_1 = 10$, $h_2 = 3$, $l = 10$, $T_h = 100$, $\lambda = 1$, $\alpha = 5$, and $T_\infty = 90$).

Stage 1: Pen and paper

- Based on the knowledge acquired when discretizing the heat equation for the inner nodes, the students are asked to derive the numerical scheme corresponding to the boundaries and the two corners.

Stage 2: Choosing a code

[2] denoted by $._W$ for "wall"

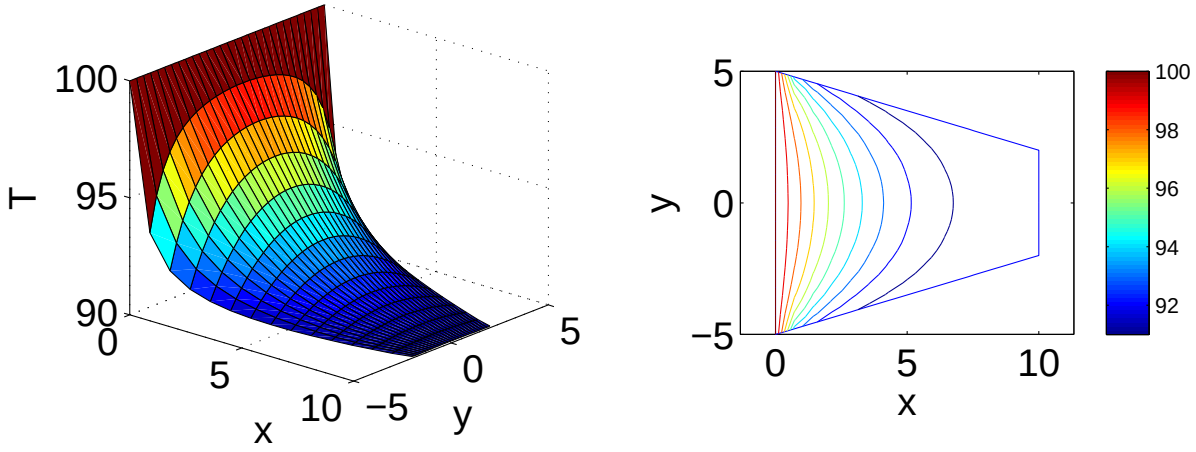


Figure 3.6: Temperature distribution in a cooling fin as surface plot on the left-hand side and as contour plot on the left.

- Each group is asked to discuss about the codes written in Session 03. Pick one of the codes.

Stage 3: Just do it

- Let's code the boundary conditions!. A text file 'stamp.m' is provided with some lines already written. Also, the files generate_stencil_east.m, generate_stencil_north.m and generate_stencil_south.m are given (but not complete).
- And what about the edge boundary conditions?

3.3.1 Useful MATLAB commands

```
switch x
case A
case B
end
```

Use this way of controlling the flow in the code to make decisions like assigning boundary conditions to boundaries.

Example:

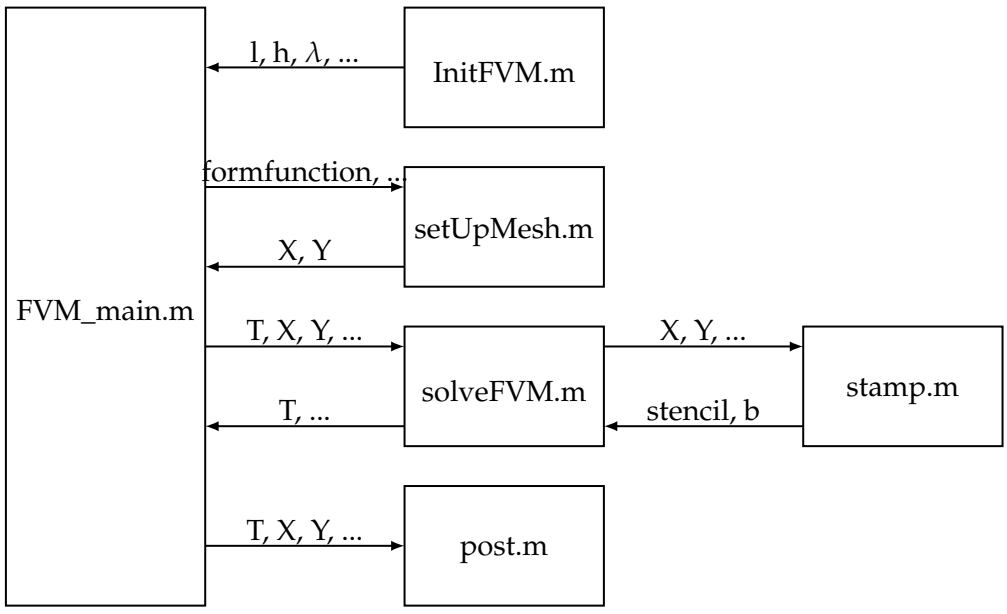
```
switch northernBoundary
  case 'Dirichlet'
  ...
  case 'Neumann'
  ...
  case 'Robin'
end
```

3.3.2 Tips

by Juan Pablo Garcia (ex-student)

- The way of working is very similar to the one related to the session of finite differences. It is then useful to review the tips provided in the previous chapter.

3.3.3 Flowchart



4

Unsteady Problems

References

[1] MORTON, K. W., AND MAYERS, D. F. *Numerical solution of partial differential equations*. Cambridge University Press, 2005.

Objectives

- Learn how to discretize the temporal operator of a transport equation.
- Learn how to compute stability of temporal schemes considering the Von Neumann stability analysis.

Contents

We have already seen how elliptic PDEs may be discretized using either Finite Difference FD, Finite Volume FV. In this chapter, we will focus on PDEs that describe unsteady processes. Let us write a convection-diffusion equation for the specific internal energy $u = cT$ of an incompressible flow (constant density ρ). Deriving this expression from Eq. (1.13) (where $\phi = u$) results in

$$\frac{\partial T}{\partial t} + \nabla \cdot \boldsymbol{v} T - \nabla \cdot \mathcal{D} \nabla T = s/\rho = s_I, \tag{4.1}$$

where the thermal diffusivity is defined as $\mathcal{D} = \lambda/(\rho c)$. Combining now the PDE associated to mass conservation $\nabla \cdot \boldsymbol{v} = 0$ with Eq. (4.1) results in

$$\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T - \mathcal{D} \nabla^2 T = s_I, \tag{4.2}$$

where $\mathcal{D}$ is considered constant in space. For a 2D flow, Eq. (4.2) becomes:

$$\frac{\partial T}{\partial t} + v_x \frac{\partial T}{\partial x} + v_y \frac{\partial T}{\partial y} - \mathcal{D} \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = s_I, \tag{4.3}$$

The second order terms (diffusive terms) of Eq. (4.3) have been already discretized by finite differences and finite volumes, as studied in previous chapters. We can express them as

$$\mathcal{D} \left(\frac{\partial^2 T}{\partial x^2} - \frac{\partial^2 T}{\partial y^2} \right) = \mathcal{R}(T), \tag{4.4}$$

where $\mathcal{R}$ represents a linear operator. This operator becomes a matrix R if discretized by a given numerical method (by FD or FV for example). The convective term of Eq. (4.3) has not yet been discretized in our course. Its discretization by FD of FV is indeed simpler than for the

second order terms. Let us denote the corresponding linear operator as $\mathcal{L}$ so that Eq. (4.3) is written now as

$$\frac{\partial T}{\partial t} + \mathcal{L}(T) - \mathcal{R}(T) = s_I, \tag{4.5}$$

or as a semi-discretized equation:

$$\frac{\partial T}{\partial t} + L\,T - R\,T = -b_I, \tag{4.6}$$

where the matrices L and R are related to the linear operators $\mathcal{L}$ and $\mathcal{R}$, respectively. In addition to Eq. (4.6), we have to consider the equations for boundary conditions, which can be of Dirichlet, Neumann or Robin type. In any case, after discretization we obtain a linear system of equations that relates the boundary nodes to the adjacent nodes. It is written as

$$B\,T = b_b \tag{4.7}$$

Equations 4.6 and 4.7 can be expressed together in a general semi-discretized equation

$$\frac{\partial T}{\partial t} - A\,T = -b, \tag{4.8}$$

where A is a $N \times N$ matrix than contains the sub-matrices $R - L$ and B. In the same way, the vector b is a global vector of size $N \times 1$ that is composed by both b_I and b_b.

4.1 Explicit time discretization

The simplest numerical scheme for time discretization, known as the *Forward Euler* (FE) scheme, is now applied to Eq. (4.8):

$$\frac{T^{n+1} - T^n}{\Delta t} - A\,T^n = -b, \tag{4.9}$$

where T^n and T^{n+1} denote the temperature T at time instants $t = n\Delta t$ and $t = (n+1)\Delta t$, respectively. It should be noted that T^{n+1} is entirely described here by the corresponding past values. Reordering Eq. (4.9) yields

$$T^{n+1} = T^n + \Delta t\,(A\,T^n - b)\,. \tag{4.10}$$

Let us emphasize that, by discretizing Eq. 4.8 explicitly in time, the temperature T^{n+1} Eq. (4.10) is obtained **without any** need to solve a linear system. As a consequence, all the solutions at each time step of the convection-diffusion equation ($T^n|_{n=1}$, $T^n|_{n=2}$, $\cdots$, $T^n|_n$) can be calculated without too much computational effort. Moreover, if a steady solution exists after a given elapsed time, this one is obtained just by advancing n times the initial solution.

Every strategy has always pros and cons. The great advantage of the Forward Euler (FE) scheme is that, as mentioned before, very simple linear systems result after time discretization, where there is not even the need to solve a linear system. Consequently, solutions at every time step are obtained in a very simple way. The big disadvantage of the FE scheme lays on its stability. Let us define stability:

Stability:	If the exact solution is bounded, the numerical solution remains bounded. It means that, for stability to hold, it is necessary that numerical errors, which are generated during the solution of discretized equations, should not be magnified.

In the worst case, explicit schemes are *unconditionally unstable*. It means that, not matter what is done, solutions will explode after certain elapsed time. In a better situation, although not the best, explicit schemes can be *conditionally stable*. As a result, there are some parameters to control so that solutions remain bounded as time grows. Most of the time, the principal parameter to control is the time step Δt as function of the cell size (Δx say).

A formal analysis of stability remains cumbersome for complex spatial discretization schemes as finite volumes (when the grid is not cartesian) and finite elements. That is the reason why most of the time formal analysis is carried out for simplified spatial schemes as finite differences. Although generality cannot be argued for such an analysis, this approach is useful in order to define non-dimensional numbers that can be used as a measure when studying stability in complex numerical problems. In the following, the Von Neumann stability analysis will be introduced.

4.1.1 Von Neumann stability analysis of FE scheme

Von Neuman analysis, also known as Fourier analysis, is a technique based on Fourier decomposition of numerical errors. It is related to FD spatial discretization of a linear PDE with constant coefficients where boundaries are assumed to be periodic. In order to introduce Von Neumann analysis, let us consider a homogeneous (s=0), linear PDE with constant coefficients (which are equal to one) with one function T that depends on two variables x (space) and t (time). This PDE reads

$$\frac{\partial T}{\partial t} + \frac{\partial T}{\partial x} - \frac{\partial^2 T}{\partial x^2} = 0, \tag{4.11}$$

This equation has particular solutions of the form $T = e^{\beta t} e^{\jmath k x}$, where β and k are real constants and $\jmath$ denotes the imaginary unit. The general solution reads as

$$T(x,t) = \sum_{m=1}^{\infty} a_m e^{\beta_m t} e^{\jmath k_m x}, \tag{4.12}$$

which is nothing but a linear combination of the particular solutions. Here, a_m are recognized as the Fourier coefficients, k_m is the *wave number* defined as $k_m = \pi m / L$, where L is the length of the domain, and β_m is a real constant that depends on m. It can also be shown that the general solution of the discretized equation associated to (4.11) at a given time $t_n = n\Delta t$ and at a given discretized point $x_i = i\Delta x$ reads [1]:

$$T_i^n = \sum_{m=1}^{M} a_m e^{\beta_m n \Delta t} e^{\jmath k_m i \Delta x}, \tag{4.13}$$

where $M = L/\Delta x$. The error in the approximation ϵ_i^n is defined by

$$\epsilon_i^n = T_i^n - T(x_i, t_n) \tag{4.14}$$

and is strongly associated to the truncation error made when discretizing. Since both the exact solution $T(x_i, t_n)$ and the discretized solution T_i^n satisfy the discretized equation exactly, the error ϵ_i^n also satisfies the dicretized equation. Accordingly, the error can be also expressed as a linear combination of the form of Eq. (4.13), or particularly as

$$\epsilon_i^n = e^{\beta_m n \Delta t} e^{\jmath k_m i \Delta x}. \tag{4.15}$$

The error ϵ_i^{n+1} and a time $t^{n+1} = (n+1)\Delta t$ is expressed as

$$\epsilon_i^{n+1} = e^{\beta_m (n+1) \Delta t} e^{\jmath k_m i \Delta x} = e^{\beta_m \Delta t}\, \epsilon_i^n \tag{4.16}$$

It is evident that whether the error grows, stays constant or decreases depends on whether $e^{\beta_m \Delta t}$ is bigger, equal or smaller than unity, respectively. In order to simplify the notation, let us define the *amplification factor* $e^{\beta_m \Delta t}$ as

$$G \equiv \frac{\epsilon_i^{n+1}}{\epsilon_i^n} = e^{\beta_m \Delta t}. \tag{4.17}$$

[1] Be aware that here $\jmath$ denotes the imaginary unit whereas i and j denote the nodes of a given computational grid.

Stability analysis of 1D convection-diffusion equation

Now that we have expressed the error in Eq. (4.15), we proceed to analyze the stability of schemes discretizing Eq. (4.11). Note that no attention is given to boundary conditions (they are considered periodic) and, therefore, their contribution on stability is neglected. In addition, since Von Neumann analysis is aimed for FD schemes, we define the operators describing convective and diffusion mechanisms as

$$\mathcal{L}(T) \approx v_x \frac{(T_{i+1} - T_{i-1})}{2\Delta x} \tag{4.18}$$

$$\mathcal{R}(T) \approx \mathcal{D} \frac{(T_{i-1} - 2T_i + T_{i+1})}{(\Delta x)^2}, \tag{4.19}$$

which are obtained using a second order accurate centered 1D-FD scheme. Replacing these operators in Eq. (4.10) and reordering yields

$$T_i^{n+1} = T_i^n - \frac{\nu}{2}(T_{i+1}^n - T_{i-1}^n) + \delta(T_{i-1}^n - 2T_i^n + T_{i+1}^n), \tag{4.20}$$

where

$$\nu = \frac{v_x \Delta t}{\Delta x} \quad \text{and} \quad \delta = \frac{\mathcal{D}\Delta t}{(\Delta x)^2}. \tag{4.21}$$

The non-dimensional numbers ν and δ are of extreme importance in the stability analysis and are known as the *Courant number* and *diffusion number*, respectively. As mentioned before, the error in the approximation ϵ_j^n also satisfies Eq. (4.20) and therefore we can write

$$\epsilon_i^{n+1} = \epsilon_i^n - \frac{\nu}{2}(\epsilon_{i+1}^n - \epsilon_{i-1}^n) + \delta(\epsilon_{i-1}^n - 2\epsilon_i^n + \epsilon_{i+1}^n). \tag{4.22}$$

Let us now divide all terms by ϵ_i^n and replace by the definition of the error (Eq. (4.15)). This results in

$$G = 1 - \frac{\nu}{2}(e^{\jmath k_m \Delta x} - e^{-\jmath k_m \Delta x}) + \delta(e^{-\jmath k_m \Delta x} - 2 + e^{\jmath k_m \Delta x}). \tag{4.23}$$

Pure convective equation

In order to simplify the analysis, let us assume first a pure convection problem ($\delta = 0$) and develop the exponential terms using the Euler's formula. It leads to

$$G = 1 - \frac{\nu}{2}\left[\cos(k_m\Delta x) + \jmath\sin(k_m\Delta x) - \cos(k_m\Delta x) + \jmath\sin(k_m\Delta x)\right] \tag{4.24}$$

$$G = 1 - \jmath\nu\sin(k_m\Delta x) \quad \Rightarrow \quad |G|^2 = 1 + \nu^2\sin^2(k_m\Delta x), \tag{4.25}$$

where $\pi/N \leq k_m\Delta x \leq \pi$ and N is the number of nodes of the discretized domain. Equation (4.25) means that, even for very large values of N, the second order centered FD scheme **always** presents $|G| > 1$, and therefore is unconditionally unstable. Let us redefine the operator $\mathcal{L}(T)$ as

$$\mathcal{L}(T) \approx v_x\frac{(T_i - T_{i-1})}{\Delta x}, \tag{4.26}$$

i.e. as a first order upwind 1D-FD scheme. The equation of the error becomes

$$\epsilon_i^{n+1} = \epsilon_i^n - \nu(\epsilon_i^n - \epsilon_{i-1}^n) \tag{4.27}$$

which leads to

$$G = 1 - \nu\left[1 - \cos(k_m\Delta x) + \jmath\sin(k_m\Delta x)\right]. \tag{4.28}$$

It can be shown that for this case, $G \leq 1$ as long as the **CFL condition** is satisfied, i. e. as long as $\mathbf{0 \leq \nu \leq 1}$. The upwind scheme is recognized to be *conditionally stable* for the pure convection equation. There is a more general way to interpret the CFL condition. In the convection equation, an hyperbolic PDE of first order, the information propagates always from the upstream to the downstream region of the domain. The velocity of this propagation is the velocity of the carrier flow. In a one dimensional domain with constant mean flow, the pure convection equation is written as Eq. (4.11) neglecting the diffusive term. The solution of this equation is actually any function f with argument $(x - v_x t)$. It means, that a perturbation f is constant over a *characteristic line*, or in other words, that a perturbation f is convected at a velocity v_x without being disturbed. If f is a Gaussian pulse (a distribution of temperature, say), this pulse is convected along the domain without losing its Gaussian distribution (see also exercises of Chapter 1). This example is illustrated in Fig. 4.1(a).

Figure 4.1(a) shows that the solution $T(x_i, t_n)$ of the PDE depends **only** on the information (on the values of T) at the points belonging to the characteristic line. Let us now spatially discretize the convection equation with the operator $\mathcal{L}$ of Eq. (4.26), and use the Foward Euler scheme with two different time steps defined as $\Delta t = \nu\Delta x/v_x$ with $\nu < 1$ and $\nu > 1$, respectively. Figure 4.1(b), illustrate the domain of dependence of T_i^n when these two different Δt are used. The *domain of dependence* of the solution T_i^n is actually the data used to compute this solution. It

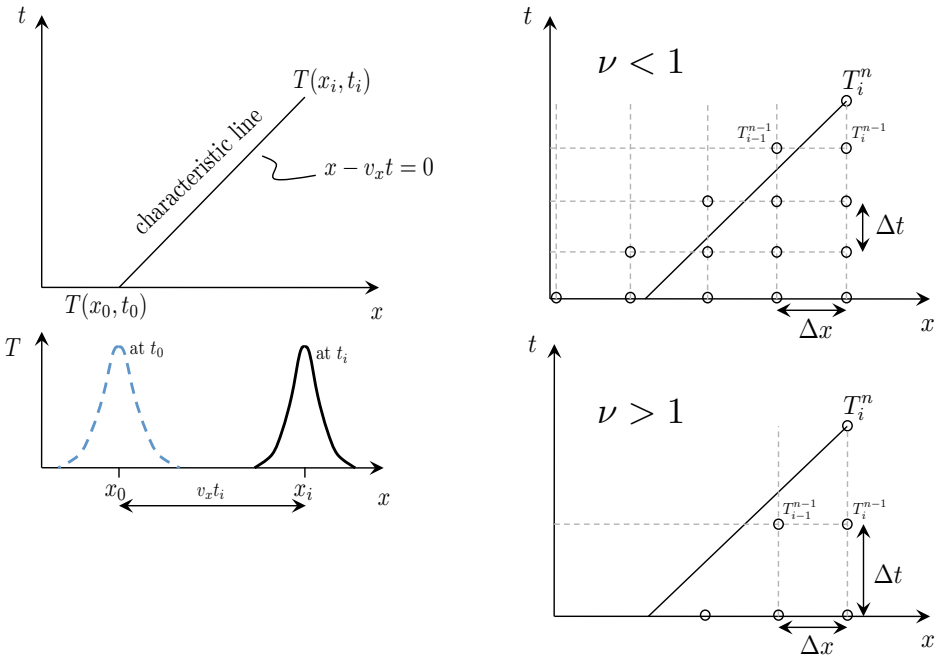


Figure 4.1: Illustration of the CFL condition.

is interesting to notice that for $\nu > 1$ the domain of dependence of the discretized equation does not contain the domain of dependance of the PDE, i.e. the characteristic line, and actually the solution diverges after a certain time. In contrast, convergence is achieved when $\nu < 1$ since the characteristic line lies within the domain of dependence of the discretized PDE. It should be stressed out, though, that the CFL condition is a necessary condition, but no sufficient, for stability.

Pure diffusion equation

The same procedure can be carried out to perform stability analysis on a pure diffusion equation. Let us consider Eq. (4.23) with $\nu = 0$:

$$G = 1 + \delta(e^{-\jmath k_m \Delta x} - 2 + e^{\jmath k_m \Delta x}). \tag{4.29}$$

and developing by means of Euler's formula

$$G = 1 + \delta\left[\cos(k_m\Delta x) - \jmath\sin(k_m\Delta x) - 2 + \cos(k_m\Delta x) + \jmath\sin(k_m\Delta x)\right] \tag{4.30}$$

$$G = 1 - 2\delta\left[1 - \cos(k_m\Delta x)\right] \quad \Rightarrow \quad G = 1 - 4\delta\sin^2(k_m\Delta x/2). \tag{4.31}$$

The worst case takes place when $k_m \Delta x = \pi$. Stability is conditioned then by

$$|1 - 4\delta| \leq 1 \quad \Rightarrow \quad \delta \leq \frac{1}{2} \tag{4.32}$$

Stability analysis of 2D diffusion equation

During this course, we focus on the study of the heat equation mainly in two dimensional domains. It is worth then to analyse the stability conditions in such cases. The solution of a PDE with constant coefficients of a function T that depends on three variables (x, y and t) is generally obtained by applying the technique of separation of variables. Accordingly, the a error is now expressed as

$$\epsilon_{i,j}^n = e^{\beta_m n \Delta t} e^{\jmath k_{x,m} i \Delta x} e^{\jmath k_{y,m} j \Delta y} \tag{4.33}$$

where $k_{x,m}$ and $k_{y,m}$ represent the wave numbers in the x and y direction, respectively. Let us define now the operator $\mathcal{R}(T)$ according to a second order centered 2D-FD scheme:

$$\mathcal{R}(T) \approx \mathcal{D} \left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{(\Delta x)^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{(\Delta y)^2} \right) \tag{4.34}$$

This defintion is now inserted in Eq. (4.10). After neglecting the convective terms, it yields

$$T_{i,j}^{n+1} = T_{i,j}^n + \frac{\mathcal{D}\Delta t}{(\Delta x)^2}(T_{i-1,j}^n - 2T_{i,j}^n + T_{i+1,j}^n) + \frac{\mathcal{D}\Delta t}{(\Delta y)^2}(T_{i,j-1}^n - 2T_{i,j}^n + T_{i,j+1}^n). \tag{4.35}$$

As mentioned before, the error in the approximation as defined in Eq. (4.3.1) also satisfies Eq. (4.35). Therefore we can write

$$\epsilon_{i,j}^{n+1} = \epsilon_{i,j}^n + \frac{\mathcal{D}\Delta t}{(\Delta x)^2}(\epsilon_{i-1,j}^n - 2\epsilon_{i,j}^n + \epsilon_{i+1,j}^n) + \frac{\mathcal{D}\Delta t}{(\Delta y)^2}(\epsilon_{i,j-1}^n - 2\epsilon_{i,j}^n + \epsilon_{i,j+1}^n) \tag{4.36}$$

Dividing now all terms by $\epsilon_{i,j}^n$, knowing that $G = \epsilon_{i,j}^{n+1} / \epsilon_{i,j}^n$ and using the definition of Eq. (4.3.1), Eq. (4.36) becomes

$$G = 1 + \frac{\mathcal{D}\Delta t}{(\Delta x)^2}(e^{-\jmath k_{x,m}\Delta x} - 2 + e^{\jmath k_{x,m}\Delta x}) + \frac{\mathcal{D}\Delta t}{(\Delta y)^2}(e^{-\jmath k_{y,m}\Delta y} - 2 + e^{\jmath k_{y,m}\Delta y}) \tag{4.37}$$

Applying now the Euler's formula[2] leads to

$$G = 1 - 4\frac{\mathcal{D}\Delta t}{(\Delta x)^2}\sin^2(k_{x,m}\Delta x/2) - 4\frac{\mathcal{D}\Delta t}{(\Delta y)^2}\sin^2(k_{y,m}\Delta y/2). \tag{4.38}$$

The expression of Eq. (4.38) holds for a rectangular domain where Δx is not necessary equal to Δy. The worst case arises when $k_{x,m}\Delta x = k_{x,m}\Delta y = \pi$. The stability condition is then

$$\left|1 - 4\frac{\mathcal{D}\Delta t}{(\Delta x)^2} - 4\frac{\mathcal{D}\Delta t}{(\Delta y)^2}\right| \leq 1 \quad \Rightarrow \quad \Delta t \leq \frac{1}{2\mathcal{D}}\left[\frac{(\Delta x)^2(\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}\right]. \tag{4.39}$$

In the particular case when $\Delta x = \Delta y$, the stability condition reduces to

$$\boxed{\Delta t \leq \frac{(\Delta x)^2}{4\mathcal{D}}}. \tag{4.40}$$

Figures 4.2, 4.3 and 4.4 show an example of temperature evolution through time in a trapezoidal fin. Two different values of Δt have been considered in order to visualize the stability of the FE scheme (explicit scheme). We have observed so far, that stability of an explicit scheme is not easy to obtain. Once a given numerical scheme is known to be consistent and in addition is proved to be stable, then we can claim that such a numerical scheme is convergent.

Convergence:	This property is achieved if the numerical solution approach the exact solution of the PDE and converge to it as the mesh size tends to zero. In other words a scheme is convergent if it is both consistent and stable.

4.2 Implicit time discretization

We have by now studied the stability issues of an explicit scheme. Although these schemes do not imply the need of solving complex linear systems at each time step Δt, they do require generally small values Δt to assure convergence. The idea is now to evaluate what happens to stability if we go to the other extreme, i.e. if we consider fully implicit schemes. In a fully implicit scheme, the time evolution of a quantity is computed based on future values of that quantity. Therefore, instead of applying the spatial operator $\mathcal{C}$ to T at $t^n = n\Delta t$, it is applied to T at $t^{n+1} = (n+1)\Delta t$. Therefore, Eq. (4.8) becomes

[2]
$$e^{\jmath\alpha} - 2 + e^{-\jmath\alpha} = 4\left(\frac{e^{\jmath\alpha/2\cdot 2} - 2 + e^{-\jmath\alpha/2\cdot 2}}{2^2}\right) = 4\left(\frac{e^{\jmath\alpha/2} - e^{-\jmath\alpha/2}}{2}\right)^2 = 4\sin^2(\alpha/2)$$

$$\frac{T^{n+1} - T^n}{\Delta t} - A\, T^{n+1} = -b. \tag{4.41}$$

Eq. (4.42) can be reorganized as

$$\underbrace{(I - \Delta t\, A)}_{A^*}\, T^{n+1} = \underbrace{T^n - \Delta t\, b}_{b^*}, \tag{4.42}$$

to make clear that a linear system $A^*\, T^{n+1} = b^*$ is needed to be solved to obtain the field T^{n+1}.

4.2.1 Von Neumann analysis

Let us now define the operators $\mathcal{L}$ and $\mathcal{R}$, as done before (the Eqs. (4.18) and (4.19)). Subsequently, we replace them in Eq. (4.42), where $\mathcal{A}$ is defined as in Eq. (4.8), and we neglect the influence of the boundary conditions on the stability analysis as previously done. We obtain

$$T_i^{n+1} + \frac{\nu}{2}(T_{i+1}^{n+1} - T_{i-1}^{n+1}) - \delta(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}) = T_i^n. \tag{4.43}$$

where it becomes evident that obtaining the value of T^{n+1} in an implicit scheme is not as easy as it was for a explicit scheme: we need now to solve a linear system at each time step Δt. Nevertheless, despite this disadvantage, there is an enormous improvement in the stability of such schemes. Applying the Von Neumann analysis for the corresponding pure diffusion problem ($\nu = 0$), it can be shown [XX1] that the amplification factor is given by

$$G = \frac{1}{1 + 4\delta \sin^2(k\Delta x/2)}. \tag{4.44}$$

Thus, the fully implicit scheme is then *unconditionally stable*, the best situation for stability, since there is no value of δ, defined as strictly positive, for which $G \geq 1$.

Figures 4.2, 4.3 and 4.4 show an example of temperature evolution through time in a trapezoidal fin. One value of Δt have been considered in order to visualize the strong stability of the BE scheme (implicit scheme).

4.3 The weighted average or θ-method

Equation (4.42) represents a fully implicit scheme, also known as *Backward Euler* (BE) Method. A generalization can be performed for cases that lay between fully explicit (Forward Euler) and

fully implicit (Backward Euler). It is done by adding a weight coefficient θ which 'tunes' the numerical scheme towards either FE or BE. The weighted expression reads:

$$\frac{T^{n+1} - T^n}{\Delta t} - \theta A\, T^{n+1} - (1-\theta) A\, T^n = -b. \tag{4.45}$$

Note that a value of $\theta = 0$ tunes the scheme to fully explicit while a value of $\theta = 1$ makes it fully implicit.

4.3.1 Von Neuman Analysis

1D convection-diffusion

The amplification of the θ-scheme is given by [XX1]

$$G = \frac{1 - 4(1-\theta)\delta \sin^2(k\Delta x/2)}{1 + 4\theta\delta \sin^2(k\Delta x/2)}. \tag{4.46}$$

The worst case takes place when $k\Delta x = \pi$. In such situation, stability is then conditioned by

$$\left| \frac{1 - 4(1-\theta)\delta}{1 + 4\theta\delta} \right| \leq 1 \quad \Rightarrow \quad \delta(1-2\theta) \geq \frac{1}{2}. \tag{4.47}$$

Equation (4.47) tells us, on the one hand, that for values $0.5 \leq \theta \leq 1$ the numerical scheme derived is expected to be unconditionally stable. On the other hand, values of θ in the range of $0 \leq \theta < 0.5$ give conditionally stable schemes. Values of θ very close to 0.5 (but still smaller) are unstable only for very large values of Δt.

A special case of the θ-method is the so called **Crank-Nicolson** scheme, in which $\theta = 0.5$. Following the analysis of order of accuracy seen in previous chapters (by applying the definition of Taylor series), it can be demonstrated that this value makes the scheme become second order accurate in time $\mathcal{O}(\Delta t)^2$. For FE and BE schemes the approximations in time are only of first order $\mathcal{O}(\Delta t)$.

2D diffusion

Analogly to Eq. (4.36), the error $\epsilon_{i,j}^{n+1}$ at (i,j) for the time step $n+1$ using the weighted average method can be expressed as:

$$\begin{aligned}\epsilon_{i,j}^{n+1} =& \epsilon_{i,j}^{n} + \Delta t\, \theta\, \mathcal{D} \left(\frac{\epsilon_{i-1,j}^{n+1} - 2\epsilon_{i,j}^{n+1} + \epsilon_{i+1,j}^{n+1}}{(\Delta y)^2} + \frac{\epsilon_{i,j-1}^{n+1} - 2\epsilon_{i,j}^{n+1} + \epsilon_{i,j+1}^{n+1}}{(\Delta x)^2} \right) + \\ & (1-\theta)\, \mathcal{D} \left(\frac{\epsilon_{i-1,j}^{n} - 2\epsilon_{i,j}^{n} + \epsilon_{i+1,j}^{n}}{(\Delta y)^2} + \frac{\epsilon_{i,j-1}^{n} - 2\epsilon_{i,j}^{n} + \epsilon_{i,j+1}^{n}}{(\Delta x)^2} \right) \end{aligned} \tag{4.48}$$

Applying the Fourier transform of the error and dividing by $\epsilon_{i,j}^{n}$, this can be rewritten as:

$$\begin{aligned} G =& 1 - e^{\beta_m \Delta t} \Delta t\, \theta\, \mathcal{D} \left(\frac{e^{\jmath k_{x,m}\Delta x} - 2 + e^{-\jmath k_{x,m}\Delta x}}{(\Delta x)^2} + \frac{e^{\jmath k_{y,m}\Delta y} - 2 + e^{-\jmath k_{y,m}\Delta y}}{(\Delta y)^2} \right) + \\ & \Delta t (1-\theta)\, \mathcal{D} \left(\frac{e^{\jmath k_{x,m}\Delta x} - 2 + e^{-\jmath k_{x,m}\Delta x}}{(\Delta x)^2} + \frac{e^{\jmath k_{y,m}\Delta y} - 2 + e^{-\jmath k_{y,m}\Delta y}}{(\Delta y)^2} \right) \\ =& 1 - \Delta t\, \mathcal{D} \left[e^{\beta_m \Delta t}\theta + 1 - \theta \right] \left(\frac{e^{\jmath k_{x,m}\Delta x} - 2 + e^{-\jmath k_{x,m}\Delta x}}{(\Delta x)^2} + \frac{e^{\jmath k_{y,m}\Delta y} - 2 + e^{-\jmath k_{y,m}\Delta y}}{(\Delta y)^2} \right) \end{aligned} \tag{4.49}$$

By definition, $G = e^{\beta_m \Delta t}$. Furthermore, we know from the previous discussion, that $e^{\jmath\alpha} - 2 + e^{-\jmath\alpha} = \sin^2(\alpha/2)$ and so we can solve for G:

$$\begin{aligned} G &= \frac{1 - \Delta t\, \mathcal{D}(1-\theta)}{1 + \Delta t\, \mathcal{D}\theta} \left(\frac{e^{\jmath k_{x,m}\Delta x} - 2 + e^{-\jmath k_{x,m}\Delta x}}{(\Delta x)^2} + \frac{e^{\jmath k_{y,m}\Delta y} - 2 + e^{-\jmath k_{y,m}\Delta y}}{(\Delta y)^2} \right) \\ &= \frac{1 - \Delta t\, \mathcal{D}(1-\theta)}{1 + \Delta t\, \mathcal{D}\theta} \left(\frac{4\sin^2(k_{x,m}\Delta x/2)}{(\Delta x)^2} + \frac{4\sin(k_{y,m}\Delta y/2)}{(\Delta y)^2} \right) \end{aligned} \tag{4.50}$$

We recall that a scheme is stable if $|G| \leq 1$. For $\theta \in [0.5, 1]$, this condition is satisfied for every Δt, for other θ values, we do a worst case estimation and receive:

$$\boxed{\Delta t \leq \frac{1}{2\mathcal{D}\,(1-2\theta)} \left[\frac{(\Delta x)^2 (\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2} \right]} \tag{4.51}$$

4.4 Predictor-corrector methods (Runge-Kutta)

As seen previously, it seems that fully explicit methods are 'at most' *conditionally stable* and only first order accurate. Is there any strategy to build explicit schemes with higher levels of accuracy and a better stability conditioning? The answer is yes. There are techniques, known

as predictor-corrector methods, in which information of the prediction made by a simple explicit scheme is not considered as the final result, but instead is used to construct more robust algorithms. Suppose then we start assuming that

$$T^{n+1} = T^n + \Delta t\ A\ T^n, \tag{4.52}$$

Equation (4.52) represents the FE scheme studied previously. Now, instead of assuming the output of this equation as the final result, we take it as a preliminary prediction and we note it with $\widetilde{}$:

$$\widetilde{T}^{n+1} = T^n + \Delta t A\ T^n \qquad \Longleftarrow \quad \text{Predictor} \tag{4.53}$$

The next step is to establish a correction formulation to correct it. A suitable one could be the Crank-Nicolson method. It yields

$$T^{n+1} = T^n + \frac{1}{2}\Delta t\ \left[A\ T^n + A\ \widetilde{T}^{n+1}\right], \qquad \Longleftarrow \quad \text{Corrector} \tag{4.54}$$

where the final estimate of T^{n+1} is obtained. This two-level method is a good example to introduce predictor-corrector methods. They are not the best to be used in practice, though, since the associated stability is not ideal. We can, nevertheless, go further. Let us now introduce multi-stage methods. These methods aim to use the information given by $(A\ T)$ at times t that lay between $n\Delta t < t < (n+1)\Delta t$. **Runge-Kutta** formulations belong to this category. Let us consider the most classical, called explicit Runge-Kutta of order 4 (some times also referred as RK4). Under this method, the final estimate of T^{n+1} is now given by:

$$\underbrace{T^{n+1} = T^n + \frac{1}{6}\Delta t\left[A\ T^n + 2A\ \dot{T}^{n+1/2} + 2A\ \ddot{T}^{n+1/2} + A\ \dddot{T}^{n+1}\right]}_{\text{Correction for } t=(n+1)\Delta t}, \tag{4.55}$$

where

$$\dot{T}^{n+1/2} = T^n + \frac{1}{2}\Delta t A\ T^n \quad \Longleftarrow \quad \text{Prediction for } t = (n+1/2)\Delta t \tag{4.56}$$

$$\ddot{T}^{n+1/2} = T^n + \frac{1}{2}\Delta t\ A\ \dot{T}^{n+1/2} \quad \Longleftarrow \quad \text{Correction for } t = (n+1/2)\Delta t \tag{4.57}$$

$$\dddot{T}^{n+1} = T^n + \Delta t A\ \ddot{T}^{n+1/2} \quad \Longleftarrow \quad \text{Prediction for } t = (n+1)\Delta t. \tag{4.58}$$

Note in Eq. (4.55) that the averaging technique used is the Simpson's rule. It means that the

correction gives more importance (more weight when averaging) to the estimates performed at times $t = (n + 1/2)\Delta t$, the times between the time step intervals, than at times $t = n\Delta t$ and $t = (n + 1)\Delta t$, i.e. the estimates at the actual intervals. It is also worth to note that Eqs. (4.56) and (4.57) are based on the FE and BE methods, respectively, whereas Eq. (4.58) is based on the midpoint rule. The method RK4 is fourth order accurate in time $\mathcal{O}(\Delta t)^4$.

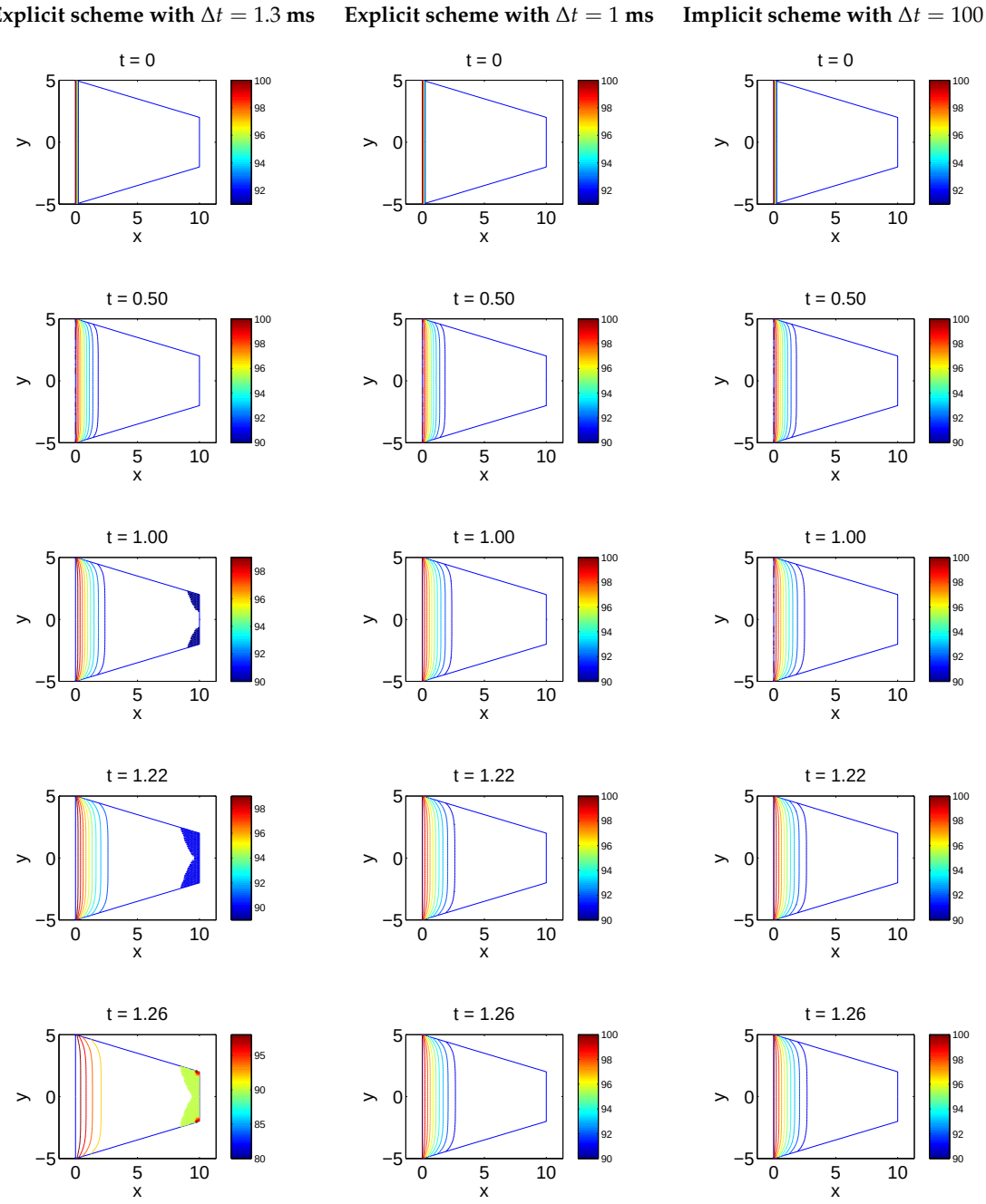


Figure 4.2: Temporal evolution of temperature field in contour plots. Time in seconds.

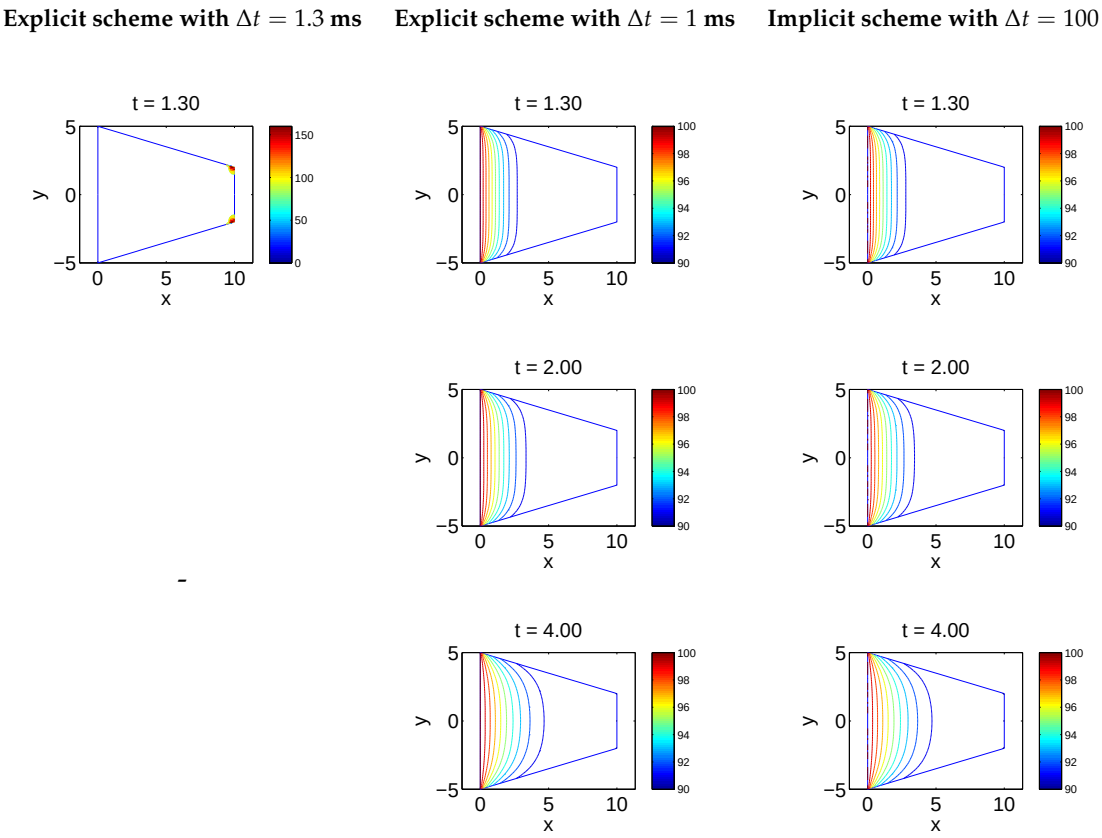


Figure 4.3: Temporal evolution of temperature field in contour plots. Time in seconds.

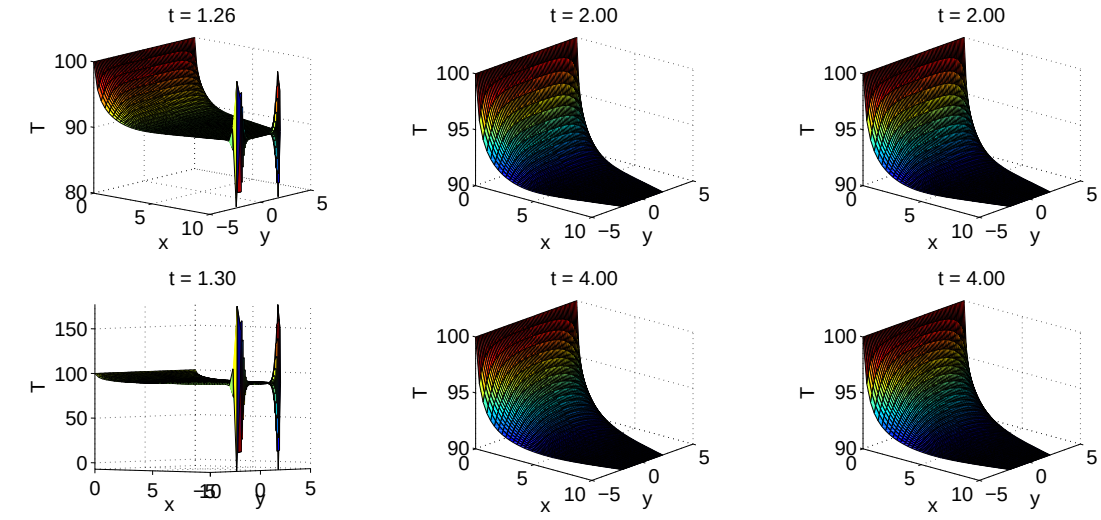


Figure 4.4: Temporal evolution of temperature field in surface plots. Time in seconds.

4.5 Exercises

Stage 1: Merging codes

- Each group is asked to discuss and choose one of the codes written in Session 04.

Stage 2: Coding temporal explicit scheme

- Add a case called 'unsteady' in solveFVM. In that way, the code will be able to perform both 'steady' and 'unsteady' computations.
- Discretize the time with an explicit scheme. Choose dirichlet Boundary conditions.
- Which should be the stability criterion to use?. Does it change if non rectangular geometries are considered?
- Impose a Neumann BC at the south and Robin BC at north and east. Is the stability criterion similar as previously ?

Stage 3: Coding temporal θ scheme

- Code the θ temporal scheme

Stage 3: Coding RK4 scheme (Optional)

- Code the RK4 scheme.
- Is there too much difference with respect to the explicit scheme?

5

Sparse Matrices and Linear Solvers

References

[XX1] MORTON, K. W. AND MAYERS, D. F. *Numerical Solution of Partial Differential Equations. 2nd editon, Chp. 7.* Cambridge University Press, 2005.

[XX2] SAAD, Y. *Iterative Methods for Sparse Linear Systems. 2nd edition*. ociety for Industrial and Applied Mathematics SIAM, 2003.

Objectives

- Get to know, what sparse matrices are and why they require less storage and less computational effort.
- Get an overview of sparse linear solvers.
- Understand preconditioning techniques.

Contents

As we have seen in the previous chapters, numerical problems derived from linear PDEs are reduced to a system of linear equations. As the system dimension grows, also does the computational time and the required storage. In this chapter, we discuss how the so-called sparse matrices, obtained after discretization, can be optimally stored so that computational efficiency increases. We will also observe how, for high dimension problems, iterative methods show better performance with respect to direct methods, provided that a suitable preconditioning is implemented. In this chapter we offer an overview of basic iterative methods based on the Richardson iteration. Projection methods, which are more complex approaches, will be also briefly described.

5.1 Sparse matrix

A linear system is usually described by

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{5.1}$$

where $\boldsymbol{x}$, $\boldsymbol{b} \in \mathbb{R}^n$ and $A = (a_{ij}) \in \mathbb{R}^{n\times n}$. As we can see from Figs. 5.1 and 5.2, a system matrix derived from PDE discretization contains generally mainly zero entries. Such matrices, populated primarily with zeros, are called *sparse matrices*. As we can observe in these figures, the ratio of non-zero elements to the possible number of entries n^2 further reduces with a growing number of nodes. This ratio is called the *matrix density*. This is an outcome of the discretization stencil which is used. For example the FD stencil in 2D (see Eqs. (2.39) and (2.42)) relates a node to its four neighbors such that in the linear system a maximum of five entries per line may occur. It should be clear then that, for high dimensions mainly, the storage of sparse matrices can be reduced significantly by storing only the nonzero entries. This means that only the values and the corresponding matrix position of each non-zero entry must be kept. By doing so, the

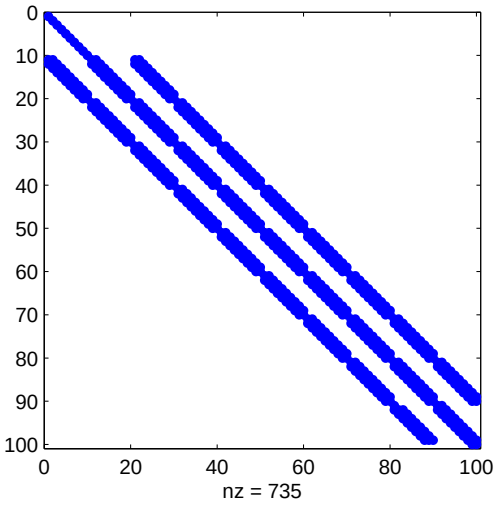


Figure 5.1: Non-zero pattern of a matrix derived from FV discretization with 100 nodes (density 0.0735).

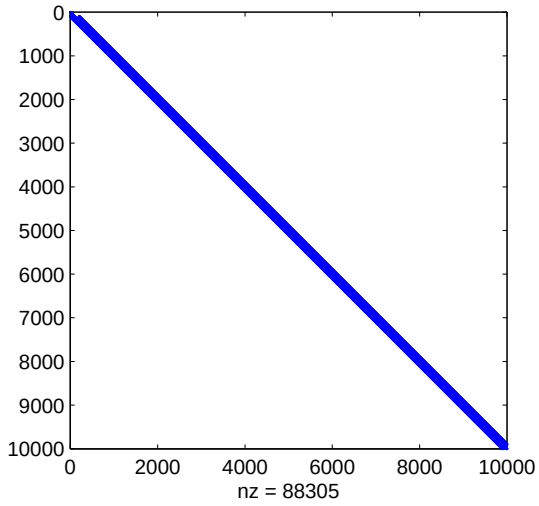


Figure 5.2: Non-zero pattern of a matrix derived from FV discretization with 10 000 nodes (density 8.8305e-04).

storage scales with the number of nonzero elements, which in turn scales with the number of nodes n. It is observed then that, when using the sparse format of storage, the storage scales in a linear way $\approx \mathcal{O}(n)$. This is, indeed, a much better situation with respect to the required storage in the full format, since it scales quadratically $\mathcal{O}(n^2)$. In Fig. 5.3 the required storage in MATLAB is plotted versus the number of unknowns n, where the dashed blue and solid red line correspond to the sparse and dense matrix, respectively.

Let us take now a look of what happens with the computational effort when either full format or sparse format is considered and a direct approach is used to solve the linear system. The computational effort for solving a dense linear system scales with $\mathcal{O}(n^3)$, when we use, say, the Gaussian elimination. Operations involving sparse matrices include many zero operations, and therefore they can be left out from the overall computation. As a result, if a matrix can be treated in a sparse representation, the computational effort can be reduced significantly. Figure 5.4 shows the computational time on Intel i7 CPU 3.4 GHz 4K/8T with 16GB Ram to solve a linear system with n number of unknowns. Despite of the more efficient computation, the numerical effort scale in most cases over-linearly ($\mathcal{O}(n^c)$ with $1 < c < 3$) when direct methods are involved to solve a linear system.

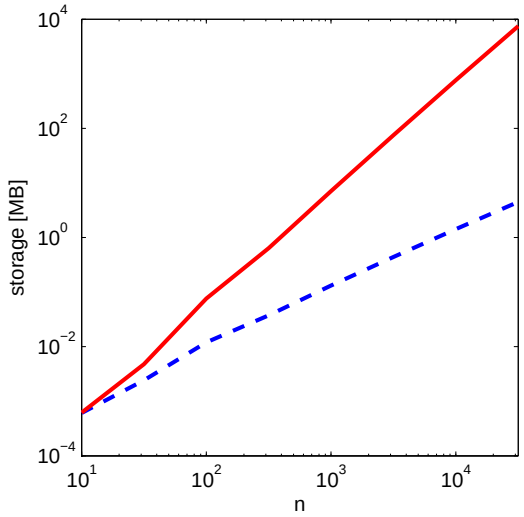


Figure 5.3: Storage in MATLAB for a n-dimensional sparse (dashed blue) and dense (solid red) system matrix form the FVM discretization.

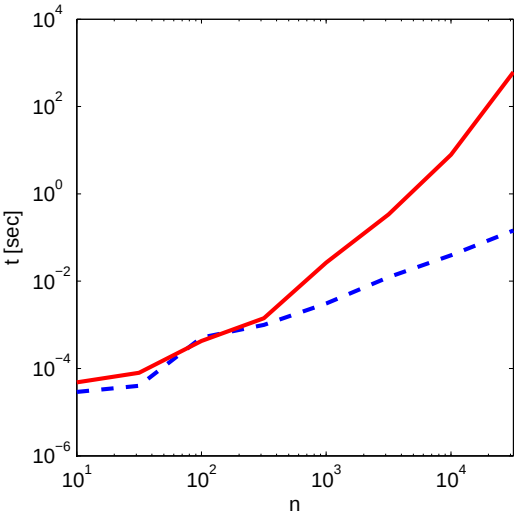


Figure 5.4: Compuational time in MATLAB to solve a n-dimensional sparse (dashed blue) and dense (solid red) system form the FVM discretization (on Intel i7 CPU 3.4 GHz 4K/8T, 16GB Ram).

5.2 Iterative solvers and preconditioning

In many applications, iterative solvers scale better than direct sparse solvers. The former can achieve a performance up to a quasi-linear behavior $\mathcal{O}(n \log n)$. The key for a good performance of such iterative solvers is a suitable preconditioning, a technique that we discuss further down in this section.

An iterative solver approximates the solution of the linear system step by step until a certain convergence criterion is satisfied. An indicator for a fast convergence is a small *condition number* $\kappa_2(A) = ||A||_2 \, ||A||_2^{-1}$. This number can be seen as a measurement of how much the solution $\boldsymbol{x}$ can change (how sensitive it is) for a small change in the right-hand side $\boldsymbol{b}$. The condition number is by definition greater or equal 1, being 1 for the identity matrix.

In practice, it is often not recommended to solve directly the original system (5.1) since the matrix A might be associated with a big value of κ_2 . Instead, a modified, so called *preconditioned*, system is solved in order to improve the conditioning of the problem. For left preconditioning, this is done by multiplying the system with a full range matrix $C^{-1} \in \mathbb{R}^{n \times n}$ as:

$$C^{-1} A \boldsymbol{x} = C^{-1} \boldsymbol{b}. \tag{5.2}$$

The matrix C is called *preconditioner*. The above system has the same solution as the original

system but, with an appropriate choice of the preconditioner, the iteration convergences much faster and the solver is hence more efficient. Usually, the matrix C^{-1} is not built explicitly, but instead C^{-1} is applied at every iteration step in an implicit way. Choosing A as preconditioner would be ideal, since $\kappa_2(A^{-1}A) = 1$. However, we would have to apply A^{-1}, which is practically the solution of the original problem. We choose then a preconditioner which is

- close to the original system matrix A
- cheap to apply.

5.3 Preconditioned Richardson iteration

A basic iterative solver is the *Richardson iteration*. Here at every iteration step m, the residual $\boldsymbol{b} - A\boldsymbol{x}$ is added to the current iteration values $\boldsymbol{x}^{(m)}$. Using preconditioning, the *preconditioned residual* $w^{(m)} = C^{-1}\left(\boldsymbol{b} - A\boldsymbol{x}^{(m)}\right)$ is added instead. Many iterative solvers are in fact such of a solver with a particular preconditioner. This method is shown in Algorithm 5.1.

Algorithm 5.1 Preconditioned Richardson iteration

Guess initial value $\boldsymbol{x}^{(0)}$
while not converged **do**
  $\boldsymbol{w}^{(m)} = C^{-1}\left(\boldsymbol{b} - A\boldsymbol{x}^{(m)})\right)$
  $\boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} + \boldsymbol{w}^{(m)}$
end while

We want to discuss when this iteration converges and therefore we rewrite the iteration as a fixed point iteration. Let x^* be the unique solution of Eq. (5.1) and $\boldsymbol{e}^{(m)} = \boldsymbol{x}^{(m)} - \boldsymbol{x}^*$ the error at the m-th iteration step. Furthermore, we define the *iteration matrix* $T = \left(I - C^{-1}A\right)$, where I denotes the n dimensional identity matrix. Then we get

$$\begin{aligned}
\boldsymbol{e}^{(m+1)} &= \boldsymbol{x}^{(m)} + C^{-1}\left(\boldsymbol{b} - A\boldsymbol{x}^{(m)}\right) - \boldsymbol{x}^* \\
&= \boldsymbol{e}^{(m)} + \underbrace{C^{-1}\boldsymbol{b}}_{=C^{-1}A\boldsymbol{x}^*} - C^{-1}A\boldsymbol{x}^{(m)} \\
&= (I - C^{-1}A)\boldsymbol{e}^{(m)} \\
&= T\boldsymbol{e}^{(m)} \\
&= T^k\boldsymbol{e}^{(0)}
\end{aligned} \tag{5.3}$$

Thus, it follows that the iteration convergences for every initial value $\boldsymbol{x}^{(0)}$ and every right-hand side $\boldsymbol{b}$, iff[1] $\lim_{m\to\infty} T^m = 0$. This is the case, iff all absolute values of the eigenvalues λ_j of T are

[1] "iff" is an common abbreviation for "if and only if".

less than 1. In terms of the *spectral radius* $\rho(T) \equiv \max_j \left|\lambda_j(T)\right|$, this criterion can be written as

$$\rho(T) < 1. \tag{5.4}$$

This means that the preconditioned Richardson iteration can also diverge! Only with a suitable preconditioning, it converges. The concrete procedure depends on the choice of the preconditioner C. Some classical solvers are listed in the following. Hereby, we split the matrix A into its diagonal D as well as into its strict lower part E and strict upper part F, so $A = D + E + F$.

- **Jacobi iteration**: Taking the diagonal D as preconditioner leads to the *Jacobi iteration*. We formulate this method in its classical pseudo code: Since D is diagonal, its diagonal values a_{ii} act only on the corresponding line i of the system. Accordingly, at the m-iteration step, we have to solve for the i-th line:

$$\begin{aligned} x_i^{(m+1)} &= x_i^{(m)} + \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{n} a_{ij}\, x_j^{(m)} \right) \\ &= \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij}\, x_j^{(m)} \right) \end{aligned} \tag{5.5}$$

  So when does this procedure convergence? We already know the answer, namely if $\rho\left(I - D^{-1}A\right) < 1$. Using the *Gershgorin circle theorem*, it can be shown that this criterion is satisfied for *diagonally dominant matrices*. Such a matrix is a matrix, where

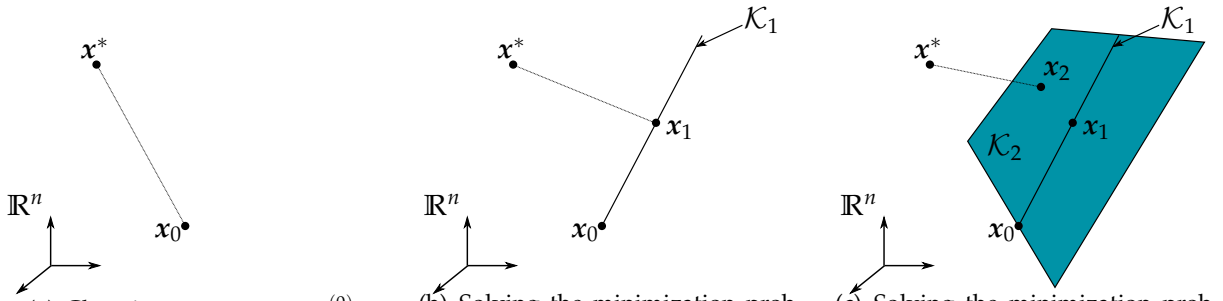
$$|a_{ii}| \geq \sum_{j \neq i} \left|a_{ij}\right| \qquad \forall i \tag{5.6}$$

  holds. Most matrices from discretization schemes satisfy this condition, e. g. for the FD scheme (see Eq. (2.42)) every diagonal entry is formed as the sum over the other entries in the same line. Other examples of methods which can be formulated as preconditioned Richardson iteration are:

- **Gauss-Seidel**: Here, not only the diagonal but also lower part is used for preconditioning. Thus, the preconditioner $C = (D + E)$. In comparison to the Jacobi iteration, this method converges mostly faster and more often but on the other hand is is a bit more laborious.

- **SOR**: The co-called *successive over relaxation* uses $C = \frac{1}{\omega}(D + \omega E)$ with the relaxation factor $\omega \in (0, 2)$. For large values of ω, the relaxation factor can speed up the convergence in comparison to the Gauss-Seidel method but it can also lead to a more unstable iteration. Small relaxation factors (from zero to one) can lead to a stable iteration, where Gauss-Seidel is unstable.

As we have seen previously, these methods do not converge for all linear systems. But similar to criterion (5.4), convergence can be guaranteed for many practical cases.



(a) Choosing a start vector $x^{(0)}$.

(b) Solving the minimization problem on $\mathcal{K}_1$.

(c) Solving the minimization problem on $\mathcal{K}_2$.

Figure 5.5: Sketch of the procedure of projection methods.

5.4 Projection methods

Powerful tools are the so called *projection methods*. Here, a brief description of two approaches is given: the *CG-method* (**c**onjugate **g**radient method) and *GMRES* (**g**eneralized **m**inimal **r**esidual **m**ethod). For details we refer to [XX2]. Both CG and GMRES, approximate the solution on a specific subspace, the so called Krylov space

$$\mathcal{K}_m = \boldsymbol{x}^{(0)} + \text{span}\{\boldsymbol{r}^{(0)}, A\boldsymbol{r}^{(0)}, ..., A^{m-1}\boldsymbol{r}^{(0)}\} \subset \mathbb{R}^n, \tag{5.7}$$

where $\boldsymbol{r}^{(0)} = \left(\boldsymbol{b} - A\boldsymbol{x}^{(0)}\right) \in \mathbb{R}^n$ denotes the initial residual vector. Now, at the m iteration step a minimization problem is solved on this subspace. For CG,

$$\boldsymbol{x}^{(m)} = \underset{\boldsymbol{x}\in\mathcal{K}_m}{\text{argmin}} \left(\frac{1}{2}\boldsymbol{x}^T C^{-1} A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}\right) \tag{5.8}$$

and for GMRES

$$\boldsymbol{x}^{(m)} = \underset{\boldsymbol{x}\in\mathcal{K}_m}{\text{argmin}} \left|\left|C^{-1}(\boldsymbol{b} - A\boldsymbol{x})\right|\right|_2. \tag{5.9}$$

If the residual is small enough, the procedure is stop, otherwise the $\mathcal{K}_m$ is extended to $\mathcal{K}_{(m+1)}$ and the minimization problem is solved on a larger space. This is repeated until the residual is below a certain boarder. For both problems, there exist very efficient implementations.

This iteration is illustrated in Fig. 5.5 where the exact solution is denoted as $\boldsymbol{x}^*$. In the first step, a initial value $\boldsymbol{x}^{(0)}$ (Fig. 5.5(a)) is set and a one-dimensional space $\mathcal{K}_1$ is constructed (Fig. 5.5(b)). Subsequently, the problem is projected on that subspace, i.e. the minimization problem is solved over the one-dimensional region. If the residual $\boldsymbol{r}_1 = \boldsymbol{b} - A\boldsymbol{x}_1$ is too large, we have to extend the space by one dimension, which leads to the 2D space $\mathcal{K}_2$ shown in (Fig. 5.5(c)). The the minimization problem is then solved on that space and we obtain $\boldsymbol{x}_2$. We decide to continue the procedure, if the residual $\boldsymbol{r}_2$ is still too large, otherwise we can stop the iteration. It can be shown that by further increasing the space dimension, the residual reduces or stays constant but never grows (a monotonical decay of the residual).

The CG-method can only be used for symmetrical matrices A (i. e. $A^T = A$) whereas GMRES can be used for all types of systems. With exact arithmetic [2], these projection methods converge for every linear system. Since a single iteration step is more expensive than for the Richardson iteration, the performance of projection methods highly depends on a suitable preconditioner. Moreover, as in the case of methods based on the Richardson iteration, these methods only show a good performance, if A is sparse!

[2] Note that the case of exact arithmetic is an ideal case. When computers are used, the machine error is introduced and, although these errors are very small (of the order of 2^{-52} for a machine with double precision), exact arithmetic cannot be argued.

5.5 Exercises

Stage 1: Unifying a code

- Each group is asked either to merge the codes or chose one code (those written in Session 04). The important point here is to spend some time sharing ideas about the way of coding.

Stage 2: Sparse Matrices (obtained from steady problems)

- Use the command *spy* to visualize the structure of your matrix.
- As observed in the lecture, storing and computing matrices in the sparse form leads to considerable improvements in computational efficiency. The first to do is then to 're-write' the code considering sparse matrices (indeed it resumes in expressing the matrix A in the sparse format).
- Compare computational time invested and computational storage of A when using either sparse or full formats. Use the commands *tic* and *toc* for time and *whos* for storage. Generate two figures similar to Figs. 5.4 and 5.3 (three different cases to generate those plots should be enough).

Stage 3: Iterative solvers 1

The backslash operator is a very efficient matlab routine to solve linear systems. This is due mainly to two reasons: 1) It uses different approaches depending on the structure of the matrix (square, diagonal, three-diagonal, symmetric, full , etc). 2) The selected approach is actually a set of Basic Linear Algebra Subprograms (BLAS). These subprograms or routines are already compiled (binary format) and therefore directly accessible for the computer. Sometimes, BLAS routines also are optimized to manage the way memory (RAM, Cache) works for a given computer.

Therefore, there is no point in comparing the backslash operator with 'self-made' matlab routines to solve linear systems. A comparison would be fair, at least, if pre-compiled version of the self-made routines are used so that our computers do not have to 'waste' time compiling: converting the very high-level language (Matlab) to machine language (binary).

What is possible is to compare self-made routines with self-made routines and, by doing so, to compare basic iterative methods for solving linear systems. At this point it is important to see the influence of preconditioning approaches and its influence in the convergence of the solution on problem

- Derive and code one algorithm for Jacobi, Gauss-Seidel and SOR, respectively.

- Create a random diagonal dominant matrix and a random vector b. Use Jacobi, Gauss-Seidel and SOR to solve the corresponding linear system.
- Create a random three-diagonal dominant matrix and a random vector b. Use Jacobi, Gauss-Seidel and SOR to solve the corresponding linear system.
- Create a full random matrix and a random vector b. Use Jacobi, Gauss-Seidel and SOR to solve the corresponding linear system.
- Plot residual Vs number of iterations (One figure for each case) for a 100×100 matrix.

note: use the command *rand* to create random arrays.

Stage 4: Iterative solvers 2

- Use Jacobi, Gauss-Seidel and SOR to solve the linear system given by the discretization of the 2D heat Equation (finite volumes)
- Plot residual Vs number of iterations

Stage 5: The wall of fame (or shame)

- Each group is asked to compute the time invested to solve a given problem (given in class) considering the best algorithm between Jacobi, Gauss-Seidel and SOR (time is taken just before and after solving the linear system). Afterwards the code should be given so that either Armin or I test it. We will put the final classification of names and times on a big poster: the wall of fame.
- The stop parameter should be defined as

$$\epsilon \leq \frac{\| b - Ax \|_2}{\| b \|_2} \tag{5.10}$$

  where $\| \cdot \|_2$ is recognized as the Euclidean norm and $\epsilon = 0.01$. Stop if more than 2000 iterations are needed.

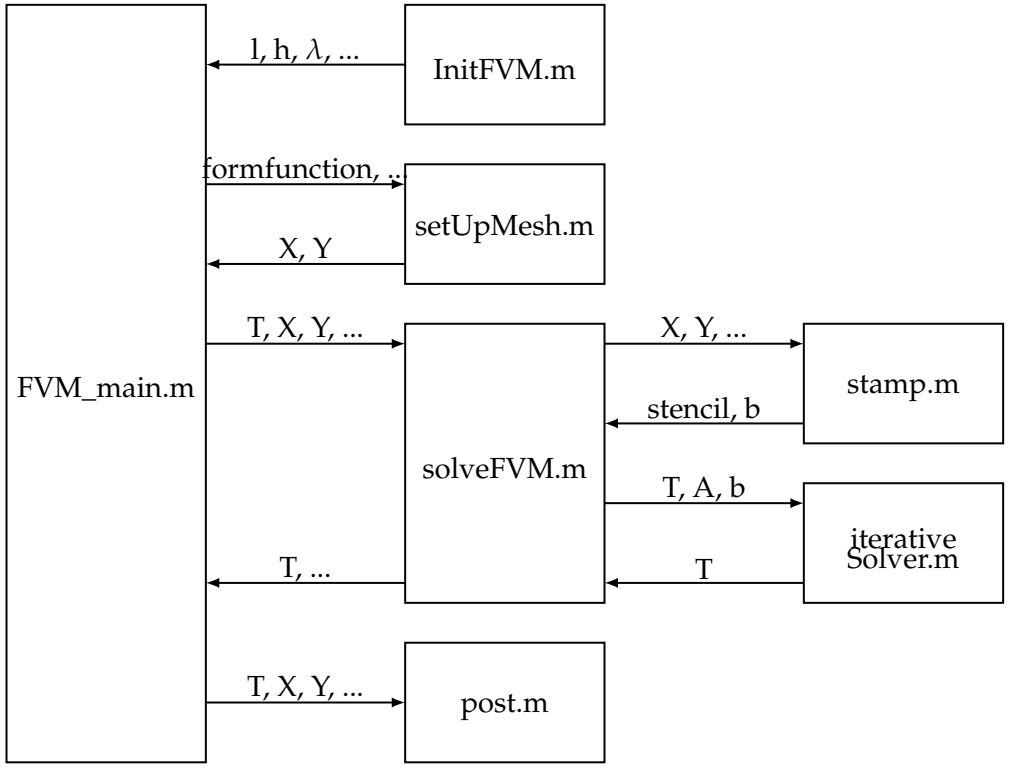
Stage 6: Very large matrices (optional)

- Use the best method between Jacobi, Gauss-Seidel and SOR to compare it with gmres (already implemented in matlab). Use different sizes (medium, large, very large) of matrices. Which one perform better?

5.5.1 Useful MATLAB commands

`max(eig(A))`	Gives the largest eigenvalue of matrix A which is equal to the spectral radius ρ of the matrix.
`spy(A)`	Plots the structure of non-zero elements of matrix A. Non-zero elements are depicted as a dot.
`tic` `toc`	the command `tic` starts a timer. The value of the timer is returned by calling `toc`.
`whos(A)`	This function gives you information about the memory used by the matrix (or variable) A.
`rand(n)`	Creates a dense $n \times n$ matrix filled with decimal values in the range $[0, 1]$.

5.5.2 Flowchart



6

Green's functions

References

[1] Kevin D. Cole, James V. Beck, A. Haiji-Sheik, and Bahman Litkouhi. *Heat conduction using Green's functions*. CRC Press, 2011.

Objectives

- Understand the great utility of Green's functions together with its corresponding advantages and disadvantages when solving partial differential equations.

Contents

So far, we have studied how to discretize the 2D heat equation in space and time. Space discretization was carried out by finite differences and finite volumes whereas time discretization was performed using both explicit and implicit schemes. Finite differences was applied in a rectangular geometry, whereas finite volumes was carried out on a non-Cartesian domain. Indeed, analytical solutions exist for solving the unsteady/steady heat equation on a rectangular topology and, as stated before, an analytical solution should be considered instead of a numerical approach if the analytical solution is both available and feasible. Generally, an analytical solution is derived for one specific partial differential equation with defined boundary conditions and is valid only for **that** case. Actually, if the partial differential equation remains the same and only small changes are applied to BC or to a given source, the previous analytical solution does not hold anymore and another analytical derivation must be performed. It would be very helpful, if the departing point of such derivations is the same for several different boundary and initial conditions. Indeed, such a procedure exists in the framework of Green's functions.

6.1 Green's function solution equation for the steady heat equation

The general solution of the heat equation in terms of Green's Function (GF) also comprises the unsteady version. Nevertheless, we will not consider variations in time in the solution. First of all, let us recall the steady heat equation:

$$\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + \frac{1}{\lambda}\dot{\omega}(x,y) = 0, \tag{6.1}$$

where $\dot{\omega}$ denotes a given source of heat placed at (x, y). Let us assume now that there is an *auxilary problem* defined by

$$\left(\frac{\partial^2 G}{\partial x^2}+\frac{\partial^2 G}{\partial y^2}\right)+\delta(\boldsymbol{x}-\boldsymbol{x_s})=0, \tag{6.2}$$

where $G(\boldsymbol{x}|\boldsymbol{x_s})$ represents a GF. This function should be read as 'the measurement of G at the observation point $\boldsymbol{x}=(x,y)$ due to a pulse δ produced at position $\boldsymbol{x_s}=(x_s,y_s)'$. The function δ represents the Dirac delta function which is mainly defined by its filter property:

$$\int_{-\infty}^{\infty} f(\boldsymbol{x})\delta(\boldsymbol{x}-\boldsymbol{x_s})\,d\boldsymbol{x}=f(\boldsymbol{x_s}) \tag{6.3}$$

for a given function $f(\boldsymbol{x})$. An important property of Green's functions is called 'reciprocity' and is expressed as

$$G(\boldsymbol{x}|\boldsymbol{x_s})=G(\boldsymbol{x_s}|\boldsymbol{x}). \tag{6.4}$$

Based on the reciprocity property, we can then express Eqs. (6.1) and (6.2) in terms of $\boldsymbol{x_s}$ as

$$\left(\frac{\partial^2 T}{\partial x_s^2}+\frac{\partial^2 T}{\partial y_s^2}\right)+\frac{1}{\lambda}\dot{\omega}(x_s,y_s)=0 \qquad \text{and} \qquad \left(\frac{\partial^2 G}{\partial x_s^2}+\frac{\partial G}{\partial y_s^2}\right)+\delta(\boldsymbol{x}-\boldsymbol{x_s})=0. \tag{6.5}$$

Subsequently, we multiply the first expression by G and the second expression by T and subtract the last from the former. The general equation results after integrating over the control area S containing the source.

$$\int_S G\left(\frac{\partial^2 T}{\partial x_s^2}+\frac{\partial^2 T}{\partial y_s^2}\right)\,d\boldsymbol{x_s}+\frac{1}{\lambda}\int_S G\,\dot{\omega}\,d\boldsymbol{x_s}-\int_S T\left(\frac{\partial^2 G}{\partial x_s^2}+\frac{\partial^2 G}{\partial y_s^2}\right)\,d\boldsymbol{x_s}-\int_S T\delta(\boldsymbol{x}-\boldsymbol{x_s})\,d\boldsymbol{x_s}=0. \tag{6.6}$$

Reordering Eq. (6.6) and knowing that $\int_S T(\boldsymbol{x_s})\delta(\boldsymbol{x}-\boldsymbol{x_s})\,d\boldsymbol{x_s}=T(\boldsymbol{x})$ results in

$$T(\boldsymbol{x})=\frac{1}{\lambda}\int_S G\,\dot{\omega}\,d\boldsymbol{x_s}+\int_S\left[G\left(\frac{\partial^2 T}{\partial x_s^2}+\frac{\partial^2 T}{\partial y_s^2}\right)-T\left(\frac{\partial^2 G}{\partial x_s^2}+\frac{\partial^2 G}{\partial y_s^2}\right)\right]\,d\boldsymbol{x_s}. \tag{6.7}$$

Leaving the first term on the RHS of Eq. (6.7) unchanged and applying Green's second identity on the second, leads to

$$T(\boldsymbol{x})=\frac{1}{\lambda}\int_S G\,\dot{\omega}\,d\boldsymbol{x_s}+\oint_{\partial S}\left[G\left(\frac{\partial T}{\partial x_s}+\frac{\partial T}{\partial y_s}\right)-T\left(\frac{\partial G}{\partial x_s}+\frac{\partial G}{\partial y_s}\right)\right]\cdot\boldsymbol{n}\,dl_s. \tag{6.8}$$

Equation (6.8) is the solution of the 2D steady heat equation. The first term on the RHS repre-

sents the contribution of the source term $\dot{\omega}$, whereas the second term accounts for the influence of boundary conditions on the final distribution of temperature $T(\boldsymbol{x})$. For the 1D case, Eq. (6.8) reduces to

$$T(x) = \frac{1}{\lambda}\int G\,\dot{\omega}\,dx_s + \left[G\frac{dT}{dx_s} - T\frac{dG}{dx_s}\right]_{\mathbb{W}}^{\mathbb{E}} \tag{6.9}$$

6.2 Treatment of boundary conditions

Equation (6.8) holds for any 2D domain. Nevertheless, Green functions are not simple to obtain (or do not exist) for complex geometries. We will consider, consequently, a 2D rectangular domain as shown in Fig. 6.1. For that case, the second term of Eq. (6.8) can be developed as follows:

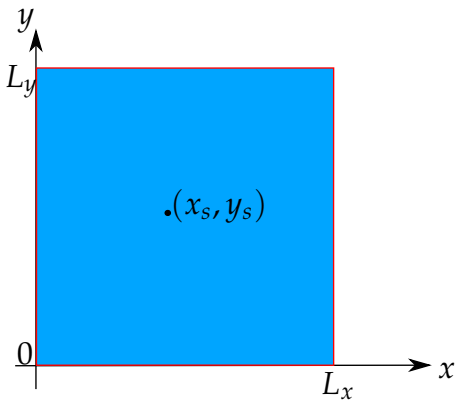


Figure 6.1: 2D rectangular domain.

$$\begin{aligned}\oint_{\partial S}\left[G\left(\frac{\partial T}{\partial x_s}+\frac{\partial T}{\partial y_s}\right)-T\left(\frac{\partial G}{\partial x_s}+\frac{\partial G}{\partial y_s}\right)\right]\cdot\boldsymbol{n}\,dl_s \\ = \int_{l^{\mathbb{E}}}\left(G\frac{\partial T}{\partial x_s}-T\frac{\partial G}{\partial x_s}\right)\cdot\boldsymbol{n}\,dy_s + \int_{l^{\mathbb{N}}}\left(G\frac{\partial T}{\partial y_s}-T\frac{\partial G}{\partial y_s}\right)\cdot\boldsymbol{n}\,dx_s \\ + \int_{l^{\mathbb{W}}}\left(G\frac{\partial T}{\partial x_s}-T\frac{\partial G}{\partial x_s}\right)\cdot\boldsymbol{n}\,dy_s + \int_{l^{\mathbb{S}}}\left(G\frac{\partial T}{\partial y_s}-T\frac{\partial G}{\partial y_s}\right)\cdot\boldsymbol{n}\,dx_s\end{aligned} \tag{6.10}$$

where $l^{\mathbb{E}}$, $l^{\mathbb{N}}$, $l^{\mathbb{W}}$ and $l^{\mathbb{S}}$ stand for the contour lines at east, north, west, and south, respectively. The boundary conditions for the auxiliary problem (Eq. (6.2)) are always homogeneous. Equations (6.11) to (6.13) list the definition of BC for both the actual problem and the auxiliary problem taking as an example the $\mathbb{W}$est or $\mathbb{E}$ast boundary.

- **Dirichlet**

$$T(L_x, y_s) = f(y_s) \qquad \text{and} \qquad G(L_x, y_s) = 0 \tag{6.11}$$

- **Neumann**

$$\frac{\partial T(L_x, y_s)}{\partial x_s} \cdot \boldsymbol{n} = \frac{\dot{q}(L_x, y_s)}{\lambda} \qquad \text{and} \qquad \frac{\partial G(L_x, y_s)}{\partial x_s} \cdot \boldsymbol{n} = 0 \tag{6.12}$$

- **Robin**

$$\lambda \frac{\partial T(L_x, y_s)}{\partial x_s} \cdot \boldsymbol{n} + \alpha \left[T(L_x, y_s) - T_\infty(L_x, y_s) \right] = 0 \qquad \text{and} \qquad \lambda \frac{\partial G(L_x, y_s)}{\partial x_s} \cdot \boldsymbol{n} + \alpha G(L_x, y_s) = 0 \tag{6.13}$$

Following the procedure introduced before, we multiply the BC of the actual problem by G and the BC of the auxiliary problem by T, and we substract the last from the former. It yields

- **Dirichlet**

$$TG - GT = G\, f \tag{6.14}$$

- **Neumann**

$$\left(G \frac{\partial T}{\partial x_s} - T \frac{\partial G}{\partial x_s} \right) \cdot \boldsymbol{n} = \frac{G\dot{q}}{\lambda}, \tag{6.15}$$

- **Robin**

$$\left(G \frac{\partial T}{\partial x_s} - T \frac{\partial G}{\partial x_s} \right) \cdot \boldsymbol{n} = \frac{\alpha G T_\infty}{\lambda} \tag{6.16}$$

where the dependence on L_x and y_s has been implicitly taken into account. Equation (6.14) is of non utility since it does not give any additional information with respect to what we already knew, namely that $G = 0$ at this boundary. On the contrary, Eqs. (6.15) and (6.16) clearly show an evaluation of the expression inside the integral of the first term at the RHS of Eq. (6.10). Let us now assume a problem in which:

- $\mathbb{E}$ast $\rightarrow$ Robin BC is applied
- $\mathbb{N}$orth $\rightarrow$ Robin BC is applied
- $\mathbb{W}$est $\rightarrow$ Dirichlet BC is applied
- $\mathbb{S}$outh $\rightarrow$ Neuman BC is applied.

The solution of this particular problem for the 2D heat equation, after replacing Eqs. (6.15) and Eq. (6.16) into Eq. (6.8), reads:

$$T(x,y) = \frac{1}{\lambda}\int_S G\ \dot{\omega}\ dy_s dx_s + \frac{1}{\lambda}\int_{l^{\mathbb{E}}} \alpha G T_\infty k dy_s + \frac{1}{\lambda}\int_{l^{\mathbb{N}}} \alpha G T_\infty dx_s + \int_{l^{\mathbb{W}}} T\frac{\partial G}{\partial x_s}\ dy_s + \frac{1}{\lambda}\int_{l^{\mathbb{S}}} G\dot{q}\ dx_s. \tag{6.17}$$

It is important to realize that all terms concerning Neumann or Robin BC will **always** be positive defined for any boundary. On the contrary, the term corresponding to Dirichlet BC will be positive for $\mathbb{S}$outh and $\mathbb{W}$ whereas it will be negative for $\mathbb{E}$ast and $\mathbb{N}$orth boundaries.

6.3 Derivation of the Green's function for a simple problem

Until now we have mentioned 'Green's functions' several times but they remain still somehow abstract. The idea of this section is to derive the GF for a simple configuration so that we get an idea of how they look like. At the end of this section some Green functions will be listed that correspond to the solution of the 1D heat equation for several types of boundary conditions. Let us then start by defining the 1D heat equation corresponding to the auxiliary problem. It reads as

$$\frac{d^2G(x|x_s)}{dx^2} + \delta(x - x_s) = 0 \qquad \text{for} \qquad 0 < x < L, \tag{6.18}$$

where x_s is associated with the position of the source and is placed somewhere between 0 and L. Let us now define the boundary conditions as follows:

$$G(0) = 0 \quad \text{at the inlet} \qquad \frac{dG(L)}{dx} = 0 \quad \text{at the outlet} \tag{6.19}$$

In order to remove the singularity of the differential equation induced by the $\delta-$source, the strategy is now to decompose the domain into two sub-domains:

$$\frac{d^2G_1(x|x_s)}{dx^2} = 0 \qquad \text{for} \qquad 0 < x < x_s \tag{6.20}$$

$$\frac{d^2G_2(x|x_s)}{dx^2} = 0 \qquad \text{for} \qquad x_s < x < L. \tag{6.21}$$

Now, we have two problems to solve. The solutions are easily obtained by integration, so that

$$G_1(x|x_s) = C_1 x + C_2 \qquad \text{for} \qquad 0 < x < x_s \tag{6.22}$$
$$G_2(x|x_s) = C_3 x + C_4 \qquad \text{for} \qquad x_s < x < L, \tag{6.23}$$

where C_1, C_2, C_3 and C_4 are constants to be defined by four different equations. Two of these equations are the boundary conditions defined in Eq. (6.19). The third equation comes from a continuity condition $G_1(x|x_s) = G_2(x|x_s)$ when $x = x_s$ so that

$$C_1 x_s + C_2 = C_3 x_s + C_4. \tag{6.24}$$

The last equation arises by integrating the 'missing' part of the problem, i. e. by integrating Eq. (6.18) in a region $x_s - \epsilon < x_s < x_s + \epsilon$, where ϵ is a very small number. This condition reads as

$$\int_{x_s-\epsilon}^{x_s+\epsilon} \lambda \frac{d^2 G(x|x_s)}{dx^2}\, dx = -\int_{x_s-\epsilon}^{x_s+\epsilon} \delta(x - x_s)\, dx. \tag{6.25}$$

This leads for $\epsilon \to 0$ to

$$\lambda \left.\frac{dG(x|x_s)}{dx}\right|_{x_s}^{x_s} = -1 \frac{dG_2(x|x_s)}{dx} - \frac{dG_1(x|x_s)}{dx} = -1.$$

Therefore, the fourth condition is expressed as

$$C_3 - C_1 = -1. \tag{6.26}$$

Resolving Eqs. (6.19), (6.24), and (6.26) yields: $C_1 = 1$, $C_2 = 0$, $C_3 = 0$ and $C_4 = x_s$. The Green's function for the problem defined by Eqs. (6.18) and (6.19) reads as

$$G(x|x_s) = \begin{cases} x & \text{for} \quad 0 < x < x_s \\ x_s & \text{for} \quad x_s < x < L \end{cases}. \tag{6.27}$$

In Table 6.1, we list some Green's functions of our interest. Note that the suffixes in the column 'Case' stand for (1) Dirichlet BC, (2) Neumann BC and (3) Robing BC.

Case	Boundary Conditions		Green Function	
	Inlet	Outlet	$0 < x < x_s$	$x_s < x < L$
X11	$G(0\|x_s) = 0$	$G(L\|x_s) = 0$	$x(1 - x_s/L)$	$x_s(1 - x/L)$
X12	$G(0\|x_s) = 0$	$\frac{dG(L\|x_s)}{dx} = 0$	x	x_s
X13	$G(0\|x_s) = 0$	$\lambda\frac{dG(L\|x_s)}{dx} + TG(L\|x_s) = 0$	$x[1 - B_2(x_s/L)/(1 + B_2)]$	$x_s[1 - B_2(x/L)/(1 + B_2)]$
X23	$\frac{dG(0\|x_s)}{dx} = 0$	$\lambda\frac{dG(L\|x_s)}{dx} + TG(L\|x_s) = 0$	$L(1 + 1/B_2 - x_s/L)$	$L(1 + 1/B_2 - x/L)$
X33	$\lambda\frac{dG(0\|x_s)}{dx} + TG(0\|x_s) = 0 \leftarrow$ Inlet $\lambda\frac{dG(L\|x_s)}{dx} + TG(L\|x_s) = 0 \leftarrow$ Outlet		$(B_1B_2x + B_1x - B_1B_2xx_s/L$ $-B_2x_s + B_2L + L)/C$	$(B_1B_2x_s + B_1x_s - B_1B_2xx_s/L$ $-B_2x + B_2L + L)/C$
where $B_1 = \alpha_1 L/\lambda$, $B_2 = \alpha_2 L/\lambda$ and $C = B_1B_2 + B_1 + B_2$				

Table 6.1: Some Green's function of interest that satisfy the 1D steady heat equation. Taken from [1].

6.4 What to integrate? (Warning)

Even for a simple case, such that one of Eq. (6.27), some confusion may be caused when integrating. Let us assume we want to integrate that 1D Green's function of Eq. (6.27) within the domain $0 < x < L$:

$$\int_0^L G(x|x_s)\, dx_s, \qquad \text{where} \qquad G(x|x_s) = \begin{cases} G_a = x & \text{for} \quad 0 < x < x_s \\ G_b = x_s & \text{for} \quad x_s < x < L \end{cases}. \tag{6.28}$$

Should we do

$$\underbrace{\int_{x_s=0}^{x_s=x} G_a\, dx_s + \int_{x_s=x}^{x_s=L} G_b\, dx_s}_{\text{Option 1}} \qquad \text{or} \qquad \underbrace{\int_{x_s=0}^{x_s=x} G_b\, dx_s + \int_{x_s=x}^{x_s=L} G_a\, dx_s}_{\text{Option 2}} \quad ? \tag{6.29}$$

Option 1 is incorrect, thus option 2 is the one we must choose. Indeed, the first integral correspond to the region $0 < x_s < x$ whereas the second integral lies between $x < x_s < L$ (note that the integration is performed with respect to x_s and not with respect to x). Accordingly, G_b corresponds to the first integral since $x_s < x$ and G_a with the second integral since $x < x_s$.

Case	Green's Function		
	Eigenfunction $X_m(x)$	Eigenvalue β_m, $m = 1, 2, ...$	Norm N_x
X11	$\sin(\beta_m x)$	$\beta_m = m\pi/L$,	$L/2$
X12	$\sin(\beta_m x)$	$\beta_m = (2m-1)\pi/2L$	$L/2$
X13	$\sin(\beta_m x)$	$\beta_m L\cot(\beta_m L) + B_2 = 0$	$L/(2\phi_{2m})$
X23	$\cos(\beta_m x)$	$\beta_m L\tan(\beta_m L) - B_2 = 0$	$L/(2\phi_{2m})$
X33	$\beta_m L\cos(\beta_m x)$ + $B_1\sin(\beta_m x)$	$\tan(\beta_m L)$ + $[\beta_m(\alpha_1+\alpha_2)/\lambda]/(\beta_m^2 - \alpha_1\alpha_2\lambda^{-2}) = 0$	$L/(2\phi_m)$
where $B_i = \alpha_i L/\lambda$, $\phi_{im} = (\beta_m^2 L^2 + B_i^2)/(\beta_m^2 L^2 + B_i^2 + B_i)$ and $\Phi_m = \phi_{2m}/(\beta_m^2 L^2 + B_1^2 + B_1\phi_{2m})$			

Table 6.2: Some Green's function of interest that satisfy the 1D steady heat equation. (Series version).

6.5 Green's functions for a rectangular domain

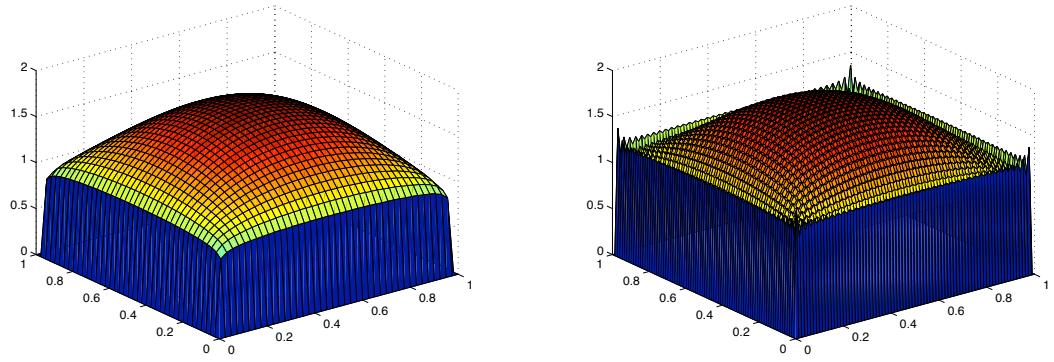
The Green's functions listed in Table 6.1 are very useful in the study of steady heat conduction in a one-dimensional domain. However, they are not unique: another family of Green's functions exist for 1D steady heat conduction. This family is based on a series expansion and is expressed as:

$$G(x|x_s) = \sum_{m=1}^{\infty} \frac{1}{\beta_m^2} \frac{X_m(x)X_m(x_s)}{N_x} \qquad \text{for} \qquad 0 < x < L_x, \tag{6.30}$$

where $X_m(x)$ and β_m denote a m^{th} eigenfunction of the system and a m^{th} eigenvalue of the system, respectively. N_x represents the norm of the m^{th} eigenfunction[1]. The eigenfunctions $X_m(x)$ and eigenvalues β_m for several boundary conditions are listed in table 6.30.

In practice, Green's functions expressed as series have an important disadvantage with respect to Green's functions expressed as polynomials (see Table 6.1): the solution based on the former is accurate as long as a sufficient number of terms is included in the sum. On the one hand, when considering homogeneous Dirichlet BC ($T = 0$), this is actually not a problem since no more than six terms are usually needed in order to get a sufficiently accurate solution. On the other hand, when non-homogenous Dirichlet BC make part of the problem, a convergence problem arise. In such a case, hundreds of terms are most of the time necessary to reach a 'fair' solution. Figure 6.2 shows an example of the difficulty of the series based Green's functions to obtain correct values of T close to the boundaries, in the case in which a temperature different from zero at the boundaries is imposed. This difficulty is highly visible at points very close to Dirichlet boundaries and is the reason why a more refined discretization may perform worse

[1] $X_m(x)$ and β_m are called eigenfunction and eigenvalue, respectively, because they satisfy $\frac{d^2X_m(x)}{dx^2} + \beta_m X(x) = 0$.



(a) (b)

Figure 6.2: Illustration of Gibbs phenomenon with Green's function series of 80 terms. (a) $\Delta x, \Delta y = 0.025$ (b) $\Delta x, \Delta y = 0.0125$.

than a coarse one. Indeed, this kind of problem is recognized as the Gibbs phenomenon.

Although GFs based on series do not behave as well as GFs based on polynomials, they are of relevance since Green's function formulations for 2D and 3D Cartesian geometries are most of the time based on them. A Green's function associated to the steady heat equation in a rectangular domain must satisfy

$$\frac{\partial^2 G_{2D}(x,y|x_s,y_s)}{\partial x^2} + \frac{\partial^2 G_{2D}(x,y|x_s,y_s)}{\partial y^2} + \delta(x-x_s)\delta(y-y_s) = 0 \qquad \text{for} \qquad 0 < x < Lx;\ 0 < y < Ly. \tag{6.31}$$

By combining two 1D Green's functions (Eq. (6.30) for x and y directions), a Green's function for a rectangular domain is constructed as

$$G_{2D}(x,y|x_s,y_s) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{1}{\beta_m^2 + \theta_n^2} \frac{X_m(x)X_m(x_s)}{N_x} \frac{Y_n(y)Y_n(y_s)}{N_y} \qquad \text{for} \qquad 0 < x < Lx;\ 0 < y < Ly, \tag{6.32}$$

where $Y_n(y)$ and θ_n denote a n^{th} eigenfunction and a n^{th} eigenvalue, respectively. N_y represents the norm of the n^{th} eigenfunction $Y_n(y)$. Values for $Y_n(y)$, θ_n and N_y are found in Table 6.2 by replacing x by y and m by n.

6.6 Discussion

It might be dissapointing that an analytical solution, such as the one expressed in Eq. (6.32), remains difficult to compute, specially for non-homogeneous Dirichlet boundary conditions. One may question then the utility of such an approach and, instead, prefer to solve the 2D heat equation by finite differences, for instance. Nevertheless, it is important to highlight the insight that the analytical solution based on Green's functions (see Eqs. (6.8) and (6.17)) give us. The first remarkable feature of a solution of the heat equation is that the distribution of temperature in the domain under study can be expressed as a superposition of solutions that correspond to both a given source and the boundary conditions of the system. A second remarkable feature is that, in contrast to the integrals associated to BC (see (6.17)), the integral associated to a source is very easily computed using a GF, as the one of Eq. (6.32), since it converges very fast. Moreover, since the integral is only performed over the domain in which the source is located, a very accurate computation may result if highly refining only the source domain. This is a much simpler approach than applying finite differences with a non-uniform mesh, so that only mesh refinement is performed in the region of the source. Since the final objective is to solve the heat equation (in 2D for our particular case) as accurately and computationally cheap as possible, a good idea is to combine numerical methods (finite differences, finite volumes or finite elements) with Green's functions. Whereas the former are used to compute the temperature distribution due to boundary conditions, the later is implemented to study the effect of heat sources within the domain. It is interesting to note that a complete study of the influence of heat sources on a given thermal system can be performed very efficiently by computing only once the temperature field due to BC, and then GF as many times as desired for the heat sources of interest.

6.7 Exercises

Stage 1: Numerical integration

Before implementing Green's functions, it is useful first to refresh our minds about numerical integration with the matlab function *trapz*

- Perform

$$\int_0^2 x \, \mathrm{d}x \tag{6.33}$$

  Note: The result should be a scalar.

- Perform and plot the corresponding result

$$\int_0^2 \int_0^2 \cos(x_s)\sin(y_s)\,\mathrm{d}x_s\mathrm{d}y_s \tag{6.34}$$

Note: the result should be a scalar

Stage 2: 2D Green's functions

For this stage, the GF of table 6.2 are considered. Compute the solution of the 2D steady heat equation based on Eq. (6.8) with the Green's function given by Eq. (6.32)

Define three points (x, y) for the observer and

- Consider a source distributed uniformly allong the domain. Take into account
  - No Boundary conditions (take the GF as if there were Dirichlet boundaries)
  - No Boundary conditions (take GF corresponding to Neumann BC for both $\mathbb{N}$orth and $\mathbb{E}$ast and Dirichlet BC on $\mathbb{W}$est and $\mathbb{S}$outh.)
- Consider a source distributed uniformly but only in a small region of the domain. Here it is possible to create a mesh for the source entouring only the region of the source. Use
  - No Boundary conditions (take the GF as if there were Dirichlet boundaries)
  - No Boundary conditions (take GF corresponding to Neumann BC for both $\mathbb{N}$orth and $\mathbb{E}$ast and Dirichlet BC on $\mathbb{W}$est and $\mathbb{S}$outh.)

Define a whole field (x, y) for the observer and

- Consider a source distributed uniformly allong the domain. Take into account
  - No Boundary conditions (take the GF as if there were Dirichlet boundaries)
  - No Boundary conditions (take GF corresponding to Neumann BC for both $\mathbb{N}$orth and $\mathbb{E}$ast and Dirichlet BC on $\mathbb{W}$est and $\mathbb{S}$outh.)
- Consider a source distributed uniformly but only in a small region of the domain. Here it is possible to create a mesh for the source entouring only the region of the source. Use
  - No Boundary conditions (take the GF as if there were Dirichlet boundaries)
  - No Boundary conditions (take GF corresponding to Neumann BC for both $\mathbb{N}$orth and $\mathbb{E}$ast and Dirichlet BC on $\mathbb{W}$est and $\mathbb{S}$outh.)

Stage 3: 2D Green's functions and FD (or FV)

Green's functions expressed as series can perform poorly when non-homogeneous Dirichlet BC are applied. Remember that one of the most important 'take away' ideas of solutions based on

GF is that these can be explained as a superposition (contribution) of source and boundaries. Therefore

- Compute a given problem in Finite Differences for a given set of boundary conditions.
- Compute one set up of that domain for several distribution of sources using Green functions.
- Compute the final solution by adding the contribution of BC computed by FD with the contribution of the source given by GF.

Stage 4: 2D Green's functions: Robin BC (optional)

From table 6.2 it is observed that the eigenvalues β_m for Robin BC are not given explicitly. Indeed, the values of β_m depend on the values of α and k and should be found by solving the equation shown in table 6.2 (column 2) for each m. Consequently, applying Robin BC by the approach of Green's functions might become really expensive.

- Use Robin BC for both $\mathbb{N}$orth and $\mathbb{E}$ast and Dirichlet BC on $\mathbb{W}$est and $\mathbb{S}$outh.

6.7.1 Useful MATLAB commands

`trapz(X,Y)`	Computes the integral of Y with respect to X using trapezoidal integration

7 Optimization

References

[1] VENKATARAMAN P. *Applied Optimization with MATLAB programming*. John Wiley and Sons, 2009.

[2] ULBRICH M. *Grundlagen der Nichtlinearen Optimierung*. Zentrum Mathematik, Technische Universität München. 2009

[3] POLIFKE, W., AND KOPITZ, J. *Wärmeübertragung. Grundlagen, analytische und numerische Methoden*. Pearson Studium, 2005.

[4] SCHMIDT, E. *Die Wärmeübertragung durch Rippen*. Z. des VDI, 70(26) 1926.

Objectives

- Understand the utility of combining optimization tools, such as the ones found in matlab, with discretizaton schemes of a given PDE.

Contents

How does the optimal cooling fin look like? which are the optimal operating conditions for a given wing to maximize lift and minimize drag? Finding an optimal configuration is an issue that arise in many engineering applications. Most of the time, such problems cannot be solved on paper. In this chapter, we have a look on how such optimization problems can be mathematically formulated and numerically solved.

7.1 Statement of the problem

The statement of the problem starts by defining an objective function z[1]:

$$z = f(\boldsymbol{x}) \tag{7.1}$$

which depends on n scalar variables $\boldsymbol{x} = (x_1, x_2 \cdots x_n)$. It is natural also that the variables x satisfy some given constrains:

$$g_i(\boldsymbol{x}) \leq, = \text{ or }, \geq b_i \tag{7.2}$$

It should be noted that the number of inequality constraints must not be greater than the number of variables since, in such a case, the problem would be overspecified. In the following we describe some of the most common particular problems of optimization.

- **General constrained optimization problem:** This is indeed the more general problem in optimization. Both the objective function and the constrains may be non-linear and the constraints can be associated with inequalities.

[1]The objective function is also known with other names such as cost function, energy function, utility function, etc.

- **Classical optimization problem:** Both the objection function and the constrains may be non-linear and the constrains are *only* associated with equations (no inequalities).
  - *Linear programming problem:* Both the objective function and the constrains are linear.
  - *Non-linear programming problem:* At least one constrain or the objective function is non-linear
  - *Quadratic programming problem:* The objective function is quadratic whereas the constrains are linear.
  - *General unconstrained problem:* The objective function may be non-linear. There are no constrains.

An 'optimal' method to solve all these problems does not exist. Instead, there is a very large range of methods specialized in eache of these subproblems.

In most of the cases, a general constrained optimization problem can be stated as a classical optimization problem. For doing so, the inequalities should be 'transformed' to equations by introducing the so called *slack* and *surplus* variables. For example, the inequalities

$$g_1(\boldsymbol{x}) \leq b_1, \qquad g_2(\boldsymbol{x}) \geq b_2 \tag{7.3}$$

are equivalent to

$$g_1(\boldsymbol{x}) + x_{n+1} = b_1, \qquad g_2(\boldsymbol{x}) - x_{n+2} = b_2 \quad \text{provided that} \quad x_{n+1}, x_{n+1} \geq 0, \tag{7.4}$$

where x_{n+1} and x_{n+2} are the slack variable and the surplus variable, respectively. Now that we have seen that almost all problems can be treated as classical optimization problems, we can then classify the corresponding methods (algorithms) in three:

- The most simple algorithms are based on the so called search methods. In this case, only the evaluation of the objective function at certain points is necessary. Although each iteration of one of these algorithms is generally computationally cheap, the number of iterations needed to find a reliable solution is sometimes very high.
- When the estimation of the first order derivative of the objective function can be computed or, at least, modelled for a defined region, methods may consider this additional information to compute the optimal solution. Convergence is generally improved with respect to search methods. Popular methods in this group are Quasi-Newton methods, interior point methods and gradient descent methods
- If the second order derivative of the objective function (called Hessian in technical language) can be assessed, at least in a approximated way, highly reliable methods can be

implemented. Although these approaches are computationally expensive per iteration, ideally not too many iterations should be necessary to compute the optimal solution. The Sequential Quadratic Programming (SQP), a Newton's based method, is a representative approach of this group.

The previous methods can be applied for both constrained and unconstrained problems. Being the later much easier to solve, a good idea is to transform constrained problems into unconstrained ones. Approaches for that purpose are commonly base on the so called *Lagrange multipliers*. Before presenting an overview of of two popular methods in optimization such as gradient descent and the Newton's method, classical definitions of both *necessary* and *sufficient* conditions for optimality are exposed.

7.2 Formulation and Optimality Condition

The *unrestricted optimization problem* is usually written as a minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}), \tag{7.5}$$

where f is the so-called *objective function* mapping the parameter space $\mathbb{R}^n$ on a scalar value. Unrestricted means, that there are no other constrains which have to be satisfied. We want to point out, that every optimization problem can be formulated as a minimization problem, since

$$\max_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^n} -f(\boldsymbol{x}). \tag{7.6}$$

As a result, the task is to find a minimum of f. We shortly recall that there are *local* and *global* minima. If there is an area $B_\epsilon(\bar{\boldsymbol{x}})$ around a point $\bar{\boldsymbol{x}}$, where

$$f(\boldsymbol{x}) \geq f(\bar{\boldsymbol{x}}) \tag{7.7}$$

holds, $\bar{\boldsymbol{x}}$ is called local minimum. If there is further more no other local minimum with a lower function value, $\bar{\boldsymbol{x}}$ is called global minimum. Unfortunately, in most cases it is only possible to determine local minima, but not global ones. Thus, we always have to keep in mind, that there might be another, even better minimum.

If the objective function f is continuously differentiable, a **necessary** condition for a local minimum $\bar{\boldsymbol{x}}$ is, that the gradient of f is zero, in formula:

$$\nabla f(\bar{\boldsymbol{x}}) = 0. \tag{7.8}$$

This is reasonable, when we keep in mind, that $-\nabla f(\boldsymbol{x})$ is a vector pointing in the direction of the steepest descent. At a minimum, there cannot be a decent in any direction. If f is twice continuously differentiable, criterion (7.8) gets **sufficient**, when additionally the *Hessian*

$\boldsymbol{H}(\bar{\boldsymbol{x}}) = \nabla^2 f(\bar{\boldsymbol{x}})$ is positive definite. This means, that

$$d^T \nabla^2 f(\bar{\boldsymbol{x}}) d > 0 \qquad \forall d \in \mathbb{R}^n \backslash \{0\}. \tag{7.9}$$

7.3 Gradient Descent

The *gradient descent* is a first-order optimization algorithm. The idea is to approach the minimum iteratively. At every iteration step k, we go in the direction of the steepest decent, i. e. in the direction of the negative gradient $-\nabla f(\boldsymbol{x}^k)$. Subsequently, we have to decide how far we go in that direction, e. g. we search on a line for the step size $\delta_k > 0$ which in best case satisfies

$$\min_{\alpha_k} f\left(\boldsymbol{x}^k - \delta_k \nabla f(\boldsymbol{x}^k)\right). \tag{7.10}$$

A way to approximate this step size is e. g. the *Armijo rule*. So, the algorithm reads as: Figure 7.1

Algorithm 7.1 Gradient Descent

Guess initial value $\boldsymbol{x}^{(0)}$
while not converged **do**
  Calculate a suitable step size δ_k
  $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \delta_k \nabla f(\boldsymbol{x}^k)$
end while

illustrates the gradient method.

7.4 Newton's Method

The purpose of Newton's method is to find a better direction than the one given by the gradient descent approach in order to minimize the number of iterations needed to find an optimal solution. As stated at the beginning of this chapter, the Newton's method is, per iteration, computationally expensive. This is due to the fact that this method requires an estimate of the Hessian $\boldsymbol{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$ in addition to first order derivatives of $f(\boldsymbol{x})$. Let us first expand $f(\boldsymbol{x}^k + \Delta\boldsymbol{x})$ at iteration k by means of Taylor series:

$$f(\boldsymbol{x}^k + \Delta\boldsymbol{x}) = f(\boldsymbol{x}^k) + \Delta\boldsymbol{x} \nabla f(\boldsymbol{x}^k) \tag{7.11}$$

where $\nabla f(\boldsymbol{x}^k)$ should be recognized as the *Jacobian* of f. In the same way, we are allowed to expand $\nabla f(\boldsymbol{x}^k + \Delta\boldsymbol{x})$ as

$$\nabla f(\boldsymbol{x}^k + \Delta\boldsymbol{x}) = \nabla f(\boldsymbol{x}^k) + \Delta\boldsymbol{x} \nabla^2 f(\boldsymbol{x}^k) \tag{7.12}$$

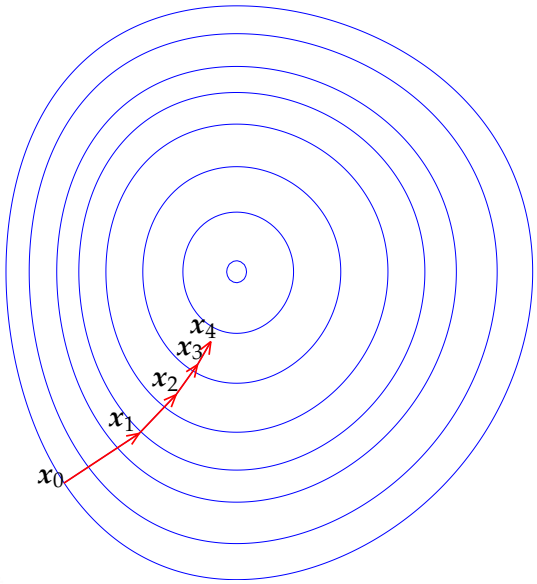


Figure 7.1: Illustration of gradient descent with step size proportional to the gradient.

Since we know that that a necessary condition for optimality is that the function gradient at the optimal point $\bar{\boldsymbol{x}}$ is zero, we assume then that

$$\nabla f(\boldsymbol{x}^{k+1}) = \nabla f(\boldsymbol{x}^k) + \Delta \boldsymbol{x} \nabla^2 f(\boldsymbol{x}^k) = 0 \tag{7.13}$$

where $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \Delta \boldsymbol{x}$. As a result, the optimal direction is given by

$$\Delta \boldsymbol{x} = -\boldsymbol{H}(\boldsymbol{x}^k)^{-1} \nabla f(\boldsymbol{x}^k). \tag{7.14}$$

We can now summarize the Newton's method with Algo. 7.4.

Algorithm 7.2 Newton Method for Optimization Problems

Guess initial value $\boldsymbol{x}^0$
while not converged **do**
  $\Delta \boldsymbol{x} = -\boldsymbol{H}(\boldsymbol{x}^k)^{-1} \nabla f(\boldsymbol{x}^k)$
  $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \Delta \boldsymbol{x}$
end while

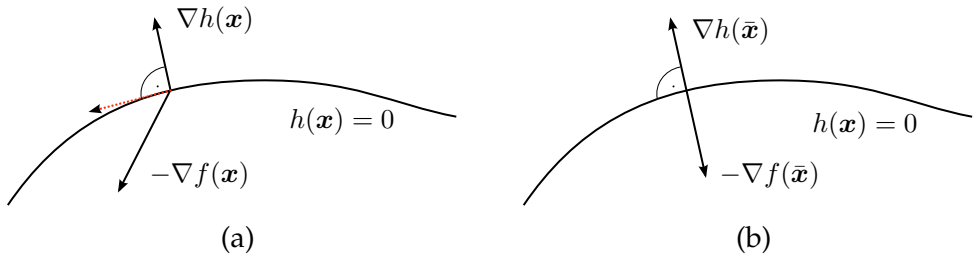


Figure 7.2: Ilustration of Lagrange Multiplier. (a) Case 1: The point $\boldsymbol{x}$ is not located at a minimum. Following the dashed arrow, the value of objective function can be decreased without hurting the constraint. (b) Case 2: The point $\bar{\boldsymbol{x}}$ is located at a minimum. There is a Lagrange multiplier λ with: $\nabla f(\bar{\boldsymbol{x}}) = \lambda \nabla h(\bar{\boldsymbol{x}})$

7.5 Constrained optimization: the method of Lagrange multipliers

Let us now consider a very popular method in constrained optimization. It is very popular because, under a rather simple analysis, a constrained problem is transformed into an unconstrained one. Once the problem is stated as an unconstrained one, all the artillery for unconstrained problems can be applied. Further on, this method is generalized by the Karush-Kuhn-Tucker conditions, which are recognized as necessary conditions for non-linear programming under both equality and inequality constrains. Since in general inequality constrains can be transform into equations (see Eq. (7.4)), we consider here only problems with equality constrains. The problem reads:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \text{subject to } h(\boldsymbol{x}) = 0. \tag{7.15}$$

With constrains, we cannot search minima in the whole space $\mathbb{R}^n$ but only in the restricted area $H = \{\boldsymbol{x} \mid h(\boldsymbol{x})\}$. But what is a necessary condition for minimum in that case? To answer that question, we can take a look at Fig. 7.2(a) where the situation for one constraint h is illustrated. Here, the point $\bar{\boldsymbol{x}}$ is not a minimum, since the value of f can be decreased when we follow the dashed arrow and still we stay on H. Considering now the case of Fig. 7.2(b), we see that the value of f can be decreased by going in the direction given by the negative gradient $-\nabla f(\bar{\boldsymbol{x}})$. Nevertheless, following that direction is not possible since we would violate the constraint h. Actually, we observe that this particular situation occurs when

$$\nabla f(\bar{\boldsymbol{x}}) = \lambda \nabla h(\bar{\boldsymbol{x}}) \tag{7.16}$$

holds. The variable λ is recognized as a *Lagrange multiplier*. This concept can be extended to the case of several constraints: At a minimum, the gradient $\nabla f(\bar{\boldsymbol{x}})$ can be constructed as a linear

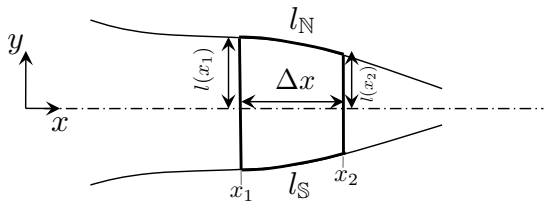


Figure 7.3: Quasi one-dimensional fin.

combination of gradients of the equality constraints ∇h_i $(i = 1, 2, \ldots, p)$. This means that there is a vector of Lagrange multipliers $\boldsymbol{\lambda} = \left(\lambda_1, \lambda_2, \ldots, \lambda_p\right)^T$ for which

$$\nabla f(\bar{\boldsymbol{x}}) + \sum_{i=1}^{p} \lambda_i \nabla h_i(\bar{\boldsymbol{x}}) = 0. \tag{7.17}$$

is satisfied. By means of the Lagrange multiplier, we can reformulate the equality restricted optimization problem (7.15) into the unrestricted one

$$\min_{\tilde{\boldsymbol{x}}} F(\tilde{\boldsymbol{x}}). \tag{7.18}$$

The new variable is $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{\lambda})$ with the objective function $F(\boldsymbol{x}, \boldsymbol{\lambda}) = f + \sum_{i=1}^{p} \lambda_i h_i(\bar{\boldsymbol{x}})$. A minimum of system (7.18), $\nabla_{\tilde{\boldsymbol{x}}} F(\bar{\boldsymbol{x}}, \boldsymbol{\lambda}) = \nabla f(\bar{\boldsymbol{x}}) + \sum_{i=1}^{p} \lambda_i \nabla h_i(\bar{\boldsymbol{x}}) = 0$ holds, which is in fact the optimality condition (7.17).

7.6 Quasi-1D approximation of a fin

In the following derivation, it is shown how heat transfer in a quasi-1D dimensional body can be described by a compact expression of the heat equation. A solution for such an expression can be easily obtained by integration. Moreover, an equation for the optimal shape can be easily derived based on the knowledge of the optimal temperature profile.

Fig. 7.3 shows an schematic description of a quasi 1D body, i.e. a body whose temperature is assumed to vary only along the longitudinal (x) axis. These assumption is indeed reasonable for bodies characterized by a low Biot number Bi$= \alpha l / \lambda$, or in other words, for bodies whose thermal conductivity λ is much higher than the convective heat transfer coefficient α of the surrounding medium. Thanks to this property, which is generally encountered in thin fins, high levels of heat transfer can be achieved. Let us start by integrating the homogeneous heat equation over a surface S and subsequently apply the divergence theorem:

$$\int_S \nabla^2 T \, dS = \oint_{\partial S} \nabla T \cdot \mathbf{n} \, \mathrm{d}l = 0. \tag{7.19}$$

We split now the line integral in four contributions:

$$\int_{-l(x_1)}^{l(x_1)} \nabla T \cdot \mathbf{n} \, \mathrm{d}y + \int_{-l(x_1)}^{l(x_1)} \nabla T \cdot \mathbf{n} \, \mathrm{d}y + \int_{l_{\mathbb{N}}} \nabla T \cdot \mathbf{n} \, \mathrm{d}l_{\mathbb{N}} + \int_{l_{\mathbb{S}}} \nabla T \cdot \mathbf{n} \, \mathrm{d}l_{\mathbb{S}} = 0. \tag{7.20}$$

Since the faces $l(x_1)$ and $l(x_2)$ are aligned with the y axis, Eq. (7.20) is simplified to

$$-\int_{-l(x_1)}^{l(x_1)} \frac{\partial T}{\partial x} \, \mathrm{d}y + \int_{-l(x_1)}^{l(x_1)} \frac{\partial T}{\partial x} \, \mathrm{d}y + \int_{l_{\mathbb{N}}} \nabla T \cdot \mathbf{n} \, \mathrm{d}l_{\mathbb{N}} + \int_{l_{\mathbb{S}}} \nabla T \cdot \mathbf{n} \, \mathrm{d}l_{\mathbb{S}} = 0. \tag{7.21}$$

Now it is assummed that the temperature T is constant along the y axis. For that reason it is possible to state that

$$\frac{\partial T}{\partial x} 2l(x) = \int_{-l(x)}^{l(x)} \frac{\partial T}{\partial x} \, \mathrm{d}y, \tag{7.22}$$

and Eq. (7.21) becomes

$$\frac{\partial T(x_1)}{\partial x} 2l(x_2) - \frac{\partial T(x_2)}{\partial x} 2l(x_1) + \int_{l_{\mathbb{N}}} \nabla T \cdot \mathbf{n} \, dl_{\mathbb{N}} + \int_{l_{\mathbb{S}}} \nabla T \cdot \mathbf{n} \, dl_{\mathbb{S}} = 0 \tag{7.23}$$

Subsequently, Eq. (7.23) is divided by Δx

$$\frac{\frac{\partial T(x_1)}{\partial x} 2l(x_2) - \frac{\partial T(x_2)}{\partial x} 2l(x_1)}{\Delta x} + \frac{1}{\Delta x} \int_{l_{\mathbb{N}}} \nabla T \cdot \mathbf{n} \, dl_{\mathbb{N}} + \frac{1}{\Delta x} \int_{l_{\mathbb{S}}} \nabla T \cdot \mathbf{n} \, dl_{\mathbb{S}} = 0 \tag{7.24}$$

Finally, we consider the limit case in which $\Delta x \to dx$. It implies also that $l_{\mathbb{N}}, l_{\mathbb{S}} \to dl_{\mathbb{N}}, dl_{\mathbb{S}}$. We have then

$$\frac{\partial}{\partial x} \left(2l(x) \frac{\partial T(x)}{\partial x} \right) + \nabla T \cdot \mathbf{n} \, \frac{dl_{\mathbb{N}}}{dx} + \nabla T \cdot \mathbf{n} \, \frac{dl_{\mathbb{S}}}{dx} = 0. \tag{7.25}$$

If we consider the special case of a fin, we can replace the last two terms of Eq. (7.25) by the corresponding boundary condition, which in this case is defined by the Robin type:

$$\nabla T \cdot \mathbf{n} = -\frac{\alpha}{\lambda} (T(x) - T_\infty) \tag{7.26}$$

and therefore

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(2l(x)\frac{\mathrm{d}T(x)}{\mathrm{d}x}\right) - 2\frac{\alpha}{\lambda}\frac{\mathrm{d}s}{\mathrm{d}x}(T(x) - T_\infty) = 0 \tag{7.27}$$

where an axisymmetric body is assumed and consequently $l_{\mathbb{N}} = l_{\mathbb{S}} = s$. Eq. (7.27) describes the heat transfer of a 2D axysimmetric fin, assuming a very low Biot number in the conjugate heat transfer problem. Now let us define a normalized temperature θ

$$\theta \equiv \frac{T(x) - T_\infty}{T_W - T_\infty}. \tag{7.28}$$

The condition of optimal heat transfer is given by Schmidt [4], where he states that the decay of temperature along the x axis must be linear. Therefore we have

$$\theta = 1 - \frac{x}{L}. \tag{7.29}$$

Defining now two non-dimensional numbers

$$\zeta(x) = \frac{l(x)}{L} \quad \text{and} \quad \xi = \frac{x}{L} \tag{7.30}$$

and assuming a long fin so that $ds \approx dx$, Eq. (7.27) becomes [3]

$$\frac{\mathrm{d}\zeta}{\mathrm{d}\xi} + \frac{\alpha L}{\lambda}(1 - \xi) = 0 \tag{7.31}$$

The solution of Eq. (7.31) give us, consequently, an expression for an optimal profile. It reads:

$$\zeta(\xi) = \frac{\alpha L}{2\lambda}(\xi - 1)^2 \tag{7.32}$$

where the boundary condition $\zeta(1) = 0$ has been applied.

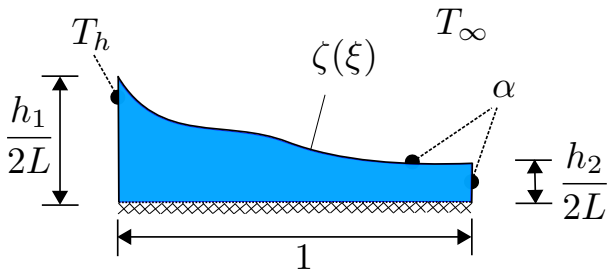


Figure 7.4: Cooling fin with a surface described by the function $\zeta(\xi)$

7.7 Exercises: Optimal Cooling Fin Shape

We apply numerical optimization to determine the optimal cooling fin shape, i. e. the shape that provides the largest heat flux through the boundaries surrounded by the atmosphere. The fin is modeled as a 2D axisymmetric body.

Stage 1: The objective function

- Assume that the profile of the fin $\zeta(\xi)$ is given by a polynomial expression of second order as

$$\zeta(\xi) = c_1 + c_2\xi + c_3\xi^2, \tag{7.33}$$

- Express the global heat flux through walls as

$$\int_{l_{\text{wall}}} \nabla T \cdot \mathbf{n}_{\text{wall}}\, dl_{\text{wall}} \quad \text{where} \quad \nabla T \cdot \mathbf{n}_{\text{wall}} = \alpha(T_{\text{wall}} - T_\infty), \tag{7.34}$$

- Create a function that computes the thermal energy dissipated through the fin for given values of the vector $\boldsymbol{c} = (c_1, c_2, c_3)^T$.

Stage 2: The constrains

1) We want $\zeta(0) = \frac{h_1}{2L}$ so that the fin model has the required height at the attachment area. Accordingly we need

$$c_1 = \frac{h_1}{2L}. \tag{7.35}$$

2) We set the length of the fin L. In correspondance, we need that

$$\zeta(1) = (c_1 + c_2 + c_3) \geq 0. \tag{7.36}$$

- Use the matlab function *fmincon* to find the optimal shape of the fin.

Stage 3: Validation

- Use the model of Eq. (7.32) to validate the results in the limit case of Bi$\ll 1$.
- Plot the Temperature along the axis of the fin. Is it linear?

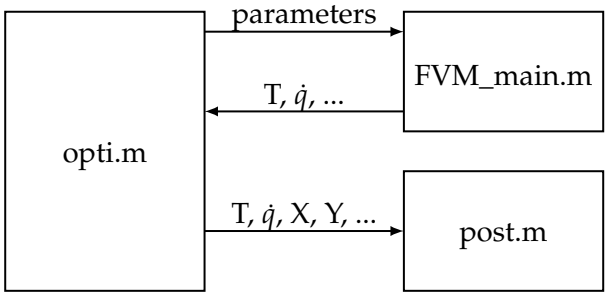
Stage 4: Repeat the previous steps for more complex fin shapes (polynomials of third order or higher):

$$\zeta(\xi) = c_1 + c_2\xi + c_3\xi^2 + c_4\xi^3 + \cdots \tag{7.37}$$

7.7.1 Useful MATLAB commands

`fmincon(...)` calls a function (e.g. our finite volume code) several times using different parameters. The optimal solution is returned along with the best parameter values. How to call this rather complex function is well described in the MATLAB documentation.

7.7.2 Flowchart



8

Finite Element Methods

Objectives

- Weak Formulation of a PDE
- Concept of the spacial discretisation using Finite Element Method

Contents

To complete the overview of the most important methods to solve PDEs, we will deal in this chapter with the *Finite Element Method (FEM)* . It is the state of the art tool for solid body simulations but is also used for CFD. Main features of FEM can be 'sketched' under a 1D approach. Extensions to 2D or 3D mainly consists in considering 2D and 3D elements, respectively, whereas the general procedure remains the same. Accordingly, we discuss in this course the FEM for solving the 1D heat equation.

8.1 Weak Form

A *strong solution* of the heat equation

$$\lambda \nabla^2 T - \rho c \frac{\partial T}{\partial t} = 0 \tag{8.1}$$

has to satisfy it at every spatial point of the domain Ω for every instance of time. For the case of a parabolic equation, as the heat equation, this strong solution is generally found without too much difficulty. In contrast, for hyperbolic problems, a strong solution sometimes does not exist. This situation arises in cases, for example, when the solution accounts for discontinuities (shock waves when solving the set of Euler equations for instance). In such points the derivatives cannot be formed and, consequently, there is no solution in a strong meaning.

To overcome this issue, the concept of *weak solutions* has been introduced. For the heat equation, it means that

$$\int_V N_i(x) \left[\lambda \nabla^2 T - \rho c \frac{\partial T}{\partial t} \right] \mathrm{d}V = 0 \tag{8.2}$$

for a given set of *test functions* N_i. The set of test functions may consists of infinitesimal many functions. It is necessary to point out that every strong solution is also a weak solution. Moreover, whenever there exits a strong solution, it is of course also the only weak solution.

From now on, we restrict the problem to be one dimensional so that $\Omega = [0, l]$. To evaluate the integral above, a common procedure is used derived from the partial integration

$$(uv)' = uv' + u'v \Longrightarrow uv' = -u'v + (uv)' \Longrightarrow \int uv' = uv - \int u'v. \tag{8.3}$$

Note that for 2D or 3D cases, Green's second identity is used instead of partial integration. Considering now

$$u = N_i(x) \quad \text{and} \quad v' = \lambda \frac{\partial^2 T(x,t)}{\partial x^2}, \tag{8.4}$$

we obtain from the first term of Eq. (8.2)

$$\int_0^l \lambda N_i(x) \frac{\partial^2 T(x,t)}{\partial x^2} \mathrm{d}x = \left[\lambda N_i(x) \frac{\partial T(x,t)}{\partial x} \right]_0^l - \int_0^l \lambda \frac{\partial N_i(x)}{\partial x} \frac{\partial T(x,t)}{\partial x} \mathrm{d}x. \tag{8.5}$$

Equation (8.2) is then expressed in its weak formulation as

$$\int_0^l \lambda \frac{\partial N_i(x)}{\partial x} \frac{\partial T(x,t)}{\partial x} \mathrm{d}x + \int_0^l \rho c N_i(x) \frac{\partial T(x,t)}{\partial t} \mathrm{d}x = \left[\lambda N_i(x) \frac{\partial T(x,t)}{\partial x} \right]_0^l \tag{8.6}$$

which has to be satisfied for all test functions N_i.

By using the weak formulation, the degree of the derivative of the temperature T is reduced by one and is "transported" to a derivative of the test function. Consequently, the aforementioned difficulties related to discontinuities in the solution can be overcome. In addition, we observe that the test functions require a certain differentiability, here at least piece wise.

8.2 Main Idea

We can use the above definition to derive a discretisation scheme. The main idea is to find a suitable finite set of base functions $H_j(x)$ by which we construct a numerical solution written as a linear combination

$$T(x,t) = \sum_{j=1}^{n} H_j(x) a_j(t). \tag{8.7}$$

As we can see here, a solution based on a FEM scheme provides a spatial function and not only the values on certain grid points or volumes, as is the case for FDM and FVM. Having chosen the base functions, the problem shrinks to find the n coefficients a_j. To do so, we have to set up n equations. This can be done by using n test functions N_j and plug them into Eq. (8.2). So, we can set up a linear equation for every test function. Summarizing, we have to construct n base functions and n test functions.

Since finding global suitable base and test functions can be challenging, especially for complex geometries, the domain is split into several subdomains in which we can define those functions independently from the other subdomains. Such a subdomain together with the corresponding sets of base and test functions is the so-called *finite element* .

8.3 Base Functions

Here, we deal with linear base functions on a 1D element $[x_{i-1}, x_i]$. The linear function $T(x)$, for $x_{i-1} \leq x \leq x_i$ can be written as a linear combination of two base functions $H_{i-1}(x)$ and $H_i(x)$ as clearly observed in Fig. 8.1. We extend this approach on the whole domain with the hat functions

$$H_i(x) = \begin{cases} \frac{x_{i-1}-x}{x_{i-1}-x_i} & , \quad x_{i-1} \leq x < x_i \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & , \quad x_i \leq x \leq x_{i+1} \\ 0 & , \quad \text{elsewise} \end{cases} . \tag{8.8}$$

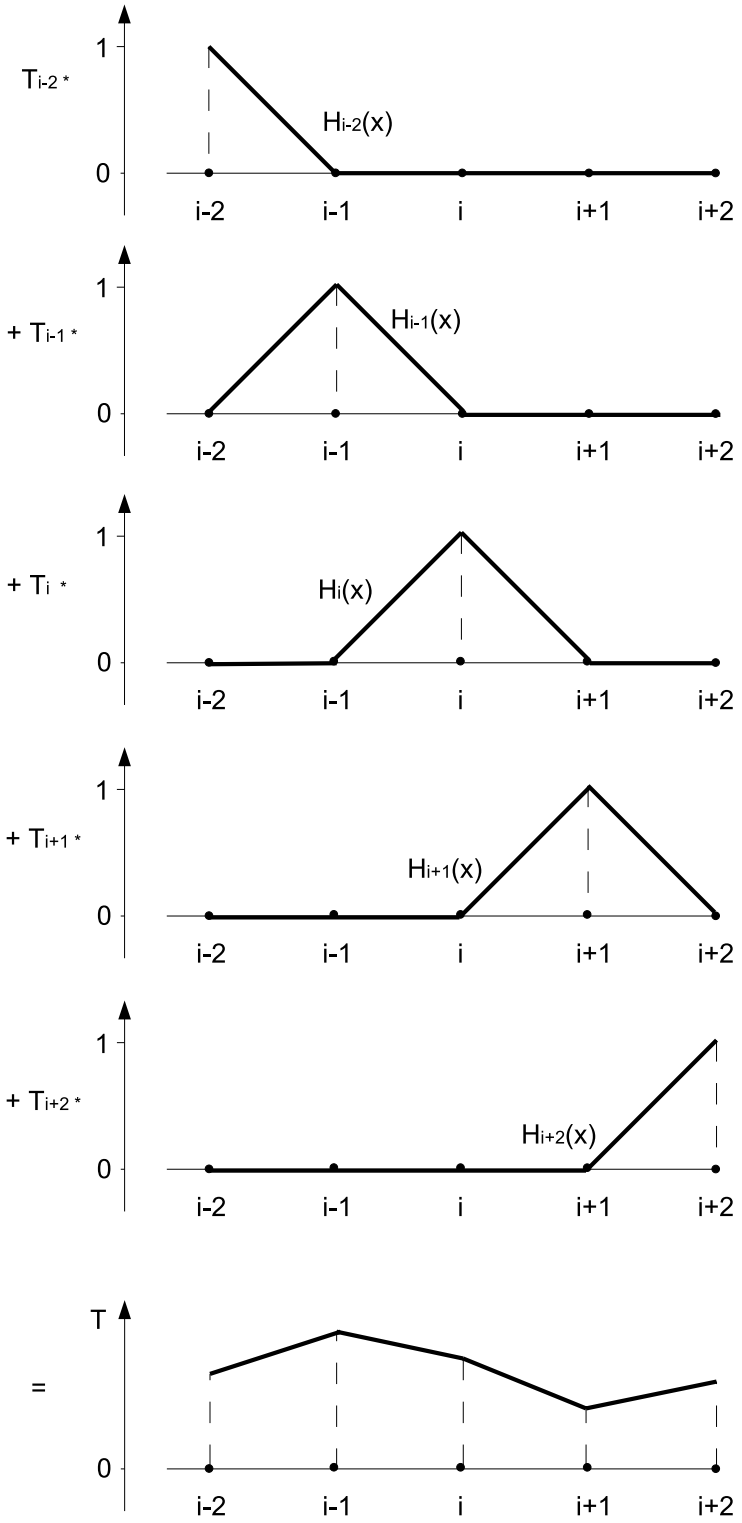


Figure 8.1: Illustration of the base functions H_i composing the piecewise linear function T form Eq. (8.9).

In total, the temperature distribution on the discretized domain with nodes $\{x_1, x_2, \ldots, x_n\}$ is described by the piecewise linear function

$$T(x,t) = \sum_{j=1}^{n} H_j(x) \cdot T_j(t). \tag{8.9}$$

The composition of T by the base functions is illustrated in Fig. 8.1. By setting the base functions in this manner, the coefficients $a_j(t)$ of Eq. (8.7) are equal to the temperature at the nodes $T_j(t) = T(x_j, t)$.

8.4 Test Functions

To close the problem, we have to define the test functions $N_i(x)$. A common approach, recognized as the Galerkin approach, is to use the same test and base functions, i. e. $N_i(x) = H_i(x)$. Equation (8.6) can thus be written as

$$\mathcal{D} \int_0^l \frac{\partial H_i(x)}{\partial x} \sum_{j=1}^{n} \frac{\partial H_j(x)}{\partial x} T_j(t)\, \mathrm{d}x + \int_0^l H_i(x) \sum_{j=1}^{n} H_j(x) \frac{\partial T_j(t)}{\partial t} \mathrm{d}x = \left[\mathcal{D} H_i(x) \frac{\partial T(x,t)}{\partial x} \right]_0^l, \tag{8.10}$$

where we assume constant material values and $\mathcal{D} = \frac{\lambda}{\rho c}$. Exchanging integration and summation, leads to

$$\mathcal{D} \sum_{j=1}^{n} T_j(t) \int_0^l \frac{\partial H_i(x)}{\partial x} \frac{\partial H_j(x)}{\partial x}\, \mathrm{d}x + \sum_{j=1}^{n} \frac{\partial T_j(t)}{\partial t} \int_0^l H_i(x) H_j(x) \mathrm{d}x = \left[\mathcal{D} H_i(x) \frac{\partial T(x,t)}{\partial x} \right]_0^l \tag{8.11}$$

for all H_i.

8.5 Element Matrix

Since we can evaluate the elements separately, we pick up an element $[x_{i-1}, x_i]$ to discuss the evaluation of the integrals in detail. Accordingly, the integrals are evaluated now from $i-1$ to i. For the element $[x_{i-1}, x_i]$, as we have seen, there are contributions only of the test functions $H_{i-1}(x)$ and $H_i(x)$. That is the reason why the sum is now evaluated only from $i-1$ to i. If higher order test functions were used, the sum would probably include a bigger amount of test functions. Equation 8.11 reduces to

node $i-1$

$$\mathcal{D} \sum_{j=i-1}^{j=i} T_j \int_{x_{i-1}}^{x_i} H'_{i-1} H'_j\, \mathrm{d}x + \sum_{j=i-1}^{j=i} \dot{T}_j \int_{x_{i-1}}^{x_i} H_{i-1} H_j \mathrm{d}x = 0 \tag{8.12}$$

node i

$$\mathcal{D}\sum_{j=i-1}^{j=i} T_j \int_{x_{i-1}}^{x_i} H_i' H_j' \,\mathrm{d}x + \sum_{j=i-1}^{j=i} \dot{T}_j \int_{x_{i-1}}^{x_i} H_i H_j \mathrm{d}x = 0 \tag{8.13}$$

where we introduced the abbreviations $H_i' = \frac{\partial H_i(x)}{\partial x}$ and $\dot{T}_j = \frac{\partial T_j(t)}{\partial t}$.

Introducing now the vectors

$$\boldsymbol{H}_E = \begin{pmatrix} \frac{x-x_{i-1}}{x_i-x_{i-1}} \\ \frac{x-x_i}{x_{i-1}-x_i} \end{pmatrix}, \boldsymbol{H}_E' = \begin{pmatrix} \frac{1}{x_i-x_{i-1}} \\ \frac{1}{x_{i-1}-x_i} \end{pmatrix}, \boldsymbol{T}_E = \begin{pmatrix} T_{i-1} \\ T_i \end{pmatrix}, \tag{8.14}$$

Eqs. (8.12) and (8.13) can be summarized in matrix vector notation as

$$\mathcal{D}\int_{x_{i-1}}^{x_i} \boldsymbol{H_E'} \boldsymbol{H_E'}^T \mathrm{d}x\, \boldsymbol{T} + \int_{x_{i-1}}^{x_i} \boldsymbol{H_E} \boldsymbol{H_E}^T \mathrm{d}x\, \dot{\boldsymbol{T}} = 0. \tag{8.15}$$

We determine the appearing integrals using the convention $\Delta x = x_i - x_{i-1}$:

$$\int_{x_{i-1}}^{x_i} \boldsymbol{H}_E' \boldsymbol{H}_E'^T \mathrm{d}x = \frac{1}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{8.16}$$

and

$$\int_{x_{i-1}}^{x_i} \boldsymbol{H}_E \boldsymbol{H}_E^T \mathrm{d}x = \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}. \tag{8.17}$$

Inserting these values in Eq. (8.15) leads finally to

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_{i-1} \\ T_i \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T}_{i-1} \\ \dot{T}_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{8.18}$$

Exactly the same procedure for element $[x_i, x_{i+1}]$ can be performed resulting in

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_i \\ T_{i+1} \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T}_i \\ \dot{T}_{i+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{8.19}$$

8.6 Boundary conditions

Note that the treatment of boundary conditions is given directly by the weak formulation of Eq. (8.6). For node 1 this equation reads:

$$\mathcal{D}\sum_{j=1}^{j=2} T_j \int_{x_1}^{x_2} H_1' H_j' \,\mathrm{d}x + \sum_{j=1}^{j=2} \dot{T}_j \int_{x_1}^{x_2} H_1 H_j \,\mathrm{d}x = -\mathcal{D}\frac{\partial T(0,t)}{\partial x} \tag{8.20}$$

and therefore the matrix for element $[x_1, x_2]$ is given by

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T}_1 \\ T_2 \end{pmatrix} = -\mathcal{D}\begin{pmatrix} \frac{\partial T(0,t)}{\partial x} \\ 0 \end{pmatrix}. \tag{8.21}$$

In a similar way Eq. (8.6) for node n reads:

$$\mathcal{D}\sum_{j=n-1}^{j=n} T_j \int_{x_{n-1}}^{x_n} H_n' H_j' \,\mathrm{d}x + \sum_{j=n-1}^{j=n} \dot{T}_j \int_{x_{n-1}}^{x_n} H_n H_j \,\mathrm{d}x = \mathcal{D}\frac{\partial T(L,t)}{\partial x} \tag{8.22}$$

with the corresponding matrix for the element $[x_{n-1}, x_n]$ defined as

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_{n-1} \\ T_n \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T}_{n-1} \\ T_n \end{pmatrix} = \mathcal{D}\begin{pmatrix} 0 \\ \frac{\partial T(L,t)}{\partial x} \end{pmatrix}. \tag{8.23}$$

It should be noted from Eqs. (8.21) and (8.23), that boundary conditions of the type Neumann and Robin are imposed naturally in the weak formulation. The only procedure necessary to do is to replaced the values of $\frac{\partial T}{\partial x}$ by the ones that need to be imposed. Dirichlet boundary conditions are applied in a hard way. It means that the row corresponding to the node at which Dirichlet is applied, is simply replaced by a row where the temperature T is directly imposed. This should be more clear by the example given in the following section.

8.7 System Matrix

To set up the system matrix, we have to merge the contributes of each element in a single linear system of equations. This will be done exemplarily for the domain $\{x_1, x_2, x_3\}$ consisting of two elements. According to the previous section, we set up two linear systems assuming a equidistant grid and constant values of thermal diffusivity $\mathcal{D}$

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T}_1 \\ T_2 \end{pmatrix} = -\mathcal{D}\begin{pmatrix} \frac{\partial T}{\partial x} \\ 0 \end{pmatrix} \tag{8.24}$$

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{pmatrix} T_2 \\ T_3 \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T_2} \\ T_3 \end{pmatrix} = \mathcal{D}\begin{pmatrix} 0 \\ \frac{\partial T}{\partial x} \end{pmatrix}. \tag{8.25}$$

We know that both the second line of system (8.24) and the first line of system (8.25) correspond to the evaluations with the test function H_2. Thus, we have to add those equations to come back to the original from of Eq. (**??**). The total system is described by

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1+1 & -1 \\ 0 & -1 & 1 \end{bmatrix}\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} + \Delta x \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{3}+\frac{1}{3} & \frac{1}{6} \\ 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T_1} \\ T_2 \\ T_3 \end{pmatrix} = \mathcal{D}\begin{pmatrix} -\frac{\partial T}{\partial x} \\ 0 \\ \frac{\partial T}{\partial x} \end{pmatrix}. \tag{8.26}$$

We observe that

- the first element contributes to the positions $(1,1)$, $(1,2)$, $(2,1)$, $(2,2)$;
- the second element contributes to the positions $(2,2)$, $(2,3)$, $(3,2)$, $(3,3)$;
- at position $(2,2)$ expressions of both elements are added ;
- the right-hand side of the system contains only heat fluxes at the boundary (and eventual sources).

If a Dirichlet boundary condition is desired, at node 1 for instance, this condition must be imposed in a hard way. It means that the first row of Eq. (8.27) should be replaced by a row that set up an equation exclusively for that node. In such a case Eq. (8.27) would turn out to be:

$$\frac{\mathcal{D}}{\Delta x}\begin{bmatrix} \Delta x & 0 & 0 \\ -1 & 1+1 & -1 \\ 0 & -1 & 1 \end{bmatrix}\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} + \Delta x \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{3}+\frac{1}{3} & \frac{1}{6} \\ 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix}\begin{pmatrix} \dot{T_1} \\ T_2 \\ T_3 \end{pmatrix} = \mathcal{D}\begin{pmatrix} T_{\mathbb{W}est} \\ 0 \\ \frac{\partial T}{\partial x} \end{pmatrix}. \tag{8.27}$$

Having set up the spacial discretisation, we can solve the steady problem by setting $\dot{T} = 0$ or the unsteady problem using the temporal discretisation schemes discussed in Chp. 4

8.8 Exercises

We want to solve

$$\frac{\mathrm{d}^2T}{\mathrm{d}x^2} = -e^x \qquad 0 < x < 1 \quad \text{with} \quad T(0) = 0 \quad \text{and} \quad T(1) = 0 \tag{8.28}$$

An estimate of the solution can be expressed as a linear combination of base functions

$$\tilde{T}(x) = c_1 H_1(x) + c_2 H_2(x), \tag{8.29}$$

where $H_1(x)$ and $H_2(x)$ are polynomials of second and third order, respectively. First, we have to find bases functions $H_1(x)$ and $H_2(x)$ that satisfy the boundary conditions. Accordingly, we can use

$$H_1(x) = x(1-x) \quad \text{and} \quad H_2(x) = x^2(1-x). \tag{8.30}$$

Stage 1: Analysis

- Derive the exact solution of Eq. (8.28).

Stage 2: Collocation method

The collocation method consists of forcing the residual

$$r(x) = \frac{\mathrm{d}^2\tilde{T}}{\mathrm{d}x^2} + e^x \tag{8.31}$$

to be zero at the so called *collocation points*.

- Choose two collocation points x_1 and x_2 within the domain. Two equations results after evaluating the residuals at the chosen points. Solve the linear system to find the values of c_1 and c_2.
- Choose three collocation points x_1, x_2, and x_3 within the domain. Three equations results after evaluating the residuals at the chosen points. Solve the overdetermined system to find the values of c_1 and c_2 such that the residual is minimal in a least square sense.
- Write down the resulting expressions for $\tilde{T}(x)$.

Stage 3: Method of least squares

The second method consists of minimizing the residual in an average sense. Now, we claim that

$$M(c_1, c_2) = \int_0^1 r(x, c_1, c_2)^2 \mathrm{d}x \quad \text{to be a minimum} \quad \Rightarrow \quad \frac{\mathrm{d}M(c_1, c_2)}{\mathrm{d}c_i} = 0. \tag{8.32}$$

- Find the values of the coefficients c_1 and c_2 such that the square of the residual averaged on the domain is minimized.
- Write down the resulting expression for $\tilde{T}(x)$.

Stage 4: Method of Galerkin

The Galerkin method consists of minimizing a weighted residual over the domain

$$\int_0^1 N_i(x) r(x) \mathrm{d}x = 0, \tag{8.33}$$

where the test functions $N_i(x)$ are taken to be the base functions $H_i(x)$.

- Find the values of the coefficients c_1 and c_2 corresponding to the Galerkin method.
- Write down the resulting expression for $\tilde{T}(x)$.

Stage 5: Method of Finite Elements combined with the Galerkin approach

- Resolve now the same equation discretizing the domain with three finite elements considering the hat base functions.
- Write down the resulting expression for $\tilde{T}(x)$.

Stage 6: Finite elements and the heat equation

- Solve the steady heat equation for:
  - $\mathbb{W}$est and $\mathbb{E}$ast with Robin BC.
  - $\mathbb{W}$est as Dirichlet BC and $\mathbb{E}$ast as Robin BC.
- Solve the unsteady heat equation, considering an implicit scheme, for:
  - $\mathbb{W}$est and $\mathbb{E}$ast with Robin BC.
  - $\mathbb{W}$est as Dirichlet BC and $\mathbb{E}$ast as Robin BC.

Stage 7: Comparison (Optional)

- Plot the solutions given by each method in a single plot.

- Repeat Stage 1 to 3 considering the linear combination

$$\tilde{T}(x) = \sum_{n=1}^{n} c_n H_n(x), \quad \text{where} \quad H_n(x) = x^n(x-1) \tag{8.34}$$

  with $n = 5$. Integrate numerically.

  Use the finite element method for 5 elements and the same hat base functions.

- Plot together the solutions given by each method.

A

Addendum to Finite Volumes

A.1 Uniform rectangular grid and Boundary Conditions

If the geometry under study is Cartesian, the following terms

- $\Delta y_{\mathbb{S}w}^{\mathbb{S}e} = \Delta y_{se}^{sw} = \Delta y_{e}^{w} = \Delta y_{ne}^{nw} = \Delta y_{\mathbb{N}e}^{\mathbb{N}w} = \Delta y_{s\mathbb{W}}^{s} = \Delta y_{n}^{n\mathbb{W}} = \Delta y_{s}^{s\mathbb{E}} = \Delta y_{n}^{n\mathbb{E}} = 0$

vanish, and the spatial derivatives of Eq. (3.16) become

$$\left.\frac{\partial T}{\partial x}\right|_s = \frac{1}{S^s}\left(\Delta y_{\mathbb{S}e}^{e} T_{se} + \Delta y_{w}^{\mathbb{S}w} T_{sw}\right), \tag{A.1}$$

$$\left.\frac{\partial T}{\partial y}\right|_s = \frac{-1}{S^s}\left(\Delta x_{\mathbb{S}w}^{\mathbb{S}e} T_{\mathbb{S}} + \Delta x_{e}^{w} T_P\right), \tag{A.2}$$

$$\left.\frac{\partial T}{\partial x}\right|_e = \frac{1}{S^e}\left(\Delta y_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta y_{n}^{s} T_{\mathbb{P}}\right), \tag{A.3}$$

$$\left.\frac{\partial T}{\partial y}\right|_e = \frac{-1}{S^e}\left(\Delta x_{s}^{s\mathbb{E}} T_{se} + \Delta x_{n\mathbb{E}}^{n} T_{ne}\right), \tag{A.4}$$

$$\left.\frac{\partial T}{\partial x}\right|_n = \frac{1}{S^n}\left(\Delta y_{e}^{\mathbb{N}e} T_{ne} + \Delta y_{\mathbb{N}w}^{w} T_{nw}\right), \tag{A.5}$$

$$\left.\frac{\partial T}{\partial y}\right|_n = \frac{-1}{S^n}\left(\Delta x_{w}^{e} T_{\mathbb{P}} + \Delta x_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}}\right), \tag{A.6}$$

$$\left.\frac{\partial T}{\partial x}\right|_w = \frac{1}{S^w}\left(\Delta y_{s}^{n} T_{\mathbb{P}} + \Delta y_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}}\right), \tag{A.7}$$

$$\left.\frac{\partial T}{\partial y}\right|_w = \frac{-1}{S^w}\left(\Delta x_{s\mathbb{W}}^{s} T_{sw} + \Delta x_{n}^{n\mathbb{W}} T_{nw}\right). \tag{A.8}$$

$$\tag{A.9}$$

The final expression simplifies then to

$$\begin{aligned}\nabla^2 T\big|_P \approx \frac{\lambda}{S^P}\left[-\Delta x_{sw}^{se}\left.\frac{\partial T}{\partial y}\right|_s + \Delta y_{se}^{ne}\left.\frac{\partial T}{\partial x}\right|_e - \Delta x_{ne}^{nw}\left.\frac{\partial T}{\partial y}\right|_n + \Delta y_{nw}^{sw}\left.\frac{\partial T}{\partial x}\right|_w\right] = \\ \frac{\lambda \Delta x_{sw}^{se}}{S^P S^s}\left(\Delta x_{\mathbb{S}w}^{\mathbb{S}e} T_{\mathbb{S}} + \Delta x_{e}^{w} T_P\right) + \frac{\lambda \Delta y_{se}^{ne}}{S^P S^e}\left(\Delta y_{s\mathbb{E}}^{n\mathbb{E}} T_{\mathbb{E}} + \Delta y_{n}^{s} T_P\right) \\ + \frac{\lambda \Delta x_{ne}^{nw}}{S^P S^n}\left(\Delta x_{w}^{e} T_P + \Delta x_{\mathbb{N}e}^{\mathbb{N}w} T_{\mathbb{N}}\right) + \frac{\lambda \Delta y_{nw}^{sw}}{S^P S^w}\left(\Delta y_{n\mathbb{W}}^{s\mathbb{W}} T_{\mathbb{W}} + \Delta y_{n}^{s} T_P\right)\end{aligned} \tag{A.10}$$

Considering rectangular cells of same size, where $\Delta x = \Delta x_{sw}^{se} = \Delta x_{\mathbb{S}w}^{\mathbb{S}e} = -\Delta x_{e}^{w} = -\Delta x_{ne}^{nw} = \Delta x_{w}^{e} = -\Delta x_{\mathbb{N}e}^{\mathbb{N}w}$, and $\Delta y = \Delta y_{se}^{ne} = \Delta y_{s\mathbb{E}}^{n\mathbb{E}} = -\Delta y_{n}^{s} T_P = -\Delta y_{nw}^{sw} = \Delta y_{s}^{n} = -\Delta y_{n\mathbb{W}}^{s\mathbb{W}}$, Eq. (A.10) breaks down into:

$$\begin{aligned}\nabla^2 T\big|_P \approx -\frac{\Delta x}{(\Delta x \Delta y)^2}\left(-\Delta x T_{\mathbb{S}} + \Delta x T_P\right) + \frac{\Delta y}{(\Delta x \Delta y)^2}\left(-\Delta y T_P + \Delta y T_{\mathbb{E}}\right) \\ + \frac{\Delta x}{(\Delta x \Delta y)^2}\left(-\Delta x T_P + \Delta x T_{\mathbb{N}}\right) - \frac{\Delta y}{(\Delta x \Delta y)^2}\left(-\Delta y T_{\mathbb{W}} + \Delta y T_P\right)\end{aligned} \tag{A.11}$$

After some algebra, Eq. (A.11) becomes

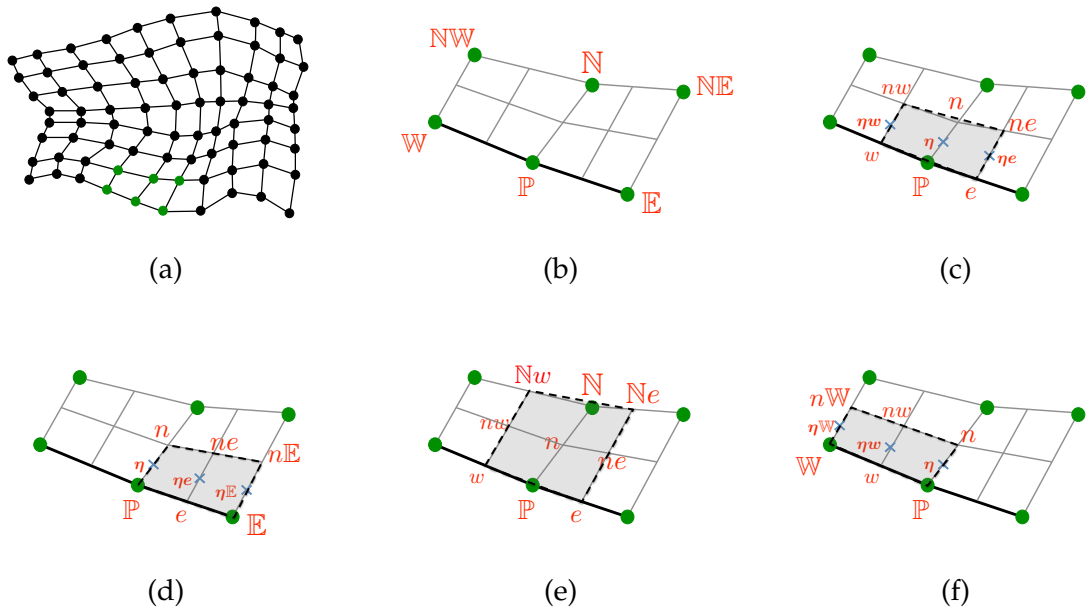
$$\nabla^2 T = \frac{1}{S^P} \oint_{\partial S^P} \nabla T \cdot \mathbf{n} dl \approx \frac{T_{\mathbb{W}} - 2T_P + T_{\mathbb{E}}}{(\Delta x)^2} + \frac{T_{\mathbb{S}} - 2T_P + T_{\mathbb{N}}}{(\Delta y)^2} \tag{A.12}$$

which is in fact the formulation given by the finite difference approach for a 2D uniform rectangular grid.

A.2 Boundary conditions

In this section we will give an example of how to discretize the Laplacian operator ∇^2 on a node which lays on the 'south' boundary. The discretization of ∇^2 on a node placed at 'east', 'north' or 'west' boundaries can be derived in a similar manner.

A.2.1 South



(a) (b) (c)

(d) (e) (f)

Figure A.1: Labeling of nodes and point between nodes in the region close to a South boundary.

Figure A.1 illustrates the cells and nodes adjancent to a given region at the south of the domain. First, the Divergence theorem is applied to the region encircling P (Fig. A.1(a)):

$$\nabla^2 T\big|_P = \frac{1}{S^\eta} \oint_{\partial S^\eta} \nabla T \cdot \mathbf{n} dl = \frac{1}{S^\eta} \left[\int_{l_w^e} (\nabla T) \cdot \mathbf{n} dl + \int_{l_e^{ne}} (\nabla T) \cdot \mathbf{n} dl + \int_{l_{ne}^{nw}} (\nabla T) \cdot \mathbf{n} dl + \int_{l_{nw}^{w}} (\nabla T) \cdot \mathbf{n} dl \right]. \tag{A.13}$$

Subsequently the mid-point rule of integration is considered. The approximation of the Laplacian operator at node P is written then as:

$$\nabla^2 T\big|_P = \frac{1}{S^\eta}\oint_{\partial S^\eta} \nabla T\cdot\mathbf{n}dS \approx \frac{1}{S^\eta}\left[\nabla T|_P\cdot\mathbf{n}\Delta l_w^e + \left.\frac{\partial T}{\partial x}\right|_{\eta e}\Delta y_e^{ne} - \left.\frac{\partial T}{\partial y}\right|_{\eta e}\Delta x_e^{ne} + \left.\frac{\partial T}{\partial x}\right|_{n}\Delta y_{ne}^{nw} - \left.\frac{\partial T}{\partial y}\right|_{n}\Delta x_{ne}^{nw} + \left.\frac{\partial T}{\partial x}\right|_{\eta w}\Delta y_{nw}^{w} - \left.\frac{\partial T}{\partial y}\right|_{\eta w}\Delta x_{nw}^{w}\right] \tag{A.14}$$

Note that the first term on the right hand side of Eq. (A.14) is the term associated to the boundary contour. It was not decomposed in x and y directions. Applying now Green's theorem to the remaining terms of Eq. (A.14) leads to

$$\left.\frac{\partial T}{\partial x}\right|_{\eta e} = \frac{1}{S^{\eta e}}\oint_{\partial S^{\eta e}} T dy = \frac{1}{S^{\eta e}}\left(\int_{\mathbb{P}}^{\mathbb{E}} T dy + \int_{\mathbb{E}}^{n\mathbb{E}} T dy + \int_{n\mathbb{E}}^{n} T dy + \int_{n}^{\mathbb{P}} T dy\right) \approx \frac{1}{S^{\eta e}}\left(\Delta y_{\mathbb{P}}^{\mathbb{E}}\, T|_e + \Delta y_{\mathbb{E}}^{n\mathbb{E}}\, T|_{\eta\mathbb{E}} + \Delta y_{n\mathbb{E}}^{n}\, T|_{ne} + \Delta y_{n}^{\mathbb{P}}\, T|_{\eta}\right) \tag{A.15}$$

$$\left.\frac{\partial T}{\partial y}\right|_{\eta e} = \frac{1}{S^{\eta e}}\oint_{\partial S^{\eta e}} T dx \approx \frac{-1}{S^{\eta e}}\left(\Delta x_{\mathbb{P}}^{\mathbb{E}}\, T|_e + \Delta x_{\mathbb{E}}^{n\mathbb{E}}\, T|_{\eta\mathbb{E}} + \Delta x_{n\mathbb{E}}^{n}\, T|_{ne} + \Delta x_{n}^{\mathbb{P}}\, T|_{\eta}\right) \tag{A.16}$$

$$\left.\frac{\partial T}{\partial x}\right|_{n} = \frac{1}{S^{n}}\int_{\partial S^{n}} T dy \approx \frac{1}{S^{n}}\left(\Delta y_{w}^{e}\, T|_P + \Delta y_{e}^{\mathbb{N}e}\, T|_{ne} + \Delta y_{\mathbb{N}e}^{\mathbb{N}w}\, T|_{\mathbb{N}} + \Delta y_{\mathbb{N}w}^{w}\, T|_{nw}\right) \tag{A.17}$$

$$\left.\frac{\partial T}{\partial y}\right|_{n} = \frac{1}{S^{n}}\int_{\partial S^{n}} T dx \approx \frac{-1}{S^{n}}\left(\Delta x_{w}^{e}\, T|_P + \Delta x_{e}^{\mathbb{N}e}\, T|_{ne} + \Delta x_{\mathbb{N}e}^{\mathbb{N}w}\, T|_{\mathbb{N}} + \Delta x_{\mathbb{N}w}^{w}\, T|_{nw}\right) \tag{A.18}$$

$$\left.\frac{\partial T}{\partial x}\right|_{\eta w} = \frac{1}{S^{\eta w}}\oint_{\partial S^{\eta w}} T dy \approx \frac{1}{S_{\eta w}}\left(\Delta y_{\mathbb{W}}^{\mathbb{P}}\, T|_w + \Delta y_{\mathbb{P}}^{n}\, T|_{\eta} + \Delta y_{n}^{n\mathbb{W}}\, T|_{nw} + \Delta y_{n\mathbb{W}}^{\mathbb{W}}\, T|_{\eta\mathbb{W}}\right) \tag{A.19}$$

$$\left.\frac{\partial T}{\partial y}\right|_{\eta w} = \frac{1}{S^{\eta w}}\oint_{\partial S^{\eta w}} T dx \approx \frac{-1}{S_{\eta w}}\left(\Delta x_{\mathbb{W}}^{\mathbb{P}}\, T|_w + \Delta x_{\mathbb{P}}^{n}\, T|_{\eta} + \Delta x_{n}^{n\mathbb{W}}\, T|_{nw} + \Delta x_{n\mathbb{W}}^{\mathbb{W}}\, T|_{\eta\mathbb{W}}\right) \tag{A.20}$$

The first term on the right hand side of Eq. (A.14) is now replaced by the desired boundary condition as follows:

Neumann BC

$$-\nabla T|_{\mathbb{P}} \cdot \mathbf{n} = \dot{q} \tag{A.21}$$

Robin BC

$$\nabla T|_{\mathbb{P}} \cdot \mathbf{n} = -\frac{\alpha}{\lambda}(T_{\mathbb{P}} - T_{\infty}) \tag{A.22}$$

Evidently for Dirichlet BC there is no necessity of deriving such a procedure, since values at the corresponding node would be directly given.

Index