4 Finite Differences

An important application of the solution methods for systems of linear equations is to linear systems arising from the discretization of partial differential equations by means of finite difference, finite volumes or finite elements.

Partial differential equations frequently occur in mathematical models that arise in many branches of science, engineering, biology and economics. Unfortunately it is rare that these equations have solutions which can be expressed in closed form, so it is common to seek approximate solutions by means of numerical methods. Nowadays this can usually be achieved very inexpensively to high accuracy and with a reliable bound on the error between the analytical solution and its numerical approximation.

As an example for such a numerical method, we are going to consider finite differences, and the resulting linear systems of equations, first.

4.1 1d boundary value problem

Problem 4.1 *Given the functions $c, f \in C^0([0,1])$ and $\gamma_0, \gamma_1 \in \mathbb{R}$, find a function $u \in C^2([0,1])$ such that*

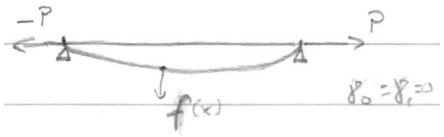
$$\begin{aligned} -u''(x) + c(x)u(x) &= f(x)\,, \qquad 0 < x < 1\,, \\ u(0) = \gamma_0\,, &\quad u(1) = \gamma_1\,. \end{aligned}$$

Problem 4.1 describes a second-order linear differential equation. It is also called a boundary value problem (BVP), because the unknown function u has to satisfy the boundary conditions $u(0) = \gamma_0$ and $u(1) = \gamma_1$.

A physical interpretation of Problem 4.1 is given as follows. The graph of the solution $u(x)$ describes the bending of a beam of length one, stretched along its axis by a force P and subject to a transverse load $f(x)$ and simply supported at its ends 0 and 1.

$$c(x) \sim \frac{P}{I(x)}\,,$$

where $I(x)$ is the principal moment of inertia of the cross-section of the beam at position x.



It can be shown that if

$$c(x) \geq 0\,, \qquad 0 < x < 1\,,$$

then there exists a unique solution to Problem 4.1. Hence we are going to assume $c(x) \geq 0$ from now on.

Example 4.2 *The unique solution to*

$$-u''(x) = -1\,, \quad u(0) = u(1) = 0\,,$$

is given by $u(x) = \frac{1}{2}x(x-1)$. Similarly, the solution to

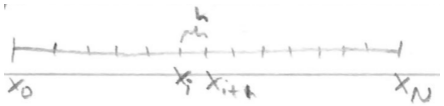
$$-u''(x) + \pi^2 u(x) = -2\pi^2 \sin(\pi x)\,, \quad u(0) = u(1) = 0\,,$$

is given by $u(x) = -\sin(\pi x)$.

In most cases, there is no known "formula" for the true solution u. We want to *approximate* the solution by using the *finite difference* method.

To this end, we introduce a uniform mesh of mesh size $h = 1/N$ over $[0,1]$.

$$x_i = ih\,, \quad 0 \le i \le N\,, \quad h = \frac{1}{N}\,.$$



In particular, $x_0 = 0$ and $x_N = 1$.

We would like to generate an approximation to the solution $u(x)$ at the grid points: Let U_i be an approximation to $u(x_i)$ for $i = 0, \ldots, N$. In order to approximate the differential equation in Problem 4.1, we derive a discrete expression for the second derivative term $-u''(x)$.

If we assume that $u \in C^4([0,1])$, then Taylor's series yields

$$u(x \pm h) = u(x) \pm hu'(x) + \tfrac{1}{2}h^2u''(x) \pm \tfrac{1}{3!}u'''(x) + \frac{1}{4!}h^4u^{iv}(x) \pm \ldots$$

Adding the two gives

$$u''(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} - \tfrac{1}{12}h^2u^{iv}(x) + \mathcal{O}(h^4)\,.$$

So if h is small, $u''(x)$ can be approximated by the *second order central difference* $\frac{u(x-h)-2u(x)+u(x+h)}{h^2}$, a so-called *finite difference*. Moreover the smaller h, the better the approximation, and in particular

$$-u''(x_i) \approx \frac{-U_{i-1} + 2U_i - U_{i+1}}{h^2}\,, \quad 1 \le i \le N-1\,.$$

Using the finite difference, we can approximate Problem 4.1 by

$$\begin{aligned} &\frac{-U_{i-1} + 2U_i - U_{i+1}}{h^2} + c_iU_i = f_i\,, \quad 1 \le i \le N-1\,, \\ &U_0 = \gamma_0\,, \quad U_N = \gamma_1\,, \end{aligned} \tag{4.1}$$

where $c_i = c(x_i)$ and $f_i = f(x_i)$ for $i = 1, \ldots, N-1$.

The linear system of equations may be written in matrix form as

$$A_hu_h = f_h\,, \tag{4.2}$$

where

$$A_h = \frac{1}{h^2}\begin{pmatrix} 2+c_1h^2 & -1 & & & & \\ -1 & 2+c_2h^2 & -1 & & & \\ & -1 & 2+c_3h^2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2+c_{N-1}h^2 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}\,,$$

and $u_h = (U_1, \ldots, U_{N-1})^T \in \mathbb{R}^{N-1}$ is the solution vector, and

$$f_h = \begin{pmatrix} f_1 + \gamma_0/h^2 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} + \gamma_1/h^2 \end{pmatrix} \in \mathbb{R}^{N-1}\,.$$

We note that A_h is a symmetric tridiagonal matrix. In the case $c(x) > 0$ it is strictly diagonally dominant, and for general $c(x) \ge 0$ it can be shown to be irreducibly diagonally

dominant. Moreover, the matrix is positive definite. To see this, and for later purposes, we first consider the case $c(x) = 0$, in which case

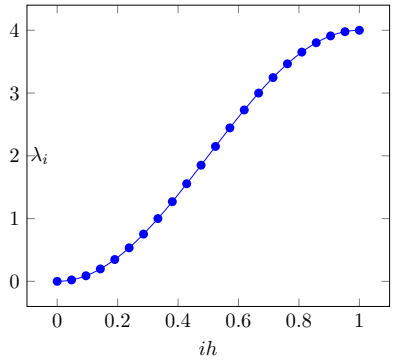
$$A_h = \frac{1}{h^2}\widetilde{T}\,, \quad \widetilde{T} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}\,.$$

It can be checked that the eigenvalues λ_i and eigenvectors w_i of $\widetilde{T}$ are given by

$$\lambda_i = 4\sin^2(\frac{i\pi}{2N}) = 4\sin^2(ih\tfrac{\pi}{2})\,, \quad \text{and} \quad [w_i]_j = \sin(\frac{ij\pi}{N}) = \sin(ijh\pi)\,, \quad 1 \le j \le N-1\,,$$

for $i = 1, \ldots, N-1$. As $\lambda_i > 0$ we see that $\widetilde{T}$ is symmetric positive definite, and then so clearly is $A_h = \frac{1}{h^2}\widetilde{T} + \mathrm{diag}(c_1, \ldots, c_{N-1})$ for $c(x) \ge 0$.

We visualize the distribution of the eigenvalues of $\widetilde{T}$ below, and note in particular that $\lambda_1 \to 0$ as $h \to 0$.



4.1.1 Solution methods

The linear system (4.2) features a tridiagonal matrix that is symmetric positive definite, so the Thomas algorithm is guaranteed to work, and is able to compute the solution in $\mathcal{O}(N)$ steps. This is optimal, and so clearly one of the best ways to find the solution u_h to (4.2).

However, in higher space dimensions the corresponding discretization will no longer yield a tridiagonal matrix, and direct methods may struggle in terms of time and memory requirements. In addition, also for applications to other, more complicated problems, it makes sense to look for alternative solution methods.

The classical iterative methods have a particularly simple form for our model problem. The Jacobi method is defined in terms of

$$U_i^{(k+1)} = [h^2 f_i + U_{i-1}^{(k)} + U_{i+1}^{(k)}]/(2 + h^2 c_i)\,, \quad i = 1\,, \ldots, N-1\,,$$

where $f_i = f(x_i)$, $c_i = c(x_i)$ as before, and where we set $U_0^{(k)} = \gamma_0$ and $U_N^{(k)} = \gamma_1$, while the Gauss–Seidel method is defined by

$$U_i^{(k+1)} = [h^2 f_i + U_{i-1}^{(k+1)} + U_{i+1}^{(k)}]/(2 + h^2 c_i)\,, \quad i = 1\,, \ldots, N-1\,.$$

The methods SOR and SSOR can be defined analogously.

What about the convergence speeds of these methods? We had seen that applied to the system $Ax = b$, the convergence rate of a classical iterative method based on the splitting $A = M - N$ critically depends on the spectral radius of the iteration matrix $C = M^{-1}N$.

Consider for simplicity the case $c(x) = 0$. Then $A_h = \frac{1}{h^2}\widetilde{T}$ and so the iteration matrix for the Jacobi method is given by

$$C_J = M^{-1}N = I - M^{-1}A_h = I - D^{-1}A_h = I - \tfrac{1}{2}\widetilde{T}\,,$$

and so the eigenvalues of C_J are $\sigma_i = 1 - 2\sin^2(ih\frac{\pi}{2})$, $i = 1, \ldots, N-1$. This yields that

$$\rho(C_J) = 1 - 2\sin^2(h\tfrac{\pi}{2}) = \cos(h\pi)\,,$$

with $\rho(C_J) \to 1$ as $h \to 0$.

On recalling (3.1), we see that our matrix A_h belongs to the class of matrices for which the spectral radius of the iteration matrix for the Gauss–Seidel method, as well as the optimal parameter ω for the SOR method can be computed. In particular, it holds that

$$\rho(C_{GS}) = \rho^2(C_J) = \cos^2(h\pi)$$

and

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2(C_J)}} = \frac{2}{1 + \sin(h\pi)}$$

with

$$\rho(C_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{1 - \sin(h\pi)}{1 + \sin(h\pi)}\,.$$

We see that each method deteriorates as h becomes small. In fact, it holds that

$$\begin{aligned}
\rho(C_J) &= \cos(h\pi) = 1 - \tfrac{1}{2}\pi^2h^2 + \mathcal{O}(h^4)\,,\\
\rho(C_{GS}) &= \cos^2(h\pi) = 1 - \pi^2h^2 + \mathcal{O}(h^4)\,,\\
\rho(C_{\omega_{opt}}) &= \frac{1 - \sin(h\pi)}{1 + \sin(h\pi)} = 1 - 2\pi h + \mathcal{O}(h^2)\,,
\end{aligned}$$

and so while the optimal SOR method converges an order of magnitude faster than the other two, all methods suffer from very slow convergence once h is getting small.

Example 4.3 *In order to reduce the size of the initial error by a factor of 1000, e.g. from 1 to* 10^{-3}*, the following number of iterations are needed for the three methods:*

$N = 1/h$	*10*	*100*
Jacobi	*138*	*13996*
Gauss–Seidel	*69*	*6998*
SOR with ω_{opt}	*17*	*195*

Similarly, we recall from (3.5) that the convergence rate of the conjugate gradient method hinges on the value of $\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}$, where in our case

$$\kappa_2(A_h) = \frac{\sin^2((1-h)\frac{\pi}{2})}{\sin^2(h\frac{\pi}{2})} = \frac{\cos^2(h\frac{\pi}{2})}{\sin^2(h\frac{\pi}{2})} = \cot^2(h\tfrac{\pi}{2}) = \frac{4}{\pi^2h^2}(1 + \mathcal{O}(h^4))\,,$$

so that

$$\frac{\sqrt{\kappa_2(A_h)} - 1}{\sqrt{\kappa_2(A_h)} + 1} = \frac{\cot(h\frac{\pi}{2}) - 1}{\cot(h\frac{\pi}{2}) + 1} = 1 - \pi h + \mathcal{O}(h^2)\,,$$

which means that the convergence is about twice as slow as for optimal SOR. However, for general matrices in practice, it may not always be possible to determine the optimal value of ω for SOR. Moreover, on using suitable preconditioners the situation for the conjugate gradient method can be improved, and so in general, the conjugate gradient algorithm is the most popular iterative solution method for symmetric positive definite matrices.

But the underlying problem remains. As h decreases, the number of iterations needed to reduce the error by a given factor will increase, meaning that it appears impossible to devise a method that produces the desired solution with complexity $\mathcal{O}(N)$. And this is where multigrid methods come in.

Note that the previous discussion regarding convergence rates of iterative methods centred on the special case $c(x) = 0$, but similar results can be expected for the general case $c(x) \geq 0$.

4.1.2 Boundary conditions

For theoretical considerations the system of linear equations making up the finite difference approximation (4.1) is usually written in the form (4.2), i.e. for $N-1$ unknowns. However, in practice it is often convenient to formally also include the boundary nodes as unknowns, and so the linear system (4.2) is replaced by

$$\widehat{A}_h \widehat{u}_h = \widehat{f}_h \,, \tag{4.3}$$

where

$$\widehat{A}_h = \begin{pmatrix} 1 & & & & \\ -\frac{1}{h^2} & \frac{2}{h^2} + c_1 & -\frac{1}{h^2} & & \\ & -\frac{1}{h^2} & \frac{2}{h^2} + c_2 & -\frac{1}{h^2} & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{h^2} & \frac{2}{h^2} + c_{N-1} & -\frac{1}{h^2} \\ & & & & 1 \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)} \,,$$

$\widehat{u}_h = (U_0, \ldots, U_N)^T \in \mathbb{R}^{N+1}$, and

$$\widehat{f}_h = \begin{pmatrix} \gamma_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \\ \gamma_1 \end{pmatrix} \in \mathbb{R}^{N+1} \,.$$

For the classical iterative methods nothing changes. As long as the initial guess satisfies the boundary conditions, and as long as the two boundary nodes are not changed by the steps of the iterative method, the algorithms will work exactly as before.

Crucially, the same holds true for the conjugate gradient method, even though the matrix $\widehat{A}_h$ is not even symmetric. The reason is that if the initial guess is chosen in the form $\widehat{u}_h^{(0)} = (\gamma_0, u_h^{(0)}, \gamma_1)^T$, then the conjugate gradient algorithm applied to the system (4.3) will always produce residuals $\widehat{r}_h^{(k)} = \widehat{f}_h - \widehat{A}_h \widehat{u}_h^{(k)}$ and iterates of the form $\widehat{r}_h^{(k)} = (0, r_h^{(k)}, 0)^T$ and $\widehat{u}_h^{(k)} = (\gamma_0, u_h^{(k)}, \gamma_1)^T$, where $r_h^{(k)}$ and $u_h^{(k)}$ are the residuals and iterates of the conjugate gradient method applied to (4.2). Hence all the theory carries across, and the method is guaranteed to converge in at most $N-1$ steps.

The same considerations apply to discretizations in higher space dimensions, and for more complicated types of boundary conditions. Here Dirichlet nodes, if part of the "unknowns",

are represented by rows from the identity matrix in the linear system (4.3), and the required boundary value appears at the corresponding location in the right hand side vector $\widehat{f}_h$.

The only situation, where some care needs to be taken, is if a direct solution method that expects a symmetric matrix is applied to the formulation (4.3). Such a method applied to (4.3) will either break down or return a wrong result, since $\widehat{A}_h$ is not symmetric. The natural fix to the problem is to "symmetrize" the linear system (4.3) by changing the columns in $\widehat{A}_h$ that correspond to Dirichlet nodes to columns from the identity matrix, and by adding the known nonzero contributions from these columns to the right hand side. For the simple 1d problem (4.3) this would result in the symmetric system

$$\begin{pmatrix} 1 & & \\ & A_h & \\ & & 1 \end{pmatrix} \widehat{u}_h = \begin{pmatrix} \gamma_0 \\ f_h \\ \gamma_1 \end{pmatrix} ,$$

which is clearly equivalent to (4.2).

4.2 2d elliptic boundary value problem

The natural extension of Problem 4.1 to 2d is given by the following elliptic partial differential equation with Dirichlet boundary conditions on the unit square.

Problem 4.4 *Let $\Omega = (0,1)^2$. Given the functions $c, f \in C^0(\overline{\Omega})$ and $g \in C^0(\partial\Omega)$, find a function $u \in C^2(\overline{\Omega})$ such that*

$$\begin{aligned} -\Delta u(x,y) + c(x,y)u(x,y) &= f(x,y) \quad \text{in } \Omega\,, \\ u(x,y) &= g(x,y) \quad \text{on } \partial\Omega\,, \end{aligned}$$

where $\Delta u = \frac{\partial^2}{\partial x^2}u + \frac{\partial^2}{\partial y^2}u$.

For simplicity, from now on we only consider the case $c(x,y) = 0$ with homogeneous boundary conditions, $g(x,y) = 0$. Then the PDE reduces to Poisson's equation on the unit square with homogeneous Dirichlet boundary conditions.

We again develop a finite difference approximation, which once again is based on a uniform mesh of mesh size $h = 1/N$ over $\overline{\Omega}$. In particular, let

$$x_i = ih\,, \quad 0 \le i \le N\,, \qquad \text{and} \qquad y_j = jh\,, \quad 0 \le j \le N\,.$$

Now combining

$$\frac{\partial^2}{\partial x^2}u(x,y) = \frac{u(x-h,y) - 2u(x,y) + u(x+h,y)}{h^2} + \mathcal{O}(h^2)$$

and

$$\frac{\partial^2}{\partial y^2}u(x,y) = \frac{u(x,y-h) - 2u(x,y) + u(x,y+h)}{h^2} + \mathcal{O}(h^2)$$

gives rise to the finite difference approximation

$$\begin{aligned} &\frac{4U_{ij} - U_{i-1,j} - U_{i+1,j} - U_{i,j-1} - U_{i,j+1}}{h^2} = f_{ij}\,, \quad 1 \le i,j \le N-1\,, \\ &U_{0j} = U_{Nj}\,, \quad 0 \le j \le N\,, \qquad U_{i0} = U_{iN}\,, \quad 0 \le i \le N\,, \end{aligned}$$

where U_{ij} approximates $u(x_i, y_j) = u(ih, jh)$ and $f_{ij} = f(x_i, y_j)$.

Once again, the system of linear equations can be written as

$$A_h\, u_h = f_h\,,$$

with $A_h \in \mathbb{R}^{(N-1)^2\times(N-1)^2}$. If the unknowns and equations are ordered in the *natural* way,

$$u_h = (U_{11}, U_{21}, \ldots, U_{N-1,1}, U_{12}, \ldots, U_{N-1,2}, \ldots, U_{1,N-1}, \ldots, U_{N-1,N-1})^T\,,$$

then the matrix can be written as

$$A_h = \frac{1}{h^2}\begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix},$$

where I is the $(N-1)\times(N-1)$ identity matrix and T is the $(N-1)\times(N-1)$ tridiagonal matrix

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}\,.$$

We note that A_h is a symmetric pentadiagonal matrix of type band$(N-1, N-1)$. Once again it can be shown that the matrix is irreducibly diagonally dominant. Moreover, the matrix is positive definite. In fact, it can be checked that the eigenvalues $\lambda_{k\ell}$ and eigenvectors $w_{k\ell}$ of $h^2 A_h$ are given by

$$\lambda_{k\ell} = 4\sin^2(kh\tfrac{\pi}{2}) + 4\sin^2(\ell h\tfrac{\pi}{2})\,, \quad \text{and} \quad [w_{k\ell}]_{ij} = \sin(ikh\pi)\sin(j\ell h\pi)\,, \quad 1 \le i, j \le N-1\,,$$

for $1 \le k, \ell \le N-1$.

4.2.1 Solution methods

In contrast to the 1d case, it is no longer possible to devise a direct solution method with linear effort in the number of unknowns, $(N-1)^2$. Using a naïve direct method would give a complexity of $\mathcal{O}(N^6)$, while utilizing the banded structure of A_h will reduce this to $\mathcal{O}(N^4)$. Using sophisticated sparse factorization methods will reduce the computational effort further, but it is not possible to introduce a direct solution method with complexity $\mathcal{O}(N^2)$.

As regards the implementation of the classical iterative solution methods, this is nearly unchanged compared to the 1d example we have considered. In terms of the convergence speeds of these methods, we observe for the Jacobi method that

$$C_J = M^{-1}N = I - M^{-1}A_h = I - D^{-1}A_h = I - \tfrac{1}{4}h^2 A_h\,,$$

and so the eigenvalues of C_J are $\sigma_{k\ell} = 1 - \sin^2(kh\frac{\pi}{2}) - \sin^2(\ell h\frac{\pi}{2})$, $k, \ell = 1, \ldots, N-1$. Hence

$$\rho(C_J) = 1 - 2\sin^2(h\tfrac{\pi}{2}) = \cos(h\pi)\,,$$

and all the results regarding the convergence rates of Jacobi, Gauss–Seidel and optimal SOR remain unchanged compared to 1d. The same holds true for the convergence rate of the conjugate gradient method.

4.3 Multigrid

We had seen that iterative methods applied to our model problems, Problem 4.1 and Problem 4.4, deteriorate when the mesh size parameter h goes to zero. The aim of multigrid methods is to overcome this deficiency and to develop methods where the convergence rate does not depend on the mesh size h.

The reason why the classical iteration methods take so many iterations to converge for small h is that certain components of the initial error $e^{(0)} = x^\star - x^{(0)}$ are only very slowly reduced.

4.3.1 Multigrid in 1d

We are going to investigate the problem in detail for the Jacobi method in 1d, but the same conclusions hold true for the other methods, and also in higher space dimensions. Recall that the error after k steps of the iterative method can be expressed as

$$e^{(k)} = M^{-1}Ne^{(k-1)} = Ce^{(k-1)} = C^k e^{(0)}\,,$$

where C is called the iteration matrix. For example, for the Jacobi method applied to our model problem with $c(x) = 0$, we had seen that $C_J = I - D^{-1}A_h = I - \frac{1}{2}\widetilde{T}$, with its eigenvalues σ_i and eigenvectors w_i given by

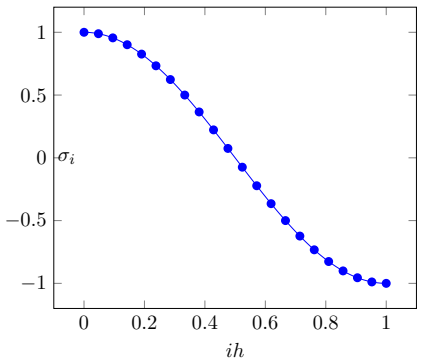
$$\sigma_i = 1 - 2\sin^2(ih\tfrac{\pi}{2}) \quad \text{and} \quad [w_i]_j = \sin(ijh\pi)\,, \quad 1 \leq j \leq N-1\,,$$

for $i = 1, \ldots, N-1$.

Hence, if we consider the basis $\{w_i\}_{i=1}^{N-1}$ of $\mathbb{R}^{N-1}$ and represent the initial error as $e^{(0)} = \sum_{i=1}^{N-1} \alpha_i w_i$, then we know that the error after k steps of the Jacobi method is going to satisfy

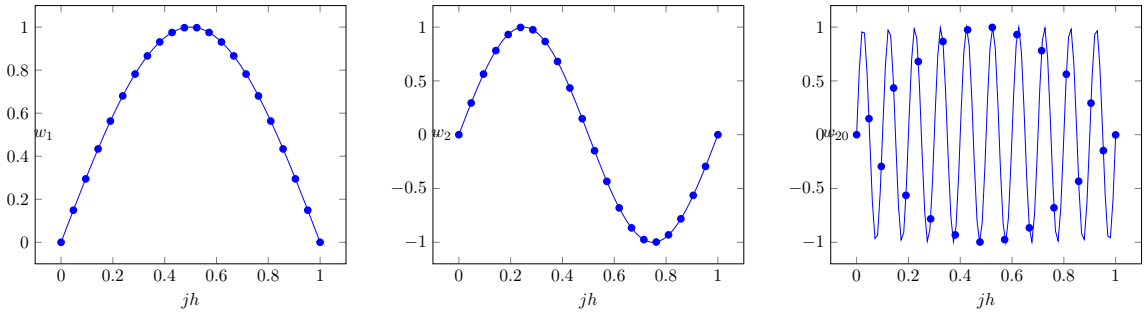
$$e^{(k)} = C^k e^{(0)} = C^k \sum_{i=1}^{N-1} \alpha_i w_i = \sum_{i=1}^{N-1} \alpha_i C^k w_i = \sum_{i=1}^{N-1} \alpha_i \sigma_i^k w_i\,.$$

In particular, the error component w_i is going to be reduced by a factor $|\sigma_i|^k$. Looking at the distribution of eigenvalues:



we see that the components w_i with i close to 1 or close to $N-1$ are going to be decreased very slowly, because $|\sigma_i|$ for these eigenvectors is very close to 1.

The eigenvectors w_i are often called "modes" and they can be visualized as approximations for functions of the form $\widehat{w}_i(x) = \sin(ix\pi)$. We plot some eigenvectors in that form below.



Hence the mode w_i corresponds to a function $\widehat{w}_i(x)$ with i half sine waves on the interval $[0,1]$. For small i these modes are called smooth, whereas for i close to $N-1$ the modes are highly oscillatory. In that language we can say that for the standard Jacobi iteration the smooth and highly oscillatory modes of the error are damped very slowly, whereas the other contributions of the error are reduced quickly.

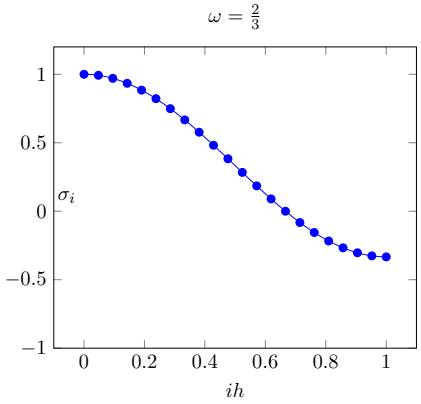
For the Jacobi method the situation can be improved by considering a weighted Jacobi iteration, similarly to weighted Gauss–Seidel. In particular, on choosing $M = \frac{1}{\omega} D$ in the splitting $A_h = M - N$, we obtain the iteration matrix

$$C_{J_\omega} = I - (\tfrac{1}{\omega} D)^{-1} A_h = I - \tfrac{1}{2}\omega \widetilde{T}$$

with the eigenvalues

$$\sigma_i = 1 - 2\omega \sin^2(ih\tfrac{\pi}{2}), \quad i = 1, \ldots, N-1,$$

and the eigenvectors as before. It can be seen that the weighted Jacobi method converges for all $\omega \in (0,1]$. Moreover, on letting $\omega = \frac{2}{3}$ we can achieve that $\sigma_i \geq -\frac{1}{3}$ for all $i = 1, \ldots, N-1$, and the overall distribution of eigenvalues now looks as follows.



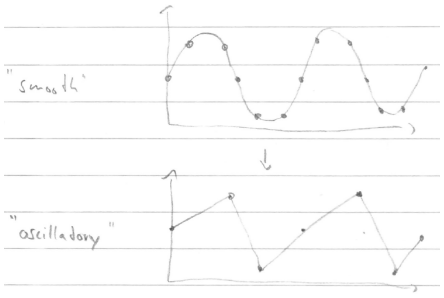
So for the weighted Jacobi method we can say that the smooth components of the error are dampened very slowly, while the oscillatory parts are dampened quickly. The same is true for the Gauss–Seidel method, although the analysis there is more complicated.

In fact, the slow reduction of the smooth components of the error is the main reason for the slow convergence of all classical iterative methods.

A possible way out is to consider the model problem not only on the given grid $\Omega_h = \{ih : i = 0, \ldots, N\}$, but also on a *coarse grid* Ω_H, e.g. for $H = 2h$.

Advantages:

- Iteration step on Ω_H is cheap. (Half the number of points if $H = 2h$.)
- $\rho(C_H) < \rho(C_h)$.
- A smooth error on Ω_h may appear more oscillatory on the grid Ω_H.



The idea is to use the information from the coarse grid for a correction step on the fine grid. To motivate the underlying idea of the correction step, assume that for the linear systems $Ax = b$ we have obtained some iterate $x^{(k)}$, with residual $r^{(k)} = b - Ax^{(k)}$. Now, if we could solve the residual equation

$$Ae = r^{(k)}$$

exactly, then $x^\star = x^{(k)} + e$ is the true solution to $Ax = b$, since $A(x^{(k)} + e) = Ax^{(k)} + Ae = b$. Of course, in practice we cannot solve the residual equation exactly, but we could obtain an approximation to e on the coarse grid, which is cheap, and then update $x^{(k)}$ on the fine grid with the obtained correction.

Given two grids Ω_h and Ω_H (e.g. $H = 2h$), the two grid correction method applied to our model problem proceeds as follows, where from now on we use the terminology "relax" to indicate a classical iterative method such as (weighted) Jacobi, Gauss–Seidel or SOR, which are often also referred to as relaxation methods.

1. Relax $Au = f$ on Ω_h to obtain an approximation $u_h^{(k)}$.
2. Compute $r = f - Au_h^{(k)}$.
3. Solve $Ae = r$ on Ω_H to obtain an approximation e_H to the error e.
4. Correct the approximation $u_h^{(k+1)} = u_h^{(k)} + e_H$.

Clearly, we need some mapping from Ω_h to Ω_H, and vice versa, to make sense of all the steps in the algorithm. From now on we fix $H = 2h$, and let $h = 1/N$ for an even number N.

For the final step, we need to be able to make sense of the error e_{2h} on the grid Ω_h. To this end, we define the *prolongation* operator

$$I_{2h}^h : \Omega_{2h} \to \Omega_h$$

as follows. Let v_h and v_{2h} be defined on Ω_h and Ω_{2h}, respectively. Then $I_{2h}^h v_{2h} = v_h$, where

$$[v_h]_{2i} = [v_{2h}]_i\,, \quad i = 0, \ldots, \tfrac{N}{2}\,, \qquad [v_h]_{2i+1} = \tfrac{1}{2}([v_{2h}]_i + [v_{2h}]_{i+1})\,, \quad i = 0, \ldots, \tfrac{N}{2} - 1\,.$$

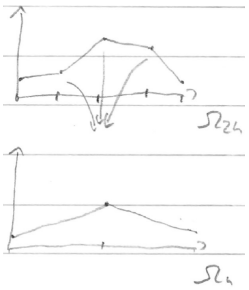
That means that the prolongation does not changes the values at points that are part of both grids, and values on the fine grid that are not part of the coarse grid are defined as the average of their neighbouring nodes. This is called linear interpolation.

In addition, in order to make sense of the residual on the coarse grid, we need to define a *restriction* operator

$$I_h^{2h} : \Omega_h \to \Omega_{2h}\,.$$

The simplest possible definition would just be an injection, where the values at the points on the coarse grid are directly inherited from the same points on the fine grid. However, in practice a *full weighting* restriction often works best as follows. Let v_h and v_{2h} be defined on Ω_h and Ω_{2h}, respectively. Then $I_h^{2h} v_h = v_{2h}$, where

$$[v_{2h}]_i = \tfrac{1}{4}([v_h]_{2i-1} + 2[v_h]_{2i} + [v_h]_{2i+1}), \quad i = 1, \ldots, \tfrac{N}{2} - 1.$$



Note that we have not specified the two boundary nodes, as the restriction operator will always be applied to residuals, which by definition will be zero at the boundary. In fact, ignoring the boundary nodes the operators can be interpreted as linear maps $I_h^{2h} : \mathbb{R}^{N-1} \to \mathbb{R}^{\frac{N}{2}-1}$ and $I_{2h}^h : \mathbb{R}^{\frac{N}{2}-1} \to \mathbb{R}^{N-1}$ that map constant vectors to the corresponding constant vectors and satisfy

$$I_{2h}^h I_h^{2h} = I \quad \text{and} \quad I_{2h}^h = \beta (I_h^{2h})^T, \tag{4.4}$$

with $\beta = 2$.

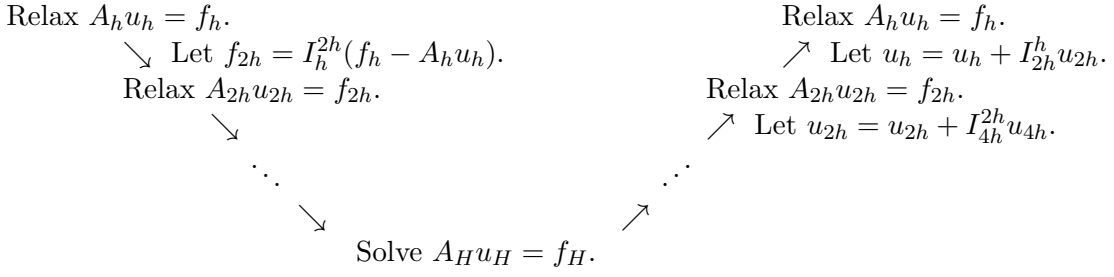
We can now make the two grid correction scheme more precise.

1. Relax $A_h u_h = f_h$ on Ω_h to obtain $u_h^{(k)}$. *(pre-smoothing)*
2. Compute $r_h = f_h - A_h u_h^{(k)}$.
3. Compute $r_{2h} = I_h^{2h} r_h$. *(restriction of residual)*
4. Solve $A_{2h} e_{2h} = r_{2h}$ on Ω_{2h}.
5. Correct fine grid solution $u_h^{(k+1)} = u_h^{(k)} + I_{2h}^h e_{2h}$. *(coarse grid correction)*
6. Relax $A_h u_h = f_h$ on Ω_h with initial guess $u_h^{(k+1)}$. *(post-smoothing)*

Here the matrix A_{2h} simply refers to the matrix obtained from a finite difference approximation of our model problem on the grid Ω_{2h}. We note that an alternative approach, which we do not pursue further, is to define $A_{2h} = I_h^{2h} A_h I_{2h}^h$ instead.

If we believe that the two grid correction method works well for the iterative solution of $Au_h = f_h$, then there is no reason not to apply the same idea to the residual equation $A_{2h} e_{2h} = r_{2h}$ on Ω_{2h} as well. If we continue this approach recursively, we arrive at the multigrid method.

In particular, we introduce a sequence of grids Ω_h, Ω_{2h}, Ω_{4h}, $\ldots$, Ω_H. Then, on renaming the residuals $r_{\ell h}$ on the coarser as $f_{\ell h}$ (they are just another right hand side vector), and on renaming the error corrections $e_{\ell h}$ on the coarser grids as $u_{\ell h}$ (they are just another solution vector), the so-called V-cycle for the multigrid method can be described as follows.



This cycle gets its name from the fact that the traversal of the sequence of grids can be visualized in the shape of a letter "V".

Algorithm 4.1: $u_h \leftarrow MGV^h(u_h, f_h)$

if $\Omega_h = \Omega_H$ **then** $u_h \leftarrow A_h^{-1} f_h$;
else
  Relax $A_h u_h = f_h$, α_1 times;
  $f_{2h} \leftarrow I_h^{2h}(f_h - A_h u_h)$;
  $u_{2h} \leftarrow 0$;
  $u_{2h} \leftarrow MGV^{2h}(u_{2h}, f_{2h})$;
  Correct $u_h \leftarrow u_h + I_{2h}^h u_{2h}$;
  Relax $A_h u_h = f_h$, α_2 times.
end

The number of pre- and post-smoothing steps is usually set to one: $\alpha_1 = \alpha_2 = 1$. Note that if $\Omega_H = \{0, \frac{1}{2}, 1\}$ is the trivial grid containing just a single inner point, then A_H is a scalar and the inversion of A_H^{-1} corresponds to a trivial division. If Ω_H is not the trivial grid, then the exact solve on the grid Ω_H is often simply replaced by a relaxation step.

What are the computational costs of a V-cycle compared to a relaxation step on the finest grid? We need to store the solution vectors $u_{\ell h}$ and $f_{\ell h}$ on all the grids, so the total storage requirement is twice that of the fine grid. Similarly, for $\alpha_1 = \alpha_2 = 1$ the total operations count is about four times that of a relaxation step on the finest grid, and in any case of order $\mathcal{O}(N)$.

Moreover, it can be shown that the convergence rate of the multigrid iterative solver does not depend on the mesh size h. In particular, in practice the number of iterations needed to reduce the error by a fixed factor does not deteriorate with h.

Example 4.5 *For Problem 4.1 consider the case $\gamma_0 = \gamma_1 = 0$ and $f = 0$, so that the true solution is $u = 0$. Let the initial guess be defined by $u_h^{(0)} = \frac{1}{10}\sin(4\pi x) + \frac{1}{2}\sin(15\pi x)$. Then in order to reduce the size of the initial error by a factor of 10^4, the following number of iterations are needed for the Gauss–Seidel, optimal SOR and V-cycle multigrid with 1 Gauss–Seidel pre- and post-smoothing step:*

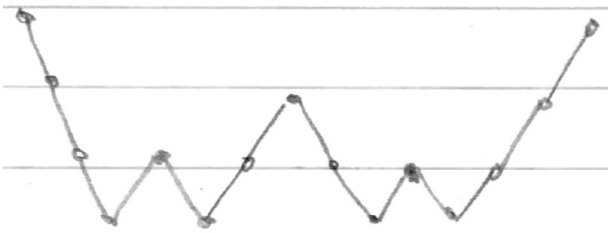
$N = 1/h$	64	128	256	512	1024
Gauss–Seidel	1408	4438	13228	34766	77377
SOR with ω_{opt}	121	242	483	965	1929
V-cycle	5	5	5	4	4

The V-cycle is just one of a family of multigrid cycling schemes. The whole family is called the μ-cycle method and is defined recursively by the following algorithm.

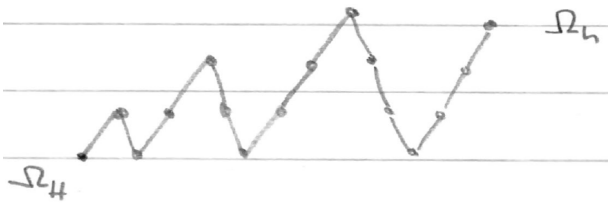
Algorithm 4.2: $u_h \leftarrow MG\mu^h(u_h, f_h)$

if $\Omega_h = \Omega_H$ **then** $u_h \leftarrow A_h^{-1} f_h$;
else
  Relax $A_h u_h = f_h$, α_1 times;
  $f_{2h} \leftarrow I_h^{2h}(f_h - A_h u_h)$;
  $u_{2h} \leftarrow 0$;
  $u_{2h} \leftarrow MG\mu^{2h}(u_{2h}, f_{2h})$, μ times;
  Correct $u_h \leftarrow u_h + I_{2h}^h u_{2h}$;
  Relax $A_h u_h = f_h$, α_2 times.
end

In practice, really only $\mu = 1$, i.e. the V-cycle, and $\mu = 2$ are used. Because of the way the levels are traversed in the case $\mu = 2$, that cycling method is called the W-cycle multigrid algorithm.



A final refinement to the multigrid idea comes in the form of nested iterations. That is, rather than starting with an arbitrary initial guess on the finest level Ω_h, we could try to prolongate a (converged) solution on the coarser grid Ω_{2h} to Ω_h, and then start the V-cycle, for example, with that improved initial guess. Applying this idea recursively gives rise to a nested iteration in order to produce a good initial guess on the finest grid. Note that in general it is not needed to solve the problems on the coarser grids to convergence, and so often just a single V-cycle is performed on each coarse grid.



The obtained algorithm is then called full multigrid. In order to increase the flexibility of the algorithm, we can replace the V-cycle with a more general μ-cycle, and we can also replace a single such step with several steps. Overall the full multigrid method is then given by the following algorithm.

Algorithm 4.3: $u_h \leftarrow FMG^h(f_h)$

if $\Omega_h = \Omega_H$ **then** $u_h \leftarrow A_h^{-1} f_h$;
else
  $f_{2h} \leftarrow I_h^{2h} f_h$;
  $u_{2h} \leftarrow FMG^{2h}(f_{2h})$;
  $u_h \leftarrow I_{2h}^h u_{2h}$;
  $u_h \leftarrow MG\mu^h(u_h, f_h)$, ν times;
end

If the solution obtained after a full multigrid step does not yet satisfy the desired stopping criterion, further μ-cycles could be applied until convergence. However, the holy grail of the multigrid method is to design the available parameters (α_1, α_2, μ, ν) such that after the full multigrid step the stopping criterion is already satisfied. The reasoning is that investing more work on the coarser grids will always be cheaper than starting a full V-cycle iteration on the finest grid, say. In practice the optimal choices for the number of pre- and post-smoothing steps α_1 and α_2, the type of μ-cycle and the number of μ-cycle steps ν are highly problem dependent. They can often be found experimentally by running test problems on coarser grids.

4.3.2 Multigrid in 2d

All the ingredients of multigrid for the model problem in 2d remain unchanged. In particular, the definitions of the coarse grid Ω_{2h} and the operator A_{2h} are obvious. Moreover, the prolongation operator $I_{2h}^h : \Omega_{2h} \to \Omega_h$ can again be defined by a natural interpolation, while the full weighting restriction $I_h^{2h} : \Omega_h \to \Omega_{2h}$ is uniquely defined by requiring that (4.4) holds. Specifically, we can define $I_{2h}^h v_{2h} = v_h$ via

$$\begin{aligned}
[v_h]_{2i,2j} &= [v_{2h}]_{ij}\,, \quad i,j = 0,\dots,\tfrac{N}{2}\,,\\
[v_h]_{2i+1,2j} &= \tfrac{1}{2}([v_{2h}]_{ij} + [v_{2h}]_{i+1,j})\,, \quad i = 0,\dots,\tfrac{N}{2}-1,\ j = 0,\dots,\tfrac{N}{2}\,,\\
[v_h]_{2i,2j+1} &= \tfrac{1}{2}([v_{2h}]_{ij} + [v_{2h}]_{i,j+1})\,, \quad i = 0,\dots,\tfrac{N}{2},\ j = 0,\dots,\tfrac{N}{2}-1\,,\\
[v_h]_{2i+1,2j+1} &= \tfrac{1}{4}([v_{2h}]_{ij} + [v_{2h}]_{i+1,j} + [v_{2h}]_{i,j+1} + [v_{2h}]_{i+1,j+1})\,, \quad i,j = 0,\dots,\tfrac{N}{2}-1\,,
\end{aligned}$$

and conversely $I_h^{2h} v_h = v_{2h}$ via

$$\begin{aligned}
[v_{2h}]_{ij} = \tfrac{1}{16}([v_h]_{2i-1,2j-1} + [v_h]_{2i-1,2j+1} + [v_h]_{2i+1,2j-1} + [v_h]_{2i+1,2j+1}\\
+ 2([v_h]_{2i,2j-1} + [v_h]_{2i,2j+1} + [v_h]_{2i-1,2j} + [v_h]_{2i+1,2j})\\
+ 4[v_h]_{2i,2j})\,, \quad i,j = 1,\dots,\tfrac{N}{2}-1\,,
\end{aligned}$$

so that (4.4) now holds with $\beta = 4$.

Remark 4.6 *A careful, but highly nontrivial, analysis for our model problems shows that the convergence rate of a V-cycle, and a W-cycle, is less than 1 and bounded away from 1. In particular, introducing the iteration matrix C_V for a V-cycle in can be shown that $\rho(C_V) \leq \varrho$, where $\varrho \approx \frac{1}{10}$, independently of h. Similar results can be shown for a wide class of elliptic partial differential equations. Hence the number of iterations to reduce the error by a fixed factor does not depend on h, and so the overall complexity of finding a solution to $A_h u_h = f_h$ is $\mathcal{O}(n)$, where $n = (N-1)^d$ is the number of unknowns, with $d = 1, 2, 3$ denoting the spatial dimension of the PDE.*

In addition, it can be shown that full multigrid maintains this optimal complexity even when the desired approximate solution to $A_h u_h = f_h$ should have an error that is smaller than the discretization error of the PDE, which for the applied finite difference scheme is $\mathcal{O}(h^2)$. In short, the multigrid method offers optimal complexity to solve $A_h u_h = f_h$, with unchallenged efficiency in 2d and 3d.

A disadvantage of multigrid is that it is heavily problem specific, and for more difficult problems it is not always straightforward to find appropriate smoothing, prolongation and restriction operators. For example, for unstructured meshes a hierarchy of grids needs to be available.

The form of multigrid method that we have considered so far is called geometric multigrid method*, as it makes use of knowledge of the geometry of the problem and of the grids introduced to approximate the solution of the PDE. An alternative procedure is* algrebraic multigrid *applied to the problem $Ax = b$, where no information about the underlying geometry or meshes is known. Here a hierarchy of levels is constructed purely based on the information found in the entries of A.*

Finally, an important application of multigrid methods is in the role of preconditioners for other iterative solution methods, for example conjugate gradient methods. This holds true in particular in higher space dimensions, and in situations where the linear operator is close or similar to a matrix A_h on which multigrid is known to work well.

4.4 Heat equation

As an example for a time dependent problem, we consider the following linear parabolic equation, at first in one space dimension.

4.4.1 Heat equation in 1d

Problem 4.7 *Given the function $u_0 \in C^0([0,1])$ and a final time $T > 0$, find a function $u \in C^2([0,1] \times [0,T])$ such that*

$$\frac{\partial}{\partial t} u(x,t) - \kappa \frac{\partial^2}{\partial x^2} u(x,t) = 0\,, \qquad 0 < x < 1\,, \quad 0 < t \leq T\,, \tag{4.5a}$$

$$u(0,t) = u(1,t) = 0\,, \qquad 0 < t \leq T\,, \tag{4.5b}$$

$$u(x,0) = u_0(x)\,, \qquad 0 \leq x \leq 1\,. \tag{4.5c}$$

The partial differential equation (4.5a) in Problem 4.7 is called the *heat equation*. In fact, $u(x,t)$ models the temperature at a point x and a time t of, for example, a metallic bar of unit length that occupies the interval $[0,1]$. Here $\kappa > 0$ represents the thermal conductivity of the material, which is assumed to be constant. The two end-points of the bar are kept at a constant temperature, which we have set to zero for simplicity. In addition, u_0 denotes an initial temperature profile.

It can be shown that the solution $u(x,t)$ to the heat equation decays over time, and that the steady state solution as $t \to \infty$ is $u = 0$. To see this, let us multiply (4.5a) with u and integrate over $[0,1]$ to obtain

$$\begin{aligned} 0 &= \int_0^1 \frac{\partial}{\partial t} u(x,t) u(x,t)\,\mathrm{d}x - \kappa \int_0^1 \frac{\partial^2}{\partial x^2} u(x,t) u(x,t)\,\mathrm{d}x \\ &= \tfrac{1}{2} \int_0^1 \frac{\partial}{\partial t} u^2(x,t)\,\mathrm{d}x + \kappa \int_0^1 \left(\frac{\partial}{\partial x} u(x,t) \right)^2 \mathrm{d}x - \kappa \left[\frac{\partial}{\partial x} u(x,t) u(x,t) \right]_{x=0}^{x=1} \\ &= \tfrac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \int_0^1 u^2(x,t)\,\mathrm{d}x + \kappa \int_0^1 \left(\frac{\partial}{\partial x} u(x,t) \right)^2 \mathrm{d}x\,, \end{aligned}$$