Numerical Algorithms for Visual Computing II

Michael Breuß

Contents

Chapter 1

Introduction

Contents: Analysis and understanding of components of numerical solvers for partial differential equations, discretisations for different types of problems.

What is a partial differential equation (PDE) ?
Typically, in an image we have 2 space dimensions, often denoted x and y. A PDE is a differential equation, relating an unknown, sought function $u(x, y)$ with its derivatives, e.g.,

$$\frac{\partial}{\partial x}u(x,y) + \frac{\partial}{\partial y}u(x,y) \quad = \quad 1$$

The partial derivatives work like usual derivatives; other variables than the indicated one are treated like constants.

Example:

$$\frac{\partial}{\partial x}\left[x^2 + y^2\right] \quad = \quad 2x$$

Which types of PDEs do exist?

- **Elliptic PDEs**. Its solutions describe equilibrium states of a given quantity. Thereby, an "equilibrium state", or simply an equilibrium, denotes a stable status which does not change in time.
- **Parabolic PDEs**. Describing time-dependent processes on the way to an equilibrium.
- **Hyperbolic PDEs**. Denoting time-dependent transport of a given quantity without an inherent tendency to an equilibrium as in the case of parabolic PDEs.

What about other PDEs?

There exists an incredibly large zoo of variations which may all require special numerical treatments.

- PDEs of mixed type
- single equations and systems of equations
- systems of PDEs of different types
- linear and non-linear ones
- PDEs of first, second, third or higher order
- etc...

What has this to do with Visual Computing?

- Elliptic PDEs arise, e.g., in tasks like image editing, image inpainting or optic flow
- Parabolic PDEs are directly related to diffusion filters
- Hyperbolic PDEs arise in mathematical morphology, shape from shading, and as a sub-model in many PDE-based filters

Chapter 2

Important Notions and Theoretical Background

We now introduce the mandatory vovabulary. We begin by summarising some important notions.

Let $u \equiv u(x, y)$ ("$\equiv$" means "identical with") be a smooth function; "smooth" means, it is as many times continously differentiable ("stetig differenzierbar") in the variables x and y as required.

Then:

- $\dfrac{\partial}{\partial x} u(x, y) = u_x(x, y) \equiv u_x$
- $\dfrac{\partial}{\partial y} u(x, y) = u_y(x, y) \equiv u_y$
- $\dfrac{\partial}{\partial x}\left(\dfrac{\partial}{\partial x} u(x, y)\right) = \dfrac{\partial^2}{\partial x^2} u(x, y) \equiv u_{xx}$
- $\nabla u = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$, "$\nabla$" is called Nabla-Operator
- $\nabla \cdot \nabla u = \begin{pmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \dfrac{\partial}{\partial x}(u_x) + \dfrac{\partial}{\partial y}(u_y) = u_{xx} + u_{yy} = \Delta u$, "$\Delta$" is the Laplace-Operator.

Definition 2.0.1 *The highest occuring derivative in a PDE determines its order.*

Examples

- $u_x + u_y = 0$ is a first-order PDE.
- $u_x + u_{yy} = 0$ is a second-order PDE.
- $\Delta u = 0$ is a second-order PDE.
- $u_{xxx} + u_y = 0$ is a third-order PDE.

Definition 2.0.2 *In a PDE relating a function u to its derivatives, we denote sometimes*

- *u as the dependent variable (as it is not known and depends on x and y).*
- *x, y as the independent variables (as they do not depend on something else).*

Definition 2.0.3 *Non-linear PDEs in a dependent variable u arise by coefficients of u and derivatives of u that depend on u itself.*

Examples

- $\Delta u = 0$ is a linear PDE
- $u\Delta u = 0$ is a non-linear PDE
- $\nabla \cdot (u^2 \nabla u) = 0$ is a non-linear PDE

Remark.
In order to fill a notional gap between non-linear PDEs and this definition of linearity, PDEs featuring space-variant coefficients are marked by the phrase “with non-constant (or variable) coefficients”.

Strictly speaking, we have abused the term PDE up to now – for example, just the equation

$$\Delta u \quad = \quad 0 \tag{2.1}$$

is not a well-defined mathematical object by itself. Further conditions must be specified in order to obtain a meaningful solution. For instance, in case of (2.1), if u is a function defined over $\mathbb{R}^2$, any function

$$\begin{aligned} u(x,y) &= c, \qquad c \in \mathbb{R} \\ u(x,y) &= ax + by, \qquad a,b \in \mathbb{R} \\ u(x,y) &= axy + by + c, \qquad a,b,c \in \mathbb{R} \\ &\vdots \end{aligned}$$

can be a solution. However, it would be convenient to single out one "interesting" solution.
It turns out, that it makes sense to distinguish initial conditions (for time-dependent problems where u depends also on a time t, $u \equiv u(x, y, t)$) and boundary conditions (where the domain of a PDE ends).

We then distinguish the following cases.

Definition 2.0.4 *Let u not only be a function of space, but also of time t, i.e. $u \equiv u(x, y, t)$. Let the time-scale of interest begin with $t = 0$. Then the initial value problem (IVP) is given by a combination*

$$\begin{cases} PDE & \text{for } u(x,y,t) \\ u(x,y,0) & = \varphi(x,y) \end{cases}$$

with a suitable function φ.
In case of a finite domain of interest Ω, it often makes sense to specify boundary conditions. We first define initial-boundary value problems (IBVPs) given by a combination

$$\begin{cases} PDE & \text{for } u(x,y,t) \ \text{with } x,y \in \Omega, t > 0 \\ u(x,y,0) & = \varphi(x,y), \qquad x,y \in \Omega \\ u(x,y,t) & = \eta(x,y,t), \qquad x,y \in \partial\Omega, \end{cases}$$

where $\partial\Omega$ is the boundary of Ω, and where Ω is an open domain, i.e. it does not include its boundary.
If u does not depend on time, no initial condition for a time-evolution is needed and we consider boundary value problems (BVPs) given by a combination

$$\begin{cases} PDE & \text{for } u(x,y), \qquad x,y \in \Omega, \\ u(x,y) & = \eta(x,y), \qquad x,y \in \partial\Omega. \end{cases}$$

Remarks:

- Hyperbolic and Parabolic PDEs lead to IVPs or IBVPs.
- Elliptic PDEs occur in BVPs.
- We will loosely identify PDEs with corresponding problems and just speak of the PDEs.
- One usually has to specify in which function space u is sought as this greatly influences many issues. We will usually assume that our functions are in L_2, L_1, H_1 ($\equiv$ first derivatives are in L_2) or in any other linear function space.

If a set-up is done properly, one can sometimes show rigorously, that one does not waste time with a pathological problem:

Definition 2.0.5 *We say that a given problem to a PDE is well-posed in the sense of Hadamard, if*

1. *the problem has a solution,*
2. *this solution is unique,*
3. *the solution depends continuously on the data given in the problem.*

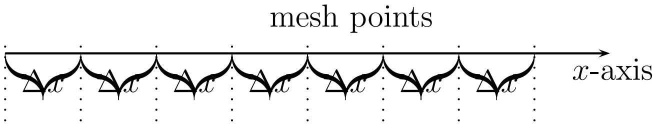
Chapter 3

Introduction to the Finite Difference Method

We begin by discretizing the spatial domain by placing a grid over it, starting in 1-D. We use the grid, or mesh, defined by

$$x_j \quad = \quad j\Delta x, \tag{3.1}$$

where Δx is the so-called mesh width and x_j are called nodes. **Sketch:**



Notationally, we use approximations

$$U_j \quad \approx \quad u(j\Delta x) \tag{3.2}$$

The next step is to approximate derivatives on this grid. Since

$$u'(x) \quad = \quad \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$

it seems that a reasonable approximation of $u'(j\Delta x)$ could be

$$\frac{U_{j+1} - U_j}{\Delta x}. \tag{3.3}$$

(3.3) is called forward difference. Other useful approximations of $u'(j\Delta x)$ are:

$$\frac{U_j - U_{j-1}}{\Delta x} \qquad \text{backward difference} \tag{3.4}$$

$$\frac{U_{j+1} - U_{j-1}}{2\Delta x} \qquad \text{central difference} \tag{3.5}$$

In a similar fashion we may approximate u'' at $j\Delta x$ by

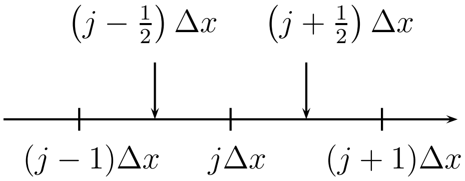
$$\frac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2} \tag{3.6}$$

In order to see that this is a reasonable approximation, consider that

$$\frac{u'((j+\frac{1}{2})\Delta x) - u'((j-\frac{1}{2})\Delta x)}{\Delta x} \tag{3.7}$$

approximates $u''(j\Delta x)$.

Sketch:


$\left(j-\frac{1}{2}\right)\Delta x$
$\left(j+\frac{1}{2}\right)\Delta x$
$(j-1)\Delta x$
$j\Delta x$
$(j+1)\Delta x$

The values $u'((j \pm \frac{1}{2})\Delta x)$ can be approximated by

$$\frac{U_{j+1} - U_j}{\Delta x} \quad \text{and} \quad \frac{U_j - U_{j-1}}{\Delta x}, \tag{3.8}$$

respectively. Then

$$\begin{aligned} u''(j\Delta x) &\approx \frac{u'((j+\frac{1}{2})\Delta x) - u'((j-\frac{1}{2})\Delta x)}{\Delta x} \\ &\approx \frac{\frac{U_{j+1} - U_j}{\Delta x} - \frac{U_j - U_{j-1}}{\Delta x}}{\Delta x} \\ &= \frac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2} \end{aligned} \tag{3.9}$$

compare (3.6).

3.1 The Local Truncation Error

As an example, let us consider the hyperbolic linear advection equation

$$\frac{\partial}{\partial t}u(x,t) + a\frac{\partial}{\partial x}u(x,t) = 0 \tag{3.10}$$

where $a \in \mathbb{R}$ is a parameter. As discretisation of $u_t + au_x = 0$ we use

$$\frac{U_{j+1}^{n+1} - U_j^n}{\Delta t} + a\frac{U_j^n - U_{j-1}^n}{\Delta x} = 0 \tag{3.11}$$

with the space-time mesh $(j\Delta x, n\Delta t)$, $j \in \mathbb{Z}$, $n \in \mathbb{N}$.

The local truncation error is then defined as follows. We assume the smoothness of an underlying solution u of the PDE $u_t + au_x = 0$. Associating then

$$\begin{cases} x & \equiv j\Delta x \\ x - \Delta x & \equiv (j-1)\Delta x \\ t & \equiv n\Delta t \\ t + \Delta t & \equiv (n+1)\Delta t \end{cases} \tag{3.12}$$

it will usually not hold exactly

$$\underbrace{\frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} + a\frac{u(x,t) - u(x-\Delta x,t)}{\Delta x}}_{=:A} \quad = \quad \underbrace{0}_{=:B} \tag{3.13}$$

(Why not? Because (3.11) is obtained by approximating $u_t + au_x = 0$. In contrast, $u(x)$ and $u(x \pm \Delta x)$ in (3.13) are supposed to be values of the exact solution u.)

The difference between the left and right hand side in (3.13) is called the local truncation error:

$$L_{\Delta t,\Delta x}(u(x,t)) \tag{3.14}$$

$$:= \quad A - B \tag{3.15}$$

$$= \quad \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} + a\frac{u(x,t) - u(x-\Delta x,t)}{\Delta x} - 0 \tag{3.16}$$

Having a look at (3.13), it is clear that $A - B$ may strongly depend on powers of Δt and Δx. We now evaluate this dependence in more detail. To this end we employ Taylor series expansions:

$$\begin{aligned} u(x,t+\Delta t) \quad &= \quad u(x,t) + [(t+\Delta t) - t] \cdot u_t(x,t) + \frac{[(t+\Delta t)-t]^2}{2} \cdot u_{tt}(x,t) \\ &\quad + \frac{[(t+\Delta t)-t]^3}{6} \cdot u_{ttt}(x,t) + \ldots \end{aligned} \tag{3.17}$$

$$\begin{aligned} u(x-\Delta x,t) \quad &= \quad u(x,t) + [(x-\Delta x) - x] \cdot u_x(x,t) + \frac{[(x-\Delta x)-x]^2}{2} \cdot u_{xx}(x) \\ &\quad + \frac{[(x-\Delta x)-x]^3}{6} \cdot u_{xxx}(x,t) + \ldots \end{aligned} \tag{3.18}$$

For some general function f, the general formula for Taylor series expansions is

$$f(x+h) \quad = \quad \sum_{k=0}^{p} \frac{h^k}{k!} f^{(k)}(x) + \mathcal{O}(h^{p+1}). \tag{3.19}$$

Thereby, $\mathcal{O}$ ("big-oh") is the so-called Landau-symbol; we will elaborate on this notion later.

Neglecting the arguments (x,t), we plug the expansions (3.17), (3.18) into (3.14):

$$\begin{aligned}
&L_{\Delta t,\Delta x}(u) \\
= \quad & \frac{\left[u + \Delta t \cdot u_t + \frac{\Delta t^2}{2} u_{tt} + \frac{\Delta t^3}{6} u_{ttt} + \mathcal{O}(\Delta t^4)\right] - u}{\Delta t} \\
& + a \frac{u - \left[u - \Delta x \cdot u_x + \frac{\Delta x^2}{2} u_{xx} + \frac{\Delta x^3}{6} u_{xxx} + \mathcal{O}(\Delta x^4)\right]}{\Delta x} - 0 \quad (3.20) \\
= \quad & u_t + \frac{\Delta t}{2} u_{tt} + \frac{\Delta t^2}{6} u_{ttt} + \mathcal{O}(\Delta t^3) + a u_x + \frac{a \Delta x}{2} u_{xx} + \frac{a \Delta x^2}{6} u_{xxx} + \mathcal{O}(\Delta x^2) \\
\overset{u_t + a u_x = 0}{=} \quad & \underbrace{\frac{\Delta t}{2} u_{tt}}_{= \mathcal{O}(\Delta t)} + \underbrace{\frac{a \Delta x}{2} u_{xx}}_{= \mathcal{O}(\Delta x)} + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)
\end{aligned}$$

The resulting assertion

$$L_{\Delta t,\Delta x}(u) \quad = \quad \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x) \tag{3.21}$$

means: If, for a smooth solution u of $u_t + a u_x = 0$ the mesh parameters Δt and Δx go to zero, then the error of the discretisation goes for $\Delta t \to 0$, $\Delta x \to 0$, with the rates Δt and Δx also to zero, respectively. By (3.21), we say that the scheme (3.11) is in time and space to the PDE $u_t + a u_x = 0$.

Let us stress, that consistency is a local assertion, stated for the point (x,t).

One usually assumes a constant coupling ratio λ between Δt and Δx:

$$\lambda \quad = \quad \frac{\Delta t}{\Delta x} \tag{3.22}$$

This implies $\lambda \cdot \Delta x = \Delta t$, i.e. $\mathcal{O}(\Delta t) = \mathcal{O}(\Delta x)$, so that we simply speak of first order consistency.

3.2 Excursion on “Big-Oh”

The Landau symbol $\mathcal{O}(\cdot)$ is defined as an equivalence class of functions.
For any given function $\varphi : \mathbb{R} \to \mathbb{R}$, we set

$$\varphi(h) = \mathcal{O}(h^p) \quad \Leftrightarrow \quad \lim_{h \to 0} \frac{\varphi(h)}{h^p} = C, \tag{3.23}$$

$C =$ constant, with (important!) $0 \leq |C| < \infty$.

Let us note the difference of (3.23) to the use of $\mathcal{O}(\cdot)$ in complexity theory. There, one is typically interested in comparing the computational effort of algorithms depending on a characteristic number n for large problems. There $\mathcal{O}(n), \mathcal{O}(n^2), \ldots,$ tells us how an algorithm performs for large n.

In contrast, we are here interested in characteristic numbers of discretisations of PDEs related to discretisation errors, i.e., we are interested in vanishing mesh widths $\Delta x, \Delta y, \Delta t \to 0$.

3.2.1 Properties of $\mathcal{O}(\cdot)$

Obviously, the definition (3.23) is not satisfied by a lot of functions φ. As $\mathcal{O}(h^p)$ can be understood as a set of functions, it seems to be appropriate to write

$$\varphi(h) \;\in\; \mathcal{O}(h^p). \tag{3.24}$$

However, the notion (3.23) is commonly used in practice. Sticking for the moment to the notation (3.24), we illustrate now further properties.

Examples

(i) Let $\varphi_1(h) = h$. Then, with $p = 1$, we obtain

$$\lim_{h\to 0} \frac{h}{h^1} \;=\; 1,$$

and with $0 \leq 1 \;\; (:= C) < \infty$ we have $\varphi_1(h) \in \mathcal{O}(h)$.

(ii) Let $\varphi_2(h) = h^2$. Then, again with $p = 1$, we obtain

$$\lim_{h\to 0} \frac{h^2}{h^1} \;=\; \lim_{h\to 0} h = 0,$$

and with $0 \leq 0 \;\; (:= C) < \infty$ we also have $\varphi_2(h) \in \mathcal{O}(h)$.

(iii) Let us consider again $\varphi_2(h)$, this time with $p = 2$:

$$\lim_{h\to 0} \frac{h^2}{h^2} \;=\; 1,$$

and with $0 \leq 1 \;\; (:= C) < \infty$ we also have $\varphi_2(h) \in \mathcal{O}(h^2)$.

Remark:
As obvious by (ii) and (iii), we have

$$\mathcal{O}(h) \supset \mathcal{O}(h^2) \supset \mathcal{O}(h^3) \supset \dots \tag{3.25}$$

(iv) Let $\varphi_3(h) = 0$. For any p, we obtain

$$\lim_{h\to 0} \frac{0}{h^p} \quad = \quad 0,$$

and with $0 \leq 0 \ \ (:= C) < \infty$, $\varphi_3(h) \in \mathcal{O}(h^p)$ for any p.

Thus, by (3.23) we have generated a complicated system of function sets! Mathematically $\mathcal{O}(h^p)$ can be understood, for any fixed p, as an ideal of a function ring.

As observable by example (iv), the zero function is a member of every ideal $\mathcal{O}(h^p)$, and thus, e.g. the operation

$$\frac{1}{\mathcal{O}(h^p)} \tag{3.26}$$

is not allowed as it can mean division by zero.

Chapter 4

Introduction to Schemes for Elliptic and Parabolic PDEs

In this paragraph, we consider the elliptic Laplace equation

$$\Delta u \quad = u_{xx} + u_{yy} \quad = 0 \tag{4.1}$$

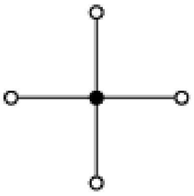
and the parabolic linear diffusion equation (or heat equation)

$$u_t \quad = \quad \Delta u \tag{4.2}$$

over a finite domain Ω. A standard discretisation of (4.1) is

$$L\left(U_{jk}\right) \;:=\; \frac{U_{j+1,k} - 2U_{jk} + U_{j-1,k}}{\Delta x^2} + \frac{U_{j,k+1} - 2U_{jk} + U_{j,k-1}}{\Delta y^2} \;=\; 0 \tag{4.3}$$

The so-called stencil (“Stempel”) of the method indicates which values participate at the point (j,k) in approximating Δu:



Stencil centered at point indexed by j,k.

Choosing $\Delta x = \Delta y$ (as usual in image processing) we can multiply (4.3) by Δx^2 to obtain

$$U_{j+1,k} - 2U_{jk} + U_{j-1,k} + U_{j,k+1} - 2U_{jk} + U_{j,k-1} \;=\; 0$$

$$\Leftrightarrow \quad U_{jk} = \frac{1}{4}(U_{j+1,k} + U_{j-1,k} + U_{j,k+1} + U_{j,k-1}) \tag{4.4}$$

We observe that U_{jk} is the arithmetic average of the neighboring values.

Theorem 4.0.1 *By scheme (4.4), a maximum or minimum of the data can only be attained at the boundary of Ω, or the discrete solution U_{jk} is constant.*

Proof. Let a finite domain be given together with a local maximum at an inner point (j, k). "Inner point" means that at $j\Delta x$, $k\Delta y$, there is not a boundary point (where typically the value is somehow prescribed). Let a strict local maximum at (j, k) be given by

$$U_{j,k} \ := \ C$$

Then formula (4.4) yields

$$C \ = \ \frac{1}{4}\underbrace{(U_{j+1,k} + U_{j-1,k} + U_{j,k+1} + U_{j,k-1})}_{<4C} \ < \ C$$

Hence, there can be no strict local extremum at any inner point. The alternative assertion of the theorem easily follows. ■

Teaser: Is this an undesired property of our method, or does this reflect a corresponding property of the discretised PDE?

A simple yet popular way to solve (4.4) consists of starting with an arbitrary first approximation $U^{(0)}_{jk}$ and iteration:

$$U^{(n+1)}_{jk} \ = \ \frac{1}{4}\left[U^{(n)}_{j+1,k} + U^{(n)}_{j-1,k} + U^{(n)}_{j,k+1} + U^{(n)}_{j,k-1}\right] \tag{4.5}$$

The method (4.5) is called Jacobi-iteration. It is easy to code (and parallelise), as one only needs to go through the list of computational points (j, k) and compute the new iteration values U^{n+1}_{jk} out of the surrounding given data. Such a formula is called explicit. However, the method (4.5) converges very slowly to the solution of (4.4).

Having thus dealt with "inner points" of a computational domain, let us also consider the boundary points in the set $\partial\Omega$. As we deal with a BVP, we have prescribed values U_{lm} for $(l\Delta x, m\Delta y) \in \partial\Omega$. The boundary values come into play if (l, m) for $(l\Delta x, m\Delta y) \in \partial\Omega$ is a neighbour of the iteration point (j, k), thus they appearing on the right hand side of (4.5).

Let us now consider $u \equiv u(x, y, t)$, and let us look at the parabolic linear diffusion equation

$$u_t \quad = \quad \Delta u. \tag{4.6}$$

Now, let $U^n_{jk} \approx u(j\Delta x, k\Delta y, n\Delta t)$, where the upper index denotes the time level ("Zeitschicht"). Then, let us approximate the time-derivative in (4.6) by a forward difference:

$$u_t(j\Delta x, k\Delta y, n\Delta t) \quad \approx \quad \frac{U^{n+1}_{jk} - U^n_{jk}}{\Delta t} \tag{4.7}$$

Evaluating $L\left(U_{jk}\right)$ at time level n, we obtain as an approximation

$$\begin{aligned} & U^{n+1}_{jk} = U^n_{jk} && + \quad \Delta t \cdot L\left(U^n_{jk}\right) \\ \Leftrightarrow \qquad & U^{n+1}_{jk} = U^n_{jk} && + \quad \frac{\Delta t}{\Delta x^2}\left(U^n_{j+1,k} - 2U^n_{jk} + U^n_{j-1,k}\right) \\ & && + \quad \frac{\Delta t}{\Delta y^2}\left(U^n_{j,k+1} - 2U^n_{jk} + U^n_{j,k-1}\right) \end{aligned} \tag{4.8}$$

Analogously to the explicitness of the Jacobi-iteration formula (4.5), the scheme (4.8) is an <u>explicit time-marching scheme</u>.

If we would approximate u_t by a backward difference, i.e.

$$u_t(j\Delta x, k\Delta y, n\Delta t) \quad \approx \quad \frac{U^n_{jk} - U^{n-1}_{jk}}{\Delta t}, \tag{4.9}$$

the resulting scheme would read

$$U^n_{jk} = U^{n-1}_{jk} \quad + \quad \Delta t \cdot L\left(U^n_{jk}\right) \tag{4.10}$$

Shifting indices $n-1 \to n$, $n \to n+1$, we obtain

$$U^{n+1}_{jk} = U^n_{jk} \quad + \quad \Delta t \cdot L\left(U^{n+1}_{jk}\right) \tag{4.11}$$

Thus, a new value U^{n+1}_{jk} depends not only on given values from time level $n\Delta t$, but also on <u>other unknowns</u> $U^{n+1}_{j\pm1,k}$, $U^{n+1}_{j,k\pm1}$. The scheme (4.10) is an example of an <u>implicit time-marching scheme</u>.

A generalisation of these two approaches is given via a parameter $\theta \in [0,1]$, defining the θ-scheme:

$$U_{jk}^{n+1} = U_{jk}^{n} \quad + \quad \Delta t \cdot \theta \cdot L\left(U_{jk}^{n}\right) + \Delta t \cdot (1-\theta) \cdot L\left(U_{jk}^{n+1}\right) \tag{4.12}$$

Accumulating the unknowns on the left hand side, one obtains a linear system of equations:

$$U_{jk}^{n+1} - \Delta t \cdot (1-\theta) \cdot L\left(U_{jk}^{n+1}\right) \quad = \quad U_{jk}^{n}\Delta t \cdot \theta \cdot L\left(U_{jk}^{n}\right) \tag{4.13}$$

There is one equation per discretisation point (j,k). This usually large system needs to be solved in order to evaluate one time step.

However, let us turn our attention back to the explicit scheme (4.8), and let us choose $\Delta x = \Delta y$ and $\Delta t := \frac{\Delta x^2}{4}$:

$$\begin{aligned}
U_{jk}^{n+1} = U_{jk}^{n} \quad &+ \quad \frac{\left(\frac{\Delta x^2}{4}\right)}{\Delta x^2}\left(U_{j+1,k}^{n} - 2U_{jk}^{n} + U_{j-1,k}^{n}\right) \\
&+ \quad \frac{\left(\frac{\Delta y^2}{4}\right)}{\Delta y^2}\left(U_{j,k+1}^{n} - 2U_{jk}^{n} + U_{j,k-1}^{n}\right) \\
= U_{jk}^{n} \quad &+ \quad \frac{1}{4}\left(U_{j+1,k}^{n} + U_{j-1,k}^{n} - 4U_{jk}^{n} + U_{j,k+1}^{n} + U_{j,k-1}^{n}\right) \\
\Leftrightarrow U_{jk}^{n+1} \quad &= \quad \frac{1}{4}\left(U_{j+1,k}^{n} + U_{j-1,k}^{n} + U_{j,k+1}^{n} + U_{j,k-1}^{n}\right) \qquad (4.14)
\end{aligned}$$

Thus, we can interprete the iteration indices (n), $(n+1)$ in the Jacobi-iteration (4.5) as time levels $n\Delta t$, $(n+1)\Delta t$. Then the Jacobi-iteration for solving the Laplace equation is identical to a specific scheme for solving the linear diffusion equation!

Technically, the solution of $\Delta u = 0$ is obtained by running $t \to \infty$ in the equation $u_t = \Delta u$, looking for so-called steady-states ("stationäre Zustände") identified by $u_t = 0$. A pointwise validity of $u_t = 0$ means, that the state u does not change anymore when going forward in time.

We obtain the following relationships:

(i) The Laplace equation can be solved by introducing a time variable t as an additional variable. The solution is retrieved as a steady-state solution of the arising diffusion equation.

(ii) Steady-state solutions of the linear diffusion equation can be computed directly by solving the Laplace-equation.

The strategy introduced in point (i) is called method of artificial time. It is frequently used for solving elliptic problems.

Having established connections, let us also stress
underlinedifferences between (4.5) and (4.14):

(i) Using (4.5) for solving an elliptic BVP, one may use arbitrary initial data $U_{jk}^{(0)}$. Moreover, the iterates are meaningless, only the steady-state solution is meaningful.

(ii) In contrast, (4.14) solves the linear diffusion equation. The initial data U_{jk}^{0} are usually <u>not</u> arbitrary, and the iterates U_{jk}^{n} are of interest.

Conclusion. Parabolic and elliptic processes are conceptually very different, but there exist important connections that are useful for computations.

Chapter 5

Elliptic PDEs: The Maximum Principle

Motivation:

- Is there an important property of solutions of elliptic PDEs?
- Can we establish a connection between analytical and discrete world?

We consider the Laplace equation

$$\Delta u \;=\; 0, \tag{5.1}$$

as well as its extension, the Poisson equation

$$\Delta u \;=\; f, \tag{5.2}$$

where f is a given function.

Theorem 5.0.2 *Let Ω be a connected, finite and open domain. Let $u(x,y)$ solving (5.1) in Ω be continuous in $\overline{\Omega} = \Omega \cup \partial\Omega$. Then the minimal and the maximal value of u are situated on $\partial\Omega$, otherwise u is constant.*

Proof. We only consider the case of a local maximum, the case of a local minimum follows analogously. For the proof, let $\vec{x} := (x,y)$. The maximum principle states, that there exist points $\vec{x}_m$ and $\vec{x}_M$ on $\partial\Omega$, such that

$$u(\vec{x}_m) \;\le u(\vec{x}) \;\le u(\vec{x}_M) \tag{5.3}$$

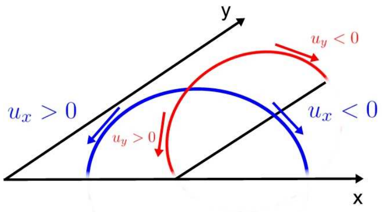
holds for all $\vec{x} \in \Omega$. We prove the non-existence of inner maxima by means of a "proof by contradiction". Therefore, assume that there exists a maximum of u in Ω. Then, it would hold there

$$u_{xx} \le 0 \quad \text{as well as} \quad u_{yy} \le 0$$

by the standard test on the second derivatives in calculus. Combined, this gives us

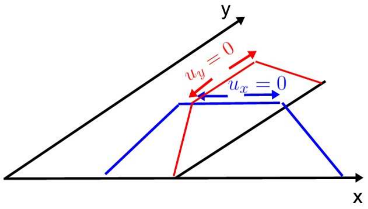
$$u_{xx} + u_{yy} \qquad (= \Delta u) \leq 0.$$

At isolated maxima it holds $u_{xx} < 0$ *and* $u_{yy} < 0$.
This case directly yields a contradiction with (5.1), however, let us also consider



Sketch.

non-strict maxima:

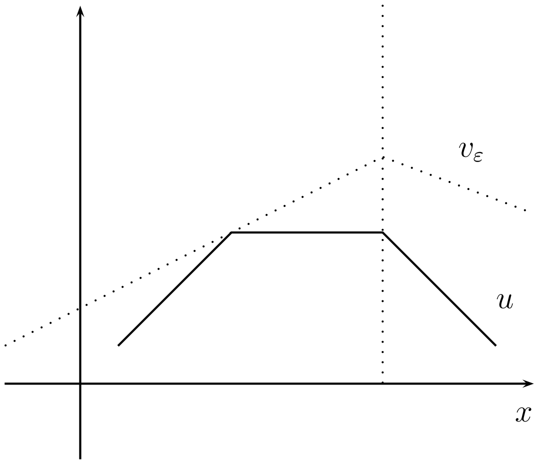


Sketch.

Therefore, let $\varepsilon > 0$, and we define the auxiliary function

$$v_\varepsilon(\vec{x}) \quad := \quad u(\vec{x}) + \varepsilon \|\vec{x}\|_2^2, \qquad \|\vec{x}\|_2^2 = x^2 + y^2. \tag{5.4}$$

Let us briefly comment on $v_\varepsilon(\vec{x})$. By (5.4), it is clear that we always add a non-negative contribution to $u(\vec{x})$. A non-strict maximum of u will be converted to a monotone part of v_ε: Especially, we may choose ε small enough, so that u decays faster than $\varepsilon\|\vec{x}\|_2^2$ grows. This leads to the existence of a strict maximum of v_ε, see sketch.



At a non-strict maximum of u,

$$\begin{aligned}
\Delta v_\varepsilon &= \Delta u + \Delta(\varepsilon \|\vec{x}\|_2^2) \\
&= \underbrace{\Delta u}_{\substack{=0 \text{ at a non-strict} \\ \text{maximum}}} + \varepsilon \Delta(x^2 + y^2) \\
&= \varepsilon \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) (x^2 + y^2) \\
&= \varepsilon \frac{\partial^2}{\partial x^2}(x^2 + y^2) + \varepsilon \frac{\partial^2}{\partial y^2}(x^2 + y^2) \\
&= \varepsilon \cdot 2 + \varepsilon \cdot 2 = 4\varepsilon > 0
\end{aligned}$$

This means $\Delta v_\varepsilon > 0$ in Ω. But, as $\Delta v_\varepsilon \leq 0$ must hold at a maximum of v_ε in Ω, v_ε cannot have a maximum in Ω. This is in contradiction to the construction of v_ε, as discussed above. Thus, the assumption that u has a non-strict maximum in Ω must be wrong.

It remains to show, that there exists a maximum of u on $\partial\Omega$. However, this follows immediately as u is assumed to be continuous, and thus it must have a maximum in $\overline{\Omega} = \Omega \cup \partial\Omega$: As Ω is excluded, $\partial\Omega$ remains for candidates. ■

Can we carry over this result to the discrete world?

Theorem 5.0.3 *If*

$$-\frac{U_{i+1,j} - 2U_{ij} + U_{i-1,j}}{\Delta x^2} - \frac{U_{i,j+1} - 2U_{ij} + U_{i,j-1}}{\Delta y^2} = -L\left(U_{ij}\right) \leq 0 \qquad (5.5)$$

on Ω, then the maximum value of U_{ij} is attained on the boundary points $\partial\Omega$.

Remark:
We may relate the sign of $L\left(U_{ij}\right)$ to the sign of a given function f within the Poisson equation.

Proof. $L\left(U_{ij}\right) \leq 0$ can be rewritten as

$$\frac{U_{ij}}{\Delta x^2} + \frac{U_{ij}}{\Delta y^2} \leq \frac{1}{2}\left(\frac{U_{i+1,j}+U_{i-1,j}}{\Delta x^2} + \frac{U_{i,j+1}+U_{i,j-1}}{\Delta y^2}\right) \tag{5.6}$$

Now, let U_{ij} be a local maximum, i.e.,

$$U_{ij} \geq \{U_{i+1,j}, U_{i-1,j}, U_{i,j+1}, U_{i,j-1}\} \tag{5.7}$$

Using (5.7) to estimate the right hand side of (5.6) gives:

$$\frac{1}{2}\left(\frac{U_{i+1,j}+U_{i-1,j}}{\Delta x^2} + \frac{U_{i,j+1}+U_{i,j-1}}{\Delta y^2}\right) \leq \frac{1}{2}\left(\frac{2U_{ij}}{\Delta x^2} + \frac{2U_{ij}}{\Delta y^2}\right) = \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)U_{ij}, \tag{5.8}$$

so that

$$\underbrace{\frac{U_{ij}}{\Delta x^2} + \frac{U_{ij}}{\Delta y^2}}_{\text{left hand side of (5.6)}} \overset{(5.8)}{\leq} \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)U_{ij} \tag{5.9}$$

Since the left and right hand side of (5.9) are the same, all inequalities are equalities. Now, let us think of (5.7) as a "3-out-of-4" procedure. First, assume $U_{ij} \geq \{U_{i-1,j}, U_{i,j+1}, U_{i,j-1}\}$. Then analogously to (5.8) we can compute using "=" instead of "$\leq$" (as this is already established):

$$\frac{1}{\Delta x^2}U_{ij} = \frac{1}{2}\left(\frac{1}{\Delta x^2}U_{i+1,j} + \frac{1}{\Delta x^2}U_{ij}\right), \tag{5.10}$$

so that $U_{i+1,j} = U_{ij}$. In the same manner, we can apply this procedure at the other values in (5.7), and thus

$$U_{ij} = \{U_{i+1,j}, U_{i-1,j}, U_{i,j+1}, U_{i,j-1}\}. \tag{5.11}$$

Hence, if U_{ij} is a local maximum in Ω, then the discrete solution U_{ij} is constant. Consequently, it cannot have a local maximum in Ω. ■

Analogously, one can prove a discrete minimum principle follows analogously for $L\left(U_{ij}\right) \geq 0$.

Conclusion:
Continuous-scale and discrete world are connected w.r.t. the maximum principle. The discrete result can be viewed as the translation of the analytical result to the discrete world. However, the techniques to establish the corresponding results are very different and do not correspond.

Chapter 6

Elliptic PDEs: Uniqueness

Motivation:

- Can we expect a unique solution to solve for numerically?
- Setting up an iterative method, do there exist several solutions depending on the initial iteration state?

By the maximum principle established in Section §5 the uniqueness of a solution follows. To see this, we define the BVPs:

$$\begin{cases} \Delta u = f & \text{in } \Omega \\ u = \varphi & \text{on } \partial\Omega \end{cases} \tag{6.1}$$

and

$$\begin{cases} \Delta v = f & \text{in } \Omega \\ v = \varphi & \text{on } \partial\Omega \end{cases} \tag{6.2}$$

We want to show $u \equiv v$ in Ω. Therefore, we set $w := u - v$ and compute

$$\Delta w \quad = \quad \Delta(u - v) = \Delta u - \Delta v = f - f = 0$$

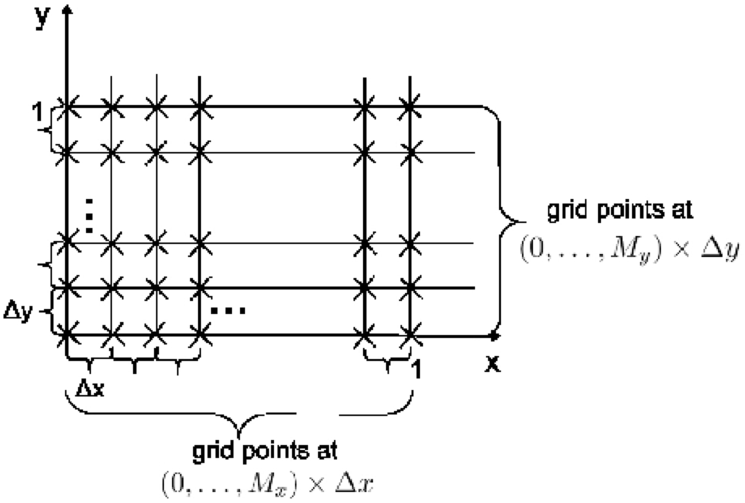
Thus, $\Delta w = 0$ in Ω. On the boundary $\partial\Omega$, we have $w = u - v = \varphi - \varphi = 0$. Let $\vec{x} = (x, y)$. By the maximum principle, there exist points $\vec{x}_m$ and $\vec{x}_M$ on $\partial\Omega$ with

$$0 \quad = \quad w(\vec{x}_m) \leq w(\vec{x}) \leq w(\vec{x}_M) = 0 \qquad \text{for all } x \in \Omega$$

Thus $w \equiv 0$ and $u \equiv v$.

We see, that the analytical uniqueness result is easily established. Can we translate it to the discrete world?

To make this issue concrete, we consider a “test-grid” on $[0,1] \times [0,1]$:


y
1
Δy
grid points at
$(0,\ldots,M_y) \times \Delta y$
Δx
1
x
grid points at
$(0,\ldots,M_x) \times \Delta x$

and approximate (6.1) by use of our standard approximation:

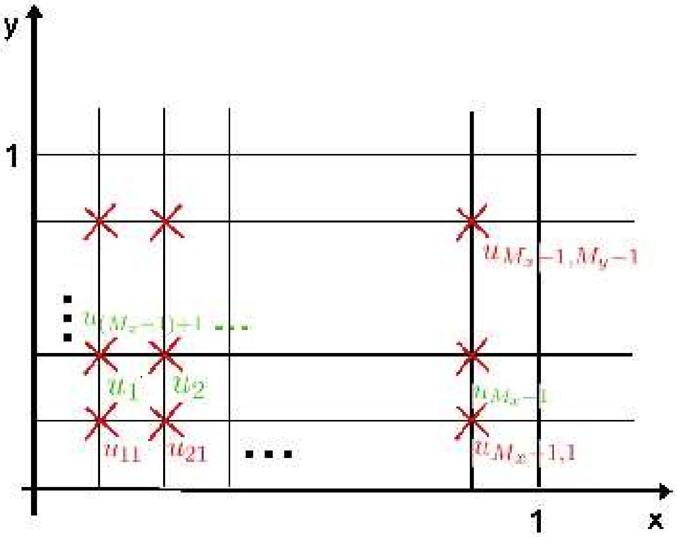
$$\begin{aligned}
\frac{U_{i+1,j}-2U_{ij}+U_{i-1,j}}{\Delta x^2} + \frac{U_{i,j+1}-2U_{ij}+U_{i,j-1}}{\Delta y^2} &= f_{ij} && \text{(6.3a)}\\
\text{for } i=1,\ldots,M_x-1,\ j=1,\ldots,M_y-1, & &&\\
U_{0,j} = \varphi_{0,j}, \qquad j=1,\ldots,M_y-1, & && \text{(6.3b)}\\
U_{M_x,j} = \varphi_{M_x,j}, \qquad j=1,\ldots,M_y-1, & && \text{(6.3c)}\\
U_{i,0} = \varphi_{i,0}, \qquad i=1,\ldots,M_x-1, & && \text{(6.3d)}\\
U_{i,M_y} = \varphi_{i,M_y}, \qquad i=1,\ldots,M_x-1. & && \text{(6.3e)}
\end{aligned}$$

We ask for the related issues of existence and uniqueness of a solution of (6.3a), as the solvability of (6.3a) is of computational importance.

We discuss two methods that can be used to ensure that (6.3a) has a unique solution. Observe that (6.3a) is a linear system of equations that can be written as

$$Ax = g, \qquad A \in \mathbb{R}^{L\times L}, \qquad x, g \in \mathbb{R}^L \tag{6.4}$$

If we could show, that A is invertible, then the existence and uniqueness of the solution of the discrete problem follows. If we order the unknowns U_{ij} in lexicographical order (as a remark, often this ordering begins top left and follows the ordering of words as read in books; we start here bottom left)



and put them into a vector

$$\left(U_{11}, \dots, U_{M_x-1,1}, U_{12}, \dots, U_{M_x-1,2}, \dots, U_{M_x-1,M_y-1}\right)^{\top} \tag{6.5}$$

the matrix A from (6.4) can be written in block form:

$$A = \begin{pmatrix} B & \frac{1}{\Delta y^2} I & & \underline{0} \\ \frac{1}{\Delta y^2} I & B & \ddots & \\ & \ddots & \ddots & \frac{1}{\Delta y^2} I \\ \underline{0} & & \frac{1}{\Delta y^2} I & B \end{pmatrix} \tag{6.6}$$

with $(M_y-1) \times (M_y-1)$ blocks, where B is the $(M_x-1) \times (M_x-1)$ matrix

$$B = \begin{pmatrix} -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) & \frac{1}{\Delta x^2} & & \underline{0} \\ \frac{1}{\Delta x^2} & \ddots & \ddots & \frac{1}{\Delta x^2} \\ \underline{0} & & \frac{1}{\Delta x^2} & -2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \end{pmatrix}, \tag{6.7}$$

and I is the $(M_x-1) \times (M_x-1)$ identity matrix.

Remark: Any zeros along the super- and subdiagonals in A are due to the boundary conditions at $x = 0$ and $x = 1$.

The first method to discuss (6.4) involves the assumption that A is positive definite.

Definition 6.0.1 *A matrix A is positive definite if $x^\top Ax > 0$ for all vectors $x \neq \underline{0}$.*

Remark:
An important special assertion valid for symmetric matrices A is: A is positive definite if and only if all eigenvalues of A are larger than zero.

One can easily show:

Corollary 6.0.1 *If A is positive definite, then A is invertible.*

It is not difficult to see that A from (6.6), (6.7) is symmetric, however, to prove that A is positive definite involves more mathematics and is much more difficult (but feasible).

The second method relies on the diagonal dominance of the matrix.

Definition 6.0.2 *$A = (a_{kl})$ is diagonally dominant respectively strictly diagonally dominant, if*

$$|a_{kk}| \geq \sum_{\substack{l=1\\l\neq k}}^{L} |a_{kl}| \quad =: \quad p_k \qquad \textit{for all } k = 1, \dots, L, \tag{6.8}$$

respectively

$$|a_{kk}| > \sum_{\substack{l=1\\l\neq k}}^{L} |a_{kl}| \quad =: \quad p_k \qquad \textit{for all } k = 1, \dots, L \tag{6.9}$$

holds.

We then obtain the following important result:

Corollary 6.0.2 *If A is strictly diagonally dominant, then A is invertible.*

The bad news is, that we cannot use Corollary 6.0.2 on our model problem. In the matrix A, though many of the rows are strictly diagonally dominant, a large number (rows associated with the interior points) are only diagonally dominant.

Let us make the following definition.

Definition 6.0.3 *The $L \times L$-matrix A is* <u>*reducible*</u>*, if either*

1. *$L = 1$ and $A = \underline{0}$, or*

2. *$L \geq 2$, and there exists a permutation matrix P and some integer r, $1 \leq r \leq L$, such that*

$$P^\top AP = \begin{pmatrix} B & C \\ \underline{0} & D \end{pmatrix},$$

where B is $r \times r$. D is $(L-r) \times (L-r)$, C is $r \times (L-r)$, and $\underline{0}$ is the $(L-r) \times r$ zero matrix. The matrix A is irreducible, if it is not reducible.

Obviously, the above information is not palatable. A more convenient characterisation in terms of systems $Ax = b$ is: A matrix A is irreducible if a change in <u>any</u> of the components of b will cause a change in the solution x.

<u>Remark:</u>
To obtain a reducible matrix in a finite difference setting means to solve a problem that can be seperated into two (or more) problems.

The assertion that we want is as follows:

Corollary 6.0.3 *If A is an irreducible diagonally dominant matrix for which $|a_{kk}| > p_k$ holds for at least one k, then A is invertible.*

<u>Result:</u>
The matrix A in (6.7) is invertible.

<u>Conclusion:</u>
It can be difficult to translate uniqueness results from the continuous-scale world to the discrete world. Also, in the discrete setting, the recommended techniques strongly rely on the chosen discretisation method.

Chapter 7

Isotropic Discretisations

We now adress the rotational invariance of PDEs and their numerical discretisations. The often encountered word isotropy has the following greek origins:

$$\begin{aligned} \text{iso} &\equiv \text{in the same way,} \\ \text{trop} &\equiv \text{rotated or directed} \end{aligned}$$

The contrary of isotropy is anisotropy, referring to a "structure" showing a prefered direction.

We consider the Laplace equation

$$\Delta u \;=\; 0, \tag{7.1}$$

the solution of which are called harmonic functions. We will rely on the following assertion (without proof).

Theorem 7.0.4 *Let u be a harmonic function in $\Omega \subset \mathbb{R}^2$. Then u possesses in Ω partial derivatives of arbitrary order.*

The point is, we can rely on the techinque of Taylor series expansions of u without need to consider their existence.

7.1 Rotational invariance of the Laplace operator

In the plane, a rotation by an angle α is described by

$$\begin{aligned} x' &= x \cdot \cos\alpha + y \cdot \sin\alpha \\ y' &= -x \cdot \sin\alpha + y \cdot \cos\alpha. \end{aligned} \tag{7.2}$$

By use of the chain rule, we compute

$$\begin{aligned}
u_x(x',y') &= \frac{\partial u}{\partial x'}\cdot\frac{\partial x'}{\partial x}+\frac{\partial u}{\partial y'}\cdot\frac{\partial y'}{\partial y}\\
u_x(x',y') &= u_{x'}\cdot\cos\alpha - u_{y'}\cdot\sin\alpha,\\
u_y(x',y') &= u_{x'}\cdot\sin\alpha + u_{y'}\cdot\cos\alpha,\\
u_{xx}(x',y') &= (u_{x'}\cdot\cos\alpha - u_{y'}\cdot\sin\alpha)_{x'}\cdot\cos\alpha - (u_{x'}\cdot\cos\alpha - u_{y'}\cdot\sin\alpha)_{y'}\cdot\sin\alpha\\
u_{yy}(x',y') &= (u_{x'}\cdot\sin\alpha + u_{y'}\cdot\cos\alpha)_{x'}\cdot\sin\alpha + (u_{x'}\cdot\sin\alpha + u_{y'}\cdot\cos\alpha)_{y'}\cdot\cos\alpha
\end{aligned}$$

Adding the last two terms together yields

$$\begin{aligned}
u_{xx}+u_{yy} &= [u_{x'x'}\cos^2\alpha - u_{y'x'}\sin\alpha\cos\alpha - u_{x'y'}\cos\alpha\sin\alpha + u_{y'y'}\sin^2\alpha]\\
&+ [u_{x'x'}\sin^2\alpha + u_{y'x'}\cos\alpha\sin\alpha + u_{x'y'}\sin\alpha\cos\alpha + u_{y'y'}\cos^2\alpha]\\
&= (u_{x'x'}+u_{y'y'})(\underbrace{\cos^2\alpha+\sin^2\alpha}_{=1}) = \underline{u_{x'x'}+u_{y'y'}}
\end{aligned}$$

Thus, the Laplace operator does not prefer any one direction over the others: An isotropic situation is modeled.

Is it possible to identify isotropy in numerical schemes?

7.2 Anatomy of a numerical discretisation

For discretising $\Delta u = u_{xx}+u_{yy}=0$, let us consider

$$u_{xx} \approx \frac{U_{i+1,j}-2U_{ij}+U_{i-1,j}}{h^2}, \tag{7.3}$$

$$u_{yy} \approx \frac{U_{i,j+1}-2U_{ij}+U_{i,j-1}}{h^2}, \tag{7.4}$$

where $h=\Delta x=\Delta y$. Summarising (7.3),(7.4) we obtain

$$\frac{1}{h^2}(U_{i+1,j}+U_{i-1,j}+U_{i,j+1}+U_{i,j-1}-4U_{ij}) = 0 \tag{7.5}$$

We now employ the Taylor series expansions to assess the local truncation error scheme (7.5), writing $u\equiv U_{ij}$.

$$\begin{aligned}
U_{i+1,j} &= u+hu_x+\frac{h^2}{2}u_{xx}+\frac{h^3}{6}u_{xxx}+\frac{h^4}{24}u_{xxxx}+\frac{h^5}{120}u_{xxxxx}+\mathcal{O}(h^6) && (7.6a)\\
U_{i-1,j} &= u-hu_x+\frac{h^2}{2}u_{xx}-\frac{h^3}{6}u_{xxx}+\frac{h^4}{24}u_{xxxx}-\frac{h^5}{120}u_{xxxxx}+\mathcal{O}(h^6) && (7.6b)\\
U_{i,j+1} &= u+hu_y+\frac{h^2}{2}u_{yy}+\frac{h^3}{6}u_{yyy}+\frac{h^4}{24}u_{yyyy}+\frac{h^5}{120}u_{yyyyy}+\mathcal{O}(h^6) && (7.6c)\\
U_{i,j-1} &= u-hu_y+\frac{h^2}{2}u_{yy}-\frac{h^3}{6}u_{yyy}+\frac{h^4}{24}u_{yyyy}-\frac{h^5}{120}u_{yyyyy}+\mathcal{O}(h^6) && (7.6d)
\end{aligned}$$

Plugging (7.6) into (7.5) yields

$$\Delta u + \frac{h^2}{12}(u_{xxxx} + u_{yyyy}) + \mathcal{O}(h^4) \quad = \quad 0. \tag{7.7}$$

The leading order error term in (7.7), i.e.,

$$\frac{h^2}{12}(u_{xxxx} + u_{yyyy}), \tag{7.8}$$

has a directional bias! This can be seen as follows. We know from §7.1, that the functions do not feature a directional preference. Setting

$$\Psi \quad := \quad \Delta u, \tag{7.9}$$

Ψ is a harmonic function without directional bias. As it is in C^∞ by Theorem 7.0.4, we can apply the Laplace operator at it yielding

$$\begin{aligned} \Delta\Psi &= \Delta(\Delta u) \\ &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u \\ &= \frac{\partial^4 u}{\partial x^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} + \frac{\partial^4 u}{\partial y^4} \\ &\neq u_{xxxx} + u_{yyyy}. \end{aligned}$$

Thus, due to the missing mixed derivatives $2\frac{\partial^4 u}{\partial x^2 \partial y^2}$ the term (7.8) incorporates a directional preference.

7.3 Isotropic second derivatives

The idea behind isotropic numerical discretisations is to remove a directional bias in the largest part of the numerical error. The conventional central difference ("C") discretisation of u_{xx} is given by

$$(u_{xx_C})_{ij} \quad = \quad \frac{1}{h^2}(U_{i+1,j} - 2U_{ij} + U_{i-1,j}), \tag{7.10}$$

which can be modeled, to leading order in the error term, as

$$(u_{xx_C})_{ij} \quad \equiv \quad \left(1 + \frac{h^2}{12}\frac{\partial^2}{\partial x^2}\right) u_{xx}. \tag{7.11}$$

In contrast, by the methodology of §7.2 the isotropic discretisation of u_{xx}, $(u_{xx_I})_{ij}$, is obtained from

$$(u_{xx_I})_{ij} \equiv \left(1+\frac{h^2}{12}\Delta\right) u_{xx}. \tag{7.12}$$

Taking into account the factorisation

$$\begin{aligned} & \left(1+\frac{h^2}{12}\frac{\partial^2}{\partial y^2}\right)\left(1+\frac{h^2}{12}\frac{\partial^2}{\partial x^2}\right) \\ = \; & 1+\frac{h^2}{12}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)+\frac{h^4}{144}\frac{\partial^4}{\partial x^2\partial y^2} \\ = \; & 1+\frac{h^2}{12}\Delta+\frac{h^4}{144}\frac{\partial^4}{\partial x^2\partial y^2}, \end{aligned} \tag{7.13}$$

we observe by (7.11) and (7.12) that with an error $\mathcal{O}(h^4)$ we can write

$$(u_{xx_I})_{ij} \equiv \left(1+\frac{h^2}{12}\frac{\partial^2}{\partial y^2}\right)(u_{xx_C})_{ij}, \tag{7.14}$$

which gives the discretisation

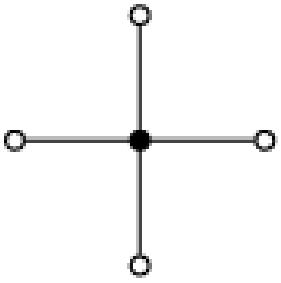
$$\begin{aligned} (u_{xx_I})_{ij} &= (u_{xx_C})_{ij}+\frac{h^2}{12}\frac{\partial^2}{\partial y^2}(u_{xx_C})_{ij} \\ &= \frac{1}{h^2}(U_{i+1,j}-2U_{ij}+U_{i-1,j}) \\ &\quad +\frac{h^2}{12}\left[\frac{1}{h^2}\left((u_{xx_C})_{i,j+1}-2(u_{xx_C})_{ij}+(u_{xx_C})_{i,j-1}\right)\right] \\ &= \frac{1}{h^2}(U_{i+1,j}-2U_{ij}+U_{i-1,j}) \\ &\quad +\frac{1}{12}\left[\frac{1}{h^2}(U_{i+1,j+1}-2U_{i,j+1}+U_{i-1,j+1})-\frac{2}{h^2}(U_{i+1,j}-2U_{ij}+U_{i-1,j})\right. \\ &\qquad \left.+\frac{1}{h^2}(U_{i+1,j-1}-2U_{i,j-1}+U_{i-1,j-1})\right] \\ &= \frac{1}{h^2}\left[\frac{10}{12}(U_{i+1,j}-2U_{ij}+U_{i-1,j})+\frac{1}{12}(U_{i+1,j+1}-2U_{i,j+1}+U_{i-1,j+1})\right. \\ &\qquad \left.+\frac{1}{12}(U_{i+1,j-1}-2U_{i,j-1}+U_{i-1,j-1})\right] \end{aligned} \tag{7.15}$$

7.4 The isotropic numerical Laplace operator

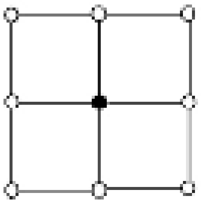
The isotropic numerical Laplace operator is obtained from the isotropic discretisations of u_{xx} and u_{yy}:

$$\Delta_I u = (u_{xx_I}) + (u_{yy_I}). \tag{7.16}$$

Such isotropic discretisations usually incorporate more computational nodes than simple schemes. For example, our "simple" scheme from (7.5) has a 5-point stencil:



In contrast, the isotropic discretisation (7.16) has a 9-point stencil:



Intuitively, this is clear, since an isotropic discretisation should incorporate points from all directions.

Chapter 8

Iterative Solvers for Linear Systems: Introduction

Motivation:
Many models lead to linear systems of equations that need to be solved, e.g. optic flow, or PDE-based compression.

Typical properties of arising systems $Ax = b$ are:

(a) A is inexact. Often, it is given by a discretisation of a PDE.

(b) b is inexact. Often, it is directly given by noisy image data.

(c) A is often sparse, i.e. only a small number of entries relative to the matrix dimension are non-zero. Usually, only these values are stored.

Consequences:

- We aim only for an approximate solution, as an exact solution will not be of better quality.
- We aim only for a method working with little more than the stored entries.

It turns out that iterative methods are good for doing the job.

8.1 Mathematical Preliminaries

We can only compile here the most important assertions we use. We will rely on the notion of the spectral radius and consider complex numbers.

Definition 8.1.1 *Let $B \in \mathbb{C}^{n \times n}$ be a matrix.*

(i) *A complex number $\lambda \in \mathbb{C}$ is called eigenvalue of B, if there exists a vector $x \neq 0$, $x \in \mathbb{C}^n$, with $Bx = \lambda x$.*

(ii) *This vector x is called eigenvector corresponding to λ.*

(iii) *The set $\sigma(B) = \{\lambda : \ \lambda$ is eigenvalue of $B\}$ is called spectrum of B.*

(iv) *The number $\rho(B) = \max\{|\lambda| : \ \lambda \in \sigma(B)\}$ is called spectral radius of B.*

We also rely on vector norms

Definition 8.1.2 *Let X be the $\mathbb{C}^n$, i.e. a complex linear space. A mapping $\|.\| : X \to \mathbb{R}$ with the properties, for $x \in \mathbb{C}^n$,*

($N1$) $\|x\| \geq 0$ *(positivity)*

($N2$) $\|x\| = 0 \Leftrightarrow x = 0$ *(definiteness)*

($N3$) $\|\alpha \cdot x\| = |\alpha| \cdot \|x\| \quad \forall x \in X, \forall \alpha \in \mathbb{C}$ *(homogenity)*

($N4$) $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in X$ *(triangle inequality)*

is called a norm on X.

Examples are

$$\|x\|_1 := \sum_{i=1}^{n} |x_i|, \tag{8.1}$$

$$\|x\|_2 := \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}, \tag{8.2}$$

$$\|x\|_\infty := \max_{i=1,\ldots,n} |x_i|. \tag{8.3}$$

We also require a convergence notion.

Definition 8.1.3 *A sequence $\{x_n\}_{n \in \mathbb{N}}$ of elements in $\mathbb{C}^n$ is convergent to the limit element $x \in \mathbb{C}^n$, if there exists for any $\varepsilon > 0$ a natural number $N = N(\varepsilon)$ with*

$$\|x_n - x\| < \varepsilon \qquad \forall n \geq N. \tag{8.4}$$

A sequence, which is not convergent, is divergent.

We now compile some important assertions on norms, convergent sequences and their relationship.

(i) If $\{x_n\}$ is a convergent sequence, then the limit element is unique.

(ii) Two norms $\|.\|_a$ and $\|.\|_b$ are equivalent, if the following assertion holds: Any sequence converges with respect to the norm $\|.\|_a$, if and only if it converges with respect to $\|.\|_b$.

(iii) For one and the same sequence $\{x_n\}$, the limit elements determined with respect to equivalent norms are identical.

(iv) On a finite-dimensional space like $\mathbb{C}^n$ or $\mathbb{R}^n$, all norms are equivalent.

We now generalise the vector norms to matrix norms by using the concept of

Definition 8.1.4 *If $B \in \mathbb{C}^{n\times n}$ and $\|.\|_a : \mathbb{C}^n \to \mathbb{R}$ is a norm, then*

$$\|B\|_a \quad := \quad \sup_{\|x\|_a=1} \|Bx\|_a \tag{8.5}$$

is the matrix norm induced by the vector norm $\|.\|_a$.

Examples are

$$\|B\|_1 \quad = \quad \max_{k=1,\dots,n} \sum_{i=1}^{n} |b_{ik}| \tag{8.6}$$

$$\|B\|_\infty \quad = \quad \max_{i=1,\dots,n} \sum_{k=1}^{n} |b_{ik}| \tag{8.7}$$

$$\|B\|_2 \quad \leq \quad \left(\sum_{i,k=1}^{n} |b_{ik}|^2\right)^{\frac{1}{2}} = \|B\|_F. \tag{8.8}$$

The norm $\|.\|_F$ is called Frobenius-norm, and it is not an induced norm.

One can prove the following assertion:

Corollary 8.1.1 *Let $B \in \mathbb{C}^{n\times n}$ and B^* be its adjoint matrix (transpose with complex conjugated entries), then*

$$\|B\|_2 \quad = \quad \sqrt{\rho(B^*B)}. \tag{8.9}$$

Remark:
For the special case of real, symmetric matrices, we obtain by (8.9) the relation $\|B\|_2 = \rho(B)$.

One of the central assertions we finally need is:

Corollary 8.1.2 *For all matrices B and all $\varepsilon > 0$, there exists a norm $\|.\|$ with*

$$\rho(B) \leq \|B\| \leq \rho(B) + \varepsilon. \tag{8.10}$$

Remark:
In general, convergence assertions not relying on a specific norm can be directly transfered to the spectral radius.

8.2 The Fix-Point-Theorem of Banach

Many iterative schemes for solving $Ax = b$ can be written in the fashion

$$x_{n+1} \quad = \quad F(x_n), \qquad n = 0, 1, 2, \ldots. \tag{8.11}$$

Two important notions are given as follows

Definition 8.2.1 *An element x in $D \subset X$ is called fixpoint of the operator $F : D \subset X \to X$, if*

$$F(x) \quad = \quad x. \tag{8.12}$$

Definition 8.2.2 *An operator $F : D \subset X \to X$ is called contracting (on D), if a number q, $0 \leq q < 1$, exists with*

$$\|F(x) - F(y)\| \quad \leq \quad q\|x - y\| \qquad \forall x, y \in D. \tag{8.13}$$

The number q is the contraction number of F.

One can prove:

Corollary 8.2.1 *Contracting operators do have at most one fix point.*

Proof. Let $x, y \in D$ be two fix points of F, then

$$\|x - y\| = \|F(x) - F(y)\| \leq q\|x - y\|,$$

so that by

$$\begin{array}{rll} & \|x - y\| & \leq q\|x - y\| \\ \Leftrightarrow & \underbrace{(1 - q)}_{\geq 0}\underbrace{\|x - y\|}_{\geq 0} & \leq 0 \end{array}$$

the relation $\|x - y\| = 0$ follows. By (N2), see Def. 8.1.2, holds $x = y$. ■

This culminates to

Theorem 8.2.1 (Theorem of Banach) *Let $F : D \to D$ be a contracting operator, then there exists exactly one fix point $x \in D$ of F, and the sequence defined by $x_{n+1} = F(x_n)$, $n = 0, 1, 2, \ldots$, converges for all initial vectors $x_0 \in D$ to x. Moreover, we obtain the following error estimates:*

$$\begin{aligned} \|x_n - x\| &\leq \frac{q^n}{1-q}\|x_1 - x_0\| \qquad \textit{(a priori)}, \\ \|x_n - x\| &\leq \frac{q}{1-q}\|x_n - x_{n-1}\| \qquad \textit{(a posteriori)}, \end{aligned}$$

where q is the contraction number of F.

Outlook. We will need the important theoretical results from §8 to establish convergence for iterative solvers, and to quantify how efficient they are.

Chapter 9

Iterative Solvers for Linear Systems: Basic Theory

We consider again the linear system of equations

$$Ax \;=\; b \tag{9.1}$$

where

- $b \in \mathbb{C}^n$ is a given right hand side (RHS),
- $A \in \mathbb{C}^{n\times n}$ is regular.

Iterative methods successively compute approximations x_m of the exact solution $A^{-1}b$ via

$$x_{m+1} \;=\; \phi(x_m, b) \qquad \text{for } m = 0, 1, \ldots, \tag{9.2}$$

for a given initial vector $x_0 \in \mathbb{C}^n$.

We deal with so-called linear schemes.

Definition 9.0.3 *An iterative scheme given by the mapping*

$$\phi : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n \tag{9.3}$$

is called linear, if matrices $M, N \in \mathbb{C}^{n\times n}$ exist, so that ϕ can be written as

$$\phi(x, b) \;=\; Mx + Nb. \tag{9.4}$$

The matrix M is called the iteration matrix of ϕ.

Remark: Note, that M and N are uniquely determined by ϕ.

We aim for constructing a sequence x_m approaching $A^{-1}b$. Thus, we define:

Definition 9.0.4 *A vector $\tilde{x} \in \mathbb{C}^n$ is the fixed point of the iterative method ϕ corresponding to $b \in \mathbb{C}^n$, if*

$$\tilde{x} \quad = \quad \phi(\tilde{x}, b). \tag{9.5}$$

We now come to two very important notions.

Definition 9.0.5 (*i*) *An iterative scheme ϕ is consistent to A if, for all $b \in \mathbb{C}^n$, the solution $A^{-1}b$ is a fixed point of ϕ corresponding to b.*

(*ii*) *An iterative scheme ϕ is convergent, if, for all $b \in \mathbb{C}^n$ and all initial vectors $x_0 \in \mathbb{C}^n$, a limit element*

$$\hat{x} \quad = \quad \lim_{m\to\infty} x_m = \lim_{m\to\infty} \phi(x_{m-1}, b) \tag{9.6}$$

exists.

Remarks:

(i) By definition, the limit element $\hat{x}$ shall arise independently of the choice of x_0.

(ii) Consistency is a necessary property of ϕ, as otherwise there is no link between ϕ and $Ax = b$.

(iii) Convergence means that a fixed point exists. Without consistency, this could be anywhere, having nothing to do with $A^{-1}b$.

Can we quantify conditions showing consistency and convergence?

Theorem 9.0.2 *A linear iterative scheme is consistent to A, if and only if*

$$M \quad = \quad I - NA \tag{9.7}$$

holds, where I is the identity Matrix.

Proof. Let $x^* = A^{-1}b$. We show both implication directions of the theorem, thus proving the equivalence assertion ("if and only if").

"$\Rightarrow$" Assumption: Let ϕ be consistent to A. Then

$$x^* = \phi(x^*, b) = Mx^* + Nb = Mx^* + NAx^*, \tag{9.8}$$

and thus

$$Mx^* = x^* - NAx^* = (I - NA)x^*, \tag{9.9}$$

so that $M = I - NA$.

"$\Leftarrow$" Assumption: $M = I - NA$.
Then, by $I = M + NA$

$$x^* = Ix^* = Mx^* + N \underbrace{Ax^*}_{=b} = Mx^* + Nb = \phi(x^*, b). \tag{9.10}$$

By $\phi(x^*, b) = x^*$ follows that x^* is a fixed point. As we did not impose any property on b, $\phi(x^*, b) = x^*$ holds for any b, and thus the consistency of ϕ to A follows. ■

Remark: Does consistency suffice as a condition to define a method ϕ that makes sense?
No, choose: $M = I, N = 0$, then we obtain $\phi(x, b) = x$. Then, for $x^* = A^{-1}b$ we have that x^* is a fixed point, but this also holds for any other x.

Theorem 9.0.3 *A linear iterative scheme ϕ is convergent, if and only if the spectral radius of the iteration matrix M satisfies the condition*

$$\rho(M) \quad < \quad 1. \tag{9.11}$$

Proof. "$\Rightarrow$" Assumption: Let ϕ be convergent.
Let λ be the eigenvalue of M with $|\lambda| = \rho(M)$, and let $\overline{x} \in \mathbb{C}^n \setminus \{0\}$ be the corresponding eigenvector. As we aim to show here that (9.11) is a necessary condition, we may specify b as $= 0 \in \mathbb{C}^n$. Then by $x_0 = \overline{x}$ we get

$$x_m \quad = \quad \phi(x_{m-1}, b) = Mx_{m-1} = M(Mx_{m-2}) = \ldots. \tag{9.12}$$

Finally, we obtain

$$x_m \quad = \quad M^m x_0 = \lambda^m x_0. \tag{9.13}$$

For $|\lambda| > 1$ we should obtain by $\|x_m\| = |\lambda|^m \cdot \|x_0\|$ the divergence of the sequence $\{x_m\}_{m\in\mathbb{N}}$.
For $|\lambda| = 1$, M is nothing else but a rotation of the eigenvector. Convergence of $\{x_m\}_{m\in\mathbb{N}}$ could be achieved for the trivial rotation, i.e., $\lambda = 1$. In the latter case $x_m = x_0$ holds for all initial vectors x_0 for all $m \in \mathbb{N}$, so that

$$\hat{x} \quad = \quad \lim_{m\to\infty} x_m = x_0$$

is a limit vector depending on x_0. Thus, the iterative scheme is not convergent because of this dependence.

In summary, $|\lambda| = \rho(M) < 1$ is a necessary condition.

"$\Leftarrow$" Assumption: $\rho(M) < 1$ (sufficient condition). By Definition and Corollary **??**, all norms are equivalent on $\mathbb{C}^n$. Thus, the convergence of ϕ can be proven in any norm. Let

$$\varepsilon \;:=\; \frac{1}{2}(1-\rho(M)) > 0, \tag{9.14}$$

then there exists by Corollary 8.1.2 a norm on $\mathbb{C}^{n\times n}$ so, that

$$q \;:=\; \|M\| \leq \rho(M) + \varepsilon. \tag{9.15}$$

For any given $b \in \mathbb{C}^n$ we define

$$\begin{aligned} F : \mathbb{C}^n &\;\to\; \mathbb{C}^n \\ x &\;\overset{F}{\mapsto}\; F(x) = Mx + Nb \end{aligned} \tag{9.16}$$

Herewith we obtain

$$\|F(x) - F(y)\| \;=\; \|Mx - My\| \leq \|M\| \cdot \|x-y\| = q\|x-y\|. \tag{9.17}$$

Let us briefly comment on the step

$$\|Mx - My\| \;\leq\; \|M\| \cdot \|x - y\|. \tag{9.18}$$

We compute in detail

$$\begin{aligned} \|Mx - My\| &= \|M(x-y)\| \\ &= \underbrace{\frac{1}{\|x-y\|}}_{\in\mathbb{R}^+} \|M(x-y)\| \cdot \|x-y\| \\ &= \|M \underbrace{\left(\frac{x-y}{\|x-y\|}\right)}_{=:z\in\mathbb{C}^n,\ \|z\|=1} \| \cdot \|x-y\| \\ &\leq \sup_{z\in\mathbb{C}^n,\ \|z\|=1} \|Mz\| \cdot \|x-y\| \\ &= \|M\| \cdot \|x-y\|. \end{aligned}$$

Let us stress, that we obtain from (9.15) and (9.17)

$$\|F(x) - F(y)\| \leq q\|x-y\|, \qquad 0 < q < 1. \tag{9.19}$$

Applying the fixed-point-theorem of Banach, see Theorem 8.2.1, we get the convergence of the sequence $x_{m+1} = F(x_m)$ for any $x_0 \in \mathbb{C}^n$, and for the unique limit element $\hat{x}$ holds:

$$\hat{x} \;=\; \lim_{m\to\infty} x_{m+1} = \lim_{m\to\infty} F(x_m) \overset{(9.16)}{=} \lim_{m\to\infty} \phi(x_m, b). \tag{9.20}$$

■

The developments combine to

Theorem 9.0.4 *Let ϕ be convergent iterative scheme also consistent to A. Then, the limit element x^* of the sequence*

$$x_m = \phi(x_{m-1}, b), \qquad m = 1, 2, \ldots \tag{9.21}$$

solves the system $Ax = b$ for any $x_0 \in \mathbb{C}^n$.

Chapter 10

Iterative Solvers for Linear Systems: Splitting-methods

The basis of splitting-methods for solving $Ax = b$ is the splitting

$$A \quad = \quad B + (A - B), \tag{10.1}$$

so that $Ax = b$ is equivalent to

$$\begin{aligned} & [B + (A - B)]x = b \\ \Leftrightarrow \quad & Bx = (B - A)x + b \\ \Leftrightarrow \quad & x = B^{-1}(B - A)x + B^{-1}b \end{aligned} \tag{10.2}$$

The idea is to choose $B \in \mathbb{C}^{n \times n}$

- as close as possible to A, but
- easily invertible.

By (10.2), it is easy to formulate splitting-methods in terms of a linear iterative scheme,

$$x_{m+1} \quad = \quad \phi(x_m, b) = \underbrace{B^{-1}(B - A)}_{=:M} x_m + \underbrace{B^{-1}}_{=:N} b, \tag{10.3}$$

where ϕ is automatically consistent to A. In case A is positive definite and symmetric (PDS), symmetric splitting-methods preserve and make use of this structure by using a B that is also PDS.
To assess the efficiency of splitting-methods, we will make use of the following (general!) assertion for the iteration matrix M.

Theorem 10.0.5 *Let $\rho(M) < 1$. Then there exists a norm so, that $q := \|M\| < 1$ holds. For any given $\varepsilon > 0$, it follows that*

$$\|x_m - A^{-1}b\| \;\leq\; \varepsilon \tag{10.4}$$

for all $m \in \mathbb{N}$ with

$$m \;\geq\; \frac{\ln\left(\dfrac{\varepsilon(1-q)}{\|x_1 - x_0\|}\right)}{\ln q} \tag{10.5}$$

Proof:
By $\rho(M) < 1$ and Corollary 8.1.2, there exists for all $\varepsilon > 0$ with $0 < \varepsilon < 1 - \rho(M)$ a norm yielding

$$\rho(M) \;\leq\; \underbrace{\|M\|}_{:=q} \leq \rho(M) + \varepsilon < 1 \tag{10.6}$$

Using the a-priori-estimate of the fixed-point-theorem of Banach, see Theorem 8.2.1, we obtain directly

$$\|x_m - A^{-1}b\| \;\leq\; \frac{q^m}{1-q}\|x_1 - x_0\|. \tag{10.7}$$

Setting concretely

$$\varepsilon \;:=\; \frac{q^m}{1-q}\|x_1 - x_0\|, \tag{10.8}$$

we obtain (10.4) via

$$\varepsilon\frac{1-q}{\|x_1 - x_0\|} \;=\; \frac{q^m}{1-q}\|x_1 - x_0\| \cdot \frac{1-q}{\|x_1 - x_0\|} = q^m \tag{10.9}$$

and using the rule $\ln q^m = m \cdot \ln q$, we get (10.5)

$$\frac{\ln\left(\dfrac{\varepsilon(1-q)}{\|x_1 - x_0\|}\right)}{\ln q} \;=\; m \qquad (\leq m+1 \leq m+2 \leq \ldots). \tag{10.10}$$

■

Remark:
By Theorem 10.0.5, it becomes clear that the aim is to choose B in a way that $\rho(M) \ll 1$.

10.1 The method of Jacobi

The simplest choice for an easily invertible approximation of A is to take just its diagonal:

$$D \;:=\; B := \operatorname{diag}(a_{11}, a_{22}, \dots, a_{nn}). \tag{10.11}$$

In this case,

$$D^{-1} \;=\; B^{-1} = \operatorname{diag}\left(\frac{1}{a_{11}}, \frac{1}{a_{22}}, \dots, \frac{1}{a_{nn}}\right). \tag{10.12}$$

Obviously, we need non-zero diagonal entries. The resulting Jacobi-scheme is

$$x_{m+1} \;=\; D^{-1}(D-A)x_m + D^{-1}b, \qquad m = 0, 1, 2, \dots, \tag{10.13}$$

reading componentwise for $x_{m+1} = (x_{m+1,1}, x_{m+1,2}, \dots, x_{m+1,n})^\top$ as:

$$x_{m+1,i} \;=\; \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_{m,j}\right) \tag{10.14}$$

for $i = 1, \dots, n$. As observable by (10.14), for the computation of x_{m+1} we only employ x_m. Consequently, the Jacobi-scheme features

(i) independency of the numbering of vector entries $x_{m+1,i}$, and

(ii) a high potential for parallel implementation.

Remark: One can prove convergence to $A^{-1}b$ of the iterates x_{m+1} of the Jacobi method under the uniqueness assumptions on A elaborated on in §6.

10.2 The Gauß-Seidel-method

We now split A as

$$A = L + D + U, \tag{10.15}$$

where D is as in 10.11, and

$$L = (l_{ij}) \quad \text{with} \quad l_{ij} = \begin{cases} a_{ij}, & i > j, \\ 0, & \text{else,} \end{cases} \tag{10.16}$$

$$U = (u_{ij}) \quad \text{with} \quad u_{ij} \begin{cases} a_{ij}, & i < j, \\ 0, & \text{else,} \end{cases} \tag{10.17}$$

i.e., L and U are the lower and upper triangular components of A, respectively. Then

$$Ax = b \;\Leftrightarrow\; (D+L)x = -Ux + b \tag{10.18}$$

and

$$x = \underbrace{-(D+L)^{-1}U}_{=:M_{\mathrm{GS}}}\, x + \underbrace{(D+L)^{-1}}_{=:N_{\mathrm{GS}}}\, b \tag{10.19}$$

With these obvious definitions for ϕ, we now aim for a similar formula as (10.14). To this end, we look at the i-th row of the system (10.18):

$$\sum_{j=1}^{i} a_{ij} x_{m+1,j} = -\sum_{j=i+1}^{n} a_{ij} x_{m,j} + b_i. \tag{10.20}$$

Computing the entries of x_{m+1} in the order $1, 2, 3, \ldots,$ up to n we assume that at the index i the values

$$x_{m+1,j} \qquad \text{for } j = 1, \ldots, i-1, \tag{10.21}$$

are already known. Plugging these into the left hand side of (10.20) gives

$$x_{m+1,i} \;=\; \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{m+1,j} - \sum_{j=i+1}^{n} a_{ij} x_{m,j}\right) \tag{10.22}$$

for $i = 1, \ldots, n$.

Remarks:

- The method relies on the ordering.
- Also here, we need $a_{ii} \neq 0$.
- One can prove convergence of $\{x_n\}$ to $A^{-1}b$ under moderate assumptions.

10.3 Relaxation methods

We now rewrite the general formula

$$x_{m+1} \;=\; B^{-1}(B-A)x_m + B^{-1}b \tag{10.23}$$

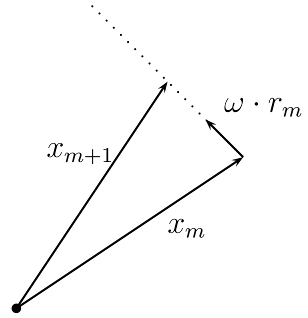
as

$$x_{m+1} = x_m + \underbrace{B^{-1}(b - Ax_m)}_{=:r_m}. \qquad (10.24)$$

Thus, we may interprete x_{m+1} as a correction of x_m making use of the vector u_m. The idea is to get a better performance by weighting the vector r_m, stretching it by some factor $\omega < 1$ ("underrelaxation") or $\omega > 1$ ("overrelaxation"), yielding:

$$x_{m+1} = x_m + \omega B^{-1}(b - Ax_m), \qquad \omega \in \mathbb{R}^+. \qquad (10.25)$$

Sketch:



Rewriting again (10.25) as

$$x_{m+1} = \underbrace{(I - \omega B^{-1}A)}_{=:M(\omega)} x_m + \underbrace{\omega B^{-1}}_{N(\omega)} b, \qquad (10.26)$$

an optimal choice for ω is obviously

$$\omega = \min_{\alpha \in \mathbb{R}^+} \rho(M(\alpha)). \qquad (10.27)$$

10.4 The SOR-scheme

We now consider the overrelaxation of the Gauß-Seidel-method ("successive-overrelaxation scheme"). Using

$$\frac{1}{a_{ii}} \left(- \sum_{j=i+1}^{n} a_{ij} x_{m,j} \right) = x_{m,i} + \frac{1}{a_{ii}} \left(- \sum_{j=i}^{n} a_{ij} x_{m,j} \right)$$

in (10.22), we rewrite the latter formula in the style of (10.25):

$$x_{m+1,i} = x_{m,i} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{m+1,j} - \sum_{j=i}^{n} a_{ij} x_{m,j} \right), \qquad (10.28)$$

obtaining

$$x_{m+1,i} \quad = \quad (1-\omega)x_{m,i} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_{m+1,j} - \sum_{j=i+1}^{n} a_{ij}x_{m,j}\right). \qquad (10.29)$$

In terms of matrices, (10.29) reads as

$$(I+\omega D^{-1}L)x_{m+1} = [(1-\omega)I - \omega D^{-1}U]x_m + \omega D^{-1}b, \qquad (10.30)$$

which is identical to

$$D^{-1}(D+\omega L)x_{m+1} = D^{-1}[(1-\omega)D - \omega U]x_m + \omega D^{-1}b. \qquad (10.31)$$

We obtain by (10.31) the SOR-scheme as

$$x_{m+1} \quad = \quad \underbrace{(D+\omega L)^{-1}[(1-\omega)D-\omega U]}_{=:M_{\mathrm{GS}}(\omega)} x_m + \underbrace{\omega(D+\omega L)^{-1}}_{=:N_{\mathrm{GS}}(\omega)} b. \qquad (10.32)$$

We consider two key assertions

Theorem 10.4.1 *For $A \in \mathbb{C}^{n\times n}$ with $a_{ii} \neq 0$, it holds:*

$$\rho(M_{GS}(\omega)) \quad \geq \quad |\omega - 1| \qquad (10.33)$$

Proof:
Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $M_{\mathrm{GS}}(\omega)$, then

$$\begin{aligned}
\prod_{i=1}^{n}\lambda_i \quad &= \quad \det M_{\mathrm{GS}}(\omega) \\
&\overset{(10.32)}{=} \quad \det((\underbrace{D+\omega L}_{\text{lower triangular}})^{-1}) \cdot \det(\underbrace{(1-\omega)D-\omega U}_{\text{upper triangular}}) \\
&\qquad \text{using } \det(AB) = \det A \cdot \det B \text{ for any } A, B \\
&= \quad \det(D^{-1}) \cdot \det((1-\omega)D) \qquad \text{as } L, U \text{ do not contribute} \\
&= \quad (\det D)^{-1} \cdot (1-\omega)^n \cdot \det D \qquad \text{by rules for determinants} \\
&= \quad (1-\omega)^n.
\end{aligned}$$

Thus,

$$\rho(M_{\mathrm{GS}(\omega)}) \quad = \quad \max_i |\lambda_i| \geq |1-\omega|, \qquad (10.34)$$

as otherwise

$$\prod_{i=1}^{n}\lambda_i \quad < \quad (1-\omega)^n$$

must hold. ■

The most important consequence of Theorem 10.4.1 is

Corollary 10.4.1 *A necessary condition for the convergence of the SOR method is $\omega \in (0,2)$.*

In order to assess the convergence of SOR, one needs to consider the Jacobi-method with

$$M_{\mathrm{J}} \;:=\; D^{-1}(D-A) \tag{10.35}$$

Let $\rho := \rho(M_{\mathrm{J}}) < 1$. Then, under some mild assumptions, one can show that $\rho(M_{\mathrm{GS}})$ is minimal for

$$\omega_{\mathrm{optimal}} \;=\; \frac{2}{1+\sqrt{1-p^2}}, \tag{10.36}$$

and then

$$\rho(M_{\mathrm{GS}}(\omega_{\mathrm{optimal}})) \;=\; \omega_{\mathrm{optimal}} - 1 = \frac{1-\sqrt{1-p^2}}{1+\sqrt{1-p^2}} \tag{10.37}$$

follows. Let us briefly consider the main step of the proof to give a flavour.
One can relate the eigenvalues $\mu \in \mathbb{C}\setminus\{0\}$ of M_{GS} and $\lambda \in \mathbb{R}$ of M_{J} by the formula

$$\lambda \;=\; \frac{\mu+\omega-1}{\omega\mu^{\frac{1}{2}}} \tag{10.38}$$

(there exists a formula for $\sqrt{\mu}$ for μ complex). From (10.38) we derive the condition

$$(\mu+\omega-1)^2 - \omega^2\mu\lambda^2 \;=\; 0 \tag{10.39}$$

By the quadratic equation (10.39) we get, e.g. by the p-q-formula two eigenvalues $\mu^{\pm}$ as

$$\mu^{\pm} \;=\; \mu^{\pm}(\omega,\lambda) = \frac{1}{2}\lambda^2\omega^2 - (\omega-1) \pm \underbrace{\lambda\omega\sqrt{\frac{1}{4}\lambda^2\omega^2-(\omega-1)}}_{=:\hat{g}}. \tag{10.40}$$

Obviously, $\mu^{\pm}$ gets minimal in the general case if there is no contribution by $\hat{g}$. Thus, we look for its zeroes:

$$g(\omega,\lambda) \;:=\; \frac{1}{4}\lambda^2\omega^2 - (\omega-1) \overset{!}{=} 0. \tag{10.41}$$

These are given by

$$\omega^{\pm} \;=\; w^{\pm}(\lambda) = \frac{2}{1\pm\sqrt{1-\lambda^2}}, \tag{10.42}$$

and by $\omega \in (0,2)$ we can neglect the "−"-case, yielding $\omega_{\mathrm{optimal}}$ from (10.36).

Chapter 11

Iterative Solvers for Linear Systems: Krylov-Subspace-Methods

We consider again

$$Ax \;=\; b, \qquad A \in \mathbb{R}^{n\times n}, \qquad b \in \mathbb{R}^n. \tag{11.1}$$

The idea we will follow is to search for an approximate solution of (11.1) in a suitable low-dimensional subspace of $\mathbb{R}^n$. To this end, let K_m and L_m be m-dimensional subspaces of $\mathbb{R}^n$.

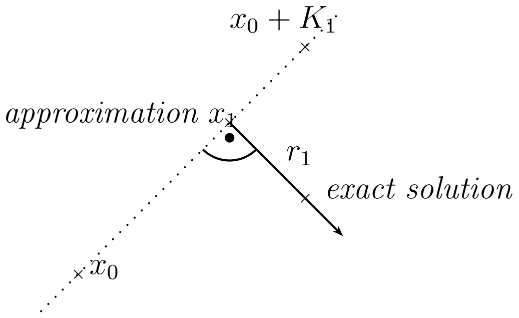
Definition 11.0.1 *A projection method computes approximate solutions $x_m \in x_0 + K_m$ obeying and making the use of orthogonality condition*

$$(b - Ax_m) \perp L_m, \tag{11.2}$$

where $x_0 \in \mathbb{R}^n$. Orthogonality is defined via the Euclidean scalar product

$$a \perp b \;:=\; (a,b)_2 = 0 \;\Leftrightarrow\; a^\top b = 0 \;\Leftrightarrow\; a_1 b_1 + \ldots + a_n b_n = 0. \tag{11.3}$$

In case of $K_m = L_m$, (11.2) says that the residual vector $r_m = b - Ax_m$ is orthogonal to K_m:
Sketch:



In this case, we have an orthogonal projection method. For $K_m \neq L_m$, the projection method is called skewed.

Let us emphasize differences between splitting schemes and projection methods.

	Splitting methods	Projection methods
Computed iterates are	$x_m \in \mathbb{R}^n$	$x_m \in x_0 + K_m$, $K_m \subset \mathbb{R}^n, \dim K_m = m \leq n$
Scheme defined by	$x_m = Mx_{m-1} + Nb$	$b - Ax_m \perp L_m$, $L_m \subset \mathbb{R}^n, \dim L_m = m \leq n$

We now get more specific.

Definition 11.0.2 *A Krylov-Subspace-Method is a projection method where K_m is chosen as*

$$K_m \;=\; K_m(A, r_0) = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \tag{11.4}$$

with $r_0 = Ax_0 - b$.

The meaning of the formula (11.4) is nothing else but an algorithm to increase successively the dimension of K_m.
Krylov-Subspace-Methods are often derived by reformulating (11.1) as a minimisation task. The two most popular schemes in this class are the CG-scheme for symmetric positive definite (SPD) matrices of Hestenes and Stiefel (1952) and the GMRES-scheme for general regular matrices of Saad and Schultz (1986). We will go for CG.

11.1 The gradient descent scheme

At the basis of all derivations is the function

$$\begin{aligned} F: \quad \mathbb{R}^n &\rightarrow \mathbb{R} \\ x &\mapsto \frac{1}{2}(Ax, x)_2 - (b, x)_2 \end{aligned} \tag{11.5}$$

together with the following observation:

Corollary 11.1.1 *If A is an SPD matrix, then the minimum x^* of F from* (11.5) *is attained if and only if $Ax^* = b$.*

We now aim to minimise F successively, starting from some point $x \in \mathbb{R}^n$ and minimising along directions $p \in \mathbb{R}^n$. For these $x, p \in \mathbb{R}^n$ we define the function

$$\begin{aligned} f_{x,p} : \qquad \mathbb{R} &\rightarrow \mathbb{R} \\ \lambda &\mapsto f_{x,p}(\lambda) := F(x + \lambda p). \end{aligned} \tag{11.6}$$

Corollary 11.1.2 *The global minimum of $f_{x,p}$, i.e. the minimum of F, starting in x and searching along $x + \lambda p$, is given by*

$$\lambda_{opt} = \frac{(r,p)_2}{(Ap,p)_2}, \tag{11.7}$$

where $r := b - Ax$.

Remark: The proof relies on elementary 1-D calculus; $f'_{x,p}(\lambda_{\text{opt}}) = 0$ and $f''_{x,p}(\lambda) > 0$ for all λ. Given thus a sequence of search directions $\{p_m\}$, we can formulate a basic algorithm.

Algorithm 10-1:

```
(1)  Choose $x_0 \in \mathbb{R}^n$.
(2)  For $m = 0, 1, \ldots$, do
       • $r_m = b - Ax_m$
       • $\lambda_m = \frac{(r_m, p_m)_2}{(Ap_m, p_m)_2}$
       • $x_{m+1} = x_m + \lambda_m p_m$
```

We now need an algorithm determining $\{p_m\}$, demanding $\|p_m\|_2 = 1$.

The gradient descent method is defined via the "downhill-choice": By

$$\nabla F(x) = \frac{1}{2}(A + A^\top)x - b \overset{A \text{ symmetric}}{=} Ax - b = -r, \tag{11.8}$$

the direction of steepest descent $-\nabla F(x)$ at x is

$$p := \begin{cases} \dfrac{r}{\|r\|_2} & \text{for } r \neq 0, \\ 0 & \text{for } r = 0, \end{cases} \tag{11.9}$$

defining $\{p_m\}$ in an obvious way. For later use, let us stress that the residual vectors define the search directions.

11.2 The method of conjugate directions

One can generalise the procedure by taking into account several search directions within the construction of the algorithm.

Definition 11.2.1 *For F as in* (11.5)*, $x \in \mathbb{R}^n$ is*

(a) optimal w.r.t. the direction $p \in \mathbb{R}^n$, if

$$F(x) \;\leq\; F(x + \lambda p) \qquad \forall \lambda \in \mathbb{R}, \tag{11.10}$$

(b) optimal w.r.t. the subspace $U \subset \mathbb{R}^n$, if

$$F(x) \;\leq\; F(x + \xi) \qquad \forall \xi \in U, \tag{11.11}$$

As in Corollary 11.1.2, one can prove

Corollary 11.2.1 *For F as in* (11.5)*, $x \in \mathbb{R}^n$ is optimal w.r.t. $U \subset \mathbb{R}^n$, if*

$$r \;=\; b - Ax \perp U \tag{11.12}$$

holds.

For defining a constructive procedure, it is important to be sure about the dimension of U. We now explore a means to ensure that a collection of search directions $p_0, \ldots, p_{m-1}$ spans a m-dimensional subspace of $\mathbb{R}^n$, so that

$$\text{dim } U_m = m \qquad \text{for } U_m = \text{span}\{p_0, \ldots, p_{m-1}\}. \tag{11.13}$$

Definition 11.2.2 *The vectors $p_0, \ldots, p_{m-1} \in \mathbb{R}^n$ are pairwise conjugate, or A-orthogonal, if*

$$(p_i, p_j)_A \;:=\; (Ap_i, p_j)_2 = 0 \qquad \forall i, j \in \{0, \ldots, m-1\} \textit{ and } i \neq j. \tag{11.14}$$

The key to success is:

Corollary 11.2.2 *For A a SPD matrix and $p_0, \ldots, p_{m-1} \in \mathbb{R}^n \setminus \{0\}$ pairwise A-orthogonal, then*

$$dim(span\{p_0, \ldots, p_{m-1}\}) \;=\; m. \tag{11.15}$$

Proof:

Let $\sum_{j=0}^{m-1} \alpha_j p_j = \vec{0}$ with the coefficients $\alpha_j \in \mathbb{R}$. If the p_j's are linearly independent, $\alpha_j = 0$ must follow, which we want to show. We compute for any p_i:

$$\begin{aligned} 0 &= (\vec{0}, Ap_i)_2 = \left(\sum_{j=0}^{m-1} \alpha_j p_j, Ap_i \right)_2 \\ &= \sum_{j=0}^{m-1} \alpha_j (p_j, Ap_i)_2 \\ &\overset{A\text{-Orthogonality}}{=} \alpha_i \underbrace{(p_i, Ap_i)_2}_{>0 \text{ as } A \text{ is SPD}} . \end{aligned}$$

From $0 = \alpha_i (p_i, Ap_i)_2$, $\alpha_i = 0$ must follow. ■

Thus, let

- $p_0, \ldots, p_{m-1} \in \mathbb{R}^n \setminus \{0\}$ be pairwise conjugate search directions,
- x_m be optimal w.r.t. $U_m = \text{span}\{p_0, \ldots, p_{m-1}\}$. Then we obtain the optimality of

$$x_{m+1} = x_m + \lambda p_m \tag{11.16}$$

w.r.t. U_{m+1}, if for $j = 0, \ldots, m$, we have:

$$0 = (b - Ax_{m+1}, p_j)_2 = \underbrace{(b - Ax_m, p_j)_2}_{=0 \text{ for } j \neq m \text{ by (11.12)}} + \underbrace{\lambda (Ap_m, p_j)_2}_{=0 \text{ for } j \neq m \text{ by (11.14)}} . \tag{11.17}$$

For the only interesting choice $j = m$ we obtain

$$\lambda = \frac{(r_m, p_m)_2}{(Ap_m, p_m)_2}. \tag{11.18}$$

This yields the following algorithm, where $\{p_m\}$ still needs to be determined.

Algorithm 10-2

(a) Choose $x_0 \in \mathbb{R}^n$

(b) $r_0 := b - Ax_0$

(c) For $m = 0, \ldots, n - 1$

- $\lambda_m = \dfrac{(r_m, p_m)_2}{(Ap_m, p_m)_2}$
- $x_{m+1} = x_m + \lambda p_m$
- $r_{m+1} = r_m - \lambda_m A p_m$.

11.3 The CG-Method

The CG-Method of Hestenes and Stiefel combines the gradient descent method with the method of conjugate directions. To this end we use as in (11.9) the residual vectors as search directions:

$$p_0 = r_0, \qquad p_m = r_m + \sum_{j=0}^{m-1} \alpha_j p_j. \tag{11.19}$$

For $\alpha_j = 0$ we retrieve the gradient descent scheme. For more general α_j, the search directions $p_0, \ldots, p_{m-1}$ are taken into account now, leaving by the α_j's m degrees of freedom for ensuring A-orthogonality. The latter condition implies

$$0 = (Ap_m, p_i)_2 \overset{(11.19)}{=} (Ar_m, p_i)_2 + \sum_{j=0}^{m-1} \alpha_j (Ap_j, p_i)_2 \tag{11.20}$$

for $i = 0, \ldots, m-1$. By $(Ap_j, p_i)_2 = 0$ for $i, j \in \{0, \ldots, m-1\}$ and $i \neq j$ follows

$$\alpha_i \quad = \quad \frac{(Ar_m, p_i)_2}{(Ap_i, p_i)_2}. \tag{11.21}$$

In principle we are done, however, in practice the scheme derived up to now is inefficient as for computing p_m via (11.19) the storage of all the search directions $p_0, \ldots, p_{m-1}$ is required. This can be circumvented by using the simplified formula

$$p_m \quad = \quad r_m + \frac{(r_m, r_m)_2}{(r_{m-1}, r_{m-1})_2} p_{m-1}, \tag{11.22}$$

which can be derived from (11.19)-(11.21). The final result is

Algorithm 10-3

```
(a)  Choose $x_0 \in \mathbb{R}^n$.
(b)  $p_0 := r_0 := b - Ax_0, \quad \alpha_0 := \|r_0\|_2^2 = (r_0, r_0)_2$
(c)  For $m = 0, \ldots, n-1$ do
```

- If $\alpha_m \neq 0$ proceed, else STOP
- $v_m := Ap_m$, $\lambda_m := \dfrac{\alpha_m}{(v_m, p_m)_2}$
- $x_{m+1} := x_m + \lambda_m p_m$
- $r_{m+1} := r_m - \lambda_m v_m$
- $\alpha_{m+1} := \|r_{m+1}\|_2^2$
- $p_{m+1} := r_{m+1} + \dfrac{\alpha_{m+1}}{\alpha_m} p_m$

Remarks:

- One can prove, that if the stopping criterion is fulfilled, the solution of (11.1) is found.
- The CG-scheme is an orthogonal Krylov-Subspace-Method.

Chapter 12

Preconditioning Linear Systems

Motivation:

- Preconditioning makes iterative solvers for linear systems $Ax = b$ efficient. This can make a difference towards real-world applications in image processing.
- Before looking for algorithms, we discuss a number indicating the efficiency of iterative solvers, the so-called condition number.

Definition 12.0.1 *If $A \in \mathbb{C}^{n\times n}$ is regular, then*

$$cond_a(A) \;\; := \;\; \|A\|_a \cdot \|A^{-1}\|_a \tag{12.1}$$

where

$$\|A\|_a \;\; := \;\; \sup_{\|x\|_a=1} \|Ax\|_a \tag{12.2}$$

is the matrix norm induced by the vector norm $\|.\|_a$.

One can show

Lemma 12.0.1 *For A regular it holds*

$$cond(A) \;\; \geq \;\; cond(I) = 1. \tag{12.3}$$

In a practical setting, the norm of the residual vector $r_m = b - Ax_m$ usually serves as a measure for the quality of x_m, since $e_m = A^{-1}b - x_m$ is not available. One reason for this is, that $r_m = 0$ means that $x_m = x$, where $Ax = b$. However, only for a small condition number cond(A) the norm or r_m is really meaningful. This is shown by

Theorem 12.0.1 *Given is an iterative scheme for solving $Ax = b$. Consider the exact error vector $e_K = A^{-1}b - x_k$ and the residual vector $r_k = b - Ax_k$ of the k-th iteration step, then*

$$\frac{1}{cond(A)} \cdot \frac{\|r_k\|}{\|r_0\|} \leq \frac{\|e_k\|}{\|e_0\|} \leq cond(A)\frac{\|r_k\|}{\|r_0\|} \leq cond(A)^2\frac{\|e_k\|}{\|e_0\|} \tag{12.4}$$

We especially see by (12.4) that the rate of decrease in the true error, $\frac{\|e_k\|}{\|e_0\|}$, is bounded by $\text{cond}(A)\frac{\|r_k\|}{\|r_0\|}$. Thus, for a large number $\text{cond}(A)$, even a small number $\|r_k\|$ does not show that x_k is a good approximate of x since $\text{cond}(A) \cdot \frac{\|r_k\|}{\|r_0\|}$ can be considerably large.

Let us also consider the influence of errors in given data that are usually collected via b.

Theorem 12.0.2 *Let x solve $Ax = b$, and let $x + \delta x$ solve $A(x + \delta x) = b + \delta b$, then*

$$\frac{\|\delta x\|}{\|x\|} \leq cond(A)\frac{\|\delta b\|}{\|b\|}. \tag{12.5}$$

Similarly as by Theorem 12.0.1, we observe by (12.5) that a small change δb may infer a large variation δx if $\text{cond}(A) \gg 1$. Summarising Theorem 12.0.1 and Theorem 12.0.2, it makes sense to transform $Ax = b$ into an equivalent system $\tilde{A}x = \tilde{b}$ with $\text{cond}(\tilde{A}) \ll \text{cond}(A)$, and to solve $\tilde{A}x = \tilde{b}$ iteratively instead of $Ax = b$.

12.1 Construction principle

We begin with

Definition 12.1.1 *Let P_L and P_R be regular. Then*

$$\begin{aligned} P_L A P_R x^P &= P_L b && (12.6a) \\ x &= P_R x^P && (12.6b) \end{aligned}$$

is the preconditioned system corresponding to $Ax = b$. If $P_L \neq I$, then P_L is called left preconditioner, and the system is preconditioned from the left. Analogously, the notion of a right preconditioning is coined. If both $P_L, P_R \neq I$, then we have a twosided preconditioning.
The conflicting goals in defining P_L, P_R are:

(a) P_LAP_R *shall approximate* I *as good as possible,* $cond(P_LAP_R) \ll cond(A)$*.*

(b) P_L, P_R *shall be easy to compute.*

(c) P_L, P_R *shall use a small amount of disk space.*

Remark: The twosided preconditioning with $P_L^\top = P_R$ is useful in the context of SPD matrices, as $P_LAP_L^\top$ is again SPD.

12.2 Construction of Splitting-based preconditioners

Recalling the splitting methods from §10, they were based on rewriting A as

$$A = B + (A - B) \tag{12.7}$$

The idea in that was that B shall be an easy to invert matrix close to A, so that the iteration matrix

$$M = B^{-1}(B - A) \tag{12.8}$$

has a small spectral radius. This idea is close to the concept of preconditioners. Thus, it seems a good idea to use $N = B^{-1}$ as P_L. We conclude these thoughts via

Definition 12.2.1 *Let* $x_{j+1} = Mx_j + Nb$ *a splitting method for solving* $Ax = b$ *with regular* N*. Then* $P := N$ *is the preconditioner associated with the splitting method.*

Knowing some matrices N from before, we summarise some preconditioners.

Splitting method	Associated preconditioner P
Jacobi	D^{-1}
Gauß-Seidel (GS)	$(D+L)^{-1}$
SOR	$\omega(D+\omega L)^{-1}$

12.3 Incomplete approaches

Another approach we just briefly sketch here is to rely on factorisations such as $A = LU$, $A = LL^\top$, or $A = QR$. In the context of sparse systems, only the structure of entries in A is used, so that the factorisations will be incomplete, e.g. $A = \hat{L}\hat{U} + F$. However, using e.g. $P = (\hat{L}\hat{U})^{-1}$ is an efficient preconditioner.
There are many other choices, reflecting a variety of approaches.

12.4 PCG - Preconditioned Conjugate Gradients

As we have in image processing a number of tasks where SPD matrices arise, e.g. Deconvolution, we discuss the preconditioning of CG in some detail.

Starting with $Ax = b$, we use a regular matrix P_L yielding

$$A^P x^P \;=\; b^P \tag{12.9}$$

with

$$A^P = P_L A P_L^\top, \quad x^P = P_L^{-\top} x, \quad b^P = P_L b, \tag{12.10}$$

where $P_L^{-\top} := (P_L^{-1})^\top$. The matrix P_L is for a SPD matrix A often given by a symmetric ansatz, e.g. a symmetric splitting method, or an incomplete Cholesky factorisation, compare Table **??**.
After a few simple manipulations of the straightforward modification of CG, the result is, for $P := P_L^\top P_L$ and a prescribed $\varepsilon \ll 1$:

Algorithm 17-1 (PCG)

```
1) Choose $x_0 \in \mathbb{R}^n$
2) $r_0 := b - Ax_0$, $\hat{p}_0 := Pr_0$, $\alpha_0^P := (r_0, \hat{p}_0)_2$
3) For $m = 0, \ldots, n-1$
    • If $\|r_m\|_2 \leq \varepsilon$ then STOP
    • ELSE:
        (i) $\hat{v}_m := A\hat{p}_m$, $\lambda_m^P := \dfrac{\alpha_m^P}{(\hat{v}_m, \hat{p}_m)_2}$
        (ii) $x_{m+1} := x_m + \lambda_m^P \hat{p}_m$
        (iii) $r_{m+1} := r_m - \lambda_m^P \hat{v}_m$
        (iv) $z_{m+1} := Pr_{m+1}$, $\alpha_{m+1}^P := (r_{m+1}, z_{m+1})_2$
        (v) $\hat{p}_{m+1} := z_{m+1} + \dfrac{\alpha_{m+1}^P}{\alpha_m^P} \hat{p}_m$
```

Remark: For the implementation one heavily relies on the fact that matrices like D, $D + R$, $D + L$, are diagonal or triangular, so that they can easily be inverted by forward or backward elimination, respectively.

Chapter 13

Diffusion Problems: Numerical Stability and FED

We consider as a model problem the discretisation

$$U_j^{n+1} \;=\; U_j^n + \frac{D\Delta t}{\Delta x^2}\left[U_{j+1}^n - 2U_j^n + U_{j-1}^n\right] \tag{13.1}$$

of the linear diffusion PDE

$$u_t \;=\; D \cdot u_{xx}\;, \quad D > 0 \tag{13.2}$$

Since linear diffusion as by (13.2) models a smoothing process equivalent to convolution with a Gaussian, we can state that by (13.1)

- no oscillations shall be induced as these are contradictive to smoothing
- there should be a discrete minimum-maximum principle

Let us investigate these points, starting by rearranging terms of (13.1):

$$U_j^{n+1} \;=\; \frac{D\Delta t}{\Delta x^2}U_{j+1}^n + \left[1 - 2\frac{D\Delta t}{\Delta x^2}\right]U_j^n + \frac{D\Delta t}{\Delta x^2}U_{j-1}^n \tag{13.3}$$

A necessary and sufficient condition for realising a discrete minimum-maximum principle is that it holds locally in all points. The corresponding property is that U_{j+1}^{n+1} shall be in the <u>convex hull</u> of U_{j+1}^n, U_j^n and U_{j-1}^n:

$$U_j^{n+1} \;:=\; \sum_{k=-1}^{1} \alpha_k U_{j+k}^n\;, \quad \alpha_k \geq 0\;, \quad \sum_{k=-1}^{1} \alpha_k = 1 \tag{13.4}$$

While the α_k as by (13.3) sum up to 1, the condition $\alpha_k \geq 0$ needs to be fulfilled especially for α_0:

$$1 - 2\frac{D\Delta t}{\Delta x^2} \overset{!}{\geq} 0 \tag{13.5}$$

As this invokes the condition

$$\Delta t \leq \frac{\Delta x^2}{2D} \tag{13.6}$$

one speaks of conditional stability.

A second approach is to understand the method (13.1) as a linear filter, and to investigate the amplification of input signals. To this end, we look at the corresponding transfer function

$$H(\omega) = 1 + \frac{D\Delta t}{\Delta x^2}\left[e^{i\omega\Delta x} - 2 + e^{-i\omega\Delta x}\right] \tag{13.7}$$

where we introduced waves in accordance to $U^n_{j+k} \rightarrow e^{i\omega(k\Delta x)}$. Rearranging terms using Euler's identity $e^{i\varphi} = \cos\varphi + i\sin\varphi$ gives

$$H(\omega) = 1 + \frac{D\Delta t}{\Delta x^2}[2\cos\omega\Delta x - 2] = 1 - \underbrace{2\frac{D\Delta t}{\Delta x^2}[1 - \cos\omega\Delta x]}_{=:A} \tag{13.8}$$

Any input signal will not be amplified if $H(\omega) \leq 1$. Since $\cos\omega\Delta x \in [-1, 1]$ for any ω, it holds $1 - \cos\omega\Delta x \geq 0$.
The "worst case" with the largest contribution from the term A in (13.8) will be for $\cos\omega\Delta x = -1$, i.e. for $\omega\Delta x = \pi + 2k\pi$, $k \in \mathbb{Z}$, or by restriction to $[0, 2\pi[$:

$$\omega\Delta x = \pi \tag{13.9}$$

In order to ensure $H(\omega) \geq -1$, we obtain the condition

$$2\frac{D\Delta t}{\Delta x^2}[1 - \cos\omega\Delta x] \overset{!}{\leq} 2 \tag{13.10}$$

which means for $\omega\Delta x = \pi$

$$2\frac{D\Delta t}{\Delta x^2}[1 - (-1)] \overset{!}{\leq} 2 \quad \Leftrightarrow \quad 1 - 2\frac{D\Delta t}{\Delta x^2} \overset{!}{\geq} 0 \tag{13.11}$$

This means of stability investigation is called von Neumann stability analysis.

Let us also note that (13.9) tells us $2\omega\Delta x = 2\pi$, where Δx is the sampling rate of our signals. This means, the "worst case" above corresponds to an input wave with the highest possible frequency that can be sampled by our mesh.

13.1 Fast Explicit Diffusion (FED)

The motivation of FED can be formulated as follows:

- Linear difusion is equivalent to convolution with a Gaussian.
- Gaussians can be approximated by repeated application of a box filter. To this end, only few of such iterations are needed.
- The point will be that the box filter tells us what to do with linear diffusion. The resulting procedure can be generalised to other settings.

Let us denote a discrete box filter of length $((2n+1)\,\Delta x$, $n \in \mathbb{N}$, by

$$\left(B_{2n+1}^{\Delta x}(f)\right)_j \;:=\; \frac{1}{2n+1} \sum_{k=-n}^{n} f_{j+k} \tag{13.12}$$

The following theorem states an important connection between box filtering and explicit diffusion schemes with variable time step sizes.

Theorem 13.1.1 *The box filter $B_{2n+1}^{\Delta x}$ is equivalent to a cycle with n explicit linear diffusion steps*

$$B_{2n+1}^{\Delta x} \;=\; \prod_{i=0}^{n-1} \left(I + \Delta t_i \Delta_d\right) \;, \quad \Delta_d U \;:=\; \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2} \tag{13.13}$$

with the varying time step sizes

$$\Delta t_i \;=\; \frac{\Delta x^2}{4\cos^2\left(\pi \dfrac{2i+1}{4n+2}\right)} \tag{13.14}$$

and corresponding stopping time

$$T_n \;:=\; \sum_{i=0}^{n-1} \Delta t_i \;=\; \frac{\Delta x^2}{3} \begin{pmatrix} n+1 \\ 2 \end{pmatrix} \tag{13.15}$$

Remarks:

- The points $1/\left(\Delta t_i\right)$ are the zeroes of corresponding Chebychev polynomials over $[0,1]$.
- The Δt_i partially violate stability conditions.

While the equivalence to box filtering means that the *complete* process should be stable when observed just at T_n, the instable steps used during the process may deteriorate the result severely in practice. To cure this, it turns out that a specific re-ordering of the time step sizes Δt_i (i.e. of the indices i) suffices:

Definition 13.1.1 *Let S_n be the set of n time steps Δt_i. The Leja orderng of the Δt_i is defined via*

$$\begin{aligned} \Delta t_0 &:= \max_{\Delta t_i \in S_n} |\Delta t_i| && (13.16) \\ \underbrace{\prod_{k=0}^{j-1} |\Delta t_j - \Delta t_k|}_{\text{“multiplicative distance”}} &:= \max_{j \le l \le n} \prod_{k=0}^{j-1} |\Delta t_l - \Delta t_k| && (13.17) \end{aligned}$$

The ordered set is denoted S_n^L.

Thus, in the Leja ordering the multiplicative distances are sucessively optimised.

The complete algorithm summarises as follows:

1) Choose stopping time and determine the set S_n of time step sizes.
2) Compute Leja ordering S_n^L of S_n.
3) Compute convolution with box filter by explicit diffusion using S_n^L.
4) Repeat the process corresponding to successive application of box filters.

Let us emphasize, that it is *not* the point that explicit diffusion steps are computationally much cheaper than a pointwise convolution with $B_{2n+1}^{\Delta x}$; the latter can be implemented very efficiently. Rather than that, the linear diffusion steps can easily be generalised to nonlinear diffusion.

Chapter 14

Parabolic problems: Diffusion equations

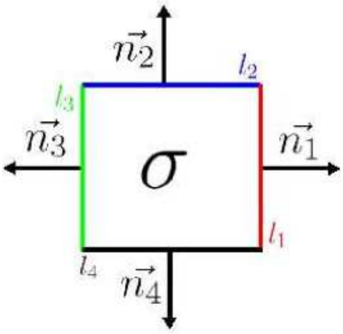
Motivation: Get an idea of diffusion filtering.

The key to understand diffusion equations is the Theorem of Gauß:

$$\int_{\sigma} \nabla(\nabla u)\vec{x} \quad = \quad \int_{\partial\sigma} \nabla u \cdot \vec{n}s, \tag{14.1}$$

where $\nabla(\nabla u) = \Delta u$, $\vec{n}$ is the outer unit normal vector of σ, and ds is a "surface element" on $\partial\sigma$.
As we usually deal with pixels in images, we have for any pixel σ:

Sketch:



- $\partial\sigma = l_1 \cup l_2 \cup l_3 \cup l_4$
- $\|\vec{n_1}\|_2 = \ldots = \|\vec{n_4}\|_2 = 1$
- $\vec{n_1} = (1,0)^\top, \vec{n_2} = (0,1)^\top,$
  $\vec{n_3} = (-1,0)^\top, \vec{n_4} = (0,-1)^\top$

Let us now look back at a complete linear diffusion equation:

$$u_t \;=\; \Delta u. \tag{14.2}$$

Integrating (14.2) over a pixel σ_i and dividing by its area we denote as $|\sigma_i|$ yields:

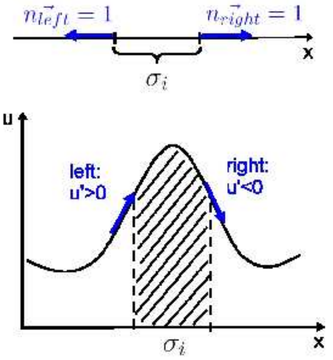
$$\frac{1}{|\sigma_i|}\int_{\sigma_i} u_t(\vec{x},t)\vec{x} \;=\; \frac{1}{|\sigma_i|}\int_{\sigma_i} \nabla(\nabla u)\vec{x} \tag{14.3}$$

Assuming smoothness of u, we draw $\frac{\partial}{\partial t}$ out of the left integal, and use (14.1) on the right hand side:

$$\frac{\partial}{\partial t}\;\underbrace{\left[\frac{1}{|\sigma_i|}\int_{\sigma_i} u(\vec{x},t)\vec{x}\right]}_{\text{average grey value over pixel}\sigma_i} \;=\; \frac{1}{|\sigma_i|}\int_{\partial\sigma_i} \nabla u\cdot\vec{n}s. \tag{14.4}$$

Let us put (14.4) into words: The temporal change of the average grey value of a pixel σ_i is determined by the flux given by ∇u at its boundary.

Sketch of 1-D situation:



left: $\nabla u\cdot\vec{n} \equiv \underbrace{u'}_{>0}\cdot(-1) < 0$

right: $\nabla u\cdot\vec{n} \equiv \underbrace{u'}_{<0}\cdot 1 < 0$

$\Rightarrow$ by the diffusion equation $u_t = u_{xx}$, the average grey value $\frac{1}{|\sigma_i|}\int_{\sigma_i} u(x,t)x$ will become smaller.

14.1 Conservation of the average grey value

Let us consider a special property of diffusion equations

$$u_t \quad = \quad \nabla \cdot (\nabla u), \tag{14.5}$$

namely the so-called divergence form: the right hand side is of the form

$$\nabla \cdot < \text{ member of the vector field } > \tag{14.6}$$

The divergence form enables the application of the Theorem of Gauß, leading to

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u_i}(t) \quad = \quad \frac{1}{|\sigma_i|}\int_{\partial\sigma_i} \nabla u \cdot \vec{n} s, \tag{14.7}$$

see (14.4), where

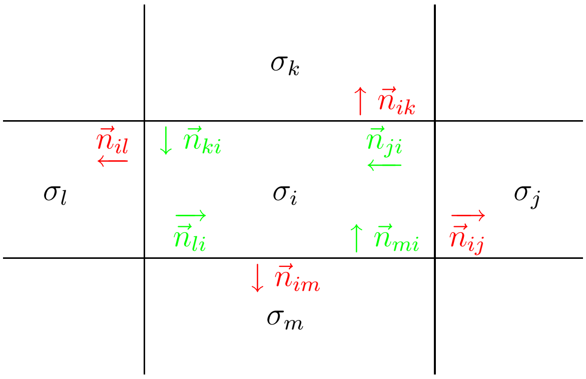
$$\overline{u_i}(t) \quad := \quad \frac{1}{|\sigma_i|}\int_{\sigma_i} u(\vec{x},t)\vec{x} \tag{14.8}$$

is the average grey value of the pixel σ_i.

In the following, let us employ, for any pixel σ_i, the ordering of boundary segments as sketched here:

$$\begin{array}{c|c|c} & l_{i2} & \\ \hline l_{i3} & \sigma_i & l_{i1} \\ \hline & l_{i4} & \end{array} \tag{14.9}$$

Also, for adjacent pixels, we specify the directions of outer unit normal vectors:


σ_k
$\uparrow \vec{n}_{ik}$
$\underset{\longleftarrow}{\vec{n}_{il}}$
$\downarrow \vec{n}_{ki}$
$\underset{\longleftarrow}{\vec{n}_{ji}}$
σ_l
σ_i
σ_j
$\overset{\longrightarrow}{\vec{n}_{li}}$
$\uparrow \vec{n}_{mi}$
$\overset{\longrightarrow}{\vec{n}_{ij}}$
$\downarrow \vec{n}_{im}$
σ_m

(14.10)

Note, that for adjacent pixels as in (14.9) - (14.10), e.g. σ_i and σ_j, we have

- $l_{i1} = l_{j3}$

- $\vec{n}_{ij} = -\vec{n}_{ji}$

As ∇u is nothing else than a vector-valued function evaluated along pixel boundaries between pixels σ_i and σ_j, we have by $\vec{n}_{ij} = -\vec{n}_{ji}$:

$$\int_{\partial\sigma_i \cap \partial\sigma_j} \nabla u \cdot \vec{n}_{ij} s \quad = \quad -\int_{\partial\sigma_i \cap \partial\sigma_j} \nabla u \cdot \vec{n}_{ji} s \tag{14.11}$$

Result No.1:
By the flux over a boundary segment $\partial\sigma_i \cap \partial\sigma_j$, no additional amount of grey is generated or annihilated. What flows out of σ_i over l_{ij} is added in σ_j, and vice versa.
Furthermore, let us briefly comment on $\nabla u \cdot \vec{n}_{ij}$. A standard formula for scalar products reads:

$$\cos \angle(\nabla u, \vec{n}_{ij}) \quad = \quad \frac{\nabla u \cdot \vec{n}_{ij}}{\|\nabla u\|_2 \cdot \|\vec{n}_{ij}\|_2} \overset{\text{here}}{=} \frac{\nabla u \cdot \vec{n}_{ij}}{\|\nabla u\|_2} \tag{14.12}$$

This means:

$$\nabla u \cdot \vec{n}_{ij} \quad = \quad \|\nabla u\|_2 \cdot \cos \angle(\nabla u, \vec{n}_{ij}). \tag{14.13}$$

Result No.2:

- if $\nabla \parallel \vec{n}_{ij}$ and they point in the <u>same</u> direction, then $\nabla u \cdot \vec{n}_{ij} = \|\nabla u\|_2$ is maximal,
- if $\nabla u \parallel \vec{n}_{ij}$ and they point in <u>opposite</u> directions, the flow given by $\nabla u \cdot \vec{n}_{ij} = -\|\nabla u\|_2$ is minimal,
- if $\nabla u \perp \vec{n}_{ij}$, we have zero flow: $\nabla u \cdot \vec{n}_{ij} = 0$.

Let us now consider diffusion over a complete image, i.e., over a connected set $\{\sigma_i\}_{i=1,\ldots,N}$ of pixels.

Sketch:

			σ_{N-1}	σ_N
$\vdots$	$\iddots$			
σ_1	σ_2	$\ldots$		

$\partial\Omega$

(14.14)

In this context, it is practical to use the <u>average grey value</u> at time t, obviously given by

$$\overline{U}(t) \quad := \quad \frac{1}{N} \sum_{i=1}^{N} \overline{u}_i(t) \tag{14.15}$$

Then the temporal change of the average grey value van be computed as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\overline{U}(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{1}{N}\sum_{i=1}^{N}\overline{u}_i(t)\right] = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_i(t)\right] \\ &\overset{(14.7)}{=} \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|\sigma_i|}\int_{\partial\sigma_i}\nabla u\cdot\vec{n}s\right]\end{aligned} \tag{14.16}$$

Having again a look at our sketch (14.14), we observe that for any pixel σ_i, we can decompose $\partial\sigma_i$ in contributions

$$\partial\sigma_i \cap \partial\sigma_j, \qquad j \in N(i), \tag{14.17}$$

where $N(i)$ is the index set corresponding to neighboring pixels of σ_i, and

$$\partial\sigma_i \cap \partial\Omega, \tag{14.18}$$

if some part of $\partial\sigma_i$ is not shared with another pixel. Thus, we obtain from (14.16)-(14.18).

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{U}(t) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|\sigma_i|}\sum_{j\in N(i)}\int_{\partial\sigma_i\cap\partial\sigma_j}\nabla u\cdot\vec{n}_{ij}s + \frac{1}{|\sigma_i|}\int_{\partial\sigma_i\cap\partial\Omega}\nabla u\cdot\vec{n}s\right] \tag{14.19}$$

Let us now employ (14.11). For this, let us note, that by the summation over all pixels $i = 1,\dots,N$ in (14.19), we have at each inner boundary segment $\partial\sigma_k \cap \partial\sigma_{k'}$, $k, k' \in \{1,\dots,N\}$, exactly <u>two</u> contributions:

$$\begin{aligned}&\sum_{i=1}^{N}\left[\frac{1}{|\sigma_i|}\sum_{j\in N(i)}\int_{\partial\sigma_i\cap\partial\sigma_j}\nabla u\cdot\vec{n}_{ij}s\right]\Bigg|_{\partial\sigma_k\cap\partial\sigma_{k'}} \\ =& \frac{1}{|\sigma_i|}\int_{\partial\sigma_k\cap\partial\sigma_{k'}}\nabla u\cdot\vec{n}_{kk'}s + \frac{1}{|\sigma_i|}\int_{\partial\sigma_{k'}\cap\partial\sigma_k}\nabla u\cdot\vec{n}_{k'k}s.\end{aligned} \tag{14.20}$$

Assuming that the pixels are the same, $|\sigma_k| = |\sigma_{k'}|$, which we usually have, since pixels in an image do not vary in size, we obtain equivalently to (14.20):

$$\frac{1}{|\sigma_i|}\left[\int_{\partial\sigma_k\cap\partial\sigma_{k'}}\nabla u\cdot\vec{n}_{kk'}s + \int_{\partial\sigma_{k'}\cap\partial\sigma_k}\nabla u\cdot\vec{n}_{k'k}s\right]. \tag{14.21}$$

Since $(\partial\sigma_k \cap \partial\sigma_{k'}) = (\partial\sigma_{k'} \cap \partial\sigma_k)$, we use (14.11) with $\vec{n}_{kk'} = -\vec{n}_{k'k}$ to conclude:

$$\begin{aligned}&\int_{\partial\sigma_k\cap\partial\sigma_{k'}} \nabla u\cdot\vec{n}_{kk'}s + \int_{\partial\sigma_{k'}\cap\partial\sigma_k} \nabla u\cdot\vec{n}_{k'k}s \\ =\;& \int_{\partial\sigma_k\cap\partial\sigma_{k'}} \nabla u\cdot\vec{n}_{kk'}s - \int_{\partial\sigma_{k'}\cap\partial\sigma_k} \nabla u\cdot\vec{n}_{kk'}s = 0.\end{aligned} \tag{14.22}$$

As we have exactly the same result than in (14.22) at any inner pixel boundary segment, we obtain

$$\sum_{i=1}^{N}\left[\frac{1}{|\sigma_i|}\sum_{j\in N(i)}\int_{\partial\sigma_i\cap\partial\sigma_j}\nabla u\cdot\vec{n}_{ij}s\right] \;=\; 0, \tag{14.23}$$

and thus

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{U}(t) \;=\; \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|\sigma_i|}\int_{\partial\sigma_i\cap\partial\Omega}\nabla u\cdot\vec{n}s\right]. \tag{14.24}$$

Result No.3:

- By the divergence form of the diffusion, all fluxes in the inner part of an image cancel each other when computing the complete flow.
- Changes of the average grey value of an image occur only due to fluxes over the image boundary if the filter is of divergence form.

Of specific interest – as it is an invariant or stability property of the diffusion problem – is the case

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{U}(t) \;=\; 0 \tag{14.25}$$

By (14.24), the property (14.25) can only be achieved if

(i) inflow and outflow over the image boundary are exactly in balance, or

(ii) $\nabla u \equiv \vec{0}$ along $\partial\Omega$.

We concentrate for the moment on situation (ii), leading to the following notion

Definition 14.1.1 *The boundary condition of the form*

$$\nabla u \;=\; 0 \qquad \text{at } \partial\Omega \tag{14.26}$$

is called von Neumann boundary conditions.

14.2 Summary of §14.1

- By von Neumann boundary conditions, the average grey value is conserved forever.
- By the divergence form, it is important that no amount of grey is generated of annihilated by fluxes over pixel boundaries.
- Changes of the average grey value take place only by the fluxes over the image boundary.

Chapter 15

Construction of numerical schemes for diffusion filters

We begin with the basic form of a diffusion equation.

$$u_t \quad = \quad \nabla \cdot (D\nabla u). \tag{15.1}$$

Using average greyvalues, this leads as in §14 to the evolution equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_i(t) \quad = \quad \frac{1}{|\sigma_i|}\int_{\partial\sigma_i}(D\nabla u)\cdot\vec{n}s, \tag{15.2}$$

i.e., taking into account the four pixel boundary segments $l_{i1}, \ldots, l_{i4}$, to

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_i(t) \quad = \quad \frac{1}{|\sigma_i|}\sum_{k=1}^{4}\int_{l_{ik}}(D\nabla u)\cdot\vec{n}s. \tag{15.3}$$

15.1 Time integration

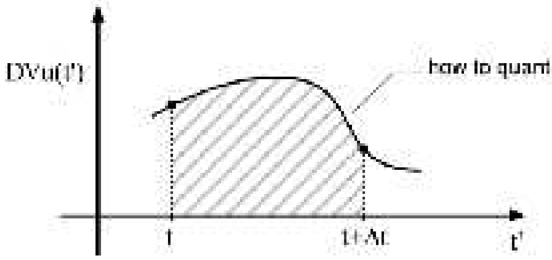
In order to obtain a numerical scheme from (15.3), one has to integrate for every pixel σ_i the ordinary differential equation (ODE) (15.3) in time. For the left hand side of (15.3) follows:

$$\int_t t+\delta t\left[\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_i(t)\right]t \quad = \quad \overline{u}_i(t+\delta t)-\overline{u}_i(t). \tag{15.4}$$

For the right hand side we have

$$\frac{1}{|\sigma_i|}\int_t t+\delta t\left\{\sum_{k=1}^{4}\int_{l_{ik}}(D\nabla u)\cdot\vec{n}s\right\}t \tag{15.5}$$

This means, we have to integrate in time the values of $D\nabla u$ along $l_{i1}, \dots, l_{i4}$, however $D\nabla u$ generally changes during the time evolution over $[t, t+\delta t]$.
Modelled in 1-D, we have to integrate at a spatial integration point the function $D\nabla u(t)$, the space variable is kept fixed:



(15.6)

For a numerical method, it is appropriate to consider a numerical approximation, or quadrature, of the above integral, which is distinguished via the choice of the quadrature rule applied to $D\nabla u(t')$ over $[t, t+\delta t]$.

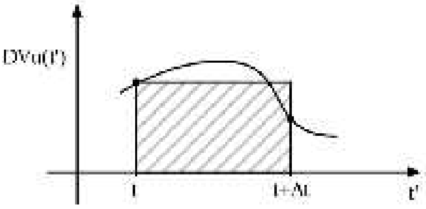
Definition 15.1.1 *Let us denote the time integration weight by*

$$t^* \;:=\; t\Theta + (1-\Theta)[t+\delta t], \qquad \Theta \in [0,1]$$

(i) The choice

$$\Theta = 1 \;\Leftrightarrow\; t^* = t \tag{15.7}$$

is called the Euler forward time integration, it is first-order accurate.
Sketch:

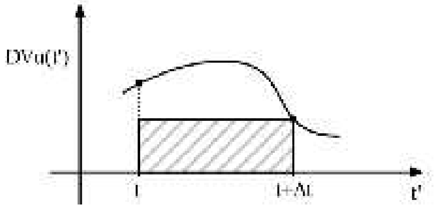


Quadrature by choosing the left boundary point of $[t, t+\delta t]$.

(ii) The choice

$$\Theta = 0 \;\Leftrightarrow\; t^* = t+\delta t \tag{15.8}$$

is called the Euler backward time integration, it is also first-order accurate.
Sketch:

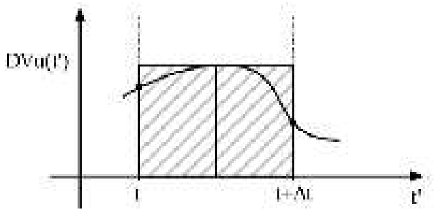


Quadrature by choosing the right boundary point of $[t, t + \delta t]$.

(iii) *The choice*

$$\Theta = \frac{1}{2} \quad \Leftrightarrow \quad t^* = t + \frac{\delta t}{2} \tag{15.9}$$

is called the Crank-Nicolson scheme. It is second-order accurate.
Sketch:



Quadrature by choosing the middle of $[t, t + \delta t]$.

(iv) *For unspecified* Θ*, or varying* Θ *during a computation, the general choice*

$$t^* \;=\; t \cdot \Theta + (1 - \Theta)[t + \delta t] \tag{15.10}$$

is often called the Θ*-scheme.*

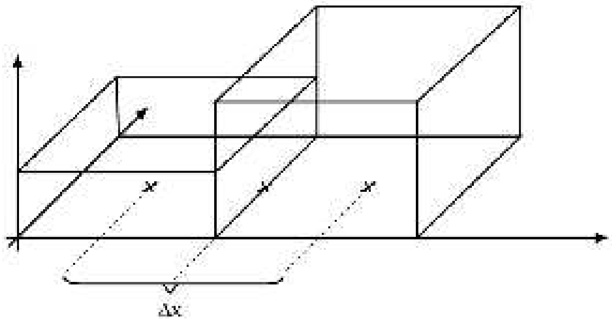
Remarks:

(i) For only one integration point, the Crank-Nicolson offers the best accuracy.

(ii) Let us stress, that image data is given at time t. Thus, except for $\Theta = 1$, i.e., except for the Euler forward method, all other choices of Θ take into account future values at $t + \delta t$. The Euler forward method is explicit. All other Θ-methods are implicit.

(iii) It is possible to circumvent some of the difficulties associated with the implicitness of the Crank-Nicolson scheme: In a first step, one may compute from the data at time t a predicted value at the integration time $t + \frac{\delta t}{2}$, and use this predicted value for integration. This leads to explicit Predictor-Corrector, or Runge-Kutta methods.

Sketch:



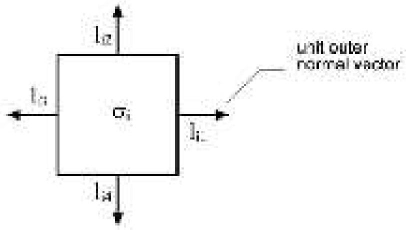
15.2 Spatial integration

Let us consider for the moment the most simple time integration method, i.e., the Euler forward scheme. This means, (15.5) yields, as the length of the interval $[t, t + \delta t]$ is δt:

$$\frac{1}{|\sigma_i|} \int_t t + \delta t \left[\sum_{k=1}^{4} \int_{l_{ik}} (D\nabla u) \cdot \vec{n} s \right] t' \;\approx\; \frac{\delta t}{|\sigma_i|} \sum_{k=1}^{4} \int_{l_{ik}} (D\nabla u)\Big|_t \cdot \vec{n} s. \quad (15.11)$$

We now omit for a simplified notation the time index t. Then, let us remember the pixel geometry.

Sketch:

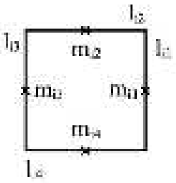


Assuming uniform pixel sizes, we can compute

$$\begin{aligned} |\sigma_i| &= l_{ik}^2 \qquad \text{for any } k \in \{1,2,3,4\}, \\ &= h^2 \qquad \text{for } h = \Delta x = \Delta y \text{ (as assumed).} \end{aligned} \quad (15.12)$$

We choose now as quadrature points along the boundary segments l_{ik} their mid points:

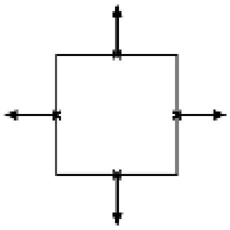
Sketch:



Then, the boundary integrals in (15.11) are approximated as

$$\int_{l_{ik}} (D\nabla u) \cdot \vec{n}s \quad \approx \quad h \cdot [(D\nabla u) \cdot \vec{n}]\Big|_{m_{ik}} \tag{15.13}$$

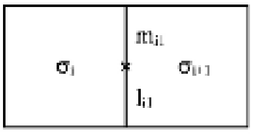
It is clear, how the unit outer normal vectors $\vec{n}$ are chosen in m_{ik}:



- along x-direction, we have $\vec{n} = (\pm 1, 0)^\top$
- along y-direction, we have $\vec{n} = (0, \pm 1)^\top$.

Thus, at each point m_{ik}, we only need the x- or y-component of $D\nabla u$, respectively. Let us consider the right boundary segment of σ_i:

Sketch:



As D typically relies in a nonlinear way on u, there is the problem, how to compute u along l_{i1} from the given greyvalue averages $\overline{u}_i, \overline{u}_{i+1}$. In diffusion problems, one can often circumvent this problem, computing

$$D_i \equiv D(\overline{u}_i) \quad \text{and} \quad D_{i+1} \equiv D(\overline{u}_{i+1}) \tag{15.14}$$

and choosing at m_{i1}:

$$D\Big|_{m_{i1}} =: \; D_{i+\frac{1}{2}} := \frac{D_i + D_{i+1}}{2}. \tag{15.15}$$

What remains is to approximate $\nabla u|_{m_{i1}}$. This is easiest done as:

$$\nabla u\Big|_{m_{i1}} = \nabla u \cdot n\Big|_{i+\frac{1}{2}} \approx \frac{1}{h}(\overline{u}_{i+1} - \overline{u}_i) \qquad (15.16)$$

In the other directions, the procedure is analogously. **Remarks:**

(i) Further refinements are possible, especially considering the evaluation of a full diffusion tensor D. For instance, one may first evaluate $D\nabla u$, and discretise the arising contributions.

(ii) To achieve higher order spatial accuracy, the procedure can become quite complicated. It is also possible not to consider the "physics" of diffusion in such a detail as above; however, a lack of accuracy in such a model may generate principle errors.

(iii) For $D = I$, we retrieve in the end from (15.14) - (15.16) the usual difference formula, for $h = \Delta x = \Delta y$, and $t = n\delta t$,

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^n - \frac{\delta t}{\Delta x^2}\left(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n\right) \qquad (15.17) \\ &\quad -\frac{\delta t}{\Delta y^2}\left(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n\right). \end{aligned}$$

(iv) Technically, we have used in this lecture a so-called Finite-Volume set-up, which is equivalent to Finite Differences but highlights the "physics" of diffusion.

Chapter 16

Hyperbolic PDEs

Motivation: Hyperbolic PDEs may arise, in Shape from Shading, mathematical morphology, skeletonisation, and other processes.

In order to assess general numerical principles for this type of equations, it is useful to consider 1-D hyperbolic PDEs of the form

$$u_t + f(u)_x = 0, \quad u \equiv u(x,t), \quad x \in \mathbb{R}, t > 0. \tag{16.1}$$

The function f in (16.1) is called flux function.

16.1 Conservation form and consistency

As the PDE (16.1) is of divergence form, it seems to be at first glance similar to diffusion equations. This leads us to formulate

Definition 16.1.1 *A discretisation of* (16.1) *which can be written in the form*

$$u_j^{n+1} \;=\; u_j^n - \frac{\Delta t}{\Delta x}(g_{j+\frac{1}{2}} - g_{j-\frac{1}{2}}) \tag{16.2}$$

is called (grey-value) conservative. The function g, where for a 3-point scheme

$$\begin{aligned} g_{j+\frac{1}{2}} &\equiv g(u_j^n, u_{j+1}^n), \\ g_{j-\frac{1}{2}} &\equiv g(u_{j-1}^n, u_j^n) \end{aligned} \tag{16.3}$$

holds, is called numerical flux function.

Remarks:

- The average grey value

$$\frac{1}{N}\sum_{j=1}^{N} u_j$$

over an interval of N pixels does not change, but only due to fluxes at the interval ends:

$$\begin{aligned}\sum_{j=1}^{N} u_j^{n+1} &= \sum_{j=1}^{N} \left(u_j^n - \frac{\Delta t}{\Delta x}(g_{j+\frac{1}{2}} - g_{j-\frac{1}{2}}) \right) \qquad (16.4)\\ &= \sum_{j=1}^{N} u_j^n - \frac{\Delta t}{\Delta x}(g_{N+\frac{1}{2}} - g_{\frac{1}{2}})\end{aligned}$$

- We can write the consistency of the method (16.2), (16.3) in terms of the condition

  $$g(u, u) = f(u) \qquad (16.5)$$

  For more arguments of g, (16.5) can be extended accordingly.

16.2 Weak formulation

In general, solutions of hyperbolic PDEs of type (16.1) have a devastating feature: even initial data given in the form of functions from C^∞ (e.g. sine, cosine) produce discontinuous solutions. This is relevant as e.g. edges are important discontinous solution features. As $\frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}$ are not defined at discontinuities, we want to get rid of the derivatives. This is done by the following trick. First, we multiply the PDE (16.1) with a so-called test function $\varphi(x, t) \in C_0^\infty$, where the lower index 0 stands for finite support, i.e.

$$\begin{aligned}\text{support}(\varphi) &= \{(x, t)|\varphi(x, t) \neq 0\} \cup \partial\{(x, t)|\varphi(x, t) \neq 0\} \qquad (16.6)\\ &\subseteq [a, b] \times [c, d], |a|, |b|, |c|, |d| < \infty,\end{aligned}$$

yielding

$$\varphi u_t + \varphi f(u)_x = 0. \qquad (16.7)$$

Now we integrate (16.7) over space (from $-\infty$ to ∞) and time (starting with $t = 0$, up to ∞):

$$\int_0^\infty \int_{-\infty}^\infty \varphi u_t + \varphi f(u)_x \mathrm{d}x\mathrm{d}t = 0. \qquad (16.8)$$

Assuming that we can switch the integration order of space and time integration, we apply the partial integration rule to obtain for the corresponding parts in (16.8):

$$\begin{aligned}\int_0 \infty \varphi u_t t &= [\varphi u]|_0^\infty - \int_0 \infty \varphi_t u t \qquad (16.9)\\ &= -\varphi(x, 0)u(x, 0) - \int_0 \infty \varphi_t u t\end{aligned}$$

(adding relevant arguments) since $\varphi \equiv 0$ for $t \to \infty$ as it has finite support, and

$$\begin{aligned} \int_{-\infty} \infty \varphi f(u)_x x &= [\varphi f(u)]|_{-\infty}^{\infty} - \int_{-\infty} \infty \varphi_x f(u) x \qquad (16.10) \\ &= -\int_{-\infty} \infty \varphi_x f(u) x \end{aligned}$$

since $\varphi \equiv 0$ for $x \to \pm\infty$ as it has finite support, see (16.6). Plugging (16.9)-(16.10) into (16.8) leads, after a few trivial rearrangements, to

$$\int_0^\infty \int_{-\infty}^\infty \varphi_t u + \varphi_x f(u) \mathrm{d}x \mathrm{d}t = -\int_{-\infty} \infty \varphi(x,0) u(x,0) x. \qquad (16.11)$$

Definition 16.2.1 *For arbitrarily chosen test functions φ and for initial values $u(x,0)$, a solution $u(x,t)$ of* (16.11) *is called a weak solution, or distributional solution, of the PDE* (16.1).

Note that derivatives are now taken from the smooth test function φ, not from u and $f(u)$ anymore. Note also that the relation (16.11) needs to hold for an infinite number of test functions.
It is crucial to understand that (16.11) is the relevant form, not the original PDE (16.1) anymore. The latter is meaningful only in parts where u is at least in C^1.

16.3 Diffusion a.k.a. viscosity solutions

While we have gained that weak solutions admit edges in solutions of PDEs, there is a trade-off. Weak solutions are in general not unique anymore. Without going into details, let us simply notice the fact that once a discontinuity arises, infinitely many (!) weak solutions are generated.
There is only one particular, “intuitively” correct weak solution we are usually interested in. The question is, how to pick exactly this one out, and how to compute it.
The basic idea is as follows. First we add a small diffusion term to the original PDE (16.1), controlled by a small parameter $\varepsilon > 0$:

$$u_t + f(u)_x = \varepsilon u_{xx}. \qquad (16.12)$$

By the diffusion term εu_{xx}, any solution of (16.12) is smeared out slightly and will be in C^∞. The idea is then to retreive the correct solution of $u_t + f(u)_x = 0$ in the limit of vanishing diffusion. This leads to

Definition 16.3.1 *By the physical interpretation of the εu_{xx}-term arising in fluid dynamics, a solution of* (16.12) *, for $\varepsilon \to 0$, is called viscosity solution of the PDE* (16.1).

Remark: The viscosity solution is also often called entropy solution in the literature, a notion arising from gas dynamics.

16.4 Theory of numerical methods

We are lucky in two good news in the form of the following theorems:

Theorem 16.4.1 (of Lax and Wendroff (1960)) *Given a numerical method approximating* (16.1) *which is consistent and conservative. Let $\lambda = \frac{\Delta t}{\Delta x}$ be a constant. Then, for $\Delta t, \Delta x \to 0$, if the scheme converges to some function, then u is a weak solution of* (16.1)*.*

For the second theorem, let us before give the following definition:

Definition 16.4.1 *Let us define explicit monotonic methods as follows. If we write an explicit 3-point scheme as*

$$u_j^{n+1} \;=\; \mathcal{H}(u_{j-1}^n, u_j^n, u_{j+1}^n) \tag{16.13}$$

then the scheme is monotone, if and only if

$$\frac{\partial \mathcal{H}}{\partial u_i^n} \;\geq\; 0 \tag{16.14}$$

for all $i \in \{j-1, j, j+1\}$ and for all possible values arising as arguments of $\mathcal{H}$.

Remark: An exact viscosity solution of (16.1) satisfies some monotonicity property:

$$u_1(x,0) \geq u_2(x,0) \;\Rightarrow\; u_1(x,t) \geq u_2(x,t) \qquad \forall t > 0, \tag{16.15}$$

if u_1, u_2 are put as initial conditions into (16.11). The above defined notion of monotonicity of a method ensures that the analogous property of approximate solutions holds at the discrete level.

Theorem 16.4.2 (of Crandall and Majda (1980)) *Consistent, conservative and monotone schemes approximating* (16.1) *also approximate its unique viscosity solution.*

To summarise, having a monotone scheme, everything is good - at least in theory. However, there is also a stone in the garden of this theory:

Theorem 16.4.3 (of Goodman and LeVeque (1985)) *Monotone methods are always only first order accurate.*

Chapter 17

High-Resolution Schemes for Hyperbolic PDEs

We now study a successful principle for solving hyperbolic PDEs by the example of the model problem

$$u_t + f(u)_x = 0 \tag{17.1}$$

Basic discretisations work as follows:

- Monotone schemes are only first-order accurate, i.e. they introduce a blurring effect, but they are needed at discontinuities.
- Higher-order schemes are accurate for smooth solutions, but they are oscillatory at discontinuities.

The idea is now to marry these two building blocks and to define a hybrid scheme. The construction is guided by the following principles:

- At discontinuities, apply the low order, monotone scheme.
- At smooth parts, apply the high-order scheme.
- Perform a mixture depending on the smoothness of the data.

We consider consistent, conservative approximations of (17.1) of the form

$$U_j^{m+1} = U_j^m - \frac{\Delta t}{\Delta x}\left(g_{j+\frac{1}{2}} - g_{j-\frac{1}{2}}\right) \tag{17.2}$$

Identifying

$$\begin{cases} g^H_{j+\frac{1}{2}}, & \text{flux of high-order method} \\ g^L_{j+\frac{1}{2}} & \text{flux of low-order method} \end{cases} \tag{17.3}$$

We may introduce a “switch function” into the numerical fluxes from (17.2) as by

$$g_{j+\frac{1}{2}} := g^L_{j+\frac{1}{2}} + \phi_{j+1} \left(g^H_{j+\frac{1}{2}} - g^L_{j+\frac{1}{2}} \right) \tag{17.4}$$

This means that for $\phi_{j+\frac{1}{2}} = 1$ we get the high-order flux and for $\phi_{j+\frac{1}{2}} = 0$ the low-order flux. We will allow ϕ to take on a wider range of values.
Following the construction idea, we will base the values of ϕ on a smoothness measure.
For a flow directed from left to right, we will use the directed ratio of consecutive gradients.

$$\Theta^+_j := \frac{U^m_j - U^m_{j-1}}{U^m_{j+1} - U^m_j} \tag{17.5}$$

For a flow in opposite direction we choose

$$\Theta^-_j := \frac{U^m_{j+1} - U^m_j}{U^m_j - U^m_{j-1}} \tag{17.6}$$

One can prove the following assertions for the limiter ϕ.

Theorem 17.0.4 (Theorem of Harten & Sweby) *For*

$$0 \leq \frac{\phi(\Theta)}{\Theta} \leq 2 \text{ and } 0 \leq \phi(\Theta) \leq 2 \tag{17.7}$$

the hybrid scheme defined by (17.2)-(17.4) does not introduce oscillations. If, in addition, ϕ satisfies

$$\phi(1) = 1 \tag{17.8}$$

and if it is pointwise a convex combination of (17.3) then the hybrid scheme will be of higher order, except at discontinuities and data extrema.

The boundaries of the resulting “allowed” regions are made up of the *Minmod-Limiter*

$$\phi(\Theta) = \max(0, \min(1, \Theta)) \tag{17.9}$$

and the Superbee-Limiter

$$\phi(\Theta) = \max(0, \min(0, \min(1, 2\Theta), \min(\Theta, 2)) \tag{17.10}$$

A practical issue arising for all explicit schemes is the so-called CFL-condition (after Courant, Friedrichs and Lewy (1928)).
We study it by the example of the linear advection equation

$$u_t = au_x = 0, \quad a > 0 \tag{17.11}$$

and its upwind discretisation

$$U_j^{m+1} = U_j^m - \frac{\Delta t}{\Delta x} a (U_j^m - U_{j-1}^m) \tag{17.12}$$

Let us simplify the initial data as

$$U_0^0 := 1, \quad U_k^0 := 0 \text{ for } k \neq 0 \tag{17.13}$$

For $a \cdot \Delta t = \Delta x$ we obtain by (17.12) and (17.13)

$$U_1^1 = 1, \quad U_k^1 = 0 \text{ for } k \neq 1 \tag{17.14}$$

i.e. the value $u_0^0 = 1$ has travelled in one time step exactly one point in flow direction. Investigating (17.12), we see that

$$U_j^{m+1} = (1 - \frac{\Delta t a}{\Delta x}) U_j^m + \frac{\Delta t a}{\Delta x} U_{j-1}^m \tag{17.15}$$

is of the format of a convex combination, and for

$$\Delta t > \frac{\Delta x}{a} \tag{17.16}$$

we would not obtain that U_j^{m+1} is in the convex hull of U_j^m and U_{j-1}^m.
We can give the condition

$$\Delta t \leq \frac{\Delta x}{a} \tag{17.17}$$

an interpretation. Rewriting it as

$$a \Delta t \leq \Delta x \tag{17.18}$$

it means that information from any one point j that travels with velocity a is only allowed to travel in one time step up to the next point $j \pm 1$ which is Δx away.
This CFL-Condition (17.18) is a necessary property for stability.
In non-linear problems one must compute the largest velocity $a_{\max}$ within the computational domain. Then the time step size Δt must be chosen such that (17.18) is satisfied for $a = a_{\max}$. This has to be done for each time step.

Chapter 18

Upwinding

The most popular schemes used for hyperbolic PDEs in image processing are so-called upwind schemes. Let us illustrate the idea at hand of the linear advection equation

$$u_t + au_x \quad = \quad 0. \tag{18.1}$$

The PDE (18.1) needs to be supplemented by an initial condition

$$u(x,0) \quad := \quad u_0(x). \tag{18.2}$$

The solution of (18.1)-(18.2) is

$$u(x,t) \quad := \quad u_0(x-at). \tag{18.3}$$

Let us verify this. Assume (18.3) is true, then:

$$\begin{aligned} \frac{\partial}{\partial t}u(x,t) &= \frac{\partial}{\partial t}[u_0(x-at)] = u_0'(x-at)\cdot(-a), & (18.4)\\ a\frac{\partial}{\partial x}u(x,t) &= a\cdot\frac{\partial}{\partial x}[u_0(x-at)] = au_0'(x-at)\cdot 1. & (18.5)\end{aligned}$$

Computing (18.4) and (18.5) gives zero as required by solving the PDE.

Let us stress, that we observe that the hyperbolic transport only goes in one direction determined by the sign of a at each point x. This is in contrast to diffusion.

The notion of “upwinding” comes from thinking of a sail boat which turns its sails into the direction of the wind. For a numerical method this means in analogy, that we have a one-sided stencil which is turned into the “wind direction” described by the velocity a:

- If we are sitting in a pixel x_j and consider transport to the right, then we feel the wind blowing from the left and use information from pixel x_{j-1} to discretise u_x:

$$u_x \approx \frac{u_j^n - u_{j-1}^n}{\Delta x} \tag{18.6}$$

- Analogously, for $a < 0$ we feel the wind blowing from the right and use information from pixel x_{j+1} to discretise u_x:

$$u_x \approx \frac{u_{j+1}^n - u_j^n}{\Delta x} \tag{18.7}$$

Using the Euler time discretisation method, we obtain for $a > 0$ the <u>upwind scheme</u>

$$u_j^{n+1} = u_j^n - a\frac{\Delta t}{\Delta x}(u_j^n - u_{j-1}^n), \tag{18.8}$$

for $a < 0$ the procedure is analogously.
We may write this in terms of a scheme function H as in (16.13), which gives

$$H(u_{j-1}^n, u_j^n) = u_j^n - a\frac{\Delta t}{\Delta x}(u_j^n - u_{j-1}^n) = \left(1 - \frac{a\Delta t}{\Delta x}\right) u_j^n + \frac{a\Delta t}{\Delta x} u_{j-1}^n. \tag{18.9}$$

The scheme is <u>monotone</u> under restriction on the time step size:

$$\begin{aligned} \frac{\partial H}{\partial u_{j-1}^n} &= \frac{a\Delta t}{\Delta x} \geq 0 \qquad \text{for } a > 0 \text{ (see above)}, \\ \frac{\partial H}{\partial u_j^n} &= 1 - \frac{a\Delta t}{\Delta x} \overset{!}{\geq} 0 \qquad \Leftrightarrow \Delta t \leq \frac{\Delta x}{a} \end{aligned} \tag{18.10}$$

Remarks:
The condition on the time step size (18.10) is called <u>CFL-condition</u> (Courant, Friedrichs, Lewy (1928)). A restriction on the time step size is typical for hyperbolic and parabolic problems. The meaning is, that information can only be transported at a rate of up to one pixel per time step.

Chapter 19

Stability

In the analysis of numerical schemes for differential equations, three concepts are intimately connected (Lax and Richtmyer (1956)):

- Consistency
- Stability
- Convergence

Consistency means especially:
The local truncation error is an expression of the mesh width h and goes to zero if $h \to 0$.

Convergence means:
For vanishing mesh width $h \to 0$, the solution of the discrete problem - i.e. of the scheme - becomes identical to the solution of the differential problem - i.e. of the PDE. This is the "ultimate property" saying that discrete and differential world fit.

What is the difference between consistency and convergence?
To give an answer, let us consider some approximation of a PDE over $\Omega = [0, 1]$, having at pixel i the local truncation error $L_i(h)$. Let us assume we have n pixels, or cells, so that $n \cdot h = 1$, i.e., Ω is made up of n cells of width h. Then, at each point the local truncation error $L_i(h)$ arises, i.e. over Ω we obtain the total approximation error $E(h) = \sum_{i=1}^{n} L_i(h)$.

Then the following issue arises:
While the individual $L_i(h)$ go to zero for $h \to 0$, there may be some subset $p \subset \Omega$

for which this goes very slowly. For $h \to 0$, and consequently $n \to \infty$ as $n \cdot h = 1$, we get

$$E(h) \quad = \quad \underbrace{\sum_{i, ih \notin p} L_i(h)}_{(a)} + \underbrace{\sum_{i, ih \in p} L_i(h)}_{(b)} . \tag{19.1}$$

While the individual terms $L_i(h)$ arising in (b) may go to zero, the sum may diverge.

Is there a remedy? Exactly the remedy to this phenomenon is often called stability property. Its role is to ensure a condition so that convergence (and $E(h) \to 0$) is guaranteed.

Stability can be understood as a combination of three aspects.

- Aspect No.1: We say that a numerical approximation of a PDE is stable, if it mimics important structural properties of the discretised PDE. As there may be several of such properties, one speaks of "stability with respect to $\langle$ important property $\rangle$".
  Example: For the Laplace equation with Dirichlet boundary conditions, we identified a minimum-maximum-principle. A stable scheme should fit a discrete minimum-maximum-principle.

- Aspect No.2: Stability is usually related to a bound on the numerical solution, or of a number extracted from it. Note that such a bound may rely on the norm in use.
  Example: For hyperbolic PDEs, the monotony property implies because of $a \geq u_0 \geq b \;\Leftrightarrow\; a \geq u \geq b$ an upper and lower bound in the L_∞-norm on the solution. A stable scheme is stable with respect to the L_∞-norm.

- Aspect No.3: A stability notion should be a necessary ingredient to prove convergence (along with corresponding assumptions on the solution).
  Example: Rotational invariance is a structural property of solutions of the Laplace equation. However, it is not crucial for convergence. Thus, it is not a stability notion, but refers to an accuracy problem in the discrete setting.

What about non-linear stability? Above we mentioned the Laplace equation which is linear. However, the discrete minimum-maximum principle is also meaningful in the nonlinear case, and it is our most important stability property in many cases, as it ensures the absence of numerical oscillations that can be misinterpreted as noise.

Chapter 20

Fast Marching

The goal is now to construct an efficient algorithm for solving the hyperbolic Eikonal equation

$$|\nabla T|F \;=\; 1 \tag{20.1}$$

subject to boundary conditions. PDEs of this type arise, e.g. in Shape from Shading. The PDE (20.1) describes the motion of a curve in its normal directions. Thereby,

- $T(x, y) \in \mathbb{R}$ is the arrival time, i.e. the time the initial curve arises at (x, y),
- F is the speed function, $F > 0$ means that the curve moves ourward.

For establishing what can be done numerically, we consider the general Hamilton-Jacobi equation

$$U_t + H(U_x, U_y) \;=\; 0. \tag{20.2}$$

The function H is known as the "Hamiltonian", for the boundary value problem (20.1) we have the Hamiltonian

$$H(U_x, U_y) \;=\; F\sqrt{U_x^2 + U_y^2} - 1 \tag{20.3}$$

Let us focus on a 1-D version of (20.2), i.e.

$$U_t + H(U_x) \;=\; 0. \tag{20.4}$$

If we let $U_x = u$ and differentiate,

$$\begin{aligned} \frac{\partial}{\partial x}[U_t + H(u)] &= (U_t)_x + H(u)_x \qquad (20.5)\\ &= (U_x)_t + H(u)_x, \end{aligned}$$

i.e.

$$u_t + H(u)_x \;=\; 0. \tag{20.6}$$

The PDE (20.6) is of the form (16.1) we are already used to, this is called conservation law form. Identifying the conservation form from (16.1) with (20.6), we obtain $H(u) = g(u, u)$ with

$$H_{i+\frac{1}{2}} \approx g(u_i^n, u_{i+1}^n), \qquad H_{i-\frac{1}{2}} \approx g(u_{i-1}^n, u_i^n) \tag{20.7}$$

at the points $(i \pm \frac{1}{2})\Delta x$, respectively.

Going now from (20.6) back to the PDE (20.4), we see that we can write this as

$$U_t + H(u) = 0, \qquad u := U_x. \tag{20.8}$$

Needing a value for $H(u_i^n)$, we get from (20.7) by shifting indices

$$H(u_i^n) \;\approx\; g(u_{i-\frac{1}{2}}, u_{i+\frac{1}{2}}) = g(U_x\big|_{i-\frac{1}{2}}, U_x\big|_{i+\frac{1}{2}}) \tag{20.9}$$

Simple and convenient discretisations of $U_x\big|_{i\pm\frac{1}{2}}$ are given by

$$U_x\big|_{i-\frac{1}{2}} \approx \frac{U_i - U_{i-1}}{\Delta x}, \qquad U_x\big|_{i+\frac{1}{2}} \approx \frac{U_{i+1} - U_i}{\Delta x}. \tag{20.10}$$

A convenient numerical flux function $\hat{g}$ for the simple Hamiltonian

$$H(u) \;=\; u^2 \tag{20.11}$$

which we use as a building block is

$$\hat{g}(u_1, u_2) \;=\; \max(u_1, 0)^2 + \min(-u_2, 0)^2, \tag{20.12}$$

after Osher and Sethian (1988), or

$$\hat{g}(u_1, u_2) \;=\; \max(u_1, -u_2, 0)^2, \tag{20.13}$$

due to Rouy and Tourin (1992).

Extending the above 1-D procedure in a natural way to our 2-D equation $F|\nabla U| = 1$, we have instead of $H(u) = u^2$ the Hamiltonian $H(u, v) = F\sqrt{u^2 + v^2} - 1$. This means, at pixel $(i\Delta x, j\Delta y)$, we have

$$F_{ij}\sqrt{\hat{g}(u_{i-\frac{1}{2},j}, u_{i+\frac{1}{2},j}) + \hat{g}(v_{i,j-\frac{1}{2}}, v_{i,j+\frac{1}{2}})} - 1 \;=\; 0 \tag{20.14}$$

with

$$u_{i\pm\frac{1}{2},j} := U_x\big|_{i\pm\frac{1}{2},j}, \quad v_{i,j\pm\frac{1}{2}} := U_y\big|_{i,j+\frac{1}{2}}, \quad U_t = 0. \tag{20.15}$$

Note, that $U_t = 0$ holds as we solve for a stationary boundary value problem.

How might one solve equation (20.14)?
Consider a grid point (i, j) and its four neighbours $(i \pm 1, j \pm 1)$. Observe that, with (20.13), (20.14) can be written as a quadratic equation for U_{ij}, assuming the values $U_{i\pm1,j}, U_{i,j\pm1}$ are given:

$$\begin{aligned} & \max\left[\frac{U_{ij} - U_{i-1,j}}{\Delta x}, \frac{U_{i+1,j} - U_{ij}}{\Delta x}, 0\right]^2 \\ + \quad & \max\left[\frac{U_{ij} - U_{i,j-1}}{\Delta y}, \frac{U_{i,j+1} - U_{ij}}{\Delta y}, 0\right]^2 \quad = \frac{1}{F_{ij}^2}. \end{aligned} \tag{20.16}$$

Thus, one solution comes from iterating, updating the value of U at each grid point according to (20.16) until a solution is reached. This is feasible but rather slow.

The key to Fast Marching lies in the observation that the iteration above obeys a causality principle.

What is causality?
We now systematically construct the solution by upwinding, using now the variable T from (20.1). Causality means that we begin from the smallest T value, which is supposed to be known - it is part of the boundary. We follow the front propagating from these points in a thin zone around them, freezing the values we obtain with increasing T.

Note, that

- the information of arrival times is propagated "downwind",
- causality means that T can only increase.

20.1 The Algorithm

The algorithm distinguishes points as **known**, **trial** and **far**.
For initialisation, tag the boundary value points as "**known**". These are the points with the minimal arrival time. Then, tag as "**trial**" all points one grid point away. Tag as "**far**" all other grid points.

After tagging points, solve (20.16) in all "**trial**" points. For this, employ the known values of T in the points tagged as "**known**". For the values of T in the "**far**" points as well as for the "**trial**" points in the neighbourhood of point (i, j) under consideration, use $T \equiv \infty$. This principle for defining the values is always used.

The update procedure then comes along as a loop.

Step 1: Let A be the "**trial**" point with the smallest T value computed via (20.16).

Step 2: Add the point A to "**known**", remove it from "**trial**".

Step 3: Tag as "**trial**" all neighbours of A that are not "**known**" or already "**trial**". Remove corresponding points from "**far**"

Step 4: Recompute the values of T at all "**trial**" neighbours of A according to (20.16).

Step 5: Go to Step 1.

Remark: This algorithm is implemented efficiently by using a heap data structure to store the T values.

Where is the difficulty?
The difficulty is, especially, to solve (20.16). The arising quadratic expression needs a distinction of the possible cases for implementation.
A second difficulty in practice is to determine suitable points with minimal T as starting points, as well as the minimal T values themselves. This can be a very hard theoretical problem.

What is the benefit?
The algorithm works very fast, as every point is only visited once. The computational complexity is at worst $\mathcal{O}(n \log n)$ if n is the number of pixels.

Chapter 21

Multigrid

We discuss the multigrid idea at hand of a model problem.

The model problem
We consider the 1-D Poisson problem

$$-u''(x) \;=\; f(x), \qquad x \in (0,1),\ u(0)=0,\ u(1)=1. \tag{21.1}$$

We define a sequence of grid parameters

$$\{h_l\}_{l=0}^{\infty}, \;\; h_0 := \frac{1}{2}, \;\; h_l := \frac{h_0}{2^l} \tag{21.2}$$

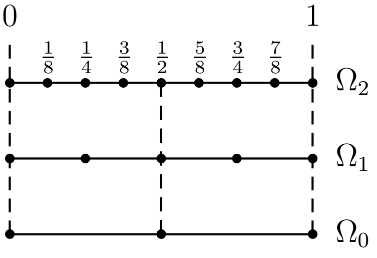
for the corresponding grid hierarchy

$$\Omega_l \;:=\; \Omega_{kl} = \{j_{k,l} \mid j = 1, \ldots, 2^{l+1}-1\} \tag{21.3}$$

for $l = 1, 2, \ldots$, where l is the grid index, see Sketch 21.
We now discretise (21.1) as usual:

$$\begin{aligned} u''(j \cdot h_l) \;&\approx:\; \frac{U_{j+1}^l - 2U_j^l + U_{j-1}^l}{h_l^2}, \\ f(j \cdot h_l) \;&\approx:\; f_j^l, \end{aligned} \tag{21.4}$$

Sketch:



where

$$j = 1, \dots, N_l := 2^{l+1} - 1. \tag{21.5}$$

Note that for our model problem, the upper index is really the grid index, not a time level. By (21.4), the linear system

$$A_l U^l \;=\; f^l \tag{21.6}$$

arises, where:

$$U^l = \begin{pmatrix} U_1^l \\ \vdots \\ U_{N_l}^l \end{pmatrix}, \quad f^l = \begin{pmatrix} f_1^l \\ \vdots \\ f_{N_l}^l \end{pmatrix}, \quad A_l = \frac{1}{h_l^2} \begin{pmatrix} 2 & -1 & & 0 \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & -1 & 2 & -1 \\ 0 & & -1 & 2 \end{pmatrix} \tag{21.7}$$

The matrix A is irreducible and diagonally dominant.

For our demonstration, we consider the Jacobi-Relaxation-method:

$$U_{m+1}^l \;=\; \underbrace{(I - \omega h_l^2 A_l)}_{=:M_l(\omega)} U_m^l + \underbrace{\omega h_l^2}_{=:N_l(\omega)} \; b. \tag{21.8}$$

One can show:

Lemma 21.0.1 *For any l, the eigenvectors $e^{l,j}$ of $M_l(\omega)$ are given by*

$$e^{l,j} \;=\; \sqrt{2h_l} \begin{pmatrix} \sin j\pi h_l \\ \vdots \\ \sin j\pi N_l h_l \end{pmatrix}, \qquad j = 1, \dots, N_l, \tag{21.9}$$

and the corresponding eigenvalues are

$$\lambda^{l,j}(\omega) \;=\; 1 - 4\omega \sin^2\left(\frac{j\pi h_l}{2}\right), \qquad j = 1, \dots, N_l. \tag{21.10}$$

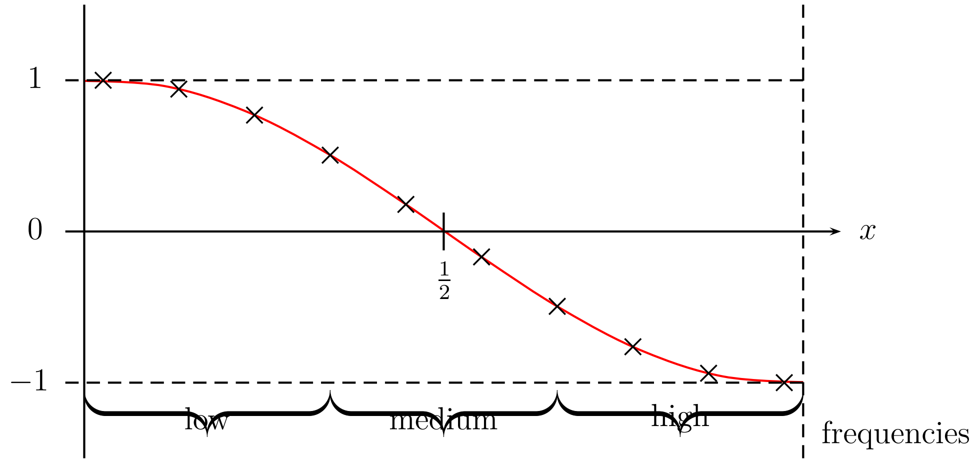
Furthermore, the vectors

$$\{e^{l,1}, \dots, e^{l,N_l}\} \tag{21.11}$$

are an orthogonal basis of the solution space $\mathbb{R}^{N_l}$. As (21.11) gives a basis, the residual between the initial vector U_0^l and the exact solution $U^{l,*} = A_l^{-1} f^l$ can be written as

$$U_0^l - U^{l,*} \;=\; \sum_{j=1}^{N_l} \alpha_j e^{l,j}, \qquad \alpha_j \in \mathbb{R}. \tag{21.12}$$

Sketch: Distribution of eigenvalues $\lambda^{l,j}$ $\left(\omega = \frac{1}{2}\right)$; $x = j \cdot h_j$ is marked by crosses.



Considering the first iteration step, we obtain

$$\begin{aligned} U_1^l - U^{l,*} &= M_l(\omega)U_0^l + N_l(\omega)f^l \\ &- (M_l(\omega)U^{l,*} + N_l(\omega)f^l) \\ &= M_l(\omega)(U_0^l - U^{l,*}) \\ &\overset{(21.12)}{=} M_l(\omega)\sum_{j=1}^{N_l}\alpha_j e^{l,j} \overset{(21.10)}{=} \sum_{j=1}^{N_l}\lambda^{l,j}(\omega)\cdot e^{l,j}. \end{aligned} \tag{21.13}$$

Repeating this process yields the general formula

$$U_m^l - U^{l,*} = \sum_{j=1}^{N_l}\alpha_j(\lambda^{l,j}(\omega))^m e^{l,j}, \qquad m = 0, 1, \ldots. \tag{21.14}$$

Remark: In multigrid, we always, and only, deal with the residual, i.e. the error function, not with the solution vector.

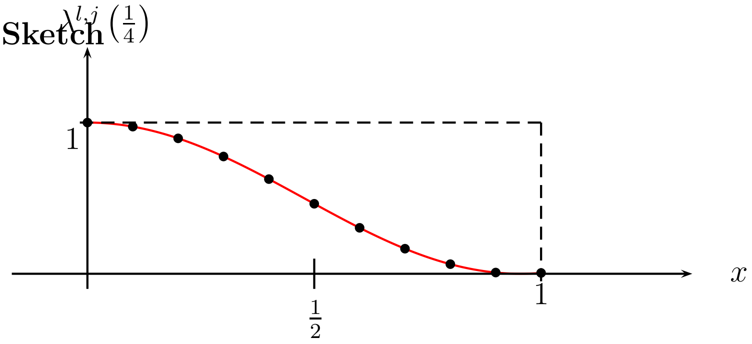
Let us have a look at Having in mind, (21.14), i.e. that the error can be expressed in powers of the eigenvalues, we gain by Sketch 21 the following insights:

(1) The error for the medium eigenvalue frequencies are strongly damped.

(2) The error for high and low frequencies are damped in a significantly better way.

(3) The finer the discretisation, the more frequencies we obtain that are only weakly damped. This is linked to the spectral radius $\rho(M_l(\omega))$ which increases, bounded by 1.

Sketch



(4) It always holds

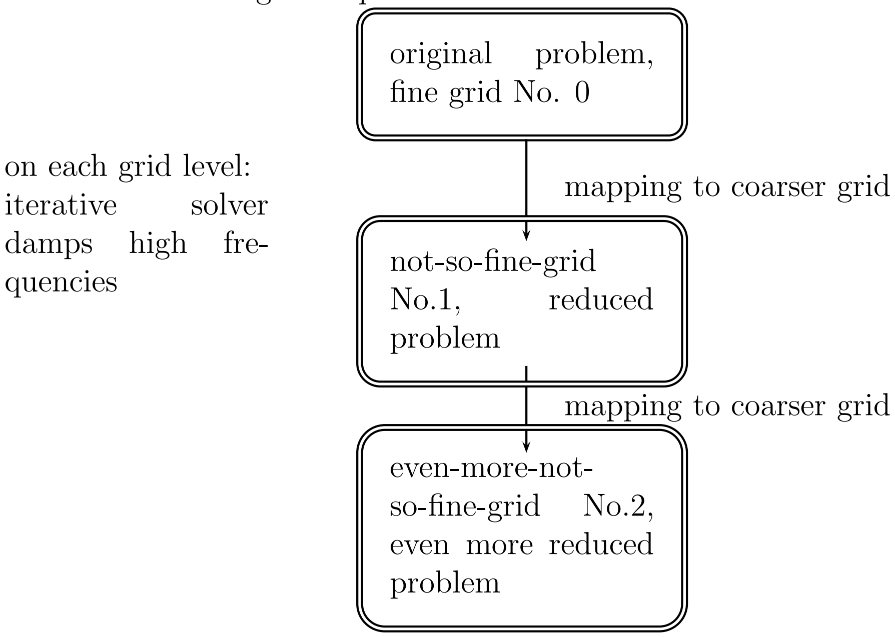
$$\lambda^{l,1}(\omega) \quad = -\lambda^{l,N_l}(\omega) \quad = \rho\left(M_l\left(\frac{1}{2}\right)\right), \tag{21.15}$$

so that the spectral radius of our method cannot be decreased by choice of ω.

In summary, we have the characteristic situation that a better approximation of u on a finer grid not only gives larger systems of equations and thus a larger computational effort per iteration, but also a drastic reduction in convergence behaviour.
The basic idea of the multigrid method is to dampen the high-frequency errors on a hierarchy of grids. To this end, it is practical to choose $\omega = \frac{1}{4}$, so that we obtain a qualitative eigenvalue distribution as in Sketch 21 After damping the high frequencies of the error, what remains are low frequencies that describe a smooth error. Consequently, an iterative scheme that does this job is called smoother. We use this knowledge of the smoothness, as for a smooth function it is reasonable to approximate it on a not-so-fine grid.

By repetition of these steps, we get the following idea of multigrid:

Sketch: Multigrid steps:

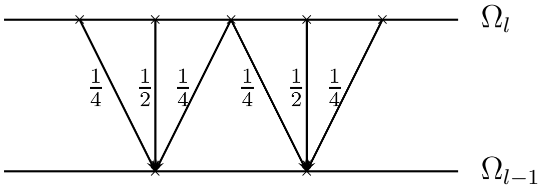

original problem, fine grid No. 0
mapping to coarser grid
not-so-fine-grid No.1, reduced problem
mapping to coarser grid
even-more-not-so-fine-grid No.2, even more reduced problem
on each grid level: iterative solver damps high frequencies

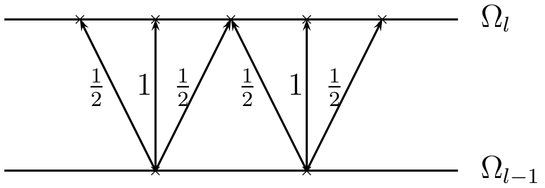
It remains to construct mappings from finer to coarser grids ("Restrictions") and interpolators from coarser to finer grids ("Prolongators"), as we wish to solve $A_l U^l = f^l$ on the finest grid.

Simple examples are as follows:
<u>Restriction</u>

$$R_l^{l-1} \;=\; \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & & 0 \\ & & 1 & 2 & 1 & & & \\ & & & \ddots & \ddots & \ddots & & \\ 0 & & & & 1 & & 2 & 1 \end{pmatrix} \in \mathbb{R}^{N_{l-1}\times N_l}, \qquad (21.16)$$

see 21.





Prolongation

$$P_{l-1}^{l} \;=\; \frac{1}{2}\begin{pmatrix} 1 & & & 0 \\ 2 & & & \\ 1 & 1 & & \\ & 2 & \ddots & \\ & 1 & \ddots & 1 \\ & & \ddots & 2 \\ 0 & & & 1 \end{pmatrix} \in \mathbb{R}^{N_l \times N_{l-1}}, \tag{21.17}$$

see 21. The computational methodology
Having after k steps of the smoother, a smooth error

$$e_k^l \;=\; U_k^l - U^{l,*}, \tag{21.18}$$

the vector e_k^l can readily be approximated on the grid Ω_{l-1}. Let us once stress again, that e_k^l is unknown, but we know it is smooth. With the defect (which we can compute!)

$$d_k^l \;:=\; A_l U_k^l - f^l \tag{21.19}$$

we get

$$\begin{aligned} A_l e_k^l &= A_l(U_k^l - U^{l,*}) \\ &= A_l U_k^l - \underbrace{A_l U^{l,*}}_{=f^l} \\ &= d_k^l. \end{aligned} \tag{21.20}$$

We get an approximation of e_k^l by solving (21.20) on the coarser grid Ω_{l-1}, prolonging the computed solution afterwards to Ω_l.
We thus consider

$$A_{l-1}e^{l-1} \;=\; d^{l-1} \tag{21.21}$$

with the restricted defect (computed with d_k^l from (21.19))

$$d^{l-1} \quad = \quad R_l^{l-1} d_k^l. \tag{21.22}$$

Having solved (21.22), we may repeat the process, or prolongate:

$$P_{l-1}^l e^{l-1} \quad = \quad P_{l-1}^l A_{l-1}^{-1} d^{l-1} \tag{21.23}$$

gives an approximation of the sought error vector e_k^l. Summarising the above steps for the two grids Ω_l, Ω_{l-1} in use gives

$$U_k^{l,\text{new}} \quad = \quad U_k^l - P_{l-1}^l A_{l-1}^{-1} R_l^{l-1} (A_l U_k^l - f^l) \tag{21.24}$$

Remarks:

- In reality, we would start on a coarse level as this is practicable.
- It pays off to consider certain sequences of Restrictions/Prolongations: V-cycles, W-cycles and combinations of them.