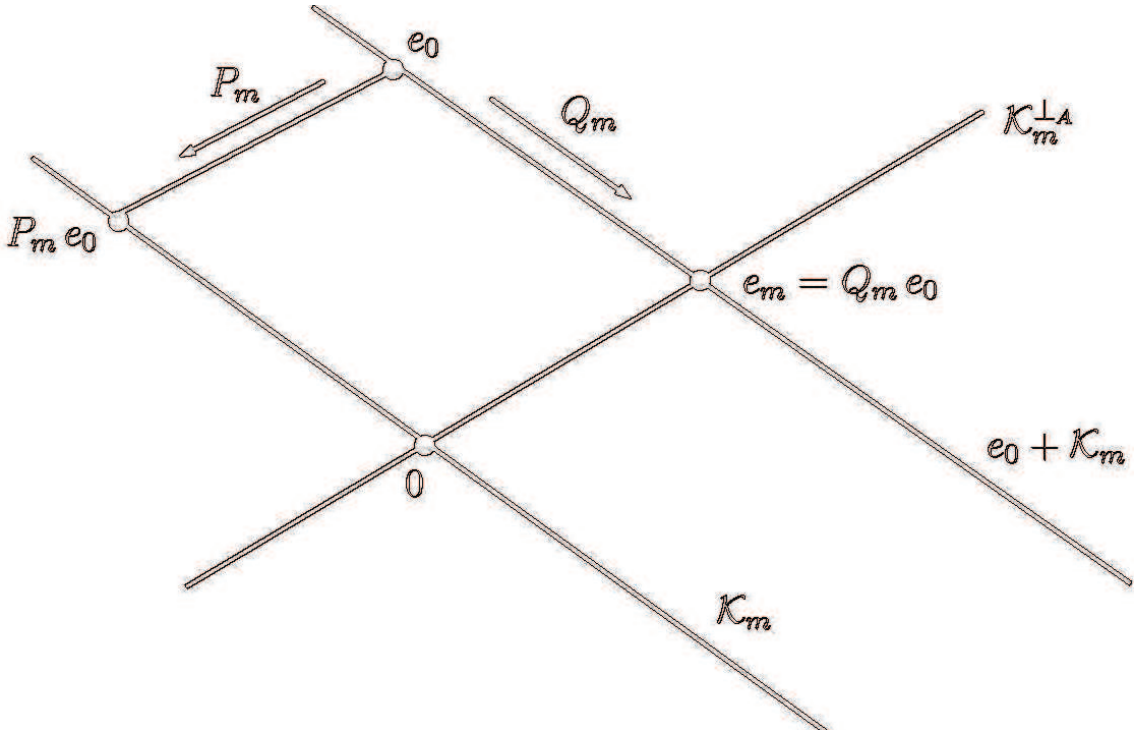
Iterative Solution of Large Linear Systems

W. Auzinger
J. M. Melenk



Wien, 2017

Contents

1 Introduction; Some Basic Facts from Linear Algebra

Numerical models are growing in size and complexity with increasing computational power being available. Many of these models ultimately reduce to solving a system of linear equations

$$A\,x = b, \quad A \in \mathbb{R}^{n\times n} \text{ or } \mathbb{C}^{n\times n}$$

of large dimension n[1] e.g., $> 10^7$. In this course we present a few of the methods devised that can be efficiently applied to problems of this size.

Fist we recall of some of the most important definitions and results from linear algebra. For further reading on these basic topics, the texts [2], [15], [22], and [23] can be recommended. The books [13] and [19] are more specific to the topic of this lecture.

We are mainly concerned with real systems, i.e., real mappings (matrices) $A \in \mathbb{R}^{n\times n}$. Most techniques can be applied to complex problems as well, but these are less 'prominent' in practice.

Further material from linear algebra is provided in later chapters, whenever necessary.

1.1 Vector norms and inner products

1. The Euclidean inner product of (column) vectors $x, y \in \mathbb{R}^n$ is denoted in two alternative ways:

$$(x, y) \equiv x^{\mathrm{T}} y = \sum_{i=1}^{n} x_i\, y_i$$

The Hilbert-space denotation (x, y) is usually appropriate for theoretical considerations. The denotation $x^{\mathrm{T}}y$ is algorithmically oriented: The inner product is a special case of matrix multiplication ('row vector · column vector').

The norm induced by the Euclidean inner product is the l^2-norm,

$$\|x\|_2^2 = (x, x) = x^{\mathrm{T}} x = \sum_{i=1}^{n} x_i^2$$

which belongs to the family of l^p norms defined via

$$\|x\|_p = \begin{cases} \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} & \text{if } \; p \in [1, \infty) \\ \max\limits_{i=1\ldots n} |x_i| & \text{if } \; p = \infty \end{cases}$$

2. The more general *M-inner product*

$$(x, y)_M = (Mx, y)_2 = x^{\mathrm{T}} M\, y$$

will also play a prominent role. Here we assume that M is a symmetric positive-definite (SPD) matrix.[2] We note that, by symmetry of M,

$$(x, y)_M = (x, My) = (Mx, y) = (y, x)_M$$

and $(x, x)_M$ is definite, i.e., $(x, x)_M > 0$ for $x \neq 0$.

[1] Facetious definition of 'large': the problem, by applying most tricks, barely fits into memory.

[2] $M \in \mathbb{R}^{n\times n}$ is symmetric if $M^{\mathrm{T}} = M$; it is positive semi-definite if $x^{\mathrm{T}} M\, x \geq 0$ for all $x \in \mathbb{R}^n$; M is positive definite if $x^{\mathrm{T}} M\, x > 0$ for all $x \neq 0$.

The M-inner product induces the M-norm

$$\|x\|_M^2 = (x,x)_M$$

Since in many applications the quantity $\frac{1}{2}x^{\mathrm{T}}Mx$ represents an energy, the M-norm $\|\cdot\|_M$ is often called the *energy norm*.

3. norms on a vector space V over $\mathbb{R}$, $x, y \in V$ satisfy by definition for all $x, y \in V$

   a) $\|\alpha x\| = |\alpha|\,\|x\| \quad \forall\ \alpha \in \mathbb{R}$ (homogeneity)

   b) $\|x\| \geq 0$, and $\|x\| = 0 \ \Leftrightarrow\ x = 0$ (definiteness)

   c) $\|x+y\| \leq \|x\| + \|y\|$ (triangle inequality)

   We recall that for arbitrary *finite-dimensional* vector spaces V any two norms $\|\cdot\|_\alpha$, $\|\cdot\|_\beta$ on V are equivalent in the sense that there exist constants $c, C > 0$ such that

$$c\|x\|_\alpha \leq \|x\|_\beta \leq C\|x\|_\alpha \quad \text{for all } x \in V$$

1.2 Matrix spectra and and matrix norms

1. Let $A \in \mathbb{R}^{n\times n}$. A scalar $\lambda \in \mathbb{C}$ is called an *eigenvalue* of A if there exists an *eigenvector* $v \in \mathbb{C}^n \setminus \{0\}$ such that $Av = \lambda v$. The set

$$\sigma(A) = \{\lambda \in \mathbb{C} :\ \lambda \text{ is an eigenvalue of } A\}$$

   is called the *spectrum* of A. If A is *symmetric*, then the eigenvalues of A are real, and the extremal eigenvalues satisfy

$$\lambda_{min} = \min_{x\neq 0} \frac{(Ax,x)}{(x,x)}, \qquad \lambda_{max} = \max_{x\neq 0} \frac{(Ax,x)}{(x,x)}$$

2. Each norm on $\mathbb{R}^n$ induces a norm $\|\cdot\|\colon \mathbb{R}^{n\times n} \to \mathbb{R}$ on the vector space of $n\times n$ matrices via

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

   These induced *matrix norms* are *submultiplicative*,

$$\|AB\| \leq \|A\|\,\|B\| \quad \forall\ A, B \in \mathbb{R}^{n\times n}$$

   The case of the Euclidean $\|\cdot\|_2$-norm is particularly prominent. We have

$$\|A^{\mathrm{T}}\|_2 = \|A\|_2$$

$$\|A\|_2^2 \leq n \max_j \sum_{i=1}^n |A_{i,j}|^2 \quad \text{and} \quad \|A\|_2^2 = \|A^{\mathrm{T}}\|_2^2 \leq n \max_i \sum_{j=1}^n |A_{i,j}|^2$$

3. The *spectral radius* $\rho(A)$ of $A \in \mathbb{R}^{n\times n}$ is defined by

$$\rho(A) = \max(\sigma(A)) = \max\{|\lambda| :\ \lambda \in \mathbb{C} \text{ is an eigenvalue of } A\}$$

   If A is symmetric or, more generally, normal, then

$$\|A\|_2 = \rho(A)$$

4. The quantity
$$\kappa_p(A) = \begin{cases} \|A\|_p \|A^{-1}\|_p & \text{if A is invertible} \\ \infty & \text{if A is singular} \end{cases}$$
is called the l^p-*condition number* of A. In particular, for symmetric positive definite matrices we have
$$\kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}} \tag{1.1a}$$

5. For general (nonsymmetric) matrices with positive eigenvalues, the analogous quantity
$$\kappa_\sigma(A) = \frac{\lambda_{max}}{\lambda_{min}} \tag{1.1b}$$
is called the *spectral condition number* of A. Note that in general this does not equal $\kappa_2(A)$ (except for SPD matrices).

The spectral radius $\rho(A)$ of a matrix A will be important for the analysis of many iterative methods. We have (see, e.g., [10, Sec. 2.9]):

Theorem 1.1 *Let $A \in \mathbb{R}^{n\times n}$ or $A \in \mathbb{C}^{n\times n}$. Then:*

(i) $\rho(A^m) = \rho(A)^m$ *for all* $m \in \mathbb{N}$.

(ii) For any norm $\|\cdot\|$ *on* $\mathbb{R}^n$. *(or on* $\mathbb{C}^n$ *if* $A \in \mathbb{C}^{n\times n}$*) we have* $\rho(A) \le \|A\|$.

(iii) For every $\varepsilon > 0$ *there exists a norm* $\|\cdot\|_\varepsilon$ *on the vector space* $\mathbb{R}^n$ *(or* $\mathbb{C}^n$*) such that*
$$\rho(A) \le \|A\|_\varepsilon \le \rho(A) + \varepsilon$$

(iv) For any norm $\|\cdot\|$ *on* $\mathbb{R}^n$ *(or on* $\mathbb{C}^n$*) we have*
$$\rho(A) = \lim_{m\to\infty} \|A^m\|^{1/m}$$

Exercise 1.1

a) Let $A \in \mathbb{R}^{n\times n}$ be normal. Then one can find a norm $\|\cdot\|$ on $\mathbb{R}^n$ such that $\rho(A) = \|A\|$. Comment on the special cases of symmetric and skew-symmetric A.

b) Show that $\|A\|_2 = \sqrt{\rho(A^{\mathrm{T}}A)}$.

c) Show that the Hölder-type inequality $\|A\|_2^2 \le \|A\|_1 \|A\|_\infty$ is valid.

d) Give an example of a matrix $A \neq 0$ such that $\rho(A) = 0$. ∎

1.3 Types of matrices and some matrix decompositions (factorizations)

1. ***Orthogonal matrices:*** A matrix A is orthogonal[3] if $AA^{\mathrm{T}} = A^{\mathrm{T}}A = I$. This means that the columns of A are orthonormal to each other (the same is true for the rows of A). Note that this implies $A^{\mathrm{T}} = A^{-1}$. An important further property of an orthogonal matrix A is that $\|Ax\|_2 = \|x\|_2$ for all $x \in \mathbb{R}^n$ i.e., an orthogonal matrix represents an *isometric* mapping.

   The complex analog is a *unitary* matrix $A \in \mathbb{C}^{n\times n}$, characterized by $AA^H = A^HA = I$. (Note $A^H = \bar{A}^{\mathrm{T}}$.)

2. ***QR-decomposition:*** Let $m \geq n$. Every matrix $A \in \mathbb{R}^{m\times n}$ can be written as $A = QR$, where $Q \in \mathbb{R}^{m\times n}$ has orthonormal columns and $R \in \mathbb{R}^{n\times n}$ is upper triangular. Actually, this is the 'reduced' QR-decomposition, which may be extended to a 'full' QR-decomposition with $Q \in \mathbb{R}^{m\times m}$, $R \in \mathbb{R}^{m\times n}$.

   The QR-decomposition can be computed in a finite number of floating point operations, e.g., using Householder or Givens transformations or a [modified] Gram-Schmidt process (see [2]).

   On the contrary, the similarity transformations involved in 3.–6. below which cannot be realized with a finite number of operations, since knowledge of the spectrum (the eigenvalues) of A is required.

3. ***Schur form:*** Every $A \in \mathbb{C}^{n\times n}$ can be written as $A = QRQ^H$, where Q is unitary and R is upper triangular. Furthermore, the eigenvalues of A appear on the diagonal of R .

4. ***Orthonormal diagonalization of symmetric matrices:*** If $A \in \mathbb{R}^{n\times n}$ is symmetric, then there exists an orthogonal matrix Q (whose columns are eigenvectors of A) and a diagonal matrix D (whose entries are the eigenvalues of A) such that $A = QDQ^{\mathrm{T}}$ (*spectral theorem for symmetric matrices*).

5. ***Unitary diagonalization of normal matrices:*** If $A \in \mathbb{R}^{n\times n}$ is normal, i.e., $A^{\mathrm{T}}A = AA^{\mathrm{T}}$, then there exists a unitary matrix $Q \in \mathbb{C}^{n\times n}$ (whose columns are eigenvectors of A) and a diagonal matrix $D \in \mathbb{C}^{n\times n}$ (whose entries are the eigenvalues of A) such that $A = QDQ^H$.

6. ***Symmetric positive definite (SPD) matrices:*** If $A \in \mathbb{R}^{n\times n}$ is SPD, i.e., if $(Ax, x) > 0$ for all $x \neq 0$, then its spectral decomposition reads $A = QDQ^{\mathrm{T}}$ with Q orthogonal and $D > 0$ (all eigenvalues of an SPD matrix are positive).
   - The SPD property is also simply denoted by $A > 0$.
   - With the extremal eigenvalues of an SPD matrix A, we have
   $$\|A\|_2 = \lambda_{max}, \quad \|A^{-1}\|_2 = \lambda_{min}^{-1}, \quad \text{and} \quad \kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}} \quad \text{(see (1.1))}$$
   - The *matrix square root* $A^{\frac{1}{2}} = QD^{\frac{1}{2}}Q^{\mathrm{T}}$ satisfying $A^{\frac{1}{2}}A^{\frac{1}{2}} = A$ is well-defined and is also SPD.

For symmetric matrices, the matrix R from 3. is exactly D from 4. We also note that the decompositions 4.–6. are 'characteristic properties', i.e., the 'if' can be replaced by 'if and only if'.

7. ***A nonsymmetric matrix A may also be (positive) definite***, i.e. it may satisfy $(Ax, x) > 0$ for all $x \neq 0$. This is the case iff its *symmetric, or real part* $\mathrm{Re}\,A = \frac{1}{2}(A + A^T)$ *is SPD*, since
   $$((A + A^{\mathrm{T}})x, x) = (Ax, x) + (x, Ax) = 2(Ax, x)$$
   In this case we say that A is positive definite (in contrast to 'SPD' which always means that A is also symmetric). We have
   $$(Ax, x) \geq \lambda_{min}(\mathrm{Re}\,A)(x, x)$$
   A general positive definite matrix is invertible, and the norm of its inverse can be estimated by
   $$\|A^{-1}\|_2 \leq \frac{1}{\lambda_{min}(\mathrm{Re}\,A)} \tag{1.2}$$

Exercise 1.2 Prove Assertion 7., in particular the estimate (1.2). ∎

[3] A more appropriate terminology would be 'orthonormal'.

1.4 Cayley-Hamilton Theorem

The Cayley-Hamilton Theorem (see, e.g., [10, Theorem 2.8.4]) states that any square matrix $A \in \mathbb{C}^{n\times n}$ satisfies it own characteristic equation, i.e., the characteristic polynomial $\chi\colon z \mapsto \det(zI - A)$, given by[4]

$$\chi(z) = (z - \lambda_1)\cdots(z - \lambda_n) = z^n + c_1 z^{n-1} + \cdots + c_{n-1} z + c_n \tag{1.3a}$$

satisfies

$$\chi(A) = A^n + c_1 A^{n-1} + \ldots + c_{n-1} A + c_n I = 0 \tag{1.3b}$$

A direct consequence of the Cayley-Hamilton theorem is that – provided A is invertible – the inverse A^{-1} can be expressed as follows (after by multiplying (1.3b) by A^{-1}):

$$A^{-1} = -\frac{1}{c_n} A^{n-1} - \frac{c_1}{c_n} A^{n-2} - \ldots - \frac{c_{n-1}}{c_n} I \tag{1.3c}$$

Note that $c_n = (-1)^n \det(A) \neq 0$ by assumption. This representation of A^{-1} in terms of a matrix polynomial of degree $n-1$ (with coefficients depending on the spectrum of A) may be a viewed as a motivation for the class of Krylov subspace methods which we consider later on.

Remark 1.1 For diagonalizable matrices A, the Cayley-Hamilton Theorem is easy to prove: $A = XDX^{-1}$ where D is a diagonal matrix with diagonal entries $D_{i,i} = \lambda_i$. For any polynomial π, we compute $\pi(A) = X\,\pi(D)\,X^{-1}$, with $\pi(D) = \mathrm{Diag}(\pi(\lambda_1), \pi(\lambda_2), \ldots, \pi(\lambda_n))$, which implies the assertion of the Cayley-Hamilton Theorem.

In the general case, one transforms A to Jordan normal form: $A = X\,J\,X^{-1}$, where the matrix J is block diagonal, and the diagonal blocks $J_i \in \mathbb{C}^{n_i\times n_i}$, $i = 1\ldots m$, are upper triangular matrices and have the eigenvalue λ_i on its diagonal. The size n_i is less than or equal to the multiplicity of the zero λ_i of the characteristic polynomial χ. Next, one observes that $\chi(A) = X\,\chi(J)\,X^{-1}$, and $\chi(J)$ is again block diagonal with the diagonal blocks being given by $\chi(J_i)$. Since for each i we can write $\chi(z) = \pi_i(z)(z - \lambda_i)^{n_i}$ for some polynomial π_i, the nilpotency property $(J_i - \lambda_i I)^{n_i} = 0$ implies $\chi(J_i) = 0$. Thus, $\chi(J) = 0$. ∎

[4] The λ_i are the eigenvalues of A; each eigenvalue with algebraic multiplicity k occurs k times in (1.3a).

2 Sparse Storage

In the introduction we have pointed out that we are targeting very large problems. One consequence of considering systems of this type is that the coefficient matrix has a huge number of elements. On the other hand, the matrices arising in most numerical simulations, for example, those based on Finite Difference (FD), Finite Element (FEM), or Finite Volume (FV) methods, are *sparse*, i.e., only few matrix entries are non-zero, due to the nature of the local approximations employed. (In practice, a matrix $A \in \mathbb{R}^{n\times n}$ is called sparse if the number of non-zero entries is $\mathcal{O}(n)$.)

The aim of *sparse storage* is to store only the non-zero elements of these matrices, but to do this in a manner that still enables efficient computations to be performed, especially matrix-vector products.

The following two standard examples are sparse matrices arising from FD or FEM methods.

Example 2.1 Let $\Omega = (0,1)$, $h = 1/(n+1)$, and $x_i = ih$, $i = 0\ldots n+1$. The FD discretization of the two-point boundary value problem (BVP), with $\Delta u = u''$,

$$-\Delta u(x) = f(x) \quad \text{on } \Omega, \quad u(0) = u(1) = 0 \tag{2.1}$$

is (with $u_i \approx u(x_i)$, $f_i = f(x_i)$)

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i, \quad i = 1\ldots n \tag{2.2}$$

where we set $u_0 = u_{n+1} = 0$. This is a system of linear equations of the form

$$Au = b, \quad A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^n$$

where the matrix A, which approximates $-h^2\Delta$, is tridiagonal and SPD with $\lambda_{max} = 4\sin^2\frac{n\pi}{2(n+1)} = \mathcal{O}(1)$ and $\lambda_{min} = 4\sin^2\frac{\pi}{2(n+1)} = \mathcal{O}(h^2)$, such that the condition number is $\kappa_2(A) = \mathcal{O}(h^{-2})$.

For this problem, all eigenvalues and eigenvectors (eigenmodes; in view of application background: think of 'discrete eigenfunctions') are explicitly known: For $i = 1\ldots n$, the vectors $w_i = \big((w_i)_1, \ldots, (w_i)_n\big)^{\mathrm{T}}$ with entries

$$(w_i)_j = \sin\frac{ji\pi}{n+1} = \sin(i\pi x_j), \quad j = 1\ldots n$$

are the eigenvectors associated with the eigenvalues $\lambda_i = 4\sin^2\dfrac{i\pi}{2(n+1)}$, $i = 1\ldots n$.

Small eigenvalues are associated with slowly varying eigenmodes; larger eigenvalues are associated with increasingly oscillating eigenmodes. Clearly, the eigenvectors w_i are mutually orthogonal since A is symmetric. ∎

Example 2.2 Let $\Omega = (0,1)^2$, $h = 1/(n+1)$. Define the nodes $P_{i,j} = (x_{i,j}) = (ih, jh)$, $i,j = 0\ldots n+1$. Consider the BVP (Dirichlet problem) for the 2D Poisson equation

$$-\Delta u(x) = f(x) \quad \text{on } \Omega, \quad u = 0 \quad \text{on } \partial\Omega \tag{2.3}$$

The FD approximations $u_{i,j}$ to the values $u(x_{i,j})$, based on the simplest 5-point discretization of the Laplacian Δ (the so-called '5-point-stencil'), are the solutions of

$$-u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} + 4u_{i,j} = h^2 f_{i,j}, \quad i,j = 1\ldots n \tag{2.4}$$

with $f_{i,j} = f(x_{i,j})$. The boundary conditions are enforced by setting $u_{0,j} = u_{n+1,j} = 0$ for $j = 0\ldots n+1$ and $u_{i,0} = u_{i,n+1} = 0$ for $i = 0\ldots n+1$.

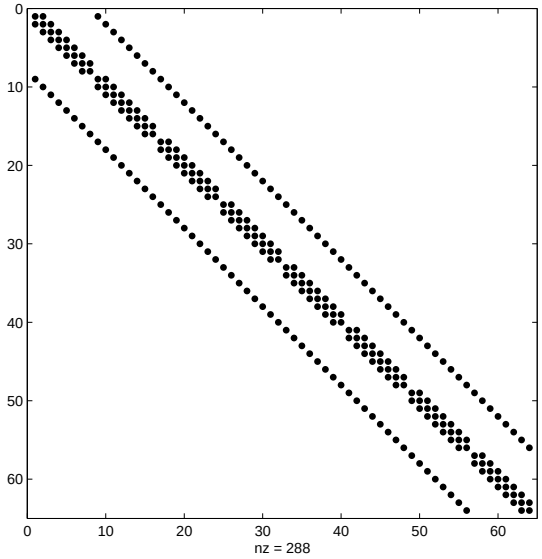


Figure 2.1: Sparsity pattern of the stiffness matrix A of a 2D Poisson problem

The system of equations (2.4) can be written in standard matrix-vector form $A\tilde{u} = b$ in the following way: A simple numbering of the nodes (x_i, y_j) and the unknowns $u_{i,j}$, $i, j = 1 \dots n$, is the 'lexicographic' (row-wise) ordering: We set $\tilde{u}_{(i-1)n+j} = u_{i,j}$. Then the matrix $A \in \mathbb{R}^{N\times N}$, with $N = n^2$, has at most 5 non-zero entries per row and column. An example of the sparsity pattern of A (for the case $n = 8$) can be seen in Fig. 2.1. The matrix A is again SPD and the eigenvalue satisfy $\lambda_{min} = 8\sin^2\frac{\pi h}{2} = \mathcal{O}(h^2)$, $\lambda_{max} = 8\cos^2\frac{\pi h}{2} = \mathcal{O}(1)$, such that $\kappa_2(A) = \mathcal{O}(h^{-2})$.

In order to describe the (mutually orthogonal) eigenvectors of A, it is convenient to use 'double index' notation, i.e., the matrix A has entries $A_{ii',jj'}$, where $i, i', j, j' \in \{1, \dots, n\}$. The eigenvectors (discrete eigenfunctions) are then given by $w_{i,j}$, $i, j = 1 \dots n$, with entries

$$(w_{i,j})_{i',j'} = \sin\frac{i\,i'\,\pi}{n+1}\,\sin\frac{j\,j'\,\pi}{n+1} = \sin(i\pi x_{i'})\,\sin(j\pi y_{j'}), \quad i', j' \in \{1, \dots, n\} \tag{2.5}$$

The corresponding eigenvalues are $\lambda_{i,j} = 4\left(\sin^2\frac{i\pi}{2(n+1)} + \sin^2\frac{j\pi}{2(n+1)}\right)$, $i, j = 1 \dots n$. ∎

For a Poisson equation in 3D (over a cube), a similar structure occurs. In this case, the resulting stiffness matrix A has at most 7 nonzero entries per row and column.

In these Poisson examples, band matrices are involved.[5] A few of the more straightforward methods for storing more general sparse matrices are considered in the following.

2.1 Coordinate format (COO)

The COO format consists of three one-dimensional arrays:[6]

- AA – an array containing the nonzero elements of $A \in \mathbb{R}^{n\times n}$
- IR – an integer array containing their row indices
- IC – an integer array containing their column indices

[5] Note, however, that the inverses of these matrices are dense.

[6] For rectangular matrices $A \in \mathbb{R}^{m\times n}$, all formats are defined in an analogous way.

Example 2.3

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 4 & -3 & 0 \\ 0 & 0 & -3 & 6 & -2 \\ 0 & 0 & 0 & -2 & 4 \end{pmatrix}$$

$$\begin{aligned} \text{AA} &= [\ 2 \quad -1 \quad 9 \quad -2 \quad 4 \quad -3 \quad -3 \quad 6 \quad -2 \quad -2 \quad 4\ ] \\ \text{IR} &= [\ 1 \quad 1 \quad 1 \quad 3 \quad 3 \quad 3 \quad 4 \quad 4 \quad 4 \quad 5 \quad 5\ ] \\ \text{IC} &= [\ 1 \quad 2 \quad 5 \quad 2 \quad 3 \quad 4 \quad 3 \quad 4 \quad 5 \quad 4 \quad 5\ ] \end{aligned}$$

■

Here, the entries are stored in row-wise order ('COOR'; 'COOC' can be defined in an analogous way). We note that the array IR contains quite a lot of repeated entries; the format is easy to handle but contains redundant information. A more economical way is to use the Compressed Row Storage (CRS) or the Compressed Column Storage (CCS) format discussed in the following.

2.2 Compressed (sparse) Row [Column] Storage formats (CRS, CCS)

The CRS format consists of three one-dimensional arrays:

- AA – an array containing the nonzero elements of $A \in \mathbb{R}^{n\times n}$
- IC – an integer array containing the column indices of the non-zero entries of A
- PR – an array of $n+1$ integer pointers;
  - — PR(i) is the position in AA and IC where the i-th row of A starts:
    The non-zero entries of the i-th row are AA([PR(i):PR(i+1)–1])
    and their column indices are IC([PR(i):PR(i+1)–1])
  - — If the i-th row of A contains only zeros, then PR(i) = PR(i+1)
  - — PR(n+1) = nnz+1, where nnz is the total number of non-zero entries of A

For A from Example 2.3 we have

$$\begin{aligned} \text{AA} &= [\ 2 \quad -1 \quad 9 \quad -2 \quad 4 \quad -3 \quad -3 \quad 6 \quad -2 \quad -2 \quad 4\ ] \\ \text{PR} &= [\ 1 \quad 4 \quad 4 \quad 7 \quad 10 \quad 12\ ] \\ \text{IC} &= [\ 1 \quad 2 \quad 5 \quad 2 \quad 3 \quad 4 \quad 3 \quad 4 \quad 5 \quad 4 \quad 5\ ] \end{aligned}$$

This is useful as the matrix vector product $y = Ax$ is simple to express (n is the number of rows in A):

```
for i = 1:n   %  MATLAB syntax
    k1 = pr(i)
    k2 = pr(i+1)-1
    y(i) = AA(k1:k2)*x(IC(k1:k2))' % inner product of row vectors AA and x
                                   % IC(k1:k2) is a vector index
end
```

In this example, e.g. for $i = 1$, we have `k1` $= 1$, `k2` $= 3$, AA(`k1:k2`) $= (2, -1, 9)$, and IC(`k1:k2`) $= (1, 2, 5)$ such that the first component y_1 of $y = Ax$ is evaluated according to

$$\texttt{AA(k1:k2)*x(IC(k1:k2))'} = \begin{pmatrix} 2 & -1 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_5 \end{pmatrix}$$

The storage saving achieved with the CRS format increases with the size of the matrix. For a matrix of dimension $n \times n$ with p or less elements per row it is required to store at most pn real values and $(p+1)n$ integers, i.e., the storage is proportional to n. For a $1,000 \times 1,000$ tridiagonal matrix, for instance, the CRS format requires less than 7,000 elements to be stored, while the full matrix has 1,000,000 elements.

Remark 2.1 Most iterative solution methods are based on the user's providing the matrix-vector multiplication $x \mapsto Ax$ by means of an appropriate *procedure* – the coefficient matrix A need not even be explicitly available for the algorithm to operate. The implication for the storage format is that (only) the multiplication $x \mapsto Ax$ has to be performed efficiently. Typically, one expects the storage format to be such that the cost of $x \mapsto Ax$ is proportional to the number of non-zero entries of A. This is the case for CRS.

In special cases, in particular constant coefficient problems, no storage of matrix entries at all is required for evaluation of $x \mapsto Ax$. Rather, this is realized by a simple loop incorporating the constant coefficients, as is the case for instance for the FD discretization of the 1D and 2D Poisson equation with constant meshwidth h. ∎

Exercise 2.1 Design analogously to the CRS format the CCS ('compressed sparse column format'). Formulate an algorithm that realizes the matrix-vector multiplication $x \mapsto Ax$. ∎

A wide variety of other sparse formats exist, often motivated from the particular structure of a problem under consideration. A classical example are *banded matrices*.

Example 2.4 A matrix $A \in \mathbb{R}^{n \times n}$ is said to be *banded* with *upper bandwidth* b_u und *lower bandwidth* b_l, if $A_{i,j} = 0$ for $j > i + b_u$ and $j < i - b_l$. If $b = b_u = b_l$, then b is called the bandwidth of A. The storage requirement amounts to $n(b_u + b_l + 1)$ real numbers. Typically, banded matrices are stored by [off-]diagonals (cf. the `diag` command in MATLAB). ∎

Exercise 2.2 Design a data structure for data-sparse storage of a *symmetric* tridiagonal matrix and realize matrix-vector multiplication. ∎

MATLAB includes many functions for use with sparse matrices.[7] The internal format is CCS (see Example 2.1), which is also called the Harwell-Boeing format. To preallocate storage for a sparse matrix A of dimension $m \times n$ with k non-zero elements the function `A = spalloc(m,n,k)` is called. The non-zero elements of A can then be entered by indexing the elements of A as usual.

The sparsity pattern of the matrix A can be viewed using the function `spy(A)`. In addition all the MATLAB functions can be used with sparse matrices including the standard addition and multiplication operations. Sparse vectors are also available but less relevant in practice.

Example 2.5

```
A = gallery('poisson',8);
spy(A)
```

This retrieves the 64×64-matrix A obtained by discretizing the Poisson equation with the 5-point stencil (see Example 2.2) from the gallery of test matrices, and then plots the sparsity pattern shown in Fig. 2.1 on p. 7. ∎

In general, the algorithms discussed from now on will be assumed to be using some appropriate sparse storage technique.[8]

[7] see `help sparse`. MATLAB is a trademark of The MathWorks, Inc.

[8] For an overview of sparse matrix formats, see `http://en.wikipedia.org/wiki/Sparse_matrix`.

3 Direct Solution Methods

For 'small' problems, direct methods (i.e., variants of Gaussian elimination) are the method of choice. While there is no general rule what 'small' means, sparse matrices arising from FD or FEM discretizations with up to 100,000 unknowns are often treated by direct methods. Especially for 2D problems, direct methods are quite popular. We refer to [8, 5] for a good discussion of such methods. The available solvers are suitable for nonsymmetric problems and support parallel computer architectures.

Generally speaking, whereas Gaussian elimination of a (full) $n \times n$-matrix requires $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ operations, typical sparse matrices can be factored with work $\mathcal{O}(n^\alpha)$ for some $1 < \alpha < 3$ (for 2D FEM-applications, we expect $\alpha = 3/2$). Since $\alpha = 1$ is not really achievable, iterative methods have to be employed for very large problems.

The main practical issues in direct solvers are

- *pivoting* (to ensure numerical stability), and
- *reordering strategies* for the unknowns so as to keep *fill-in* small.

These two requirements are usually incompatible, and a compromise has to be made, e.g. by allowing non-optimal pivot elements to some extent. Here we consider the important special case of sparse SPD matrices, since these can be factored without pivoting in a numerically stable way via the Cholesky algorithm. For SPD matrices, one may therefore concentrate on reordering strategies to minimize fill-in.

3.1 Fill-in

For an SPD matrix $A \in \mathbb{R}^{n\times n}$ its *envelope* (or *profile*) is defined as the set of double indices

$$\mathrm{Env}(A) = \{(i,j) : J_i(A) \le j < i\}, \quad \text{where} \quad J_i(A) = \min\{j : A_{i,j} \neq 0\}$$

The envelope corresponds to a variable band structure which contains the indices of all non-zeros entries of the (strictly) lower part of A. A key feature for sparse direct solvers is the observation that also the Cholesky factor has non-zero entries only within the envelope of A:

Theorem 3.1 *Let $A \in \mathbb{R}^{n\times n}$ be SPD, and let $L \in \mathbb{R}^{n\times n}$ be its lower triangular Cholesky factor, i.e., $L\,L^T = A$. Then $L_{i,j} = 0$ for $j < i$ and $(i,j) \notin \mathrm{Env}(A)$.*

Proof: The proof (an exercise) follows from inspection of the way the Cholesky decomposition is computed;[9] see e.g. [2]. An analogous result holds for LU-decompositions of general matrices (provided the LU-decomposition can be performed without pivoting). □

Example 3.1 Banded matrices are a special case: Here, $J_i(A) = \max\{1, i-b\}$ for all i, where b is the bandwidth. The storage requirement is then $\mathcal{O}(n\,b)$. The factorization of A is done with work $\mathcal{O}(n\,b^2)$. ∎

Theorem 3.1 shows that the sparsity pattern of a matrix A is *roughly* inherited by the Cholesky factor. More precisely, *the envelope is inherited*. Indices $(i,j) \in \mathrm{Env}(A)$ for which $A_{i,j} = 0$ but $L_{i,j} \neq 0$ are called *fill-in*. Since, generally speaking, the majority of the $(i,j) \in \mathrm{Env}(A)$ will be filled in during the factorization, many efficient sparse direct solvers aim at finding a reordering of the unknowns such that the envelope is small. In other words: they are based on finding a permutation matrix P such that $\mathrm{Env}(P^{\mathrm{T}}A\,P)$ is small. The Reverse Cuthill-McKee (RCM) ordering (see Sec. 3.2) is a classical example.

[9] The Cholesky decomposition $A = L\,L^{\mathrm{T}}$ is based on a modified elimination procedure, i.e., $A = LU$ with additional scaling of columns such that $U = L^{\mathrm{T}}$, and $1 \neq L_{i,i} > 0$.

$$A = \begin{pmatrix} 1 & & & 1 & & & 1 \\ & 1 & & 2 & & & 2 \\ & & 1 & 3 & & & 3 \\ 1 & 2 & 3 & 15 & & & 18 \\ & & & & 1 & & 5 \\ & & & & & 1 & 6 \\ 1 & 2 & 3 & 18 & 5 & 6 & 92 \end{pmatrix} \qquad L = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ 1 & 2 & 3 & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix}$$

$$\hat{A} = \begin{pmatrix} 92 & 6 & 5 & 18 & 3 & 2 & 1 \\ 6 & 1 & & & & & \\ 5 & & 1 & & & & \\ 18 & & & 15 & 3 & 2 & 1 \\ 3 & & & 3 & 1 & & \\ 2 & & & 2 & & 1 & \\ 1 & & & 1 & & & 1 \end{pmatrix} \qquad \hat{L} = \begin{pmatrix} 9.5917 & & & & & & \\ 0.6255 & 0.7802 & & & & & \\ 0.5213 & -0.4180 & 0.7440 & & & & \\ 1.8766 & -1.5047 & -2.1601 & 2.1327 & & & \\ 0.3128 & -0.2508 & -0.3600 & 0.5899 & 0.6014 & & \\ 0.2085 & -0.1672 & -0.2400 & 0.3933 & -0.7075 & 0.4644 & \\ 0.1043 & -0.0836 & -0.1200 & 0.1966 & -0.3538 & -0.8444 & 0.3015 \end{pmatrix}$$

Figure 3.1: Top: arrowhead matrix $A \in \mathbb{R}^{7\times 7}$ and its Cholesky factor `L = chol(A)'`. Bottom: effect of reversing the numbering.

Example 3.2 (fill-in) The 7×7 'arrowhead matrix' A shown in Fig. 3.1 has a 'good' ordering: Since *all* entries of A within the envelope are non-zero, there is *no fill-in*. Reversing the order of equations and unknowns, which leads to the matrix $\hat{A} = P^{\mathrm{T}} A P$ with corresponding permutation P, gives rise to a disaster: The envelope is the full lower part of $\hat{A}$, an in $\hat{L}$ shows that complete fill-in has taken place. ∎

A closer look at fill-in for SPD matrices.

Fill-in takes place at (i,j) if $A_{i,j} = 0$ but $L_{i,j} \neq 0$. Due to Theorem 3.1, fill-in can only occur within the envelope of A. We now determine the fill-in more precisely by means of an inductive procedure describing Cholesky elimination in a similar way as described in [2]: Let $A \in \mathbb{R}^{n\times n}$ be SPD. Elementary calculations then show (with $a_{11} \in \mathbb{R}$, $a = A([2\!:\!n], 1) \in \mathbb{R}^{n-1}$, $\bar{A} = A([2\!:\!n],[2\!:\!n]) \in \mathbb{R}^{(n-1)\times(n-1)}$):

$$A = A^{(1)} = \begin{pmatrix} a_{11} & a^{\mathrm{T}} \\ a & \bar{A} \end{pmatrix} = \underbrace{\begin{pmatrix} \sqrt{a_{11}} & 0 \\ \frac{a}{\sqrt{a_{11}}} & I_{n-1} \end{pmatrix}}_{=:L_1} \begin{pmatrix} 1 & 0 \\ 0 & \underbrace{\bar{A} - \frac{aa^{\mathrm{T}}}{a_{11}}}_{=:A^{(2)}} \end{pmatrix} \underbrace{\begin{pmatrix} \sqrt{a_{11}} & \frac{a^T}{\sqrt{a_{11}}} \\ 0 & I_{n-1} \end{pmatrix}}_{=L_1^T} \tag{3.1a}$$

In this way we have eliminated the first row and column, and the rest of the job consists in continuing this procedure by factoring the matrix[10] $A^{(2)} = \bar{A} - \frac{1}{a_{11}} aa^{\mathrm{T}} \in \mathbb{R}^{(n-1)\times(n-1)}$ in an analogous way. This gives

$$A^{(2)} = L_2 \begin{pmatrix} 1 & 0 \\ 0 & A^{(3)} \end{pmatrix} L_2^{\mathrm{T}}, \tag{3.1b}$$

where $A^{(3)} \in \mathbb{R}^{(n-2)\times(n-2)}$ is again SPD, and $L_2 \in \mathbb{R}^{(n-1)\times(n-1)}$ has a structure similar to that of L_1. Thus,

$$A = A^{(1)} = \underbrace{L_1}_{=:\tilde{L}_1} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & L_2 \end{pmatrix}}_{=:\tilde{L}_2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & A^{(3)} \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & L_2 \end{pmatrix}^{\mathrm{T}}}_{=\tilde{L}_2^T} \underbrace{L_1^{\mathrm{T}}}_{=\tilde{L}_1^T} \tag{3.1c}$$

[10] The matrix $A^{(2)}$ is again SPD since $L_1^{-1} A L_1^{-T}$ is SPD, see (3.1a).

Proceeding in this way, we obtain a factorization

$$A = \tilde{L}_1 \tilde{L}_2 \cdots \tilde{L}_{n-1}\, I\, \tilde{L}_{n-1}^{\mathrm{T}} \cdots \tilde{L}_2^{\mathrm{T}} \tilde{L}_1^{\mathrm{T}} =: LL^{\mathrm{T}}, \quad \text{with} \quad L = \tilde{L}_1 \tilde{L}_2 \cdots \tilde{L}_{n-1}$$

Remark 3.1 The k-th step corresponds to elimination of the variable x_k. The factors $\widetilde{L}_i$ are lower triangular matrices with a special structure: $(\tilde{L}_k)_{i,i} = 1$ for $i \neq k$, and the only non-trivial column of $\tilde{L}_k$ is column k. Clearly, $L = \tilde{L}_1 \cdots \tilde{L}_{n-1}$ is again lower triangular; moreover, we have $L(:,k) = \tilde{L}_k(:,k)$, which is analogous to the case of standard LU-decomposition, where the lower factor L is nothing but the recombination of elementary elimination matrices L_k with one one-trivial column k, see [2]. ∎

Constructing the Cholesky decomposition in this way is known as the 'outer product variant', since the updates for the $A^{(k)}$ are expressed by outer (dyadic) vector products; see (3.1a). This is formalized in Alg. 3.1 representing the explicit algorithmic outcome of the inductive process indicated above.

Algorithm 3.1 Cholesky decomposition – outer product variant

% returns the Cholesky factor $L \in \mathbb{R}^{n\times n}$ of an SPD matrix $A = A^{(1)} \in \mathbb{R}^{n\times n}$
% note: *unusual choice of indices (see* (3.1)*):*
% the matrices $A^{(k)} \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$ are of the form $(A^{(k)}_{i,j})_{\mathbf{i,j=k}}^{\mathbf{n}}$

1: $A^{(1)} = A, \quad L = 0 \in \mathbb{R}^{n\times n}$,
2: **for** $k = 1 \ldots n-1$ **do**
3: $\quad L_{k,k} = \sqrt{A^{(k)}_{k,k}}$
4: $\quad L([k+1:n],k) = \frac{1}{L_{k,k}} \cdot A^{(k)}([k+1:n],k)$ % column vector
5: $\quad A^{(k+1)}([k+1:n],[k+1:n]) =$
$\qquad A^{(k)}([k+1:n],[k+1:n]) - L([k+1:n],k) \cdot (L([k+1:n],k))^{\mathrm{T}}$ % outer product
6: **end for**

We see how fill-in arises: The first column $L_{:,1} = \frac{1}{a_{11}} A([2:n],1)$ of the Cholesky factor L has non-zero entries only where the first column of $A^{(1)} = A$ has non-zero entries, see (3.1a). *The second column of L has non-zero entries where the first column of the submatrix $A^{(2)} = \bar{A} - \frac{1}{a_{11}} aa^T \in \mathbb{R}^{(n-1)\times(n-1)}$ has non-zero entries,* and so on. In general, we expect $L_{i,k} \neq 0$ if $A^{(k)}_{i,k} \neq 0$. From the update formula for the matrices $A^{(k)}$ (see lines 3–5 of Alg. 3.1), we have

$$A^{(k+1)}_{i,j} = A^{(k)}_{i,j} - \frac{1}{A^{(k)}_{k,k}} A^{(k)}_{i,k} A^{(k)}_{k,j}, \qquad i,j = k+1 \ldots n$$

Hence, for $i,j \geq k+1$, we have $A^{(k+1)}_{i,j} \neq 0$ if[11]

$$A^{(k)}_{i,j} \neq 0 \quad \text{or} \quad A^{(k)}_{i,k} A^{(k)}_{k,j} \neq 0$$

Another way of putting it is: $A^{(k+1)}_{i,j} \neq 0$ if either $A^{(k)}_{i,j} \neq 0$, or if in $A^{(k)}$ the indices i,j are *connected to each other via the index k*, i.e., $A^{(k)}_{i,k} \neq 0$ together with $A^{(k)}_{k,j} \neq 0$. Based on this observation, one can precisely characterize the fill-in process for the Cholesky decomposition.

[11] In a strict sense, this is not an 'if and only if' situation since cancellation can take place, i.e., $A^{(k)}_{i,j} = \frac{1}{A^{(k)}_{k,k}} A^{(k)}_{i,k} A^{(k)}_{k,j}$. This (unlikely) cancellation will be ignored.

Fill-in from a graph theoretical point of view.

An elegant way to study fill-in is done in terms of graphs. A graph $G = (V, E)$ consists of a set V of *nodes* and a set of *edges* $E \subseteq V \times V$. Edges are denoted as pairs $(v, v') \in V \times V$ with two distinct[12] elements.

The sparsity pattern of a general matrix A can be represented by a graph $G = (V, E)$ with nodes V and edges E, its so-called *adjacency graph.* Here,

- the set V of nodes is simply the set of unknowns $\{x_i, i = 1 \dots n\}$ (or the corresponding indices i),
- two nodes $x_i \neq x_j$ are connected by an edge $(x_i, x_j) \in E$ iff $A_{i,j} \neq 0$, i.e. if equation i involves the unknown x_j.

In the general setting of nonsymmetric matrices this gives a *directed* graph where vertices x_i and x_j are connected by a *directed edge* (x_i, x_j) if $A_{i,j} \neq 0$. A directed edge (x_i, x_j) is visualized by an arrow pointing from node x_i to node x_j.

For a symmetric matrix A we have $A_{i,j} = A_{j,i}$, thus E is symmetric: $(x_i, x_j) \in E$ iff $(x_j, x_i) \in E$, and this is represented by $\{x_i, x_j\}$. In other words: We use *undirected* graphs. An (undirected) edge $\{x_i, x_j\}$ is visualized by a line joining the nodes x_i and x_j.

The *degree* $\deg_G(v)$ of a node $v \in V$ is the number of edges emanating from v.

In the outer product variant of Cholesky decomposition, we denote by $G^{(k)} = (V^{(k)}, E^{(k)})$ the adjacency graph of $A^{(k)} \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$. The above discussion shows that the graph $G^{(k+1)}$ for $A^{(k+1)}$ is obtained from the graph $G^{(k)}$ by

removing the 'pivot node' x_k, removing the edges emanating from x_k, and adding edges $\{x_i, x_j\}$ that connect those nodes x_i, x_j that have the property that $\{x_i, x_k\} \in E^{(k)}$ together with $\{x_k, x_j\} \in E^{(k)}$.

We formalize this in Alg. 3.2. We call the sequence of graphs $(G^{(k)})_{k=1}^n$ the *elimination history.*

Algorithm 3.2 Elimination pattern via graph transformation

% input: graph $G = (V, E)$
% output: graph $G' = (V', E')$ obtained by eliminating node $v \in V$
1: $V' = V \setminus \{v\}$
2: $E' = \{\{v_1, v_2\} \in V' \times V' : \{v_1, v_2\} \in E \text{ or } \big(\{v_1, v\} \in E \text{ and } \{v, v_2\} \in E\big)\}$

Example 3.3 (See Fig. 3.2.) Let

$$A = A^{(1)} = \left(\begin{array}{c|cccc} * & 0 & \boxed{*} & * & * & 0 \\ \hline 0 & * & * & 0 & 0 & * \\ * & * & * & 0 & 0 & 0 \\ \boxed{*} & 0 & \boxed{0} & * & 0 & 0 \\ * & 0 & 0 & 0 & * & 0 \\ 0 & * & 0 & 0 & 0 & * \end{array}\right)$$

with $G^{(1)} = (V^{(1)}, E^{(1)})$, $V^{(1)} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $E^{(1)} = \{\{x_1, x_3\}, \{x_1, x_4\}, \{x_1, x_5\}, \{x_2, x_3\}, \{x_2, x_6\}\}$. Elimination of x_1 results in the 5×5-submatrix

[12] We point out that this implies that the graph has no 'loops' connecting a node with itself. In other words: Information about the diagonal entries is not contained in the graph because it is of no interest for the study of fill-in.

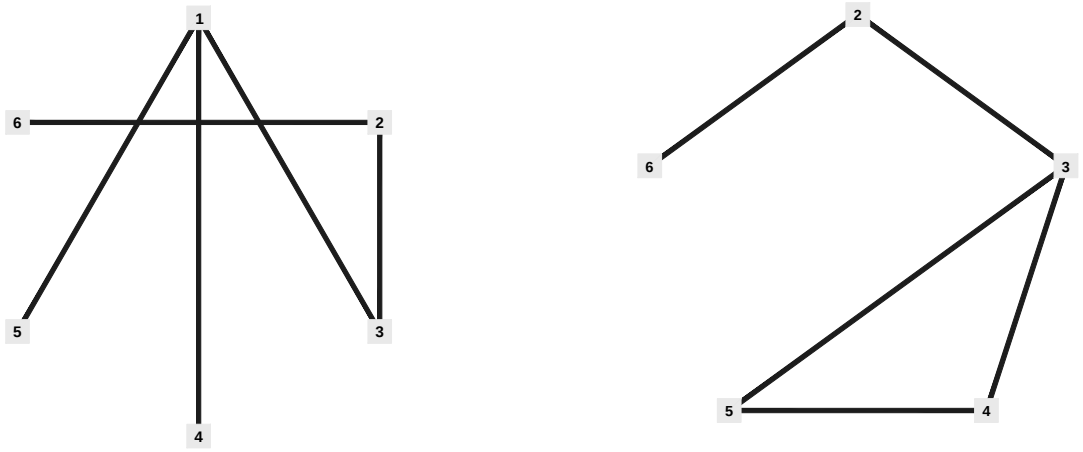


Figure 3.2: Adjacency graphs $G^{(1)}$ and $G^{(2)}$ from Example 3.3

$$A^{(2)} = \begin{pmatrix} * & * & 0 & 0 & * \\ * & * & * & * & 0 \\ 0 & \boxed{*} & * & * & 0 \\ 0 & * & * & * & 0 \\ * & 0 & 0 & 0 & * \end{pmatrix}$$

with $G^{(2)} = (V^{(2)}, E^{(2)})$, $V^{(2)} = \{x_2, x_3, x_4, x_5, x_6\}$, $E^{(2)} = \{\{x_2, x_3\}, \{x_2, x_6\}, \boxed{\{x_3, x_4\}}, \{x_3, x_5\}, \{x_4, x_5\}\}$.

Here, for instance, elimination of x_1 gives rise to fill-in at the position indicated by $\boxed{*}$ in $A^{(2)}$, because $A^{(1)}_{4,1} \neq 0$ and $A^{(1)}_{1,3} \neq 0$. This, among others, generates the edge $\boxed{\{x_3, x_4\}} \in E^{(2)}$. ∎

The above discussion shows that the elimination of node x_k produces column $L(:,k)$ of the Cholesky factor L with the property that $L_{i,k} \neq 0$ iff $\{x_i, x_k\} \in E^{(k)}$. Hence, *the number of non-zero entries in column $L(:,k)$ is given by the degree of node $x_k \in V^{(k)}$*. The memory requirement to store the Cholesky factor L is therefore given by

$$\mathrm{Mem}(L) = \sum_{k=1}^{n} \mathrm{Mem}(L(:,k)) = n + \sum_{k=1}^{n-1} \deg_{G^{(k)}}(x_k)$$

where n represents storage of the diagonal. However, it is generally very difficult to *predict* what the precise amount of fill-in will be during the elimination process.

Elimination graph and reordering.

A reordering of the unknowns (and analogous simultaneous reordering of the equations) corresponds to a permutation of the columns and rows of A, i.e., it leads to the SPD matrix $P^{\mathrm{T}} A P$ for some permutation matrix P. The graph $G^{(1)}$ is (up to labeling of nodes) independent of the permutation P. In terms of graphs, we note that if we eliminate the permuted nodes one by one (with Alg. 3.2), then we obtain a sequence of graphs $(G^{(k)})_{k=1}^{n}$ that corresponds to the Cholesky decomposition of $P^{\mathrm{T}} A P$, and the permutation P describes the order in which the nodes are eliminated. Hence, we arrive at the following result concerning the fill-in for Cholesky decomposition with any ordering:

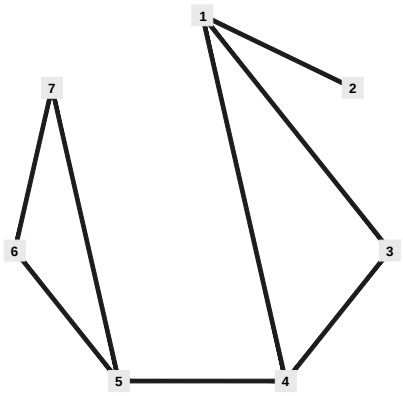


Figure 3.3: Here, $S = \{4, 5\}$ is a separator.

Theorem 3.2 *Let $G^{(1)} = (V^{(1)}, E^{(1)})$ be the adjacency graph for an SPD matrix A. Eliminate sequentially nodes $v_1, v_2, \ldots, v_n$ using Alg. 3.2 and denote by $G^{(k)}$, $k = 1 \ldots n$, the graphs obtained in this process. Then the sequence $(G^{(k)})_{k=1}^n$ represents the elimination history for the Cholesky decomposition of $P^T A P$, where P is determined by the order in which the nodes are eliminated. The location of the non-zero entries of the Cholesky factor L of $P^T A P$ can be read off $(G^{(k)})_{k=1}^n$: For $i > k$ there holds $L_{i,k} \neq 0$ if $\{v_i, v_k\} \in E^{(k)}$. The total memory requirement to store L is*

$$n + \sum_{k=1}^{n-1} \deg_{G^{(k)}}(x_k) \tag{3.2}$$

Graph theory terminology.

Neighbors and degree: A *neighbor* of a node v is another node that is connected to v by an edge. By $\mathrm{Adj}(v)$ we denote the set of all neighbors of $v \in V$.[13] Recall that the *degree* $\deg(v)$ of a node $v \in V$ is the number of edges emanating from v, i.e., $\deg(v) = |\mathrm{Adj}(v)|$. More generally, for a subset $V' \subseteq V$, the union of all nodes *outside* V' connected with some node $v' \in V'$ is denoted by

$$\mathrm{Adj}(V') = \bigcup_{v' \in V'} \mathrm{Adj}(v') \setminus V'$$

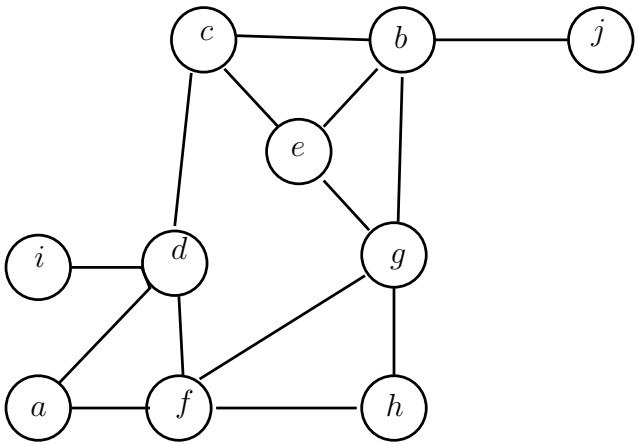
Path: A path connecting a node $v \in V$ to a node $v' \in V$ is a tuple of nodes v_i, $i = 1 \ldots \ell + 1$, with $v_1 = v$ and $v_{\ell+1} = v'$ such that each $\{v_i, v_{i+1}\} \in E$.

Connected graph, diameter of a graph: A graph is *connected* if any two nodes can be connected by a path. The *distance* $d(v, v')$ between two nodes v, v' is the length of the shortest path connecting v with v'. The *diameter* of a connected graph G is $\mathrm{diam}(G) = \max_{v,v' \in V} d(v, v')$, i.e., the longest distance between two nodes .

Separator: A subset $S \subseteq V$ is called a *separator* of G, if the graph G' obtained from G by removing the nodes of S and the edges emanating from S is *not* connected. That is, $V \setminus S$ has the form $V \setminus S = V_1 \dot\cup V_2$, and every path connecting a node $v_1 \in V_1$ with a node $v_2 \in V_2$ intersects the separator S. See Fig. 3.3.

Eccentricity of a node v, peripheral node: The *eccentricity* $e(v)$ of a node $v \in V$ is defined as $e(v) = \max_{v' \in V} d(v, v')$. A node $v \in V$ with $e(v) = \mathrm{diam}(G)$ is called *peripheral*. A peripheral node is 'far outside' because there exists a path of maximal length emanating from it.

[13] Note: since we have excluded $\{v, v\}$ from the set of edges, we have $v \notin \mathrm{Adj}(v)$.



i	v	content of FIFO
1	g	h, e, b, f
2	h	e, b, f
3	e	b, f, c
4	b	f, c, j
5	f	c, j, a, d
6	c	j, a, d
7	j	a, d
8	a	d
9	d	i
10	i	–

Figure 3.4: Example of Cuthill-McKee algorithm with non-peripheral starting node g.

3.2 Standard ordering strategies

As we have seen in Example 3.2, the order in which the unknowns are numbered can have a tremendous effect on the amount of fill-in, which in turn affects the storage requirement for the Cholesky factor and the time to compute it. Modern sparse direct solvers analyze the matrix A prior to factorization and aim at determining a good ordering of the unknowns. Since the problem of finding a permutation matrix P that minimizes the amount of fill-in is a very hard problem, various heuristic strategies have been devised. Popular ordering methods are:

1. *Reverse Cuthill-McKee:* Realized in MATLAB as `symrcm` (symmetric version). This ordering may be motivated by minimizing the bandwidth of the reordered matrix $P^{\mathrm{T}}AP$.
2. *Nested dissection:* This ordering originates from FD / FEM applications with *substructuring*.
3. *(Approximate) minimum degree:* The approximate minimum degree ordering of [1] is currently the most popular choice, realized in MATLAB as `symamd` (symmetric version). It aims at minimizing the amount of fill-in.

We refer to [16, 5, 8] for good surveys on the topic.

[Reverse] Cuthill-McKee.

The Cuthill-McKee (CM) and the Reverse Cuthill-McKee (RCM) orderings can be viewed as attempts to minimize the bandwidth of a sparse matrix in a cheap way. The underlying heuristic idea is as follows:

In order to obtain a small bandwidth, it is important to *assign neighboring nodes in the graph G numbers that are close together*. Hence, as soon as one node is assigned a number, then all its neighbors that have not been assigned a number yet should be numbered as quickly as possible – see Alg. 3.3.

This is a typical example of a 'greedy algorithm' based at a *brute force*, locally optimal strategy with the hope that the outcome is also near to optimal in the global sense.

The choice of a starting node is, of course, important. One wishes to choose a peripheral node as a starting node. Since these are in practice difficult to find, one settles for a pseudo ('nearly') peripheral node, as described in [8].

It has been observed that better orderings are obtained by reversing the Cuthill-McKee ordering. In fact, it can be shown, [16, p. 119], that RCM is always better than CM in the sense that $|\mathrm{Env}(P_{RCM}^{\mathrm{T}}AP_{RCM})| \leq |\mathrm{Env}(P_{CM}^{\mathrm{T}}AP_{CM})|$. The RCM algorithm is shown in Alg. 3.4.

Algorithm 3.3 Cuthill-McKee

1: choose a starting node v and put it into a FIFO % 'first in – first out' – a queue, or pipe
2: **while** (FIFO $\neq \emptyset$) {
3: take first element v of FIFO and assign it a number
4: let $V' \subseteq \mathrm{Adj}(v)$ be those neighbors of v that have not been numbered yet;
5: put them into the FIFO in ascending order of degree (ties are broken arbitrarily).
6: }

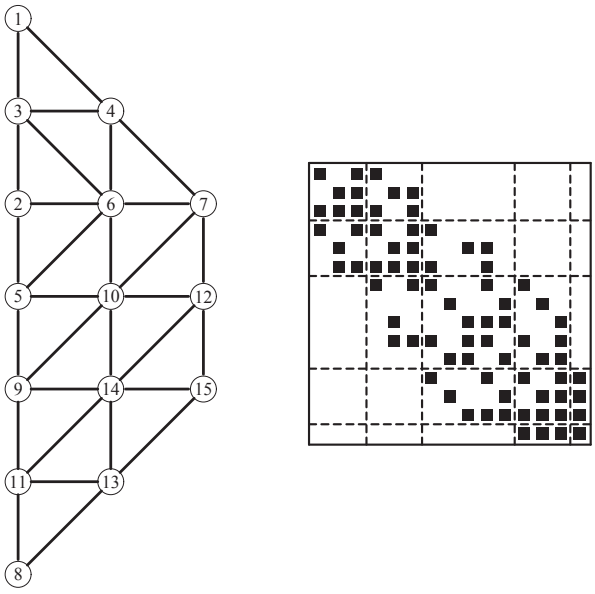


Figure 3.5: Example of Reverse Cuthill-McKee algorithm (example taken from [19]).

Algorithm 3.4 Reverse Cuthill-McKee

1: choose as a starting node v a peripheral or pseudo-peripheral node
2: determine the CM ordering using Alg. 3.3.
3: reverse the CM ordering to get the RCM ordering.

Nested Dissection.

The key idea of nested dissection is to find a *separator S that is small*. Recall that a separator S for the graph G partitions the set of nodes V into three disjoint sets,

$$V = V_1 \dot\cup V_2 \dot\cup S$$

with the property that *no* edges exist that connect nodes of V_1 directly with nodes of V_2. If we number the nodes of V_1 first, then the vertices of V_2 and the nodes of S last, the matrix A has the following block structure:

$$A = \begin{pmatrix} A_{V_1,V_1} & 0 & A_{V_1,S} \\ 0 & A_{V_2,V_2} & A_{V_2,S} \\ A_{S,V_1} & A_{S,V_2} & A_{S,S} \end{pmatrix}$$

Theorem 3.1 tells us that the Cholesky decomposition of A will inherit the two 0-blocks which are outside $\mathrm{Env}(A)$. Therefore it is highly desirable to choose the separator S to be *small*, because then these 0-blocks

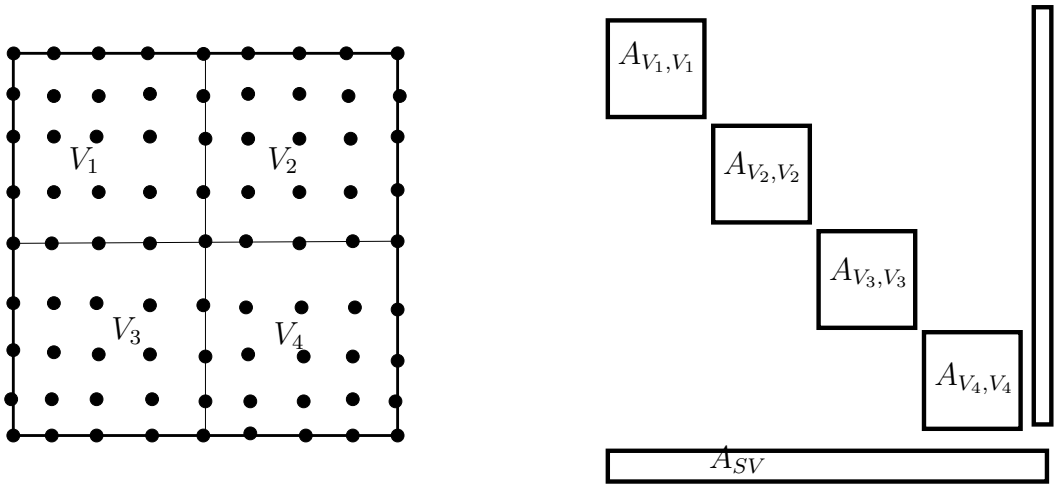


Figure 3.6: left: nested dissection of a square. right: block structure of the resulting matrix.

are large. (In the extreme case $S = \emptyset$ the adjacency graph of A is not connected, and A becomes block diagonal.)

Nested dissection is usually applied *recursively* to the sets V_1 and V_2, resulting in a 'fork-like' structure for Env(A) (see, e.g., Fig. 3.7), where the 'prongs' are the blocks created by the separators. The nested dissection Algorithm is specified in Alg. 3.5.

Algorithm 3.5 Nested Dissection

% input: adjacency graph $G = (V, E)$ of A
% output: numbering for the vertices

1: select a separator S and sets V_1, V_2 such that
   a) $V = V_1 \dot\cup V_2 \dot\cup S$,
   b) S is small,
   c) no node of V_1 shares an edge with any node of V_2.
2: number the nodes of V by
   a) numbering those of V_1 first (i.e., recursive call of this algorithm with V_1 as input),
   b) numbering those of V_2 second (i.e., recursive call of this algorithm with V_2 as input),
   c) numbering those of S last.

The 'art' in nested dissection lies in finding good separators. We illustrate that it is a feasible task. In fact, for the 2D Poisson problem on a square (Example 2.2) the choice of good separators leads to very little fill-in, namely $\mathcal{O}(N \log N)$:

Example 3.4 We consider a uniform mesh as in Example 2.2. For simplicity, we assume that n is a power of 2: $n = 2^m$. As indicated in Fig. 3.6, the unit square is split into 4 boxes. By our choice of n, the two lines that split the square into 4 squares are mesh lines.

In this example, it is better to separate the unknowns into 5 sets V_1, V_2, V_3, V_4, and the separator S. This splitting is done as indicated in Fig 3.6: All nodes in the box V_1 are numbered first, then those of the box V_2, then those of the box V_3, then those of box V_4, and finally those that lie in the separator S. Then the system matrix A has the block structure depicted in Fig. 3.6. Each of the matrices $A_{V_1,V_1}, \ldots, A_{V_4,V_4}$ has size $\approx N/4$ and is, up to the size, essentially the same as the original matrix A. The last block, A_{SV} is of size $\mathcal{O}(\sqrt{N}) \times N$. Since the submatrices A_{V_i,V_i} are similar to A, we can repeat the procedure recursively. An idea of the structure of the resulting envelope can be obtained from Fig. 3.7, where the sparsity pattern of the Cholesky factor for the case $n = 30$ is plotted.

A careful analysis given in [7] shows for this model problem that with this ordering of the unknowns, the memory requirement for the Cholesky factor is $\mathcal{O}(N \log N)$, and the number of arithmetic operations to compute it is $\mathcal{O}(N^{3/2})$. Thus the fill-in is considerably smaller than for straightforward lexicographic ordering, where the memory requirement for the Cholesky factor is $\mathcal{O}(N^{3/2})$. ∎

Example 3.5 We again consider the 2D Poisson problem on a square (Example 2.2) with the original numbering ('lexicographic' ordering). We observe that the bandwidth of the matrix A with this ordering in $b = \mathcal{O}(n) = \mathcal{O}(\sqrt{N})$. The size of the envelope is then $E_N = \mathcal{O}(Nb) = \mathcal{O}(N^{3/2})$, which is significantly larger than that of nested dissection. Fig. 3.7 shows that virtually the full envelope is filled during the factorization. ∎

Minimum degree ordering

RCM ordering aims at minimizing the bandwidth of a matrix A. The ultimate goal, however, is to minimize the fill-in rather than the bandwidth. This is the starting point of the *minimum degree algorithm*. Finding the ordering that really minimizes the fill-in is a hard problem; the minimum degree algorithm, [20], is a greedy algorithm that aims at minimizing the fill for each column $L_{:,k}$ of the Cholesky factor separately.

The algorithm proceeds by selecting a starting node v_1 of $V^{(1)}$ and computes $G^{(2)}$; then a node $v_2 \in V^{(2)}$ is selected, $G^{(3)}$ is computed, etc. From (3.2), we see that the choice of node v_k adds $\deg_{G^{(k)}}(v_k)$ to the memory requirement for the Cholesky factor. The 'greedy' strategy is to select v_k (given that $v_1, \ldots, v_{k-1}$ have already been selected) such that $\deg_{G^{(k)}}(v_k)$ is minimized, i.e., we *choose a node of minimum degree*. This procedure is formalized in Alg. 3.6.

Algorithm 3.6 Minimum degree ordering

1: set up the graph $G^{(1)}$ for the matrix $A^{(1)} = A$
2: **for** k = 1:N **do**
3: select a node of $V^{(k)}$ with minimum degree and label it x_k
4: determine the graph $G^{(k+1)}$ obtained from $G^{(k)}$ by eliminating node x_k
5: **end for**

The minimum degree algorithm is quite costly – various cheaper variations such as as the approximate minimum degree algorithm of [1] are used in practice.

Example 3.6 (fill-in) The matrix $A \in \mathbb{R}^{900\times 900}$ of Example 2.2 with $n = 30$ has 5 non-zero entries per row. Different orderings have a considerable effect on the amount of fill-in (see Fig. 3.7): Whereas the lower part of A has $2,640$ non-zero entries, the Cholesky factor of A has $27,029$; using RCM ordering reduces this to $19,315$. Nested dissection ordering is even better with $13,853$ non-zero entries. The best is the approximate minimum degree ordering with $10,042$ non-zero entries. see Fig. 3.7. Calculations are done in MATLAB with: A = `gallery('poisson',30)`; p = `symrcm`(A); Lrcm = `chol`(A(p,p)); (approximate) minimum degree ordering is obtained by setting p = `symamd`(A). ∎

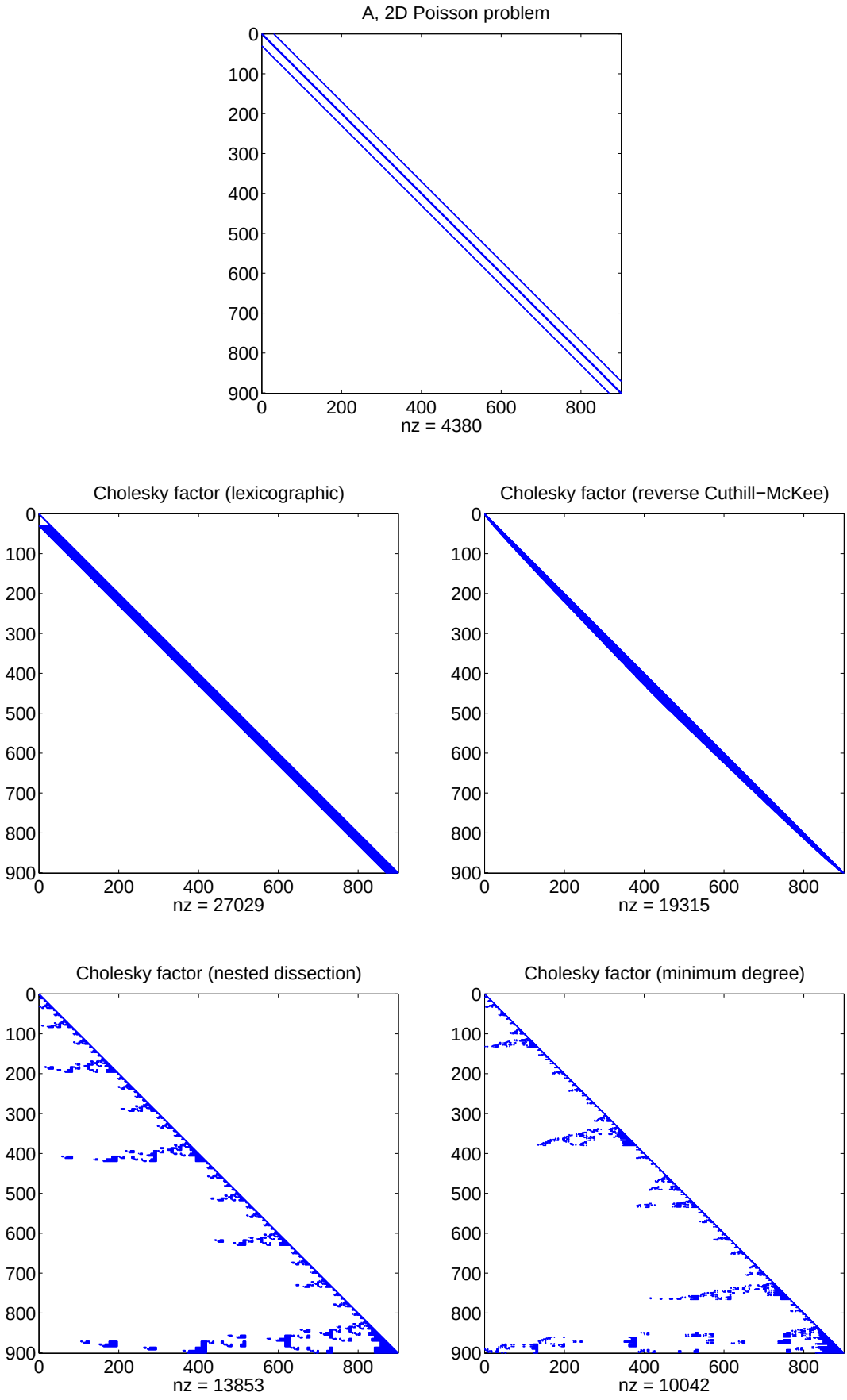


Figure 3.7: Fill-in for `gallery('poisson',30)` and various ordering strategies.

4 A Fast Poisson Solver Based on FFT

For systems with special structure, fast direct solutions can be designed on the basis of the Fast Fourier Transform (FFT), a fast implementation of the Discrete Fourier Transform (DFT), see [2]. Classical examples are systems represented by *circulant matrices*. Another important class are *Toeplitz matrices*, with a constant value along each diagonal. The FD matrices A for the 1D and 2D Poisson equation (Examples 2.1 and 2.2) are Toeplitz matrices; moreover, their eigenvalues and eigenvectors are explicitly known. This information can be exploited to design fast solvers for such special problems, which are often called *Fourier-spectral methods*.

In this section, the imaginary unit is denoted by $\boldsymbol{i}$, in contrast to i which is used for indexing.

4.1 The 1D case

The FD matrix from Example 2.1 reads

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \tag{4.1}$$

Since A is tridiagonal and SPD, the solution of[14] $Au = b$ can be performed in $\mathcal{O}(n)$ operations by straightforward elimination (Cholesky), i,e., with optimal computational effort.

Nevertheless, we consider another fast algorithm, a spectral method, mainly as preparation for the 2D case. This is based on the spectral decomposition (i.e, the orthonormal diagonalization)

$$A = Q\,D\,Q^{\mathrm{T}} \tag{4.2a}$$

with[15]

$$Q_{i,j} = \frac{(w_j)_i}{\|w_j\|_2} = c_n \sin(j\pi x_i), \qquad D_i = \lambda_i = 4\sin^2\frac{i\pi}{2(n+1)} \tag{4.2b}$$

Here $h = 1/(n+1)$ is the stepsize and $x_i = i\,h = i/(n+1)$ is the i-th grid point, see p. 6. Thus, the solution of the system $Au = b$ amounts to evaluation of

$$u = A^{-1}b = Q\,D^{-1}\,Q^{\mathrm{T}}b = Q\,D^{-1}\,Q\,b \tag{4.3}$$

(note that in this example, the matrix Q is not only orthonormal but also symmetric). Here, two matrix-vector multiplications $Q \cdot v$ are involved, with $\mathcal{O}(n^2)$ effort if performed in a naive manner. However, this can be realized in a fast way based on the *Discrete Sine Transform* related to FFT as explained in the sequel. This leads to an evaluation algorithm for (4.3) with effort $\mathcal{O}(n \log n)$ according to the computational complexity of FFT.

[14] Here we write u for the discrete solution approximating a continuous function $u(x)$.

[15] In (4.2b), the eigenvectors have been normalized such that Q is indeed an orthogonal matrix. Note that

$$c_n \equiv \|w_1\|_1 = \ldots = \|w_n\|_n = \sqrt{\frac{2}{n+1}}$$

The Discrete Sine Transform (DST).

Consider a vector $v = (v_1, \ldots, v_n)^{\mathrm{T}} \in \mathbb{R}^n$ and $\hat{v} = Q\,v = (\hat{v}_1, \ldots, \hat{v}_n)^{\mathrm{T}}$ with Q from (4.2b),

$$\hat{v}_k = \sum_{j=1}^{n} Q_{k,j}\, v_j = c_n \sum_{j=0}^{n} \sin(k\pi x_j)\, v_j, \quad k = 1 \ldots n \qquad \big(c_n = \sqrt{\tfrac{2}{n+1}}\,\big) \tag{4.4}$$

The transformation (4.4) is called the *Discrete Sine Transform* (DST);[16] the transformed vector $\hat{v}$ represents the expansion coefficients of v w.r.t. the orthonormal basis represented by Q. Here, $\sin(k\pi x_j)$ is the imaginary part of $\mathrm{e}^{\boldsymbol{i}k\pi x_j}$, hence

$$\hat{v}_k = c_n\, \mathrm{Im}\Big(\sum_{j=0}^{n} \mathrm{e}^{\boldsymbol{i}k\pi x_j}\, v_j \Big), \quad k = 1 \ldots n$$

In the usual notation from the DFT (Discrete Fourier Transform), we write

$$\sum_{j=0}^{n} \mathrm{e}^{\boldsymbol{i}k\pi x_j}\, v_j = \sum_{j=0}^{n} \omega^{kj}\, v_j, \quad \text{with} \quad \omega = \mathrm{e}^{\frac{\pi \boldsymbol{i}}{n+1}} = \mathrm{e}^{\frac{2\pi \boldsymbol{i}}{2n+2}} = (2n{+}2)\text{-th root of unity}$$

This a 'half part' from the sum appearing in the inverse DFT (IDFT) of the extended vector

$$V = (V_0, V_1, \ldots, V_{2n+1})^{\mathrm{T}} := (0, v_1, \ldots, v_n, 0, \ldots, 0)^{\mathrm{T}} \in \mathbb{R}^{2n+2}$$

Thus

$$\sum_{j=0}^{n} \omega^{kj}\, v_j = \sum_{j=0}^{2n+1} \omega^{kj}\, V_j$$

With a fast $\mathcal{O}(n \log n)$ `[i]fft` implementation of the [I]DFT available, for computing $\hat{v} = Q\,v$ we can proceed in the following way:

- Extend $v = (v_1, \ldots, v_n)$ to $V = (V_0, V_1, \ldots, V_{2n+1}) = (0, v_1, \ldots, v_n, 0, \ldots, 0)$
- Compute $\hat{V} = c_n\,(\hat{V}_0, \hat{V}_1, \ldots, \hat{V}_{2n+1})$
- Set $\hat{v} = \mathrm{Im}\,(\hat{V}_1, \ldots, \hat{V}_n)$

This corresponds to the version available in MATLAB: `dst` and its inverse `idst` for the direct and fast evaluation of (4.4).

Remark 4.1 As a word of warning, note that the usual implementation of such transformations do not use orthonormal scaling, This parallels the usual 'non-orthonormal representation' of classical Fourier basis functions, with the main purpose of avoiding computation of square roots like in c_n. For instance, in MATLAB, the functions `dst` and `idst` carry a different scaling. To use them correctly, you have to check this in the documentation. ∎

[16] The sine transform is identical with its inverse, a property which parallels the fact that Q is symmetric.

The related *Discrete Cosine Transform* (DCT) is, e.g., also used in image compression algorithms (e.g., underlying the classical `jpg` digital image format).

4.2 The 2D case

For the 1D case above, the spectral solver based on the DST is slightly less efficient than tridiagonal Cholesky. For the 2D case (Example 2.2), the situation is different. Here, the FD matrix takes the block-sparse form (see Fig. 2.1)

$$A = \begin{pmatrix} T & -I & & & & \\ -I & T & -I & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & -I & T & -I \\ & & & & -I & T \end{pmatrix} \in \mathbb{R}^{N\times N}, \quad N = n^2 \tag{4.5a}$$

with the tridiagonal $n\times n$ blocks

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{n\times n}, \qquad I = I_{n\times n} \tag{4.5b}$$

The bandwidth of the large matrix A is n, and when applying Cholesky decomposition a certain amount of fill-in is encountered, even when applying reordering strategies from Sec. 3.

For deriving an alternative direct solution method based on a 2-dimensional DST, we write vectors $v \in \mathbb{R}^N = \mathbb{R}^{n^2}$ in analogous partitioned form,

$$v = \begin{pmatrix} v^{(1)} \\ v^{(2)} \\ \vdots \\ \vdots \\ v^{(n-1)} \\ v^{(n)} \end{pmatrix} \in \mathbb{R}^N, \quad \text{with} \quad v^{(i)} = \begin{pmatrix} v_1^{(i)} \\ v_2^{(i)} \\ \vdots \\ v_{n-1}^{(i)} \\ v_n^{(i)} \end{pmatrix} \in \mathbb{R}^n \tag{4.6}$$

Then, the linear FD system $A\,u = b$ can be written in the block form

$$-u^{(i-1)} + Tu^{(i)} - u^{(i+1)} = b^{(i)}, \quad i = 1\ldots n \tag{4.7a}$$

where we set $u^{(0)} = u^{(n+1)} = 0$. From the 1D case we see that T has the spectral decomposition

$$T = Q\,S\,Q^{\mathrm{T}}$$

with (see (4.2b))

$$Q_{i,j} = \frac{(w_j)_i}{\|w_j\|_2} = c_n \sin(j\pi x_i), \quad S_i =: \sigma_i = 2 + 4\sin^2\frac{i\pi}{2(n+1)}, \quad i = 1\ldots n$$

With $y^{(i)} = Q\,u^{(i)}$, multiplying the systems (4.7a) by Q gives

$$-y^{(i-1)} + S\,y^{(i)} - y^{(i+1)} = Q\,b^{(i)}, \quad i = 1\ldots n \tag{4.7b}$$

with the diagonal matrix $S = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ replacing T from (4.7a).

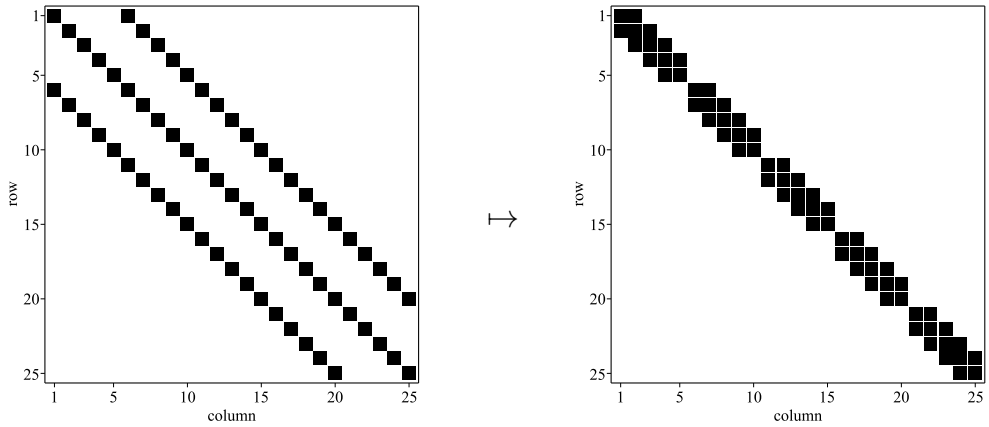


Figure 4.1: 2D Poisson system transformed via 1D-DST, followed by reordering of equations and unknowns.

This latter system decouples via reordering the unknowns: Let us permute vectors $v \in \mathbb{R}^N = \mathbb{R}^{n^2}$ from (4.6) according to

$$v \;\mapsto\; \hat{v} = \begin{pmatrix} \tilde{v}^{(1)} \\ \tilde{v}^{(2)} \\ \vdots \\ \vdots \\ \tilde{v}^{(n-1)} \\ \tilde{v}^{(n)} \end{pmatrix}, \quad \text{with} \quad \tilde{v}^{(i)} = \begin{pmatrix} v_i^{(1)} \\ v_i^{(2)} \\ \vdots \\ v_i^{(n-1)} \\ v_i^{(n)} \end{pmatrix} \in \mathbb{R}^n$$

i.e., by switching from lexicographic row-wise to lexicographic column-wise ordering of the unknowns of the computational grid. Then, the set of linear systems (4.7b) transforms into a set of independent tridiagonal systems of dimension $n \times n$,

$$\tilde{S}\,\tilde{y}^{(i)} = Q\,b^{(i)}, \quad i = 1 \ldots n, \qquad \text{with} \quad \tilde{S} = \begin{pmatrix} \sigma_1 & -1 & & & \\ -1 & \sigma_2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & \sigma_{n-1} & -1 \\ & & & -1 & \sigma_n \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{4.7c}$$

This corresponds to a permutation of the system represented by (4.7b); see Fig. 4.1 for a visualization The right-hand sides $Q\,b^{(i)}$ can be evaluated via the 1D-DST in $n \cdot \mathcal{O}(n \log n)$ operations, and the resulting tridiagonal systems are of the same type as for the 1D Poisson case and can be solved by tridiagonal Cholesky in $n \cdot \mathcal{O}(n)$ operations. Finally, the solution vectors $\tilde{y}^{(i)}$ are recombined into the original order, and the resulting $y^{(i)}$ are transformed back using 1D-DST to yield the solution components $u^{(i)} = Q\,y^{(i)}$.

If, in the solution step, we replace Cholesky by the 1D spectral method, the outcome is a pure 2D spectral method applied to $A = Q\,D\,Q^{\mathrm{T}}$, realized by a 2D-DST transform.

The overall computational and memory effort for this fast Poisson solver is $\mathcal{O}(n^2 \log n) = \mathcal{O}(N \log N)$, nearly optimal in the problem dimension $N = n^2$. However, one should keep in mind that the reordering steps may be nontrivial to implement on parallel computer architectures, since global communication takes place for larger problems. (This challenge is also typical for most FFT-based computational techniques.)

In MATLAB, an implementation of this solver is available via the function `poisolv` contained in the Partial Differential Equations Toolbox. 3D Poisson solvers of a similar type can also be designed. See also [12] for a discussion of related methods, e.g., for more general problem geometries.

5 Basic Iterative Methods

5.1 Linear iterative methods, convergence analysis

We consider solving linear systems of the form[17]

$$A\,x = b, \quad A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^n \tag{5.1}$$

and assume that A is invertible. Many iterative methods[18] for solving (5.1) have the form of a *fixed point iteration*

$$x_{k+1} = \Phi(x_k; b), \quad k = 0, 1, 2, \ldots \tag{5.2}$$

with some mapping Φ, starting from an initial value x_0. Solutions x_* of equation $x_* = \Phi(x_*; b)$ are called *fixed points* of (5.2). Since the iteration mapping Φ is kept fixed, this is also called a *stationary iteration*.

Definition 5.1 *The fixed point iteration* (5.2) *is said to be*

- ***consistent*** *with the invertible matrix A, if for every b the solution $x_* = A^{-1}b$ is a fixed point of* (5.2)*;*
- ***convergent****, if for every b there exists a vector x_* such that for every starting vector x_0 the sequence $(x_k) = (x_k)_{k=0}^{\infty}$ defined by* (5.2) *converges to x_*;*[19]
- ***linear****, if Φ is an affine mapping of the form $\Phi(x; b) = Mx + Nb$, i.e.,*

$$x_{k+1} = Mx_k + Nb \tag{5.3}$$

M is called the ***iteration matrix****.*[20]

The matrices M, N from (5.3) need to satisfy certain conditions in order to ensure consistency with a given matrix A. We have:

Lemma 5.1 *Let A be invertible. Then the fixed point iteration* (5.3) *is consistent with A if and only if*

$$M = I - NA \quad \Leftrightarrow \quad (I - M)^{-1}N = A^{-1}$$

Proof: With $x = Ab$ the fixed point equation is equivalent to $x = Mx + NAx$ for arbitrary x, and this is equivalent to $I = M + NA$. □

We will see that the asymptotic convergence behavior of the iteration (5.3) for $k \to \infty$ is determined by the spectrum of the iteration matrix M. We start with the following special (linear) variant of the Banach Fixed Point Theorem:

Theorem 5.1 *Let $\rho(M) < 1$. Then there exists a unique fixed point x_* of the iteration* (5.3)*,*

$$x_* = (I - M)^{-1}Nb \in \mathbb{R}^n$$

and the iteration converges to x_* for any starting value $x_0 \in \mathbb{R}^n$.

[17] The results of this section are easily generalized to the complex case $A \in \mathbb{C}^{n\times n}$.

[18] See also [2, Ch. 7].

[19] We stress that we require a) convergence and b) a unique limit x_* – independent of the starting vector x_0.

[20] Later we will see that in typical situations the matrix N should be invertible; N^{-1} is then called a *preconditioner*.

Proof: Since $\rho(M) < 1$, we conclude $1 \notin \sigma(M)$, i.e., there exists a unique fixed point $x_* = (I - M)^{-1} N b \in \mathbb{R}^n$. By e_k we denote the error of the k-th iterate, $e_k = x_k - x_*$. By subtracting the equations $x_{k+1} = Mx_k + Nb$ and $x_* = Mx_* + Nb$, we obtain a recursion for the errors e_k:

$$e_{k+1} = Me_k = M^2 e_{k-1} = \cdots = M^{k+1} e_0 \tag{5.4}$$

By Theorem 1.1 we can find a norm $\|\cdot\|_\varepsilon$ on $\mathbb{R}^n$ such that $\|M\|_\varepsilon \le \rho(M) + \varepsilon < 1$. In this norm, we obtain from (5.4)

$$\|e_{k+1}\|_\varepsilon = \|M^{k+1} e_0\|_\varepsilon \le \|M^{k+1}\|_\varepsilon \|e_0\|_\varepsilon \le \|M\|_\varepsilon^{k+1} \|e_0\|_\varepsilon$$

Since $\|M\|_\varepsilon < 1$, we conclude that the sequence $(e_k)_{k=0}^\infty$ converges to 0 in the norm $\|\cdot\|_\varepsilon$. Since all norms on the finite-dimensional space $\mathbb{R}^n$ are equivalent, we conclude that the sequence $(x_k)_{k=0}^\infty$ converges to x_* in any norm. □

Corollary 5.1 *Let the fixed point iteration* (5.3) *be consistent with the invertible matrix A, and $\rho(M) < 1$. Then for every starting value x_0 the sequence $(x_k)_{k=0}^\infty$ converges to the solution x_* of $Ax_* = b$.*

Conversely, it is straightforward to show that for $\rho(M) \ge 1$ the iteration does not converge or converges to different fixed points depending on x_0.

Remark 5.1 The natural interpretation of the matrix N is that of an *approximate inverse*, i.e., $N \approx A^{-1}$ in some specific sense – but is essential that N (and M) can be efficiently evaluated. $N = A^{-1}$ (together with $M = 0$) would solve the problem in one step (direct solution). For $N \approx A^{-1}$ the matrix $M = I - NA$ is expected be 'small', as required.

The invertibility of N is necessary for convergence: Otherwise the matrix $I - M = NA$ would have a nontrivial kernel and eigenvalue $\lambda = 0$, which contradicts the convergence condition $1 \notin \sigma(M)$.

A fixed point iteration (5.3) consistent with A can be formulated in three (equivalent) ways:

(i) ***First normal form*** (5.3): $x_{k+1} = Mx_k + Nb$, with $M = I - NA$.

(ii) ***Second normal form:*** $x_{k+1} = x_k + N(b - Ax_k)$. This shows that the correction added to x_k is a *linear image of the residual*

$$b - Ax_k =: r_k$$

(iii) ***Third normal form:*** $W(x_{k+1} - x_k) = b - Ax_k$, with $W = N^{-1}$; W is an approximation to A. This shows that the correction $\delta_k := x_{k+1} - x_k$ is obtained by solving the linear system $W\delta_k = r_k$.[21] ■

The third normal form reveals the heuristic idea behind a consistent iteration. For $W \approx A$, a sufficiently good approximation to A, we may expect

$$Wx_* - Wx_k \approx Ax_* - Ax_k = b - Ax_k = r_k$$

which motivates the choice for x_{k+1}.

For the error $e_k = x_k - x_*$ we have

$$e_{k+1} = M e_k \tag{5.5}$$

and the residual $r_k = b - Ax_k$ satisfies $r_k = -Ae_k$. Thus, also $r_{k+1} = Mr_k$ holds. If the iteration matrix $M = I - NA$ satisfies $\rho(M) < 1$, then we call $\rho(M)$ the *(asymptotic) convergence factor* of the iteration (5.3). The expression $-\log_{10} \rho(M)$ is called the *(asymptotic) convergence rate.* This measures the number of correct decimal digits which are gained in each iteration step (in an asymptotic sense).

[21] Note the formal analogy to a [quasi-]Newton method for solving a nonlinear system $F(x) = b$, with residual $b - F(x_k)$.

Remark 5.2 The convergence result from Theorem 5.1 supports the expectation that the error is reduced by a factor $q \in (0,1)$ in each step of the iteration. However, in general this is true only in an asymptotic, or 'mean' sense: Consider

$$q = \limsup_{k\to\infty} \left(\frac{\|e_k\|}{\|e_0\|} \right)^{1/k}$$

where $\|\cdot\|$ is a norm which we are interested in, since (asymptotically) the error is reduced by a factor q in each step. From (5.4) we obtain

$$\|e_k\| = \|M^k e_0\| \le \|M^k\| \, \|e_0\|$$

and therefore

$$q = \limsup_{k\to\infty} \left(\frac{\|e_k\|}{\|e_0\|} \right)^{1/k} \le \limsup_{k\to\infty} \left(\frac{\|M^k\| \, \|e_0\|}{\|e_0\|} \right)^{1/k} = \limsup_{k\to\infty} \|M^k\|^{1/k} = \rho(M)$$

by Theorem 1.1. We note that the precise choice of norm is immaterial for the asymptotic convergence behavior.[22] The meaning of the asymptotic convergence rate $-\log_{10} \rho(M)$ is the number of steps required to reduce (*asymptotically*) the error by a factor 10. ∎

Remark 5.3 In general, for a given norm, e.g. for $\|\cdot\|_2$, the behavior of the error in a single step may have nothing to do with the size of $\rho(M) < 1$. For (highly) non-normal M, $\|M\|_2$ may be > 1 (even $\gg 1$). In other words: The spectral radius $\rho(M)$ only determines the *asymptotic* convergence rate in general. A trivial but instructive example is the Jordan block

$$M = \begin{pmatrix} 0 & m & & & & \\ & 0 & m & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & m \\ & & & & & 0 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

Here, $\rho(M) = 0$, $M^n = 0$ (nilpotency), but $\|M\|_1 = \|M\|_2 = \|M\|_\infty = |m|$.

In such a situation the convergence behavior of the stationary iteration will be non-monotonic or erratic, in particular in the initial phase. ∎

5.2 Splitting methods

Early iterative methods (including the classical Jacobi and Gauss-Seidel methods to be discussed in the sequel) are *splitting methods* for solving (5.1). A splitting method is determined by writing

$$A = G - H$$

and using the approximate inverse $N = G^{-1} \approx A^{-1}$. Equivalently, this is obtained by rewriting equation (5.1) as

$$Ax = b \quad \Leftrightarrow \quad Gx = Hx + b \quad \Leftrightarrow \quad x = G^{-1}Hx + G^{-1}b$$

which has fixed point form (5.3) and is, by construction, consistent with the matrix A. The corresponding fixed point iteration reads

$$x_{k+1} = \underbrace{G^{-1}H}_{=M} x_k + \underbrace{G^{-1}}_{=N} b = (I - G^{-1}A)\, x_k + G^{-1} b = x_k + G^{-1}(b - Ax_k)$$

In practice, the approximate inverse $N = G^{-1}$ is, of course, not explicitly computed; only the action $z \mapsto G^{-1} z$ needs to be realized computationally. For efficient methods, the action $z \mapsto G^{-1} z$, or equivalently, the solution of the correction equation $G\delta_k = r_k$, needs to be 'cheap'.

[22] We refer to [19, Sec. 4.2] for a more detailed discussion.

Richardson, Jacobi, Gauss-Seidel.

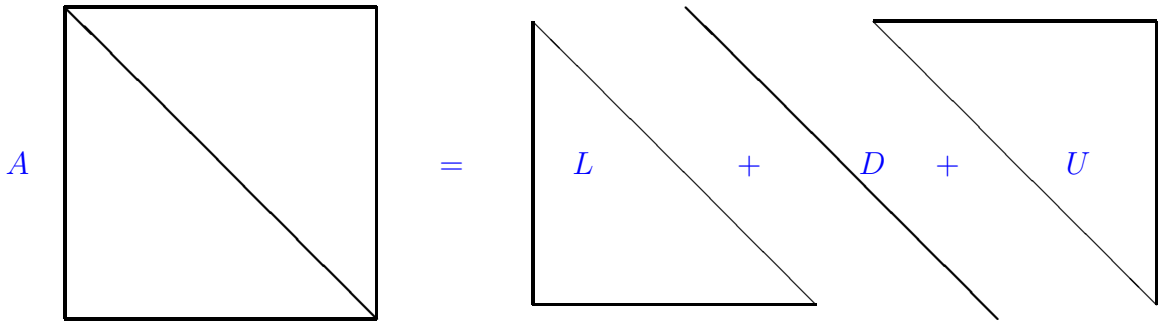
• ***Richardson:*** We start with a trivial, basic case: $A = I - (I - A)$. This leads to the *Richardson iteration:*

$$x_{k+1} = x_k + (b - A x_k) \tag{5.6}$$

Here, $N = I$ is taken as an 'approximate inverse' for A.

Remark 5.4 We note that the second normal form of a (consistent) linear iteration shows that every linear iteration for a system $Ax = b$ can be interpreted as the Richardson iteration applied to the transformed problem $NAx = Nb$. In this interpretation we call N (or N^{-1}) a *preconditioner*; we need $N \approx A^{-1}$ to expect convergence of the preconditioned Richardson iteration. ∎

In the following we throughout write $A = D + L + U$, where D, L, U are the diagonal, the (strictly) lower, and the (strictly) upper part of A, respectively.



• ***Jacobi:*** Choose $G = D$ and $H = -(L + U)$. The approximate inverse is $N = D^{-1}$, and the Jacobi iteration is given by

$$x_{k+1} = x_k + D^{-1}(b - A x_k) \quad \Leftrightarrow \quad D x_{k+1} = D x_k + (b - A x_k) \tag{5.7a}$$

This means that, for $i = 1 \dots n$, the i-the equation is solved for the i-th component, where the other components take the values from the previous iteration step. The inversion of the diagonal matrix D (which must be invertible) is trivial. In component notation, the Jacobi iteration can be written as

$$(x_{k+1})_i = (x_k)_i + \frac{1}{A_{i,i}} \Big(b_i - \sum_{j=1}^{n} A_{i,j} (x_k)_j \Big), \quad i = 1 \dots n \tag{5.7b}$$

• ***Gauss-Seidel:*** We choose $G = (L + D)$ and $H = -U$. The approximate inverse is $N = (L + D)^{-1}$, which leads to the forward Gauss-Seidel iteration

$$x_{k+1} = x_k + (L + D)^{-1}(b - A x_k) \quad \Leftrightarrow \quad D x_{k+1} = D x_k + (b - L x_{k+1} - (D + U) x_k) \tag{5.8a}$$

Provided that D is invertible, the action $z \mapsto (L + D)^{-1} z$ is easily realized by forward substitution since $L + D$ is lower triangular. In component notation, the Gauss-Seidel iteration can be written as

$$(x_{k+1})_i = (x_k)_i + \frac{1}{A_{i,i}} \Big(b_i - \sum_{j=1}^{i-1} A_{i,j} (x_{k+1})_j - \sum_{j=i}^{n} A_{i,j} (x_k)_j \Big), \quad i = 1 \dots n \tag{5.8b}$$

The method works like Jacobi but, in the inner iteration, the updates $(x_{k+1})_i$ immediately replace the old values $(x_k)_i$.

Completely analogous to the forward Gauss-Seidel iteration is the backward Gauss-Seidel iteration, which corresponds to the splitting $G = (D + U)$, $H = -L$, and thus

$$x_{k+1} = x_k + (D+U)^{-1}(b - Ax_k), \qquad \text{i.e.,} \quad N = (D+U)^{-1} \tag{5.9}$$

Remark 5.5 The Richardson and Jacobi methods are independent of the numbering of the unknowns; the forward and backward Gauss-Seidel methods are not. E.g., the backward Gauss-Seidel method is obtained from the forward version by reversing the numbering of the unknowns. The Jacobi method has much more potential for parallelization. On the other hand, Gauss-Seidel is more economic concerning storage and, as we will see, usually has more favorable convergence properties. ∎

Damped Richardson, Jacobi, SOR, SSOR.

Each of the above methods can be modified by *damping*, where a given approximate inverse N is replaced by ωN,

$$x_{k+1} = x_k + \omega N(b - Ax_k) \tag{5.10}$$

Here, $\omega \in \mathbb{R}$ is the damping (or: relaxation) parameter. E.g., for $N = I$, we obtain the *damped Richardson method,*

$$x_{k+1} = x_k + \omega\,(b - Ax_k) \tag{5.11}$$

and for $N = D^{-1}$ we obtain the *damped Jacobi method*,

$$x_{k+1} = x_k + \omega\, D^{-1}(b - Ax_k) \tag{5.12}$$

The so-called SOR method (**s**uccessive **o**ver**r**elaxation[23]) is obtained in a slightly different manner, by introducing a relaxation factor ω in the component formulation (5.8b) of the Gauss-Seidel method,

$$(x_{k+1})_i = (x_k)_i + \frac{\omega}{A_{i,i}}\Big(b_i - \sum_{j=1}^{i-1} A_{i,j}\,(x_{k+1})_j - \sum_{j=i}^{n} A_{i,j}\,(x_k)_j\Big), \quad i = 1\ldots n \tag{5.13a}$$

In matrix notation, this reads

$$\begin{aligned} & Dx_{k+1} = Dx_k + \omega(b - Lx_{k+1} - (D+U)x_k) \\ \Leftrightarrow \quad & Dx_{k+1} = (D+\omega L)x_k + \omega(b - Lx_{k+1} - Ax_k) \\ \Leftrightarrow \quad & x_{k+1} = x_k + \omega(D+\omega L)^{-1}(b - Ax_k), \quad \text{i.e.,} \quad N = \omega(D+\omega L)^{-1} \end{aligned} \tag{5.13b}$$

Of course, instead of the forward Gauss-Seidel method, one could start from the backward Gauss-Seidel method – then, the matrix U replaces L in (5.13b).

A disadvantage of the Gauss-Seidel and, more generally, the SOR methods is that the iteration matrix M is not symmetric even if the original matrix A is symmetric. This can be overcome by defining a symmetric version, the SSOR (Symmetric SOR) by applying first a step of SOR based on the forward Gauss-Seidel method and then an SOR step based on the backward Gauss-Seidel method:

$$\begin{aligned} x_{k+\frac{1}{2}} &= x_k + \omega(D+\omega L)^{-1}(b - Ax_k), \\ x_{k+1} &= x_{k+\frac{1}{2}} + \omega(D+\omega U)^{-1}(b - Ax_{k+\frac{1}{2}}) \end{aligned} \tag{5.14a}$$

A short calculation shows

[23] More precisely: the choice $\omega > 1$ is called *overrelaxation* whereas the choice $\omega < 1$ corresponds to *underrelaxation*. In the examples we will consider, overrelaxation is advantageous.

$$x_{k+1} = x_k + \omega(2-\omega)(D+\omega U)^{-1} D (D+\omega L)^{-1}(b - Ax_k) \tag{5.14b}$$

with symmetric iteration matrix for symmetric A, i.e., for $U = L^{\mathrm{T}}$.

If A is SPD, then $D > 0$ and the SSOR iteration matrix is also SPD:

$$\big((D+\omega L^{\mathrm{T}})^{-1} D (D+\omega L)^{-1} x, x\big) = \big(D(D+\omega L)^{-1}x, (D+\omega L)^{-1}x\big) > 0 \quad \text{for} \quad x \neq 0$$

We collect the iteration matrices M for these methods:

method	iteration matrix $M = I - NA$	approximate inverse N
damped Richardson	$M^{Rich} = I - \omega A$	ωI
damped Jacobi	$M_\omega^{Jac} = I - \omega D^{-1}A$	ωD^{-1}
forward Gauss-Seidel	$M^{GS} = I - (D+L)^{-1}A$	$(D+L)^{-1}$
SOR	$M_\omega^{SOR} = I - \omega(D+\omega L)^{-1}A$	$\omega(D+\omega L)^{-1}$
SSOR	$M_\omega^{SSOR} = I - \omega(2-\omega)(D+\omega U)^{-1}D(D+\omega L)^{-1}A$	$\omega(2-\omega)(D+\omega U)^{-1}D(D+\omega L)^{-1}$

Jacobi, Gauss-Seidel, SOR and SSOR in the SPD case.

For the case where A is SPD, a fairly general convergence theory can be established for these methods. First we collect some properties of symmetric and/or SPD matrices.

For a symmetric matrix $A \in \mathbb{R}^{n\times n}$ we write $A \geq 0$ if A is positive semidefinite, and $A > 0$ if A is positive definite. For two symmetric matrices $A, B \in \mathbb{R}^{n\times n}$ we write $A \geq B$ if $A - B \geq 0$; we write $A > B$ if $A - B > 0$.

Lemma 5.2 *Let $A, B \in \mathbb{R}^{n\times n}$ be symmetric. Then:*

(i) $A > [\geq]\, 0 \quad \Leftrightarrow \quad C^{\mathrm{T}}AC > [\geq]\, 0$ *for all invertible* $C \in \mathbb{R}^{n\times n}$.

(ii) $A > [\geq]\, B \quad \Leftrightarrow \quad C^{\mathrm{T}}AC > [\geq]\, C^{\mathrm{T}}BC$ *for all invertible* $C \in \mathbb{R}^{n\times n}$.

(iii) $A, B > [\geq]\, 0 \quad \Rightarrow \quad A + B > [\geq]\, 0$

(iv) $\lambda I < A < \Lambda I \quad \Leftrightarrow \quad \sigma(A) \subset (\lambda, \Lambda)$ *(analogous assertion for* $\leq$*)*

(v) $A > [\geq]\, B > 0 \quad \Leftrightarrow \quad 0 < A^{-1} < [\leq]\, B^{-1}$

Proof: We will only prove (v), the remaining cases being simple (exercise). If you, e.g., prove (i), observe the meaning of this assertion (a simple 'change of coordinates'-argument).

For (v), we will only show that $A \geq B > 0$ implies $B^{-1} \leq A^{-1}$. Since B is SPD, we can define the SPD matrix $B^{-\frac{1}{2}}$. Then, by (i), we infer that $A \geq B$ implies $X := B^{-\frac{1}{2}} A B^{-\frac{1}{2}} \geq I$. Hence, all eigenvalues of the symmetric matrix X are ≥ 1 (see (iv)). Thus, all eigenvalues of the symmetric matrix X^{-1} are all ≤ 1, i.e., $I \geq X^{-1} = B^{\frac{1}{2}} A^{-1} B^{\frac{1}{2}}$. Multiplying both sides by the symmetric matrix $B^{-\frac{1}{2}}$ and recalling (i) gives $B^{-1} \geq A^{-1}$. □

For an SPD matrix $A = Q\Lambda Q^{\mathrm{T}} > 0$ with Q orthogonal and $\Lambda > 0$ diagonal, the square root $A^{\frac{1}{2}}$,

$$A^{\frac{1}{2}} = Q\Lambda^{\frac{1}{2}} Q^{\mathrm{T}}$$

and its inverse $A^{-\frac{1}{2}} = Q\Lambda^{-\frac{1}{2}} Q^{\mathrm{T}}$ are also SPD. Furthermore,[24] each SPD matrix A defines an 'energy product' and associated energy norm,

[24] Vectors and matrices are understood in original coordinates in $\mathbb{R}^n$. On could also use transformed coordinates; i.e., $x = A^{-\frac{1}{2}} A^{\frac{1}{2}} x \in \mathbb{R}^n$ has coordinate vector $\hat{x} = A^{\frac{1}{2}} x$ with respect to the canonical A-orthogonal basis $A^{-\frac{1}{2}}$, and a linear mapping represented by $M \in \mathbb{R}^{n\times n}$ is represented by $\hat{M} = A^{\frac{1}{2}} M A^{-\frac{1}{2}}$ in these new coordinates. This notation is used in (5.15a),(5.15b) and in Exercise 5.1.

$$(x,y)_A = (Ax,y) = (\hat{x},\hat{y}), \qquad \|x\|_A = (Ax,x)^{\frac{1}{2}} = \|\hat{x}\|_2 \tag{5.15a}$$

where $\hat{x} = A^{\frac{1}{2}}x,\ \hat{y} = A^{\frac{1}{2}}y$. The corresponding matrix norm is

$$\|M\|_A = \|A^{\frac{1}{2}}MA^{-\frac{1}{2}}\|_2 = \|\hat{M}\|_2 \tag{5.15b}$$

We note some further properties of general inner products (energy products) $(u,u)_A = (Au,u)$, generalizing well-known identities from the Euclidean case:

Exercise 5.1

a) Let $A \in \mathbb{R}^{n\times n}$ be SPD. Show: The adjoint M^A of a matrix M with respect to the $(\cdot,\cdot)_A$ inner product is given by

$$M^A = A^{-1}M^{\mathrm{T}}A \tag{5.16}$$

Note that M is A-selfadjoint iff $M^{\mathrm{T}}A = AM \Leftrightarrow A^{-\frac{1}{2}}M^{\mathrm{T}}A^{\frac{1}{2}} = A^{\frac{1}{2}}MA^{-\frac{1}{2}}$, i.e., if $\hat{M} = A^{\frac{1}{2}}MA^{-\frac{1}{2}}$ is symmetric.

b) An 'A-orthogonal' pair of vectors x, y, satisfying

$$(x,y)_A = (Ax,y) = (\hat{x},\hat{y}) = 0$$

is also called *[A-]conjugate*. A-orthogonality of a linear mapping represented by a matrix Q is defined in the usual way:

$$(Qx,Qy)_A \equiv (x,y)_A, \quad \text{i.e.,} \quad Q^{\mathrm{T}}AQ = A$$

Show that this is equivalent to $Q^AQ = I$.

c) A matrix P satisfying

$$P^{\mathrm{T}}AP = I, \quad \text{i.e., the columns } p_i \text{ of } P \text{ are pairwise } A\text{-conjugate: } (p_i,p_j)_A = (Ap_i,p_j) = (\hat{p}_i,\hat{p}_j) = \delta_{i,j}$$

is called [A-]conjugate. Show that this is equivalent to $P^AP = A^{-1}$. ∎

The diagonal part D of an SPD matrix A satisfies $D > 0$ because $D_{i,i} = (Ae_i,e_i) > 0$. In the following theorem, a stronger property in the sense of 'diagonal dominance' is involved.

Theorem 5.2 *Let $A \in \mathbb{R}^{n\times n}$ be SPD. Then,*

$$\rho(M_\omega^{Jac}) < 1 \quad \Leftrightarrow \quad 0 < A < \tfrac{2}{\omega}D$$

Proof: $M_\omega^{Jac} = I - \omega D^{-1}A$ is not symmetric but A-selfadjoint (see Exercise 5.1):

$$\left(M_\omega^{Jac}\right)^A = A^{-1}\left(M_\omega^{Jac}\right)^{\mathrm{T}}A = A^{-1}(I - \omega AD^{-1})A = I - \omega D^{-1}A = M_\omega^{Jac}$$

Equivalently, $A^{\frac{1}{2}}M_\omega^{Jac}A^{-\frac{1}{2}} = I - \omega A^{\frac{1}{2}}D^{-1}A^{\frac{1}{2}}$ is symmetric with real spectrum. We have

$$\sigma(M_\omega^{Jac}) = \sigma(A^{\frac{1}{2}}M_\omega^{Jac}A^{-\frac{1}{2}}) = \sigma(I - \omega A^{\frac{1}{2}}D^{-1}A^{\frac{1}{2}})$$

From Lemma 5.2, (iv) we obtain

$$\rho(M_\omega^{Jac}) < 1 \ \Leftrightarrow\ -I < I - \omega A^{\frac{1}{2}}D^{-1}A^{\frac{1}{2}} < I \ \Leftrightarrow\ 0 < \omega A^{\frac{1}{2}}D^{-1}A^{\frac{1}{2}} < 2I$$
$$\Leftrightarrow\ 0 < \omega D^{-1} < 2A^{-1} \ \Leftrightarrow\ 0 < A < \tfrac{2}{\omega}D$$

which concludes the proof. □

Under the condition of Theorem 5.2, the damped Jacobi iteration is contractive in the energy norm: For the error $e_k = x_k - x_*$ we have

$$\|A^{\frac{1}{2}} e_{k+1}\|_2 = \|A^{\frac{1}{2}} M_\omega^{Jac} e_k\|_2 = \|A^{\frac{1}{2}} M_\omega^{Jac} A^{-\frac{1}{2}} A^{\frac{1}{2}} e_k\|_2 < \|A^{\frac{1}{2}} M_\omega^{Jac} A^{-\frac{1}{2}}\|_2 \|A^{\frac{1}{2}} e_k\|_2$$

or equivalently,

$$\|e_{k+1}\|_A < \underbrace{\|M_\omega^{Jac}\|_A}_{<1} \|e_k\|_A$$

This strictly contractive behavior does, in general, not hold in the Euclidean norm.[25]

Theorem 5.2 shows that, due to $D > 0$, choosing the relaxation parameter $\omega > 0$ sufficiently small (depending on A) will guarantee that the damped Jacobi method converges for every SPD matrix A. Note, however, that an estimate for the *rate of convergence*, or contractivity factor, i.e., for $\rho(M_\omega^{Jac}) = \|M_\omega^{Jac}\|_A$ is not available from the theorem. For $\omega \downarrow 0$ we have $M_\omega^{Jac} = I - \omega D^{-1} A \to I$; in the limit, convergence is assured but must be expected to be slow. We return to the question of convergence rates later on.

The regime of damping parameters for which the damped Jacobi method converges depends on the problem. This is in contrast to the SOR method, which converges for *arbitrary* SPD matrices for any value of the relaxation parameter $\omega \in (0, 2)$, see Theorem 5.3 below.

Exercise 5.2 Let A, N be SPD and consider the iteration $x_{k+1} = x_k + N(b - Ax_k)$. Let $M = I - W^{-1}A$ ($W = N^{-1}$) be the iteration matrix. Assume that W is SPD and even satisfies

$$2W > A > 0 \tag{5.17a}$$

a) Show: $\rho(M) = \|M\|_A < 1$.

b) Show: If for some $0 < \lambda \le \Lambda$ there holds

$$0 < \lambda W \le A \le \Lambda W$$

then $\sigma(M) \subseteq [1 - \Lambda, 1 - \lambda]$, and thus,

$$\rho(M) \le \max\{|1 - \lambda|, |\Lambda - 1|\}$$

c) Show: for $\omega \in (0, 1]$, the damped SSOR method (5.14b) converges for all SPD matrices A.

d) Show that a) can be weakened in the following way: We do not require that N, W are symmetric, but we only assume that W is positive definite, i.e., $2\,\mathrm{Re}\, W = W + W^{\mathrm{T}} > 0$, and replace condition (5.17a) by

$$W + W^{\mathrm{T}} > A > 0 \tag{5.17b}$$

Then, $\rho(M) = \|M\|_A < 1$.

Hint: Use (5.16) and express $N + N^{\mathrm{T}}$ by means of $W + W^{\mathrm{T}}$. ∎

Theorem 5.3 *Assume $\omega \in \mathbb{R}$ and $A \in \mathbb{R}^{n\times n}$ SPD. Then,*

$$\rho(M_\omega^{SOR}) = \|M_\omega^{SOR}\|_A < 1 \quad \textit{for} \quad \omega \in (0, 2)$$

[25] Also for other classes of iterative methods to be discussed later, energy estimates are often more natural (and easier to derive) in the SPD case.

Proof: We make use of Exercise 5.2, d). For the SOR method we have

$$W = N^{-1} = \omega^{-1}D + L \quad \text{and} \quad W + W^T = 2\omega^{-1}D + L + L^{\mathrm{T}}$$

Thus,

$$W + W^T - A = 2\omega^{-1}D + L + L^{\mathrm{T}} - D - L - L^{\mathrm{T}} = (2\omega^{-1} - 1)\, D$$

which is > 0 iff $\omega \in (0, 2)$. □

Remark 5.6 An alternative more technical proof, based on direct investigation of the spectrum of M_ω^{SOR}, reveals that $\rho(M_\omega^{SOR}) < 1$ *if and only if* $\omega \in (0, 2)$. ∎

Convergence of SSOR(ω).

Exercise 5.2 shows that SSOR(ω) converges for $\omega \in (0, 1]$. In fact, it converges for all $\omega \in (0, 2)$ as the following exercise shows.

Exercise 5.3 Let A be SPD and denote by $M_\omega^{SOR} = I - \omega\,(D + \omega\, L)^{-1}A$ the iteration matrix of the (forward) SOR method and by $\bar{M}_\omega^{SOR} = I - \omega\,(D + \omega\, L^{\mathrm{T}})^{-1}A$ the iteration matrix of the backward SOR method. The iteration matrix $M_\omega^{SSOR} = \bar{M}_\omega^{SOR}\, M_\omega^{SOR}$ of the damped SSOR method is given by

$$M_\omega^{SSOR} = I - \omega(2 - \omega)(D + \omega\, L^{\mathrm{T}})^{-1} D\, (D + \omega\, L)^{-1} A$$

a) Show: $\bar{M}_\omega^{SOR}$ is the adjoint of M_ω^{SOR} with respect to the $(\cdot, \cdot)_A$ inner product, i.e., $\bar{M}_\omega^{SOR} = (M_\omega^{SOR})^A$. Conclude that $\sigma(M_\omega^{SSOR}) \subseteq \mathbb{R}_0^+$, i.e., the spectrum is non-negative.

b) Using b), show: $\|M_\omega^{SSOR}\|_A = \|M_\omega^{SOR}\|_A^2$. ∎

Exercise 5.3 shows that for an SPD matrix A SSOR(ω) converges (even monotonically in the energy norm) if SOR(ω) converges monotonically in the energy norm. Due to Theorem 5.3 this is indeed the case for $\omega \in (0, 2)$.

Jacobi and Gauss-Seidel for diagonally dominant matrices.

Definition 5.2 ([ir]reducible matrix) *A matrix $A \in \mathbb{R}^{n\times n}$ is called **reducible** if there exists a permutation of the indices, i.e., a permutation matrix P such that*

$$P^T A\, P = \begin{pmatrix} A_1 & 0 \\ A_3 & A_2 \end{pmatrix} \tag{5.18}$$

*where A_1 and A_2 are non-empty square subblocks. Matrices that cannot be transformed in this way a called **irreducible**.*

Reducibility of A means that, after an appropriate reordering, the system $Ax = b$ can be split into two subsystems where the subsystem involving A_1 can be solved independently. On the other hand, irreducibility means that the system is 'fully coupled'.

Definition 5.3 (directed adjacency graph) *(cf. Sec. 3) Let $A \in \mathbb{R}^{n\times n}$. The graph*

$$G = G(A) = (V, E) \quad \textit{with} \quad V = \{1, \ldots, n\} \quad \textit{and} \quad E = \{\,(i, j) : A_{i,j} \neq 0\,\}$$

is called the directed adjacency graph of A.

*The index j is said to be **adjacent** to i if $(i, j) \in E$, i.e., if variable x_j is present in equation i.*

*The index j is said to be **connected to** i if there exists a chain of indices $i_1, i_2, \ldots, i_k$ such that $(i, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k), (i_k, j) \in E$.*

Lemma 5.3 *A matrix $A \in \mathbb{R}^{n\times n}$ is irreducible iff each index $j \in \{1,\dots,n\}$ is connected to each $i \in \{1,\dots,n\}$.*

Proof: Let A be reducible. Then, after relabeling (permutation) of the indices it takes the form as on the right-hand side of (5.18), with $A_1 \in \mathbb{K}^{|N_1|\times|N_1|}$, $A_2 \in \mathbb{K}^{|N_2|\times|N_2|}$ where[26] $N_1 = \{1,\dots,n_1\}$ and $N_2 = \{n_1+1,\dots,n\}$. Let $i \in N_1$ and $j \in N_2$. *We claim:* j is not connected to i.

Suppose otherwise. Then there exists a sequence $(i,i_1),(i_1,i_2),\dots,(i_k,j) \in E$, i.e., every $A_{i_l,i_{l+1}} \neq 0$ for $l = 0\dots k$ (we set $i_0 = i$, $i_{k+1} = j$). Since $i \in N_1$, $j \in N_2$ and $N_1 \dot\cup N_2 = \{1,\dots,n\}$, there must exist at least one pair (i_l, i_{l+1}) with $i_l \in N_1$ and $i_{l+1} \in N_2$. However, the structure (5.18) implies $A_{i',j'} = 0$ for all $i' \in N_1$, $j' \in N_2$, which leads to the desired contradiction.

Conversely, suppose the existence of i and j such that j is not connected to i. To show that A is reducible, we distinguish the cases $j \neq i$ and $j = i$.

– *Case $j \neq i$:* We partition the index set: $\{1,\dots,n\} = N_1 \dot\cup N_2$, where

$$N_1 = \{i\} \cup \{i' \colon i' \text{ is connected to } i\},$$
$$N_2 = \{1,\dots,n\} \setminus N_1 = \{j' \colon (i \neq) j' \text{ is not connected to } i\}$$

By assumption, $N_2 \neq \emptyset$.

We claim: $A_{i',j'} = 0$ for all $i' \in N_1$ and $j' \in N_2$.

Suppose otherwise. Then there exists $i' \in N_1$ and $j' \in N_2$ such that $A_{i',j'} \neq 0$, i.e., $(i',j') \in E$. Thus, since i' is connected to i, we conclude that j' is connected to i. In other words: $j' \in N_1$, which contradicts $j' \in \{1,\dots,n\} \setminus N_1$.

Now we may renumber the indices such that those of N_1 are listed first and then those of N_2 to obtain the desired reducible structure (5.18).

– *Case $j = i$:* As in the case $j \neq i$, we define $N_1 = \{i' \colon i' \text{ is connected to } i\}$ and $N_2 = \{1,\dots,n\} \setminus N_1 = \{i' \colon i' \text{ is not connected to } i\}$. By assumption, $i \in N_2$, thus $N_2 \neq \emptyset$. If $N_1 \neq \emptyset$, then we may reason as in the first case that $A_{i',j'} = 0$ for all $i' \in N_1$ and $j' \in N_2$. It therefore remains to consider the case $N_1 = \emptyset$. This means $A_{i,i'} = 0$ for all $i' \in \{1,\dots,n\}$, i.e., the matrix A has a null row.

Renumbering the indices such that i appears last guarantees again that we obtain the structure (5.18) with $A_1 = 0 \in \mathbb{K}^{1\times1}$ and $A_2 \in \mathbb{K}^{(n-1)\times(n-1)}$. □

Definition 5.4 (strict and irreducible diagonal dominance) *A matrix $A \in \mathbb{R}^{n\times n}$ is called **strictly diagonally dominant** if*

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}| < |A_{i,i}|, \quad i = 1\dots n$$

*A is called **irreducibly diagonally dominant** if*

(i) A is irreducible,

(ii) A is weakly diagonally dominant, i.e., $\displaystyle\sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}| \leq |A_{i,i}|, \quad i = 1\dots n,$

(iii) $\exists\, \hat\imath \in \{1,\dots,n\}$ *such that* $\displaystyle\sum_{\substack{j=1\\ j\neq \hat\imath}}^{n} |A_{\hat\imath,j}| < |A_{\hat\imath,\hat\imath}|.$

[26] Indices are already relabeled here.

Exercise 5.4 Show that the matrices for the 1D and 2D Poisson problem (i.e., the matrices of Examples 2.1 and 2.2) are irreducibly diagonally dominant. ∎

For strictly diagonally dominant A we have $A_{i,i} \neq 0$ for all i. Thus, D is invertible, and therefore the Jacobi and Gauss-Seidel methods are well-defined. More generally, we have[27]

Theorem 5.4 *Let $A \in \mathbb{R}^{n\times n}$ be strictly diagonally dominant or irreducibly diagonally dominant. Then, A as well as its diagonal D are invertible.*

Proof:

- Let A be strictly diagonally dominant. Then D is invertible. *We claim:* A is also invertible.

  Suppose otherwise. Then $Ax = 0$ for some $x \neq 0$. Without loss of generality we assume $\|x\|_\infty = 1$, with $|x_i| = 1$ for some index i. Then, $\sum_{j=1}^n A_{i,j}\, x_j = 0$ implies

  $$|A_{i,i}| = |A_{i,i}\, x_i| = \Big|\sum_{\substack{j=1\\ j\neq i}}^{n} A_{i,j}\, x_j\Big| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}\, x_j| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}|$$

  which contradicts strict diagonal dominance.

- Let A be irreducibly diagonally dominant. *We claim:* A is invertible.

  Suppose otherwise. Then $Ax = 0$ for some $x \neq 0$. Without loss of generality we assume $\|x\|_\infty = 1$. We partition the index set: $\{1, \ldots, n\} = N_1 \dot\cup N_2$, where

  $$N_1 = \{k : |x_k| = 1\}, \quad N_2 = \{k : |x_k| < 1\}$$

  Here $N_1 \neq \emptyset$. If $N_2 = \emptyset$, then $x \neq 0$ is constant up to signs $\pm$, which implies $\sum_{j=1}^n \pm A_{i,j} = 0$ for $i = 1 \ldots n$. Thus,

  $$|A_{i,i}| = \Big|\sum_{\substack{j=1\\ j\neq i}}^{n} \pm A_{i,j}\Big| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}|, \quad i = 1 \ldots n$$

  which contradicts assumption (iii) about A. Thus, also $N_2 \neq \emptyset$. Due to irreducibility (assumption (i)) there exists at least one pair of indices $i \in N_1$ and $k \in N_2$ with $A_{i,k} \neq 0$. Together with $|x_k| < 1$ this implies

  $$|A_{i,i}| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}|\,|x_j| < \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}|$$

  which contradicts weak diagonal dominance (assumption (ii)).

  We claim: D is also invertible.

  Suppose otherwise. Then $A_{i,i} = 0$ for some index i. Then, by diagonal dominance, the i-th row of A is zero, which contradicts the invertibility of A. □

Exercise 5.5 Let A be strictly diagonally dominant or irreducibly diagonally dominant with positive diagonal $D > 0$. Show that all eigenvalues of A have positive real part. (In particular, if A is also symmetric, then A must be SPD.)

Hint: Let $\lambda \in \mathbb{C}$ be an eigenvalue of A with associated eigenvector $x \in \mathbb{C}^n$. Assume without loss of generality that $\|x\|_\infty = x_i = 1$ for some index i. Derive an upper bound for $|\lambda - a_{ii}|$, following a similar strategy as in the proof of Theorem 5.4. ∎

[27] All results in this section are equally valid for $A \in \mathbb{C}^{n\times n}$.

Theorem 5.5 *Let $A \in \mathbb{R}^{n\times n}$ be strictly diagonally dominant or irreducibly diagonally dominant. Then*

$$\rho(M^{Jac}) < 1 \quad \textit{and} \quad \rho(M^{GS}) < 1$$

Proof: We consider the error propagation $e \mapsto M\,e$.

– *Step 1.* We show $\|M^{Jac}\|_\infty < 1$ for A strictly diagonally dominant. From $M^{Jac} = I - D^{-1}A = D^{-1}(D-A)$ we obtain

$$(M^{Jac}\,e)_i = -\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{A_{i,j}}{A_{i,i}}\,e_j, \quad i = 1\ldots n$$

This gives

$$\|M^{Jac}\,e\|_\infty = \max_i \left|(M^{Jac}\,e)_i\right| = \max_i \Big|\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{A_{i,j}}{A_{i,i}}\Big|\,\|e\|_\infty < \|e\|_\infty \tag{5.19}$$

Thus, $\rho(M^{Jac}) \le \|M^{Jac}\|_\infty < 1$.

– *Step 2.* We show $\rho(M^{GS}) < 1$ for A strictly diagonally dominant. Let λ be an eigenvalue of $M^{GS} = I - (L+D)^{-1}A = -(L+D)^{-1}U$ with corresponding eigenvector x. Without loss of generality we assume $\|x\|_\infty = 1$, with $|x_i| = 1$ for some index i. Then, $-Ux = \lambda(L+D)\,x$, i.e.,

$$-\sum_{j>i} A_{i,j}\,x_j = \lambda\,A_{i,i}\,x_i + \lambda\sum_{j<i} A_{i,j}\,x_j, \quad i = 1\ldots n$$

This implies

$$|\lambda| \le \frac{\left|\sum_{j>i} A_{i,j}\,x_j\right|}{|A_{i,i}\,x_i| - \left|\sum_{j<i} A_{i,j}\,x_j\right|} \le \frac{\sum_{j>i}|A_{i,j}|}{|A_{i,i}| - \sum_{j<i}|A_{i,j}|}, \quad i = 1\ldots n$$

where the denominator is positive due to strict diagonal dominance. Thus,

$$|\lambda| \le \frac{\sum_{j>i}|A_{i,j}|}{\sum_{j>i}|A_{i,j}| + \underbrace{\left[|A_{i,i}| - \sum_{j>i}|A_{i,j}| - \sum_{j<i}|A_{i,j}|\right]}_{>0}} < 1$$

again due to strict diagonal dominance.

– *Step 3.* We show $\rho(M^{Jac}) < 1$ for A irreducibly diagonally dominant. To this end, let $\lambda \in \mathbb{C}$ with $|\lambda| \ge 1$. We show that λ cannot be an eigenvalue of M^{Jac}, i.e., we show that $M^{(\lambda)} := M^{Jac} - \lambda I$ is invertible:

Since A is irreducible, the same is true for $D-A$ and $M^{Jac} = D^{-1}(D-A)$ as well as $M^{(\lambda)}$. Due to diagonal dominance of A we have

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |M^{(\lambda)}_{i,j}| = \sum_{\substack{j=1\\ j\neq i}}^{n} |M^{Jac}_{i,j}| = \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{|A_{i,j}|}{|A_{i,i}|} \le 1 \le |\lambda| = |M^{(\lambda)}_{i,i}|, \quad i = 1\ldots n \tag{5.20a}$$

since M^{Jac} has diagonal 0. Furthermore, by assumption on A, for at least one index $i = \hat{\imath}$ we have '<1' instead of '≤ 1' in (5.20a). This shows that $M^{(\lambda)}$ is also irreducibly diagonally dominant and therefore invertible (Theorem 5.4), which concludes Step 3.

– *Step 4.* We show $\rho(M^{GS}) < 1$ for A irreducibly diagonally dominant. We proceed in the same way as in Step 3, with $M^{(\lambda)} = M^{GS} - \lambda I$, $|\lambda| \geq 1$:

The matrix $M^{(\lambda)} = -(L+D)^{-1}U - \lambda I$ is invertible iff $\tilde{M}^{(\lambda)} = \lambda(L+D) + U$ is invertible. $\tilde{M}^{(\lambda)}$ is irreducible since $A = D + L + U$ is irreducible and $\lambda \neq 0$. Due to diagonal dominance of A we have

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |\tilde{M}^{(\lambda)}_{i,j}| = |\lambda| \sum_{j=1}^{i-1} |A_{i,j}| + \sum_{j=i+1}^{n} |A_{i,j}| \leq |\lambda| \sum_{\substack{j=1\\ j\neq i}}^{n} |A_{i,j}| \leq |\lambda|\,|A_{i,i}| = |\tilde{M}^{(\lambda)}_{i,i}|, \quad i = 1 \ldots n \tag{5.20b}$$

In the same way as in Step 3 we conclude that $M^{(\lambda)}$ is irreducibly diagonally dominant and therefore invertible. □

Remark 5.7 For the strictly diagonally dominant case, (5.19) provides an upper bound for $\|M^{Jac}\|_\infty < 1$ in terms of the matrix entries, which depends on the 'degree of diagonal dominance'. It can also be shown that, in this case, we have $\|M^{GS}\|_\infty \leq \|M^{Jac}\|_\infty < 1$; see [2, Sec 7.6].

A more detailed analysis shows that for diagonally dominant and irreducibly diagonally dominant matrices, we have $\rho(M^{GS}) \leq \rho(M^{Jac}) < 1$, i.e., the Gauss-Seidel method converges faster than the Jacobi method; cf. [10],[13]. For certain (other) types of matrices, this can even be quantified: For example, for consistently ordered matrices (see Sec. 5.3 below) one can show $\rho(M^{GS}) = (\rho(M^{Jac}))^2$ if $\rho(M^{Jac}) < 1$. It is a good rule of thumb to expect the Gauss-Seidel method to be superior to the Jacobi method. ∎

5.3 Model problem and consistent ordering

A shortcoming of the results obtained so far is that they do not provide useful quantitative estimates for the convergence rate, in particular for the Gauss-Seidel method. It is also not so clear whether the introduction of the relaxation parameter ω for the Gauss-Seidel method can improve things substantially. For a class of matrices with a special [block-]band structure, it turns out that it is possible to choose the relaxation parameter ω in such a way that the convergence is significantly sped up.

In the literature, matrices to which this special type of analysis applies are called *consistently ordered* (see Def. 5.5 below). This definition is motivated by the special structure of the matrix A for the 2D Poisson problem in (see Example 2.2). In fact, the 2D Poisson problem provides the example *par excellence* of a consistently ordered matrix. This special property is usually not retained for more general finite difference matrices; in this sense, the theory based on consistent ordering is sort of a model problem analysis. From a historical point of view, this analysis was a significant step towards a deeper understanding of the convergenc properties of SOR, at least for one highly relevant class of application problems.

In a nutshell, the results are:

- The (by far) most prominent example of a consistently ordered matrix is the matrix arising from the 2D Poisson problem (Example 2.2).
- For consistently ordered matrices, the Gauss-Seidel method (i.e., $\omega = 1$) converges at twice the rate of the Jacobi method (if it converges): $\rho(M^{GS}) = (\rho(M^{Jac}))^2$ (see Theorem 5.6).
- For consistently ordered matrices, the optimal damping parameter ω_{opt} for the Gauss-Seidel method is available explicitly in terms of $\beta = \rho(M^{Jac}) < 1$ (see Theorem 5.7): $\omega_{opt} = 2/(1+\sqrt{1-\beta^2})$ and $\rho(M^{SOR}_{\omega_{opt}}) = \omega_{opt} - 1$. For the case β close to 1 one sees that the optimal damping leads to a *significant* improvement: Let $\beta = 1 - \delta$ for some (small) $\delta > 0$. Then, $\omega_{opt} = 2 - \mathcal{O}(\sqrt{\delta})$ and $\rho(M^{SOR}_{\omega_{opt}}) = 1 - \mathcal{O}(\sqrt{\delta})$.

An illustration of the performance of the optimally damped SOR method for the 2D Poisson problem is given in Example 5.3 below. In particular, we will see that $\rho(M^{Jac}) = 1 - \mathcal{O}(h^2)$, whereas $\rho(M^{SOR}_{\omega_{opt}}) = 1 - \mathcal{O}(h)$.

Consistent ordering.

Definition 5.5 *A matrix $A \in \mathbb{R}^{n\times n}$ with diagonal D, strict lower part L, and strict upper part U is said to be **consistently ordered** if the eigenvalues of $A(z) = z\,D^{-1}L + \frac{1}{z}D^{-1}U$ are independent of $z \in \mathbb{C}\setminus\{0\}$.*

Example 5.1 A first class of matrices that are consistently ordered are *block tridiagonal* matrices of the form

$$A = \begin{pmatrix} D_1 & T_{12} & & & \\ T_{21} & D_2 & T_{23} & & \\ & T_{32} & D_3 & \ddots & \\ & & \ddots & \ddots & T_{p-1,p} \\ & & & T_{p,p-1} & D_p \end{pmatrix} = L + D + U$$

where the diagonal blocks D_i are themselves diagonal matrices. To see this, we verify that $D^{-1}L + D^{-1}U$ and $zD^{-1}L + \frac{1}{z}D^{-1}U$ are similar matrices for all $0 \neq z \in \mathbb{C}$:

$$zD^{-1}L + \tfrac{1}{z}D^{-1}U = X\big(D^{-1}L + D^{-1}U\big)X^{-1} \quad \text{for} \quad X = \begin{pmatrix} I & & & & \\ & zI & & & \\ & & z^2I & & \\ & & & \ddots & \\ & & & & z^{p-1}I \end{pmatrix}$$ ∎

Tridiagonal matrices (e.g., those arising in Example 2.1) fit into the setting of Example 5.1. The 2D situation of Example 2.2 does not. However, the matrix of Example 2.2 is also consistently ordered as the following example shows:

Example 5.2 Block tridiagonal matrices whose diagonal block T_i are tridiagonal matrices and whose off-diagonal block are diagonal matrices are consistently ordered. To see this, consider such a matrix in the form

$$B = \begin{pmatrix} T_1 & D_{12} & & & \\ D_{21} & T_2 & D_{23} & & \\ & D_{32} & T_3 & \ddots & \\ & & \ddots & \ddots & D_{p-1,p} \\ & & & D_{p,p-1} & T_p \end{pmatrix}$$

We proceed as above by similarity transformations, with $X = \mathrm{Diag}(I, zI, z^2I, \ldots, z^{p-1}I)$ as above. A calculation shows

$$XBX^{-1} = \begin{pmatrix} T_1 & z^{-1}D_{12} & & & \\ zD_{21} & T_2 & z^{-1}D_{23} & & \\ & zD_{32} & T_3 & \ddots & \\ & & \ddots & \ddots & z^{-1}D_{p-1,p} \\ & & & zD_{p,p-1} & T_p \end{pmatrix}$$

So far, we have neither exploited the assumption that the off-diagonal blocks are diagonal nor that the diagonal blocks are tridiagonal. The next similarity transformation is again done by a block diagonal matrix, $C = \mathrm{Diag}(C_1, C_2, \ldots, C_p)$, where the matrices C_i (all of the same size $n \in \mathbb{N}$) are given by $C_i = \mathrm{Diag}(1, z, z^2, \ldots, z^{n-1})$. Observing that the off-diagonal block $D_{i,j}$ are diagonal matrices then allows us to conclude

$$CXBX^{-1}C^{-1} = \begin{pmatrix} C_1T_1C_1^{-1} & z^{-1}D_{12} & & & \\ zD_{21} & C_2T_2C_2^{-1} & z^{-1}D_{23} & & \\ & zD_{32} & C_3T_3C_3^{-1} & \ddots & \\ & & \ddots & \ddots & z^{-1}D_{p-1,p} \\ & & & zD_{p,p-1} & C_pT_pC_p^{-1} \end{pmatrix}$$

Next, the fact that the diagonal blocks T_i are tridiagonal matrices implies that, upon writing $T_i = L_i + D_i + U_i$ (lower, diagonal, and upper part), we have $C_iT_iC_i^{-1} = D_i + zL_i + \frac{1}{z}U_i$. From this, one can conclude as in Example 5.1 that the original matrix B is consistently ordered. ∎

For the class of consistently ordered matrices, the following theorems have been proved by D. Young; see [19].

Theorem 5.6 *Let A be consistently ordered. Then, $\rho(M^{GS}) = (\rho(M^{Jac}))^2$. In particular, the Gauss-Seidel method converges iff the Jacobi method converges.*

Theorem 5.7 *Assume:*

(i) $\omega \in (0, 2)$,

(ii) M^{Jac} *has only real eigenvalues,*

(iii) $\beta = \rho(M^{Jac}) < 1$,

(iv) A *is consistently ordered.*

Then: $\rho(M_\omega^{SOR}) < 1$, *with*

$$\rho(M_\omega^{SOR}) = \begin{cases} 1 - \omega + \frac{1}{2}\omega^2\beta^2 + \omega\beta\sqrt{1 - \omega + \frac{\omega^2\beta^2}{4}} & \text{for} \quad 0 < \omega \le \omega_{opt} \\ \omega - 1 & \text{for} \quad \omega_{opt} \le \omega < 2 \end{cases} \tag{5.21a}$$

where ω_{opt} is given by

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \beta^2}} \tag{5.21b}$$

The value ω_{opt} minimizes $\rho(M_\omega^{SOR})$, which then takes the value $\rho(M_{\omega_{opt}}^{SOR}) = \omega_{opt} - 1$.

Corollary 5.2 (SPD case) *Let A be SPD and consistently ordered. Let $\omega \in (0, 2)$. Then $\sigma(M^{Jac}) \subseteq \mathbb{R}$ and $\rho(M^{Jac}) < 1$, and therefore the assertions of Theorem 5.7 hold.*

Proof: Theorem 5.3 implies $\rho(M^{GS}) < 1$. Theorem 5.6 then gives $\rho(M^{Jac}) < 1$. Furthermore, the spectrum of M^{Jac} is real, since $\sigma(M^{Jac}) = \sigma(I - D^{-1}A) = \sigma(I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}) \subset \mathbb{R}$ because $I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is symmetric. □

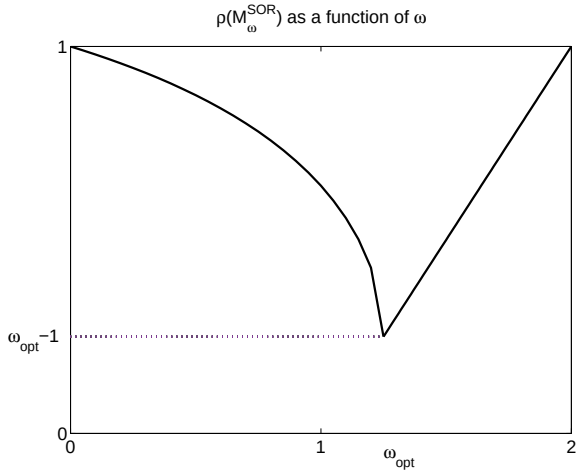


Figure 5.1: Asymptotic contraction rate $\rho(M_\omega^{SOR})$ of SOR in dependence of ω.

Discussion of Young's theorem.

By Example 5.2, the matrix for the 2D Poisson problem of Example 2.2 is consistently ordered. Hence, Young's theorem is applicable. We can compute $\sigma(M^{Jac})$ explicitly: According to Example 2.2, we have $\sigma(A) = \{4\sin^2\frac{i\pi}{n+1} + 4\sin^2\frac{j\pi}{n+1} : 1 \le i,j \le n\}$. Since the diagonal D of A is given by $D = 4I$, we easily compute

$$\sigma(M^{Jac}) = \sigma(I - D^{-1}A) = \left\{ 1 - \sin^2\tfrac{i\pi}{n+1} - \sin^2\tfrac{j\pi}{n+1} : 1 \le i,j \le n \right\}$$

From this we can conclude, with $h = \frac{1}{n+1}$ and some $c > 0$ independent of h (and thus of the problem size):

$$\rho(M^{Jac}) = 1 - ch^2 + \mathcal{O}(h^3)$$

This allows us to compute the optimal relaxation parameter

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - (\rho(M^{Jac}))^2}} = 2 - c'h + \mathcal{O}(h^2), \quad c' > 0 \text{ suitable}$$

We conclude that

$$\rho(M_{\omega_{opt}}^{SOR}) = 1 - c'h + \mathcal{O}(h^2)$$

Thus the SOR-method with optimally chosen relaxation parameter leads to a significant improvement of the convergence rate.

Example 5.3 We consider the matrix A from Example 2.2 for the cases $n = 10$ (i.e., $h \approx 0.1$) and $n = 100$ (i.e., $h \approx 0.01$). The right-hand side b is chosen as $b = (1, 1 \ldots 1)^T$; $x_0 = (1, 1 \ldots 1)^T$. We compare the Jacobi, the Gauss-Seidel, the optimally relaxed SOR-method and the CG method (see Chap. 8).

Fig. 5.2 shows the residual in the $\|\cdot\|_2$-norm versus the iteration count. We note that the Jacobi and Gauss-Seidel methods converge (visible for $n = 10$, invisible for $n = 100$); indeed, the Gauss-Seidel method converges at twice the rate of the Jacobi method, and the optimally relaxed SOR-method is significantly faster. We also observe that the CG method is vastly superior; here we also observe a superlinear convergence behavior. ∎

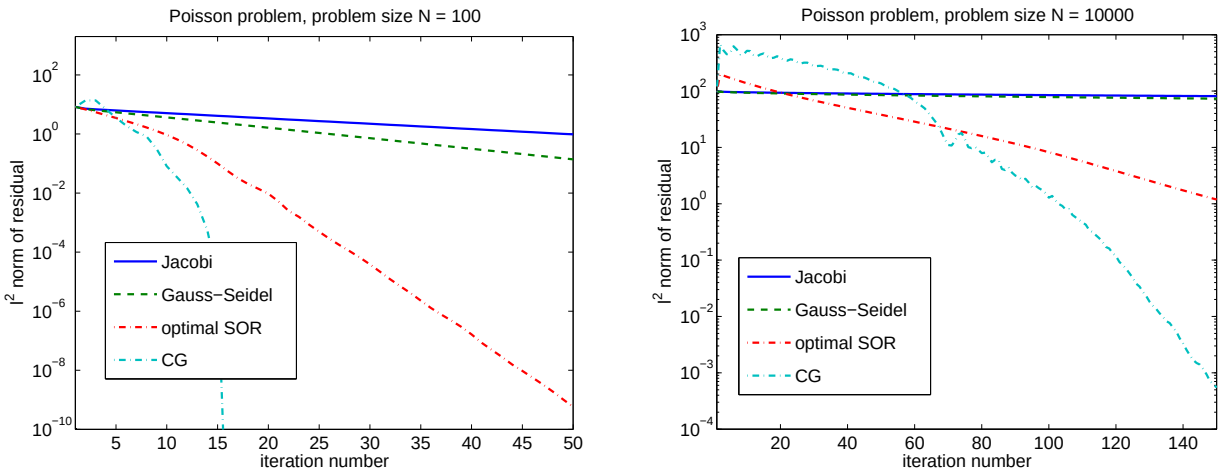


Figure 5.2: 2D Poisson problem: Comparison of Jacobi, Gauss-Seidel, optimal SOR, and CG method

Remark 5.8 In practice, β is not known and therefore also ω_{opt}. A possible technique is as follows: Choose $\omega < \omega_{opt}$ (e.g., $\omega = 1$). Perform a few SOR-steps and monitor the behavior of the iterates $\|x_{k+1} - x_k\|_2$. This gives an indication of $\rho(M_\omega^{SOR})$ and therefore, using (5.21a) (note: $\omega < \omega_{opt}$), of β. With the aid of (5.21b) one can then get an improved estimate for ω_{opt}. As long as $\omega < \omega_{opt}$ one can proceed in this fashion. Since (see Fig. 5.1) the function $\omega \mapsto \rho(M_\omega^{SOR})$ has a very steep slope for $\omega < \omega_{opt}$, one should tend to choose ω slightly larger than an (estimated) ω_{opt}. ∎

Classes of consistently ordered matrices; 'Property A'.

The property of a matrix to be consistently ordered does depend on the ordering. It is therefore of interest to identify matrices A for which permutation matrices P exist such that $P^T A P$ is consistently ordered. Examples of such matrices are those for which a suitable renumbering of the unknowns leads to block tridiagonal form, where the diagonal blocks are diagonal matrices (see Example 5.1), which is addressed by the term 'Property A'.

Let us see under which conditions on the adjacency graph G of A such a structure can be achieved. To this end, let A be a block tridiagonal matrix with p blocks on the diagonal. This numbering of the unknowns corresponds to a partitioning of the set V of vertices of G into p pairwise disjoint sets S_i, $i = 1 \dots p$: $V = S_1 \dot\cup S_2 \cdots \dot\cup S_p$. The fact that the diagonal blocks are diagonal matrices and that A is block tridiagonal implies:

- No node of a set S_i is connected to another node within the same S_i.
- Nodes of S_i are only connected to nodes of S_{i-1} or S_{i+1}.

Example 5.4 In Fig. 5.3 we illustrate the situation for the special case $p = 2$. In particular, we note that for the matrix A of the 2D Poisson problem, we can find a numbering which brings A to the desired form: The 'red-black ordering' (or: chequerboard ordering) as shown in the right part of Fig. 5.3 yields a partition of the indices that realizes the splitting $S_1 \dot\cup S_2$ with the desired property. The sparsity pattern of the reordered matrix is shown in Fig. 5.4. Since the reordered matrix is again consistently ordered, the convergence results related to Young's Theorem apply also to the reordered system, in particular for Gauss-Seidel and SOR.

We also note that, due to this reordering, the update steps for the (reordered) 'black' unknowns x_i in the first half step of the iteration can be computed independently, because the i-th equation involves only values of 'red' unknowns x_j, $j \neq i$, available from the previous iteration step. In the sequel, also the updates for the red unknowns are independent from each other. Thus, red-black ordering also enables an efficient parallelization of the SOR steps for this example. ∎

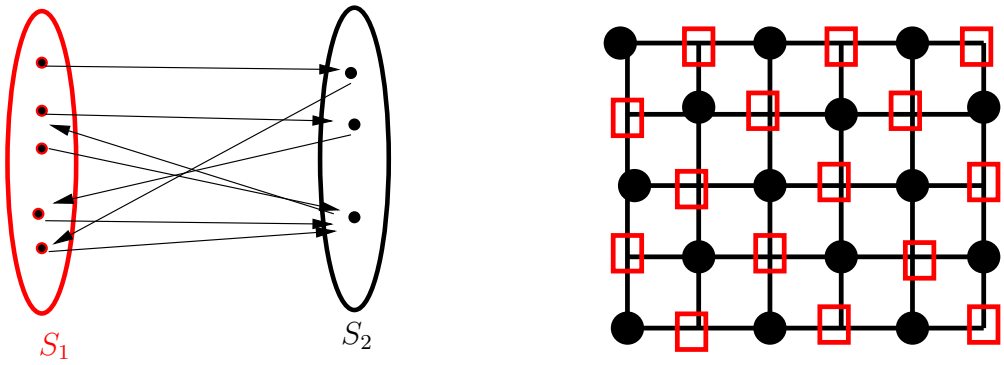


Figure 5.3: Left: illustration of property A. Right: partitioning ('red-black coloring') for model problem of Example 2.2 that realizes Property A

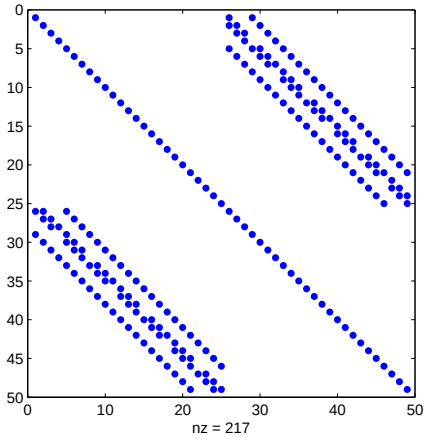


Figure 5.4: Sparsity pattern for 2D Poisson problem after red-black ordering

Optimal damping parameters for Richardson and Jacobi (SPD case).

The choice of optimal damping parameters for the Richardson and Jacobi methods is much simpler. We start with the Richardson method:

Lemma 5.4 *Let $A \in \mathbb{R}^{n\times n}$ be SPD, with smallest and largest eigenvalues $\lambda_{min} = \lambda_{min}(A)$, $\lambda_{max} = \lambda_{max}(A)$. Then the optimal damping parameter for the Richardson iteration (iteration matrix $M_\omega^{Rich} = I - \omega A$) is given by*

$$\omega_{opt} = \frac{2}{\lambda_{max} + \lambda_{min}}, \quad \text{with} \quad \rho(M_{\omega_{opt}}^{Rich}) = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}$$

Proof: The spectrum of M^{Rich} is given by $\sigma(M_\omega^{Rich}) = 1 - \omega\,\sigma(A) \subseteq [1 - \omega\lambda_{max}, 1 - \omega\lambda_{min}]$. Hence, for real ω, we obtain $\rho(M_\omega^{Rich}) = \max\{|1 - \omega\lambda_{max}|, |1 - \omega\lambda_{min}|\}$. Graphical considerations then show that $\rho(M_\omega^{Rich})$ is minimal for $1 - \omega\,\lambda_{min} = -(1 - \omega\,\lambda_{max})$, i.e., $\omega = \omega_{opt} = \frac{2}{\lambda_{min}+\lambda_{max}}$, with

$$1 - \omega_{opt}\,\lambda_{max} = \frac{\lambda_{min} - \lambda_{max}}{\lambda_{max} + \lambda_{min}}, \quad \text{and} \quad 1 - \omega_{opt}\,\lambda_{min} = \frac{\lambda_{max} - \lambda_{min}}{\lambda_{max} + \lambda_{min}}$$

See Fig. 5.5. □

If $0 < [\lambda, \Lambda] \supset [\lambda_{min}, \lambda_{max}]$, i.e., with lower resp. upper bounds for λ_{min} and λ_{max}, then $\omega_{opt} = \frac{2}{\lambda+\Lambda}$ is the appropriate choice, with $\rho(M_{\omega_{opt}}^{Rich}) = \frac{\Lambda-\lambda}{\Lambda+\lambda}$.

One may proceed analogously for the Jacobi method. Assume that

$$\lambda D \le A \le \Lambda D$$

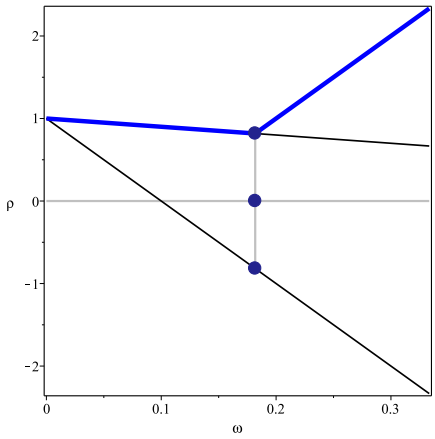


Figure 5.5: Visualization of the proof of Lemma 5.4, with $\lambda_{min} = 1$, $\lambda_{max} = 10$: $1 - \omega\lambda_{min}$ (upper thin line), $1 - \omega\lambda_{max}$ (lower thin line), $\max\{|1 - \omega\lambda_{min}|, |1 - \omega\lambda_{max}|\}$ (thick line).

for some $\Lambda > \lambda > 0$. Then, from Lemma 5.2 we have $\sigma(M_\omega^{Jac}) = \sigma(I - \omega D^{-1}A) \subseteq [1 - \omega\Lambda, 1 - \omega\lambda]$. Then the optimal damping parameter for the Jacobi method is again given by $\omega_{opt} = \frac{2}{\lambda+\Lambda}$, with

$$\rho(M_{\omega_{opt}}^{Jac}) = \frac{\Lambda - \lambda}{\Lambda + \lambda}$$

Block versions.

The Jacobi method and the Gauss-Seidel method can also be employed in block versions. Namely, let A be partitioned in the form

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & \vdots \\ A_{2,1} & A_{2,2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & A_{p,p} \end{pmatrix}$$

where the entries $A_{i,j}$ are matrix blocks.

The *block Jacobi* method consists then in defining the block diagonal matrix

$$D = \begin{pmatrix} A_{1,1} & & & \\ & A_{2,2} & & \\ & & \ddots & \\ & & & A_{p,p} \end{pmatrix}$$

and performing the iteration $x_{k+1} = x_k + D^{-1}(b - Ax)$. The block Gauss-Seidel and the block SOR methods are defined in an analogous way. Convergence theories exist for these cases as well.

If A corresponds to a FD or FEM matrix like in the 2D Poisson example, with original lexicographic ordering of the unknowns, then the diagonal blocks $A_{i,i}$ are tridiagonal matrices, and inversion of D amounts to solving one-dimensional problems represented by small tridiagonal systems, e.g. via Cholesky decomposition. The higher effort for such a block iteration is usually paid off by a faster convergence rate.

In the context of FD methods, such a version of block relaxation scheme is also called 'line relaxation', where the coupling of unknowns in one of the coordinate directions is retained. Another variant is 'alternating direction line relaxation' (ADI), where the ordering of unknowns varied in in an alternating fashion, similarly as in SSOR; cf. e.g. [19].

6 Chebyshev Acceleration and Semi-iterative Methods

Choosing the optimal relaxation parameter ω_{opt} is not an easy task in general. Chebyshev acceleration and its variants are an alternative, not uncommon tool to accelerate the convergence of a sequence $(x_k)_{k=0}^{\infty}$. Let this sequence be generated by the *primary iteration*

$$x_{k+1} = Mx_k + Nb \tag{6.1}$$

We assume $\rho(M) < 1$, such that (6.1) is convergent. We ask: Can we construct, for every k, an approximation $\tilde{x}_k$ based on $x_0, \dots, x_k$ such that the new sequence $(\tilde{x}_k)_{k=0}^{\infty}$ features *faster* convergence towards the fixed point $x_* = (I - M)^{-1}Nb$ of (6.1)? To this end we make an ansatz for a *secondary iteration* in form of a weighted sum,

$$\tilde{x}_k = \sum_{m=0}^{k} a_{k,m}\, x_m \tag{6.2a}$$

for some parameters $a_{k,m}$ to be chosen. With the explicit (discrete variation-of-constant) representation for the x_m,

$$x_m = M^m x_0 + \sum_{\ell=1}^{m} M^{m-\ell} Nb$$

and the polynomials

$$p_k(t) = \sum_{m=0}^{k} a_{k,m}\, t^m \in \mathcal{P}_k, \qquad q_{k-1}(t) = \sum_{m=0}^{k} a_{k,m} \sum_{\ell=1}^{m} t^{m-\ell} \in \mathcal{P}_{k-1}$$

the secondary iteration (6.2a) can be written as

$$\tilde{x}_k = p_k(M)\, x_0 + q_{k-1}(M) Nb \tag{6.2b}$$

with the matrix polynomials $p_k(M)$ and $q_{k-1}(M)$. Of course we require that the parameters $a_{k,m}$ be chosen such that a fixed point of (6.1) is reproduced, i.e., if we would consider $(x_k)_{k=0}^{\infty}$ to be the constant sequence $(x_*)_{k=0}^{\infty}$, then the sequence $(\tilde{x}_k)_{k=0}^{\infty}$ defined by (6.2a) should be the same constant sequence. Thus, we require the *consistency condition*

$$1 = \sum_{m=0}^{k} a_{k,m} = p_k(1) \quad \forall\, k \in \mathbb{N}_0$$

i.e., all $\tilde{x}_k$ are weighted means of the x_m, $m = 0 \dots k$. Under this assumption we can express the error $\tilde{e}_k = \tilde{x}_k - x_*$ of the secondary iteration in terms of the primary error $e_k = x_k - x_*$:

$$\tilde{e}_k = \tilde{x}_k - x_* = \sum_{m=0}^{k} a_{k,m}(x_m - x_*) = \sum_{m=0}^{k} a_{k,m}\, e_m = \sum_{m=0}^{k} a_{k,m} M^m e_0 = p_k(M)\, e_0 \tag{6.3}$$

This formula for the error gives an indication of how the coefficients $a_{k,m}$, or, equivalently, the polynomials p_k should be chosen: we should choose p_k such that $\|p_k(M)\|$ is small (or even minimal) in some norm of interest. Since sometimes information about location of the spectrum of M is available, we state:

- By Exercise below 6.1, $\sigma(p(M)) = p(\sigma(M))$ for any matrix M and any polynomial p. Thus,

$$\rho(p(M)) = \max\{|p(\lambda)| : \lambda \in \sigma(M)\} \tag{6.4}$$

- Normal matrices M satisfy $\|M\|_2 = \rho(M)$. Hence, for normal matrices M, since $p(M)$ is also normal, there holds $\rho(p(M)) = \|p(M)\|_2$ for all polynomials p. In particular, this is true for symmetric matrices M.

- Let A be SPD and let M be 'A-normal', i.e., $M^A M = MM^A$, e.g., M is A-selfadjoint. Then, analogously as for the case of normal M we have $\rho(M) = \|M\|_A$, and $\rho(p(M)) = \|p(M)\|_A$ for all polynomials p.

These considerations suggest that it is reasonable to seek $p_k \in \mathcal{P}_k$ as the solution of the minimization problem

$$\min_{\substack{p_k \in \mathcal{P}_k \\ p_k(1)=1}} \max_{\lambda \in \sigma(M)} |p_k(\lambda)| \tag{6.5a}$$

Since this problem is still hard to solve, we settle for less: If Γ is a closed subset of the open unit disc in $\mathbb{C}$ such that $\sigma(M) \subseteq \Gamma$, then we could seek to solve the minimization problem

$$\min_{\substack{p_k \in \mathcal{P}_k \\ p_k(1)=1}} \max_{z \in \Gamma} |p_k(z)| \tag{6.5b}$$

Of course, this still requires some *a priori* knowledge about the location of the spectrum. Here we consider the case that $\sigma(A) \subseteq \Gamma = [\alpha, \beta]$, an interval on the real line. In this case, the minimization problem (6.5b) can be solved explicitly (Corollary 6.1 below). As we will see in Sec. 6.2, the numerical realization can also be achieved in an efficient way.

Exercise 6.1 Let M be an arbitrary square matrix, and let p be a polynomial. Show (e.g., using the Jordan form): $\sigma(p(M)) = p(\sigma(M))$, i.e., the eigenvalues of $p(M)$ are given by $p(\lambda)$, where λ runs over all eigenvalues of M. ∎

6.1 Chebyshev polynomials

The *Chebyshev polynomials* of the first kind, $T_k \in \mathcal{P}_k$, are defined by the three-term recurrence

$$T_0(\xi) = 1, \quad T_1(\xi) = \xi, \quad \text{and} \quad T_{k+1}(\xi) = 2\xi T_k(\xi) - T_{k-1}(\xi), \quad k \geq 1 \tag{6.6a}$$

It can be verified by induction (see [2]) that these polynomials can be expressed in closed form as [28]

$$T_k(\xi) = \begin{cases} \cos(k \arccos \xi), & |\xi| \leq 1 \\ \frac{1}{2}\left[(\xi + \sqrt{\xi^2 - 1})^k + (\xi + \sqrt{\xi^2 - 1})^{-k}\right], & |\xi| \geq 1 \end{cases} \tag{6.6b}$$

Among the numerous remarkable properties of Chebyshev polynomials, we note that they are the solutions of an optimization problem of the form considered in (6.5b):

Theorem 6.1 *Let $[\alpha, \beta] \subseteq \mathbb{R}$ be a non-empty interval, and let γ be any real scalar outside this interval. Then the minimum*

$$\min_{\substack{p \in \mathcal{P}_k, \\ p(\gamma)=1}} \max_{t \in [\alpha,\beta]} |p(t)| \tag{6.7a}$$

is attained by the polynomial [29]

$$p(t) = C_k(t) = \frac{T_k(1 + 2\frac{t-\beta}{\beta-\alpha})}{T_k(1 + 2\frac{\gamma-\beta}{\beta-\alpha})} \tag{6.7b}$$

Furthermore, the minimizer is unique.

---

[28] For $|\xi| \geq 1$, the $T_k(\xi)$ can also be expressed in terms of cosh and arcosh, see [2].

[29] Here, $C_k(\gamma) = 1$. Note that for $t \in [\alpha, \beta]$ we have $1 + 2\frac{t-\beta}{\beta-\alpha} \in [-1, 1]$.

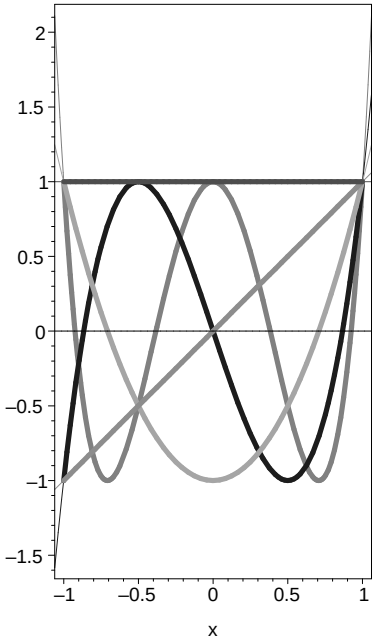


Figure 6.1: Chebyshev polynomials of the first kind

Proof: A detailed proof is given in [10, Sec. 7.3]. Here we show that $C_k(t)$ from (6.7b) indeed solves the minimization problem (6.7a); uniqueness follows along the same lines.

By affine transformation of t (which does not change the L^∞-norm $\|\cdot\|_\infty$), we may restrict ourselves to the standard interval $[\alpha,\beta]=[-1,1]$, with $|\gamma|>1$. Then, $C_k(t)=C_\gamma T_k(t)$ with constant $C_\gamma = 1/T_k(\gamma)$. The Chebyshev polynomial $T_k(\xi)=\cos(k\arccos\xi)$ attains the values ±1 at the points $\xi_i=\cos(i\pi/k), i=0\ldots k$, and it alternates between 1 and -1 (i.e., $T_k(\xi_i)$ and $T_k(\xi_{i+1})$ have opposite signs). Furthermore, $\|T_k\|_{L^\infty[-1,1]}=1$, implying $\|C_k\|_{L^\infty[-1,1]}=|C_\gamma|$.

Assume now the existence of $\pi\in\mathcal{P}_k$ such that $\pi(\gamma)=1$ and $\|\pi\|_{L^\infty[-1,1]}<\|C_k\|_{L^\infty[-1,1]}=|C_\gamma|$. Then, the polynomial $r=C_k-\pi$ changes sign k times in the interval $[-1,1]$ [sketch]. Thus, r has at least k zeros in $[-1,1]$. Additionally, $r(\gamma)=0$. Hence, $r\in\mathcal{P}_k$ has at least $k+1$ zeros; thus, $r\equiv 0$, which leads to a contradiction. □

It will be relevant to have quantitative bounds for the minimal value in (6.7a):

Corollary 6.1 *Under the assumptions of Theorem 6.1 and with $\alpha<\beta<\gamma$, we have*

$$\min_{\substack{p\in\mathcal{P}_k\\ p(\gamma)=1}} \max_{t\in[\alpha,\beta]} |p(t)| = \frac{1}{T_k\big(1+2\frac{\gamma-\alpha}{\beta-\alpha}\big)} = 2\frac{c^k}{1+c^{2k}}, \qquad c=\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \quad \kappa=\frac{\gamma-\alpha}{\gamma-\beta}$$

The same bound holds for the case $\gamma<\alpha<\beta$, if κ is replaced by $\kappa=\frac{\gamma-\beta}{\gamma-\alpha}$.

Proof: The key is to observe that $|T_k(\xi)|\le 1$ for $\xi\in[-1,1]$. This implies that the polynomial C_k from (6.7b) satisfies

$$\max_{t\in[\alpha,\beta]} |C_k(t)| = \frac{1}{|T_k(1+2\frac{\gamma-\beta}{\beta-\alpha})|}$$

The assertion then follows from the explicit representation of T_k given in (6.6b) and some manipulations (see, e.g., [10, Sec. 7.3.3] for details). □

6.2 Chebyshev acceleration for $\sigma(M) \subseteq (-1,1)$

We now assume $\sigma(M) \subseteq (-1,1)$ (corresponding to the case of a convergent primary iteration with M having a real spectrum). Moreover, we assume that parameters $-1 < \alpha < \beta < 1$ are known such that $\sigma(M) \subseteq [\alpha, \beta]$. With these parameters α, β and $\gamma = 1$, we use the polynomials

$$p_k(t) = C_k(t)$$

explicitly given by (6.7b) to define the secondary iteration (6.2a). This results in a 'Chebyshev-type accelerated' iteration scheme. Note that this is a consistent choice since $C_k(1) = 1$ for all k.

Improved convergence behavior of the Chebyshev iterates in selfadjoint cases.

To quantify the convergence behavior of the secondary Chebyshev iteration, we consider the case that the iteration matrix M is B-selfadjoint with respect to the energy product $(\cdot,\cdot)_B$ for some SPD matrix B, i.e., $(Mx, y)_B \equiv (x, My)_B$ $(M^B = M)$. Any such matrix has a real spectrum. Furthermore, we now assume knowledge of $\rho \in (0,1)$ such that $\sigma(M) \subseteq [-\rho, \rho]$. We then choose the polynomials p_k defining the secondary iteration (6.2b) as $p_k(t) = C_k(t)$ with $\alpha = -\rho$, $\beta = \rho$, and $\gamma = 1$, i.e.,

$$\tilde{x}_k = \sum_{m=0}^{k} a_{k,m}\, x_m = p_k(M)\, x_0 + q_{k-1}(M) N b, \qquad a_{k,m} = \text{coefficients of } p_k \tag{6.8a}$$

with

$$p_k(t) = C_k(t) = \frac{T_k(1 + 2\,\frac{t-\rho}{\rho-(-\rho)})}{T_k(1 + 2\,\frac{1-\rho}{\rho-(-\rho)})} = \frac{T_k(t/\rho)}{T_k(1/\rho)} \tag{6.8b}$$

We then obtain from Corollary 6.1

$$\rho(p_k(M)) = \max_{\lambda\in\sigma(M)} |p_k(\lambda)| \le \max_{\lambda\in[-\rho,\rho]} |p_k(\lambda)| = \frac{2\,c^k}{1+c^{2k}} \le 2\,c^k, \quad c = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \quad \kappa = \frac{1+\rho}{1-\rho}$$

The assumption that M is B-selfadjoint implies $\|p(M)\|_B = \rho(p(M))$ for all polynomials p. Hence, for the primary and secondary errors e_k, $\tilde{e}_k$ we have (see (6.3))

$$\Big(\frac{\|e_k\|_B}{\|e_0\|_B}\Big)^{\frac{1}{k}} \le \|M^k\|_B^{\frac{1}{k}} = \rho(M),$$

$$\Big(\frac{\|\tilde{e}_k\|_B}{\|e_0\|_B}\Big)^{\frac{1}{k}} \le \|p_k(M)\|_B^{\frac{1}{k}} = \big(\rho(p_k(M))\big)^{\frac{1}{k}} \le 2^{\frac{1}{k}} c \le c$$

In Remark 5.2, we have seen that the convergence factors

$$q_{primary} = \limsup_{k\to\infty} \Big(\frac{\|e_k\|_B}{\|e_0\|_B}\Big)^{\frac{1}{k}} \le \rho(M),$$

$$q_{Cheb} = \limsup_{k\to\infty} \Big(\frac{\|\tilde{e}_k\|_B}{\|e_0\|_B}\Big)^{\frac{1}{k}} \le c$$

are good measures for the asymptotic behavior of the convergence speed. To compare $q_{primary}$ with q_{Cheb}, let us assume that the parameter ρ is the best possible choice, i.e., $\rho = \rho(M) < 1$. The interesting case is $\rho = 1-\delta$ for small $\delta > 0$. With $c = \frac{1-1/\sqrt{\kappa}}{1+1/\sqrt{\kappa}}$ and $\kappa = \frac{1+\rho}{1-\rho}$, some analysis shows $c < 1-\sqrt{2}\sqrt{\delta}+\delta/2 = 1-c'\sqrt{\delta}$ with some $c' > 0$. We therefore arrive at

$$q_{primary} \le 1-\delta, \qquad q_{Cheb} \le 1 - c'\sqrt{\delta} \tag{6.9}$$

In practice we typically have $q_{primary} = \rho$, such that for small δ, Chebyshev acceleration will noticeably improve the convergence behavior.

Numerical realization.

At first glance, a drawback of Chebyshev acceleration appears to be that the definition of $\tilde{x}_k$ according to (6.8a) requires knowledge of *all* primary iterates $x_0, \ldots, x_k$. In view of storage restrictions, this may be difficult to realize in practice. However, exploiting the three-term recurrence (6.6a) for the Chebyshev polynomials T_k removes this restriction.

The polynomials p_k from (6.8b) also satisfy a three-term-recurrence: A brief calculation shows

$$p_{k+1}(t) = \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} t\, p_k(t) - \frac{\mu_{k-1}}{\mu_{k+1}} p_{k-1}(t), \quad k \geq 1, \quad \text{with} \quad \mu_k = T_k(1/\rho) \tag{6.10a}$$

and initial functions

$$p_0(t) = 1, \qquad p_1(t) = \frac{T_1(t/\rho)}{T_1(1/\rho)} = \frac{t/\rho}{1/\rho} = t \tag{6.10b}$$

i.e., $a_{0,0} = 1$ and $a_{1,0} = 0$, $a_{1,1} = 1$. We also observe

$$\mu_{k+1} = \frac{2}{\rho}\mu_k - \mu_{k-1} \tag{6.10c}$$

which follows from $p_k(1) = 1$ for all k together with (6.10a). We are now ready to implement Chebyshev acceleration. With (6.10a) and $x_* = \lim_{k\to\infty} x_k$ we obtain from the error equation $\tilde{e}_k = p_k(M)\, e_0$:

$$\begin{aligned}
\tilde{x}_{k+1} &= x_* + \tilde{e}_{k+1} \\
&= x_* + p_{k+1}(M)\, e_0 \\
&= x_* + \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} M\, p_k(M)\, e_0 - \frac{\mu_{k-1}}{\mu_{k+1}} p_{k-1}(M)\, e_0 \\
&= x_* + \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} M\, \tilde{e}_k - \frac{\mu_{k-1}}{\mu_{k+1}} \tilde{e}_{k-1} \\
&= x_* + \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} M(\tilde{x}_k - x_*) - \frac{\mu_{k-1}}{\mu_{k+1}} (\tilde{x}_{k-1} - x_*) \\
&= \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} M\, \tilde{x}_k - \frac{\mu_{k-1}}{\mu_{k+1}} \tilde{x}_{k-1} + \frac{1}{\mu_{k+1}} \Big(\mu_{k+1} - \frac{2}{\rho}\mu_k M + \mu_{k-1} \Big) x_*
\end{aligned}$$

We now exploit the fact that x_* is a fixed point of the basic iteration, i.e., $x_* = Mx_* + N\,b$. This together with (6.10c) allows us to remove the appearance of x_* and to obtain a direct three-term recurrence for the $\tilde{x}_k$, without explicit use of the primary iterates x_k:

$$\tilde{x}_{k+1} = \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} M\tilde{x}_k - \frac{\mu_{k-1}}{\mu_{k+1}} \tilde{x}_{k-1} + \frac{2}{\rho}\frac{\mu_k}{\mu_{k+1}} N\,b, \quad \text{with} \quad \tilde{x}_0 = x_0, \quad \tilde{x}_1 = x_1 = Mx_0 + N\,b$$

Algorithm 6.1 Chebyshev acceleration

% input: primary iteration $x_{k+1} = Mx_k + Nb$;
% assumption: $\sigma(M) \subseteq [-\rho, \rho] \subseteq (-1, 1)$

1: Choose $x_0 \in \mathbb{R}^n$
2: $\tilde{x}_0 = x_0$, $\tilde{x}_1 = x_1 = Mx_0 + N\,b$
3: $\mu_0 = 1, \mu_1 = 1/\rho$;
4: **for** $k = 1, 2, \ldots$ **do**
5: $\quad \mu_{k+1} = \frac{2}{\rho}\mu_k - \mu_{k-1}$ % use recursion for μ_k instead of definition $\mu_k = T_k(1/\rho)$

$$\tilde{x}_{k+1} = 2\frac{\mu_k}{\rho\,\mu_{k+1}} M\tilde{x}_k - \frac{\mu_{k-1}}{\mu_{k+1}} \tilde{x}_{k-1} + 2\frac{\mu_k}{\rho\,\mu_{k+1}} N\,b$$

6: **end for**

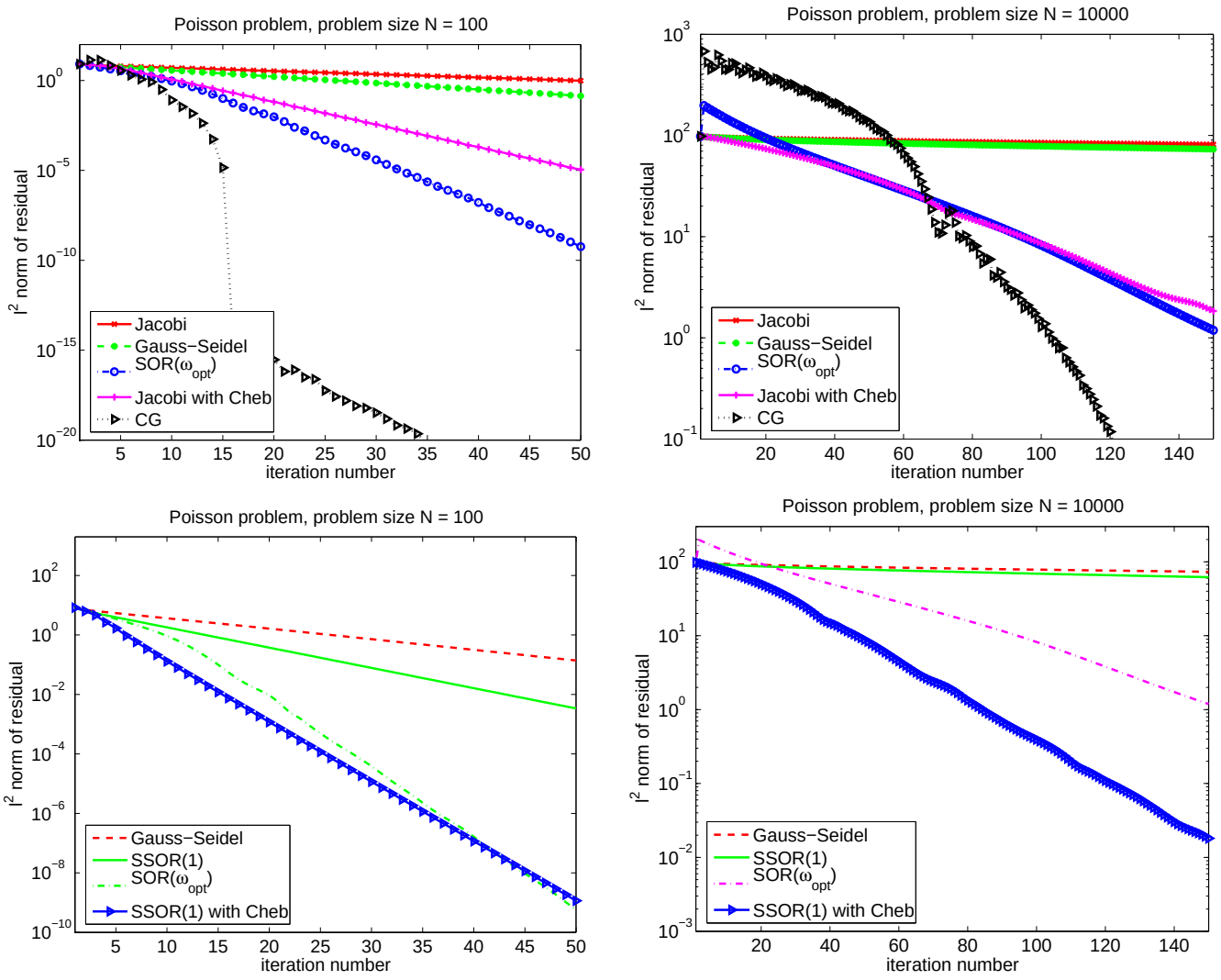


Figure 6.2: Chebyshev acceleration for the 2D Poisson problem (Example 2.2) applied to Jacobi iteration ($\rho = \rho(M^{Jac})$) and symmetric Gauss-Seidel (= SSOR(1)) with $\rho = \rho(M^{Jac})^4$.

6.3 Numerical example

We illustrate Chebyshev acceleration again for the model problem of Example 2.2. Since we require the iteration matrix to have a real spectrum, we would like it to be selfadjoint. For the model problem of Example 2.2, the iteration matrix $M^{Jac} = I - D^{-1}A$ of the Jacobi method is indeed symmetric.[30] For Chebyshev acceleration of the Jacobi method we assume that the parameter ρ has been chosen as $\rho = \rho(M^{Jac})$.

We also employ Chebyshev acceleration for the Gauss-Seidel method. Since $\sigma(M^{GS})$ is not necessarily real, we consider its symmetric variant, i.e., SSOR with $\omega = 1$. From Corollary 5.6 we know that $\rho(M^{GS}) = \rho(M^{SOR(1)}) = \rho(M^{Jac})^2$. For our calculations, we employ $\rho = \rho(M^{Jac})^4$ since we heuristically expect $\rho(M^{SSOR(1)}) \leq \rho(M^{GS})^2$ (note: SSOR(1) is effectively two Gauss-Seidel steps). Fig. 6.2 shows the performance of various iterative methods including the Chebyshev accelerated versions of the Jacobi method and of SSOR(1). We observe that Chebyshev acceleration does indeed significantly improve the convergence.

[30] For a general SPD matrix A, M^{Jac} is A-selfadjoint; cf. the proof of Theorem 5.2.

A brief discussion is in order. We have already seen that $\rho(M^{Jac}) = 1 - ch^2 + \mathcal{O}(h^3)$. Hence, from our discussion in (6.9) we infer that $\limsup_{k\to\infty}(\rho(p_k(M^{Jac})))^{1/k} = 1 - c'h + \mathcal{O}(h^{3/2})$ for some suitable c'. For the Chebyshev acceleration based on the SSOR(1) method, we have chosen $\rho = \rho(M^{Jac})^4$; again we get $\rho = 1 - c''h^2 + \mathcal{O}(h^3)$ and thus conclude that the Chebyshev accelerated SSOR(1) has a contraction rate of $1 - c'''h + \mathcal{O}(h^{3/2})$.

Exercise 6.2 Formulate the Chebyshev acceleration algorithm for the general case that the iteration matrix M satisfies $\sigma(M) \subseteq [\alpha, \beta]$. Let M^{SSOR} be the iteration matrix for the symmetric Gauss-Seidel iteration applied to the matrix A for the 2D Poisson problem. Compare the convergence behavior of the SSOR(1) method with the accelerated version. ∎

Remark 6.1 Formally comparing primary and secondary iterates,

$$x_k = M^k x_0 + \ldots, \qquad \tilde{x}_k = p_k(M)\, x_0 + \ldots$$

shows that Chebyshev acceleration is a more global technique, aiming at an optimal solution in the subspace spanned by

$$\operatorname{span}\{x_0, Mx_0, \ldots, M^k x_0\} \subseteq \mathbb{R}^n$$

Such a subspace is called a *Krylov space*. In later sections we study an important class of so-called Krylov subspace methods, where optimal iterates (optimal in some specific sense) are more systematically constructed as elements of Krylov spaces $\operatorname{span}\{x_0, A\,x_0, \ldots, A^k\,x_0\}$ for increasing k. ∎

7 Gradient Methods

Motivated by the rather slow convergence of classical iterative methods, and in view of the sensitivity of acceleration with respect to estimated parameters, a variety of alternative methods have been proposed. We first consider methods applicable to SPD matrices A, which often arise as a result of the discretization of elliptic operators, e.g., the matrices of Examples 2.1 and 2.2. Later on we relax this condition and consider nonsymmetric equations. Recall that an SPD matrix $A > 0$ satisfies

$$x^{\mathrm{T}}Ax = (x, Ax) = (Ax, x) = (x, x)_A = \|x\|_A^2 > 0 \quad \forall\, x \neq 0$$

In particular, all eigenvalues of A are positive.

The aim of *gradient methods* is to minimize the quadratic functional $\phi\colon \mathbb{R}^n \mapsto \mathbb{R}$,

$$\phi(x) = \tfrac{1}{2}(Ax, x) - (b, x) \tag{7.1a}$$

for given $b \in \mathbb{R}^n$. This is related to the linear system $Ax = b$ in the following way: Due to

$$\begin{aligned}\phi(x+h) - \phi(x) &= \tfrac{1}{2}\big[(A(x+h),(x+h)) - (Ax,x)\big] - \big[(b,x+h) - (b,x)\big] \\ &= \tfrac{1}{2}\big[(Ax,x) + (Ax,h) + (Ah,x) + (Ah,h) - (Ax,x)\big] - (b,h) \\ &= (Ax,h) + (Ah,h) - (b,h) = (Ax-b,h) + \mathcal{O}(\|h\|^2)\end{aligned}$$

we identify the gradient of ϕ as the (negative) residual[31]

$$\nabla\phi(x) = -(b - Ax) \tag{7.1b}$$

Moreover, the Hessian of ϕ is given by the Jacobian of $\nabla\phi$,

$$H\phi(x) = J(\nabla\phi)(x) \equiv A > 0 \tag{7.1c}$$

Thus, the functional ϕ has a unique minimum at x_*, the stationary point of ϕ, satisfying $\nabla\phi(x_*) = Ax_* - b = 0$. We conclude that

For $A > 0$, ***solving*** $Ax = b$ ***is equivalent to finding the minimum of*** $\phi(x)$ ***from*** (7.1a).

Exercise 7.1 Let x_* be the solution of $Ax = b$, $A > 0$. Show that

$$\phi(x) - \phi(x_*) = \tfrac{1}{2}(A(x - x_*), x - x_*) = \tfrac{1}{2}\|x - x_*\|_A^2 \tag{7.2}$$

and conclude again that ϕ has indeed a unique minimum. ∎

Exercise 7.1 also shows that

Minimization of $\phi(x)$ ***over any subdomain*** $D \subseteq \mathbb{R}^n$
is equivalent to minimization of the error $e = x - x_*$ ***in the energy norm.***

Remark 7.1 The equivalence of solving $Ax = b$ with $A > 0$ and minimization of $\phi(x)$ from (7.1a) parallels the variational formulation of a selfadjoint elliptic PDE. The simplest 2D example is the Poisson equation (2.3) with homogeneous Dirichlet boundary conditions, with the corresponding energy functional

$$\phi(u) = \tfrac{1}{2}a(u,u) - (f,u)_{L^2(\Omega)}, \qquad \text{with} \quad a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx \tag{7.3}$$

[31] For general $A \in \mathbb{R}^{n\times n}$ we have $\nabla\phi(x) = (\mathrm{Re}\,A)\,x - b$, $\mathrm{Re}\,A = \frac{1}{2}(A + A^{\mathrm{T}})$.

In the Sobolev space $H_0^1(\Omega)$, the unique minimum of $\phi(u)$ from (7.3) is attained for $u_* =$ weak solution of the boundary value problem (2.3). This is also referred to as the *Dirichlet principle*. In this context, ∇u may be considered as an analog of the discrete object $A^{\frac{1}{2}}x$, and $\|u\|_{H^1(\Omega)} = a(u,u)^{\frac{1}{2}} = \|\nabla u\|_{L^2(\Omega)}$ is the corresponding energy norm. The integration-by-parts identity $\int_\Omega (-\Delta u)v \, dx = \int_\Omega \nabla u \cdot \nabla v \, dx = a(u,v)$ (for $u, v \in H_0^1(\Omega)$) corresponds to the identity $(Ax, y) = (A^{\frac{1}{2}}x, A^{\frac{1}{2}}y) = (x,y)_A$. ∎

Simple iterative schemes for minimizing ϕ from (7.1a) are 'descent methods' and proceed as follows: Starting from an initial vector x_0, the iteration is defined by

$$x_{k+1} = x_k + \alpha_k d_k \tag{7.4}$$

where the *search direction* $0 \neq d_k \in \mathbb{R}^n$ and the *step length* $\alpha_k \in \mathbb{R}$ are to be chosen. Typically, once a search direction $d_k \neq 0$ is chosen, the step length α_k is taken as the minimizer of the one-dimensional minimization problem (a so-called *line search*):

$$\textbf{\textit{Find the minimizer}}\ \alpha_k \in \mathbb{R}\ \textbf{\textit{of}}\ \alpha \mapsto \phi(x_k + \alpha d_k). \tag{7.5}$$

This minimization problem is easily solved since it is quadratic and convex in α: Define $\psi(\alpha) = \phi(x_k + \alpha d_k)$. Application of the chain rule gives

$$\psi'(\alpha) = \big(\nabla\phi(x_k + \alpha d_k), d_k\big) = \big(A(x_k + \alpha d_k) - b, d_k\big) = \big(\alpha A d_k - r_k, d_k\big) = \alpha(Ad_k, d_k) - (r_k, d_k)$$

with the residual

$$r_k = b - Ax_k = -\nabla\phi(x_k) \tag{7.6}$$

Moreover, $\psi(\alpha)$ is of course convex: $\psi''(\alpha) \equiv (Ad_k, d_k) > 0$. The condition on a minimizer α_k of ψ is $\psi'(\alpha_k) = 0$. Thus,

$$\alpha_k = \frac{(d_k, r_k)}{(d_k, d_k)_A} = \frac{(d_k, r_k)}{\|d_k\|_A^2} \tag{7.7}$$

Exercise 7.2 Show that the choice of α_k in (7.7) leads to an approximation $x_{k+1} = x_k + \alpha_k d_k$ such that

$$\phi(x_{k+1}) - \phi(x_k) = -\frac{1}{2}\frac{|(d_k, r_k)|^2}{\|d_k\|_A^2} \tag{7.8}$$

Thus, if d_k is chosen such that $(d_k, r_k) \neq 0$, then indeed $\phi(x_{k+1}) < \phi(x_k)$. ∎

Remark 7.2 Some of the basic iterative methods are related to descent methods, or are descent methods 'in disguise':

For $A \in \mathbb{R}^{n\times n}$, consider the case where the first n search directions $d_0, \ldots, d_{n-1}$ are chosen as the unit vectors, $d_k = (0, \ldots, \overset{k}{\underset{\downarrow}{1}}, \ldots, 0)^{\mathrm{T}}$. Then we obtain from (7.7):

$$\alpha_k = \frac{(d_k, r_k)}{\|d_k\|_A^2} = \frac{(r_k)_k}{A_{k,k}}$$

hence

$$x_{k+1} = x_k + \frac{(r_k)_k}{A_{k,k}} d_k$$

which exactly corresponds to the k-th update in the *inner loop* of a Gauss-Seidel step (5.8a). Note that n of these updates result in a single Gauss-Seidel step, and further steps are obtained by repeating this procedure with cyclic choice of search directions (= unit vectors).

In the simplest version of the Richardson iteration (5.6) we simply take $d_k = r_k$ and $\alpha_k = 1$, but this does not minimize ϕ in the search direction r_k. The locally optimal choice (7.7) corresponds to the choice $\omega = \alpha_k = \frac{\|r_k\|_2^2}{\|r_k\|_A^2}$ for the relaxation parameter and leads to the steepest descent method discussed in the sequel. ∎

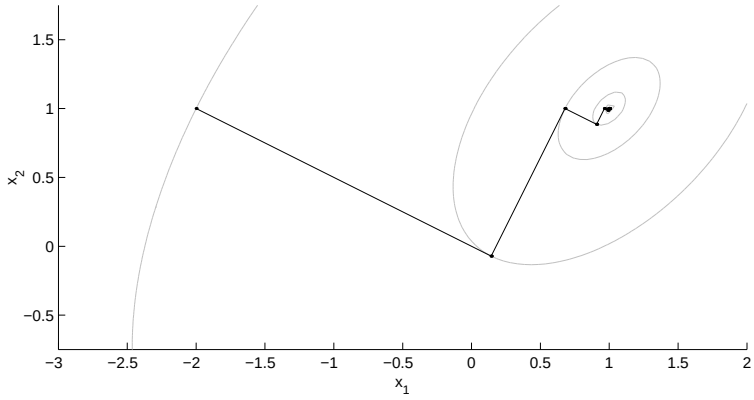


Figure 7.1: SD convergence path for a 2×2 matrix A with condition number $\kappa_2(A) = 3$

7.1 The Method of Steepest Descent (SD) for SPD systems

We need to specify the search direction d_k in the iteration (7.4). As shown in Exercise 7.1, most choices of d_k will lead to $\phi(x_{k+1}) - \phi(x_k) < 0$. The steepest descent method is a 'greedy' algorithm in the sense that it chooses d_k as the *local direction of steepest descent*, which is given by

$$d_k = -\nabla\phi(x_k) = r_k$$

This choice for the search direction, together with the step length α_k given by (7.7), leads to the Steepest Descent (SD) Algorithm formulated in Alg. 7.1.[32]

Algorithm 7.1 Steepest Descent (SD)

1: Choose $x_0 \in \mathbb{R}^n$
2: **for** $k = 0, 1, \ldots$ **do**
3: $\quad r_k = b - Ax_k$
4: $\quad \alpha_k = (r_k, r_k)/(Ar_k, r_k)$
5: $\quad x_{k+1} = x_k + \alpha_k r_k$
6: **end for**

Orthogonal search directions.

A consequence of our choice for the step length α_k in (7.7) is that

In SD, consecutive search directions are orthogonal to each other.

To see this, we observe

$$d_{k+1} = b - Ax_{k+1} = b - A(x_k + \alpha_k d_k) = r_k - \alpha_k A d_k$$

Inserting the definition of α_k given by (7.7) gives, with $d_k = r_k$,

$$(d_{k+1}, d_k) = (r_k, d_k) - \alpha_k(Ad_k, d_k) = (r_k, r_k) - \frac{(r_k, r_k)}{(Ar_k, r_k)}(Ar_k, r_k) = 0 \tag{7.9}$$

This behavior is visualized in Fig. 7.1.

[32] Basically, the idea of SD is applicable to any convex minimization problem.

Convergence of the SD method.

In order to quantify the speed of convergence of the steepest descent iteration, we use the *Kantorovich inequality* as a technical tool. Let $A > 0$ be a real SPD matrix, and λ_{max} and λ_{min} its largest and smallest eigenvalues. Then, for all $x \in \mathbb{R}^n$, the Kantorovich inequality states

$$\frac{(Ax,x)\,(A^{-1}x,x)}{(x,x)\,(x,x)} \le \frac{(\lambda_{min}+\lambda_{max})^2}{4\,\lambda_{min}\,\lambda_{max}} \tag{7.10}$$

For a proof see [19, Lemma 5.1].[33]

Now we study the magnitude of the error vectors $e_k = x_k - x_*$ in the energy norm $\|\cdot\|_A$. Note that $Ae_k = -r_k$, where $r_k = b - Ax_k$ is the k-th residual. From (7.2) and (7.8) we obtain with $d_k = r_k$:

$$\tfrac{1}{2}\|e_{k+1}\|_A^2 = \phi(x_{k+1}) - \phi(x_*) = \big(\phi(x_{k+1}) - \phi(x_k)\big) + \big(\phi(x_k) - \phi(x_*)\big) = -\frac{1}{2}\frac{|(r_k,r_k)|^2}{\|r_k\|_A^2} + \frac{1}{2}\|e_k\|_A^2 \tag{7.11a}$$

Now we use the Kantorovich inequality (7.10) and identity $r_k = -Ae_k$ to estimate

$$\frac{|(r_k,r_k)|^2}{\|r_k\|_A^2} = \frac{|(r_k,r_k)|^2}{(Ar_k,r_k)} \ge \frac{4\,\lambda_{min}\,\lambda_{max}}{(\lambda_{min}+\lambda_{max})^2}\,(A^{-1}r_k,r_k) = \frac{4\,\lambda_{min}\,\lambda_{max}}{(\lambda_{min}+\lambda_{max})^2}\,(e_k,Ae_k) = \frac{4\,\lambda_{min}\,\lambda_{max}}{(\lambda_{min}+\lambda_{max})^2}\,\|e_k\|_A^2$$

Together with (7.11a) this gives

$$\begin{aligned}\|e_{k+1}\|_A^2 &\le \|e_k\|_A^2\left(1 - \frac{4\,\lambda_{min}\,\lambda_{max}}{(\lambda_{min}+\lambda_{max})^2}\right) = \|e_k\|_A^2\,\frac{(\lambda_{max}-\lambda_{min})^2}{(\lambda_{max}+\lambda_{min})^2} \\ &= \left(\frac{\lambda_{max}-\lambda_{min}}{\lambda_{max}+\lambda_{min}}\right)^2\|e_k\|_A^2 = \left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^2\|e_k\|_A^2\end{aligned} \tag{7.11b}$$

with the condition number $\kappa_2(A) = \lambda_{max}/\lambda_{min}$. From this reasoning we obtain

---

Theorem 7.1 *For the SD iteration applied to an SPD system $Ax = b$, the error e_k after k steps is bounded in the energy norm by*

$$\|e_k\|_A \le \left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^k \|e_0\|_A \tag{7.12}$$

I.e., the asymptotic convergence rate is bounded by $\frac{\kappa_2(A)-1}{\kappa_2(A)+1}$.

Hence $e_k \to 0$ as $k \to \infty$. Evidently, the speed of convergence depends on the spectrum of A. In particular, when the condition number $\kappa_2(A)$ is large, the contours of the functional ϕ, which are elliptic in shape, are long drawn-out, and the poor convergence rate suggested by (7.11b) is graphically explained by 'zig-zag'-paths similar to the one shown in Fig. 7.2. This illustrates a worst case,[34] which occurs for an initial error close to the eigenvector associated with λ_{max}.

[33] The Kantorovich inequality is an example for a 'strengthened Cauchy-Schwarz (CS) inequality'. Using CS together with $\|A\|_2 = \lambda_{max}$, $\|A^{-1}\|_2 = 1/\lambda_{min}$ we would obtain the elementary, larger bound $\lambda_{max}/\lambda_{min} = \kappa_2(A)$ on the right-hand side of (7.10). In (7.12), this would result in the larger factor $\left(\frac{\kappa_2(A)-1}{\kappa_2(A)}\right)^{k/2}$. For $\kappa = \kappa_2(A) \to \infty$ the Kantorovich inequality gains a factor $\approx 1/4$, and with $\varepsilon = 1/\kappa$ this gives the following bounds for the asymptotic convergence rates: $\approx 1 - 2\varepsilon$ (Kantorovich) vs. $1 - \varepsilon/2$ (CS).

[34] The best case: For $e_0 =$ any eigenvector of A, the SD iteration always would find the exact solution x_* in one step, independent of the problem dimension (simple proof!). This is, of course, in no way a practical situation. Moreover, as illustrated in Fig. 7.2, a small deviation from the eigenvector associated with $\lambda_{\max}$ will lead to a very poor convergence behavior for the case of large $\kappa_2(A)$.

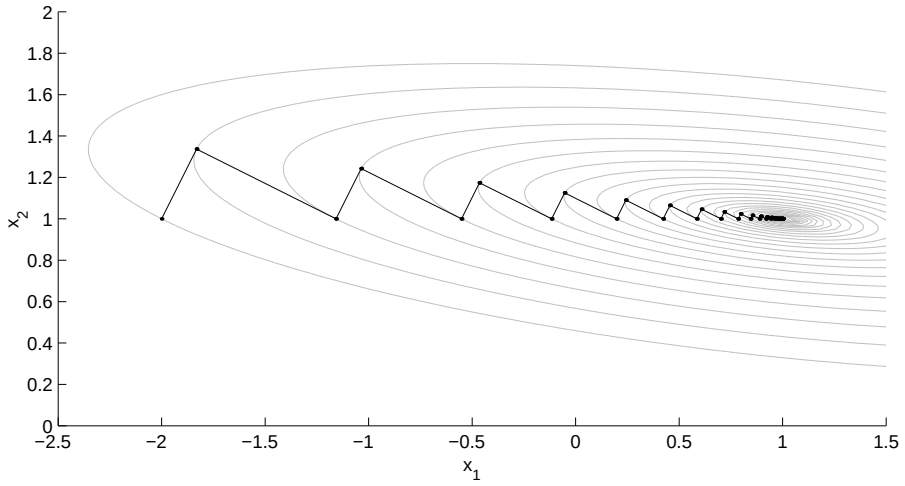


Figure 7.2: SD convergence path for a 2×2 matrix A with $\kappa_2 \approx 25$

Example 7.1 The performance of the SD method can now be compared with that of the Jacobi and Gauss-Seidel methods. The convergence paths for these three methods are shown in Fig. 7.3. The system considered is again that of Example 2.2, i.e., the matrix A arises from discretizing the 2D Poisson equation with the 5 point finite difference stencil over an 8×8 mesh or an 11×11 mesh. The right-hand side b is taken as $b = (1, 1, \ldots, 1)^{\mathrm{T}}$, and the starting vector is $x_0 = b$. We are only plotting a 2D projection of the solution vector in Fig. 7.3, and therefore the steepest descent orthogonality property is not graphically observed. Note that the SD iteration slows down with increasing k. ∎

The SD method does not prove to be a great improvement over the classical iterative methods. Nevertheless, it comes with a number of new concepts including formulating the given problem as a minimization problem and considering the relationship between consecutive search directions in a minimization procedure. These concepts are extended in the following in order to generate more successful iterative procedures.

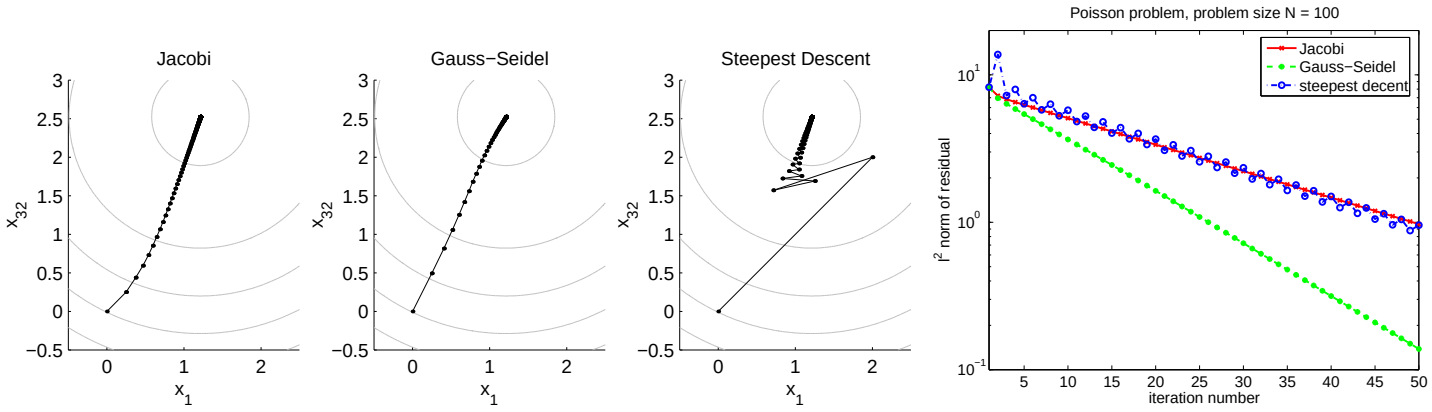


Figure 7.3: Convergence paths for the Jacobi, Gauss-Seidel and SD methods for the Poisson problem from Example 2.2. Left: case $N = 49$. Right: case $N = 100$.

7.2 Nonsymmetric steepest descent algorithms

In the steepest descent algorithm we have required A to be SPD in order for the functional ϕ to have a unique minimum at the solution of $Ax = b$. Variations on the steepest descent algorithm for nonsymmetric systems have also been developed, see [19]. The most general, but by far not computationally cheapest or most efficient requires only that A be invertible. Then, since $A^{\mathrm{T}}A$ is SPD, Alg. 7.1 can be applied to the normal equations

$$A^{\mathrm{T}}A\,x = A^{\mathrm{T}}b$$

This procedure is called the *residual norm steepest descent method*, and the functional being minimized in this case is

$$\psi(x) = \tfrac{1}{2}(Ax, Ax) - (Ax, b)$$

This method *minimizes the ℓ_2-norm of the residual,* $\|Ax - b\|_2^2$. However, in view of the convergence result (7.11b), it is now the condition number of $A^{\mathrm{T}}A$, which is typically much larger than that of A, that controls the convergence rate of the iteration.[35]

7.3 Gradient methods as projection methods

One of the main characteristics of the SD method is that consecutive search directions (i.e., the residuals) are orthogonal, (7.9), which implies that *the ℓ_2-projection of the new residual onto the previous one is zero.* Another way of putting it is: The approximation x_{k+1} is defined as the solution of

$$\textbf{\textit{Find}}\ x_{k+1} \in x_k + \operatorname{span}\{r_k\}\ \textbf{\textit{such that}}\ r_{k+1} \perp r_k\textbf{\textit{.}}$$

This is a local condition concerning consecutive residuals only; the 'search history', i.e., the information about the previous search directions $r_0, \ldots, r_{k-1}$ is not exploited. One may hope that including this information in the method leads to faster convergence, similar as for semi-iterative acceleration (see Sec. 6).

Krylov subspace methods are based on this idea: the approximation x_{k+1} is constructed such that the residual r_{k+1} is 'orthogonal' (in some appropriate sense to be specified below) to *all* previous residuals, search directions, or a related set of vectors.

Remark 7.3 Brief review on orthogonal projectors:

Let $\mathbb{R}^n = K \oplus K^\perp$, with $K \perp K^\perp$, be an orthogonal subspace decomposition of $\mathbb{R}^n$. Let

$$K = \operatorname{span}\{u_1, \ldots, u_m\}, \qquad K^\perp = \operatorname{span}\{v_1, \ldots, v_{n-m}\}$$

with $(u_i, u_j) = \delta_{i,j}$, $(v_i, v_j) = \delta_{i,j}$, and $(u_i, v_j) = 0$. The union of the u_i and v_j is an orthogonal basis of the full space $\mathbb{R}^n$. For given $x \in \mathbb{R}^n$ we consider the corresponding Fourier expansion

$$x = \sum_{i=1}^{m} (x, u_i)\, u_i + \sum_{j=1}^{n-m} (x, v_j)\, v_j =: Px + Qx \tag{7.13}$$

This defines a pair (P, Q) of *orthogonal projectors.* We say that P *projects onto K along $K^\perp$*, and vice versa.

[35] As before, (7.11b) represents only an upper bound for the error. Nevertheless, the bound describes the overall convergence behavior quite realistically.

From (7.13), and with

$$U = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ u_1 & u_2 & \vdots & u_m \\ | & | & & | \\ | & | & & | \end{array} \right) \in \mathbb{R}^{n\times m}, \qquad V = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ v_1 & v_2 & \vdots & v_{n-m} \\ | & | & & | \\ | & | & & | \end{array} \right) \in \mathbb{R}^{n\times(n-m)}$$

we obtain the matrix representations for the projectors $P, Q \in \mathbb{R}^{n\times n}$:

$$\sum_{i=1}^{m} u_i (u_i^{\mathrm{T}} x) = \sum_{i=1}^{m} (u_i u_i^{\mathrm{T}})\, x = U U^{\mathrm{T}} x = P x \in K$$

and analogously for the orthogonal complement,

$$\sum_{j=1}^{n-m} v_j (v_j^{\mathrm{T}} x) = \sum_{j=1}^{n-m} (v_j v_j^{\mathrm{T}})\, x = V V^{\mathrm{T}} x = Q x \in K^{\perp}$$

From the orthonormality relations $U^{\mathrm{T}}U = I_{m\times m}$, $V^{\mathrm{T}}V = I_{(n-m)\times(n-m)}$, and $U^{\mathrm{T}}V = 0_{m\times(m-n)}, V^{\mathrm{T}}U = 0_{(n-m)\times m}$ we obtain the characteristic identities for a pair of orthogonal projectors:

$$P^2 = P = P^{\mathrm{T}}, \quad Q^2 = Q = Q^{\mathrm{T}}, \quad P\,Q = Q\,P = 0_{n\times n}$$

An orthogonal projector is idempotent and symmetric.

We also note the Pythagorean identity

$$\|x\|_2^2 \equiv \|P x\|_2^2 + \|Q x\|_2^2$$

■

Exercise 7.3 [See Exercise 5.1][36]

Consider a decomposition $\mathbb{R}^n = K \oplus K^{\perp_A}$ analogously as above, with $(\cdot,\cdot)$ throughout replaced by $(\cdot,\cdot)_A$, and A-conjugate bases $\{u_1,\ldots,u_m\}$, $\{v_1,\ldots,v_{n-m}\}$, i.e., $(u_i,u_j)_A = \delta_{i,j}$, $(v_i,v_j)_A = \delta_{i,j}$, and $(u_i,v_j)_A = 0$. I.e., U and V are A-conjugate matrices, satisfying $U^{\mathrm{T}}AU = I_{m\times m}, V^{\mathrm{T}}AV = I_{(n-m)\times(n-m)}$, and $U^{\mathrm{T}}AV = 0_{m\times(m-n)}$, $V^{\mathrm{T}}AU = 0_{(n-m)\times m}$.

Show that the corresponding pair (P,Q) of 'A-conjugate' projectors onto K and $K^{\perp_A}$ is given by

$$P = U U^{\mathrm{T}} A, \quad Q = V V^{\mathrm{T}} A$$

and P and Q satisfy

$$P^2 = P = P^A, \quad Q^2 = Q = Q^A, \quad P\,Q = Q\,P = 0_{n\times n}$$

where M^A is the adjoint of a matrix $M \in \mathbb{R}^{n\times n}$ with respect to $(\cdot,\cdot)_A$, i.e. (see (5.16))

$$M^A = A^{-1} M^{\mathrm{T}} A$$

Check the Pythagorean identity

$$\|x\|_A^2 \equiv \|P x\|_A^2 + \|Q x\|_A^2$$

■

[36] For $x \in \mathbb{R}^n$ one may also denote $x^A = x^{\mathrm{T}}A = (Ax)^{\mathrm{T}}$, with $(x,x)_A = x^A x$, but this notation is not standard and we do not use it in the following.

8 The Conjugate Gradient (CG) Method for SPD Systems

8.1 Motivation

The Conjugate Gradient method is the example *par excellence* of a *Krylov subspace method*. The idea underlying this class of methods can, e.g., be motivated by the Cayley-Hamilton Theorem, which allowed us in Sec. 1.4 to construct the inverse of the matrix A as a polynomial in A (with coefficients depending on A). Let us write the solution x_* of $Ax = b$ in the form

$$x_* = x_0 + A^{-1} r_0$$

with $r_0 = b - Ax_0$. In principle, we can construct $A^{-1} r_0$ using the Cayley-Hamilton Theorem as

$$A^{-1} r_0 = \left(d_1 I + d_2 A + \ldots + d_{n-1} A^{n-1}\right) r_0 = q_{n-1}(A)\, r_0 \tag{8.1a}$$

where the coefficients are defined in terms of the characteristic polynomial $\chi(A)$. However, this is not feasible in practice. We note that this formula implies

$$A^{-1} r_0 \in \mathcal{K}_n = \operatorname{span}\{r_0, Ar_0, A^2 r_0 \ldots, A^{n-1} r_0\} \tag{8.1b}$$

and therefore the solution x_* satisfies

$$x_* \in x_0 + \mathcal{K}_n$$

More generally, for $m \geq 1$ we define the m-th *Krylov space* of A with respect to an initial residual r_0 as

$$\mathcal{K}_m = \mathcal{K}_m(A, r_0) = \operatorname{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}, \quad \text{with} \quad \dim(K_m) \leq m \tag{8.2}$$

Krylov subspace methods aim at finding, for some $m \ll n$, an approximation

$$x_m \in x_0 + \mathcal{K}_m$$

which is close to the exact solution x_*.

We start with the most prominent and historically earliest example, the Conjugate Gradient (CG) method for SPD systems,[37] developed by Hestenes, Stiefel, and Lanczos from 1950 on. The CG method is, compared to SD, a more clever descent method for the minimization of $\phi(x) = \frac{1}{2}(Ax, x) - (b, x)$.

8.2 Introduction to the CG method

Let $A \in \mathbb{R}^{n \times n}$ be SPD. The CG method may be motivated and described in different ways (we come back to this later on). A first idea is to try to proceed in a similar way as in the Steepest Descent method, but using search directions which are – in contrast to (7.9) – A-orthogonal[38] to each other, i.e.,

$$(d_{k+1}, d_k)_A = (Ad_{k+1}, d_k) = 0, \quad k = 0, 1, \ldots \tag{8.3}$$

with $d_0 := r_0$. Let us first motivate this choice.

[37] The CG method was elected as on of the 'Top 10 Algorithms of the 20th Century' by a SIAM committee in 2000.

[38] Instead of 'A-orthogonal' we will also use the term 'A-conjugate' or simply 'conjugate'. 'Orthogonal' means 'ℓ_2-orthogonal'.

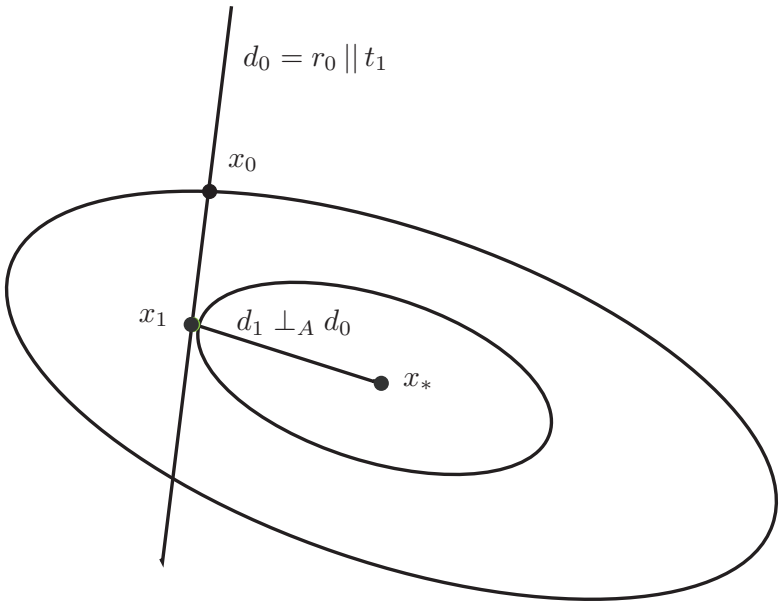


Figure 8.1: The CG method for $n = 2$.

The special case $n = 2$: geometric interpretation

For $n = 2$, the contour lines $C_{\phi\,=\text{const.}}$ of $\phi(x) = \frac{1}{2}(Ax, x) - (b, x)$ are ellipses[39] centered at x_*. Suppose $x_1 \in C = C_{\phi\,=\text{const.}}$ is obtained by one SD step starting from $x_0 \in \mathbb{R}^2$, with $r_1 = b - Ax_1 \perp r_0$. We do not know the error $e_1 = x_1 - x_*$, but we know the direction tangential to C: For a local parametrization $x = x(s)$ of C (with $x(0) = x_1$) we have

$$0 \equiv \frac{d}{ds}\phi(x(s)) = \big(\nabla\phi(x(s)), x'(s)\big)$$

Hence the tangential vector $t_1 := x'(0)$ to C at x_1 satisfies

$$t_1 \perp \nabla\phi(x_1) = Ax_1 - b = Ae_1, \quad \text{i.e.,} \quad t_1 \perp r_1, \quad t_1 \perp_A e_1, \quad \text{and} \quad t_1 \parallel r_0$$

In particular, t_1 is A-conjugate to the error e_1. Now, a single line search *along the direction A-conjugate to t_1* will yield the exact solution x_*, because the directions of t_1 and of Ae_1 correspond to a pair of *conjugate diameters* of the ellipse C, such that Ae_1 points in the direction of the center x_* of the ellipse. We can realize this procedure in an explicit way (see Fig. 8.1):

(i) Choose $d_0 = r_0$ and perform a line search along d_0 (first step as in the SD method),

$$x_1 = x_0 + \alpha_0 d_0, \quad \text{with} \quad \alpha_0 = \frac{(d_0, r_0)}{\|d_0\|_A^2}$$

and evaluate the new residual

$$r_1 = b - Ax_1 = r_0 - \alpha_0 A d_0$$

(ii) Instead of choosing $r_1 \perp r_0$ as the new search direction as in SD, we apply one step of Gram-Schmidt orthogonalization w.r.t. $(\cdot,\cdot)_A$ to construct a search vector $d_1 \perp_A d_0$, i.e.,

$$d_1 = r_1 + \beta_0 d_0, \quad \text{with} \quad \beta_0 = -\frac{(d_0, r_1)_A}{\|d_0\|_A^2}$$

[39] Note that, due to (7.2), $\|e\|_A = \text{const.}$ along each contour.

As argued above, the exact solution x_* is now given by

$$x_2 = x_1 + \alpha_1 d_1, \quad \alpha_1 = \frac{(d_1, r_1)}{\|d_1\|_A^2}$$

For a formal, algebraic proof of this fact we note that x_2 has the form

$$x_2 = x_0 + \alpha_0 d_0 + \alpha_1 d_1 = x_0 + \frac{(d_0, r_0)}{\|d_0\|_A^2} d_0 + \frac{(d_1, r_1)}{\|d_1\|_A^2} d_1 \tag{8.4a}$$

with $\alpha_0 d_0 \in \mathcal{K}_1(A, r_0)$ and $\alpha_0 d_0 + \alpha_1 d_1 \in \mathcal{K}_2(A, r_0)$. Now we compare this with the Fourier representation of $x_* - x_0$ in the A-conjugate basis $\{d_0, d_1\}$,

$$\begin{aligned} x_* &= x_0 + \frac{(x_* - x_0, d_0)_A}{\|d_0\|_A^2} d_0 + \frac{(x_* - x_0, d_1)_A}{\|d_1\|_A^2} d_1 \\ &= x_0 + \frac{(d_0, b - A x_0)}{\|d_0\|_A^2} d_0 + \frac{(d_1, b - A x_0)}{\|d_1\|_A^2} d_1 \\ &= x_0 + \frac{(d_0, r_0)}{\|d_0\|_A^2} d_0 + \frac{(d_1, r_0)}{\|d_1\|_A^2} d_1 \\ &= x_0 + \frac{(d_0, r_0)}{\|d_0\|_A^2} d_0 + \frac{(d_1, r_1 + \alpha_0 A d_0)}{\|d_1\|_A^2} d_1 \\ &= x_0 + \frac{(d_0, r_0)}{\|d_0\|_A^2} d_0 + \frac{(d_1, r_1)}{\|d_1\|_A^2} d_1 \end{aligned} \tag{8.4b}$$

This is identical with (8.4a), which shows that the procedure indeed yields the exact solution x_* after two steps. Note that the basis $\{d_0, d_1\}$ was not given a priori but was constructed in course of the iteration by one step of Gram-Schmidt orthogonalization.

The ingenious idea behind the CG method is that this procedure can be generalized in a very efficient way to arbitrary dimension n by means of a simple recursion. This results in a direct solution method for SPD systems which terminates latest at $x_n = x_*$. But we will also see that the x_m, $m = 0, 1, \ldots$ show a systematic convergence behavior vastly superior to the SD iterates.

Fourier expansion with respect to conjugate directions.

For $n = 2$ we have realized requirement (8.3) in a constructive way, ending up with a direct solution procedure $x_0 \to x_1 \to x_2 = x_*$. Before we discuss the generalization of this procedure leading to the CG method, we assume for the moment that we *already know* a pairwise A-conjugate basis

$$\{d_0, d_1, \ldots, d_{n-1}\} \text{ in } \mathbb{R}^n, \quad \text{with} \quad d_j \perp_A d_k \quad \text{for} \quad j \neq k$$

Then, for any given initial guess x_0, the solution x_* of $Ax = b$ can be written in terms of the Fourier expansion[40] of $x_* - x_0 = -e_0$ as (see (8.4b))

$$x_* = x_0 + \sum_{k=0}^{n-1} \frac{(-e_0, d_k)_A}{(d_k, d_k)_A} d_k = x_0 + \sum_{k=0}^{n-1} \frac{(d_k, r_0)}{\|d_k\|_A^2} d_k$$

[40] For the special case $x_0 = 0$ we have $e_0 = -x_*$ and $r_0 = b$.

We see that using the energy product enables an explicit Fourier representation of x_* in terms of r_0 and the d_k, where the required inner products can be expressed *without explicit reference to* x_*. This would not be possible for a representation w.r.t. a basis orthogonal in ℓ_2.

Now, for $m < n$, we consider the truncated Fourier expansion

$$x_m = x_0 + \sum_{k=0}^{m-1} \frac{(d_k, r_0)}{\|d_k\|_A^2} d_k =: x_0 + z_m \in x_0 + \operatorname{span}\{d_0, \dots, d_{m-1}\} =: x_0 + K_m \tag{8.5}$$

Here,

$$z_m = \sum_{k=0}^{m-1} \frac{(d_k, r_0)}{\|d_k\|_A^2} d_k = \sum_{k=0}^{m-1} \frac{(-e_0, d_k)_A}{\|d_k\|_A^2} d_k = P_m \cdot (-e_0) \in K_m$$

is the A-conjugate projection of $-e_0$ onto $K_m = \operatorname{span}\{d_0, \dots, d_{m-1}\}$ (see Exercise 7.3). By $P_m \colon \mathbb{R}^n \to K_m$ we denote the corresponding projector.

Thus, z_m satisfies the the *minimizing property*

$$z_m = \operatorname*{arg\,min}_{z \in K_m} \|z - (-e_0)\|_A$$

With $x_m = x_0 + z_m$ and $z_m - (-e_0) = (x_m - x_0) - (x_* - x_0) = x_m - x_*$ this shows that x_m is the solution of a minimization problem; namely,

$$x_m = \operatorname*{arg\,min}_{x \in x_0 + K_m} \|x - x_*\|_A \tag{8.6}$$

For the error $e_m = x_m - x_*$ we have

$$e_m = e_0 + z_m = (I - P_m)\, e_0 =: Q_m e_0 \perp_A K_m \tag{8.7}$$

Thus, e_m is the A-conjugate projection of e_0 onto the complement $K_m^{\perp_A} = \operatorname{span}\{d_m, \dots, d_{n-1}\}$ of K_m, which becomes 'smaller' with increasing m.

The corresponding matrix representation is obtained as follows: With the normalized basis vectors $\tilde{d}_k = d_k / \|d_k\|_A$ and $\tilde{D}_m = \left(\tilde{d}_0 \,\middle|\, \dots \,\middle|\, \tilde{d}_{m-1} \right)$ we have $P_m = \tilde{D}_m \tilde{D}_m^{\mathrm{T}} A$, see Exercise 7.3. Thus,

$$x_m = x_0 + z_m = x_0 - P_m e_0 = x_0 - \tilde{D}_m \tilde{D}_m^{\mathrm{T}} A e_0 = x_0 + \tilde{D}_m \tilde{D}_m^{\mathrm{T}} r_0 \tag{8.8}$$

which is identical with (8.5).

Producing iterates x_m in this way may be called a 'method of conjugate directions', a simple and straightforward projection method. However, in view of practical realization we observe:

- We may not expect in general that a (problem-dependent) A-conjugate basis is a priori known.
- Such a basis may, in principle, be generated by a Gram-Schmidt process starting from an arbitrary basis.[41] However, for larger values of m storage requirements, in particular, will become restrictive, because all d_k must be stored in memory to perform the orthogonalization.

This raises two questions:

- Can we generate an A-conjugate basis[42] *on the fly* in course of an iteration $x_0 \to x_1 \to \dots$, e.g., from the successive residuals?
- If yes, can we limit the complexity (in terms storage and flops) of the resulting iterative process?

For the CG method described in the sequel, both goals are achieved in a very satisfactory way for the case of SPD systems. Here, the basis K_m will depend on the initial guess x_0.

[41] One might think about the question what it would mean to start from the Cartesian basis.

[42] Recall that the SD iteration on the fly generates a sequence of orthogonal search directions (residuals).

8.3 Derivation of the CG method

In its essence, the CG method may be viewed as a clever realization of the idea to expand with respect to conjugate directions. Starting from an initial guess x_0, the x_m are constructed as elements of the *affine Krylov spaces* $x_0 + \mathcal{K}_m$, $\mathcal{K}_m = \mathcal{K}_m(A, r_0)$ (see (8.1b)) of increasing dimension. Here, Krylov spaces come into play because, in each iteration step, one multiplication with A is performed to compute the current residual. We will see that the $\mathcal{K}_m$ are spanned by the successive residuals (gradients) r_k, as well as by successive 'conjugated gradients' d_k (the search directions) constructed from the r_k.

The start is the same as for $n = 2$, from x_0 with $0 \neq r_0 = b - Ax_0 \in \mathcal{K}_1$.

- First step $x_0 \to x_1$: Choose $d_0 = r_0 \in \mathcal{K}_1$ and perform a line search,

$$x_1 = x_0 + \alpha_0 d_0 \in x_0 + \mathcal{K}_1, \qquad \alpha_0 = \frac{(d_0, r_0)}{\|d_0\|_A^2} = \frac{\|r_0\|_2^2}{\|d_0\|_A^2} \neq 0 \tag{8.9a}$$

  and compute the new residual

$$r_1 = r_0 - \alpha_0 A d_0 \in \mathcal{K}_2 \tag{8.9b}$$

  Stop if $r_1 = 0$. Otherwise, $\{r_0, r_1\}$ is an orthogonal basis of $\mathcal{K}_2$, since the new residual r_1 is orthogonal to the previous search direction $d_0 = r_0$.

  As a preparation for the next iteration step, we compute the new search direction d_1 by the A-conjugate Gram-Schmidt orthogonalization step

$$d_1 = r_1 + \beta_0 d_0 \in \mathcal{K}_2, \qquad \beta_0 = -\frac{(d_0, r_1)_A}{\|d_0\|_A^2} = \frac{\|r_1\|_2^2}{\|r_0\|_2^2} \neq 0 \tag{8.9c}$$

  where the latter identity for β_0 follows from $(d_0, r_1)_A = \frac{1}{\alpha_0}(\alpha_0 A d_0, r_1) = \frac{1}{\alpha_0}(r_0 - r_1, r_1)$ (see (8.9b)) together with (8.9a) and $r_1 \perp r_0$. By construction, $\{d_0, d_1\}$ is an A-conjugate basis of $\mathcal{K}_2$.

- We now proceed by induction, which leads to a complete description of the CG algorithm and its essential properties. Each iteration step is analogous to step (ii) for the case $n = 2$, see (8.12a) below.

  For $m \geq 2$ we inductively assume that we have recursively computed x_k, r_k and d_k for $k = 1 \ldots m-1$ by successive line search in the same way as for the first step, and that the following identities hold true:

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1} \in x_0 + \mathcal{K}_k, \qquad \alpha_{k-1} = \frac{(d_{k-1}, r_{k-1})}{\|d_{k-1}\|_A^2} = \frac{\|r_{k-1}\|_2^2}{\|d_{k-1}\|_A^2} \neq 0 \tag{8.10a}$$

  with the residuals

$$r_k = r_{k-1} - \alpha_{k-1} A d_{k-1} \in \mathcal{K}_{k+1} \tag{8.10b}$$

  and the line search directions

$$d_k = r_k + \beta_{k-1} d_{k-1} \in \mathcal{K}_{k+1}, \qquad \beta_{k-1} = -\frac{(d_{k-1}, r_k)_A}{\|d_{k-1}\|_A^2} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \neq 0 \tag{8.10c}$$

  We also inductively assume that, for $k = 1 \ldots m-1$,

$$\begin{aligned} &\{r_0, r_1, \ldots, r_k\} \text{ is an orthogonal basis of } \mathcal{K}_{k+1} \\ &\{d_0, d_1, \ldots, d_k\} \text{ is an } A\text{-conjugate basis of } \mathcal{K}_{k+1} \end{aligned} \tag{8.11a}$$

  which automatically implies that all the $\mathcal{K}_k$ are of maximal dimension k.

  In particular, note that

$$r_{m-1} \perp \mathcal{K}_{m-1}, \qquad d_{m-1} \perp_A \mathcal{K}_{m-1} \tag{8.11b}$$

- Now we take a step $m-1 \to m$ and verify that all the properties assumed inductively remain valid.

  The next iterate is defined via the current search direction d_{m-1} (already determined in the preceding step),

$$x_m = x_{m-1} + \alpha_{m-1}\, d_{m-1} \in x_0 + \mathcal{K}_m, \qquad \alpha_{m-1} = \frac{(d_{m-1}, r_{m-1})}{\|d_{m-1}\|_A^2} = \frac{\|r_{m-1}\|_2^2}{\|d_{m-1}\|_A^2} \neq 0 \tag{8.12a}$$

  The first expression for α_{m-1} comes from the line search (minimization of ϕ along $x_{m-1} + \alpha\, d_{m-1}$). The second identity for α_{m-1} follows inductively from the definition of r_{m-1} and d_{m-1} together with $r_{m-1} \perp d_{m-2}$ (orthogonality of the new residual with respect to the old search direction in line search):

$$(d_{m-1}, r_{m-1}) = (r_{m-1} + \beta_{m-2}\, d_{m-2}, r_{m-1}) = (r_{m-1}, r_{m-1})$$

  The new residual evaluates to

$$r_m = r_{m-1} - \alpha_{m-1}\, A\, d_{m-1} \in \mathcal{K}_{m+1} \tag{8.12b}$$

  If $r_m = 0$, the iteration stops with $x_m = x_*$. Otherwise, we again have $0 \neq r_m \perp d_{m-1}$, but even more:

  Claim: $r_m \perp \mathcal{K}_m$.

  Proof: Since, by construction, $r_m \perp d_{m-1}$ and $\mathcal{K}_m = \mathcal{K}_{m-1} \oplus \operatorname{span}\{d_{m-1}\}$ it remains to show that $r_m \perp \mathcal{K}_{m-1}$. The vector r_m is a linear combination of r_{m-1} and $A\, d_{m-1}$. From (8.11b) we have

$$r_{m-1} \perp \mathcal{K}_{m-1}, \qquad A\, d_{m-1} \perp \mathcal{K}_{m-1}$$

  which immediately yields $r_m \perp \mathcal{K}_{m-1}$, as proposed.

  Now we perform the same A-conjugate Gram-Schmidt orthogonalization step as before to construct a new search direction d_m which is conjugate to d_{m-1}:

$$d_m = r_m + \beta_{m-1}\, d_{m-1} \in \mathcal{K}_{m+1}, \qquad \beta_{m-1} = -\frac{(d_{m-1}, r_m)_A}{\|d_{m-1}\|_A^2} = \frac{\|r_m\|_2^2}{\|r_{m-1}\|_2^2} \neq 0 \tag{8.12c}$$

  where the latter identity for β_{m-1} follows from $(d_{m-1}, r_m)_A = \frac{1}{\alpha_{m-1}} (\alpha_{m-1}\, A\, d_{m-1}, r_m) = \frac{1}{\alpha_{m-1}} (r_{m-1} - r_m, r_m)$ (see (8.12b)) together with (8.12a)) and $r_m \perp r_{m-1}$.

  Claim: $d_m \perp_A \mathcal{K}_m$.

  Proof: Since, by construction, $d_m \perp_A d_{m-1}$ and $\mathcal{K}_m = \mathcal{K}_{m-1} \oplus \operatorname{span}\{d_{m-1}\}$ it remains to show that $d_m \perp_A \mathcal{K}_{m-1}$. The vector d_m is a linear combination of r_m and d_{m-1}. From (8.11b) we have $d_{m-1} \perp_A \mathcal{K}_{m-1}$. Concerning r_m, we *exploit the symmetry of* A and identity (8.10b) to compute, for an arbitrary basis vector d_{k-1} of $\mathcal{K}_{m-1}$, i.e., for $k = 1 \ldots m-1$:

$$(r_m, d_{k-1})_A = (A r_m, d_{k-1}) = \tfrac{1}{\alpha_{k-1}} (r_m, \alpha_{k-1}\, A\, d_{k-1}) = \tfrac{1}{\alpha_{k-1}} (r_m, \underbrace{r_{k-1} - r_k}_{\in \mathcal{K}_m}) = 0, \quad k = 1 \ldots m-1$$

  Together with $r_m \perp \mathcal{K}_m$, which has been proved before, this yields $d_m \perp_A \mathcal{K}_{m-1}$, as proposed.

  This completes the induction. In particular, we again have (see (8.11a))

$$\begin{aligned} &\{r_0, r_1, \ldots, r_m\} \text{ is an orthogonal basis of } \mathcal{K}_{m+1} \\ &\{d_0, d_1, \ldots, d_m\} \text{ is an } A\text{-conjugate basis of } \mathcal{K}_{m+1} \end{aligned} \tag{8.13}$$

Algorithm 8.1 Conjugate Gradient (CG) method

% input: A SPD, b, x_0

Compute $r_0 = b - A x_0$ and set $d_0 = r_0$
for $k = 0, 1, \ldots,$ until convergence **do**
  $\alpha_k = (r_k, r_k)/(A d_k, d_k)$
  Compute $x_{k+1} = x_k + \alpha_k d_k$
  Compute $r_{k+1} = r_k - \alpha_k A d_k$
  if $r_{k+1} = 0$ **then** stop
  $\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k)$
  Compute $d_{k+1} = r_{k+1} + \beta_k d_k$
end for

Remark 8.1

- In the last step of the proof, the symmetry of A plays an essential role (apart from the fact that the SPD property of A ensures that the conjugation process is well-defined and does nor break down unless a zero residual is encountered).

  The single, 'local' orthogonalization step within each iteration automatically leads to complete orthogonal and conjugate bases of the Krylov spaces according to (8.13). This is in sharp contrast to a general Gram-Schmidt procedure, where all orthogonality relations have to be explicitly enforced – a global process.

  This short recurrence generating a sequence of conjugate vectors is formally closely related to short recurrences generating orthogonal polynomials of a real variable t (see, e.g., [2, Chap. 6]). In this analogy, multiplication with the real variable $t = \bar{t}$ raising the power of a polynomial parallels multiplication with the symmetric matrix $A = A^{\mathrm{T}}$ raising the dimension of the Krylov subspace. See also Sec. 9.3 below.

  Later on we give an alternative explanation of this outstanding feature of the CG iteration.

- Actually, 'Conjugate Gradients' is a misnomer (a translation error?): The gradients (residuals) r_k are not conjugate but orthogonal; they are conjugate**d** to become A-conjugate (in form of the d_k).

- The CG algorithm is a very simple recurrence formulated in Alg. 8.1. There exist alternative formulations of this iteration; e.g., the recurrence for the x_k can be extracted from the recurrence for the r_k without explicit reference to the d_k; cf. e.g. [19].

- By construction, $x_k \in x_0 + \mathcal{K}_k = x_0 + \mathrm{span}\{r_0, A r_0, A^2 r_0, \ldots, A^{k-1} r_0\}$; the successive powers emanate from the computation of the residual (8.12b) in each step, which enters the definition of d_k and x_{k+1}.

  In the absence of roundoff error, there are two possibilities:

  – If a zero residual $r_k = 0$ is 'luckily' encountered in course of the iteration, the exact solution $x_k = x_* \in x_0 + \mathcal{K}_k$ has been found.
  – Otherwise, the $\mathcal{K}_k$ form an increasing sequence of subspaces of increasing dimension k,

$$x_k - x_0 \in \mathcal{K}_k = \mathrm{span}\{r_0, A r_0, A^2 r_0, \ldots, A^{k-1} r_0\} = \mathrm{span}\{r_0, \ldots, r_k\} = \mathrm{span}\{d_0, \ldots, d_k\}$$

  This shows that, in principle, CG is a direct method: The exact solution x_* is found after n steps at the latest. However, we will see that we can prove a estimate for the error $e_k = x_k - x_*$ after k steps, similarly for the SD method (but with a significantly better quality); see Sec. 8.5 below.

- The computational complexity of CG is usually dominated by the residual evaluations, in form of a single computation of $A d_k$ per step. In many practical situations the cost for evaluating the residual is $\mathcal{O}(n)$ and comparable to the other vector operations involved.

  The storage requirements are moderate, namely also $\mathcal{O}(n)$; only a few vectors have to be kept in memory at the same time.

- As we have seen, the orthogonality/conjugacy relations (8.13) are inherent to the CG iteration, but only if exact arithmetic is assumed. In practice, these relations will be contaminated by roundoff error, with a certain effect on the convergence behavior. ∎

8.4 CG as a projection method and its relation to polynomial approximation

Let us recapitulate the essential approximation properties of the CG iterates.

- In Sec. 7, our starting point was to interpret the solution x_* as the minimizer of the quadratic form

$$\phi(x) = \tfrac{1}{2}(Ax, x) - (b, x) = \phi(x_*) + \tfrac{1}{2}\|x - x_*\|_A^2$$

  see (7.1a),(7.2). The CG method realizes an iterative line search starting from x_0 and locally minimizing $\phi(x)$, in a sequential fashion, i.e., minimizing $\|x - x_*\|_A$ along successively constructed conjugate directions d_k emanating from x_{k-1}, with $\operatorname{span}\{d_0, \dots, d_{m-1}\} = \mathcal{K}_m$, resulting in (cf. (8.12a)):

$$x_m = x_0 + \sum_{k=0}^{m-1} \alpha_k d_k = x_0 + \sum_{k=0}^{m-1} \frac{(d_k, r_k)}{(d_k, d_k)_A} d_k \in x_0 + \mathcal{K}_m \tag{8.14}$$

  This is the same as (8.5), with $K_m = \mathcal{K}_m = \mathcal{K}_m(A, r_0)$, and thus, (8.6) holds true:

$$x_m = \underset{x \in x_0 + \mathcal{K}_m}{\arg\min} \|x - x_*\|_A \tag{8.15a}$$

  For the error $e_m = x_m - x_*$ this implies

$$e_m = \underset{e \in e_0 + \mathcal{K}_m}{\arg\min} \|e\|_A \tag{8.15b}$$

  which also shows monotonic convergence in the energy norm.

- Thus we have identified CG as a *projection method* – which was essentially our goal formulated in Sec. 8.2 – with the A-conjugate projector P_m onto $\mathcal{K}_m$ (cf. (8.7)),

$$x_m = x_0 - P_m(x_0 - x_*) = x_0 - P_m e_0 \in x_0 + \mathcal{K}_m$$

  For the error $e_m = x_m - x_*$ we have

$$e_m = (I - P_m)\, e_0 =: Q_m e_0$$

  with the A-conjugate projector Q_m onto $\mathcal{K}_m^{\perp_A}$.

- By construction, we have $e_m = e_0 - P_m e_0 \in e_0 + \mathcal{K}_m$. Thus we can write x_m and e_m in terms of *matrix polynomials*,

$$x_m = x_0 + p_{m-1}(A)\, r_0 \in x_0 + \mathcal{K}_m, \tag{8.16a}$$
$$e_m = e_0 + p_{m-1}(A)\, r_0 = (I - p_{m-1}(A)\, A)\, e_0 =: q_m(A)\, e_0 \tag{8.16b}$$

Here, $p_{m-1} \in \mathcal{P}_{m-1}$ is called the *CG polynomial*. Comparing (8.16a) with identity $x_* = x_0 + A^{-1} r_0$ we see that $p_{m-1}(A)$ is an approximation to A^{-1} which depends on x_0 and varies from step to step.

The optimality property (8.15b) can now be re-interpreted as a statement on the optimality of the matrix polynomial $q_m(A) = I - p_{m-1}(A)\, A$, where

$$q_m(\lambda) = 1 - p_{m-1}(\lambda)\lambda \in \mathcal{P}_m, \quad \text{with} \quad q_m(0) = 1 \tag{8.17}$$

We want $q_m(A)$ to be 'small' for good convergence.

We collect these findings in the following theorem.

Theorem 8.1 *The error $e_m = x_m - x_*$ of the CG iterate x_m is the A-conjugate projection of the initial error e_0 onto $\mathcal{K}_m^{\perp_A}$ along $\mathcal{K}_m = \mathcal{K}_m(A, r_0)$,*

$$e_m = Q_m\, e_0 = q_m(A)\, e_0 \perp_A \mathcal{K}_m \tag{8.18a}$$

and it satisfies

$$\|e_m\|_A = \min_{e \in e_0 + \mathcal{K}_m} \|e\|_A = \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \|q(A)\, e_0\|_A \le \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \|q(A)\|_A \cdot \|e_0\|_A \tag{8.18b}$$

The error is related to the the spectrum $\sigma(A)$ via

$$\frac{\|e_m\|_A}{\|e_0\|_A} \le \max_{\lambda \in \sigma(A)} |q_m(\lambda)| = \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \max_{\lambda \in \sigma(A)} |q(\lambda)| \tag{8.18c}$$

Proof: Relation (8.18c) follows from (8.18b) together with

$$\|q(A)\|_A = \max_{\lambda \in \sigma(A)} |q(\lambda)|$$

since $\|q(A)\|_A = \|q(A)\|_2 = \rho(q(A))$ for A SPD, and because the eigenvalues of $q(A)$ are $q(\lambda)$, $\lambda \in \sigma(A)$, cf. Exercise 6.1. □

Note that, in course of the iteration, the CG matrix polynomial $p_{m-1}(A)$ is not explicitly constructed in form of a matrix. It is 'inherent' to the process; the CG iteration realizes the appropriate image of r_0. Explicit computation of the projection matrices is useless and would also be numerically expensive.

The projection property of the CG iteration may also be stated in the following way, cf. Fig. 8.2.

Corollary 8.1 *The m-th CG iterate x_m is uniquely characterized as the solution of the projection problem*

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that the error}}\ e_m = x_m - x_*\ \textbf{\textit{satisfies}}\ e_m \perp_A \mathcal{K}_m\,. \tag{8.19a}$$

This is equivalent to Galerkin orthogonality,

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that its residual}}\ r_m = b - A x_m\ \textbf{\textit{satisfies}}\ r_m \perp \mathcal{K}_m\,. \tag{8.19b}$$

In Sec. 9 we will derive the CG method in an alternative fashion, by requiring the optimality property formulated in Theorem 8.1 or Corollary 8.1, respectively, and algorithmically realizing the corresponding projections, where, in a first step, orthogonal bases of the Krylov spaces are constructed. This procedure will also lead to a better insight to the simplicity of the CG procedure, and it will also lead us to more general projection methods to be studied in Sec. 10.

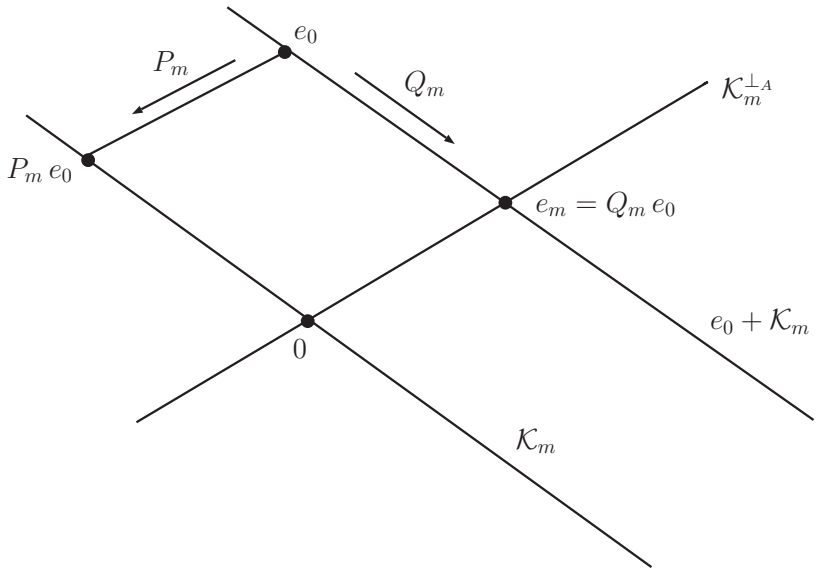


Figure 8.2: CG as a projection method

8.5 Convergence properties of the CG method

We are now in a position to analyze the convergence of the CG method and to derive a priori error estimates. To this end we may now choose any polynomial $q \in \mathcal{P}_m$ with $q(0) = 1$ to estimate the right hand side of (8.18c) from Theorem 8.1. A reasonable universal bound is obtained in a similar way as in Sec. 6: Assume $\sigma(A) \subseteq [\alpha, \beta]$ and seek a polynomial $p(\lambda)$ which attains the minimum

$$\min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \max_{\lambda \in [\alpha,\beta]} |q(\lambda)|$$

From Corollary 6.1 (with $\gamma = 0$) we know that this minimal q is given by a transformed Chebyshev polynomial, and this results in the error bound

$$\|e_m\|_A \leq \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \max_{\lambda \in \sigma(A)} |q(\lambda)|\, \|e_0\|_A \leq \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \max_{\lambda \in [\alpha,\beta]} |q(\lambda)|\, \|e_0\|_A$$

$$\leq 2\frac{c^m}{1+c^{2m}}\|e_0\|_A\,, \quad c = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \quad \kappa = \frac{\beta}{\alpha}$$

As a consequence, taking $[\alpha, \beta] = [\lambda_{min}, \lambda_{max}]$, and with the condition number $\kappa_2(A) = \lambda_{max}/\lambda_{min}$, we obtain

Theorem 8.2 *For the CG iteration applied to an SPD system $Ax = b$, the error in the energy norm is bounded by*

$$\|e_m\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^m \|e_0\|_A \tag{8.20}$$

This bound is similar to that obtained for the SD method, see Theorem 7.1, except that now the condition number of A is replaced by its square root! For large $\kappa_2(A)$ we have

$$\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1} \sim 1 - \frac{2}{\sqrt{\kappa_2(A)}}$$

and convergence to a specified tolerance may be expected after $\mathcal{O}(\sqrt{\kappa_2(A)})$ steps.

There is another major difference between SD and CG: While (with the exception of trivial starting vectors) the convergence behavior of the steepest descent method is accurately described by the condition number of the matrix (i.e., the ratio of the extremal eigenvalues), the whole distribution of the spectrum of A influences the convergence behavior of the CG-algorithm. In particular, the bound (8.20) is often too pessimistic.

Theorem 8.3 *If A has only $m < n$ distinct eigenvalues then, for any initial guess x_0 the CG iteration converges in at most m steps.*

Proof: Under the assumption of the theorem we may decompose the initial error e_0 in terms of the eigenbasis of A in the form[43]

$$e_0 = \sum_{i=1}^{m} \varepsilon_i v_i \in \operatorname{span}\{v_1, \ldots, v_m\} \tag{8.21}$$

where the v_i are certain eigenvectors of A (normalized in ℓ_2), with corresponding eigenvalues $\lambda_i > 0$, $i = 1 \ldots m$. Thus, the initial residual satisfies

$$r_0 = -A e_0 = -\sum_{i=1}^{m} \lambda_i \varepsilon_i v_i \in \operatorname{span}\{v_1, \ldots, v_m\}$$

and for *all* iterated residuals we have

$$A^k r_0 = -\sum_{i=1}^{m} \lambda_i^{k+1} \varepsilon_i v_i \in \operatorname{span}\{v_1, \ldots, v_m\} \quad k = 0, 1, \ldots$$

This shows

$$\mathcal{K}_k = \mathcal{K}_k(A, r_0) \subseteq \operatorname{span}\{v_1, \ldots, v_m\} \quad \text{for all} \quad k \geq 0$$

In particular, the dimension of the Krylov spaces

$$\mathcal{K}_k = \operatorname{span}\{r_0, A r_0, \ldots, A^{k-1} r_0\} = \operatorname{span}\{r_0, \ldots, r_{k-1}\}$$

does not grow beyond m. This shows that (at latest)

$$r_m = 0 \quad \Rightarrow \quad e_m = 0$$

because otherwise we would have $r_m \perp \mathcal{K}_m$ and $\dim(\mathcal{K}_{m+1}) = \dim(\operatorname{span}\{r_0, \ldots, r_{m-1}, r_m\}) = m + 1$, a contradiction. □

We may also express this property in terms of the optimal polynomial q_m inherent to CG (see (8.17)). Consider the characteristic polynomial of A,

$$\chi(z) = (z - \lambda_1)^{\nu_1} \cdots (z - \lambda_m)^{\nu_m}$$

where ν_i is the algebraic multiplicity of λ_i. The *minimal polynomial* $\mu(z)$ of A is defined as the monic polynomial of minimal degree which vanishes at the spectrum[44] of A. Now, by assumption, $\mu(z)$ is given by the polynomial of degree m,

$$\mu(z) = (z - \lambda_1) \cdots (z - \lambda_m), \qquad \mu(\lambda_i) = 0, \quad i = 1 \ldots m$$

and the right hand side of (8.18c) attains its minimal value 0 for the rescaled version of $\mu(z)$,

$$q(\lambda) = \left(1 - \tfrac{\lambda}{\lambda_1}\right) \cdots \left(1 - \tfrac{\lambda}{\lambda_m}\right), \qquad q(0) = 1$$

which shows $e_m = 0$.

[43] This is true because any linear combination of eigenvectors associated with an eigenvalue λ_i is again such an eigenvector in the eigenspace associated with λ_i.

[44] Analogously to $\chi(A) = 0$ (Cayley-Hamilton Theorem) we also have $\mu(A) = 0$.

Remark 8.2

- The argument in the proof of Theorem 8.3 also shows that the assertion remains true for arbitrary $A > 0$ if e_0 is a linear combination of $m < n$ eigenvectors of A, which is of course not a very practical assumption.
- From the above considerations we conclude that the CG iteration will rapidly converge of either A is well-conditioned (untypical!) or, by a heuristic argument, if the eigenvalues of A are concentrated in a few 'clusters'.
- Actually, the convergence properties of the CG iteration are more complex. In most cases, *superlinear* convergence is observed, i.e., the rate of convergence improves in course of the iteration. Theoretical explanations of this phenomenon are rather subtle.
- We also conclude that one way of improving the convergence behavior of the CG method is to reduce the condition number of the linear system by *preconditioning* (to be discussed in Sec. 12). This is by far the most popular method of improving the convergence behavior of the CG method. A preconditioner which 'bunches eigenvalues' can also improve the performance; however, this requires a very detailed knowledge of the structure of the matrix under consideration. ∎

8.6 CG in MATLAB: The function pcg

The following is a copy of the help page for the CG method implemented in MATLAB. Note that, alternatively to the matrix A, a function `AFUN` is sufficient which realizes the operation $x \mapsto Ax$. The function `pcg` also supports preconditioning in several variants, where the concrete preconditioner has to be provided by the user. As for A, the preconditioner may be specified in form of an evaluation function `MFUN`.

Since a posteriori error estimation is a difficult topic in general, the tolerance parameter `TOL` refers to the size of the 'backward error', i.e., the relative residual norm $\|r_k\|_2/\|b\|_2$.

```
PCG   Preconditioned Conjugate Gradients Method.

   X = PCG(A,B) attempts to solve the system of linear equations A*X=B for
   X. The N-by-N coefficient matrix A must be symmetric and positive
   definite and the right hand side column vector B must have length N.

   X = PCG(AFUN,B) accepts a function handle AFUN instead of the matrix A.
   AFUN(X) accepts a vector input X and returns the matrix-vector product
   A*X. In all of the following syntaxes, you can replace A by AFUN.

   X = PCG(A,B,TOL) specifies the tolerance of the method. If TOL is []
   then PCG uses the default, 1e-6.

   X = PCG(A,B,TOL,MAXIT) specifies the maximum number of iterations. If
   MAXIT is [] then PCG uses the default, min(N,20).

   X = PCG(A,B,TOL,MAXIT,M) and X = PCG(A,B,TOL,MAXIT,M1,M2) use symmetric
   positive definite preconditioner M or M=M1*M2 and effectively solve the
   system inv(M)*A*X = inv(M)*B for X. If M is [] then a preconditioner
   is not applied. M may be a function handle MFUN returning M\X.

   X = PCG(A,B,TOL,MAXIT,M1,M2,X0) specifies the initial guess. If X0 is
   [] then PCG uses the default, an all zero vector.
```

```
[X,FLAG] = PCG(A,B,...) also returns a convergence FLAG:
 0 PCG converged to the desired tolerance TOL within MAXIT iterations
 1 PCG iterated MAXIT times but did not converge.
 2 preconditioner M was ill-conditioned.
 3 PCG stagnated (two consecutive iterates were the same).
 4 one of the scalar quantities calculated during PCG became too
   small or too large to continue computing.
[X,FLAG,RELRES] = PCG(A,B,...) also returns the relative residual
NORM(B-A*X)/NORM(B). If FLAG is 0, then RELRES <= TOL.

[X,FLAG,RELRES,ITER] = PCG(A,B,...) also returns the iteration number
at which X was computed: 0 <= ITER <= MAXIT.

[X,FLAG,RELRES,ITER,RESVEC] = PCG(A,B,...) also returns a vector of the
estimated residual norms at each iteration including NORM(B-A*X0).

Example:
   n1 = 21; A = gallery('moler',n1);  b1 = A*ones(n1,1);
   tol = 1e-6;  maxit = 15;  M = diag([10:-1:1 1 1:10]);
   [x1,flag1,rr1,iter1,rv1] = pcg(A,b1,tol,maxit,M);
Or use this parameterized matrix-vector product function:
   afun = @(x,n)gallery('moler',n)*x;
   n2 = 21; b2 = afun(ones(n2,1),n2);
   [x2,flag2,rr2,iter2,rv2] = pcg(@(x)afun(x,n2),b2,tol,maxit,M);

See also bicg, bicgstab, bicgstabl, cgs, gmres, lsqr, minres, qmr,
symmlq, tfqmr, ichol, function_handle.
```

8.7 CGN: CG applied to the normal equations

As argued in Sec. 7.2 for the SD method, CG may in principle applied to arbitrary systems via solution of the normal equations

$$A^{\mathrm{T}}A\,x = A^{\mathrm{T}}\,b$$

where $A^{\mathrm{T}}A$ takes the role of A before. Again, this will be not a successful approach in general because $\sqrt{\kappa_2(A^{\mathrm{T}}A)} = \kappa_2(A)$. Moreover, the additional evaluations $x \mapsto A^{\mathrm{T}}x$ make each iteration step more expensive – or even unfeasible, if an evaluation procedure for A^{T} is not directly available.

Remark 8.3 One could also think of solving the normal equations my the convergent Gauss-Seidel or SOR method, see Theorem 5.3. This is closely related to the so-called *Kaczmarz iteration,* which is unconditionally convergent, but also with a very poor convergence rate in the typical case of an ill-conditioned matrix A. ∎

9 General Approach Based on Orthogonalization of $\mathcal{K}_m$. The Arnoldi and Lanczos Procedures

In this section we study the construction of orthogonal bases in Krylov spaces. The basic procedure is a variation of the well-known Gram-Schmidt algorithm, leading to a 'projected version' of a matrix $A \in \mathbb{R}^{n\times n}$ in form of a Hessenberg matrix. This construction is the basis for general Krylov subspace methods to be studied in the sequel.

9.1 The Arnoldi procedure for $A \in \mathbb{R}^{n\times n}$

A matrix $A \in \mathbb{R}^{n\times n}$ and a given vector $r_0 \in \mathbb{R}^n$ define a sequence of Krylov subspaces[45]

$$\mathcal{K}_m = \mathcal{K}_m(A, r_0) = \operatorname{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}, \quad m = 0, 1, 2, \ldots$$

We call

$$K_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ r_0 & Ar_0 & \vdots & A^{m-1}r_0 \\ | & | & & | \\ | & | & & | \end{array}\right) \in \mathbb{R}^{n\times m} \tag{9.1}$$

the corresponding *Krylov matrix*.

Constructing an orthonormal basis $\{v_1, \ldots, v_m\}$ of $\mathcal{K}_m$ is of interest on its own; it is a basic technique for what follows, and for general algorithms based on Krylov sequences. In principle, we know how to construct $\{v_1, \ldots, v_m\}$: Apply the Gram-Schmidt algorithm (or another equivalent orthonormalization procedure) to the Krylov vectors $r_0, Ar_0, \ldots$ (the columns of K_m). However, these are not given a priori but are to be computed *on the fly* in course of the orthonormalization process.

The so-called *Arnoldi iteration* is a clever implementation of this procedure. The resulting orthonormal vectors v_j are called *Arnoldi vectors:*

Choose $v_1 = r_0/\|r_0\|_2$. ***Then, for*** $j = 1 \ldots m$, ***multiply the current Arnoldi vector*** v_j ***by*** A ***and orthonormalize*** Av_j ***against all previous Arnoldi vectors*** $v_1, \ldots, v_j$, ***resulting in*** v_{j+1}.

This leads to the following iteration (with the intermediate unnormalized orthogonalized vectors w_j):

$$v_1 = r_0/\|r_0\|_2 \tag{9.2a}$$

and

$$w_j = Av_j - (Av_j, v_1)\,v_1 - (Av_j, v_2)\,v_2 - \ldots - (Av_j, v_j)\,v_j\,; \quad v_{j+1} = \frac{w_j}{\|w_j\|_2}, \quad j = 1, 2, \ldots \tag{9.2b}$$

This is also formulated in Alg. 9.1, where we define

$$h_{ij} = (Av_j, v_i), \quad i \le j, \quad \text{and} \quad h_{j+1,j} = \|w_j\|_2 \tag{9.3}$$

As long as $w_j \neq 0$ we have due to (9.2b) (for which, by construction, $w_j \perp \operatorname{span}\{v_1, \ldots, v_j\}$):

$$h_{j+1,j}^2 = (w_j, w_j) = (Av_j + \text{lin.comb. of } \{v_1, \ldots, v_j\}, w_j) = (Av_j, \|w_j\|_2 v_{j+1}) = h_{j+1,j}\,(Av_j, v_{j+1})$$

hence

$$\|w_j\|_2 = h_{j+1,j} = (Av_j, v_{j+1}) \tag{9.4}$$

[45] Assume for the moment that the $A^j r_0$ are linearly independent, such that $\mathcal{K}_m$ is of maximal dimension $m \le n$.

Algorithm 9.1 Arnoldi iteration (classical Gram-Schmidt variant)

1: $v_1 = r_0/\|r_0\|_2$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: **for** $i = 1, 2, \ldots, j$ **do**
4: Compute $h_{ij} = (A v_j, v_i)$
5: **end for**
6: Compute $w_j = A v_j - \sum_{i=1}^{j} h_{ij} v_i$
7: $h_{j+1,j} = \|w_j\|_2$
8: **if** $h_{j+1,j} = 0$ **then** stop
9: $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**

A *breakdown* occurs if a $w_j = 0$ is encountered. Provided the iteration does not break down, i.e., as long as $w_j \neq 0$, it generates the *Arnoldi matrix*

$$V_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ v_1 & v_2 & \vdots & v_m \\ | & | & & | \\ | & | & & | \end{array} \right) \in \mathbb{R}^{n \times m} \tag{9.5}$$

and the 'look-ahead' vector v_{m+1}. We also define the rectangular upper Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$,

$$\bar{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \ldots & h_{2m} \\ & h_{32} & h_{33} & \ldots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix} = \begin{pmatrix} (Av_1, v_1) & (Av_2, v_1) & (Av_3, v_1) & \ldots & (Av_m, v_1) \\ (Av_1, v_2) & (Av_2, v_2) & (Av_3, v_2) & \ldots & (Av_m, v_2) \\ & (Av_2, v_3) & (Av_3, v_3) & \ldots & (Av_m, v_3) \\ & & \ddots & \ddots & \vdots \\ & & & (Av_{m-1}, v_m) & (Av_m, v_m) \\ & & & & (Av_m, v_{m+1}) \end{pmatrix}$$

If breakdown occurs in the m-th step, $w_m = 0$ is still well-defined but not v_{m+1}, and the algorithm stops. In this case, the last row of $\bar{H}_m$ is zero, i.e., $h_{m+1,m} = 0$.

From the structure of V_m and $\bar{H}_m$ it is obvious that, from step to step, a further column is added to these matrices. The dimension of $\bar{H}_m$ increases in each step.

Lemma 9.1 *Assuming that Alg. 9.1 does not terminate prematurely, the vectors v_j, $j = 1 \ldots m$, form an orthonormal basis of the Krylov space $\mathcal{K}_m$, i.e., $V_m^T V_m = I_{m\times m}$. Furthermore, $P_m = V_m V_m^T \in \mathbb{R}^{n\times n}$ is the orthogonal projector onto $\mathcal{K}_m$.*

Proof: An exercise: An induction argument shows that the vectors v_j, $j = 1 \ldots m$, are indeed orthonormal. A second induction argument reveals $v_j \in \mathcal{K}_j$ for $j = 1 \ldots m$, and $P_m = V_m V_m^{\mathrm{T}}$ is the orthogonal projector onto $\mathcal{K}_m$ (see Remark 7.3). □

Since by Lemma 9.1, the Arnoldi vectors v_j are orthonormal and since $Av_j \in \operatorname{span}\{v_1, \ldots, v_{j+1}\}$ (see (9.2b)), each Av_j can also be expressed in terms of its Fourier expansion in terms of the v_i with $j+1$ terms.

E.g., for $j = 1$ we have

$$A v_1 = \underbrace{(A v_1, v_1)}_{h_{11}} v_1 + w_1 = \underbrace{(A v_1, v_1)}_{h_{11}} v_1 + \underbrace{(A v_1, v_2)}_{h_{21}} v_2$$

(see (9.4)). For general j, with $h_{ij} = (A v_j, v_i)$ we have

$$A v_j = \sum_{i=1}^{j} h_{ij} v_i + w_j = \sum_{i=1}^{j+1} h_{ij} v_i, \quad j = 1 \ldots m-1 \tag{9.6a}$$

A special case occurs for $j = m$ if, in the last step, $w_m = 0$ and v_{m+1} is not defined. But in any case, we have

$$A v_m = \sum_{i=1}^{m} h_{im} v_i + w_m \tag{9.6b}$$

from the last Arnoldi step in (9.2b), with $w_m = h_{m+1,m} v_{m+1}$ *if* v_{m+1} is well-defined.

In matrix notation, (9.6) is equivalent to

$$A V_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ A v_1 & A v_2 & \vdots & A v_m \\ | & | & & | \\ | & | & & | \end{array} \right) = V_{m+1} \bar{H}_m = V_m H_m + w_m e_m^{\mathrm{T}} \in \mathbb{R}^{n\times m} \tag{9.7}$$

where the square upper Hessenberg matrix $H_m \in \mathbb{R}^{m\times m}$ is obtained from $\bar{H}_m$ by removing its last row, and where

$$e_m = (0, 0, \ldots, 0, 1)^{\mathrm{T}} \in \mathbb{R}^m$$

denotes the m-th unit vector in $\mathbb{R}^m$.

We also conclude:

Theorem 9.1 *The Arnoldi procedure generates a reduced QR-decomposition of the Krylov matrix K_m (see (9.1)) in the form*

$$K_m = V_m R_m \tag{9.8}$$

with V_m from (9.5) satisfying $V_m^T V_m = I_{m\times m}$, and with an upper triangular matrix $R_m \in \mathbb{R}^{m\times m}$.

Furthermore, with the $m\times m$-upper Hessenberg matrix

$$H_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \ldots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \ldots & h_{2m} \\ & h_{32} & h_{33} & \ldots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \end{pmatrix} = \begin{pmatrix} (A v_1, v_1) & (A v_2, v_1) & (A v_3, v_1) & \ldots & (A v_m, v_1) \\ (A v_1, v_2) & (A v_2, v_2) & (A v_3, v_2) & \ldots & (A v_m, v_2) \\ & (A v_2, v_3) & (A v_3, v_3) & \ldots & (A v_m, v_3) \\ & & \ddots & \ddots & \vdots \\ & & & (A v_{m-1}, v_m) & (A v_m, v_m) \end{pmatrix}$$

we have

$$V_m^T A V_m = H_m \in \mathbb{R}^{m\times m} \tag{9.9}$$

Proof: The matrix R_m in (9.8) is implicitly defined by the orthogonalization process (9.2b): The first columns of K_m are linear combinations of the first j columns of V_m, which is equivalent to (9.8).

Furthermore, left multiplication of identity (9.7) by V_m^{T} yields

$$V_m^{\mathrm{T}} A V_m = V_m^{\mathrm{T}} V_m H_m + V_m^{\mathrm{T}} w_m e_m^{\mathrm{T}} = H_m,$$

due to the orthogonality relations $V_m^{\mathrm{T}} V_m = I_{m\times m}$ and $V_m^{\mathrm{T}} w_m = 0$. □

Remark 9.1 Now we see what the Arnoldi process accomplishes: Apart from computing an orthogonal basis of $\mathcal{K}_m$, it maps A to a smaller upper Hessenberg matrix H_m by an orthogonal transformation, see (9.9).

For $m = n$, the outcome would be identical with the well-known orthogonal similarity transformation to upper Hessenberg form (cf., e.g., [2]). For the case $m < n$ relevant here, this corresponds to a reduced version of such a decomposition, where the resulting small matrix $H_m = V_m^{\mathrm{T}} A V_m \in \mathbb{R}^{m\times m}$ is a *projected version of A acting in $\mathcal{K}_m$*: Consider $x \in \mathcal{K}_m$ and express it in the basis V_m, i.e., $x = V_m u_m$ with corresponding coefficient vector $u_m \in \mathbb{R}^m$. Applying A to x and projecting the result Ax back to $\mathcal{K}_m$ yields

$$V_m V_m^{\mathrm{T}} A x = V_m V_m^{\mathrm{T}} A V_m u_m = V_m H_m u_m \tag{9.10}$$

i.e., *the coefficient vector of the result expressed in the basis V_m is given by $H_m u_m$*. We can also write

$$V_m H_m V_m^{\mathrm{T}} = V_m V_m^{\mathrm{T}} A V_m V_m^{\mathrm{T}} = P_m A P_m =: A_m \in \mathbb{R}^{n\times n}$$

with the orthogonal projector $P_m = V_m V_m^{\mathrm{T}}$ onto $\mathcal{K}_m$. The matrix A_m is a rank-m approximation to A. ∎

The orthogonal basis delivered by the Arnoldi procedure will be used in Sec. 10 for the construction of approximate solutions x_m for general linear systems $Ax = b$.

Exercise 9.1 Show by means of an induction argument with respect to powers of A:

$$p(A)\, r_0 = V_m\, p(H_m) V_m^{\mathrm{T}} r_0 = \|r_0\|_2\, V_m\, p(H_m)\, e_1$$

for all $p \in \mathcal{P}_{m-1}$, where $e_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$ is the first unit vector in $\mathbb{R}^m$. ∎

9.2 The MGS (Modified Gram-Schmidt) variant

In practice, the classical Gram-Schmidt algorithm is usually implemented in an alternative, multiplicative way which is numerically more robust; this is known as the *Modified Gram-Schmidt algorithm* (MGS). Let us describe this variant in the context of the Arnoldi procedure.

Consider the j-step of the Arnoldi iteration (9.2b),

$$w_j = A v_j - (A v_j, v_1)\, v_1 - \ldots - (A v_j, v_j)\, v_j, \qquad v_{j+1} = w_j / \|w_j\|_2 \tag{9.11a}$$

where the $v_1, \ldots, v_j$ are already orthonormal. We have

$$(A v_j, v_i)\, v_i = (v_i^{\mathrm{T}} A v_j)\, v_i = v_i\, (v_i^{\mathrm{T}} A v_j) = (v_i v_i^{\mathrm{T}})\, A v_j = \hat{P}_i\, A v_j$$

with the orthogonal rank-1-projectors $\hat{P}_i = v_i v_i^{\mathrm{T}}$ onto $\mathrm{span}\{v_i\}$. Thus, (9.11a) is equivalent to

$$w_j = (I - \hat{P}_1 - \ldots - \hat{P}_j) A v_j = (I - P_j)\, A v_j = Q_j\, A v_j \tag{9.11b}$$

with the pair of orthogonal projectors

$$P_j = \hat{P}_1 + \ldots + \hat{P}_j = v_1 v_1^{\mathrm{T}} + \ldots + v_j v_j^{\mathrm{T}} = V_j V_j^{\mathrm{T}} \text{ projecting onto } \mathrm{span}\{v_1, \ldots, v_j\} = \mathcal{K}_j, \text{ and } Q_j = I - P_j$$

The idea behind MGS is to organize these successive projections in a different fashion. Let

$$\hat{Q}_i = I - \hat{P}_i = I - v_i v_i^{\mathrm{T}}, \quad i = 1 \ldots j$$

denote the rank-$(n-1)$-projectors onto the orthogonal complements $\big(\mathrm{span}\{v_i\}\big)^{\perp}$ along $\mathrm{span}\{v_i\}$.

Algorithm 9.2 Arnoldi iteration (Modified Gram-Schmidt variant)

```
v_1 = r_0/‖r_0‖_2
for j = 1, 2, ..., m do
    Initialize w_j = A v_j
    for i = 1, 2, ..., j do
        Compute h_ij = (w_j, v_i)
        Update w_j = w_j − h_ij v_i
    end for
    h_{j+1,j} = ‖w_j‖_2
    if h_{j+1,j} = 0 then stop
    v_{j+1} = w_j/h_{j+1,j}
end for
```

Due to the orthonormality of the v_i it is easy to verify by that

$$\begin{aligned}\hat{Q}_j \hat{Q}_{j-1} \cdots \hat{Q}_1 &= (I - v_j v_j^{\mathrm{T}})(I - v_{j-1} v_{j-1}^{\mathrm{T}}) \cdots (I - v_1 v_1^{\mathrm{T}}) \\ &= I - v_j v_j^{\mathrm{T}} - \ldots - v_1 v_1^{\mathrm{T}} = Q_j\end{aligned} \tag{9.12}$$

With this representation for Q_j, (9.2b) can be rewritten in form of a recursion in terms of the $\hat{Q}_j = (I - v_j v_j^{\mathrm{T}})$, replacing Q_j in (9.11b) by $\hat{Q}_j \hat{Q}_{j-1} \cdots \hat{Q}_1$ from (9.12):

$$\begin{aligned}
& & v_1 &= r_0/\|r_0\|_2 \\
w_1 &= (I - v_1 v_1^{\mathrm{T}}) A v_1, & v_2 &= w_1/\|w_1\|_2 \\
w_2 &= (I - v_2 v_2^{\mathrm{T}})(I - v_1 v_1^{\mathrm{T}}) A v_2, & v_3 &= w_2/\|w_2\|_2 \\
&\vdots & &\ddots \\
w_m &= (I - v_m v_m^{\mathrm{T}}) \cdots (I - v_1 v_1^{\mathrm{T}}) A v_m, & v_{m+1} &= w_m/\|w_m\|_2
\end{aligned}$$

Together with the identity $(I - v_i v_i^{\mathrm{T}})w = w - (w, v_i)v_i$ this leads to Alg. 9.2, which is mathematically equivalent to Alg. 9.1 but usually less sensitive to cancellation effects.

Remark 9.2 The Arnoldi iteration can also be realized via successive Housholder reflections, as in the classical, full orthogonal reduction of a square matrix to upper Hessenberg form. For details see [19]. ∎

9.3 The Lanczos procedure for symmetric $A \in \mathbb{R}^{n\times n}$

Assume $A = A^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ is symmetric, and apply the Arnoldi procedure (9.2) for given r_0. By symmetry, we immediately obtain from (9.9):

$$H_m = V_m^{\mathrm{T}} A V_m = H_m^{\mathrm{T}} \quad \text{is also symmetric}$$

Since $H_m \in \mathbb{R}^{m\times m}$ is upper Hessenberg (see Theorem 9.1), it must be tridiagonal. In this case we write

$$H_m =: T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}, \qquad \begin{aligned} \alpha_j &= h_{jj} = (A v_j, v_j), \\ \beta_j &= h_{j,j-1} = (A v_{j-1}, v_j) = (A v_j, v_{j-1}) \end{aligned} \tag{9.13}$$

Algorithm 9.3 Lanczos iteration

$\beta_1 = 0$, $v_0 = 0$
$v_1 = r_0/\|r_0\|_2$
for $j = 1, 2, \ldots, m$ **do**
  Initialize $w_j = A v_j - \beta_j v_{j-1}$
  Compute $\alpha_j = (w_j, v_j)$
  Update $w_j = w_j - \alpha_j v_j$
  $\beta_{j+1} = \|w_j\|_2$
  if $\beta_{j+1} = 0$ **then** stop
  $v_{j+1} = w_j/\beta_{j+1}$
end for

With this denotation, Alg. 9.2 specializes to the simple recursion formulated in Alg. 9.3.[46]

Exercise 9.2 Use the tridiagonal structure of the matrices T_m to conclude that $v_{j+1} \in \operatorname{span}\{v_{j-1}, v_j, A v_j\}$; specifically, verify the three-term recurrence

$$A v_j = \beta_{j+1} v_{j+1} + \alpha_j v_j + \beta_j v_{j-1} \tag{9.14}$$

Note: β_{j+1} can be obtained by normalizing $w_j = A v_j - \alpha_j v_j - \beta_j v_{j-1}$.

Remark: Due to this simple recursion, algorithms based on the Lanczos process can be organized in a way such that only three vectors need to be kept in memory at the same time. ∎

The Lanczos procedure is a simple three-term recurrence which we have obtained from the general (Arnoldi) case via symmetry, $A = A^{\mathrm{T}}$. For better insight, we now directly derive the Lanczos iteration from scratch. This parallels the derivation of a three-term recurrence for orthogonal polynomials of a real variable x as, e.g., described in [2, Lemma 6.4]. In the present context, multiplication by the symmetric matrix A is the analog of multiplication by $x \in \mathbb{R}$.

We start with $v_0 = 0$ and $v_1 = r_0/\|r_0\|_2$. Now, for given $m \geq 1$ we assume inductively that the orthonormal vectors $v_1, \ldots, v_m$ have been obtained. In order to find $v_{m+1} \perp \operatorname{span}\{v_1, \ldots, v_m\}$, we make the ansatz for the unnormalized version w_m of v_{m+1},

$$w_m = A v_m - \alpha_m v_m - \beta_m v_{m-1}$$

For arbitrary coefficients α_m and β_m we then have, exploiting the symmetry of A,

$$\begin{aligned}(w_m, v_j) &= (A v_m, v_j) - \alpha_m (v_m, v_j) - \beta_m (v_{m-1}, v_j) \\ &= (v_m, A v_j) - \alpha_m (v_m, v_j) - \beta_m (v_{m-1}, v_j)\end{aligned}$$

By induction, for $j = 1 \ldots m-2$ we have $(v_m, v_j) = (v_{m-1}, v_j) = 0$ and $A v_j \in \mathcal{K}_{j+1} \subseteq \mathcal{K}_{m-1} \perp v_m$, thus also $(v_m, A v_j) = 0$ holds. This shows that for arbitrary coefficients α_m and β_m our ansatz *automatically* satisfies

$$(w_m, v_j) = 0, \quad j = 1 \ldots m-2$$

In order to fix α_m and β_m it remains to require

$$\begin{aligned}0 \stackrel{!}{=} (w_m, v_m) &= (A v_m, v_m) - \alpha_m (v_m, v_m) - \beta_m (v_{m-1}, v_m) \\ &= (A v_m, v_m) - \alpha_m\end{aligned}$$

[46] In practice, the Lanczos iteration is quite sensitive to numerical loss of orthogonality due to round-off. Much research has been invested to fix this problem with reasonable effort by means of some form of re-orthogonalization; cf. [19].

and

$$0 \stackrel{!}{=} (w_m, v_{m-1}) = (A\,v_m, v_{m-1}) - \alpha_m\,(v_m, v_{m-1}) - \beta_m\,(v_{m-1}, v_{m-1})$$
$$= (A\,v_m, v_{m-1}) - \beta_m$$

This directly yields $\alpha_m = (A\,v_m, v_m)$ and $\beta_m = (A\,v_m, v_{m-1})$, hence

$$w_m = A\,v_m - (A\,v_m, v_m)\,v_m - (A\,v_m, v_{m-1})\,v_{m-1}$$

which leads to the three-term recurrence relation for the $v_{m+1} = w_m/\|w_m\|_2$ and the tridiagonal structure $H_m = T_m$, see (9.13).

Exercise 9.3 Extend this induction argument to show $\beta_m = \|w_{m-1}\|_2$. (We have already seen this identity in (9.4).) ∎

9.4 Arnoldi / Lanczos and polynomial approximation

Similarly as for the CG method (cf. Theorem 8.1), there is an intimate connection between the Arnoldi / Lanczos procedure and a polynomial approximation problem:

Arnoldi/Lanczos approximation problem:

Find a monic polynomial $p_m \in \mathcal{P}_m$ ***such that*** $\|p_m(A)\,r_0\|_2$ ***becomes minimal.*** (9.15)

The solution of this problem is characterized by the following theorem:

Theorem 9.2 *As long as the Arnoldi/Lanczos iteration does not break down (i.e., the Krylov matrix K_m has full rank), the approximation problem* (9.15) *has a unique solution p_m, which is given by the characteristic polynomial of H_m [T_m].*

Proof: Since $\mathcal{K}_m = \mathcal{K}_m(A, r_0) = \mathrm{span}\{r_0, A\,r_0, \ldots, A^{m-1}\,r_0\}$ and since the columns of V_m form a basis of $\mathcal{K}_m$, for any monic polynomial $p_m \in \mathcal{P}_m$ the vector $p_m(A)\,r_0$ can be written as

$$p_m(A)\,r_0 = A^m\,r_0 - V_m\,u_m \in A^m\,r_0 \oplus \mathcal{K}_m$$

with some coefficient vector $u_m \in \mathbb{R}^m$. Thus, (9.15) is equivalent to a linear least squares problem:

$$\textit{Find } u_m \in \mathbb{R}^m\textit{, i.e., } V_m\,u_m \in \mathcal{K}_m \textit{ such that } \|V_m\,u_m - A^m\,r_0\|_2 \textit{ becomes minimal.} \qquad (9.16)$$

Under the assumption of the theorem, i.e. if V_m has full rank m, the this least squares problem has a unique solution u_m, with a residual of the form $p_m(A) = A^m\,r_0 - V_m\,u_m$, characterized by the orthogonality relation[47]

$$V_m^{\mathrm{T}}\,(\underbrace{A^m\,r_0 - V_m\,u_m}_{=\,p_m(A)\,r_0}) = 0 \quad \Leftrightarrow \quad p_m(A)\,r_0 \perp \mathcal{K}_m \qquad (9.17)$$

In order to find p_m, we now consider the Arnoldi/Lanczos factorization $H_m = V_m^{\mathrm{T}}\,A\,V_m$ (see (9.9)). Due to $r_0, A\,r_0, \ldots, A^{m-1}\,r_0 \in \mathcal{K}_m$ and since $V_m V_m^{\mathrm{T}}$ projects onto $\mathcal{K}_m$, we have

$$V_m^{\mathrm{T}}\,A\,r_0 = V_m^{\mathrm{T}}\,A\,V_m V_m^{\mathrm{T}}\,r_0 = H_m V_m^{\mathrm{T}}\,r_0$$
$$V_m^{\mathrm{T}}\,A^2\,r_0 = V_m^{\mathrm{T}}\,A\,V_m V_m^{\mathrm{T}}\,A\,r_0 = V_m^{\mathrm{T}}\,A\,V_m V_m^{\mathrm{T}}\,A\,V_m V_m^{\mathrm{T}}\,r_0 = H_m^2\,V_m^{\mathrm{T}}\,r_0$$
$$\ldots$$
$$V_m^{\mathrm{T}}\,A^m\,r_0 = \ldots = H_m^m\,V_m^{\mathrm{T}}\,r_0$$

[47] (9.17) is the system of Gaussian normal equations for the least squares problem (9.16).

hence[48]

$$V_m^{\mathrm{T}}\, p_m(A)\, r_0 = p_m(H_m)\, V_m^{\mathrm{T}} r_0 \quad \text{for all} \quad p_m \in \mathcal{P}_m \tag{9.18}$$

Now we consider the characteristic polynomial χ_m of H_m. $\chi_m \in \mathcal{P}_m$ is monic and satisfies $\chi_m(H_m) = 0$ (Cayley-Hamilton). Together with (9.18) we conclude

$$0 = \chi_m(H_m)\, V_m^{\mathrm{T}} r_0 = V_m^{\mathrm{T}} \chi_m(A)\, r_0$$

Thus, $p_m = \chi_m$ satisfies (9.17), and we already know that this solution is unique. □

Remark 9.3 Due to Theorem 9.2, H_m is in a certain sense an approximation to A with the property that its characteristic polynomial χ_m is an approximation to χ, the characteristic polynomial of A, for which $\chi(A) = 0$ (Cayley-Hamilton). This approximation is optimal in the sense of (9.15).

Theorem 9.2 also provides a motivation for approximating $\sigma(A)$ by $\sigma(H_m)$, e.g. by starting the Arnoldi iteration from some random r_0. The eigenvalues of the H_m are also called *Ritz values*, and the method may be considered as a generalization of the ordinary power iteration. For details see e.g. [9],[11],[22]. Approximation results are e.g., available for the symmetric case; see Sec. 15. ∎

Exercise 9.4 Consider the projected matrix[49]

$$A_m = V_m H_m V_m^{\mathrm{T}} = P_m A P_m \in \mathbb{R}^{n\times n}, \qquad P_m = V_m V_m^{\mathrm{T}}$$

Show that each eigenvalue of H_m is also an eigenvalue of A_m, and all other eigenvalues of A_m are zero. ∎

9.5 Krylov spaces and matrix decompositions. The direct Lanczos method for symmetric systems (D-Lanczos)

As in Sec. 9.3 we now assume that A is symmetric (at this point, not necessarily SPD). The case of a general matrix A is considered in Sec. 10.

Since the columns of the Lanczos matrix V_m represent an orthonormal basis of $\mathcal{K}_m$, we can use this as the basis for a projection method, in the spirit of the ***Galerkin orthogonality requirement***[50]

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that its residual}}\ r_m = b - A x_m\ \textbf{\textit{satisfies}}\ r_m \perp \mathcal{K}_m\,. \tag{9.19}$$

which is identical with the characterization (8.19b) obtained for the CG iterates in Corollary 8.1.

We now realize a procedure for computing such an x_m. We make an ansatz for x_m in terms of the basis $V_m = \left(v_1 \,\middle|\, \ldots \,\middle|\, v_m\right)$ delivered by the Lanczos procedure,

$$x_m = x_0 + V_m u_m \tag{9.20a}$$

where the coefficient vector $u_m \in \mathbb{R}^m$ is to be determined. With

$$A x_m = A x_0 + A V_m u_m, \qquad r_m = b - A x_m = r_0 - A V_m u_m$$

[48] See Exercise 9.1 for a closely related identity.

[49] See also Remark 9.1.

[50] Note that (9.19) is *not* equivalent to the (also very reasonable) *minimal residual requirement* of minimizing $\|r_m\|_2$ over all possible $r_m \in r_0 + A\mathcal{K}_m$, which will be considered later on.

In the SPD case, (9.19) is also equivalent to minimizing $\|e_m\|_A$ over all possible error vectors $e_m \in e_0 + \mathcal{K}_m$, see characterization (8.19a) in Corollary 8.1.

enforcing the orthogonality condition (9.19) leads to the requirement

$$r_0 - AV_m u_m \perp \mathcal{K}_m, \quad \text{i.e,} \quad (r_0 - AV_m u_m, V_m v_m) = 0 \quad \text{for all } v_m \in \mathbb{R}^m$$

which is equivalent to the system of normal equations

$$V_m^{\mathrm{T}}(r_0 - AV_m u_m) = 0 \quad \Leftrightarrow \quad \underbrace{V_m^{\mathrm{T}} A V_m}_{=T_m} u_m = V_m^{\mathrm{T}} r_0 = \beta e_1, \quad \beta = \|r_0\|_2$$

with $e_1 = (1, 0, \ldots, 0)^{\mathrm{T}} \in \mathbb{R}^m$, because $v_1 = r_0/\|r_0\|_2$ (see (9.2b)). Thus, u_m is determined by the tridiagonal system

$$T_m u_m = \beta e_1 \tag{9.20b}$$

which is a projected version of the original system $Ax = b$ in the sense of (9.19).

If the Lanczos iteration does not break down, the system (9.20b) has a unique solution which can be computed via LU-decomposition of T_m, and then x_m is given by (9.20a). But if we do it in such a 'naive' way, the full matrix V_m has to stored in order to compute x_m. Alternatively, the computation of V_m, i.e., the Lanzcos iteration, can be repeated while assembling x_m.

The better option is to 'intertwine' the Lanczos iteration with the elimination process for the system (9.20b), i.e., orthonormalization and solving for u_m are performed simultaneously. This will lead us to an efficient iterative scheme. To this end, we first study the detailed recursive structure of the LU-decomposition of tridiagonal matrices of in the following exercise.

Exercise 9.5 Assume that for $m = 1, 2, \ldots$, the tridiagonal matrices

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix} \in \mathbb{R}^{m\times m}$$

from (9.13) admit LU-decompositions $T_m = L_m U_m$. Of course, L_m and U_m are bidiagonal, and we adopt the notation

$$T_m = L_m U_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \lambda_{m-1} & 1 & \\ & & & \lambda_m & 1 \end{pmatrix} \begin{pmatrix} \eta_1 & \omega_2 & & & \\ & \eta_2 & \omega_3 & & \\ & & \ddots & \ddots & \\ & & & \eta_{m-1} & \omega_m \\ & & & & \eta_m \end{pmatrix} \tag{9.21}$$

a) Verify the following recursive formulas for the values λ_m, ω_m, η_m in (9.21):

$$\omega_m = \beta_m, \qquad \lambda_m = \frac{\beta_m}{\eta_{m-1}}, \qquad \eta_m = \alpha_m - \lambda_m \omega_m \tag{9.22}$$

Conclude that the matrices L_m and U_m are recursively obtained from L_{m-1}, U_{m-1} by adding one row and column, i.e.,

$$L_m = \left(\begin{array}{cc|c} & & 0 \\ & L_{m-1} & \vdots \\ & & 0 \\ \hline 0 & \lambda_m & 1 \end{array}\right), \qquad U_m = \left(\begin{array}{cc|c} & & 0 \\ & U_{m-1} & \omega_m \\ & & \\ \hline 0 & & \eta_m \end{array}\right) \tag{9.23}$$

b) Given the factors $L, U \in \mathbb{R}^{m\times m}$ of the LU-decomposition of a matrix $T \in \mathbb{R}^{m\times m}$ in the form (9.23),

$$L = \left(\begin{array}{c|c} L' & 0 \\ \hline \ell^{\mathrm{T}} & 1 \end{array}\right), \qquad U = \left(\begin{array}{c|c} U' & u \\ \hline 0^{\mathrm{T}} & \eta \end{array}\right)$$

with $L', U' \in \mathbb{R}^{(m-1)\times(m-1)}$ and $\ell, u \in \mathbb{R}^{m-1}$, conclude that L^{-1} and U^{-1} can be written as

$$L^{-1} = \left(\begin{array}{c|c} L'^{-1} & 0 \\ \hline -\ell^{\mathrm{T}} L'^{-1} & 1 \end{array}\right), \qquad U^{-1} = \left(\begin{array}{c|c} U'^{-1} & -\frac{1}{\eta} U'^{-1} u \\ \hline 0^{\mathrm{T}} & \frac{1}{\eta} \end{array}\right)$$

This means that after adding one row and column to the tridiagonal matrix T to obtain the new tridiagonal matrix T', the triangular inverses L'^{-1}, U'^{-1} can be also obtained from L^{-1} and U^{-1} by simply adding one row and column. ∎

Combining the results of Exercise 9.5 with (9.20) leads to the representation of the iterates x_m in the form

$$x_m = x_0 + V_m u_m = x_0 + V_m U_m^{-1} L_m^{-1} \beta e_1$$

with L_m and U_m from (9.21). From the Lanczos iteration (Alg. 9.3) we have a short recurrence for the columns v_j of V_m. The matrices U_m^{-1} and L_m^{-1} can be computed according to Exercise 9.5. Hence, we expect to be able to find a short recurrence for the vectors x_m in form of an iteration of Krylov type. To this end we substitute

$$D_m = V_m U_m^{-1}, \qquad z_m = L_m^{-1} \beta e_1$$

and obtain

$$x_m = x_0 + D_m z_m \tag{9.24}$$

To derive a recurrence for the x_m, we first consider the vectors z_m: From Exercise 9.5 we obtain

$$z_m = L_m^{-1} \beta e_1 = \left(\begin{array}{c|c} L_{m-1}^{-1} & 0 \\ \hline \ell_m^{\mathrm{T}} & 1 \end{array}\right) \left(\begin{array}{c} \beta e_1 \\ \hline 0 \end{array}\right) = \left(\begin{array}{c} z_{m-1} \\ \hline \zeta_m \end{array}\right), \quad \text{with} \quad \zeta_m = \beta \ell_m^{\mathrm{T}} e_1 \tag{9.25a}$$

i.e., the vector z_m is obtained from z_{m-1} by adding one scalar entry, ζ_m.

Next, the explicit form of U_m and its inverse allows us to infer from Exercise 9.5:

$$D_m = V_m U_m^{-1} = \left(\begin{array}{c|c} V_{m-1} & v_m \end{array}\right) \left(\begin{array}{c|c} U_{m-1}^{-1} & u_m \\ \hline 0^{\mathrm{T}} & \frac{1}{\eta_m} \end{array}\right) = \left(\begin{array}{c|c} D_{m-1} & d_m \end{array}\right), \quad \text{with} \quad d_m = V_{m-1} u_m + \frac{1}{\eta_m} v_m \tag{9.25b}$$

i.e., the matrix D_m is obtained from D_{m-1} by adding one column, d_m.

Inserting these findings about D_m and z_m into (9.24), we conclude

$$x_m = x_0 + D_m z_m = x_0 + \left(\begin{array}{c|c} D_{m-1} & d_m \end{array}\right) \left(\begin{array}{c} z_{m-1} \\ \zeta_m \end{array}\right) = x_0 + D_{m-1} z_{m-1} + \zeta_m d_m = x_{m-1} + \zeta_m d_m, \quad \text{with} \quad \zeta_m = \beta \ell_m^{\mathrm{T}} e_1$$

with ζ_m and d_m from (9.25). In this way we have obtained a simple update formula for the iterates x_m, with certain search directions d_m and associated weights ζ_m. We now investigate in more detail how the d_m and the weights ζ_m can be computed efficiently.

The search direction d_m is the m-th column of $D_m = V_m U_m^{-1}$. It can be determined by considering the m-th column of the product $V_m = D_m U_m$ with U_m from (9.21). We have

$$V_{j,m} = \sum_{i=1}^{m} D_{j,i} U_{i,m} = D_{j,m-1} \omega_m + D_{j,m} \eta_m, \quad j = 1 \ldots m$$

This gives a recurrence for the search directions of the form

$$\eta_m d_m = v_m - \omega_m d_{m-1} \tag{9.26}$$

This formula is also correct for $m = 1$ provided we set $d_0 = 0$.

Furthermore, since z_m is found by forward substitution on the system

$$L_m z_m = \beta e_1$$

with L_m from (9.21), we have

$$z_m = \begin{pmatrix} z_{m-1} \\ \zeta_m \end{pmatrix}, \quad \text{with} \quad \zeta_m = -\lambda_m \zeta_{m-1} \tag{9.27}$$

i.e., the last element of the vector z_m is just the last element from z_{m-1} multiplied by $-\lambda_m$.

Finally, x_m is updated in each step according to

$$x_m = x_{m-1} + \zeta_m d_m \tag{9.28}$$

All this results in the so-called *D-Lanczos algorithm*, Alg. 9.4. It incorporates the following recurrences:

(i) the recurrence for the Lanczos vectors v_m together with the definition of the scalars α_m and β_m (see Alg. 9.3);

(ii) the formulas for the entries $\lambda_m, \omega_m, \eta_m$ of the LU-decomposition of T_m (see (9.22));

(iii) the recurrence for the search directions d_m, see (9.26);

(iv) the recurrence (9.27) for the weights ζ_m; and finally,

(v) the update formula (9.28) for the iterates x_m.

Algorithm 9.4 D-Lanczos

Compute $r_0 = b - Ax_0$; $\zeta_1 = \beta = \|r_0\|_2$; $v_1 = r_0/\beta$
$\lambda_1 = \beta_1 = 0$, $d_0 = 0$;
for $m = 1, 2, \ldots,$ **do**
  Compute $w_m = Av_m - \beta_m v_{m-1}$ and $\alpha_m = (w_m, v_m)$
  if $m > 1$ **then** compute $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ and $\zeta_m = -\lambda_m \zeta_{m-1}$
  $\eta_m = \alpha_m - \lambda_m \beta_m$
  $d_m = (v_m - \beta_m d_{m-1})/\eta_m$
  $x_m = x_{m-1} + \zeta_m d_m$
  if x_m has converged, **then** stop
  $w_m = w_m - \alpha_m v_m$; $\beta_{m+1} = \|w\|_2$; $v_{m+1} = w_m/\beta_{m+1}$
end for

Remark 9.4 The D-Lanczos algorithm relies on the symmetry of the matrix A, but it is not assumed that A be SPD. However, the above derivation *assumes* the existence of an LU-decomposition of the matrices T_m. If A is SPD, then we will see that this assumption is valid, and the D-Lanczos produces iterates identical to the CG iterates, as shown in the next section. If A is merely symmetric but indefinite, then it is possible that the D-Lanczos algorithm breaks down: It may happen that $\eta_m = 0$ in (9.21). ∎

9.6 From Lanczos to CG

The D-Lanczos algorithm relies on linking recurrences for the Krylov vectors v_m and the search directions d_m to each other by means of the entries of the LU-decomposition of the matrices T_m. We now show that this can be written in a simpler way, where the decomposition is not explicitly required. For SPD systems we will see that this is always possible, i.e., the iteration does not break down, and we show that the result is exactly the CG algorithm from Sec. 8.

Lemma 9.2 *Let $A \in \mathbb{R}^{n\times n}$ be symmetric. Let x_m, $m = 0, 1 \ldots$ be the sequence of approximations obtained by the D-Lanczos Algorithm 9.4 – we assume that the algorithm does not break down. Let $r_m = b - Ax_m$ be the sequence of residuals. Then:*

(i) $r_m = \sigma_m v_{m+1}$ *for some* $\sigma_m \in \mathbb{R}$;

(ii) the residuals r_m *are pairwise orthogonal, i.e.,* $(r_i, r_j) = 0$ *for* $i \neq j$;

(iii) the search directions d_m *are pairwise conjugate, i.e.,* $(Ad_i, d_j) = 0$ *for* $i \neq j$.

Proof:

- *ad (i):* Use $e_m = (0, 0, \ldots, 0, 1)^{\mathrm{T}} \in \mathbb{R}^m$, consider

$$\bar{T}_m = \begin{pmatrix} T_m \\ \hline t_{m+1,m}\, e_m^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{(m+1)\times m}$$

  satisfying $AV_m = V_{m+1}\bar{T}_m$ (see (9.7)), and compute the residual:[51]

$$\begin{aligned} r_m &= b - Ax_m = b - A(x_0 + V_m u_m) = r_0 - AV_m u_m = \beta v_1 - V_{m+1}\bar{T}_m u_m \\ &= V_m(\underbrace{\beta e_1 - T_m u_m}_{=0}) - v_{m+1}(t_{m+1,m}\, e_m^{\mathrm{T}} u_m) = \sigma_m v_{m+1} \end{aligned} \tag{9.29}$$

  for some scalar σ_m.

- *ad (ii):* Due to (9.29) the residuals r_m are multiples of the v_{m+1} which are pairwise orthogonal. Thus the residuals are orthogonal as well.

- *ad (iii):* Consider the matrix product

$$D_m^{\mathrm{T}} A D_m = U_m^{-\mathrm{T}} V_m^{\mathrm{T}} A V_m U_m^{-1} = U_m^{-\mathrm{T}} T_m U_m^{-1} = U_m^{-\mathrm{T}} L_m$$

  I.e., the lower triangular matrix $U_m^{-\mathrm{T}} L_m$ equals the symmetric matrix $D_m^{\mathrm{T}} A D_m$, which therefore must be a diagonal matrix. Thus, the vectors d_m form a conjugate set, i.e., $(Ad_i, d_j) = 0$ for $i \neq j$. □

We now assume that A is SPD. In this case,

$$T_m = V_m^{\mathrm{T}} A V_m \quad \text{is clearly also SPD,}$$

therefore the LU-decomposition $T_m = L_m U_m$ is well-defined. In particular, the η_m computed by the D-Lanczos algorithm do not vanish (cf. (9.21),(9.22),(9.26)).

Lemma 9.2 describes exactly the orthogonality relationships which we have established for the CG method in Sec. 8.3. This is not surprising, since the Galerkin orthogonality requirement (9.19) which led us to D-Lanczos is identical with (8.19b) (cf. Corollary 8.1) which characterizes the CG iterates. Therefore, the resulting iterates x_m must be identical. In particular, the D-Lanczos algorithm does not break down in the SPD case.

[51] Here we use the identity $AV_m = V_{m+1}\bar{T}_m$, see (9.7).

Remark 9.5

- Alternatively, the CG iteration can also be deduced from the Cholesky decomposition of the SPD matrix T_m, in a similar way as D-Lanczos was deduced from the LU-decomposition of T_m, see [18, Sec. 1.2.7].
- Comparing (8.16a) with (9.20) we see that, with the CG polynomial $p_{m-1}(A) \approx A^{-1}$, the CG iterates x_m can be written in the following equivalent ways:

$$\begin{aligned} x_m &= x_0 + V_m T_m^{-1} V_m^{\mathrm{T}} r_0 \\ &= x_0 + p_{m-1}(A) r_0 \\ &= x_0 + V_m p_{m-1}(T_m) V_m^{\mathrm{T}} r_0 \\ &\approx x_0 + A^{-1} r_0 = x_* \end{aligned} \tag{9.30}$$

- The superlinear convergence behavior of the CG iterates observed in practice can be explained in the following way (we give no details): We have $x_m = x_0 + V_m u_m$ where u_m is the solution of (9.20b. It can be shown that the extremal Ritz values, i.e., the eigenvalues of T_m, feature fast convergence towards the extreme eigenvalues of A. For the later iteration steps this means that the iterates behave in a way as if these eigenvalues were 'removed', entailing a smaller effective condition number. ∎

10 General Krylov Subspace Methods, in particular GMRES

We now assume that $A \in \mathbb{R}^{n\times n}$ is an arbitrary square matrix. In general, Krylov subspace approximations are defined via a projection property, which has the general form of a ***Petrov-Galerkin orthogonality requirement:***

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that its residual}}\ r_m = b - Ax_m\ \textbf{\textit{satisfies}}\ r_m \perp \mathcal{L}_m\,. \tag{10.1}$$

Here, the *ansatz space* (or solution space) is $\mathcal{K}_m = \mathcal{K}_m(A, r_0)$. Different Krylov methods differ in the choice of the *test space* $\mathcal{L}_m$. We will consider three choices:

(i) ***'Full orthogonalization methods'*** ('FOM' or 'OrthoRes'; e.g., D-Lanczos and conjugate gradient (CG) methods):

We choose $\mathcal{L}_m = \mathcal{K}_m$ (cf. (9.19)) and require Galerkin orthogonality,

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that its residual}}\ r_m = b - Ax_m\ \textbf{\textit{satisfies}}\ r_m \perp \mathcal{K}_m\,. \tag{10.2a}$$

With $r_m = -Ae_m = -A(x_m - x_*)$ this is equivalent to the requirement $Ae_m \perp \mathcal{K}_m$. In particular,

$$e_m \perp A\mathcal{K}_m \quad \text{for } A \text{ symmetric}, \qquad e_m \perp_A \mathcal{K}_m \quad \text{for } A \text{ SPD} \tag{10.2b}$$

If A is SPD, we already know from Sec. 8.4 that this has always a unique solution x_m satisfying

$$\|e_m\|_A = \|x_m - x_*\|_A = \min_{x\in x_0+\mathcal{K}_m} \|x - x_*\|_A = \min_{e\in e_0+\mathcal{K}_m} \|e\|_A \tag{10.2c}$$

i.e., the energy norm of the error is minimized over $e_0 + \mathcal{K}_m$.

(ii) ***'[Generalized] minimal residual methods'*** (MINRES, GMRES):

We choose $\mathcal{L}_m = A\mathcal{K}_m$, i.e.,

$$\textbf{\textit{Find}}\ x_m \in x_0 + \mathcal{K}_m\ \textbf{\textit{such that its residual}}\ r_m = b - Ax_m\ \textbf{\textit{satisfies}}\ r_m \perp A\mathcal{K}_m\,. \tag{10.3a}$$

This is equivalent to the requirement $Ae_m \perp A\mathcal{K}_m$ and, as shown below, it is equivalent to

$$\|r_m\|_2 = \|b - Ax_m\|_2 = \min_{x\in x_0+\mathcal{K}_m} \|b - Ax\|_2 = \min_{r\in r_0+A\mathcal{K}_m} \|r\|_2 \tag{10.3b}$$

which motivates the term 'minimal residual method'.

(iii) ***'Biorthogonal methods':***

Here, $\mathcal{L}_m = \mathcal{K}_m(A^{\mathrm{T}}, \widetilde{r}_0)$ for some $\widetilde{r}_0$. This choice will lead to the Biconjugate Gradient method (BiCG) and its variants.

The orthogonality conditions (10.2a), (10.3a) are illustrated in Fig. 10.1.

Lemma 10.1 *Conditions* (10.3a) *and* (10.3b) *are equivalent.*

Proof: This is directly related to the characterization of the solution of a linear least squares problem: For given x_0 with residual $r_0 = b - Ax_0$, we consider an arbitrary $x \in x_0 + \mathcal{K}_m$, with residual vector $r \in r_0 + A\mathcal{K}_m = r_0 + \mathcal{L}_m$. Let us write $r = r_0 + l$ with $l \in \mathcal{L}_m$. Furthermore, we decompose the initial residual according to

$$r_0 = P_m r_0 + Q_m r_0 \in \mathcal{L}_m \oplus \mathcal{L}_m^{\perp}$$

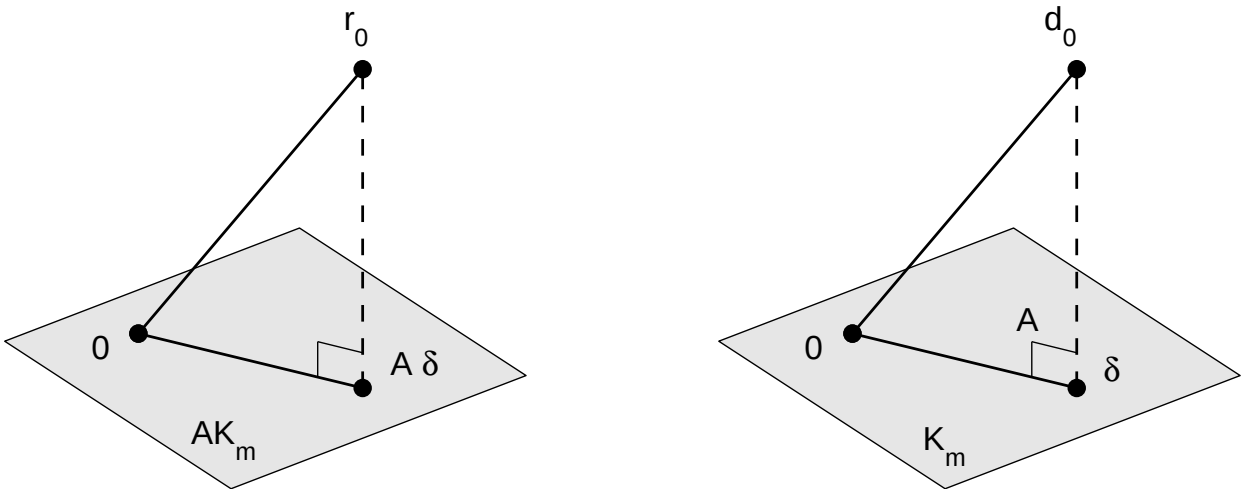


Figure 10.1: The orthogonality conditions (10.3a) and (10.2a)

where P_m is the orthogonal projector onto $\mathcal{L}_m$, and $Q_m = I - P_m$. Thus, r has the unique decomposition

$$r = (l + P_m r_0) + Q_m r_0 \in \mathcal{L}_m \oplus \mathcal{L}_m^\perp,$$

satisfying the Pythagoraean identity

$$\|r\|_2^2 = \|\, l + P_m r_0\|_2^2 + \|Q_m r_0\|_2^2$$

Since $Q_m r_0$ is a priori fixed, this attains its minimal value for $l = -P_m r_0$, i.e., the minimum is attained at

$$r = r_m = r_0 - P_m r_0 = Q_m r_0 \in \mathcal{L}_m^\perp \perp \mathcal{L}_m$$

which is exactly the Petrov-Galerkin orthogonality requirement $r_m \perp \mathcal{L}_m = A\mathcal{K}_m$.

This argument can also be reversed: If r satisfies $r \perp \mathcal{L}_m = A\mathcal{K}_m$, then $r = Q_m r$ and $\|r\|_2^2 = \|Q_m r\|_2^2 = \|Q_m r_0\|_2^2$, which is the minimal attainable value. □

For A invertible, the corresponding solution value x_m with $b - Ax_m = r_m$ is uniquely defined by the minimal value r_m: $x_m = A^{-1}(b - r_m)$. In the following we study the computation of x_m by means of a Krylov subspace method. Similarly as the D-Lanczos iteration was derived from the Lanczos procedure in Sec. 9.5, our starting point is the general Arnoldi procedure for general matrices A, see Sections 9.1 and 9.2.

10.1 Computing the x_m

As for all Krylov subspace methods, approximations x_m, $m = 0, 1, \ldots$ are computed until an approximation x_m is found which is sufficiently accurate. It is, of course, essential that the x_m can be computed efficiently from the orthogonality conditions (10.2a) or (10.3a). The general procedure is:

- Construct the $n \times m$ Arnoldi matrix $V_m = \left(v_1 \middle| \ldots \middle| v_m \right)$. By construction, its columns v_j form an orthogonal basis of $\mathcal{K}_m$.
- Construct the $n \times m$ matrix $W_m = \left(w_1 \middle| \ldots \middle| w_m \right)$ such that its columns w_j form a basis of $\mathcal{L}_m$. For $\mathcal{L}_m = A\mathcal{K}_m$ this is given by $W_m = AV_m$, which is generated in course of the Arnoldi procedure.

- Represent the approximate solution as

$$x_m = x_0 + V_m u_m \tag{10.4}$$

where $u_m \in \mathbb{R}^m$ is a vector of weights to be determined. Note that $r_m = r_0 - AV_m u_m$.

- Enforcing either of the orthogonality conditions (10.2a),(10.3a) leads to the system of normal equations

$$W_m^{\mathrm{T}}(b - Ax_m) = 0 \quad \Leftrightarrow \quad W_m^{\mathrm{T}} A V_m u_m = W_m^{\mathrm{T}} r_0 \tag{10.5a}$$

from which the approximate solution x_m follows in the form

$$x_m = x_0 + V_m u_m = x_0 + V_m \left(W_m^{\mathrm{T}} A V_m\right)^{-1} W_m^{\mathrm{T}} r_0 \tag{10.5b}$$

Note that the matrix $W_m^{\mathrm{T}} A V_m$ is only of size $m \times m$; therefore its inversion is affordable for $m \ll n$.

Exercise 10.1

a) Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Use (8.1b) to conclude that the approximation x_n obtained in the n-th step (10.4) according to the MINRES criterion (10.3b) is the exact solution x_* of $Ax = b$.

b) Assume additionally that for some $m \le n$ there holds $\mathcal{K}_m = \mathcal{K}_n$. Show that then already

$$x_m = x_{m+1} = \cdots = x_*$$

Hint: Show that $\mathcal{K}_k = \mathcal{K}_m = \mathcal{K}_n$ also for all $k > m$. ∎

Remark 10.1 One may also think of proceeding in the following way, motivated by Exercise 9.1: For all $p \in \mathcal{P}_{m-1}$ he have

$$V_m\, p(H_m)\, V_m^{\mathrm{T}} r_0 = p(A)\, r_0$$

Formally replacing $p(A)$ by A^{-1} would suggest the approximate identity

$$V_m H_m^{-1} V_m^{\mathrm{T}} r_0 \approx A^{-1} r_0$$

which gives

$$x_m := x_0 + V_m H_m^{-1} V_m^{\mathrm{T}} r_0 \approx x_0 + A^{-1} r_0 = x_*$$

A look at (10.5b) shows that this gives exactly a FOM method,[52] $W_m = V_m$. (Recall that $H_m = V_m^{\mathrm{T}} A V_m$.)

This type of FOM approximation is also often used in an analogous way for approximating other matrix functions $f(A)$ valuated at some given vector y, e.g., the matrix exponential $f(A) = \exp(A)$ (instead of $f(A) = A^{-1}$), making use of $f(H_m)$:

$$f(A)\, y \approx V_m\, f(H_m)\, V_m^{\mathrm{T}} y = V_m\, f(H_m)\, \beta\, e_1, \qquad \beta = \|y\|_2$$

∎

10.2 The GMRES (Generalized Minimal Residual) method

GMRES is the most popular Krylov subspace method applicable to any invertible matrix A, based on the Petrov-Galerkin orthogonality (minimal residual) requirement (10.3a), $\mathcal{L}_m = A\mathcal{K}_m$. For given m, GMRES computes an orthogonal basis $\{v_1, \ldots, v_m\}$ of $\mathcal{K}_m = \mathcal{K}_m(A, r_0)$ and solves a linear system of dimension m equivalent to (10.5a) in an efficient way.

The first step of the GMRES algorithm is to generate the set of basis vectors $v_1 \ldots, v_m$ by means of the Arnoldi procedure, in the formulation based on Modified Gram-Schmidt, see Sec. 9.2 (Alg. 9.2).

[52] See (9.30) for the CG case.

Exercise 10.2 Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Assume $r_0 \neq 0$ and let $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ be defined by the Arnoldi algorithm,

$$\bar{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2m} \\ & h_{32} & h_{33} & \dots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix} = \begin{pmatrix} (Av_1,v_1) & (Av_2,v_1) & (Av_3,v_1) & \dots & (Av_m,v_1) \\ (Av_1,v_2) & (Av_2,v_2) & (Av_3,v_2) & \dots & (Av_m,v_2) \\ & (Av_2,v_3) & (Av_3,v_3) & \dots & (Av_m,v_3) \\ & & \ddots & \ddots & \vdots \\ & & & (Av_{m-1},v_m) & (Av_m,v_m) \\ & & & & (Av_m,v_{m+1}) \end{pmatrix}$$

Consider the subdigonal entries of $\bar{H}_m$, and assume $h_{j+1,j} \neq 0$ for $j = 1 \dots m-1$. Show:

a) $\mathcal{K}_m = \operatorname{span}\{v_1, \dots, v_m\}$ and $\dim \mathcal{K}_m = m$.

b) $\bar{H}_m$ has full column rank: $\operatorname{rank} \bar{H}_m = m$ (cf., e.g., (9.7)).

c) If $h_{m+1,m} = 0$, then $A^m r_0 \in \mathcal{K}_m$ (!). Conclude that $\mathcal{K}_m = \mathcal{K}_{m+1} = \dots = \mathcal{K}_n$. ∎

Exercise 10.2 shows: As long as Alg. 9.2 does not break down, i.e., if $h_{j+1,j} \neq 0$ for $j = 1 \dots m-1$, the Arnoldi vectors $\{v_1, \dots, v_m\}$ form an orthogonal basis of $\mathcal{K}_m$ (this fact was already established in Lemma 9.1 above). Note that it is essential for $V_m \in \mathbb{R}^{n\times m}$ to have full rank in order for the expression $(W_m^{\mathrm{T}} A V_m)^{-1}$ in (10.5b) to be meaningful. The case of a breakdown, i.e., $h_{m+1,m} = 0$ with $h_{j+1,j} \neq 0$ for $j = 1 \dots m-1$, is called a *lucky breakdown*[53] because, as we will see below, in this case we already hit the exact solution x_*. (This is a consequence of Exercise 10.1.)

Let us assume the no-breakdown condition

$$h_{j+1,j} \neq 0 \quad \text{for } j = 1 \dots m-1 \tag{10.6}$$

With

$$\beta = \|r_0\|_2$$

we have $\beta v_1 = r_0$, and

$$\beta V_{m+1} e_1 = \beta v_1 = r_0 \tag{10.7}$$

with $e_1 = (1, 0, 0, \dots, 0)^{\mathrm{T}} \in \mathbb{R}^{m+1}$.

The optimality criterion (10.3b) for GMRES is the key to compute x_m in the form $x_m = x_0 + V_m u_m$. Using (10.7) and the identity $AV_m = V_{m+1}\bar{H}_m$ (see (9.7)) we can write the residual of a vector $x = x_0 + V_m u_m$ in the form

$$b - Ax = r_0 - AV_m u_m = \beta v_1 - V_{m+1}\bar{H}_m u_m = V_{m+1}(\beta e_1 - \bar{H}_m u_m)$$

Moreover, since the columns of V_{m+1} are orthonormal, we have $\|b - Ax\|_2 = \|V_{m+1}(\beta e_1 - \bar{H}_m u_m)\|_2 = \|\beta e_1 - \bar{H}_m u_m\|_2$. Thus, the optimality criterion (10.3b) is equivalent to a least square problem for u_m: Find $u_m \in \mathbb{R}^m$ for which the minimum

$$\min_{x \in x_0 + \mathcal{K}_m} \|b - Ax\|_2 = \min_{u_m \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m u_m\|_2 \tag{10.8}$$

is obtained. Since assumption (10.6) guarantees that $\bar{H}_m$ has full rank, problem (10.8) can be uniquely solved for u_m. One way to realize this is to set up and solve the normal equations

$$\bar{H}_m^{\mathrm{T}} \bar{H}_m u_m = \bar{H}_m^{\mathrm{T}} \beta e_1 \tag{10.9}$$

[53] GMRES has no 'serious breakdown' scenario; however, a 'computational breakdown' may occur if the matrices to be stored in memory become too large. See Remark 10.2 below.

for the minimization problem (10.8), which is equivalent to (10.5a). For this system, for example, Cholesky decomposition of $\bar{H}_m^{\mathrm{T}} \bar{H}_m$ could be employed (this would lead to a cost $\mathcal{O}(m^3)$). However, in view of the fact that $\bar{H}_m$ has upper Hessenberg form, it is customary and more efficient to employ a QR-decomposition of $\bar{H}_m$ instead, as discussed in the sequel.

Exercise 10.3 Brief review on the solution of linear least squares problems (cf. e.g. [2]):

Let $k \leq n$ and $B \in \mathbb{R}^{n \times k}$ have full rank, and let $b \in \mathbb{R}^n$. Show:

a) The minimization problem (least-squares problem)

$$\text{Find } u_{opt} \in \mathbb{R}^k \text{ such that } \|b - B\,u_{opt}\|_2 = \min_{u \in \mathbb{R}^k} \|b - B\,u\|_2 \tag{10.10}$$

has a unique solution u_{opt}, given by the solution of the normal equations

$$B^{\mathrm{T}} B\,u = B^{\mathrm{T}} b$$

b) Let B be decomposed in the form $B = Q\bar{R}$, where $Q \in \mathbb{R}^{n \times n}$ is orthogonal and $\bar{R} \in \mathbb{R}^{n \times k}$ has (generalized) upper triangular form,

$$\bar{R} = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

with $R \in \mathbb{R}^{k \times k}$ is upper triangular.

(i) Show that the assumption that B has full (column) rank implies that R is invertible.

(ii) Set $\tilde{b} = Q^{\mathrm{T}} b \in \mathbb{R}^n$ and decompose it as $\tilde{b} = (\tilde{b}_1^{\mathrm{T}}, \tilde{b}_2^{\mathrm{T}})^{\mathrm{T}}$, where $\tilde{b}_1 \in \mathbb{R}^k$ and $\tilde{b}_2 \in \mathbb{R}^{n-k}$.
Show: The solution $u_{opt} \in \mathbb{R}^k$ of the minimization problem (10.10) is given by

$$u_{opt} = R^{-1} \tilde{b}_1$$

(iii) Show that the minimizer u_{opt} satisfies $\|b - B\,u_{opt}\|_2 = \|\tilde{b}_2\|_2$. Thus, the minimal residual norm can be computed independent of u_{opt}.

(iv) Show that, actually, a *reduced QR-decomposition* is sufficient, i.e., only the first k columns of Q are required. These columns form an orthonormal basis of the column space of A. ∎

The least squares problem (10.8) has the standard form considered in Exercise 10.3, i.e., u_m is the solution of the minimization problem

$$\textit{Find } u_m \in \mathbb{R}^m \textit{ such that } \|\beta\,e_1 - \bar{H}_m\,u_m\|_2 \textit{ is minimal.} \tag{10.11}$$

The QR-decomposition of $\bar{H}_m$ is easy to realize because the upper Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1) \times m}$ is already 'close to upper tridiagonal': Only the m non-zero elements of $\bar{H}_m$ below the diagonal need to annihilated. This is typically accomplished using Givens rotations (cf. e.g. [2]).

This QR-decomposition gives $\bar{H}_m = Q_{m+1} \bar{R}_m$, where $Q_{m+1} \in \mathbb{R}^{(m+1) \times (m+1)}$ is orthogonal and $\bar{R}_m \in \mathbb{R}^{(m+1) \times m}$ has the form

$$\bar{R}_m = \left(\begin{array}{c} R_m \\ \hline 0 \end{array} \right)$$

with $R_m \in \mathbb{R}^{m \times m}$ upper triangular. As discussed in Exercise 10.3, R_m is invertible. Hence, the approximate solution $u_m \in \mathbb{R}^m$ can be obtained via back substitution from

$$R_m\,u = \tilde{g}$$

where $\tilde{g}$ consists of the first m components of $\beta\,Q_{m+1}^{\mathrm{T}} e_1 = (\tilde{g}^{\mathrm{T}}, g')^{\mathrm{T}}$ with $\tilde{g} \in \mathbb{R}^m$ and $g' \in \mathbb{R}$. The pseudo-code for this basic form of the *GMRES algorithm* is given as Alg. 10.1. It consists of the Arnoldi iteration up to dimension m, followed by the solution of the resulting least squares problem (10.11).

Algorithm 10.1 GMRES (basic form)

1: Compute $r_0 = b - A x_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
2: Allocate the $(m+1)\times m$-matrix $\bar{H}_m$ and initialize elements h_{ij} to zero
3: **for** $j = 1, 2, \ldots, m$ **do**
4: Compute $w_j = A v_j$
5: **for** $i = 1, \ldots, j$ **do**
6: $h_{ij} = (w_j, v_i)$
7: $w_j = w_j - h_{ij} v_i$
8: **end for**
9: $h_{j+1,j} = \|w_j\|_2$.
10: **if** $h_{j+1,j} = 0$ set $m = j$ and **goto** 12 % lucky breakdown
11: $v_{j+1} = w_j / h_{j+1,j}$
12: **end for**
13: Compute u_m as the minimizer of $\|\beta e_1 - \bar{H}_m u_m\|_2$ and set $x_m = x_0 + V_m u_m$

Exercise 10.4 Use the fact that computing the QR-decomposition of the upper Hessenberg matrix $\bar{H}_m$ requires $\mathcal{O}(m^2)$ flops. Assume that the matrix $A \in \mathbb{R}^{n\times n}$ is sparse and that the cost of a matrix-vector multiplication $x \mapsto Ax$ is $\mathcal{O}(n)$. Show that the cost of GMRES is $\mathcal{O}(m^2 n)$ flops. What is its memory requirement? ∎

Remark 10.2 Some comments on Alg. 10.1:

- Our derivation of Alg. 10.1 assumed the no-breakdown condition (10.6). GMRES terminates upon finding the first $j = m$ with $h_{m+1,m} = 0$, i.e., $h_{j+1,j} \neq 0$ for $j = 1 \ldots m-1$ and $h_{m+1,m} = 0$. This situation is called a *lucky breakdown,* since from Exercise 10.1 we know that then $\mathcal{K}_m = \mathcal{K}_n$; furthermore, from Exercise 10.1 we get $x_m = x_*$, i.e., the GMRES happens to hit exactly the desired solution x_*. Concerning the computation of x_m, we note that Exercise 10.2 shows $\dim \mathcal{K}_m = m$ and that $\bar{H}_m$ has full rank, such that the computation of x_m in line 12 of Alg. 10.1 is possible.

- In practice, the GMRES algorithm is implemented in a different way: The parameter m is not fixed a priori. Instead, a maximum number m_{max} is given, typically dictated by computational resources, in particular, the memory capacity available.

- The vectors $v_1, v_2, \ldots$ are computed successively together with the matrices $\bar{H}_m$. Note that, for computing the optimal u_m and extracting the approximate solution $x_m = x_0 + V_m u_m$ at the end, the full matrix $V_m \in \mathbb{R}^{n\times m}$ – which becomes large for increasing m– has to be kept in memory![54] Once the vectors $v_1, \ldots, v_{m-1}$ and the matrix $\bar{H}_{m-1}$ have already been computed, it is sufficient to compute the next Arnoldi vector v_m, and the matrix $\bar{H}_m$ is obtained from $\bar{H}_{m-1}$ by adding one column and the entry $h_{m+1,m}$. Also, it is possible to update the required QR-decompositions in an efficient way from step to step.

- An appropriate termination condition (typically, the size of the residual $\|r_m\|_2 = \|b - A x_m\|_2$) is employed to stop the iteration. Note that the algorithm does not automatically provide x_m explicitly once only u_m has been computed. However, the residual norm

$$\|r_m\|_2 = \|b - A x_m\|_2 = \|\beta e_1 - \bar{H}_m u_m\|_2$$

  is easily evaluated on the basis of Exercise 10.3; see also [19].

[54] For CG and D-Lanczos the situation is different because the v_i can be reconstructed from a three-term recurrence.

- If the maximum number of iterations has been reached without triggering the termination condition, then a *restart* is usually performed, i.e., GMRES is started afresh with the last approximation $x_{m_{max}}$ as the initial guess. This is called *restarted GMRES*(m_{max}). After a restart, the old data are 'forgotten', and storage requirements for the v_j etc. begin to accumulate from scratch.
- A more detailed description of the various practical implementation issues can be found in [19]. ∎

Remark 10.3 *Faute de mieux*, the residual $\|b - Ax_m\|_2$ is typically used as a stopping criterion. It should be noted that for matrices A with large $\kappa_2(A)$, the error may be large in spite of the residual being small:

$$\frac{\|x_m - x_*\|_2}{\|x_*\|_2} \leq \kappa_2(A) \frac{\|b - Ax_m\|_2}{\|b\|_2}$$

In this sense, only the backward error will be small under such a termination condition based on the norm of the residual. ∎

Remark 10.4 The Krylov method analogous to GMRES for the special case of a symmetric, possibly indefinite matrix A is called the *Minimal Residual Method* (MINRES). Similarly like the D-Lanczos method (which is of FOM type), MINRES involves tridiagonal matrices $H_m = T_m$, which leads to compact update formulas similar to D-Lanczos.

The denotation GMRES is due to historical reasons, because the special symmetric version called MINRES was introduced earlier in the literature. ∎

10.3 GMRES in MATLAB: The function `gmres`

Example 10.1 MATLAB has a robust version of restarted GMRES that can be used for experimentation. Applying this version of GMRES to the SPD matrix $A \in \mathbb{R}^{1806\times 1806}$ `bcsstk14.mtx` from the `MatrixMarket` collection, with exact solution $x = (1, 1, \cdots, 1)^{\mathrm{T}}$ results in the convergence history plotted in Fig. 10.2. We note that the residual decays as the number if iterations increases. When the number of iterations reaches the problem size, the exact solution should be found. As in this example, this does not happen in practice due to round-off problems, but the residual is quite small.

In computational practice, GMRES is usually employed in connection with a suitable *preconditioner*. From this we expect a significant improvement of the convergence behavior. Preconditioning is discussed in Sec. 12. ∎

The following is a printed copy of the help page for the GMRES method implemented in MATLAB. As for CG, alternatively to the matrix A, a function `AFUN` is sufficient which realizes the operation Ax.

The function `gmres` also supports a restarting strategy and preconditioning in several variants, but the concrete preconditioner has to be provided by the user. As for A, the preconditioner may be specified in form of a matrix or a function handle.

```
GMRES   Generalized Minimum Residual Method.

    X = GMRES(A,B) attempts to solve the system of linear equations A*X = B
    for X.  The N-by-N coefficient matrix A must be square and the right
    hand side column vector B must have length N. This uses the unrestarted
    method with MIN(N,10) total iterations.

    X = GMRES(AFUN,B) accepts a function handle AFUN instead of the matrix
```

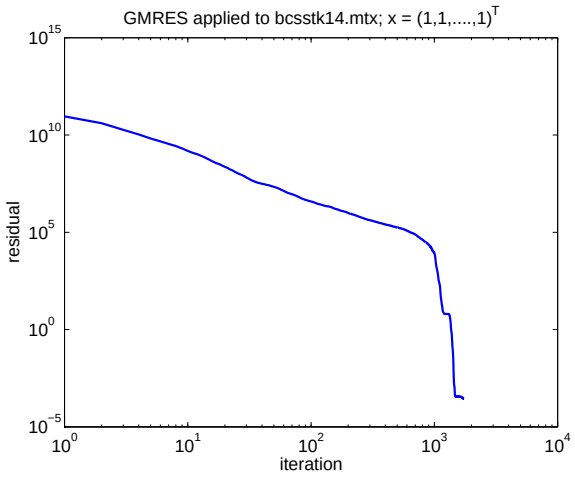


Figure 10.2: Convergence history of GMRES (A is SPD).

```
A. AFUN(X) accepts a vector input X and returns the matrix-vector
product A*X. In all of the following syntaxes, you can replace A by
AFUN.

X = GMRES(A,B,RESTART) restarts the method every RESTART iterations.
If RESTART is N or [] then GMRES uses the unrestarted method as above.

X = GMRES(A,B,RESTART,TOL) specifies the tolerance of the method.  If
TOL is [] then GMRES uses the default, 1e-6.

X = GMRES(A,B,RESTART,TOL,MAXIT) specifies the maximum number of outer
iterations. Note: the total number of iterations is RESTART*MAXIT. If
MAXIT is [] then GMRES uses the default, MIN(N/RESTART,10). If RESTART
is N or [] then the total number of iterations is MAXIT.

X = GMRES(A,B,RESTART,TOL,MAXIT,M) and
X = GMRES(A,B,RESTART,TOL,MAXIT,M1,M2) use preconditioner M or M=M1*M2
and effectively solve the system inv(M)*A*X = inv(M)*B for X. If M is
[] then a preconditioner is not applied.  M may be a function handle
returning M\X.

X = GMRES(A,B,RESTART,TOL,MAXIT,M1,M2,X0) specifies the first initial
guess. If X0 is [] then GMRES uses the default, an all zero vector.

[X,FLAG] = GMRES(A,B,...) also returns a convergence FLAG:
 0 GMRES converged to the desired tolerance TOL within MAXIT iterations.
 1 GMRES iterated MAXIT times but did not converge.
 2 preconditioner M was ill-conditioned.
 3 GMRES stagnated (two consecutive iterates were the same).

[X,FLAG,RELRES] = GMRES(A,B,...) also returns the relative residual
NORM(B-A*X)/NORM(B). If FLAG is 0, then RELRES <= TOL. Note with
preconditioners M1,M2, the residual is NORM(M2\(M1\(B-A*X))).

[X,FLAG,RELRES,ITER] = GMRES(A,B,...) also returns both the outer and
inner iteration numbers at which X was computed: 0 <= ITER(1) <= MAXIT
and 0 <= ITER(2) <= RESTART.
```

```
[X,FLAG,RELRES,ITER,RESVEC] = GMRES(A,B,...) also returns a vector of
the residual norms at each inner iteration, including NORM(B-A*X0).
Note with preconditioners M1,M2, the residual is NORM(M2\(M1\(B-A*X))).

Example:
   n = 21; A = gallery('wilk',n);  b = sum(A,2);
   tol = 1e-12;  maxit = 15; M = diag([10:-1:1 1 1:10]);
   x = gmres(A,b,10,tol,maxit,M);
Or, use this matrix-vector product function
   %-----------------------------------------------------------------%
   function y = afun(x,n)
   y = [0; x(1:n-1)] + [((n-1)/2:-1:0)'; (1:(n-1)/2)'].*x+[x(2:n); 0];
   %-----------------------------------------------------------------%
and this preconditioner backsolve function
   %----------------------------------------%
   function y = mfun(r,n)
   y = r ./ [((n-1)/2:-1:1)'; 1; (1:(n-1)/2)'];
   %----------------------------------------%
as inputs to GMRES:
   x1 = gmres(@(x)afun(x,n),b,10,tol,maxit,@(x)mfun(x,n));

See also bicg, bicgstab, bicgstabl, cgs, lsqr, minres, pcg, qmr, symmlq,
tfqmr, ilu, function_handle.
```

10.4 Convergence properties of the GMRES method

A convergence analysis for GMRES can be done along similar lines as for the CG method in Sec. 8.5.

Lemma 10.2 *Let x_m be the approximate solution obtained in the m-th step of the GMRES algorithm, with residual $r_m = b - Ax_m$. Then, r_m is of the form*

$$r_m = r_0 - p_{m-1}(A)Ar_0 = q_m(A)\,r_0, \quad q_m(\lambda) = 1 - p_{m-1}(\lambda)\,\lambda$$

with the 'GMRES polynomial' $p_{m-1} \in \mathcal{P}_{m-1}$, *and*

$$\|r_m\|_2 = \|\big(I - p_{m-1}(A)A\big)\,r_0\|_2 = \min_{p\in\mathcal{P}_{m-1}} \|\big(I - p(A)A\big)\,r_0\|_2$$

or equivalently, $q_m \in \mathcal{P}_m$ with $q(0) = 1$ and

$$\|r_m\|_2 = \|q_m(A)\,r_0\|_2 = \min_{\substack{q\in\mathcal{P}_m \\ q(0)=1}} \|q(A)\,r_0\|_2$$

Proof: By construction, x_m minimizes the 2-norm of the residual in the affine space $x_0 + \mathcal{K}_m$. Thus, the assertion follows from the fact that

$$x_0 + \mathcal{K}_m = \{x_0 + p(A)\,r_0 : p \in \mathcal{P}_{m-1}\} \quad \Rightarrow \quad r_m = b - Ax_m \in \{r_0 - p(A)Ar_0 : p \in \mathcal{P}_{m-1}\} \qquad \square$$

Lemma 10.2 shows that GMRES iteration can be identified with a polynomial approximation problem reflecting the minimal residual property, similar to the 'Arnoldi/Lanczos approximation problem' (9.15):

GMRES approximation problem:

Find a polynomial $q_m \in \mathcal{P}_m$ ***with*** $q_m(0) = 1$ ***such that*** $\|q_m(A)\,r_0\|_2$ ***becomes minimal.*** (10.12)

Theorem 10.1 *Assume that A is diagonalizable, $A = X\Lambda X^{-1}$, where $\Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. Let*

$$\epsilon_m = \min_{\substack{q\in\mathcal{P}_m \\ q(0)=1}} \max_{i=1\ldots n} |q(\lambda_i)| \tag{10.13a}$$

Then the norm of the m-th residual is bounded by

$$\|r_m\|_2 \leq \kappa_2(X)\,\epsilon_m\,\|r_0\|_2 \tag{10.13b}$$

with $\kappa_2(X) = \|X\|_2\,\|X^{-1}\|_2$.

Proof: Consider an arbitrary polynomial $q \in \mathcal{P}_m$ with $q(0) = 1$, and $x \in x_0 + \mathcal{K}_m$ such that $b - Ax = q(A)\,r_0$. Then,

$$\|r\|_2 = \|b - Ax\|_2 = \|q(A)\,r_0\|_2 = \|X\,q(\Lambda)\,X^{-1}\,r_0\|_2 \leq \kappa_2(X)\,\|q(\Lambda)\|_2\,\|r_0\|_2$$

with

$$\|q(\Lambda)\|_2 = \max_{i=1\ldots n} |q(\lambda_i)|$$

Since x_m minimizes the residual norm over $x_0 + \mathcal{K}_m$, then for any such polynomial q we have

$$\|r_m\|_2 = \min_{x\in x_0+\mathcal{K}_m} \|b - Ax_m\|_2 \leq \kappa_2(X)\,\epsilon_m\,\|r_0\|_2$$

with ϵ_m from (10.13a), as asserted. □

The quantity ϵ_m is related to the question whether such a polynomial q_m exists which is 'small on the spectrum' of A, in the sense of (10.13a). If appropriate a priori information about the spectrum is available, $\|\varepsilon_m\|_2$ can be estimated in a similar way as in the convergence proof for the CG method (see Sec. 8.5).

Example 10.2 Consider an invertible matrix A with real spectrum contained in an interval $[\alpha, \beta] \subseteq \mathbb{R}^+$. From Corollary 6.1 (which is based on the Chebyshev min-max Theorem 6.1), with $\gamma = 0$, we conclude

$$\begin{aligned}\epsilon_m &\leq \min_{\substack{q\in\mathcal{P}_m \\ q(0)=1}} \max_{i=1\ldots n} |q(\lambda_i)| \leq \min_{\substack{q\in\mathcal{P}_m \\ q(0)=1}} \max_{\lambda\in[\alpha,\beta]} |q(\lambda)| \\ &\leq 2\,\frac{c^m}{1+c^{2m}}, \qquad c = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \quad \kappa = \frac{\beta}{\alpha}\end{aligned}$$

However, if A far from normal, the quantity $\kappa = \beta/\alpha$, which is related to the spectral condition number $\kappa_\sigma(A)$, is not related to the quantity $\kappa_2(A)$. ∎

Moreover, the condition number $\kappa_2(X)$ of the matrix of eigenvectors appears in the estimate for $\|r_m\|_2$. For normal A we have $\kappa_2(X) = 1$. In general, however, $\kappa_2(X)$ is typically not known and can be very large, even if the condition number $\kappa_2(A)$ is of moderate size, e.g. if A is close to a non-diagonalizable matrix. The simple convergence result from Theorem 10.1 is therefore of limited practical value.

Remark 10.5 For GMRES, the required effect of a preconditioner to accelerate convergence is less clear cut than, e.g., for the CG method. In general, 'bunching of eigenvalues' is still a good strategy, as ϵ_m can be bounded by a complex analog of the Chebyshev min-max Theorem; see [19].

Preconditioning is the topic of Sec. 12 ff. ∎

Exercise 10.5 Let $A \in \mathbb{R}^{n\times n}$ be of the bidiagonal form

$$A = \begin{pmatrix} 1 & c_1 & & & & \\ & 1 & c_2 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & c_{n-2} & \\ & & & & 1 & c_{n-1} \\ & & & & & 1 \end{pmatrix}$$

with $c_i \neq 0$. Show: The eigenvalue $\lambda = 1$ has geometric multiplicity 1. What is the corresponding eigenspace? Furthermore, n is the smallest integer m for which $(A - I)^m = 0$, and the characteristic polynomial $\chi(z)$ of A is identical with its minimal polynomial, $\chi(z) = \mu(z) = (z-1)^n$. Try to explain how the GMRES iteration applied to A behaves. (You may also test.)

Hint: The dimension of the eigenspace $\ker(A - I)$ is 1. Let u_1 denote an eigenvector. A first principal vector u_2 is a vector satisfying $Au_2 = u_2 + u_1$, or equivalently, $u_2 \in \ker((A-I)^2)$, and so on. What is the dimension of $\ker((A-I)^2)$? Now, consider higher and higher principal vectors, and the study the behavior of GMRES when applied to $r_0 = u_1$, $r_0 = u_2$, and so on. ∎

A GMRES residual decay estimate in the positive definite case.

Let us consider systems with a (not necessarily symmetric) positive definite matrix A. In particular, we assume

$$(Ax, x) \geq \gamma\,(x, x) \quad \text{for all} \quad x \in \mathbb{R}^n \tag{10.14}$$

i.e., $\mathrm{Re}A \geq \gamma$. (In the context of elliptic PDEs, such a quantitative condition is also called *coercivity*, or *ellipticity*, see (A.2).)

The m-th iterate of the GMRES method minimizes the residual norm $\|r_m\|_2$ over the affine Krylov space $r_0 + A\mathcal{K}_m$, see (10.3b). Since $r_{m-1} \in \mathcal{K}_m$, this implies

$$\begin{aligned} \|r_m\|_2^2 &\leq \min_{\alpha\in\mathbb{R}} \|r_{m-1} - \alpha A r_{m-1}\|_2^2 \\ &= \min_{\alpha\in\mathbb{R}} \left\{ \|r_{m-1}\|_2^2 - 2\alpha\,(Ar_{m-1}, r_{m-1}) + \alpha^2 \|Ar_{m-1}\|^2 \right\} \end{aligned}$$

The minimum on the right hand side is attained at

$$\alpha_{min} = \frac{(Ar_{m-1}, r_{m-1})}{\|Ar_{m-1}\|^2}$$

giving

$$\|r_m\|_2^2 \leq \left(1 - \frac{(Ar_{m-1}, r_{m-1}) \,/\, \|r_{m-1}\|_2^2}{\|Ar_{m-1}\|^2}\right) \|r_{m-1}\|_2^2$$

Together with assumption (10.14) this gives

$$\|r_m\|_2 \leq \sqrt{1 - \frac{\gamma^2}{\|A\|_2^2}}\, \|r_{m-1}\|_2 \tag{10.15}$$

The decay in (10.15) is uniform, i.e., the convergence of the residuals towards zero is geometric.

For this simple estimate we only have compared two successive residuals, but we have not made use of the global minimal residual property. Therefore, a further consequence of estimate (10.15) that for the positive definite case, also the restarted GMRES produces residuals with uniformly decreasing norms.

Exercise 10.6 Reformulate the estimate (10.15) in terms of the spectrum and condition number of A for the case where A is SPD. ∎

11 Methods Based on Biorthogonalization. BiCG

The main disadvantage of GMRES is its high memory requirement. For symmetric matrices A, this can be circumvented by using D-Lanczos, MINRES, or CG. The main reason for these savings in the symmetric case is that – in the notation of Sec. 10 – the matrix $W_m^{\mathrm{T}} A V_m$ becomes tridiagonal. In general, this property is also retained in *biorthogonalization methods* which we now consider.

Remark 11.1 Consider the Lanczos decomposition from Sec. 9.3 for symmetric $A \in \mathbb{R}^{n\times n}$,

$$V_m^{\mathrm{T}} A V_m = T_m \in \mathbb{R}^{m\times m} \quad \text{tridiagonal, symmetric} \tag{11.1a}$$

where the columns $v_i \in \mathbb{R}^n$ of V_m are orthonormal. As discussed before, this can be viewed as 'reduced version' of the full orthonormal triangularization

$$V^{\mathrm{T}} A V = T \in \mathbb{R}^{n\times n} \quad \text{tridiagonal, symmetric} \tag{11.1b}$$

where the v_i form an orthonormal basis of the full space $\mathbb{R}^n$. The decompositions (11.1a) and (11.1b) can be obtained in floating point arithmetic in a finite number of steps, in contrast to the eigendecomposition

$$X^{\mathrm{T}} A X = \Lambda \in \mathbb{R}^{n\times n} \quad \text{diagonal} \tag{11.2}$$

where the columns x_i of X form an orthonormal basis of eigenvectors of A.

The idea of biorthogonality is not so far off. Consider an arbitrary, nonsymmetric diagonalizable matrix $A \in \mathbb{R}^{n\times n}$, with eigendecomposition

$$X^{-1} A X = \Lambda \in \mathbb{R}^{n\times n} \quad \text{diagonal} \tag{11.3a}$$

Here, the columns x_j of X form a (generally non-orthonormal) basis of eigenvectors of A; they are also called *right eigenvectors* of A. Transposing (11.3a) gives

$$X^{\mathrm{T}} A^{\mathrm{T}} X^{-\mathrm{T}} = \Lambda \tag{11.3b}$$

This shows that the columns y_i of $X^{-\mathrm{T}}$, i.e., the rows of X^{-1} are the (right) eigenvectors of A^{T}. They are also called *left eigenvectors* of A. Now, identity $I = X^{-1}X$ shows

$$(y_i, x_j) = \delta_{i,j}$$

i.e., the pair of eigenbases $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ is *biorthogonal*. With $Y = X^{-\mathrm{T}}$ we can write (11.3a) in the form

$$Y^{\mathrm{T}} A X = \Lambda \in \mathbb{R}^{n\times n} \quad \text{diagonal} \tag{11.4}$$

In view of (11.1b), we may again ask for a directly computable modification of such a decomposition in the form

$$W^{\mathrm{T}} A V = T \in \mathbb{R}^{n\times n} \quad \text{(unsymmetric) tridiagonal} \tag{11.5a}$$

with a biorthogonal matrix pair (V, W), i.e., $W^{\mathrm{T}} V = I$, or, more generally, a reduced version analogous to (11.1a),

$$W_m^{\mathrm{T}} A V_m = T_m \in \mathbb{R}^{m\times m} \quad \text{(unsymmetric) tridiagonal} \tag{11.5b}$$

with $V_m, W_m \in \mathbb{R}^{n\times m}$ satisfying $W_m^{\mathrm{T}} V_m = I_{m\times m}$. This is exactly what the *Lanczos biorthogonalization procedure* aims to realize. ∎

11.1 Lanczos biorthogonalization

Let $v_1, w_1 \in \mathbb{R}^n$ be a pair of vectors, and consider the Krylov spaces $\mathcal{K}_m(A, v_1)$ and $\mathcal{K}_m(A^{\mathrm{T}}, w_1)$. The classical algorithm to compute two sets of vectors $\{v_1, \ldots, v_m\}$ and $\{w_1, \ldots, w_m\}$ which span these two spaces and which are biorthonormal, i.e., $(v_i, w_j) = \delta_{i,j}$, is Alg. 11.1 specified below.

This algorithm, the *Lanczos biorthogonalization procedure,* is a generalization of the Lanczos orthonormalization procedure discussed in Sec. 9. We wish to find a pair of biorthogonal matrices $V_m, W_m \in \mathbb{R}^{n\times m}$,

$$V_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ v_1 & v_2 & \vdots & v_m \\ | & | & & | \\ | & | & & | \end{array}\right), \quad W_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ w_1 & w_2 & \vdots & w_m \\ | & | & & | \\ | & | & & | \end{array}\right), \qquad W_m^{\mathrm{T}} V_m = I_{m\times m} \tag{11.6}$$

The idea due to Lanczos is to use two simultaneous Lanczos processes for A and A^{T}. We make the following ansatz motivated by the symmetric Lanczos procedure:

$$\begin{aligned} A V_m &= V_{m+1} \bar{T}_m, \\ A^{\mathrm{T}} W_m &= W_{m+1} \bar{S}_m \end{aligned} \tag{11.7}$$

with a pair of $(m+1)\times m$ tridiagonal matrices

$$\bar{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \\ & & & & \delta_{m+1} \end{pmatrix}, \qquad \bar{S}_m = \begin{pmatrix} \alpha_1 & \delta_2 & & & \\ \beta_2 & \alpha_2 & \delta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \delta_m \\ & & & \beta_m & \alpha_m \\ & & & & \beta_{m+1} \end{pmatrix}, \tag{11.8}$$

The desired relations (11.7) are equivalent to a pair of three-term recurrences (cf. (9.14) for the symmetric case),

$$\begin{aligned} A v_j &= \beta_j v_{j-1} + \alpha_j v_j + \delta_{j+1} v_{j+1} \\ A^{\mathrm{T}} w_j &= \delta_j w_{j-1} + \alpha_j w_j + \beta_{j+1} w_{j+1} \end{aligned} \tag{11.9}$$

for $j = 1 \ldots m$, with $\beta_1 = \delta_1 = 0$, from which it will be possible to compute $v_2, v_3, \ldots$ and $w_2, w_3, \ldots$ as long as the β_j and δ_j do not vanish.

So far we have not used the biorthogonality requirement for the vectors v_i and w_j. If this is assured, together with (11.7) it will lead us to a decomposition of the desired form (11.5b),

$$W_m^{\mathrm{T}} A V_m = W_m^{\mathrm{T}} V_{m+1} \bar{T}_m = \left(I_{m\times m} \,\middle|\, 0 \right) \bar{T}_m = T_m$$

where the square tridiagonal matrix $T_m \in \mathbb{R}^{m\times m}$ is defined from $\bar{T}_m$ (or, equivalently, from $\bar{S}_m^{\mathrm{T}}$) by removing its last row,

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \in \mathbb{R}^{m\times m} \tag{11.10}$$

Algorithm 11.1 Lanczos biorthogonalization

1: Choose two vectors v_1, w_1 with $(v_1, w_1) = 1$
2: Set $\beta_1 = \delta_1 = 0$, $w_0 = v_0 = 0$
3: **for** $j = 1, 2, \ldots,$ **do**
4: $\quad \alpha_j = (A v_j, w_j)$
5: $\quad \hat{v}_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$
6: $\quad \hat{w}_{j+1} = A^{\mathrm{T}} w_j - \alpha_j w_j - \delta_j w_{j-1}$
7: $\quad \delta_{j+1} = \sqrt{|(\hat{v}_{j+1}, \hat{w}_{j+1})|}$. If $\delta_{j+1} = 0$ **stop**
8: $\quad \beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$
9: $\quad w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
10: $\quad v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
11: **end for**

With this notation, (11.7) can be written as (cf. (9.7))

$$\begin{aligned} A V_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^{\mathrm{T}} \\ A^{\mathrm{T}} W_m &= W_m T_m^{\mathrm{T}} + \beta_{m+1} w_{m+1} e_m^{\mathrm{T}} \end{aligned} \tag{11.11}$$

with $e_m = (0, 0, \ldots, 1)^{\mathrm{T}}$.

The algorithmic realization consists in implementing the pair of three-term recurrences (11.7) and enforcing biorthogonality. This fixes the coefficients α_j, β_j and δ_j in the following, recursive way:

Assume that, for $j \geq 1$, $v_1, \ldots, v_j$ and $w_1, \ldots, w_j$ already satisfy biorthogonality. We wish to construct v_{j+1} and w_{j+1} according to (11.7) in a way such that the extended sequences $v_1, \ldots, v_{j+1}$ and $w_1, \ldots, w_{j+1}$ are also biorthogonal. Let

$$\begin{aligned} \hat{v}_{j+1} &= A v_j - \alpha_j v_j - \beta_j v_{j-1}, \\ \hat{w}_{j+1} &= A^{\mathrm{T}} w_j - \alpha_j w_j - \delta_j w_{j-1} \end{aligned}$$

Consider the inner product $(\hat{v}_{j+1}, w_j)$ and enforce biorthogonality:

$$0 \stackrel{!}{=} (\hat{v}_{j+1}, w_j) = (A v_j, w_j) - \alpha_j \underbrace{(v_j, w_j)}_{=1} - \beta_j \underbrace{(v_{j-1}, w_j)}_{=0}$$

This shows that α_j has to be chosen as

$$\alpha_j = (A v_j, w_j)$$

Exactly the same choice for α_j also leads to $(\hat{w}_{j+1}, v_j) = 0$, as is easy to check. Furthermore, we rescale $\hat{v}_{j+1}$ and $\hat{w}_{j+1}$ such that, with the normalization coefficients β_{j+1} and δ_{j+1}, the resulting normalized vectors $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$ and $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$ satisfy $(v_{j+1}, w_{j+1}) = 1$. This is the standard choice, see Alg. 11.1. This choice is not unique – the only requirement is $\delta_{j+1}\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})$. However, for other choices the algorithm has to be appropriately modified, see [19].

The final outcome is Alg. 11.1, resulting in a pair of *full biorthonormal bases* $\{v_1 \ldots v_m\}$ and $\{w_1 \ldots w_m\}$ of $\mathcal{K}_m(A, v_1)$ and $\mathcal{K}_m(A^{\mathrm{T}}, w_1)$, respectively. The proof of full biorthonormality is the topic of the following exercise.

Exercise 11.1 Show that the above construction automatically leads to a full biorthonormal sequence. In particular, we have

$$0 = (v_{j+1}, w_1) = \ldots = (v_{j+1}, w_{j-1}), \qquad 0 = (w_{j+1}, v_1) = \ldots = (w_{j+1}, v_{j-1})$$

Hint: Proof by induction, exploiting the structure of the three-term recurrences for the v_j and w_j. See Sec. 9.3 for the analogous argument in the symmetric case. ∎

The main advantage over the Arnoldi procedure consists in the fact that only short recurrences are involved, which results in significant savings of computing costs and storage. On the other hand, the fact that the transpose A^{T} or a corresponding evaluation procedure is required may be a significant disadvantage in applications where this is not readily available or expensive to evaluate.

Note that a *serious breakdown* occurs if δ_{j+1} as defined in line 7 of Alg. 11.1 vanishes, in particular if $\hat{v}_{j+1} \neq 0$ and $\hat{w}_{j+1} \neq 0$ but $(\hat{v}_{j+1}, \hat{w}_{j+1})$ vanishes or becomes very small. Such a breakdown is hardly predictable; in the literature, strategies have been developed where the freedom in the choice of the β_j and δ_j is exploited to avoid breakdowns, e.g., by *Look-Ahead*-strategies. Still, this is a difficult topic; see [19] for some details.

Summary of the properties of Lanczos biorthogonalization:

Theorem 11.1 *Assume that Alg. 11.1 does not break down before step m. Then the vectors v_j and w_j, $j = 1 \ldots m$, are biorthogonal. Moreover, $\{v_1, \ldots, v_m\}$ is a basis of $\mathcal{K}_m(A, v_1)$ and $\{w_1, \ldots, w_m\}$ is a basis of $\mathcal{K}_m(A^T, w_1)$. The following identities hold true:*

$$\begin{aligned} AV_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^T \\ A^T W_m &= W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T \\ W_m^T A V_m &= T_m \end{aligned} \tag{11.12}$$

with T_m from (11.10).

In general, the bases $\{v_1, \ldots, v_m\}$ and $\{w_1, \ldots, w_m\}$ are not orthogonal themselves.

Theorem 11.1 can be interpreted as follows. The matrix T_m is a projected version of A corresponding to an *oblique* (non-orthogonal) projection onto $\mathcal{K}_m(A, v_1)$. More precisely: Consider the matrix

$$V_m T_m W_m^{\mathrm{T}} = V_m W_m^{\mathrm{T}} A V_m W_m^{\mathrm{T}} \in \mathbb{R}^{n \times n}$$

By construction, $V_m W_m^{\mathrm{T}}$ is an oblique projector onto $\mathrm{span}\{v_1, \ldots, v_m\} = \mathcal{K}_m(A, v_1)$ because due to biorthogonality, $V_m W_m^{\mathrm{T}} V_m W_m^{\mathrm{T}} = V_m W_m^{\mathrm{T}}$. An analogous interpretation can be given for T_m^{T}, which is a projected version of A^{T} (projection onto $\mathcal{K}_m(A^{\mathrm{T}}, w_1)$).

In the following section we consider a Krylov subspace method, BiCG, which is derived in a similar way from Alg. 11.1 as D-Lanczos and GMRES were derived from the Lanczos and Arnoldi processes.

In Alg. 11.1, both mappings $x \mapsto Ax$ and $x \mapsto A^{\mathrm{T}}x$ are involved, and similar operations are performed with them. This can be pursued further to see that, if two linear systems involving A and A^{T} have to be solved, methods based on Lanczos biorthogonalization appear to be natural. Otherwise it may be favorable to avoid explicit evaluation of A^{T}; we briefly discuss such techniques in Sec. 11.3.

11.2 BiCG

According to p. 84, case (iii), the approximation x_m by a biorthogonal method is defined via the Petrov-Galerkin condition

Find $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ ***such that its residual*** $r_m = b - Ax_m$ ***satisfies*** $r_m \perp \mathcal{L}_m = \mathcal{K}_m(A^{\boldsymbol{T}}, w_1)$. (11.13)

We have:

Lemma 11.1 *Assume that Alg. 11.1 does not break down before step m. Assume that T_m is invertible. Then,* (11.13) *has a unique solution of the form*

$$x_m = x_0 + V_m u_m \tag{11.14a}$$

where u_m is the solution of

$$T_m u_m = W_m^T r_0 = \beta e_1, \quad \beta = \|r_0\|_2 \tag{11.14b}$$

Proof: Exercise – essentially along the same lines as for D-Lanczos (see Sec. 9.5). □

Remark 11.2 The related so-called QMR (*Quasi Minimal Residual*) method is also based on Lanczos biorthogonalization. Here, in contrast to (11.14b), u_m is determined by minimizing $\|\beta e_1 - \bar{T}_m u_m\|_2$ (similarly as for GMRES), with $e_1 = (1, 0, \ldots, 0)^{\mathrm{T}} \in \mathbb{R}^{m+1}$ and $\bar{T}_m \in \mathbb{R}^{(m+1)\times m}$ from (11.8). ∎

Remark 11.3 Consider the 'dual' linear system $A^{\mathrm{T}} x = b'$, where b' is a given vector. If x_0' is an initial guess and $w_1 = r_0' = b' - A^{\mathrm{T}} x_0'$, then (under the assumptions of Lemma 11.1) the solution u_m' of

$$T_m^{\mathrm{T}} u_m' = V_m^{\mathrm{T}} r_0' = \beta' e_1, \quad \beta' = \|r_0'\|_2$$

leads to an approximation $x_m' = x_0' + W_m u_m'$ which solves the Petrov-Galerkin problem (a dual version of (11.13))

$$\textit{Find } x_m \in x_0 + \mathcal{K}_m(A^T, r_0') \textit{ such that } r_m' = b' - A^T x_m' \perp \mathcal{L}_m' = \mathcal{K}_m(A, v_1)\,. \tag{11.15}$$

This follows from the fact that, in the Lanczos biorthogonalization procedure, the matrices A and A^{T} occur in a way *pari passu*. In other words: simply change the role of A and A^{T}. As a consequence, by considering the Krylov spaces $\mathcal{K}_m(A, r_0)$ and $\mathcal{K}_m(A^{\mathrm{T}}, r_0')$, it is possible to solve $Ax = b$ and $A^{\mathrm{T}} x' = b'$ simultaneously. ∎

The procedure towards an algorithm now parallels that of our derivation of the D-Lanczos method (and the CG algorithm); we will be slightly more brief here. In order to simplify the notation, we assume that we wish to solve simultaneously the systems $Ax = b$ and $A^{\mathrm{T}} x' = b'$. We also assume that initial guesses x_0 and x_0' are prescribed. These determine the initial residuals $r_0 = b - Ax_0$, $r_0' = b' - A^{\mathrm{T}} x_0'$ and thus the vectors v_1 and w_1, which in turn determine the Krylov spaces $\mathcal{K}_m(A, v_1)$ and $\mathcal{K}_m(A^{\mathrm{T}}, w_1)$.

We assume that the matrix T_m delivered by Lanczos biorthogonalization has an LU-decomposition $T_m = L_m U_m$. Then the approximate solutions x_m, x_m' are given by

$$x_m = x_0 + D_m z_m, \qquad x_m' = x_0' + D_m' z_m'$$

where $D_m = \left(d_0 \,\middle|\, \ldots \,\middle|\, d_{m-1} \right) = V_m U_m^{-1}$, $D_m' = \left(d_0' \,\middle|\, \ldots \,\middle|\, d_{m-1}' \right) = W_m L_m^{-\mathrm{T}}$, and

$$z_m = L_m^{-1}(\beta e_1), \qquad z_m' = U_m^{-\mathrm{T}}(\beta' e_1)$$

In order to formulate the BiCG algorithm, what has to be done is to combine the Lanczos biorthogonalization process with the LU-decomposition of the tridiagonal matrix T_m into a single iteration:

Proceeding in the same way as in the derivation of the D-Lanczos method, one concludes that the matrices D_m, D_m' are obtained from D_{m-1}, D_{m-1}' by simply appending one column. Also, z_m and z_m' are obtained from z_{m-1}, z_{m-1}' by adding one entry. As for the D-Lanczos method we therefore obtain

$$x_m = x_{m-1} + \zeta_m d_{m-1}, \qquad x_m' = x_{m-1}' + \zeta_m' d_{m-1}'$$

for suitable ζ_m, ζ_m'. Furthermore, from $V_m = D_m U_m$ and $W_m = D_m' L_m^{\mathrm{T}}$ we have

$$d_{m-1} \in \mathrm{span}\{d_{m-2}, v_m\}, \qquad d_{m-1}' \in \mathrm{span}\{d_{m-2}', w_m\}$$

Now, instead of explicitly computing the update formulas for ζ_m, ζ_m' and d_m, d_m', we may exploit the (bi)orthogonality conditions in the same way as in the derivation of the CG algorithm from D-Lanczos:

(i) The search directions d_j, d'_j form a A-biorthogonal system: This follows from

$$\big(D'_m\big)^{\mathrm{T}} A\, D_m = L_m^{-1} W_m^{\mathrm{T}} A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I$$

(ii) The residuals r_m, r'_m are multiplies of v_{m+1} and w_{m+1}, respectively, and are therefore biorthogonal. This follows as in the CG-case: From the definition of x_m and Theorem 11.1 we get

$$\begin{aligned} r_m &= b - A x_m = b - A(x_0 + V_m u_m) = r_0 - A V_m u_m = \beta v_1 - (V_m T_m + \delta_{m+1} v_{m+1} e_m^{\mathrm{T}}) u_m \\ &= V_m(\beta e_1 - T_m u_m) + \delta_{m+1} v_{m+1}(e_m^{\mathrm{T}} u_m) = \delta_{m+1} v_{m+1}(e_m^{\mathrm{T}} u_m) \end{aligned}$$

An analogous reasoning shows that r'_m is a multiple of w_{m+1}.

We collect these findings:

$$\begin{aligned} x_{m+1} &= x_m + \alpha_m d_m, & x'_{m+1} &= x'_m + \alpha'_m d'_m \\ r_{m+1} &= r_m - \alpha_m A d_m, & r'_{m+1} &= r'_m - \alpha'_m A^{\mathrm{T}} d'_m \\ d_{m+1} &= r_{m+1} + \beta_m d_m, & d'_{m+1} &= r'_{m+1} + \beta'_m d'_m \end{aligned} \tag{11.16}$$

Using the above orthogonality conditions, we obtain after some manipulations

$$\alpha_m = \alpha'_m = \frac{(r_m, r'_m)}{(A d_m, d'_m)}, \qquad \beta_m = \beta'_m = \frac{(r_{m+1}, r'_{m+1})}{(r_m, r'_m)} \tag{11.17}$$

This leads to Alg. 11.2. (Here we ignore the update for the x'_m, which can be omitted if we are only interested in solving $Ax = b$.)

Algorithm 11.2 BiCG

Compute $d_0 = r_0 = b - A x_0$
Choose r'_0 such that $(r_0, r'_0) \neq 0$
for $m = 0, 1 \ldots$ **do**
  $\alpha_m = \frac{(r_m, r'_m)}{(A d_m, d'_m)}$
  $x_{m+1} = x_m + \alpha_m d_m$
  $r_{m+1} = r_m - \alpha_m A d_m$
  $r'_{m+1} = r'_m - \alpha_m A^{\mathrm{T}} d'_m$
  $\beta_m = (r_{m+1}, r'_{m+1})/(r_m, r'_m)$
  $d_{m+1} = r_{m+1} + \beta_m d_m$
  $d'_{m+1} = r'_{m+1} + \beta'_m d'_m$
end for

Remark 11.4 If a dual problem $A^{\mathrm{T}} x_* = b'$ is also to be solved, then r'_0 depends on the initial guess x'_0 and the update for the x'_m needs also to be explicitly carried out. ∎

Exercise 11.2 Show that the BiCG method produces residuals r_m, r'_m and search directions d_m, d'_m of the form

$$r_m = \varphi_m(A) r_0, \quad r'_m = \varphi_m(A^{\mathrm{T}}) r'_0; \qquad d_m = \pi_m(A) r_0, \quad d'_m = \pi_m(A^{\mathrm{T}}) r'_0$$

where φ_m and π_m are polynomials of degree m. ∎

11.3 A brief look at CGS and BiCGStab

In many applications the matrix A is not explicitly available, but merely a routine that provides the matrix-vector multiplication $x \mapsto Ax$. In such a situation, the operation $x \mapsto A^{\mathrm{T}}x$ is not necessarily available, and therefore the BiCG-algorithm is not directly applicable. One of the reasons for developing the *Conjugate Gradient Squared* (CGS) and the *BiConjugate Gradient Stabilized* (BiCGStab) methods was to circumvent this problem. Here we illustrate the main ideas of the CGS method, which is a 'transpose-free' formulation of the BiCG method. Working out all the details is a job involving manipulating Krylov recurrences.

Our starting point is the observation from Exercise 11.2. With the corresponding notation, the α_m and β_m from (11.17) satisfy

$$\alpha_m = \frac{(\varphi_m(A)\,r_0, \varphi_m(A^{\mathrm{T}})\,r_0')}{(A\,\pi_m(A)\,r_0, \pi_m(A^{\mathrm{T}})\,r_0')} = \frac{(\varphi_m^2(A)\,r_0, r_0')}{(A\,\pi_m^2(A)\,r_0, r_0')}\,,$$
$$\beta_m = \frac{(\varphi_{m+1}(A)\,r_0, \varphi_{m+1}(A^{\mathrm{T}})\,r_0')}{(\varphi_m(A)\,r_0, \varphi_m(A^{\mathrm{T}})\,r_0')} = \frac{(\varphi_{m+1}^2(A)\,r_0, r_0')}{(\varphi_m^2(A)\,r_0, r_0')}$$

We see: If we can derive a recursion for the vectors $\varphi_m^2(A)\,r_0$ and $\pi_m^2(A)\,r_0$, then we can compute the parameters α_m and β_m in a transpose-free way, i.e., without explicit reference to A^{T}.

The CGS algorithm computes approximations $\hat{x}_m$ – equivalent to the x_m delivered by BiCG – with residuals $r_m = \hat{r}_m$ of the form

$$\hat{r}_m = \varphi_m^2(A)\,r_0 \tag{11.18}$$

We now show that recurrences for the iterates $\hat{x}_m$ and the residuals $\hat{r}_m$ can indeed be realized. This is achieved by some algebraic manipulation of some formulas: From the recurrences (11.16) for the residuals r_m and the search directions d_m in BiCG, we obtain a pair of recursions for the polynomials φ_m and π_m:

$$\varphi_{m+1}(\lambda) = \varphi_m(\lambda) - \alpha_m\,\lambda\,\pi_m(\lambda), \tag{11.19a}$$
$$\pi_{m+1}(\lambda) = \varphi_{m+1}(\lambda) + \beta_m\,\pi_m(\lambda) \tag{11.19b}$$

Squaring yields

$$\varphi_{m+1}^2(\lambda) = \varphi_m^2(\lambda) - 2\,\alpha_m\,\lambda\,\varphi_m(\lambda)\,\pi_m(\lambda) + \alpha_m^2\,\lambda^2\,\pi_m^2(\lambda), \tag{11.20a}$$
$$\pi_{m+1}^2(\lambda) = \varphi_{m+1}^2(\lambda) + 2\,\beta_m\,\varphi_{m+1}(\lambda)\,\pi_m(\lambda) + \beta_{m+1}^2\,\pi_m^2(\lambda) \tag{11.20b}$$

From this we can obtain recurrences for the polynomials φ_m^2 and π_m^2 by introducing the auxiliary polynomial

$$\psi_m(\lambda) = \varphi_{m+1}(\lambda)\,\pi_m(\lambda)$$

since the other cross term $\varphi_m(\lambda)\,\pi_m(\lambda)$ can be computed from the functions $\varphi_m^2(\lambda)$, $\pi_m^2(\lambda)$ and $\psi_m(\lambda)$ as (see (11.20b))

$$\varphi_m(\lambda)\,\pi_m(\lambda) = \varphi_m(\lambda)\big(\varphi_m(\lambda) + \beta_{m-1}\,\pi_{m-1}(\lambda)\big) = \varphi_m^2(\lambda) + \beta_{m-1}\,\varphi_m(\lambda)\,\pi_{m-1}(\lambda) \tag{11.21}$$

Combining (11.19a) with (11.20a) and (11.21) yields a recurrence for the polynomials ψ_m:

$$\begin{aligned}\psi_m(\lambda) = \varphi_{m+1}(\lambda)\,\pi_m(\lambda) &= \big(\varphi_m(\lambda) - \alpha_m\,\lambda\,\pi_m(\lambda)\big)\pi_m(\lambda) = -\alpha_m\,\lambda\,\pi_m^2(\lambda) + \varphi_m(\lambda\,\pi_m(\lambda)\\ &= -\alpha_m\,\lambda\,\pi_m^2(\lambda) + \varphi_m^2(\lambda) + \beta_{m-1}\,\varphi_m(\lambda)\,\pi_{m-1}(\lambda)\\ &= -\alpha_m\,\lambda\,\pi_m^2(\lambda) + \varphi_m^2(\lambda) + \beta_{m-1}\,\psi_{m-1}(\lambda)\end{aligned}$$

This also allows us to obtain a recurrence for the residuals $\hat{r}_m$ from (11.18). In order to motivate the recurrence for the approximations $\hat{x}_m$, we make the ansatz $\hat{x}_{m+1} = \hat{x}_m + \alpha_m\,\hat{d}_m$ for some search direction $\hat{d}_m$. The latter has to satisfy

$$\hat{r}_{m+1} = b - A\,\hat{x}_{m+1} = b - A\,(\hat{x}_m + \alpha_m\,\hat{d}_m) = \hat{r}_m - \alpha_m\,A\,\hat{d}_m$$

Since we require $\hat{r}_m = \varphi_m^2(A)r_0$, we obtain in view of (11.20a)

$$\begin{aligned} -\alpha_m A\hat{d}_m &= \hat{r}_{m+1} - \hat{r}_m = \varphi_{m+1}^2(A)\,r_0 - \varphi_m^2(A)\,r_0 \\ &= A\big(-2\alpha_m \pi_m(A)\,\varphi_m(A) + \alpha_m^2 A\pi_m^2(A)\big)\,r_0 \end{aligned}$$

From this, we readily infer that $\hat{d}_m$ is given by

$$\hat{d}_m = \big(2\pi_m(A)\varphi_m(A) - \alpha_m A\pi_m^2(A)\big)r_0$$

Collecting these findings, for the vectors

$$\hat{r}_m = \varphi_m^2(A)\,r_0, \quad \hat{p}_m = \pi_m^2(A)\,r_0, \quad \hat{q}_m = \varphi_{m+1}(A)\,\pi_m(A)\,r_0$$

we obtain the following recursions (with $\hat{x}_0 = x_0$, $\hat{r}_0 = b - A\hat{x}_0$, $\hat{p}_0 = \hat{r}_0$, $\beta_0 = 0$):

$$\begin{aligned} \alpha_m &= (\hat{r}_m, r_0')/(A\hat{p}_m, r_0'), \\ \hat{d}_m &= 2\hat{r}_m + 2\beta_{m-1}\hat{q}_{m-1} - \alpha_m A\hat{p}_m, \\ \hat{q}_m &= \hat{r}_m + \beta_{m-1}\hat{q}_{m-1} - \alpha_{m-1}A\hat{p}_m, \\ \hat{x}_{m+1} &= \hat{x}_m + \alpha_m \hat{d}_m, \\ \hat{r}_{m+1} &= \hat{r}_m - \alpha_m A\hat{d}_m, \\ \beta_m &= (\hat{r}_{m+1}, r_0')/(\hat{r}_m, r_0'), \\ \hat{p}_{m+1} &= \hat{r}_{m+1} + \beta_m(2\hat{q}_m + \beta_j \hat{p}_m) \end{aligned}$$

Remark 11.5 We refer to [19] for a slightly different formulation of the CGS algorithm. From the above derivation of CGS it is not clear that it is indeed a Krylov subspace method. In the survey article [6] for a more detailed discussion of the fact that the CGS can be viewed as a Krylov subspace method is given. However, here the inner product with respect to which orthogonality is required is problem dependent. ∎

It has been observed in numerical examples that the convergence of CGS is often 'erratic' and the residuals can become very large. In order to remedy this, the BiCGStab variant was introduced, which usually leads to a 'smoother' convergence behavior – this is mainly experimental evidence. We refer to [19] for a more detailed description of the algorithm.

Many other variants have been developed, trying to combine desired virtues as non-erratic convergence behavior, look-ahead strategies to avoid breakdowns, and transpose-free formulation. This is still an active research area; 'the universal method' which solves all problems with best efficiency does not exist. Particular classes of problems are often better understood (the SPD case is a typical example).

This leads us to the next topic, preconditioning. Here one tries to combine a standard technique like CG or GMRES with a preprocessing step, aiming at solving an equivalent but 'less harmful' problem at some extra cost per step. Most successful preconditioning technique are taylored to particular problem structures; finding a good preconditioner is the main job in order to realize an efficient solution algorithm for a particular class of problems.

Example 11.1 In some sense, the matrix A from Exercise 10.5 may be call an *enfant terrible* from the point of view of Krylov subspace methods. Consider the case $n = 10$ with upper diagonal 1, i.e., a Jordan block of dimension 10:

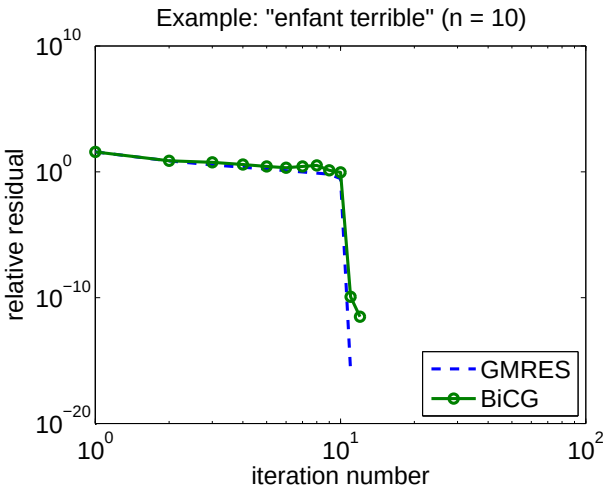


Figure 11.1: Convergence history for Example 11.1.

$$A = \begin{pmatrix} 1 & 1 & & & & & & & & \\ & 1 & 1 & & & & & & & \\ & & 1 & 1 & & & & & & \\ & & & 1 & 1 & & & & & \\ & & & & 1 & 1 & & & & \\ & & & & & 1 & 1 & & & \\ & & & & & & 1 & 1 & & \\ & & & & & & & 1 & 1 & \\ & & & & & & & & 1 & 1 \\ & & & & & & & & & 1 \end{pmatrix}$$

While direct elimination ist trivial, any of the unpreconditioned methods is able to find a reasonable solution before n steps. Figure 11.1 shows the convergence history for GMRES and BiCG; the only difference is that BiCG finds, up to round-off, the exact solution 'already' after $n-1$ steps. For CGS and BiCGSTAB, a very similar behavior is observed.

Trying to give a heuristic interpretation of this effect, one may say that all these methods try to minimize the norm of some polynomial $p_m(A)$ over the spectrum of A. For a highly nonnormal matrix like in Exercise 11.1, the location of the spectrum by no means contains complete information about the behavior of A. For all these methods, significant nonnormality is a critical issue which can, in general, only coped with by appropriate preconditioning techniques. ∎

12 Preconditioning

12.1 General remarks; preconditioned GMRES

Preconditioning transforms the original linear system $Ax = b$ into an equivalent one which is (hopefully) easier to solve by an iterative technique. A good preconditioner M is an approximation for A which can be efficiently inverted, chosen in a way that using $M^{-1}A$ or AM^{-1} instead of A leads to a better convergence behavior. [55]

In fact, CG or GMRES are rarely used directly. In practice, they are usually applied in collaboration with some preconditioner. Historically, the full acceptance of the CG method startet around 1970 (20 years after its invention), when the first practicable preconditioning techniques were designed.

Note that, whereas CG or GMRES are 'general purpose' techniques (CG for SPD matrices, GMRES for general matrices), a preconditioner typically incorporates information about a specific problem under consideration. Thus, the general purpose solver serves as a 'template' which is adapted to a particular class of problems via preconditioning.

There are three general types of preconditioning.

- *Left preconditioning* by a matrix $M_L \approx A$:
$$M_L^{-1} A x = M_L^{-1} b \tag{12.1a}$$
Here, the *preconditioned residual* $\hat{r} = M_L^{-1}(b - Ax) = M_L^{-1} r$ can be interpreted as an approximation for the 'exact correction' $A^{-1} r = x_* - x = -e$, i.e., the (negative) error of the current iterate x.

- *Right preconditioning* by a matrix $M_R \approx A$:
$$A M_R^{-1} y = b, \quad x = M_R^{-1} y \tag{12.1b}$$
This involves a substitution y for the original variable x.

- *Split (two-sided) preconditioning* with a pair of matrices M_L, M_R such that $M_L M_R \approx A$:
$$M_L^{-1} A M_R^{-1} y = M_L^{-1} b, \quad x = M_R^{-1} y \tag{12.1c}$$
Split preconditioning encompasses both the left and the right methods by setting $M_R = I$ or $M_L = I$, respectively.

Naturally, an important feature of the preconditioning matrices will be that they are easily inverted in the sense that the 'approximate problem' $Mv = y$ can be solved relatively cheaply.[56] In the algorithms below, evaluation of $v = M^{-1}y$ is to be understood as the solution of $Mv = y$.

The split preconditioned GMRES method is formulated in Alg. 12.1. It can be understood as simply applying the classical GMRES (Alg. 10.1) to
$$M_L^{-1} A M_R^{-1} y = M_L^{-1} b$$
with exact solution $y_* = M_R x_*$ and a starting vector $y_0 = M_R x_0$. The relations $x_* = M_R^{-1} y_*$, $x_0 = M_R^{-1} y_0$, and setting $x_m = M_R^{-1} y_m = M_R^{-1}(y_0 + V_m u_m) = x_0 + M_R^{-1} V_m u_m$, allows one to write the GMRES iteration in such a way that the auxiliary variables y_i do not appear explicitly but only the original iterates x_i, which are the ones of interest. However, the Arnoldi vectors v_j are to be identified with the transformed variable y.

[55] Here we use the traditional notation M for a preconditioner. Do not mix it up with the meaning of M in Section 5, where it denotes the iteration matrix of a stationary scheme.

Sometimes the inverse $M^{-1} \approx A^{-1}$ is called a preconditioner. The action of M and that of M^{-1} should not be mixed up.

[56] Recall 'rules': Iterative methods only use the matrix-vector product $x \mapsto Ax$; a practical preconditioner M is typically only given as the action $x \mapsto M^{-1}x$; in fact, the matrix M is often not explicitly available.

Algorithm 12.1 Preconditioned GMRES

Compute $r_0 = M_L^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
Allocate the $(m+1) \times m$ matrix $\bar{H}_m$ and initialize elements h_{ij} to zero
for $j = 1, 2, \ldots, m$ **do**
  Compute $w_j = M_L^{-1} A M_R^{-1} v_j$
  for $i = 1, \ldots, j$ **do**
    $h_{ij} = (w_j, v_i)$
    $w_j = w_j - h_{ij} v_i$
  end for
  $h_{j+1,j} = \|w_j\|_2$
  if $h_{j+1,j} = 0$ set $m = j$ and **goto** 13 % lucky breakdown
  $v_{j+1} = w_j / h_{j+1,j}$
end for
Compute u_m as the minimizer of $\|\beta e_1 - \bar{H}_m u\|_2$ and set $x_m = x_0 + M_R^{-1} V_m u_m$

Remark 12.1 The left preconditioned GMRES minimizes the residual norm $\|M_L^{-1}(b - Ax_m)\|_2$ over a suitable Krylov subspace. Right preconditioning, on the other hand, minimizes the original residual $\|b - Ax_m\|_2$. ∎

A natural question is whether left, right, or even split preconditioning is to be preferred. In many cases, the convergence behavior is not significantly different. This is not completely unexpected in view of the fact that the spectra of AM^{-1} (corresponding to right preconditioning) and $M^{-1}A$ (corresponding to left preconditioning) coincide.

Let M be a preconditioner. A standard version of a *split preconditioning* is obtained by means of the LU-decomposition $M = LU$, choosing $M_L = L$ and $M_R = U$. Many preconditioners used in application problems are constructed in this form; often, M is not directly computed but L and U are chosen appropriately, with $LU \approx A$.

12.2 PCG: Preconditioned CG

We discuss left preconditioning of the CG method, i.e., we consider solving

$$M^{-1}Ax = M^{-1}b \tag{12.2}$$

where A is SPD. It will be natural and convenient to choose the preconditioner M to be SPD as well.

The CG algorithm in its original form (Alg. 8.1) is not directly applicable to the new system (12.2), because the matrix $M^{-1}A$ is not SPD in general. However, it is SPD with respect to another inner product on $\mathbb{R}^n$:

Exercise 12.1 Let $A, M \in \mathbb{R}^{n\times n}$ be SPD. Define the M-inner product $(\cdot,\cdot)_M$ by $(x,y)_M = (Mx, y) = x^{\mathrm{T}} M y$. Show:

a) $M^{-1}A$ is selfadjoint with respect to the inner product $(\cdot,\cdot)_M$, i.e., $(M^{-1}Ax, y)_M = (x, M^{-1}Ay)_M$ for all $x, y \in \mathbb{R}^n$.

b) $M^{-1}A$ is positive definite with respect to $(\cdot,\cdot)_M$, i.e., $(M^{-1}Ax, x)_M > 0$ for all $0 \neq x \in \mathbb{R}^n$. ∎

Hence we may apply the CG-algorithm 8.1 to the preconditioned system (12.2), replacing the standard Euclidian inner product $(\cdot,\cdot)$ by the new inner product $(\cdot,\cdot)_M$. With the preconditioned residuals $\hat{r}_k = M^{-1} r_k$ this leads to:

1. $r_0 = b - Ax_0,\ \hat{r}_0 = M^{-1} r_0,\ \hat{d}_0 = \hat{r}_0$
2. **for** $k = 0, 1, \ldots$ until convergence **do**
3. $\quad \alpha_k = (\hat{r}_k, \hat{r}_k)_M / (M^{-1} A \hat{d}_k, \hat{d}_k)_M$
4. $\quad x_{k+1} = x_k + \alpha_k \hat{d}_k$
5. $\quad \hat{r}_{k+1} = \hat{r}_k - \alpha_k M^{-1} A \hat{d}_k$
6. $\quad \beta_k = (\hat{r}_{k+1}, \hat{r}_{k+1})_M / (\hat{r}_k, \hat{r}_k)_M$
7. $\quad \hat{d}_{k+1} = \hat{r}_{k+1} + \beta_k \hat{d}_k$
8. **end for**

The difficulty with this algorithm is that the evaluation of $(\cdot, \cdot)_M$ involved in the computation of the β_k requires the matrix-vector multiplication $x \mapsto Mx$. However, the 'rule' of our preconditioning strategies is that only the matrix-vector multiplications $x \mapsto Ax$ and $x \mapsto M^{-1}x$ should be used. However, it is possible to rewrite this algorithm in a way avoiding matrix-vector multiplications $x \mapsto Mx$, by re-introducing the original residual $r_k = M\hat{r}_k$. Updating r_k along with the vectors x_k, $\hat{r}_k$, and $\hat{d}_k$ then leads to the *preconditioned CG-algorithm* formulated in Alg. 12.2, with a slightly increases memory requirement.

Algorithm 12.2 Left-preconditioned Conjugate Gradient method

1: Compute $r_0 = (b - Ax_0),\ \hat{r}_0 = M^{-1} r_0,\ \hat{d}_0 = \hat{r}_0$
2: **for** $k = 0, 1, \ldots$ until convergence **do**
3: $\quad \alpha_k = (r_k, \hat{r}_k)/(A\hat{d}_k, \hat{d}_k)$
4: $\quad x_{k+1} = x_k + \alpha_k \hat{d}_k$
5: $\quad r_{k+1} = r_k - \alpha_k A \hat{d}_k$
6: $\quad \hat{r}_{k+1} = M^{-1} r_{k+1}$
7: $\quad \beta_k = (r_{k+1}, \hat{r}_{k+1})/(r_k, \hat{r}_k)$
8: $\quad \hat{d}_{k+1} = \hat{r}_{k+1} + \beta_k \hat{d}_k$
9: **end for**

Exercise 12.2 Consider the right-preconditioned system $AM^{-1}u = b$, where $u = Mx$. Show that the matrix AM^{-1} is symmetric positive definite with respect to the inner product $(\cdot, \cdot)_{M^{-1}}$ defined by $(x, y)_{M^{-1}} = (M^{-1}x, y)$. Formulate the (right-)preconditioned CG algorithm. Formulate it in such a way that the auxiliary variable u and the iterates $u_j = Mx_j$ do not explicitly appear in the algorithm, but only the original iterates $x_j = M^{-1}u_j$. ∎

Exercise 12.3 Assume that the Cholesky decomposition $M = LL^{\mathrm{T}}$ of an SPD preconditioner M is available. Consider the split-preconditioned system with $M_L = L$ and $M_R = L^{\mathrm{T}}$, i.e., applying the CG algorithm to the SPD system $L^{-1}AL^{-\mathrm{T}}u = L^{-1}b$.

Show: The iterates $x_m = L^{-\mathrm{T}}u_m$ coincide with those of the left-preconditioned CG method, i.e., Alg. 12.2 with preconditioner M. Furthermore, show that the iterates also coincide with the iterates obtained from right preconditioned CG as developed in Exercise 12.2. (Note that the same argument holds for any SPD preconditioner M implicitly specified by a regular matrix C with $M = CC^{\mathrm{T}}$.) ∎

12.3 Preconditioning in MATLAB

The MATLAB implementations, in particular `pcg` and `gmres`, support left preconditioning, see Sec. 8.6 and 10.3. The preconditioner $M = M_L$ may be specified directly as a matrix or via two matrices M_1 and M_2 such that $M_L = M_1 M_2$ (e.g., in form of an LU-decomposition of M for easy inversion). In both cases, applying the preconditioner means solving systems of the form $My = r$, see Alg. 10.1 and 12.2. Similarly as for A itself (function handle `AFUN`), the backsolving function $r \mapsto M^{-1}r$ may be specified in form of a function handle[57] `MFUN`.

[57] Note: While `AFUN` corresponds to $x \mapsto Ax$, `MFUN` corresponds to $x \mapsto M^{-1}x$.

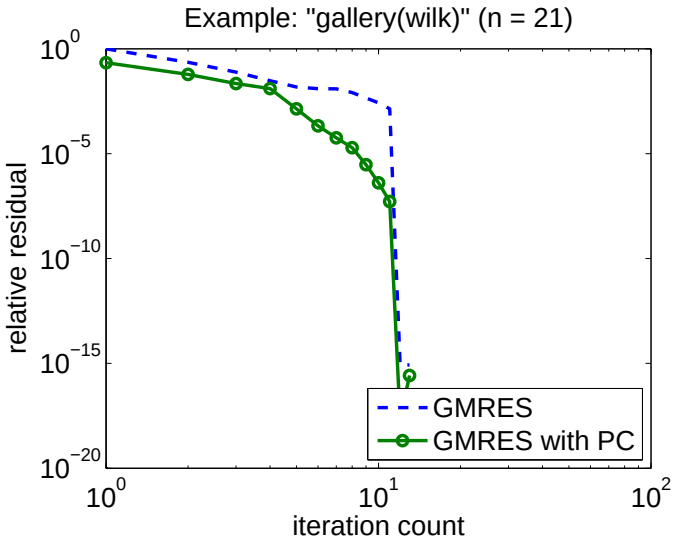


Figure 12.1: Convergence history for Example 12.1.

Example 12.1 This is just for trying out the way of usage: We run the example from `doc gmres` with and without preconditioning; see Fig. 12.1. In this example, A is symmetric tridiagonal and not far from diagonally dominant; $M = \mathrm{diag}(A)$ is therefore a plausible preconditioner.

```
      n = 21;
      A = gallery('wilk',n);
      b = sum(A,2);
      tol = 1e-12;
      maxit = n;
      [x1,flag1,relres1,iter1,resvec1] = gmres(A,b,n,tol,maxit);
      M = diag([10:-1:1 1 1:10]);
      [x2,flag2,relres2,iter2,resvec2] = gmres(A,b,n,tol,maxit,M);
%
%    Or, use this matrix-vector product function
%
%       function y = afun(x,n)
%       y = [0; x(1:n-1)] + [((n-1)/2:-1:0)'; (1:(n-1)/2)'].*x+[x(2:n); 0];
%
%    and this preconditioner backsolve function
%
%       function y = mfun(r,n)
%       y = r ./ [((n-1)/2:-1:1)'; 1; (1:(n-1)/2)'];
%
%    as inputs to GMRES:
%
      [x2,flag2,relres2,iter2,resvec2] = gmres(@(x)afun(x,n),b,n,tol,maxit,@(x)mfun(x,n));
%
      loglog([1:length(resvec1)+1],[resvec1./norm(b,2);norm(b-A*x1,2)/norm(b,2)],'--',...
       [1:length(resvec2)+1],[resvec2./norm(b,2);norm(b-A*x2,2)/norm(b,2)],'-o',...
       'LineWidth',2);
      set(gca,'FontSize',16)
      legend('GMRES','GMRES with PC','Location','SouthEast')
      xlabel('iteration count')
      ylabel('relative residual')
      title('Example: "gallery(wilk)" (n = 21)')
      set(gca,'XTick',[1 10 100 1000 10^4 10^5])
```

■

12.4 Convergence behavior of PCG

We consider the PCG algorithm with left preconditioning as described in Sec. 12.2. The convergence analysis follows along similar lines as in Sec. 8.5.

In its essence, the PCG algorithm coincides with the standard CG algorithm; only the standard Euclidean inner product $(\cdot,\cdot)$ is replaced by the M-inner product $(\cdot,\cdot)_M$. Moreover, $M^{-1}A$ plays the role of A, and PCG is just a Krylov subspace method for solving $M^{-1}Ax = M^{-1}b$. Thus we conclude that the iterates x_m are uniquely characterized by the FOM-type Galerkin orthogonality in the form

$$\hat{r}_m = M^{-1}r_m = M^{-1}(b - Ax_m) \perp_M \hat{\mathcal{K}}_m \tag{12.3}$$

where the Krylov space $\hat{\mathcal{K}}_m = \mathcal{K}_m(M^{-1}A, \hat{r}_0)$ is given by

$$\hat{\mathcal{K}}_m = \operatorname{span}\{\hat{r}_0, \dots, (M^{-1}A)^{m-1}\hat{r}_0\}, \qquad \hat{r}_0 = M^{-1}r_0 = M^{-1}(b - Ax_0)$$

Due to $(\hat{r}, v)_M = (M^{-1}r, v)_M = (r, v)$, the Galerkin orthogonality condition (12.3) involving $\hat{\mathcal{K}}_m$ is equivalent to the original Galerkin condition for the unpreconditioned residual r_m, but with $\mathcal{K}_m$ replaced by $\hat{\mathcal{K}}_m$. Thus we again have (see (8.6))

$$\|x_m - x_*\|_A = \inf_{x \in x_0 + \hat{\mathcal{K}}_m} \|x - x_*\|_A \tag{12.4a}$$

Exactly as in the case of standard CG we obtain, using $\hat{\mathcal{K}}_m = \operatorname{span}\{\hat{r}_0, \dots, (M^{-1}A)^{m-1}\hat{r}_0\}$, that the $x_m \in x_0 + \hat{\mathcal{K}}_m$ can be written in the form

$$x_m = x_0 + p_{m-1}(M^{-1}A)\hat{r}_0$$

with the PCG polynimial $p_{m-1} \in \mathcal{P}_{m-1}$. Thus, with $e_0 = x_0 - x_*$,

$$\begin{aligned} x_m - x_* &= e_0 + x_m - x_0 = e_0 + p_{m-1}(M^{-1}A)\hat{r}_0 = e_0 + p_{m-1}(M^{-1}A)M^{-1}r_0 \\ &= e_0 - p_{m-1}(M^{-1}A)M^{-1}Ae_0 = \big(I - p_{m-1}(M^{-1}A)M^{-1}A\big)e_0 =: q_m(M^{-1}A)e_0 \end{aligned}$$

Thus, the optimality condition (12.4a) gives

$$\|e_m\|_A = \|x_m - x_*\|_A = \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \|q(M^{-1}A)e_0\|_A \tag{12.4b}$$

where the minimum is attained by some optimal polynomial $q = q_m$.

Due to the fact that A and M are SPD, $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}$ is also SPD, and the 'preconditioned spectrum'

$$\sigma(M^{-1}A) = \sigma(M^{-\frac{1}{2}}AM^{-\frac{1}{2}})$$

is positive. Thus we arrive at

$$\|e_m\|_A = \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \|q(M^{-1}A)e_0\|_A \le \min_{\substack{q \in \mathcal{P}_m \\ q(0)=1}} \max_{i=1\dots n} |q(\lambda_i)| \cdot \|e_0\|_A \tag{12.5}$$

where the $\lambda_i > 0$ are the eigenvalues of $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}$ and also of $M^{-1}A$.

The outcome is the same as in Theorem 8.1 for the CG method, with A replaced by $M^{-1}A$. Thus, identity (12.5) shows:

The convergence of PCG depends on the spectrum of the preconditioned matrix $M^{-1}A$.

As in the case of the CG method, an overall (worst case) bound involves the ratio of the largest and the smallest eigenvalue of $M^{-1}A$. We formulate this as an exercise.

Exercise 12.4

a) Show: The speed of convergence of the PCG iteration can be estimated by

$$\|e_m\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|e_0\|_A, \quad \kappa = \kappa_\sigma(M^{-1}A) = \frac{\lambda_{max}}{\lambda_{min}}$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalues of $M^{-1}A$, and $\kappa_\sigma(M^{-1}A) = \lambda_{max}/\lambda_{min}$ is the spectral condition number (in general, this is not identical with $\kappa_2(M^{-1}A)$).

b) Show that a characterization of λ_{min} and λ_{max} which may be easier to check is the following: λ_{min} is the largest and λ_{max} is the smallest number such that

$$\lambda_{min}\, M \leq A \leq \lambda_{max}\, M$$

is valid.

c) Show: If $\alpha, \beta > 0$ can be found such that

$$\alpha\, M \leq A \leq \beta\, M$$

then $\alpha \leq \lambda_{min}$, $\beta \geq \lambda_{max}$, and thus, $\kappa_\sigma(M^{-1}A) \leq \beta/\alpha$. ∎

12.5 Preconditioning techniques in general

As mentioned before, CG and GMRES are fairly general solution techniques. In practice, the preconditioner $M \approx A$ is chosen in dependence on properties of the matrix A and is *the* key component of the iterative solution method. A few general comments concerning the choice/design of a preconditioner are:

- $M^{-1}A$ should be 'close' to the identity matrix or at least have a spectrum that is clustered.
- The operation $x \mapsto M^{-1}x$ should be cheap/easy to perform. Note that this condition depends also on the computer architecture employed. As we will see below, a Gauss-Seidel preconditioner may be better than a Jacobi preconditioner (in terms of iteration count), but a Jacobi preconditioner requires less communication and is often more natural and efficient to realize on a parallel computer.

A good preconditioner M typically depends on the problem under consideration. For many preconditioning techniques, theoretical results about their performance are available (e.g., via bounds for $\kappa_\sigma(M^{-1}A)$ in the case of PCG), but often, testing with representative examples is equally important.

In this section we discuss a few preconditioning techniques that could be applied in many circumstances. We point out that these techniques are fairly general and may not always be very effective. However, since they are easy to use they are often worth an initial investigation. A more detailed discussion of these techniques can be in found in [19].

Classical iterative schemes as preconditioners.

Every linear stationary iteration

$$x_{k+1} = x_k + N(b - Ax_k) = (I - NA)\,x_k + N\,b = G\,x_k + N\,b \tag{12.6}$$

induces a preconditioner. Here, for the iteration matrix of the scheme we write $I - NA = G$. [58] Convergent linear iterations satisfy $\rho(G) < 1$, i.e., G is 'small'; in other words: NA is 'close to' I because N plays the role of an approximate inverse. Hence, the matrix $W = N^{-1}$ may be used as a preconditioner M – it is an approximation for A.

[58] This notation G is different from Sec. 5 ($G{=}M$); in the present context, M always denotes a preconditioner.

For the classical splitting methods (Jacobi, Gauss-Seidel, SOR, SSOR) such a preconditioner $W = N^{-1}$ can be read off directly from the identity $G = I - NA$; the formulas given in Sec. 5.2, p. 30. For example, with $A = L + D + U$,

- Jacobi preconditioner: $M_J = D$
- Gauss-Seidel preconditioner: $M_{GS} = L + D$
- Symmetric Gauss-Seidel preconditioner: $M_{SGS} = (L + D)D^{-1}(D + U)$

Jacobi preconditioning may be interpreted as a diagonal rescaling of the original matrix A as $D^{-1}A$, resulting in a matrix with unit diagonal.

We stress that the action of these preconditioners can be easily realized since the matrices $L + D$ and $D + U$ have triangular structure. We also note that M_{SGS} is symmetric if the underlying matrix A is. This allows us to employ it as a preconditioner for PCG.

More generally, applying several steps of a convergent linear stationary iteration (12.6) define a preconditioner. Performing k steps means that $G = I - N\,A$ is replaced by

$$G^k = (I - NA)^k = I - N_k\,A \quad \text{with} \quad N_k = (I - G^k)\,A^{-1} \approx A^{-1}$$

and the corresponding preconditioner is

$$M = N_k^{-1} = A\,(I - G^k)^{-1} \approx A \tag{12.7}$$

In the following theorem we relate the quality of of a preconditioner based on a stationary linear iteration with iteration matrix G to its convergence speed, i.e., to the contraction factor $\|G\| \le \xi < 1$, assuming the iteration is convergent in some norm $\|\cdot\|$.

Theorem 12.1 *Let $M \approx A$ be the preconditioner* (12.7) *defined by k steps of the linear stationary iteration* (12.6)*, with an iteration matrix $G = I - NA$ satisfying $\|G\| \le \xi < 1$. Then,*

(i) $\|M^{-1}A\| \le 1 + \xi^k$

(ii) $\operatorname{cond}(M^{-1}A) \le \dfrac{1 + \xi^k}{1 - \xi^k}$

(iii) If $\|G\|$ is an ℓ_2-contraction, i.e., if $\|G\|_2 \le \xi < 1$, then $M^{-1}A$ is positive definite; in particular,

$$(M^{-1}Ax, x) \ge (1 - \xi^k)\,\|x\|_2^2 \quad \text{for all} \quad x \in \mathbb{R}^n$$

Proof: From (12.7) we have $M^{-1}A = I - G^k$.

- *ad (i):* From (12.7),
$$\|M^{-1}A\| = \|(I - G^k)\| \le 1 + \|G^k\| \le 1 + \xi^k$$
- *ad (ii):* To estimate $\|(M^{-1}A)^{-1}\| = \|(I - G^k)^{-1}\|$ from above, we consider an arbitrary $x \in \mathbb{R}^n$ and estimate $\|(I - G^k)\,x\|$ from below: With $\|G^k x\| \le \xi^k\,\|x\| < \|x\|$ we have
$$\|(I - G^k)\,x\| \ge \|x\| - \|G^k\,x\| \ge (1 - \xi^k)\,\|x\|$$
This implies $\|(M^{-1}A)^{-1}\| \le (1 - \xi^k)^{-1}$, and together with (i) this gives (ii).

- *ad (iii):* For arbitrary $x \in \mathbb{R}^n$,

$$\begin{aligned}(M^{-1}Ax, x) &= ((I - G^k)x, x) = (x, x) - (G^k x, x) \\ &\geq \|x\|_2^2 - \|G^k x\|_2 \, \|x\|_2 \geq (1 - \xi^k) \, \|x\|_2^2\end{aligned}$$

which proves (iii). □

Remark 12.2 Theorem 12.1, (iii) shows that under the assumption $\|G\|_2 \leq \xi < 1$ the preconditioned matrix $M^{-1}A$ is positive definite and satisfies the coercitivity condition (10.14) with $\tau = 1 - \xi^k$, entailing the residual decay estimate (10.15) for preconditioned GMRES. Therefore, under these circumstances preconditioning leads to regular convergence behavior, in contrast to the 'arbitrarily bad' convergence behavior encountered in examples like that one considered in Exercise 11.1. ∎

Remark 12.3 The convergence criterion $\rho(G) < 1$ is rather strict. On the other hand, if a preconditioner is only 'good', it is better than nothing. In particular, something like

$$\|M^{-1}A\| \leq C, \quad \|A^{-1}M\| \leq C, \qquad C \text{ a moderate-sized constant}$$

is sufficient to ensure a satisfactory convergence property of a preconditioned Krylov subspace method. Cf. also the related Exercise 12.4 for the case of PCG – here, spectra are involved instead of norms.

This reasoning shows that the choice of an appropriate preconditioner is in some sense easier than searching for one that leads to a convergent linear iteration scheme, because the requirements on the preconditioner are less strict. This is why the combination of Krylov methods with preconditioning has become so popular.

In practice, the linear system $Ax = b$ to be solved is a member of a family of systems with increasing dimension n (typically, $n \to \infty$ for some discretization parameter $h \to 0$). In this case, a *perfect* preconditioner may characterized by the property

$$\|M^{-1}A\| \leq C, \quad \|A^{-1}M\| \leq C, \qquad C \text{ a moderate-sized constant } \textit{independent of } n,$$

of course at moderate computational cost for the preconditioning step. ∎

Incomplete decompositions: ILU and its variants.

A popular class of preconditioners are incomplete decompositions. Such a type of preconditioning is also one of the historically earliest examples (around 1970) which made the PCG method become widely accepted.

In Sec. 3 on direct methods we have seen that computing the LU-decomposition of a sparse matrix A may result in considerable fill-in. In view of the fact that a preconditioner just needs to be an approximation to A, one may consider computing an *approximate decomposition* $\tilde{L}\tilde{U} \approx A$, where $\tilde{L}$ and $\tilde{U}$ should also be sparse. Choosing $M = \tilde{L}\tilde{U}$ then leads to an efficient evaluation of $x \mapsto M^{-1}x$, since $y = \tilde{L}^{-1}x$ and $M^{-1}x = \tilde{U}^{-1}y$ can easily be realized by forward and backward substitution.

MATLAB has built-in functions to compute these incomplete factors as well as few other variants, e.g., MILU (see the book [19] for further details). See: `help ilu`, `help ichol`.

ILU(0).

Let $NZ(A)$ denote the set of index pairs (i,j) with $A_{i,j} \neq 0$, the 'sparsity pattern' of A. In the ILU(0) technique, the factors $\tilde{L}$ and $\tilde{U}$ are required to satisfy:

$$\text{(i) } \tilde{L} \text{ and } \tilde{U} \text{ have the same sparsity pattern as } A, \quad \text{i.e., } NZ(\tilde{L}+\tilde{U}) = NZ(A) \tag{12.8a}$$

$$\text{(ii) the non-zero entries of } \tilde{L} \text{ and } \tilde{U} \text{ are such that } \quad (\tilde{L}\tilde{U})_{i,j} = A_{i,j} \text{ for all index pairs } (i,j) \in NZ(A) \tag{12.8b}$$

Algorithmically, this can be realized by modifying standard LU-decomposition in such a way that only the non-zero entries of the factors L and U are computed; the remaining fill-in is simply ignored. To see how this works, consider e.g. the Crout scheme for computing an LU-decomposition (see [2]): Assume, for example, that $a_{21} = a_{3n} = 0$. The the scheme is modified enforcing $\ell_{21} = 0$ and $u_{3n} = 0$:

$$\begin{array}{|ccccc|}\hline 1 & & & & \\ \mathbf{0} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & & \ddots & \ddots & \\ \ell_{n1} & \dots & \dots & \ell_{n,n-1} & 1 \\ \hline \end{array} \cdot \begin{array}{|ccccc|}\hline u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ & u_{22} & u_{23} & \dots & u_{2n} \\ & & u_{33} & \dots & \mathbf{0} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \\ \hline \end{array} = \begin{array}{|ccccc|}\hline a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ \mathbf{0} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & \mathbf{0} \\ \vdots & & & \ddots & \vdots \\ a_{n1} & \dots & \dots & \dots & a_{nn} \\ \hline \end{array}$$

Now, in the usual way, by comparing coefficients, we successively compute

- the first row of U (= first row of A),
- the first column of L,
- the second row of U,
- the second column of L,
- the third row of U,
- and so on.

Here, the equations corresponding to the indices in $NZ(A)$ are processed in the usual way, and the equations involving the zero entries of A are simply *ignored.* If N_0 is the number of non-zero entries in A, we have N_0 less unknowns, and N_0 less parameters ℓ_{ij} and $u_{i,j}$ to be determined (compared to a full LU-decomposition).

An equivalent version in the spirit of Gaussian elimination is executed by Alg. 12.3. The algorithm overwrites the matrix A with the factors $\tilde{L}$ and $\tilde{U}$ (as usual, the diagonal contains the diagonal of $\tilde{U}$ since the diagonal of $\tilde{L}$ has fixed entries 1).

Comments concerning existence, uniqueness, and implementation of ILU(0):

Algorithm 12.3 ILU(0) – KIJ-variant

% overwrites A with incomplete LU-decomposition; entries $L_{i,i} = 1$ not stored

```
1: for k = 1, 2, ..., n − 1 do
2:   for i = k+1 ... n and (i, k) ∈ NZ(A) do
3:     a_ik = a_ik / a_kk
4:     for j = k+1 ... n and (i, j) ∈ NZ(A) do
5:       a_ij = a_ij − a_ik a_kj
6:     end for
7:   end for
8: end for
```

- There is a priori no guarantee that an incomplete decomposition can be computed. For certain classes of matrices such as *M-matrices,*, however, it is known that an incomplete LU-decomposition exists, see [19, Theorem 10.1]. A matrix A is called an M-matrix if

  - $A_{i,i} > 0 \quad \forall\, i$
  - $A_{i,j} \leq 0 \quad \forall\, i, j \quad \text{with} \quad i \neq j$
  - A is invertible, with $(A^{-1})_{i,j} \geq 0 \quad \forall\, i, j$

  In particular, the matrices A from the Poisson example 2.2 are M-matrices.

- By construction, we have
$$A = \tilde{L}\tilde{U} - R$$
with a 'residual matrix' R.

- In practice, instead of Alg. 12.3 a different variant of Gaussian elimination is employed. For sparse matrices A which are stored row by row (e.g., in the CRS-format) it is better to rearrange the three loops (over i, j and k) such as to operate on whole rows of A (row-wise elimination). This leads to the so-called IKJ-variant of (incomplete) LU-decomposition.

Remark 12.4 ILU(0) employs the same sparsity pattern as the matrix A. One could employ other sparsity patterns for the factors $\tilde{L}$ and $\tilde{U}$. One way to choose them is done in the ILU(p) methods (see [19]), where $p \in \mathbb{N}_0$ is a measure for the amount of fill-in we wish to accept.

A drawback of the ILU(0) strategy is that it ignores the actual size of the entries of the exact factors L and U. ILUT based on 'thresholding' tries to incorporate this into the method. In this approach the sparsity pattern of the factors $\tilde{L}$ and $\tilde{U}$ is determined *on the fly* during the decomposition process, since the magnitude of the entries of L and U is not a priori known. ∎

Incomplete Cholesky (ICC).

For SPD matrices A, it is common to use the Cholesky decomposition $A = LL^{\mathrm{T}}$ instead of the LU-decomposition. This sparked the development of *incomplete Cholesky decomposition* techniques analogous to ILU and ILUT.

For classical Cholesky applied to an SPD matrix A, all occurring square roots are well-defined positive numbers, such that is always guaranteed that the diagonal of L is positive. If we use a version of incomplete Cholesky decomposition for preconditioning in PCG, we have to ensure that $M = \tilde{L}\tilde{L}^{\mathrm{T}}$ is positive definite. For this purpose it may be necessary to 'strengthen' the diagonal entries.

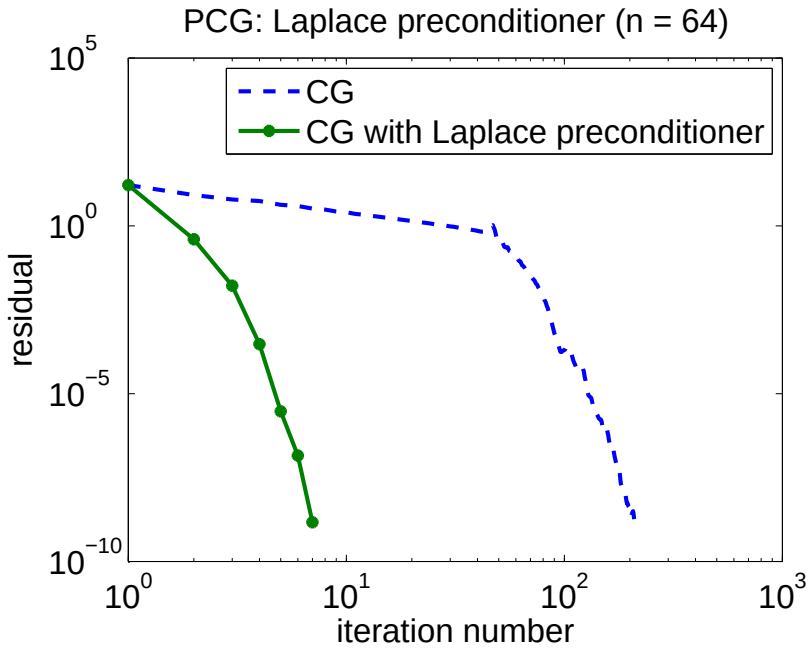


Figure 12.2: Convergence history for Example 12.2.

Fast direct solvers for simplified problems. A Poisson preconditioner.

For special prominent problem types, fast direct solvers are sometimes available. An example ist the fast Poisson solver discussed in Sec. 4. If, for instance, we wish to solve a stationary diffusion-reaction equation of the form

$$-\Delta u(x,y) + c(x,y)\,u(x,y) = f(x,y) \quad \text{on } \Omega, \qquad u|_{\partial\Omega} = 0$$

on a rectangular domain Ω, then – after discretization via FD or FEM – it is tempting to use the fast solver for the Poisson case (with $c(x,y) = 0$) for preconditioning.

After FD discretization as in Example 2.2, the problem takes the form of a linear system $Ax = b$ with $A = M + h^2 C$, where M is the Poisson FD matrix approximating $-h^2\Delta$ and $C = \mathrm{diag}(c_{ij})$ with $c_{i,j} = c(x_i, y_j)$. For $c(x,y) \geq 0$ the given problem is well-posed. With M as preconditioner we obtain

$$\|M^{-1}A\|_2 = \|I + M^{-1}h^2 C\|_2 \leq 1 + h^2\,\|M^{-1}\|_2\,\|C\|_2 \approx 1 + \frac{h^2\,\|c\|_\infty}{2\,h^2\,\pi^2} = 1 + \frac{\|c\|_\infty}{2\,\pi^2}$$

independent of $h = 1/(n+1)$, and by a similar argument using the uniform boundedness of A^{-1} we also can argue that $\|A^{-1}M\|_2$ is bounded independently of h. Thus, our 'Poisson preconditioner' is expected to be very effective – on the other hand it cannot be called 'cheap' because in each step a Poisson-like problem has to be solved.

Example 12.2 We compare standard CG and 'Poisson-preconditioned CG' for $n = 64$ and $c(x,y)$ such that $\|C\|_2 \approx 10$. CG satisfies the residual tolerance 1E-10 after 210 steps; with preconditioning we need 6 steps to converge; see Fig. 12.2. ∎

The preconditioner from Example 12.2 is easy to implement, but still the computational effort is significantly higher than for a single Poisson solution. For such types of problems more advanced and effective techniques exist which are also applicable for more general differential operators and geometries. We will discuss such techniques to some extent in Sec. 13 and 14.

Approximate inverses.

For notational convenience and variety of presentation, we here consider the generation of right preconditioners.

Starting from the observation that the (right) preconditioner M should satisfy $AM^{-1} \approx I$, *approximate inverse preconditioners* are directly aiming at a matrix $M^{-1} =: X$ (hopefully invertible) such that

$$\|AX - I\|$$

is small in some norm. The matrix-vector multiplication $x \mapsto Xx$ is then taken as the action of the preconditioner. For this to be viable, the matrix X needs to be sparse. One possible way of generating X is to prescribe its sparsity pattern and then try to construct X as the minimizer of

$$\min\{\|AX - I\|_F : X \in \mathbb{R}^{n\times n} \text{ with prescribed sparsity pattern}\} \tag{12.9}$$

where the Frobenius matrix norm is given by $\|X\|_F^2 = \sum_{i,j=1}^n X_{i,j}^2$. (An alternative to prescribing the sparsity pattern of X would again be to employ some dropping strategies, e.g., to keep only the p largest entries of each row or column of X.)

The choice of the Frobenius norm is, of course, somewhat arbitrary, but is convenient since minimizing the objective functional

$$\varphi(X) := \|AX - I\|_F^2$$

is equivalent to minimizing the $\|\cdot\|_2$-norm of the columns of $AX - I$, seen as one 'long vector'. The corresponding inner product is

$$(X, Y)_F = \sum_{i,j=1}^n X_{i,j} Y_{i,j} = \operatorname{trace}(Y^{\mathrm{T}} X), \qquad \|X\|_F^2 = (X, X)_F$$

Thus, (12.9) is an unconstrained minimization problem, namely simply a standard least squares problem associated with the over-determined system $AX = I$ of n^2 equations in m unknowns, where m is the number of prescribed non-zero entries for the solution X.

The only formal difference to a standard least squares problem consists in the fact that a matrix X is looked for. Instead of rewriting this in terms of 'long vectors' and proceeding in a standard way, it is more natural here to stick to the matrix formulation. At the desired solution X, the gradient of φ must vanish. To represent the gradient in form of a matrix $D = D(X) \in \mathbb{R}^{n\times n}$ (analogous to A and X), we consider the local linearization of $\varphi(X) = \|AX - I\|_F^2$ at some matrix X in the form

$$\varphi(X+H) = \varphi(X) + (D\varphi(X), H)_F + \mathcal{O}(\|H\|_F^2)$$

Then, $D(X)$ is the matrix representation for the gradient (the Fréchet derivative) of φ at X. The question is now to determine $D(X)$.

Exercise 12.5 Expand $\varphi(X+H)$ to conclude

$$\varphi(X+H) = \varphi(X) - 2\,(A^{\mathrm{T}}R, H)_F + \|AH\|_F^2 \quad \text{with the residual matrix} \quad R = I - AX \tag{12.10}$$

Note that this expansion is, in fact, exact because $\varphi(X)$ is a quadratic functional. ∎

Equation (12.10) shows that the matrix representation for the gradient $D(X)$ is given by

$$D(X) = -2\,A^{\mathrm{T}}R = 2\,A^{\mathrm{T}}\,(AX - I)$$

and the system

$$D(X) = 0 \quad \Leftrightarrow \quad A^{\mathrm{T}}(AX - I) = 0$$

is the system of normal equations for our minimization problem. For the prescribed sparsity pattern of X with m entries, its dimension is $m \times m$.

In practice (for large matrices A), direct solution of this system is often out of scope. As an alternative, an iterative technique in the like steepest descent (SD) may be used. The formulation of the corresponding algorithm is analogous to the SD algorithm from Sec. 7.1, but with respect to the inner product $(X, Y)_F = \mathrm{trace}(Y^{\mathrm{T}}X)$, with the negative gradient matrix $D(X)$ as search direction, see Alg. 12.4.

However, in this approach the iterates X will tend to become denser at each step. Therefore it is essential to apply a numerical dropping strategy for the undesired elements. But then descent is no longer guaranteed, i.e., we do not necessarily have $\varphi(X_{new}) < \varphi(X_{old})$. An alternative would be to apply numerical dropping to the search direction S before updating X. However, this does not directly control the amount of fill-in in the iterates X. See [19] fur further remarks.

Algorithm 12.4 Global Steepest Descent algorithm

1: Choose an initial guess X with given sparsity pattern
2: Until convergence **do**
3: $\quad R = I - AX, \quad S = A^{\mathrm{T}}R$
4: $\quad \alpha = \|S\|_F^2 \,/\, \|AS\|_F^2$
5: $\quad X = X + \alpha S$
6: $\quad$ Apply numerical dropping to X
7: **end do**

Alternatively, it has been proposed to use the residual matrix $R = I - AX$ as the search direction, see Alg. 12.5.

Algorithm 12.5 Global Minimal Residual Descent algorithm

1: Choose an initial guess X with given sparsity pattern
2: Until convergence **do**
3: $\quad R = I - AX$
4: $\quad \alpha = \mathrm{trace}(R^{\mathrm{T}}AR) \,/\, \|AR\|_F^2$
5: $\quad X = X + \alpha R$
6: $\quad$ Apply numerical dropping to X
7: **end do**

Exercise 12.6 Show that (apart from the dropping step) Alg. 12.4 is indeed the steepest descent algorithm for the problem under consideration. (Cf. the derivation of the SD algorithm in Sec. 7.1, but note the different inner product.) Furthermore, show that formulation of Alg. 12.5 is correct. ∎

In both algorithms, the residual matrix R needs to be explicitly stored. The occurring scalar quantities, including $\|AS\|_F^2$ and $\mathrm{trace}(R^{\mathrm{T}}AR)$ can be computed from the successive columns of AS, etc., which can be successively computed, used, and discarded again. Thus, the matrix products AS and $R^{\mathrm{T}}AR$ need not to be explicitly stored.

Column-oriented technique.

The objective functional $\varphi(X) = \|AX - I\|_F^2$ can also be expressed as

$$\varphi(X) = \sum_{j=1}^{n} \|Ax_j - e_j\|_2^2$$

in column-wise notation. Since the j-th column x_j of X occurs only in the j-th term, minimization of φ evidently decouples into n individual minimization problems

$$\varphi_j(X) := \|Ax_j - e_j\|_2^2 \mapsto \text{min!}, \quad j = 1 \ldots n \tag{12.11}$$

An attractive feature of this formulation is that the minimization can be done for all columns in parallel. Each minimization can be performed by taking a sparse initial guess and solving approximately the n parallel linear subproblems (12.11) with a few steps of a nonsymmetric descent-type method. A basic version based on steepest descent in residual direction(s) is formulated in Alg. 12.6.

Algorithm 12.6 Approximate inverse via MR Iteration

```
X = X_0        % initial guess for X, e.g., X = I
for each column j = 1...n do
    set x_j = X_0 e_j
    for k = 1... until some stopping criterion is met do
        r_j = e_j - A x_j
        α_j = (r_j, A r_j) / (A r_j, A r_j)
        x_j = x_j + α_j r_j
        apply a dropping strategy for the entries of x_j
    end for
end for
```

In [19], this class of preconditioning techniques is studied in more detail. Among others, this includes also 'factored approximate inverses', a more systematic approach of ILU type.

Polynomial preconditioners.

Another class of preconditioners – in the spirit of finding an approximate inverse – are polynomial preconditioners. From the Cayley-Hamilton Theorem we know that an invertible matrix $A \in \mathbb{R}^{n\times n}$ can be represented as $A^{-1} = p(A)$ for some $p \in \mathcal{P}_{n-1}$. One may hope to find good approximations using polynomials of much smaller degree. The following choices are just two examples. In both cases, computing a (left-)preconditioned residual amounts to evaluating a term of the form $p(A)x$ with some low degree polynomial p.

- *Neumann polynomials:* If A is SPD, then the system $Ax = b$ is equivalent to $\omega Ax = \omega b$ $(\omega \neq 0)$. If we choose the damping parameter $0 < \omega < \|A\|_2$, then $\|I - \omega A\|_2 < 1$, i.e., the Neumann series

$$(\omega A)^{-1} = \sum_{i=0}^{\infty} (I - \omega A)^i$$

  is convergent. Hence, truncating this series, i.e., taking $M^{-1} = \sum_{i=0}^{m} (I - \omega A)^i$ for rather small m may lead to a good preconditioner for the linear system $\omega Ax = \omega b$.

- *Chebyshev polynomials:* We seek a preconditioner $M^{-1} \approx A^{-1}$ in the form $M^{-1} \approx p_m(A)$ for some $p_m \in \mathcal{P}_m$. Aiming at minimizing $\|I - p_m(A)A\|$ in some norm we are led – as in our discussion of Chebyshev acceleration in Sec. 6 – to minimizing the spectral radius $\rho(I - p_m(A)A)$, i.e., to find $p_m \in \mathcal{P}_m$ such that $\max\{|1 - \lambda p_m(\lambda)| : \lambda \in \sigma(A)\}$ is minimized.

  In practice, the spectrum $\sigma(A)$ is not known but possibly an inclusion set $E \subseteq \mathbb{C}$ with $\sigma(A) \subseteq E$. In this case, we would seek $p \in \mathcal{P}_m$ such that $\max_{x \in E} |1 - x\,p(x)|$ is minimized. If, for example, E is an interval on the real line, then the minimizer $p_m \in \mathcal{P}_m$ is given by the scaled Chebyshev polynomial described in Corollary 6.1. The preconditioner is then taken as $x \mapsto M^{-1}x = p_m(A)\,x$.

Block preconditioners.

For most preconditioners also block variants exist, e.g., for a matrix A given in block form

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,k} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k,1} & A_{k,2} & \cdots & A_{k,k} \end{pmatrix}$$

one may choose the preconditioner

$$M_J^{-1} = \begin{pmatrix} A_{1,1}^{-1} & & & \\ & A_{2,2}^{-1} & & \\ & & \ddots & \\ & & & A_{k,k}^{-1} \end{pmatrix}$$

which is the *block Jacobi preconditioner*. Here, of course, one assumes that the diagonal blocks $A_{i,i}$ are cheaply invertible.

12.6 Further numerical examples

Poisson problem.

The results obtained with the preconditioning methods described when applied to CG and the SPD matrix 'poisson' from the MATLAB gallery (cf. Example 2.2) are displayed in Table 12.1.

dimension $N \times N$	N = 64	N = 256		N = 1,024		N = 4,096		N = 16,384	
original	10	28	(0.06)	59	(0.38)	119	(2.31)	239	(17.36)
ICC(0)	11	19	(0.11)	30	(0.50)	55	(3.74)	100	(39.16)
SGS	11	19	(0.11)	34	(0.55)	60	(2.74)	118	(22.19)

Table 12.1: Iteration counts and times obtained using MATLAB's implementation of PCG and ICC applied to the 'Poisson' matrix, with a convergence tolerance of 10^{-8}.

Note that, with the exception of diagonal preconditioning, a decrease in the number of iterations required to achieve convergence is obtained; however, the computational time is not reduced. This is probably due to the relatively cheap cost of a CG iteration for this pentadiagonal matrix compared to the setup of the preconditioning matrix and its inversion. The reduction in the number of iterations required corresponds to a reduction in the condition number of the preconditioned system.

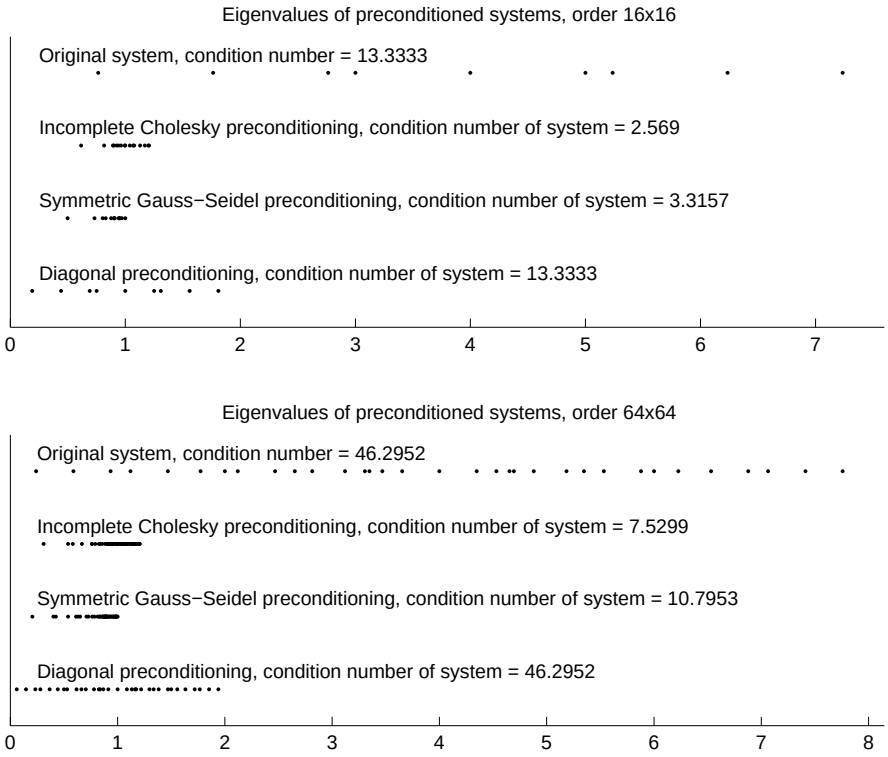


Figure 12.3: Eigenvalues of preconditioned versions of the matrix 'Poisson' from the MATLAB gallery.

For low values of n the eigenvalues of these systems are shown in Fig. 12.3. This also includes diagonal preconditioning which shifts the spectrum but has no effect on convergence because the diagonal is constant for this problem.

In general, preconditioning by the diagonal generates some benefit and due to the simplicity of this method it is often worth an initial investigation – it is simply a rescaling of the problem which might be useful for the case where the matrix elements strongly varying in size.

A sterner test for preconditioned CG can be constructed from one of the test matrices available from the large collection at the `Matrix Market`[59]. The matrix selected is NOS6, a matrix resulting from discretizing Poisson's equation on an L-shaped domain. The matrix is SPD, dimension 675×675 and with an estimated condition number of 8×10^6. The result obtained are displayed in Table 12.2. For this problem we observe that the diagonal preconditioning did a relatively good job, mostly due to its computational cheapness.

	iteration count	time (seconds)
original	1408	2.917
ICC(0)	87	0.599
SGS	42	0.277
Diagonal	103	0.388

Table 12.2: Iteration counts and times obtained using MATLAB's implementation of PCG and ICC applied to the NOS6 matrix, with a convergence tolerance of 10^{-8}.

[59] `http://math.nist.gov/MatrixMarket`: *A visual repository of test data for use in comparative studies of algorithms for numerical linear algebra, featuring nearly 500 sparse matrices from a variety of applications, as well as matrix generation tools and services.*

	iteration count	time (seconds)
original	$93 + (84 \times 100)$	123.593
ILU(τ) $\tau = 10^{-2}$	10	0.125
ILU(0)	29	0.359
SGS	$31 + (11 \times 100)$	20.179

Table 12.3: Iteration counts and times obtained using MATLAB's implementation of restarted GMRES (restart value 100) and ILU applied to the PORES3 matrix, with a convergence tolerance of 10^{-8}.

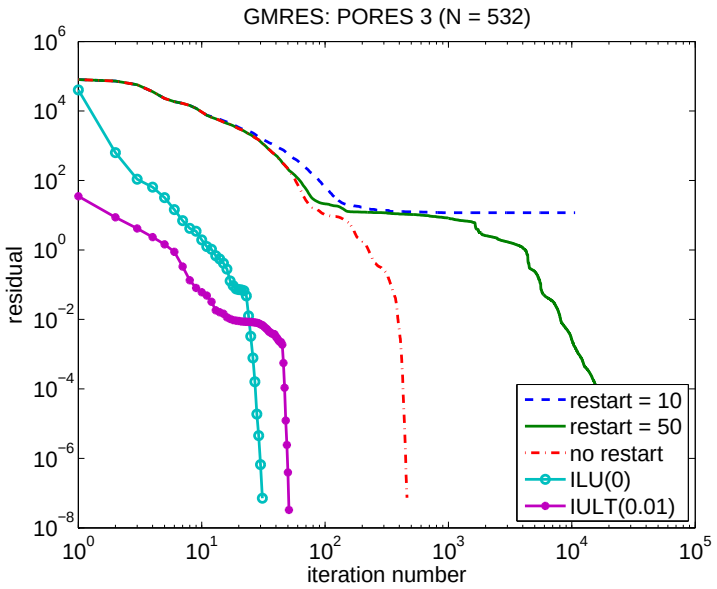


Figure 12.4: Performance of restarted preconditioned GMRES for PORES3

An unsymmetric problem.

Computational results obtained with the preconditioned GMRES algorithm are displayed in Fig. 12.4 and Table 12.3. The matrix considered is PORES3 from the `Matrix Market`. PORES3 an unsymmetric matrix resulting from modeling oil reservoirs, with dimension 532×532 and an estimated condition number of 6.6×10^5.

From Table 12.3 we see that GMRES is quite responsive to preconditioning: ILU(0) and ILUT(10^{-4}) perform very well. Fig. 12.4 shows the convergence behavior (residual vs. iteration number) for preconditioned GMRES and restarted GMRES with different restart values. Note a difficulty of restarted GMRES which quite often arises: If the restart value is too small, then restarted GMRES does not converge.

Although the preconditioning techniques presented here can be quite effective in accelerating the convergence rate of Krylov subspace methods, we typically find that, when applied to discretizations of PDEs, the number of iterations required to achieve convergence remains linked to size of the problem under consideration.

Ideally, however, one would like to have preconditioners that perform well *irrespective of the problem size*. This goal, the *Holy Grail* of preconditioning, can be achieved for certain problem classes. For example, *multigrid* and the so-called *BPX preconditioner* achieve this goal for discretizations stemming from elliptic partial differential equations.

13 Multigrid Methods (MG)

The performance of classical Krylov techniques such as CG and GMRES coupled with the simple preconditioners discussed above typically deteriorate as the problem size increases, i.e., the number of iterations required to reach a given accuracy increases as the problem size increases. We now show how multigrid (MG) techniques can overcome these difficulties. MG can itself be viewed as a 'stand alone' iterative scheme. In particular, however, it can be very successfully employed as a preconditioner for CG or GMRES.

Since a MG method combines related problems of different dimensions, we indicate the dimension via an index, see (13.1a). For iteration, we now use upper indexing, u^m instead of u_m.

13.1 1D elliptic model problem. FD vs. FEM approximation

In order to illustrate the main MG idea, we consider solving the linear system arising from the 1D Poisson Example 2.1. It will be convenient to scale the tridiagonal matrix of Example 2.1 in a different way, as usual in the FEM context. We consider

$$A_N u_N = b_N \tag{13.1a}$$

where the tridiagonal matrix $A_N \in \mathbb{R}^{(N-1)\times(N-1)}$ is given by

$$A_N = \frac{1}{h_N}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \tag{13.1b}$$

Here the parameter h_N is the mesh size given by

$$h_N = \frac{1}{N} \tag{13.1c}$$

In the FD context, the right-hand side b_N is given by $b_N = h_N \big(f(x_1), f(x_2), \ldots, f(x_{N-2}), f(x_{N-1})\big)^{\mathrm{T}}$ with $x_i = i\,h_N$. The solution of the linear system (13.1a) represents approximations to the nodal values of the exact solution $u_*(x)$ at the mesh points x_i.

In the FEM context (see Appendix A), the approximating system is setup in a different manner. Here one proceeds from the weak formulation of the original problem or its formulation as minimization problem, cf. Remark 7.1. In the simplest case, an approximation to u_* is sought for in the space $\hat{U}_N$ of piecewise linear functions $\hat{u}_N = \hat{u}_N(x)$ with respect to the given mesh, with zero boundary values. With[60]

$$a(\hat{u}_N, \hat{v}_N) = \int_{\Omega=(0,1)} \hat{u}_N' \hat{v}_N', \qquad b(\hat{v}_N) = (f, \hat{v}_N)_{L^2(\Omega)} = \int_{\Omega=(0,1)} f\,\hat{v}_N \tag{13.2a}$$

the FEM approximation $\hat{u}_N^* \approx u_*$ is defined as the solution of the discrete problem in weak formulation, via the Galerkin requirement

$$\textbf{\textit{Find}}\ \ \hat{u}_N \in \hat{U}_N\ \ \textbf{\textit{such that}}\ \ a(\hat{u}_N, \hat{v}_N) = b(\hat{v}_N)\ \ \textbf{\textit{for all}}\ \ \hat{v}_N \in \hat{U}_N\,. \tag{13.2b}$$

Functions $\hat{u}_N \in \hat{U}_N$ are of the form

$$\hat{u}_N(x) = \sum_{i=1}^{N-1} u_i\,\hat{\phi}_{N,i}(x) \tag{13.3}$$

[60] In practice, $(f, \hat{v}_N)_{L^2(\Omega)}$ is approximated by quadrature.

where $\hat{\phi}_{N,i}$ denotes the i-th piecewise linear *hat function* satisfying $\hat{\phi}_{N,i}(x_j) = \delta_{ij}$. These hat functions form a basis of the linear space $\hat{U}_N$, and by $u_N = \big(u_1, \ldots, u_{N-1}\big)^{\mathrm{T}} \in \mathbb{R}^{N-1}$ we denote the coefficient vector of the representation of a function $\hat{u}_N$ in this basis. It is easy to verify that

$$a(\hat{u}_N, \hat{v}_N) \equiv (A_N\, u_N, v_N)$$

I.e., A_N from (13.1a) is also the FEM stiffness matrix in this case. Therefore, (13.2b) is equivalent to

$$(A_N\, u_N, v_N) = b(\hat{v}_N) \quad \forall\ \hat{v}_N \in \hat{U}_N \tag{13.4}$$

which in turn is equivalent to (13.1a), with $b_N = b(\hat{v}_N)$.

The exact solution u_N^* of (13.1a), with $b_N = b(\hat{v}_N)$ corresponds to the exact solution $\hat{u}_N^*$ of (13.2b). In this spirit, we identify vectors $u_N \in \mathbb{R}^{N-1}$ with piecewise linear functions $\hat{u}_N \in C[0,1]$ via the relations

$$\hat{u}_N(x_i) = u_i, \quad i = 1 \ldots N-1, \quad x_i = i\,h_N$$

In the sequel, we frequently refer to this isomorphism,

$$u_N \in \mathbb{R}^{N-1} \quad \leftrightsquigarrow \quad \hat{u}_N \in \hat{U}_N \subseteq C[0,1] \quad \text{(with zero boundary values)}. \tag{13.5}$$

13.2 Error smoothing

Example 13.1 We consider the boundary problem $-u''(x) = 1$ on $\Omega = (0,1)$, with Dirichlet boundary conditions $u(0) = u(1) = 0$. The exact solution is[61] $u_*(x) = \frac{1}{2}x(1-x)$.

Let us denote by u_N^m the m-th iterate of the Jacobi method applied to (13.1a), with u_N^* = exact solution of (13.1a). Fig. 13.1 shows the convergence history ($\|u_N^m - u_N^*\|_\infty$ versus the iteration number m, for $u_N^0 = 0$), for different problem sizes N. Observe, in particular, the well-known degradation of the convergence behavior as the problem size increases. ∎

A more precise analysis of the convergence behavior of the Jacobi method applied to (13.1a) can be based on the fact that the eigenvalues and the eigenvectors of the iteration matrix are explicitly known (see Sec. 2): The vectors $w_k \in \mathbb{R}^{N-1}$, $k = 1 \ldots N-1$, with

$$(w_k)_i = \sin(ik\pi h_N) = \sin(k\pi x_i), \quad i = 1 \ldots N-1 \tag{13.6a}$$

are the eigenvectors of A_N, with corresponding eigenvalues

$$\lambda_k = \frac{4}{h_N} \sin^2\!\big(\tfrac{k\pi}{2}\, h_N\big), \quad k = 1 \ldots N-1 \tag{13.6b}$$

[61] For this example, the FD and FEM discretizations are equivalent, with $b_N = h_N\,(1, 1, \ldots, 1)^T$. The nodal values $u_*(x_i)$ are exactly reproduced by the discrete solution because the local discretization errors vanish.

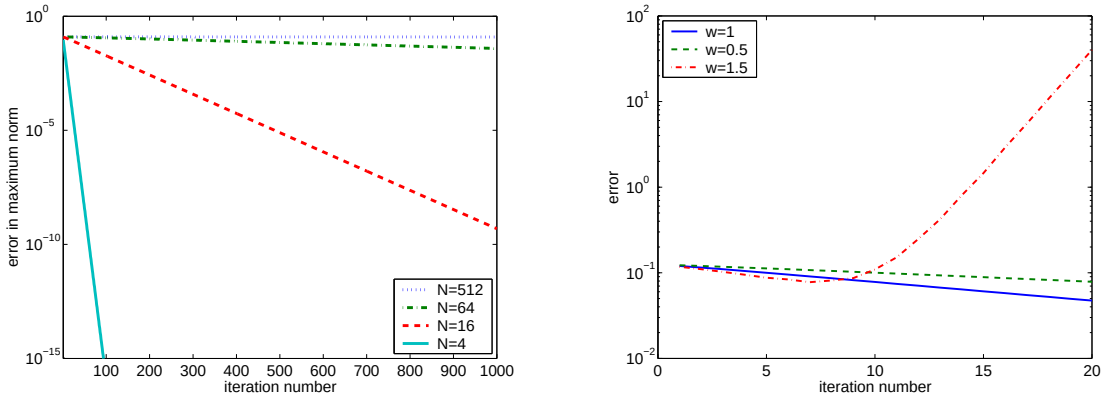


Figure 13.1: Left: Performance of the undamped Jacobi method for solving (13.1a) for different problem sizes N. Right: Varying, for fixed N, the damping parameter ω does not improve convergence.

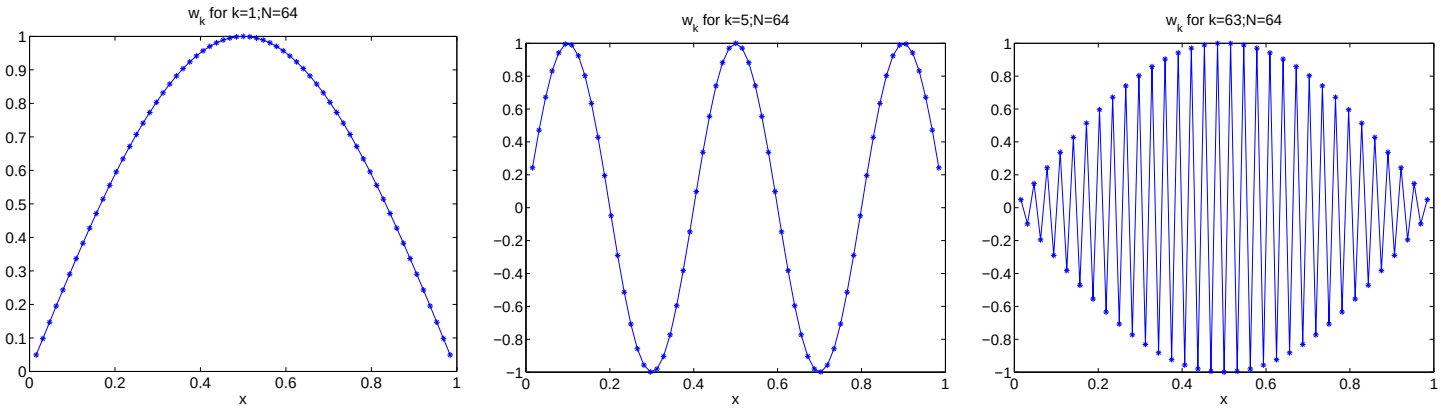


Figure 13.2: Illustration of w_1 (low frequency), w_5, and w_{N-1} (high frequency); $N = 64$.

The eigenvectors w_k can be identified with the piecewise linear functions $\hat{w}_k$ plotted in Fig. 13.2 for different values of k. Whereas the 'low frequencies' $\hat{w}_k$ for small values of k correspond to slowly ('smoothly') varying functions, the 'high frequencies' $\hat{w}_k$ corresponding to large values of k are rapidly oscillating. Note that at the higher frequencies the oscillating behavior of the analogous original eigenfunctions of $-u''$ is not correctly reproduced on the grid due to an 'aliasing effect'.

Now we study the performance of the damped Jacobi iteration, with damping factor $\omega \in (0, 2)$, for solving (13.1a):

$$u_N^{\nu+1} = u_N^\nu + \omega D_N^{-1}(b_N - A_N u_N^\nu), \quad \nu = 0 \ldots m-1 \tag{13.7}$$

starting from some initial approximation u_N^0. Here, A_N is given by (13.1a) and $D_N = (2/h_N) I$ is the diagonal of A_N.

Let $G_{N,\omega} = G_{N,\omega}^{Jac} = I - \omega D_N^{-1} A_N$ denote the iteration matrix. For all k, the vector w_k from (13.6a) is an eigenvector of the symmetric matrix $G_{N,\omega}$, with eigenvalue

$$\gamma_k(\omega) = 1 - \omega \frac{h_N}{2} \lambda_k = 1 - 2\omega \sin^2(\tfrac{k\pi}{2} h_N) \tag{13.8}$$

Example 13.2 We consider the case $N = 64$ and analyze the case where the initial error $e_N^0 \in \mathbb{R}^{N-1}$ is one of the eigenmodes, $e_N^0 = w_k$. The error after step m then satisfies (with $h_N = 1/N$)

$$\|e_N^m\|_2 = \|(G_{N,\omega})^m e_N^0\|_2 = \|\gamma_k(\omega)^m w_k\|_2 = |\gamma_k(\omega)|^m = |1 - 2\omega \sin^2(\tfrac{k\pi}{2} h_N)|^m$$

As k varies between 1 and $N-1$, the contraction factor $|\gamma_k(\omega)|$ varies between $1 - \mathcal{O}(h_N^2)$ and numbers that are very close to zero.

- For the case $\omega = 1$, the left plot in Fig. 13.3 illustrates the value $|\gamma_k(1)|$ by plotting $m \in \mathbb{N}$ over k such that $|\gamma_k(1)|^m \approx 0.01$.

- The center plot in Fig. 13.3 shows the behavior for the case $\omega = \omega_{opt} = 2/3$, which gives 'optimal damping of high frequencies' in the sense of Exercise 13.1 below. Components corresponding to high frequencies are reduced quickly, whereas the low frequency components are not significantly reduced.

- The right plot in Fig. 13.3 shows the convergence history for $\omega = \omega_{opt} = 2/3$ for a case where all modes are equally present in e_N^0. The error reduction rate is quite good at the beginning, due to the fact that the high frequency error modes are quickly damped out. Later on, as the error becomes smoother, the iteration begins to stagnate. ∎

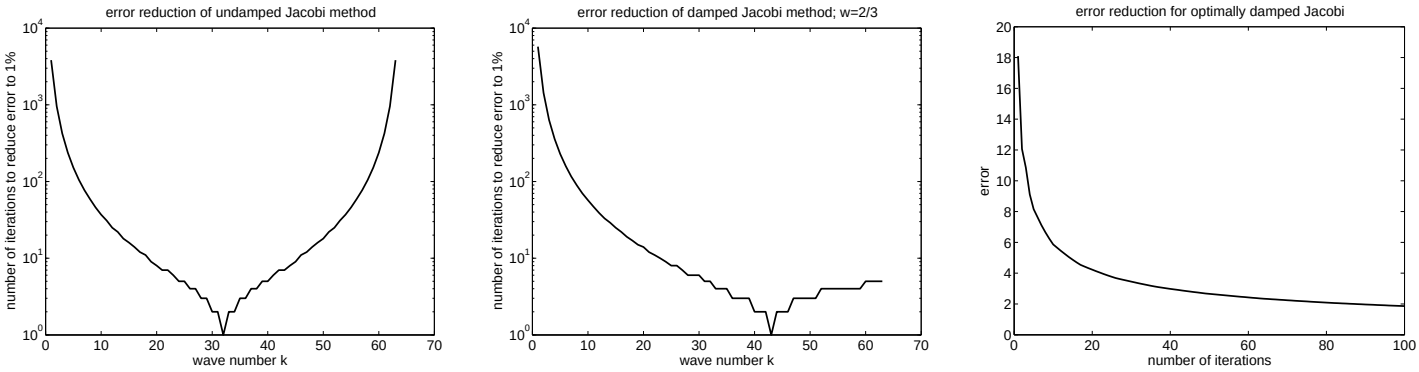


Figure 13.3: Left: Contraction property of undamped ($\omega = 1$) Jacobi method in dependence on the wave number k. Center: Contraction property of damped ($\omega = 2/3$) Jacobi method in dependence on the wave number k. Right: Convergence behavior of damped ($\omega = 2/3$) Jacobi for initial error $e_0 = \sum_{k=1}^{N-1} w_k$.

Example 13.2 shows that the error components of *some* of the 'frequencies' (or 'modes') are reduced very quickly in a few Jacobi steps, while the error components of other modes are hardly reduced. The conclusion is that we should design iterative schemes that are based on *two* principles: Use a damped Jacobi method to reduce the error of *some* modes; another procedure will have to be designed to effectively reduce the error in the remaining modes.

Accepting the fact that we cannot reduce all error components uniformly well, we settle for efficiently reducing some of them. For reasons that will become clear soon, we wish to *reduce the high frequency components*. The optimal value of the damping parameter ω then turns out to be $\omega_{opt} = 2/3$:

Exercise 13.1 Show: the minimizer ω_{opt} of the function

$$\bar{\rho}(G_{N,\omega}) = \max_{N/2 \le k \le N-1} |\gamma_k(\omega)| = \max_{N/2 \le k \le N-1} \left|1 - 2\omega \sin^2(\tfrac{k\pi}{2} h_N)\right|$$

(with $h_N = 1/N$) satisfies $\lim_{\substack{N\to\infty \\ (h_N \to 0)}} \omega_{opt} = 2/3$. Also show that $\bar{\rho}(G_{N,\omega_{opt}}) \le \frac{1}{3}$ for all $h_N > 0$. ∎

Thus we have identified the optimal damping parameter as $\omega_{opt} = 2/3$. This choice indeed leads to a quick reduction of the high frequency components corresponding to the the upper half of the discrete spectrum.

Example 13.3 We again consider the model problem (13.1a) and use the damped Jacobi method with damping parameter $\omega_{opt} = 2/3$. For $N = 32$ and a random starting vector x_0 we plot the errors $\hat{e}_N^0$, $\hat{e}_N^3$, $\hat{e}_N^6$, and $\hat{e}_N^9$ in Fig. 13.4. We observe that the initial error e_N^0, being randomly chosen, has high and low frequency components. The damped Jacobi method damps out the high frequency components quickly: the strong spikes of $\hat{e}_N^0$ are no longer present in $\hat{e}_N^3$ and the error $\hat{e}_N^3$ (and even more so $\hat{e}_N^6$) is rather slowly varying. Iterating further does not result in a significant error reduction because the contraction rate of the damped Jacobi method is close to 1 for the low frequency components. ∎

Another reasonable choice is, e.g., $\omega = 1/2$. In this case the 'medium frequencies' are damped more slowly than for $\omega = 2/3$; the overall damping factor for the upper half of the spectrum is only $\frac{1}{2}$ instead of $\frac{1}{3}$. On the other hand, the highest frequencies are those for which the damping effect is most significant (also note that all eigenvalues of $G_{N,\frac{1}{2}}$ are positive). This is illustrated in Fig. 13.5 (left = low frequencies; right = high frequencies).

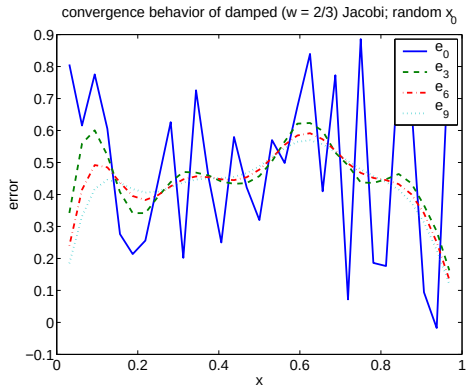


Figure 13.4: Error reduction of optimally damped Jacobi method with random initial vector ($N = 32$).

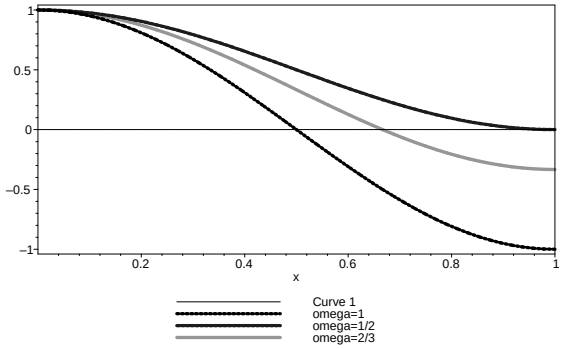


Figure 13.5: Eigenvalue portrait for $G_{N,\omega}$, with three different values of ω.

13.3 The two-grid scheme (TG) in abstract formulation

We have seen in Example 13.3 that the Jacobi method with damping parameter $\omega_{opt} = 2/3$ is quite effective at reducing high frequency error modes. After a few steps, the error e_N^m resp. $\hat{e}_N^m$ has become smooth in the sense that, while possibly being large, it is not strongly varying. Thus, we may hope that we could construct a good approximation of $\hat{e}_N^m$ on a *coarser mesh*, e.g. with mesh size $2\,h_N$. This is the idea of the two-grid method, which consists of two steps:

1. ***Smoothing step:*** Perform m steps of the damped Jacobi method for $A_N\,u_N = b_N$ to yield u_N^m.
2. ***Coarse grid correction (CGC):*** Solve a related problem of size $n < N$ (typically, $n = {}^N\!/_2$ in the 1D case) whose solution approximates the error $e_N^m = u_N^m - u_N^*$. Then correct u_N^m. The correction will stem from solving a problem that is posed on a coarser grid with mesh size $2\,h_N$.

Let us first describe the coarse grid correction step in a formal, algebraic way. The correction step is simply one step (perhaps more of them) of a linear iterative scheme, where the original matrix A_N is approximated by its 'coarse counterpart' A_n. In addition, we need a *restriction operator* $R_{n,N}$ and *a prolongation (interpolation) operator* $P_{N,n}$ acting between the solution spaces of dimension (essentially) N and n. With an appropriate choice for $R_{n,N}$ and $P_{N,n}$, the coarse grid correction step is realized in the usual way in terms of correction by a linear image of the residual of the smoothed-out approximation u_N^m, similarly as in stationary iterative schemes:

$$u_N^m \;\mapsto\; u_N^{TG} = u_N^m + P_{N,n}\,A_n^{-1}\,R_{n,N}\,(b_N - A_N\,u_N^m)$$

i.e., $P_{N,n}\,A_n^{-1}\,R_{n,N}$ is used to approximate A_N^{-1}. This may be called the 'strong form' of the correction step and will be appropriate in the setting of FD methods. (Within the FEM context we will use an appropriate 'weak' formulation.)

Algorithmically, a CGC step amounts to:

1. Compute the residual $r_N^m = b_N - A_N u_N^m$
2. Restrict the residual to the 'coarser space': $r_n^m = R_{n,N} r_N^m$
3. Solve the correction equation $A_n \delta_n = r_n^m$
4. Prolongate the correction δ_n and add it to u_N^m, i.e., compute $u_N^{TG} = u_N^m + P_{N,n}\delta_n$

The *coarse grid correction* $P_{N,n}\delta_n$ is an approximation for the error $-e_N^m = u_N^* - u_N^m$.

Note that the approximation $P_{N,n} A_n^{-1} R_{n,N}$ for A_N^{-1} has reduced rank. Therefore the corresponding error amplification matrix

$$G_N^{CGC} = I_N - P_{N,n} A_n^{-1} R_{n,N} A_N$$

cannot be a contraction, because $G_N^{CGC} v_N = v_N$ for $A_N v_N \in \ker(R_{n,N})$, which typically consists of 'unsmooth' objects. This shows that coarse-grid correction only makes sense in combination with an appropriate smoothing procedure.

Additional smoothing steps following the coarse-grid correction are a further option.

Galerkin approximations on subspaces.

MG methods can be applied in the context of any discretization approach, e.g., finite difference (FD) methods. In view of application to systems arising from a FEM discretization, a 'weak' formulation of coarse-grid correction is appropriate, involving Galerkin orthogonality analogously as in (13.2b).

Let us first describe, in a general, abstract form, the Galerkin approximation of the solution $u_N^* \in \mathbb{R}^{N-1}$ of a linear system

$$A_N u_N = b_N, \quad A_N \in \mathbb{R}^{(N-1)\times(N-1)} \tag{13.9a}$$

in a smaller subspace of $\mathbb{R}^{N-1}$.

Due to finite dimension, (13.9a) is equivalent to the following weak formulation, with $V_N := R^{N-1}$:

$$\textbf{\textit{Find}}\ u_N \in V_N\ \textbf{\textit{such that}}\ \ (A_N u_N, w_N) = (b_N, w_N)\ \ \textbf{\textit{for all}}\ \ w_N \in V_N\,. \tag{13.9b}$$

Let $V_n \subseteq V_N$ be a subspace of dimension $n-1 < N-1$. The corresponding *Galerkin approximation* in V_n is defined by

$$\textbf{\textit{Find}}\ v_n \in V_n\ \textbf{\textit{such that}}\ \ (A_N v_n, w_n) = (b_N, w_n)\ \ \textbf{\textit{for all}}\ \ w_n \in V_n\,. \tag{13.10a}$$

In order to reformulate (13.10a) again as a system of linear algebraic equations, let $\{p_n^1, \ldots, p_n^{n-1}\}$ be a basis of V_n. Let $P_{N,n} = \left(p_n^1 \middle| \ldots \middle| p_n^{n-1}\right) \in \mathbb{R}^{(N-1)\times(n-1)}$. Seeking v_n in the form $v_n = P_{N,n} y_n$ with coefficient vector $y_n \in \mathbb{R}^{n-1}$ allows us to rewrite (13.10a) as

$$\textit{Find}\ \ y_n \in \mathbb{R}^{n-1}\ \ \textit{such that}\ \ (A_N P_{N,n} y_n, P_{N,n} z_n) = (b_N, P_{N,n} z_n) \quad \forall\ z_n \in \mathbb{R}^{n-1}\,, \tag{13.10b}$$

which in turn is equivalent to the algebraic system

$$\textit{Compute the solution}\ \ y_n \in \mathbb{R}^{n-1}\ \ \textit{of}\ \ P_{N,n}^{\mathrm{T}} A_N P_{N,n} y_n = P_{N,n}^{\mathrm{T}} b_N\,. \tag{13.10c}$$

Here,

- the matrix $P_{N,n} \in \mathbb{R}^{(N-1)\times(n-1)}$ is called the ***prolongation matrix***, and
- its transpose $R_{n,N} := P_{N,n}^{\mathrm{T}} \in \mathbb{R}^{(n-1)\times(N-1)}$ plays the role of a ***restriction matrix***.

Remark 13.1 The Galerkin subspace approximation (13.10) is analogous to the Galerkin approach underlying Krylov methods of FOM type like CG. Indeed, by rearranging terms in (13.10a), we see that this is equivalent to the *Galerkin residual orthogonality ('OrthoRes') requirement*

$$\textbf{\textit{Find}}\ v_n \in V_n\ \ \textbf{\textit{such that}}\ \ r_n = b_N - A_N v_n \perp V_n\,.$$

Recognizing this as the defining condition in (10.2a), we can conclude as in the proof of (10.2c) that, for SPD matrices A_N, the Galerkin solution v_n satisfies the *best approximation property*

$$\|v_n - u_N^*\|_{A_N} = \min_{w_n \in V_n} \|w_n - u_N^*\|_{A_N} \tag{13.11}$$

in the energy norm $\|\cdot\|_{A_N}$. ∎

Galerkin coarse grid approximation.

The resulting Galerkin form of the coarse grid correction (CGC) for a step of the abstract two-grid method amounts to the following procedure. After having performed m steps of the damped Jacobi iteration for $A_N u_N = b_N$ yielding u_N^m with residual $r_N^m = b_N - A_N u_N^m$, the error $e_N^m = u_N^m - u_N^*$ satisfies

$$A_N(-e_N^m) = r_N^m \tag{13.12}$$

and $-e_N^m$ would be the 'exact correction' of u_N^m since $u_N^* = u_N^m - e_N^m$. Now, as before we consider the weak form of the error equation (13.12) and approximate $-e_N^m$ by its Galerkin approximation $P_{N,n}\delta_n \in V_n$ in an analogous way as in (13.10), but now with right-hand side r_N^m. The Galerkin-type coarse grid correction is computed in the following way. We consider the approximation to A_N as in (13.10c),

$$A_n^{Galerkin} = P_{N,n}^{\mathrm{T}} A_N P_{N,n} \in \mathbb{R}^{(n-1)\times(n-1)} \tag{13.13}$$

defined via an appropriately chosen *Galerkin pair* $P_{N,n}$ (prolongation) and $R_{n,N} = P_{N,n}^{\mathrm{T}}$ (restriction), and proceed as analogously as before:

1. Compute the residual $r_N^m = b_N - A_N u_N^m$
2. Restrict the residual to the subspace V_n: $r_n^m = P_{N,n}^{\mathrm{T}} r_N^m$
3. Solve the correction equation $A_n^{Galerkin}\delta_n = r_n^m$
4. Prolongate the correction δ_n and add it to u_N^m, i.e., compute $u_N^{TG} = u_N^m + P_{N,n}\delta_n$

$P_{N,n}\delta_n$ is the coarse grid correction to u_N^m.

For the case of an SPD matrix A_N, analogously to (13.11) we have

$$\|u_N^{TG} - u_N^*\|_{A_N} = \|P_{N,n}\delta_n - (-e_N^m)\|_{A_N} = \min_{w_n \in V_n} \|w_n - (-e_N^m)\|_{A_N} \tag{13.14}$$

The error $P_{N,n}\delta_n - (-e_N^m)$ is orthogonal to V_n with respect to $(\cdot,\cdot)_{A_N}$, the coarse grid correction $P_{N,n}\delta_n$ is the projection of $-e_N^m$ onto V_n with respect to $(\cdot,\cdot)_{A_N}$, and $P_{N,n}A_n^{-1}P_{N,n}^{\mathrm{T}}A_N$ is the corresponding A_N-selfadjoint projector:

$$P_{N,n}A_n^{-1}P_{N,n}^{\mathrm{T}}A_N(-e_N^m) = P_{N,n}A_n^{-1}P_{N,n}^{\mathrm{T}}r_N^m = P_{N,n}\delta_n$$

Exercise 13.2 Check that the two-grid method is a linear iteration. In particular: If $S_N = G_{N,\omega}^{Jac} = I - \omega D_N^{-1}A_N$ is the iteration matrix of the damped Jacobi method (or some other linear smoothing procedure), then the iteration matrix G_N^{TG} of the two-grid with m smoothing steps is given by

$$G_N^{TG} = (I_N - P_{N,n}A_n^{-1}P_{N,n}^{\mathrm{T}}A_N)\,S_N^m \tag{13.15a}$$

with $A_n = A_n^{Galerkin}$ from (13.13). Conclude that (in any norm)

$$\|G_N^{TG}\| \le \|A_N^{-1} - P_{N,n}A_n^{-1}P_{N,n}^{\mathrm{T}}\|\,\|A_N S_N^m\| \tag{13.15b}$$

∎

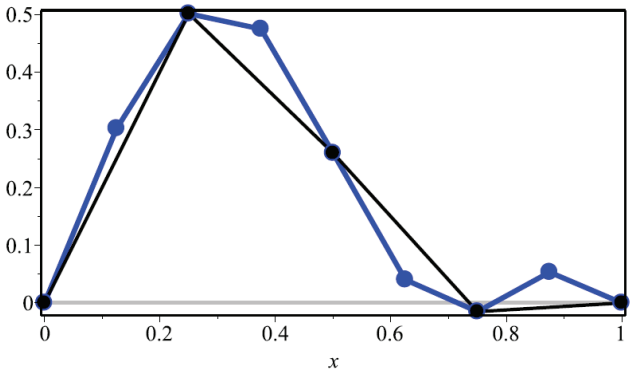


Figure 13.6: Piecewise linear functions in $\hat{U}_N$ and $\hat{U}_{N/2}$.

13.4 The two-grid method for the 1D model problem

We return to the design of the coarse grid correction for the 1D Poisson model problem. Our first goal is to choose an appropriate subspace V_n together with the appropriate prolongation and restriction operators.

In the FD context, all relevant objects (vectors) are to be identified with grid functions. In the FEM context, it is useful to think in terms of functions $\hat{u}_N$ and identify them with coefficient vectors u_N in a canonical way, in the spirit of (13.5). To this end we define the space of piecewise affine functions $\hat{U}_N$ on the given (equidistant) mesh by

$$\hat{U}_N = \{\hat{u}_N \in C[0,1]\colon\ \hat{u}_N|_{\mathcal{I}_i} \in \mathcal{P}_1 \ \text{ for } \ i = 0 \ldots N-1\}$$

with the subintervals

$$\mathcal{I}_i = (x_i, x_{i+1}), \quad i = 0 \ldots N-1, \qquad x_i = i\,h_N, \quad h_N = \frac{1}{N}$$

Note that the isomorphism (13.5), $u_N \in \mathbb{R}^{N-1} \ \leftrightsquigarrow\ \hat{u}_N \in \hat{U}_N$ maps $u_N \in \mathbb{R}^{N-1}$ to an element of $\hat{u}_N \in \hat{U}_N$.

We choose $n = N/2$ and consider[62] the spaces $\hat{U}_N$ and $\hat{U}_{N/2}$. Clearly, $\hat{U}_{N/2} \subseteq \hat{U}_N$, with the natural injection given by the prolongation operator $\hat{I}_{N,N/2}\colon \hat{U}_{N/2} \to \hat{U}_N$. In the two-grid method, this plays the role of the prolongation $P_{N,N/2}$.

Exercise 13.3 Define the matrix $I_{N,N/2}$ by

$$I_{N,N/2} = \begin{pmatrix} \frac{1}{2} & & & & & \\ 1 & 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ 0 & 1 & & & & \\ & \frac{1}{2} & \frac{1}{2} & & & \\ & 0 & 1 & 0 & & \\ & & \frac{1}{2} & \frac{1}{2} & & \\ & & & \ddots & \ddots & \\ & & & & & 1 \\ & & & & & \frac{1}{2} \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N/2-1)} \tag{13.16}$$

Let $\hat{u}_{N/2} \in \hat{U}_{N/2} \subseteq \hat{U}_N$ be associated with the coefficient vector $u_{N/2} \in \mathbb{R}^{N/2-1}$.

[62] For simplicity, we assume that N is even – in fact, later we will assume that $N = 2^L$ for some $L \in \mathbb{N}$.

a) Show: The vector $u_N \in \mathbb{R}^{N-1}$ corresponding to $\hat{u}_{N/2}$ (viewed as an element of $\hat{U}_N$!) is given by $u_N = I_{N,N/2}\, u_{N/2}$. This corresponds to linear interpolation (sketch).

b) Give an interpretation of the corresponding restriction operator $R_{N/2,N}\colon \hat{U}_N \to \hat{U}_{N/2}$ represented by the matrix

$$I_{N,N/2}^{\mathrm{T}} \in \mathbb{R}^{(N/2-1)\times(N-1)} \tag{13.17}$$

Note that this restriction is not the trivial, pointwise one; rather, it involves local weighting.

Remark: The row sums in $I_{N,N/2}^{\mathrm{T}}$ are 2, not 1. If we scale the inner products on the two levels in a more natural way by a factor h and $2h$, respectively, then the corresponding adjoint is represented by $\frac{1}{2} I_{N,N/2}^{\mathrm{T}}$.

c) A trivial restriction $\hat{J}_{N/2,N}\colon \hat{U}_N \to \hat{U}_{N/2}$ is represented by the matrix

$$J_{N/2,N} = \begin{pmatrix} 0 & 1 & 0 & & & & & & \\ & & 0 & 1 & 0 & & & & \\ & & & & 0 & 1 & 0 & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & 0 & 1 & 0 \end{pmatrix} \in \mathbb{R}^{(N/2-1)\times(N-1)} \tag{13.18}$$

Provide the geometrical interpretation of the trivial restriction operator $\hat{J}_{N/2,N}$ and verify that $J_{N/2,N}$ is the left-inverse of $I_{N,N/2}$.

d) Show that the identities

$$I_{N,N/2}^{\mathrm{T}} A_N = A_{N/2} J_{N/2,N} \tag{13.19a}$$

and

$$A_{N/2} = A_{N/2}^{Galerkin} := I_{N,N/2}^{\mathrm{T}} A_N I_{N,N/2} \tag{13.19b}$$

are valid.

e) Show that the A_N-conjugate projector onto the range of the prolongation matrix $I_{N,N/2}$ is given by

$$I_{N,N/2} A_{N/2}^{-1} I_{N,N/2}^{\mathrm{T}} A_N = I_{N,N/2} J_{N/2,N} : \quad \mathbb{R}^{N-1} \to \mathbb{R}^{N-1} \tag{13.20}$$

■

Remark 13.2 Identity (13.19b) is specific to the 1D model problem, and it simplifies the analysis of the two-grid method in this case. In general, such an identity does not hold. ■

In matrix notation, the natural injection $I_{N,N/2}$ allows us identify a subspace of $\mathbb{R}^{N-1}$ suitable for the coarse grid correction, namely the space $V_{N/2} = I_{N,N/2}\, \mathbb{R}^{N/2-1}$ of dimension $N/2 - 1$. The prolongation operator $\hat{I}_{N,N/2}$ represented by the matrix $I_{N,N/2}$ plays the role of the prolongation $P_{N,n}$ from Sec. 13.3.

We are now ready to turn to the coarse-grid Galerkin approximation of the error $e_N^m = u_N^m - u_N^*$ for a given u_N^m (obtained after m smoothing steps), and formulate it in terms of the algebraic objects (matrices and coefficient vectors) involved in the numerical computation. The error satisfies $A_N(-e_N^m) = r_N^m = b_N - A_N u_N^m$. The Galerkin correction technique described above, with the subspace $V_{N/2} = I_{N,N/2}\mathbb{R}^{N/2-1}$, gives the correction $\delta_{N/2}$ on the coarse grid as the solution of (see (13.19b))

$$A_{N/2}^{Galerkin}\, \delta_{N/2} = I_{N,N/2}^{\mathrm{T}} r_N^m \tag{13.21}$$

with $I_{N,N/2}^{\mathrm{T}}$ from (13.17). The matrix $A_{N/2} = A_{N/2}^{Galerkin}$ is a projected version of A_N, similarly as the matrix $H_m = V_m^{\mathrm{T}} A V_m$ in the context of Krylov methods (cf. (9.9)).

The outcome is the two-grid algorithm (Alg. 13.1), with the canonical choice for the Galerkin pair, $I_{N,N/2}$ (prolongation) and $I_{N,N/2}^{\mathrm{T}}$ (restriction).

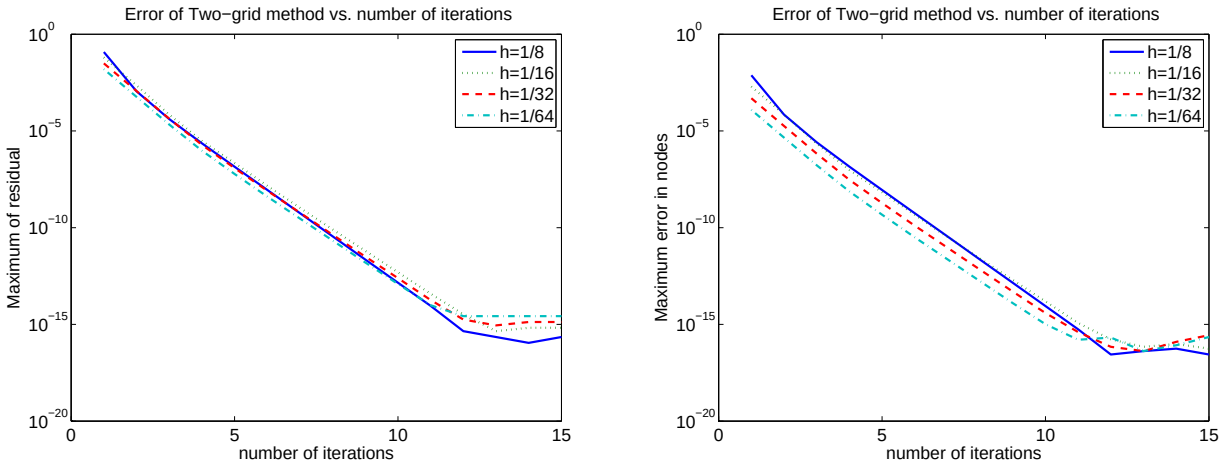


Figure 13.7: Performance of the two-grid method for the 1D Poisson model problem.

Algorithm 13.1 Two-grid method

1: Choose initial guess $u_N^{TG,0}$
2: **for** $j = 1, 2, \ldots$ until convergence **do**
3: *Smoothing:* do m steps of the damped Jacobi method starting from $u_N^{TG,j-1}$ to obtain u_N^m
4: Compute and restrict the residual: $I_{N,N/2}^{\mathrm{T}} r_N^m = I_{N,N/2}^{\mathrm{T}} (b_N - A_N u_N^m)$
5: *Coarse grid correction:* solve (13.21) for $\delta_{N/2}$
6: *Prolongate and apply the correction:* $u_N^{TG,j} = u_N^m + I_{N,N/2}\, \delta_{N/2}$
7: **end for**

Example 13.4 For the 1D model problem, the two-grid method has very good convergence properties as is visible in Fig. 13.7. We note in particular that its performance (measured in terms of the amount of error reduction per iteration) is *independent of the problem size.* ∎

13.5 Two grid convergence analysis for the 1D model problem

In this section we derive a bound for the spectral norm of the two-grid error amplification matrix (see (13.15a))

$$G_N^{TG} = (I_N - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{\mathrm{T}}\, A_N)\, S_N^m \tag{13.22a}$$

with $A_{N/2} = A_{N/2}^{Galerkin}$ from (13.19b). This analysis is based on separately estimating the factors in (see (13.15b))

$$\|G_N^{TG}\| \le \|A_N^{-1} - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{\mathrm{T}}\|\, \|A_N\, S_N^m\| \tag{13.22b}$$

The following analysis applies in the FEM as well as in the FD context, because the matrices involved are the same.

Smoothing property.

We consider a smoothing procedure based on damped Jacobi iteration with damping parameter $\omega = \frac{1}{2}$ (see Fig 13.5). We have

$$A_N\, S_N^m = A_N \big(I - \omega D_N^{-1} A_N\big)^m = A_N \big(I - \tfrac{1}{2}\tfrac{h}{2} A_N\big)^m \tag{13.23a}$$

with eigenvalues (cf. (13.8))

$$\mu_k = \lambda_k\big(1 - \tfrac{1}{2}\tfrac{h_N}{2}\lambda_k\big)^m, \qquad \lambda_k = \lambda_k(A) = \tfrac{4}{h_N}\sin^2(\tfrac{k\pi}{2}h_N), \quad k = 1\ldots N-1, \qquad h_N = \frac{1}{N}$$

Thus,

$$\mu_k = \frac{1}{h_N}\Big(4\sin^2(\tfrac{k\pi}{2}h_N)\big(1 - \tfrac{1}{4}4\sin^2(\tfrac{k\pi}{2}h_N)\big)^m\Big), \quad k = 1\ldots N-1 \tag{13.23b}$$

The eigenvalues of S^m are contained in $(0,1)$ (see (13.8)). Application of the difference operator A has a strong effect on the small eigenvalues associated with high oscillatory eigenmodes, and a small effect on the larger eigenvalues associated with smoother eigenmodes. The overall behavior is described in the following theorem.

Theorem 13.1 *For $m \geq 1$, the eigenvalues μ_k of $A_N S_N^m$ satisfy the uniform bound*

$$|\mu_k| = \mu_k \leq \frac{1}{h_N}\frac{8}{5m+3}, \quad k = 1\ldots N-1 \tag{13.24}$$

Proof: Each of the μ_k given in (13.23b) is of the form

$$\mu_k = \frac{4}{h_N}\big(\theta_k(1-\theta_k)^m\big) \quad \text{with} \quad \theta_k = \sin^2(\tfrac{k\pi}{2}h_N) \in [0,1] \tag{13.25}$$

Thus, estimating the μ_k reduces to an upper estimate for the values of the function $\varphi(\theta) = \theta(1-\theta)^m$ for $\theta \in [0,1]$. Elementary analysis shows that the maximum of $\varphi(\theta)$ is attained at $\theta_{max} = 1/(m+1)$, with

$$\varphi(\theta_{max}) = \frac{1}{m+1}\Big(1 - \frac{1}{m+1}\Big)^m = \frac{m^m}{(m+1)^{m+1}}$$

We estimate the denominator from below using three leading terms from its binomial expansion:

$$(m+1)^{m+1} = \sum_{k=0}^{m+1}\binom{m+1}{k}m^k \geq m^{m+1} + (m+1)m^m + \frac{m(m+1)}{2}m^{m-1} = \frac{m^m}{2}(5m+3)$$

This gives

$$\varphi(\theta_{max}) \leq \frac{2}{5m+3}$$

Together with (13.25) this results in (13.24). □

Since A is selfadjoint with respect to $\|\cdot\|_2$ as well as $\|\cdot\|_A$, we conclude the ***smoothing property*** in the following form:

Corollary 13.1 *For $A_N S_N^m$ from* (13.23a)*, the bounds*

$$\|A_N S_N^m\|_2 \leq \frac{1}{h_N}\frac{8}{5m+3}, \quad m \geq 1; \qquad h_N = \frac{1}{N} \tag{13.26}$$

are valid.

Approximation property.

In view of (13.22b), a bound for

$$\|A_N^{-1} - I_{N,N/2}A_{N/2}^{-1}I_{N,N/2}^{\mathrm{T}}\|$$

remains to be determined. We use the denotation and the results from Exercise 13.3 to rewrite this in the form

$$\begin{aligned} A_N^{-1} - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{\mathrm{T}} &= \left(I_{N,N} - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{\mathrm{T}}\, A_N\right) A_N^{-1} \\ &= \left(I_{N,N} - I_{N,N/2}\, J_{N/2,N}\right) A_N^{-1} \end{aligned} \tag{13.27}$$

We also recall from Exercise 13.3 that

$$I_{N,N/2}\, J_{N/2,N} = \begin{pmatrix} \frac{1}{2} & & & & & & \\ 1 & & & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & & \\ & 1 & & & & & \\ & \frac{1}{2} & \frac{1}{2} & & & & \\ & & 1 & & & & \\ & & \frac{1}{2} & \ddots & & & \\ & & & \ddots & \ddots & \frac{1}{2} & \\ & & & & \ddots & 1 & \\ & & & & & \frac{1}{2} & \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}$$

(*columns with odd index are zero*) is the A_N-conjugate projector onto the range of the prolongation matrix $I_{N,N/2}$ (along the kernel of the restriction matrix $I_{N,N/2}^{\mathrm{T}}$). Thus,

$$I_{N,N} - I_{N,N/2}\, J_{N/2,N} = \begin{pmatrix} 1 & -\frac{1}{2} & & & & \\ & & & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & & \\ & & & & & \\ & & -\frac{1}{2} & 1 & -\frac{1}{2} & \\ & & & \ddots & \ddots & \\ & & & & -\frac{1}{2} & 1 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}$$

(*rows with even index are zero*) is the A_N-conjugate projector onto the kernel of the restriction matrix $I_{N,N/2}^{\mathrm{T}}$ (along the range of the prolongation matrix $I_{N,N/2}$).

The norm of $I_{N,N} - I_{N,N/2}\, J_{N/2,N}$ is $\mathcal{O}(1)$. However, in (13.27) it is applied to A_N^{-1} which is the discrete analog of the Green's function of the original problem and has a smoothing behavior, and $I_{N,N} - I_{N,N/2}\, J_{N/2,N}$ 'behaves small' on smooth objects.

In particular, each second (non-zero) line of $I_{N,N} - I_{N,N/2}\, J_{N/2,N}$ is identical with the corresponding row of A_N, up to a scalar factor. In fact, we have

$$I_{N,N} - I_{N,N/2}\, J_{N/2,N} = E \cdot \frac{h_N}{2} A_N \quad \Rightarrow \quad \left(I_{N,N} - I_{N,N/2}\, J_{N/2,N}\right) A_N^{-1} = \frac{h_N}{2} E$$

with $E = \mathrm{diag}(1, 0, 1, 0, \ldots, 1, 0, 1)$. Together with $\|E\|_2 = 1$, these considerations this leads us to the ***approximation property***:

Theorem 13.2 *For $A_N^{-1} - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{T}$ the estimate*

$$\|A_N^{-1} - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^{T}\|_2 = \|\left(I_{N,N} - I_{N,N/2}\, J_{N/2,N}\right) A_N^{-1}\|_2 \le \frac{h_N}{2} \tag{13.28}$$

is valid.

Together with the smoothing property (Corollary 13.1) we thus obtain a convergence result for the 2-grid method:

Theorem 13.3 *The spectral norm of the error amplification matrix* $G_N^{TG} = (I_N - I_{N,N/2}\, A_{N/2}^{-1}\, I_{N,N/2}^T\, A_N)\, S_N^m$ *of the two-grid method satisfies*

$$\|G_N^{TG}\|_2 \le \frac{4}{5m+3}\,, \qquad m \ge 1 \tag{13.29}$$

independent (!) of the dimension N.

Remark 13.3

- Together with Theorem 12.1 we see that TG can play the role of an optimal preconditioner for a Krylov subspace method.
- A similar analysis applies, with respect to the energy norm, for the case where m smoothing steps are applied before and after the two-grid correction (pre- and post-smoothing).
- In general, relation $A_{N/2} = A_{N/2}^{Galerkin}$ is not satisfied. Then the analysis of the approximation property is more involved. ∎

13.6 TG in the FEM context: Formulation in terms of continuous functions

In computational practice, the coarse grid correction is an algebraic process, as in Alg. 13.1. However, for a better theoretical understanding, in particular in the context of FEM discretizations, see Appendix A, it is useful also to reformulate and view TG in terms of continuous functions.

Modified notation. Here and in the following section, instead of a dimension index we prefer to use the mesh size parameters h (fine grid) and H (coarse grid) as indices, as in Sec. A.5 ff. The nodal basis functions are now denoted by

$$\hat{v}_{h,i}, \quad i = 1 \ldots N = \dim(\hat{U}_h)$$

For each function $\hat{u}_h \in \hat{U}_h$ we denote by u_h the associated vector of coefficients in its nodal basis representation. For a family of quasi-uniform triangulations it can be shown that this natural isomorphism $u_h \leftrightsquigarrow \hat{u}_h$ is uniformly continuous, i.e., for the standard 2-norm on $\mathbb{R}^N$ we have[63]

$$c\,h\,\|u_h\|_2 \le \|\hat{u}_h\|_{L^2(\Omega)} \le C\,h\,\|u_h\|_2 \tag{13.30}$$

with universal constants c and C such that C/c is of moderate size.

Also in the sequel, 'uniformly' means that the corresponding assertion is valid for all objects in a family of quasi-uniform triangulations.

Consider u_h^m obtained after m smoothing steps and its associated function $\hat{u}_h^m \in \hat{U}_h$, with error $\hat{e}_h^m = \hat{u}_h^m - \hat{u}_h^*$, where $\hat{u}_h^*$ is the exact (discrete) solution at level h. A single step of the two-grid method seeks an approximation $\hat{u}_h^{TG}$ to $\hat{u}_h^*$ of the form

$$\hat{u}_h^{TG} = \hat{u}_h^m + \hat{\delta}_H$$

with $\hat{\delta}_H \in \hat{U}_H \subseteq \hat{U}_h$. It is natural to aim for $\hat{\delta}_H$ to be the best approximation to the error $-\hat{e}_H^m$, i.e., we seek $\hat{\delta}_H$ such that the error in the energy norm is minimized. The minimizer $\hat{\delta}_H$ is characterized by the Galerkin orthogonality property

[63] (13.30) is valid for the 2D case. Note that we here are assuming that $\|\cdot\|_2$ is the standard norm on $\mathbb{R}^N$, i.e., corresponding to the inner product $(u_h, v_h)_2 = \sum_i u_{h,i}\, v_{h,i}$. An alternative would be to scale this inner product by a factor h^2 such that it becomes an direct analogue of $(\cdot,\cdot)_{L^2(\Omega)}$, but we do not use such a scaling here.

$$a\big(\hat{u}_h^{TG} - \hat{u}_h^*, \hat{w}_H\big) = a\big((\hat{u}_h^m + \hat{\delta}_H) - \hat{u}_h^*, \hat{w}_H\big) = 0 \quad \forall\ \hat{w}_H \in \hat{U}_H$$

Rearranging terms results in the equivalent formulation

$$\textit{Find } \hat{\delta}_H \in \hat{U}_H \textit{ such that } a(\hat{\delta}_H, \hat{w}_H) = a(-\hat{e}_h^m, \hat{w}_H) \quad \forall\ \hat{w}_H \in \hat{U}_H\,. \tag{13.31a}$$

From (13.2b) we see that (13.31a) can be rewritten as

$$\textit{Find } \hat{\delta}_H \in \hat{U}_H \textit{ such that } a(\hat{\delta}_H, \hat{w}_H) = b(\hat{w}_H) - a(\hat{u}_h^m, \hat{w}_H) \ \forall\ \hat{w}_H \in \hat{U}_H\,. \tag{13.31b}$$

Here the right-hand side is the residual of $\hat{u}_h^m$ in the weak sense.

Note that the solution $\hat{\delta}_H$ of (13.31a) is the projection, with respect to the the energy product $a(\cdot,\cdot)$, of $-\hat{e}_h^m$ onto the coarse space $\hat{U}_h$: The error $\delta_H + \hat{e}_h^m$ is orthogonal to $\hat{U}_H$ with respect to $a(\cdot,\cdot)$.

The functions $\hat{\delta}_H$ and $\hat{w}_H$ are associated with coefficient vectors δ_H and w_H via the FEM isomorphism, and the canonical embedding $\hat{U}_H \subseteq \hat{U}_h$ is represented by a prolongation matrix $P_{h,H}$, the analog of $I_{N,N/2}$ above, which corresponds to piecewise linear interpolation also in higher-dimensional cases, e.g., for the 2D Poisson equation.

In matrix notation, (13.31) reads

$$\textit{Find } \delta_H \textit{ such that } (A_h\, P_{h,H}\, \delta_H, P_{h,H}\, w_H) = (r_h^m, P_{h,H}\, w_H) \quad \forall\ w_H \in \hat{U}_H\,.$$

with the residual $r_h^m = b_h - A_h\, u_h^m$; cf. (13.21).

Remark 13.4 The matrices

$$P_{h,H} \quad \text{and} \quad R_{H,h}$$

play the role of *Galerkin prolongation* and *restriction* matrices, respectively. In particular, for an SPD problem the choice $R_{H,h} = P_{h,H}^{\mathrm{T}}$ is natural because the coarse grid Galerkin approximation matrix

$$A_H = A_H^{Galerkin} = R_{H,h}\, A_h\, P_{h,H} = P_{h,H}^{\mathrm{T}}\, A_h\, P_{h,H} \tag{13.32}$$

is also SPD.

In general, $A_H^{Galerkin} = A_H$ is not necessarily true, and general multigrid techniques work with A_H as directly given by the discretization on the coarse level. Different choices for $P_{h,H}$ and $R_{H,h}$, are possible, and this choice is one of several parameters influencing the convergence behavior. (The choice of the smoothing procedure is, of course, also essential, and it is not straightforward in general.) ∎

13.7 Two grid convergence analysis for the 2D Poisson equation

The analysis from Sec. 13.5 is rather *brute force* – everything is more or less explicitly computed. Now we aim for a more general TG convergence analysis and we illustrate this for the case of the 2D Poisson equation

$$-\Delta u(x) = f(x) \ \text{ on } \Omega, \quad u = 0 \ \text{ on } \partial\Omega \tag{13.33}$$

on a convex polygonal domain Ω. We consider a FEM approximation of this problem based on a quasi-uniform triangulation $\mathcal{T}_h$ of Ω associated with an approximating FEM space $\hat{U}_h$ as described in Sec. A.5. Let $\mathcal{T}_H$, with associated space $\hat{U}_H$, be a coarser triangulation of Ω in the sense that each triangle $T_H \in \mathcal{T}_H$ is the union of several smaller triangles $T_h \in \mathcal{T}_h$.

Besides damped Jacobi we shall also consider the symmetric Gauss-Seidel scheme as a smoother.

Smoothing property.

Let us verify the smoothing property for an appropriately damped Jacobi and the symmetric Gauss-Seidel method. We consider m smoothing steps

$$u_h^{\nu+1} = u_h^\nu + W_h^{-1}(b_h - A_h u_h^\nu), \quad \nu = 0 \dots m-1$$

starting from an initial approximation u_h^0, with stiffness matrix A_h and $W_h \approx A_h$.

Damped Jacobi smoother. This has the iteration matrix $S_h = G_h^{Jac} = I - W_h^{-1}A_h$ with $W_h = \omega^{-1}D_h$. We have

$$(D_h)_{i,i} = (A_h)_{i,i} = a(\hat{v}_{h,i}, \hat{v}_{h,i}) \geq \gamma \|\hat{v}_{h,i}\|^2_{H^1(\Omega)} \geq \tilde{\gamma}$$

where[64] $\tilde{\gamma}$ depends on the ellipticity parameter γ and on the ratio C/c from (13.30). Thus, $\|D_h^{-1}\|_2 \leq \tilde{\gamma}^{-1}$. Moreover, since A_h is uniformly sparse with bounded entries (see Sec. A.5 – 2D case!), there exists a constant c_1 with $\|A_h\|_2 \leq c_1$. Therefore, for the damping parameter ω chosen such that $\omega < c_1\tilde{\gamma}^{-1}$ we have

$$\rho(W_h^{-1}A_h) \leq \|W_h^{-1}A_h\|_2 = \|\omega D_h^{-1}A_h\|_2 < 1$$

Denoting

$$B_h := W_h^{-\frac{1}{2}} A_h W_h^{-\frac{1}{2}}, \quad \text{with} \quad \sigma(B_h) \in [0,1]$$

after some rearrangement we can write

$$A_h\big(G_{h,\omega}^{Jac}\big)^m = W_h^{\frac{1}{2}} B_h (I - B_h)^m W_h^{\frac{1}{2}}$$

The simple spectral argument from the proof of Theorem 13.1 shows $\|B_h(I-B_h)^m\|_2 = \sigma(B_h(I-B_h)^m) \leq 2/(5m+3)$. In this way we obtain the smoothing property in the form

$$\|A_h S_h^m\|_2 \leq \|B_h(I-B_h)^m\|_2 \|W_h\|_2 \leq \frac{2\omega^{-1}}{5m+3}\|D_h\|_2 \leq \frac{2\omega^{-1}}{5m+3}\|A_h\|_2 \quad \text{for} \quad S_h = G_{h,\omega}^{Jac} \tag{13.34}$$

It can be seen that for general triangulations the appropriate damping parameter ω may be problematic to estimate. In contrast, for the symmetric Gauss-Seidel smoother we do not need damping to ensure error smoothing, as we show now.

Symmetric Gauss-Seidel smoother. From (5.14b), this has the iteration matrix $G_h^{SGS} = I - W_h^{-1}A_h$, where, with $A_h = L_h + D_h + L_h^{\mathrm{T}}$,

$$W_h = (L_h + D_h)\, D_h^{-1}\, (D_h + L_h^{\mathrm{T}}) = A_h + L_h D_h^{-1} L_h^{\mathrm{T}} \geq A_h$$

which due to Lemma 5.2, (ii) and (iv), also implies

$$B_h = W_h^{-\frac{1}{2}} A_h W_h^{-\frac{1}{2}} \leq I \quad \text{and} \quad \sigma(B_h) \in [0,1]$$

Furthermore, $\|W_h\|_2$ can be estimated in terms of $\|A_h\|_2$ by the following reasoning. The support of each basis function $\hat{v}_{h,i}$ has a nonempty intersection only with a small number of supports of other basis functions. For a family quasi-uniform triangulations this number is independent on the actual grid considered. This implies that there exists a moderate-sized constant C such that

$$\|L_h\|_2^2 \leq \|L_h\|_1 \|L_h\|_\infty = \Big(\max_i \sum_{j<i} |(A_h)_{i,j}|\Big)\Big(\max_i \sum_{j>i} |(A_h)_{i,j}|\Big) \leq C\Big(\max_{i,j} |(A_h)_{i,j}|\Big)^2 \leq C\,\|A_h\|_2^2$$

Thus, observing $\|D_h^{-1}\|_2 \leq c_1\tilde{\gamma}^{-1}\|A_h\|^{-1}$ we see that there exists a moderate-sized constant C with

$$\|W_h\|_2 = \|A_h + L_h D_h^{-1} L_h^{\mathrm{T}}\|_2 \leq \|A_h\|_2 + \|L_h\|_2^2 \|D_h^{-1}\|_2 \leq C\,\|A_h\|_2 \tag{13.35}$$

[64] Simplest case: For the uniform triangulation of the 2D Poisson equation on the unit square we have $\tilde{\gamma} = 4$.

Now, in an analogous way as for damped Jacobi, we obtain the smoothing property

$$\|A_h\, S_h^m\|_2 \le \frac{2\,C}{5m+3}\,\|A_h\|_2, \quad \text{with } C \text{ from (13.35)}, \quad \text{for } S_h = G_h^{SGS} \tag{13.36}$$

Approximation property.

In view of (13.36), we now aim for proving an approximation property in the form

$$\|(P_{h,H}\, A_H^{-1}\, P_{h,H}^{\mathrm{T}} - A_h^{-1})\, b_h\|_2 \le C_A\, \|A_h\|_2^{-1}\, \|b_h\|_2 \tag{13.37a}$$

with the Galerkin coarse grid approximation matrix (13.32), for arbitrary right-hand sides b_h. With the denotation $u_h = A_h^{-1}\, b_h$ and $u_H = A_H^{-1} P_{h,H}^{\mathrm{T}}\, b_h$, the desired estimate (13.37a) is equivalent to

$$\|P_{h,H}\, u_H - u_h\|_2 \le C_A\, \|A_h\|_2^{-1}\, \|b_h\|_2 \tag{13.37b}$$

To link the desired estimate to FEM approximation properties, we consider the functions $\hat{u}_h \in \hat{U}_h$, $\hat{u}_H \in \hat{U}_H$ associated with u_h, u_H. These are the solutions of

$$\begin{aligned} a(\hat{u}_h, \hat{v}_h) &= (\hat{b}_h, \hat{v}_h)_{L^2(\Omega)} \quad \text{for all } \hat{v}_h \in \hat{U}_h, \\ a(\hat{u}_H, \hat{v}_H) &= (\hat{b}_h, (P_{h,H}\, v_H)^{\wedge})_{L^2(\Omega)} = (\hat{b}_h, \hat{v}_H)_{L^2(\Omega)} \quad \text{for all } \hat{v}_H \in \hat{U}_H \end{aligned}$$

where $\hat{b}_h$ is associated with b_h. To relate $\hat{u}_h$ and $\hat{u}_H$ to each other we consider the solution u of

$$a(u, v) = (\hat{b}_h, v)_{L^2(\Omega)} \quad \text{for all } v \in H_0^1(\Omega)$$

Now we can invoke the regularity and FEM convergence results from Appendix A (see Theorem A.3, Theorem A.5 and, in particular, the refined convergence estimate (A.27)) to conclude

$$\|\hat{u}_h - u\|_{L^2(\Omega)} \le C\, h^2\, \|\hat{b}_h\|_{L^2(\Omega)}, \qquad \|\hat{u}_H - u\|_{L^2(\Omega)} \le C\, H^2\, \|\hat{b}_h\|_{L^2(\Omega)}$$

This implies

$$\|\hat{u}_H - \hat{u}_h\|_{L^2(\Omega)} \le C\, h^2\, \|\hat{b}_h\|_{L^2(\Omega)}$$

and in turn, together with (13.30),

$$\|P_{h,H}\, u_H - u_h\|_2 \le C_1\, \|b_h\|_2 = C_1\, \|A_h\|_2\, \|A_h\|_2^{-1}\, \|b_h\|_2 \tag{13.38}$$

with an appropriate constant C_1.

Furthermore, by construction (see Sec. A.5) the elements of the stiffness matrix A_h are $\mathcal{O}(1)$, and due to its sparse structure this implies the existence (uniform for $h \to 0$) of a constant C_2 with $\|A_h\|_2 \le C_2$. Combining this with (13.38) gives the desired estimate (13.37b) with $C_A = C_1\, C_2$.

Convergence of the TG method.

Combining (13.34) or (13.36), respectively, with (13.37a) we obtain an result analogous to Theorem 13.3, namely contractivity of the TG iteration matrix G_h^{TG} in the 2-norm for a sufficiently large number of smoothing sweeps.

13.8 Another look at TG for SPD systems. TG as a preconditioner for CG

In the following exercises we study a two-grid (TG) scheme with symmetric smoothing, i.e. the smoothing procedure applied after the coarse grid correction is adjoint to the initial smoother. In particular, in Exercise 13.5 we consider *TG as a preconditioner for CG*. It should be noted that the results of these exercises can be extended to general multigrid schemes of Galerkin type (see Sec. 13.9 below).

Exercise 13.4 Let $A = A_N \in \mathbb{R}^{N\times N}$ be SPD. The iteration matrix (amplification matrix) of a TG scheme with symmetric pre- and post-smoothing has the form

$$G^{TG} = (I - H^{\mathrm{T}}A)(I - CA)(I - HA) =: (I - TA) \tag{13.39}$$

with a smoother $S = I - HA$ and its adjoint $I - H^{\mathrm{T}}A$, and $C = PA_n^{-1}P^{\mathrm{T}}$ where $P \in \mathbb{R}^{N\times n}$ is a prolongation matrix and $0 < A_n = P^{\mathrm{T}}AP \in \mathbb{R}^{n\times n}$ is the Galerkin approximation of A on a coarser level, i.e., in a subspace V_n of dimension $n < N$.

(This may result from our context of 'geometric multigrid' (GMG) as introduced above, where we identify vectors with functions. However, it can also be seen in the context of 'algebraic multigrid' (AMG), which directly works with the matrix-vector formulation.)

We assume that $P \in \mathbb{R}^{N\times n}$ has full rank n, and $V_n = \mathrm{image}\,(P)$. Then, $PP^{\mathrm{T}} \in \mathbb{R}^{N\times N}$ with $\mathrm{image}(PP^{\mathrm{T}}) = V_n$. Show:

a) The A-adjoint of $(I - HA)$ is $(I - HA)^A = (I - H^{\mathrm{T}}A)$.

b) $I - CA$ is A-selfadjoint, and the same is true for $G^{TG} = I - TA$.

c) $I - CA$ is *non-expansive* (in the A-norm), i.e., $\rho(I - CA) = \|I - CA\|_A = 1$, hence $\|G^{TG}\|_A = \|I - TA\|_A < 1$, provided $\|I - HA\|_A < 1$.

(Note: $\|G^{TG}\|_A$ 'significantly < 1' requires an appropriate smoother.)

Hint: The subpace- (coarse grid-) correction is a Galerkin approximation. With the error $e = u - u_*$ for given u we have $r = b - Au = -Ae$. The Galerkin subspace (two-grid-) correction is given by $\delta = -PA_n^{-1}P^{\mathrm{T}}Ae$, and this is nothing but the A -best approximation in V_n for the 'exact correction' $-e$. Check once more this fact, i.e., check the Galerkin orthogonality relation $(\delta + e) \perp_A V_n$ by evaluating the inner product $(\delta + e, PP^{\mathrm{T}}y)_A$ for arbitrary vectors $y \in \mathbb{R}^N$. Conclude that $I - GA$ is non-expansive (Pythagoras).

d) Show that the Galerkin approximation operator

$$CA = P\,A_n^{-1}\,P^{\mathrm{T}}A$$

is the A-orthogonal projector onto V_n. (This property is equivalent to Galerkin (A-)orthogonality.) ∎

Exercise 13.5 Let A be SPD and assume that the TG amplification matrix $G^{TG} = I - TA$ from (13.39) is contractive, i.e., $\rho(g^{TG}) = \rho(I - TA) < 1$ (see Exercise 13.4).

Show: The preconditioner T defined in this way is SPD, as required for CG preconditioning.

Hint: Verify that T is symmetric and that $\rho(I - TA) = \|I - TA\|_A < 1$ holds. The desired property $T > 0$ then follows by means of a spectral argument. ∎

Remark 13.5 Algorithmically, TG as preconditioner for CG is realized as follows:

For a given iterate u, with error $e = u - u_*$ and residual $r = b - Au = -Ae$, we wish to approximate the exact correction $-e$, which is the solution $\varepsilon = -e$ of the system $A\varepsilon = r$. To this end we approximate this solution by a TG step applied to $A\varepsilon = r$ starting from $\varepsilon_0 = 0$, i.e. (in the notation from (13.39))

$$\varepsilon = \varepsilon_0 + T(r - A\varepsilon_0) = Tr \approx -e$$

This means that $T \approx A^{-1}$ exactly plays the role of the preconditioner M^{-1}, as expected. If you use MATLAB/`pcg`, specify a function `MFUN(r)` which performs such a TG step for the system $A\varepsilon = r$ starting from $\varepsilon_0 = 0$. The result is the *preconditioned residual* $\hat{r} = \varepsilon = Tr \approx -e$. ∎

13.9 Multigrid (MG)

Multigrid in 1D.

The two-grid method is mainly of theoretical interest, since implementation of Alg. 13.1 requires the exact solution of a problem of size $N/2$ in each step of the iteration. As we have discovered in Exercise 13.3, for the 1D Poisson model the Galerkin coarse grid approximation $I_{N,N/2}^{\mathrm{T}} A_N I_{N,N/2}$ is actually identical to $A_{N/2}$.

The coarse grid problem to be solved is of the same type as the original problem. This suggests to proceed in a recursive fashion: Instead of solving exactly on the coarse grid, we treat it just like the fine grid problem by performing some smoothing steps and then move to an even coarser grid. In this way we proceed further until the coarse grid problem is sufficiently small to be solved by a direct method.

Assuming (for simplicity of presentation) that the initial problem size satisfies

$$N = 2^L$$

for some $L \in \mathbb{N}$, this idea leads to Alg. 13.2, in recursive formulation, which realizes one so-called *cycle* of a basic MG algorithm. It is called with some initial guess u_0; the number of 'pre-smoothing' steps m_{pre} and optional 'post-smoothing' steps m_{post} are up to the user.

Note that 0 is the natural initial choice for the correction δ at all coarser levels.

As discussed above, in the 1D case the natural Galerkin choice for the restriction and prolongation operators is $R_{N/2,N} = I_{N,N/2}^{\mathrm{T}}, P_{N,N/2} = I_{N,N/2}$.

Algorithm 13.2 Basic MG cycle (1D)

% calling sequence: $u = MG(u_0, b, N)$; input: initial guess u_0; output: approximation u

if N is sufficiently small, compute $u = A_N^{-1} b$
else
  do $m = m_{pre}$ steps of smoothing (e.g., damped Jacobi) with initial guess u_0 to obtain u_N^m
  $r_m = b - A_N u_N^m$
  $\delta = MG(0, R_{N/2,N} r_m, N/2)$ % solve for correction recursively with initial guess 0
  $\tilde{u} = u_N^m + P_{N,N/2}\,\delta$
  do $m = m_{post}$ steps of smoothing (e.g., damped Jacobi) with initial guess $\tilde{u}$ to obtain u
end if
return u

Example 13.5 We illustrate the convergence behavior of the basic MG algorithm applied to the 1D model problem in Fig. 13.8. The plot shows the performance of the iteration $u_{i+1} = MG(u_i, b, N)$ for different values of $h = 1/N$. Here, $m_{pre} = 3$ and $m_{post} = 0$ were used. ∎

The convergence behavior of the MG method in Example 13.5 is quite satisfactory. The following exercise shows that the complexity of the algorithm is also optimal, i.e., one cycle of the MG algorithm has computational complexity $\mathcal{O}(N)$.

Exercise 13.6 Denote by $C_{MG}(N)$ the cost of one MG cycle (Alg. 13.2) called with problem size $N = 2^L$. Show: $C_{MG}(N) \le C_{MG}(N/2) + c(1 + m_{pre} + m_{post})N$ for some constant $c > 0$. Conclude that the cost of a complete cycle is $\mathcal{O}(N)$. ∎

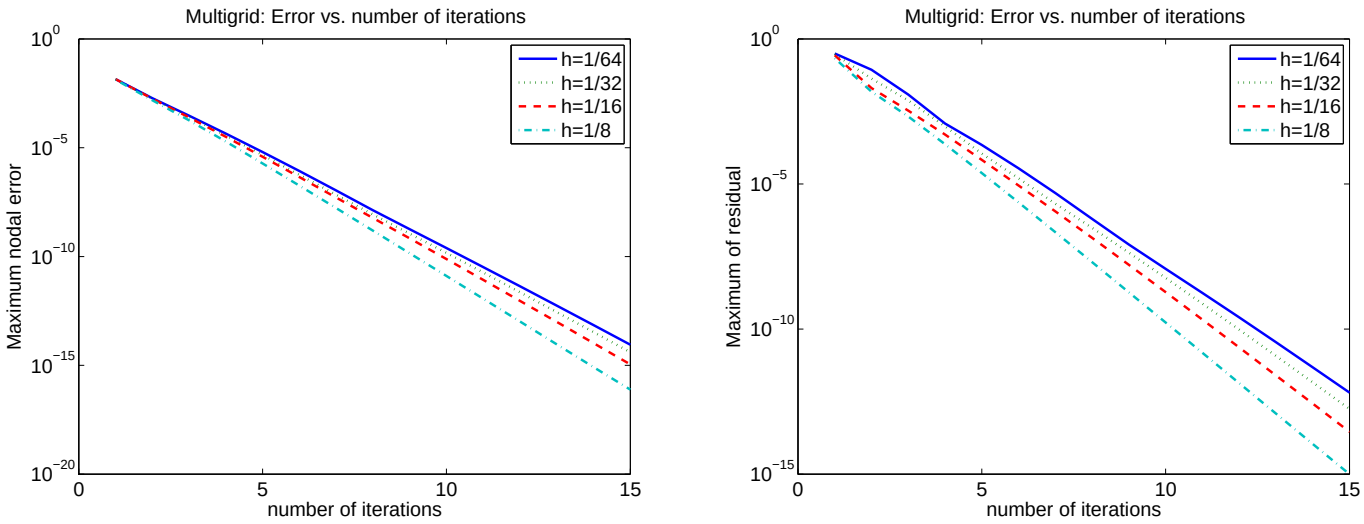


Figure 13.8: Convergence behavior of basic MG: left: $\|u_N^m - u\|_\infty$; right: $\|r_m\|_\infty$.

Multigrid in more generality.

Alg. 13.2 is formulated for the 1D model problem. A more general view, which is also applicable to problems in higher dimensions, is the following. We stick to a FEM-like terminology oriented to elliptic problems. Suppose a sequence of meshes $\mathcal{T}_l$, $l = 0, 1 \ldots$ with corresponding approximation spaces (in the standard case, spaces of piecewise linear functions). For simplicity we assume:

(i) The mesh size h_l of mesh $\mathcal{T}_l$ is $h_l \sim 2^{-l}$.

(ii) The spaces are *nested*: $\hat{U}_l \subseteq \hat{U}_{l+1}$ for $l = 0, 1 \ldots$ We write $N_l = \dim V_l$.

The spaces $\hat{U}_l$ are spanned by bases (e.g., the piecewise linear hat functions); the natural embedding $\hat{U}_l \subseteq \hat{U}_{l+1}$ then corresponds to a *prolongation operator* (matrix) $P_{l+1,l} \in \mathbb{R}^{N_{l+1} \times N_l}$. Its transpose $R_{l,l+1} = P_{l+1,l}^{\mathrm{T}}$ is the *restriction operator*. By A_l we denote the stiffness matrix arising from the underlying bilinear form $a(\cdot,\cdot)$ and the choice of basis for the space $\hat{U}_l$. In simple standard situations, $P_{l+1,l}$ can be chosen in such a way that the Galerkin identity

$$A_{l-1} = P_{l,l-1}^{\mathrm{T}} A_l P_{l,l-1}$$

remains valid. We have seen that this facilitates the convergence analysis, but the formulation of the MG algorithm does not depend on this property; it also not valid or desirable in all applications. Also a choice of the restriction operator $R_{l-1,l} \neq P_{l,l-1}^{\mathrm{T}}$ may be reasonable.

The basic MG algorithm 13.2 is reformulated as Alg. 13.3. Note that the assumption $h_l \sim 2^{-l}$ guarantees that $N_l \sim 2^{ld}$, where $d \in \mathbb{N}$ is the spatial dimension.

Alg. 13.3 is called the *V-cycle*; see Fig. 13.9. Instead of solving exactly on a coarser level, which corresponds to the two-grid algorithm, one single approximate solution step is performed by a recursive call at each level l.

Error amplification operator of the V-cycle: recursive representation.

For convergence analysis, an MG cycle is interpreted in a recursive way as a perturbed TG cycle. For the TG cycle on level l, let us denote the iteration matrix (or error amplification operator) by

$$G_l^{TG} = S_l^{(post)} (I_l - C_l^{TG} A_l) S_l^{(pre)}, \qquad C_l^{TG} = P_{l,l-1} A_{l-1}^{-1} R_{l-1,l} \tag{13.40}$$

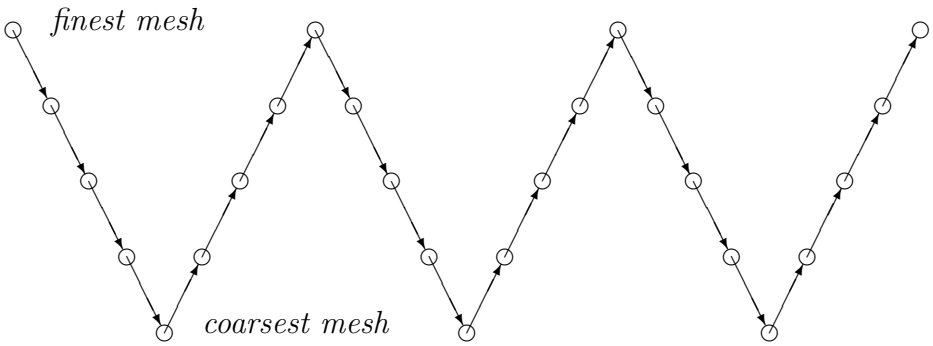


Figure 13.9: Three V-cycles

Algorithm 13.3 Basic MG ('V-Cycle')

% calling sequence: $u = MG(u_0, b, l)$; input: initial guess u_0; output: approximation x

if level l is sufficiently small, compute $u = A_l^{-1} b$
else
    do $m = m_{pre}$ steps of smoothing (damped Jacobi) with initial guess u_0 to obtain u_N^m
    $r_m = b - A_l u_N^m$
    $\delta = MG(0, R_{l-1,l} r_m, l-1)$ % solve for correction recursively with initial guess 0
    $\tilde{u} = u_N^m + P_{l,l-1}\,\delta$
    do $m = m_{post}$ steps of smoothing (damped Jacobi) with initial guess $\tilde{u}$ to obtain u
end if
return u

In the level-l MG version of the V-cycle, A_{l-1}^{-1} is replaced by its level-$(l-1)$ MG approximation, which we denote by $N_{l-1}^{(l-1)}$. For the resulting MG analog of (13.40) we write

$$G_l^{(l)} = S_l^{(post)}(I_l - C_l^{(l-1)} A_l)\, S_l^{(pre)}, \qquad C_l^{(l-1)} = P_{l,l-1}\, N_{l-1}^{(l-1)}\, R_{l-1,l} \tag{13.41a}$$

Here, the approximate inverse operator $N_{l-1}^{(l-1)}$ is related to the corresponding level-$(l-1)$ MG amplification operator $G_{l-1}^{(l-1)}$ by

$$G_{l-1}^{(l-1)} = I_{l-1} - N_{l-1}^{(l-1)} A_{l-1}, \qquad \text{i.e.,} \qquad N_{l-1}^{(l-1)} = A_{l-1}^{-1} - G_{l-1}^{(l-1)} A_{l-1}^{-1}$$

Thus,

$$C_l^{(l-1)} = P_{l,l-1}\, N_{l-1}^{(l-1)}\, R_{l-1,l} = \underbrace{P_{l,l-1}\, A_{l-1}^{-1}\, R_{l-1,l}}_{= C_l^{TG}} - P_{l,l-1}\, G_{l-1}^{(l-1)} A_{l-1}^{-1}\, R_{l-1,l}$$

hence (13.41a) can be written in the recursive form

$$G_l^{(l)} = \underbrace{S_l^{(post)}(I_l - C_l^{TG} A_l)\, S_l^{(pre)}}_{= G_l^{TG}} + S_l^{(post)} P_{l,l-1}\, G_{l-1}^{(l-1)}\, A_{l-1}^{-1}\, R_{l-1,l}\, A_l\, S_l^{(pre)} \tag{13.41b}$$

Assuming $\|S_l^{(post)}\| < 1$, $\|S_l^{(pre)}\| < 1$, $\|P_l^{l,l-1}\| < C$, and

$$\|A_{l-1}^{-1}\, R_{l-1,l}\, A_l\, S_l^{(pre)}\| \le C, \quad C \text{ a uniform constant (typically } \ge 1), \tag{13.42}$$

this gives the recursive estimate

$$\|G_l^{(l)}\| \le \|G_l^{TG}\| + C\,\|G_{l-1}^{(l-1)}\|$$

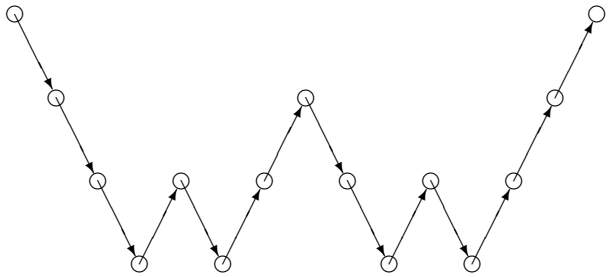


Figure 13.10: A W-cycle

with some constant C. With the abbreviations

$$\kappa_l^{TG} = \|G_l^{TG}\|, \qquad \kappa_l^{(l)} = \|G_l^{(l)}\|$$

this gives the recursion

$$\kappa_2^{(2)} = \kappa_2^{TG}, \quad \text{and} \quad \kappa_l^{(l)} \le \kappa_l^{TG} + C\,\kappa_{l-1}^{(l-1)}, \quad l = 3, 4, \ldots \tag{13.43}$$

We see that, even for $\|\kappa_l^{TG}\| < 1$ on all levels l (i.e., uniform contractivity of the TG cycle), it is not possible to derive uniform contraction bounds for the V-cycle in this way. Actually, existing convergence proofs for the V-cycle rely on a more explicit representation for $G_l^{(l)}$; see for instance [18, Sec. 3.3].

W-cycle, μ-cycle.

Viewing MG as a (linear) iteration scheme, it is natural to attempt to improve the approximation by μ *repeated* recursive calls. This leads to the so-called μ-*cycle* formulated in Alg. 13.4. The case $\mu = 1$ corresponds to the V-cycle (Alg. 13.3); the case $\mu = 2$ leads to the so-called *W-cycle*, visualized in Fig. 13.10.

We modify the above recursive representation for the case of the μ-cycle. Again we start with the normal TG cycle (13.40):[65]

$$G_l^{TG} = S_l^{(post)}(I_l - C_l^{TG} A_l)\, S_l^{(pre)}, \qquad C_l^{TG} = P_{p,l-1}\, A_{l-1}^{-1}\, R_{l-1,l} \tag{13.44}$$

In the level-l MG version of the μ-cycle, A_{l-1}^{-1} is replaced by its level-$(l-1)$ μ-cycle approximation $N_{l-1}^{(l-1)}$. For the resulting μ-cycle MG analog of (13.44) we write

$$G_l^{(l)} = S_l^{(post)}(I_l - C_l^{(l-1)} A_l)\, S_l^{(pre)}, \qquad C_l^{(l)} = P_{l,l-1} N_{l-1}^{(l-1)} R_{l-1,l} \tag{13.45a}$$

Here, the (approximate inverse) operator $N_{l-1}^{(l-1)}$ is related to μ-fold application of the corresponding level-$(l-1)$ MG amplification operator:

$$\big(G_{l-1}^{(l-1)}\big)^{\mu} = I_{l-1} - N_{l-1}^{(l-1)} A_{l-1}, \quad \text{i.e.,} \quad N_{l-1}^{(l-1)} = A_{l-1}^{-1} - \big(G_{l-1}^{(l-1)}\big)^{\mu} A_{l-1}^{-1}$$

Thus, (13.45a) can be written in the recursive form

$$G_l^{(l)} = \underbrace{S_l^{(post)}(I_l - C_l^{TG} A_l)\, S_l^{(pre)}}_{= G_l^{TG}} + S_l^{(post)} P_{l,l-1} \big(G_{l-1}^{(l-1)}\big)^{\mu} A_{l-1}^{-1} R_{l-1,l} A_l S_l^{(pre)} \tag{13.45b}$$

[65] This is slightly simplified: According to Alg. 13.4, we would start with μ TG coarse grid corrections. The difference between these versions is not essential.

In a similar way as for the V-cycle (see (13.43)) this gives the recursive estimate

$$\kappa_2^{(2)} = \kappa_2^{TG}, \quad \text{and} \quad \kappa_l^{(l)} \le \kappa_l^{TG} + C\left(\kappa_{l-1}^{(l-1)}\right)^{\mu}, \quad l = 3, 4, \ldots \tag{13.46}$$

for the contraction rate of the μ-cycle.

Exercise 13.7 Assume $\mu = 2$ and $\kappa_l^{TG} \le \rho \le \frac{1}{4C}$ on all levels l, where $C \ge 1$ is the constant in (13.46). (Thus, $\rho \le \frac{1}{4}$ is necessarily assumed.) Show: The W-cycle contraction rate $\kappa_l^{(l)}$ can be uniformly bounded by

$$\kappa_l^{(l)} \le \frac{1 - \sqrt{1 - 4C\rho}}{2C} \le 2\rho \le \frac{1}{2}$$

on all levels $l = 2, 3, 4, \ldots$.

Hint: The sequence $(\kappa_l^{(l)})$ is strictly increasing and majorized by the sequence defined by

$$\xi_2 = \rho, \qquad \xi_l = \rho + C\,\xi_{l-1}^2, \quad l = 3, 4, \ldots$$

Consider the latter as a monotonously increasing fixed point iteration. ∎

This argument can also be generalized to the case $\mu > 2$.

Algorithm 13.4 Multigrid (μ-Cycle)

% Calling sequence: $u = MG(u_0, b, l, \mu)$; input: initial guess u_0; output: approximation u
% $\mu = 1 \rightarrow$ V-cycle; $\mu = 2 \rightarrow$ W-cycle
if level l is sufficiently small, compute $u = A_l^{-1} b$
else
  do $m = m_{pre}$ steps of smoothing (damped Jacobi) with initial guess u_0 to obtain u_N^m
  $r_m = b - A_l\, u_N^m$
  $\delta^{(0)} = 0$
  for $\nu = 1$ to μ **do** $\delta^{(\nu)} = MG(\delta^{(\nu-1)}, R_{l-1,l} r_m, l-1, \mu)$
  $\tilde{u} = u_N^m + P_{l,l-1}\delta^{(\mu)}$
  do $m = m_{post}$ steps of smoothing (damped Jacobi) with initial guess $\tilde{u}$ to obtain u
end if
return u

On the basis the recursion (13.46), Exercise 13.7 provides a rather general convergence argument for the W-cycle. Still, we have assumed that (13.42) holds, i.e.,

$$\|A_{l-1}^{-1} R_{l-1,l} A_l S_l\| \le C$$

which looks quite natural but needs to be argued. Direct evaluation for the 1D and 2D Poisson examples shows that, with respect to the energy norms involved, C is indeed a moderate-sized a uniform constant ≥ 1. In the Galerkin context one may think of rewriting this as

$$A_{l-1}^{-1} R_{l-1,l} A_l S_l = \underbrace{A_{l-1}^{-1} R_{l-1,l} A_l P_{l,l-1}}_{= I_{l-1}} R_{l-1,l} S_l + A_{l-1}^{-1} R_{l-1,l} A_l (I_l - P_{l,l-1} R_{l-1,l}) S_l$$

and try to estimate (in norm) the second term on the right hand side, which is still not straightforward.

The cost of the μ-cycle MG method clearly grows with μ. The next exercise shows the trend.

Exercise 13.8 Let $d \geq 1$ be the spatial dimension and $N = N_L$ be the problem size of the finest mesh. Show: The cost of the μ-cycle is $\mathcal{O}(N)$ for $\mu \leq 2^d - 1$ and $\mathcal{O}(N \log N)$ for $\mu = 2^d$. ∎

Example 13.6 The TG and the MG methods are linear iterations. For the 1D model problem, Table 13.1 shows estimates for the corresponding contraction rates in the energy norm, obtained by numerical experiment: We observe the error reduction of MG for a random initial vector u_0, with $m_{pre} = m$ presmoothing steps and $m_{post} = 0$ postsmoothing steps. As to be expected, the two-grid method has the best contraction rate. The W-cycle ($\mu=2$) is very close to the TG method.

We also see that the contraction rate is rather small even for $\mu=1$ (V-cycle). ∎

V-cycle

N	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
$m=1$	0.333	0.330	0.327	0.323	0.320	0.320	0.313	0.312
$m=2$	0.156	0.175	0.191	0.198	0.203	0.205	0.207	0.207
$m=3$	0.089	0.105	0.118	0.127	0.131	0.134	0.136	0.138

W-cycle

N	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
$m=1$	0.333	0.329	0.312	0.320	0.330	0.329	0.328	0.327
$m=2$	0.116	0.116	0.114	0.116	0.116	0.115	0.115	0.114
$m=3$	0.073	0.077	0.077	0.078	0.076	0.077	0.077	0.077

Two-grid method

N	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
$m=1$	0.333	0.314	0.326	0.321	0.329	0.326	0.327	0.325
$m=2$	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
$m=3$	0.074	0.076	0.076	0.078	0.078	0.078	0.078	0.077

Table 13.1: Estimated contraction rates for V-cycle ($\mu = 1$), W-cycle ($\mu = 2$), and two-grid method.

Remark 13.6 The so-called *F-cycle* ('flexible cycle') is an intermediate version between V-cycle and W-cycle. A recursive call of the F-cycle corresponds to one recursive call of a V-cycle follows by one recursive call of a W-cycle. ∎

13.10 Nested Iteration and Full Multigrid (FMG)

One of the basic questions in iterative solution techniques is finding a good starting vector. *Nested iteration* is a general technique for this. The basic idea is the following: A good starting vector u_0 for the fine grid problem $A_N u = b_N$, might be the appropriately prolongated solution of a coarse grid problem. Since the coarse grid problem cannot be solved exactly either, we solve it iteratively and need a good starting vector for that iteration as well.

Effectively, we start at the coarsest level, where an exact solution is available; then, we prolongate this solution to a finer level where an approximate solution technique such as MG can be used; the approximate solution obtained in this way is transferred to the next fine grid as a starting point for a MG iteration, etc., until we reach the finest level with a good initial guess for the final MG cycle.

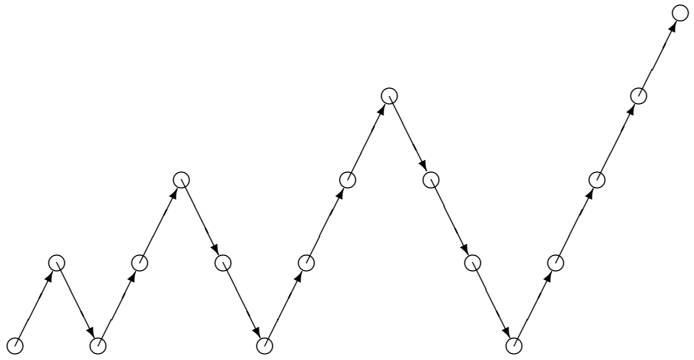


Figure 13.11: Full Multigrid (FMG)

In Fig. 13.11 a single pass of this procedure is visualized (think of starting from the left, with exact solution at the coarsest level, and 'going up'). However, such a single pass not necessarily yields a sufficiently accurate result, and the process is again iterated. To this end we use a *recursive* approach: *For an intermediate approximation on level l, we compute the new approximation by coarse grid correction using FMG on level $l-1$, see Alg. 13.5, with* 0 *as the natural initial guess for the correction.* In general, one pass of the FMG procedure uses μ' MG cycles of the type μ-cycle (Alg. 13.4) at each level l.

Algorithm 13.5 Full Multigrid

% calling sequence: $u = FMG(u_0, b, l)$; input: initial guess u_0; output: approximation u

% μ, $\mu' \geq 1$ given

if l is sufficiently small, **compute** $u = A_l^{-1} b$
else
  $r_l = b - A_l u_0$
  $\delta = FMG(0, R_{l-1,l} r_l, l-1)$
  $u^{(0)} = u_0 + P_{l,l-1}\delta$
  for $\nu = 1$ to μ' **do** $u^{(\nu)} = MG(u^{(\nu-1)}, b, l, \mu)$
  $u = u^{(\nu)}$
end if
return u

Unless some initial approximation u_0 is available, the process is initiated by calling $FMG(0, b, l)$, i.e., $u_0 = 0$ with initial residual b. Thus, $FMG(0, R_{l-1,l} b, l-1)$ is called in order to obtain a good initial solution. Due to the recursion, this means that we 'go up' from the bottom level with FMG, interpolation and μ' additional MG steps to obtain a first approximation u_1 on level l, with residual $r_l = b - A u_1$. When applying FMG iteratively, in the second iteration $FMG(u_1, b, l)$ calls $FMG(0, R_{l-1,l} r_l, l-1)$, and again we go up from the bottom level with FMG, interpolation and μ' additional MG steps to obtain a next approximation u_2 on level l, etc.

Example 13.7 We illustrate the performance of the FMG algorithm for the 1D model problem in Fig. 13.12, i.e., we plot the errors $\|u_m - u\|_\infty$ and the energy norm error $\|u_m - u\|_{A_L}$ of the iteration $u_{m+1} = FMG(u_m, f, L)$ for different values of L (with $N = 2^L$). We note the considerable performance improvement due to the good initial guesses. Here, $m_{pre} = 3$ and $m_{post} = 0$ and $\mu = \mu' = 1$. ∎

The FMG method is of course more expensive. However, the cost of one cyle of FMG is still $\mathcal{O}(N)$ as for the standard MG cycle.

Optimal convergence properties of FMG.

The numerical evidence of Example 13.7 shows that the advantage of FMG becomes more pronounced as the problem size N increases. The following Theorem 13.4 is a way of formalizing this observation. We

N	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}
$\frac{t_{FMG}}{t_{MG}}$	3.3	3.2	3.9	4.6	5	4.6	4.8	4.3	4	3.4	2.9	2.6	2.5	2.5	2.5	2.5

Table 13.2: Ratio of CPU-time per iteration of FMG vs. MG ($m_{pre} = 3$).

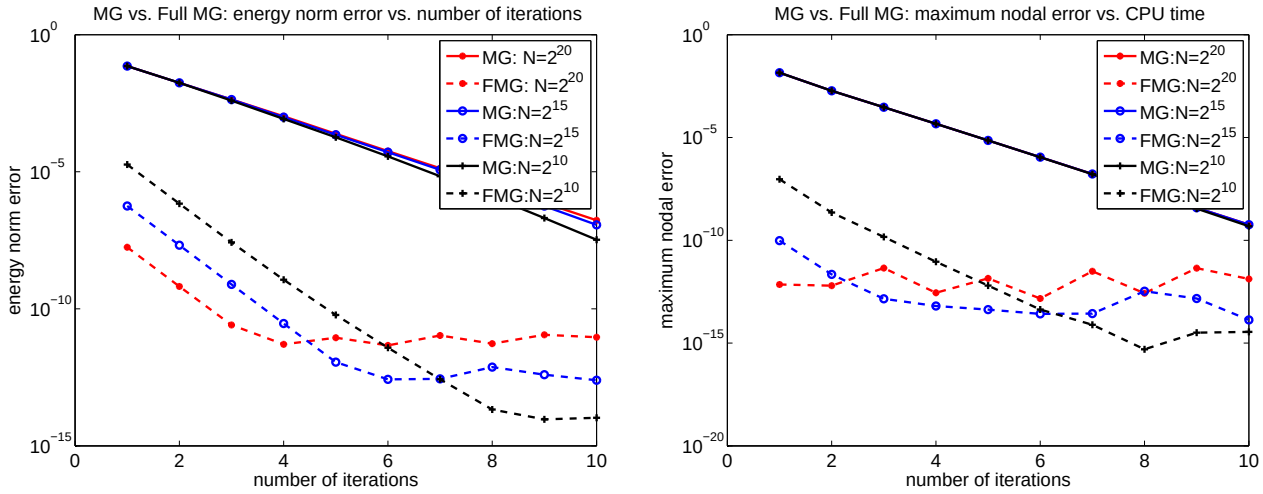


Figure 13.12: Convergence behavior of FMG compared with MG.

consider again a sequence of meshes $\mathcal{T}_l$ with mesh sizes $h_l \sim 2^{-l}$. We assume that the *exact* (Galerkin) approximation $u_l \in \hat{U}_l$ on level l to the exact solution u_* of the original (PDE) problem satisfies (typically, in the energy norm $\|\cdot\| = \|\cdot\|_A$),

$$\|u_l - u_*\| \leq K\, h_l^p \tag{13.47}$$

for some $p > 0$ and all l.

Theorem 13.4 *Let $\bar{c} = \sup_l h_{l-1}/h_l$, and let $\kappa = \kappa(\mu)$ be the contraction rate of the multigrid μ-cycle. Let p be as in* (13.47)*, and let μ' be the number of μ-cycles used in the FMG algorithm 13.5. Assume $\bar{c}^p \kappa^{\mu'} < 1$. Then there exists a constant $C' > 0$ such that one cycle of FMG results in an approximation $\tilde{u}_l \in \hat{U}_l$ which satisfies*

$$\|\tilde{u}_l - u_*\| \leq C' K h_l^p$$

Proof: We proceed by induction on l. For the approximations $\tilde{u}_l$ (obtained by FMG) to the exact solutions u_l on level l, we denote the 'algebraic' FMG error on level l by $\tilde{e}_l = \tilde{u}_l - u_l$. Clearly, $\tilde{e}_0 = 0$. FMG on level l consists of μ' steps of classical MG (with contraction rate $\kappa = \kappa(\mu)$), with an initial error $\tilde{e}_{l-1} = \tilde{u}_{l-1} - u_l$. Hence,

$$\begin{aligned}
\|\tilde{e}_l\| &\leq \kappa^{\mu'} \|\tilde{u}_{l-1} - u_l\| \leq \kappa^{\mu'} \big(\|\tilde{u}_{l-1} - u_{l-1}\| + \|u_{l-1} - u_l\|\big) \\
&\leq \kappa^{\mu'} \big(\|\tilde{u}_{l-1} - u_{l-1}\| + \|u_{l-1} - u_*\| + \|u_l - u_*\|\big) \\
&\leq \kappa^{\mu'} \big(\|\tilde{e}_{l-1}\| + K h_{l-1}^p + K h_l^p\big) \leq \kappa^{\mu'} \big(\|\tilde{e}_{l-1}\| + (1 + \bar{c}^p)\, K\, h_l^p\big)
\end{aligned}$$

Iterating this inequality, we obtain with $\tilde{e}_0 = 0$:

$$\|\tilde{e}_l\| \leq K\,(1 + \bar{c}^p)\, \kappa^{\mu'} \big(h_l^p + \kappa^{\mu'} h_{l-1}^p + \kappa^{2\mu'} h_{l-2}^p + \cdots + \kappa^{l\mu'} h_0^p\big)$$

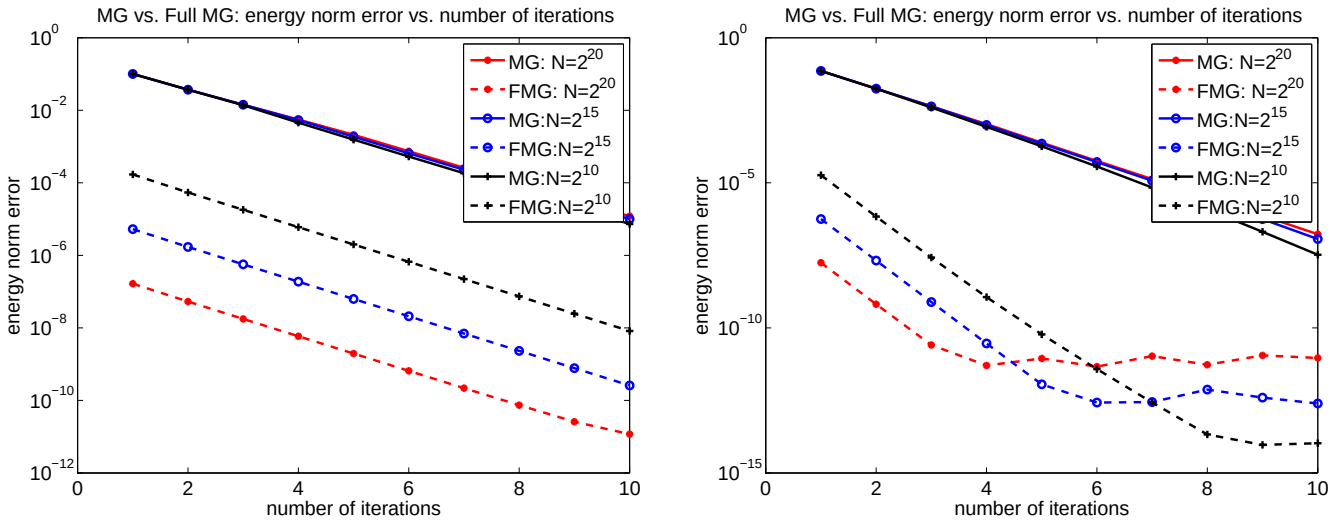


Figure 13.13: Convergence behavior of MG and FMG for $\mu = 1$, $\mu' = 1$, $m_{post} = 0$. Left: $m_{pre} = 1$. Right: $m_{pre} = 3$. For comparison: the discretization errors (in energy norm) for $l = 10$, $l = 15$, and $l = 20$ are: $e_{10} = 2.82_{10}-4$, $e_{15} = 8.81_{10}-4, e_{20} = 2.75_{10}-7$.

The definition of $\bar{c}$ implies $h_{l-i} \le \bar{c}^i h_l$. Hence, together with assumption $\bar{c}^p \kappa^{\mu'} < 1$ we obtain

$$\|\tilde{e}_l\| \le K\,(1+\bar{c})\,\kappa^{\mu'} h_l^p \big(1 + \kappa^{\mu'}\bar{c}^p + \kappa^{2\mu'}\bar{c}^{2p} + \cdots + \kappa^{l\mu'}\bar{c}^{lp}\big) \le \frac{K(1+\bar{c})}{1-\kappa^{\mu'}\bar{c}^p}\,\kappa^{\mu'} h_l^p$$

Thus,

$$\|\tilde{u}_l - u_*\| \le \|\tilde{e}_l\| + \|u_l - u_*\| \le C' K h_l^p$$

with an appropriate constant C'. □

Theorem 13.4 implies the following observations:

- Consider FMG for linear systems arising after discretization of elliptic boundary value problems. The exact solutions $u_l \in \hat{U}_l$ are approximations to the unknown solution u_*. Therefore, it suffices to obtain approximations $\tilde{u}_l \approx u_l$ up to the discretization error $u_l - u_* \sim h_l^p$. Theorem 13.4 shows that this can be achieved with *one* cycle of FMG. The cost for one cycle of FMG is $\mathcal{O}(N_L)$.
- Standard MG starts with the initial guess 0. Hence, with a level-independent contraction rate $\rho < 1$, the error after k steps of MG is $\mathcal{O}(\rho^k)$. Thus, to reach the level of the discretization error $u_L - u_* \sim h_L^p$, one needs $\mathcal{O}(|\log h_L|^p) = \mathcal{O}(L)$ steps. In terms of the problem size $N_L \sim h_L^d$, we need $\mathcal{O}(\log N_L)$ steps; the total cost is therefore $\mathcal{O}(N_L \log N_L)$.
- The condition $\bar{c}^p \kappa^{\mu'} < 1$ could, in principle, be enforced by increasing μ' (note: μ' enters only linearly in the complexity estimates for FMG), or by increasing the number of smoothing steps.

Example 13.8 In Fig. 13.13 we demonstrate the behavior of the MG method compared with the FMG method for the 1D model problem $-u'' = 1$. The left figure shows the case $m_{pre} = 1$, whereas the right figure shows the case $m_{pre} = 3$. We note that FMG with $m_{pre} = 1$ attains an approximation at the level of the discretization error after a single pass. ■

Remark 13.7 Using multigrid as a preconditioner for a Krylov subspace method works in an analogous way as for a two-grid preconditioner; see Remark 13.5. ■

13.11 Nonlinear problems

Like other stationary iterative schemes, MG is not a priori a linear technique. It can be adapted to nonlinear problems, with some reorganization concerning the correction steps, where nonlinearity is taken into account.

Full approximation scheme (FAS).

The basic idea is rather general. Consider a system of equations in the form

$$A_N(u) = b_N \tag{13.48}$$

with a nonlinear mapping $A_N : \mathbb{R}^N \to \mathbb{R}^N$. We assume that the problem is well-posed (with, at least, a locally unique solution) and consider a (simpler) approximation $\tilde{A}_N(u) \approx A_N(u)$. The mapping $\tilde{A}_N$ is an (in general, also nonlinear) preconditioner for A_N.

Assume that a first guess u_0 for the solution u_* of (13.48) is available. For $\tilde{A}_N(u) \approx A_N(u)$ we may expect

$$\tilde{A}_N(u_*) - \tilde{A}_N(u_0) \approx A_N(u_*) - A_N(u_0) = b_N - A_N(u_0) = r_0 \tag{13.49}$$

Here, the right hand side is the residual r_0 of u_0 with respect to (13.48). This suggests to compute a corrected approximation u_1 as the solution of

$$\tilde{A}_N(u_1) = \tilde{A}_N(u_0) + r_0 = b_N + (\tilde{A}_N - A_N)u_0 \tag{13.50}$$

If $\tilde{A}_N$ is linear, this can be written as the usual 'correction scheme' as in stationary iteration methods, where the correction δ is a linear image of the residual,

$$\tilde{A}_N \delta = r_0, \qquad u_1 = u_0 + \delta \tag{13.51}$$

In the general, nonlinear case, (13.50) is solved directly for the new approximation u_1. This is called a 'full approximation scheme' (FAS).

In the context of MG, $\tilde{A}_N$ is defined via a coarse grid approximation A_n of A_N of dimension $n < N$, and we have to be careful concerning the intergrid transfer. Assume that u_0 is an appropriately smoothed approximation on the finer grid. With our usual notation for the restriction and prolongation operators, the nonlinear FAS-type two-grid method reads:

1. Restrict u_0 to the coarser space: $\tilde{u}_0 = R_{n,N}\, u_0$
2. Compute the residual and restrict it to the coarser space: $\tilde{r}_0 = R_{n,N}\, r_0 = R_{n,N}(b_N - A_N(u_0))$
3. Solve $A_n(\tilde{v}) = A_n(\tilde{u}_0) + \tilde{r}_0$
4. Compute the correction on the coarse grid: $\delta = \tilde{v} - \tilde{u}_0$
5. Prolongate the correction δ and add it to u_0, i.e., compute $u_1 = u_0 + P_{N,n}\, \delta$

Note that the coarse grid solution $\tilde{v}$ is not directly prolongated, but the coarse-grid *correction* δ obtained by the FAS step is prolongated and added to u_0. In this way, for the linear case the two-grid FAS scheme becomes equivalent to our two-grid 'correction scheme' (CS) formulation.

The generalization to FAS-type multigrid is straightforward (V-cycle, μ-cycle, FMG). Note that the multigrid version requires the solution of a nonlinear problem *only on the coarsest grid*, e.g., by a procedure of Newton type. More implementation details and examples for applying FAS-type multigrid to nonlinear boundary value problems can be found in [4].

Exercise 13.9 Formulate the nonlinear FAS-type TG coarse grid correction in detail for a standard finite difference discretization of

$$-u'' + \phi(u)u = f(x)$$

with homogeneous Dirichlet boundary conditions. (This problem is well-posed for $\phi(u) \geq 0$.) ∎

The smoothing procedure can usually be chosen in a similar way as for related linear(ized) problem.

Example 13.9 For the problem from Exercise 13.9, for instance, discretization leads to an algebraic system of the form

$$(A_N + \Phi_N(u))u = b_N$$

with a nonlinear diagonal mapping $\Phi_N(u)u = (\phi(u_1)u_1, \ldots, \phi(u_n)u_N)$. The corresponding damped Jacobi smoother is based on inverting the corresponding 'diagonal preconditioner'

$$S_N(u) = (D_N + \Phi_N(u))u, \quad D_N = \mathrm{diag}(A_N)$$

A standard Jacobi step can be defined in an FAS-type manner:

$$u_0 \;\mapsto\; u_1 = \text{solution of } \; S_N(u_1) = S_N(u_0) + r_0, \quad r_0 = b_N - (A_N + \Phi_N(u_0))u_0$$

In practice, this is realized by a (scalar) Newton procedure, with the Jacobian

$$DS_N(u) = D_N + D\Phi_N(u)$$

A single Newton step starting from u_0 takes the form

$$u_1^{(1)} = u_0 - (DS_N(u_0))^{-1} r_0$$

or equivalently, $u_1^{(1)} = u_0 + \delta$, where the correction δ is the solution of

$$DS_N(u_0)\,\delta = r_0$$

For damped Jacobi we set $u_1^{(1)} = u_0 + \omega\,\delta$, with an appropriate damping factor ω. ∎

Naturally, the detailed choice for the algorithmic components is not obvious in the nonlinear case. Some detailed case studies are presented in [4].

Nonlinear Galerkin and FAS type two-grid scheme.

The weakly nonlinear Poisson problem from Exercise 13.9 is an example of a nonlinear elliptic problem for which the linear theory can be extended in a rather straightforward manner. An introduction to the theory of nonlinear elliptic problems can be found in [23]. Here, we restrict ourselves to the abstract specification of the weak formulation, together with the Galerkin/FEM approximation and the corresponding two-grid procedure.

The weak formulation of a nonlinear elliptic problem takes the form

$$a(u, v) = f(v) \quad \forall\, v \in V \tag{13.52}$$

with a bilinear form $a(u, v)$ which depends in a nonlinear way on u. (In the simplest case of a second order problem posed on a domain Ω with homogeneous Dirichlet boundary conditions, $V = H_0^1(\Omega)$.)

As in the linear case, a Galerkin/FEM approximation defined on a finite-dimensional subspace $\hat{U}_N \subseteq V$ via the (nonlinear) Galerkin conditions

$$a(\hat{u}, \hat{v}) = f(\hat{v}) \quad \forall\, \hat{v} \in \hat{U}_N \tag{13.53}$$

With the usual FEM isomorphism $u \in \mathbb{R}^{N-1} \leftrightsquigarrow \hat{u} \in \hat{U}_N$, this corresponds to a nonlinear algebraic system

$$A_N(u) = b_N$$

Consider an approximation $\hat{u}_0 \in \hat{U}_N$ for the solution $\hat{u}_N$ of (13.53) obtained after smoothing,[66] with error $\hat{e}_0 = \hat{u}_0 - \hat{u}_N$. Error and residual are related via[67]

$$a(\underbrace{\hat{u}_0 - \hat{e}_0}_{=\,\hat{u}_N}, \hat{v}) - a(\hat{u}_0, \hat{v}) = f(\hat{v}) - a(\hat{u}_0, \hat{v}) \quad \forall\, \hat{v} \in \hat{U}_N \tag{13.54}$$

In the nonlinear two-grid method we seek an approximation $\hat{u}_N^{TG}$ to $\hat{u}_N$ of the form

$$\hat{u}_N^{TG} = \hat{u}_0 + \hat{\delta}_{N/2}$$

with $\hat{\delta}_{N/2} \in \hat{U}_{N/2} \subseteq \hat{U}_N$. As in the linear case, the coarse-grid correction $\hat{\delta}_{N/2}$ is an approximation to the unknown 'exact correction' $-\hat{e}_0$; from (13.54) we see that, at least formally, it is defined in a natural way by the Galerkin condition

$$a(\hat{u}_0 + \hat{\delta}_{N/2}, \hat{w}) - a(\hat{u}_0, \hat{w}) = f(\hat{w}) - a(\hat{u}_0, \hat{w}) =: \hat{r}_0(\hat{w}) \quad \forall\, \hat{w} \in \hat{U}_{N/2} \tag{13.55}$$

which simplifies to (13.31b) in the linear case. Since the coarse grid correction step is realized within the smaller subspace $\hat{U}_{N/2}$,on the left-hand side of (13.55) we replace $\hat{u}_0$ by $\tilde{\hat{u}}_0 \in \hat{U}_{N/2}$. This leads to

$$\text{Find } \hat{\delta}_{N/2} \in \hat{U}_{N/2} \text{ such that } \; a(\underbrace{\tilde{\hat{u}}_0 + \hat{\delta}_{N/2}}_{=:\,\hat{v}_{N/2}}, \hat{w}) - a(\tilde{\hat{u}}_0, \hat{w}) = \hat{r}_0(\hat{w}) \quad \forall\, \hat{w} \in \hat{U}_{N/2}\,. \tag{13.56}$$

This is realized by an FAS step:

$$\text{Compute } \hat{v}_{N/2} \in \hat{U}_{N/2} \text{ such that } \; a(\hat{v}_{N/2}, \hat{w}) = a(\tilde{\hat{u}}_0, \hat{w}) + \hat{r}_0(\hat{w}) \quad \forall\, \hat{w} \in \hat{U}_{N/2} \tag{13.57}$$

followed by

$$\hat{\delta}_{N/2} = \hat{v}_{N/2} - \tilde{\hat{u}}_0$$

and the new approximation $\hat{u}_1$ is obtained in the form

$$\hat{u}_1 = \hat{u}_0 + \hat{\delta}_{N/2}$$

We see that this process is nothing but a 'weak formulation' of the FAS coarse grid correction scheme from Sec. 13.11. In matrix/vector formulation it is the same, with an appropriate definition of the coarse grid Galerkin approximation $A_{N/2} = A_{N/2}(u)$.

[66] Remark concerning notation: In the linear case we have used the notation u_m instead of u_0 (Sec. 13.6). All other denotations, in particular for the natural restriction and prolongation operators, are the same as in Sec. 13.6.

[67] The right hand side of (13.54) is called the *weak residual* of $\hat{u}_0$; more precisely: the weak residual of $\hat{u}_0$ is the $\hat{u}_0$-dependent functional $\hat{r}_0 : \hat{v} \mapsto f(\hat{v}) - a(\hat{u}_0, \hat{v})$.

14 Substructuring Methods

Multigrid techniques are relatively easy to implement on simple geometries and regular grids. If more complicated geometries are involved, it is often useful to use substructuring techniques, e.g., by partitioning the underlying domains into several subdomains and use some divide-and-conquer technique in the preconditioning process. An example is an L-shaped domain in $\mathbb{R}^2$ partitioned into three rectangles. This approach is called *domain decomposition*. Domain decomposition may also be motivated by storage limitations (the solver need to be based on smaller problems). Some of these techniques are also natural candidates for parallelization. Domain decomposition is also a useful approach if the underlying problem is of a different nature in different parts of the domain, or if fast direct solvers can be used for the subproblems.

Like multigrid methods, substructuring techniques are frequently used for preconditioning Krylov subspace methods like CG or GMRES.

14.1 Subspace corrections

In this section we introduce substructuring techniques using a general, abstract formulation. We will also see that many of the methods discussed before fit naturally into this framework. We consider an abstract variational problem, in weak formulation, on a finite-dimensional space V,

$$\text{Find } u \in V \text{ such that } \quad a(u,v) = f(v) \quad \forall\, v \in V \tag{14.1}$$

with a bounded SPD[68] bilinear form $a(\cdot,\cdot)$ and associated energy norm $\|u\|_A = \sqrt{a(u,u)}$.

Assume that an approximation u_0 for the exact solution u_* of (14.1) is given, with error $e_0 = u_0 - u_*$ and weak residual r_0,

$$r_0(v) = f(v) - a(u_0, v) = a(-e_0, v), \quad v \in V \tag{14.2}$$

Exact solution of the the correction equation

$$\text{Find } \delta \in V \text{ such that } \quad a(\delta, v) = r_0(v) \quad \forall\, v \in V \tag{14.3}$$

would result in the negative error, $\delta = -e_0$, i.e., the exact correction such that $u_0 + \delta = u_*$. For the construction of a preconditioner we approximate (14.3) by means of subspace correction techniques.

Let $V_1 \subseteq V$ be a linear subspace of V. Analogously as for the two-grid method, the associated 'subspace correction', i.e., the solution $\delta_1 \in V_1$ of

$$a(\delta_1, v_1) = r_0(v_1) \quad \forall\, v_1 \in V_1 \tag{14.4}$$

is the projection[69] with respect to the $a(\cdot,\cdot)$ inner product of the negative error $-e_0 = u_* - u_0$ onto the subspace V_1. The improved approximation

$$u_1 = u_0 + \delta_1 \tag{14.5}$$

is optimal in the following sense: It satisfies the Galerkin orthogonality relation

[68] Here, SPD means that $a(\cdot,\cdot)$ is symmetric and coercive, or 'elliptic', i.e., $a(u,u) \geq \gamma\,(u,u) > 0$ uniformly for all $u \in V$.

[69] Projection of $-e_0$:

$$\begin{aligned} & a(\delta + e_0, v_1) = 0 \quad \forall\, v_1 \in V_1 \\ \Leftrightarrow \quad & a(\delta, v_1) = a(-e_0, v_1) = r_0(v_1) \quad \forall\, v_1 \in V_1 \end{aligned}$$

$$e_1 = u_1 - u_* = \delta_1 - (-e_0) \perp_A V_1$$

which can be written in terms of the new residual $r_1(v) = f(v) - a(u_1, v) = a(-e_1, v)$,

$$\begin{aligned} r_1(v_1) &= f(v_1) - a(u_1, v_1) = a(-e_1, v_1) \\ &= \underbrace{f(v_1) - a(u_0, v_1)}_{= r_0(v_1)} - \underbrace{a(\delta_1, v_1)}_{= r_0(v_1)} = 0 \quad \forall\, v_1 \in V_1 \end{aligned} \tag{14.6}$$

This means that the new residual $r_1(v)$ vanishes on the subspace V_1, and is in turn equivalent to the best approximation property

$$\|u_1 - u_*\|_A = \min_{v_1 \in V_1} \|v_1 - u_*\|_A \tag{14.7}$$

More generally, we consider a family of subspaces $V_0, \ldots, V_N$ which span V,

$$\sum_{i=0}^{N} V_i = V \tag{14.8}$$

i.e., each $v \in V$ can be written in the form $v = \sum_{i=0}^{N} v_i$ with $v_i \in V_i$. The sum in (14.8) is not necessarily assumed to be a direct sum ($\bigoplus$): Some of the V_i may be 'overlapping' (i.e., the intersections $V_i \cap V_j$ may have a positive dimension). In this case the representation $v = \sum_{i=0}^{N} v_i$ is not unique.

Moreover, the individual subspace corrections analogous to (14.4) may be not exact. We assume that the subspace corrections are of a more general form, obtained as the solutions $\delta_i \in V_i$ of

$$a_i(\delta_i, v_i) = r_0(v_i) \quad \forall\, v_i \in V_i \tag{14.9}$$

where $a_i(\cdot, \cdot)$ is an SPD bilinear form on V_i which is not necessarily identical with the action of the given form $a(\cdot, \cdot)$ restricted to V_i. (Think, for instance, of a multigrid cycle playing the role of an approximate local solver.)

Different versions of substructuring methods are characterized by the way how the corresponding subspace corrections are combined. We will see that, typically, the outcome is a preconditioner for (14.1) which may also be written in a weak form, analogous to (14.1),

$$\text{Find } \delta \in V \text{ such that } \; b(\delta, v) = r_0(v) \quad \forall\, v \in V \tag{14.10}$$

The bilinear form $b(\cdot, \cdot)$ is an approximation for $a(\cdot, \cdot)$; it represents the preconditioner. It is usually not written down explicitly; rather, it is implicitly defined by the particular type of subspace correction algorithm used.

For studying the properties of $b(\cdot, \cdot)$ it is not essential what particular functional appears on the right hand side in (14.10). Therefore, to study the action of the preconditioner we again write

$$\text{Find } \tilde{u} \in V \text{ such that } \; b(\tilde{u}, v) = f(v) \quad \forall\, v \in V \tag{14.11}$$

in analogy to (14.1), and we consider 'subspace solutions' instead of 'subspace corrections'.

14.2 Additive Schwarz (AS) methods

The simplest version of a global technique is to compute independent subspace solutions $u_i \in V_i$ and add them up,

$$\tilde{u} = \sum_{i=0}^{N} u_i \tag{14.12a}$$

where the u_i are the solutions of

$$a_i(u_i, v_i) = f(v_i) \quad \forall\, v_i \in V_i, \qquad i = 0 \dots N \tag{14.12b}$$

For historical reasons, this is called an *additive Schwarz method* (AS). Naturally, the u_i can be computed in parallel because there is no data dependency between the subproblems. Note that even for the case of exact local solvers, $a_i = a|_{V_i}$, and despite $\sum_{i=0}^{N} V_i = V$, the add-up subspace solution $\tilde{u}$ from (14.12a) is not the exact solution of (14.1).

For a study of the properties of an AS preconditioner it is favorable to refer to an explicit basis representation.

Representation of an AS preconditioner.

We identify the n-dimensional space V with[70] $\mathbb{R}^n$, and choose a basis $\{x_1, \dots, x_n\}$ for V and arbitrary bases $\{x_{i,1}, \dots, x_{i,n_i}\}$ for each subspace V_i, where $n = \dim(V)$ and $n_i = \dim(V_i)$. Let

$$X = \left(x_1 \,\middle|\, \dots \,\middle|\, x_n \right) \in \mathbb{R}^{n \times n} \qquad \text{and} \qquad X_i = \left(x_{i,1} \,\middle|\, \dots \,\middle|\, x_{i,n_i} \right) \in \mathbb{R}^{n \times n_i}, \quad i = 0 \dots N$$

be the columnwise matrix representations of these bases. For the representation of vectors v, v_i with respect to these bases we use the denotation[71] (with $v \in V$, $v' \in \mathbb{R}^n$, and $v_i \in V_i$, $v_i'' \in \mathbb{R}^{n_i}$),

$$v = X\, v' \qquad \text{and} \qquad v_i = X_i\, v_i'', \quad i = 0 \dots N$$

With respect to the global basis X, the stiffness matrix $A \in \mathbb{R}^{n \times n}$, i.e., the SPD matrix representation of $a(\cdot, \cdot)$ is given by

$$A = \left(a_{j,k}\right) = a(x_j, x_k)$$

Let the $A_i \in \mathbb{R}^{n_i \times n_i}$ analogously defined as the local stiffness matrices of the (inexact) bilinear forms $a_i(\cdot, \cdot)$ with respect to the bases of the V_i, i.e.,

$$A_i = \left((a_i)_{j,k}\right) = a_i(x_{i,j}, x_{i,k}), \quad i = 0 \dots N$$

By assumption on the $a_i(\cdot, \cdot)$, the matrices A_i are also SPD.

The linear functional $f(v)$ is represented in a similar way: For $v = X\, v'$ we have

$$f(v) = {f'}^{\mathrm{T}} v'$$

(Riesz representation with some $f' \in \mathbb{R}^n$), and for $v_i = X_i\, v_i''$ we have a representation

$$f(v_i) = {f_i''}^{\mathrm{T}} v_i'', \quad i = 0 \dots N$$

[70] As usual, this corresponds to a canonical isomorphism, e.g., between functions and associated coordinate vectors in the FEM context.

[71] For each i, the index $''$ is used as a 'generic' symbol for the individual coordinate vectors with respect to the individual bases in the subspaces V_i.

with $f_i'' \in \mathbb{R}^{n_i}$. In this notation, the original and subspace problems take the form

$$A\,u' = f', \qquad \text{and} \qquad A_i\,u_i'' = f_i'', \quad i = 0 \ldots N$$

Let $E_i \in \mathbb{R}^{n \times n_i}$ denote the matrix representation of the embedding $V_i \subseteq V$, i.e., E_i is defined by $X E_i = X_i$,

$$E_i = X^{-1} X_i \in \mathbb{R}^{n \times n_i}, \quad i = 0 \ldots N \tag{14.13}$$

In other words, E_i transforms the i-th coordinate representation of some $v_i \in V_i$ into the global coordinate representation with respect to the basis X. In particular, for each fixed i and $v_i \in V_i$ we have

$$v_i = X\,v_i' = X_i\,v_i'', \quad \Rightarrow \quad v_i' = E_i\,v_i'' \tag{14.14a}$$

and

$$f(v_i) = f'^{\mathrm{T}} v_i' = f_i''^{\mathrm{T}} v_i'' \quad \Rightarrow \quad f_i'' = E_i^{\mathrm{T}} f' \tag{14.14b}$$

We see that the E_i and E_i^{T} play the role of prolongation and restriction operators, respectively.

Now each individual subspace solution u_i from (14.12b) is represented as $u_i = X_i\,u_i''$, where u_i'' is the solution of

$$A_i\,u_i'' = f_i'', \quad i = 0 \ldots N \tag{14.15}$$

Thus,[72]

$$u_i' = E_i\,u_i'' = E_i\,A_i^{-1} f_i'' = E_i\,A_i^{-1} E_i^{\mathrm{T}} f_i' =: \hat{B}_i\,f_i', \quad i = 0 \ldots N \tag{14.16}$$

Lemma 14.1 *For any choice of bases X and X_i, the matrix representation of the preconditioner defined by* (14.12) *is given in the following way: The symmetric matrix*

$$\hat{B} := \sum_{i=0}^{N} \hat{B}_i = \sum_{i=0}^{N} E_i\,A_i^{-1} E_i^{T} \in \mathbb{R}^{n \times n} \tag{14.17a}$$

is SPD, and $B := \hat{B}^{-1}$ is the matrix representation of the AS preconditioner $\tilde{u}$ from (14.11)*, i.e., we have $\tilde{u} = X\,\tilde{u}'$, where $\tilde{u}'$ is the solution of*

$$B\,\tilde{u}' = f' \tag{14.17b}$$

Remark 14.1 Note that

$$B \approx A \quad \text{and} \quad \hat{B} = B^{-1} \approx A^{-1}$$

We also note that for an exact Galerkin subspace correction, i.e. $a_i(\cdot,\cdot) = a(\cdot,\cdot)$, we have

$$A_i = E_i^T A\,E_i, \quad \text{and} \quad \hat{B}_i = E_i \left(E_i^{\mathrm{T}} A\,E_i\right)^{-1} E_i^{\mathrm{T}}$$

and $\hat{B}_i\,A$ is the A-orthogonal projector onto the range of E_i: $(\hat{B}_i\,A)^A = A^{-1}(\hat{B}_i\,A)^{\mathrm{T}} A = \hat{B}_i\,A$, and

$$\left(\hat{B}_i\,A\right)^2 = E_i \left(E_i^{\mathrm{T}} A\,E_i\right)^{-1} E_i^{\mathrm{T}} A\,E_i \left(E_i^{\mathrm{T}} A\,E_i\right)^{-1} E_i^{\mathrm{T}} A = \hat{B}_i\,A$$

For general A_i, we call $\hat{B}_i\,A$ with the symmetric matrix $\hat{B}_i$ from (14.16) a 'projection-like' operator. ∎

Proof: (of Lemma 14.1) For arbitrary $f' \in \mathbb{R}^n$ we have

$$\left(\hat{B} f', f'\right) = \left(\sum_{i=0}^{N} E_i\,A_i^{-1} E_i^{\mathrm{T}} f', f'\right) = \sum_{i=0}^{N} \left(E_i\,A_i^{-1} E_i^{\mathrm{T}} f', f'\right) = \sum_{i=0}^{N} \left(A_i^{-1} E_i^{\mathrm{T}} f', E_i^{\mathrm{T}} f'\right)$$

with $E_i^{\mathrm{T}} f' \in \mathbb{R}^{n_i}$. For $f' \neq 0$ each term in the sum is positive because the A_i have been assumed to be SPD. This proves that the matrix $\hat{B}$ defined in (14.17a) is also SPD.

[72] Note that the matrices $\hat{B}_i$ as well as $\hat{B}$ below refer to the global basis X.

Furthermore, the AS subspace solution is defined by the summing up the individual subspace solutions u_i, cf. (14.12). With (14.16) we obtain

$$\tilde{u}' = \sum_{i=0}^{N} \hat{B}_i f' = \hat{B} f' = B^{-1} f'$$

This completes the proof. □

The SPD matrix $B = \hat{B}^{-1}$ is implicitly defined by (14.17a), and the solution of (14.17b) represents the action of AS preconditioning with respect to the basis X. The SPD matrix B represents an SPD bilinear form $b(\cdot,\cdot)$ in V via the identity

$$b(u,v) = (B u', v') \quad \text{for} \quad u = X u',\ v = X v' \tag{14.18a}$$

Therefore the action of AS preconditioning corresponds to the solution of an approximate variational problem of the form

$$\text{Find } \tilde{u} \in V \text{ such that } b(\tilde{u}, v) = f(v) \quad \forall\, v \in V \tag{14.18b}$$

with $b(\cdot,\cdot)$ from (14.18a). This may also be written in the form

$$\text{Find } \tilde{u} \in V \text{ such that } a(P_{\text{AS}}^{-1}\tilde{u}, v) = f(v) \quad \forall\, v \in V \tag{14.19a}$$

where P_{AS} is defined as the projection-like operator $P_{\text{AS}}\colon V \to V$ (cf. Remark 14.1) with matrix representation

$$P_{\text{AS}} \quad \leftrightsquigarrow \quad P'_{\text{AS}} = B^{-1}A = \hat{B}A = \sum_{i=0}^{N} \hat{B}_i A \tag{14.19b}$$

with $\hat{B}_i = E_i A_i^{-1} E_i^{\mathrm{T}}$ defined in (14.17a). This corresponds to writing $A P_{\text{AS}}'^{-1} \tilde{u}' = f'$ instead of $B\tilde{u}' = f'$.

The operator P_{AS} can also be defined in the following way equivalent to (14.19a),

$$b(P_{\text{AS}} u, v) = a(u,v) \quad \text{for all } u, v \in V \tag{14.20}$$

We also define the individual projection-like operators $P_i\colon V \to V_i$ with matrix representation

$$P_i \quad \leftrightsquigarrow \quad P'_i = \hat{B}_i A = E_i A_i^{-1} E_i^{\mathrm{T}} A, \quad i = 0 \ldots N \tag{14.21}$$

such that

$$P_{\text{AS}} = \sum_{i=0}^{N} P_i \tag{14.22}$$

Consider arbitrary $u = X u' \in V$ and $v_i = X v_i' \in V_i$ and observe $P_i' u' = E_i (A_i^{-1} E_i^{\mathrm{T}} A) u'$, i.e., $(A_i^{-1} E_i^{\mathrm{T}} A) u'$ is the representation of $P_i' u'$ in the local coordinate system in V_i (cf. (14.14a)). This gives

$$a_i(P_i u, v_i) = (A_i A_i^{-1} E_i^{\mathrm{T}} A u', v_i'') = (A u', E_i v_i'') = (A u', v_i') = a(u, v_i)$$

which yields the fundamental identity analogous to (14.20),

$$a_i(P_i u, v_i) = a(u, v_i) \quad \text{for all } u \in V,\ v_i \in V_i \tag{14.23}$$

Lemma 14.2 *The operators P_i, $i = 0 \ldots N$, and P_{AS} are selfadjoint with respect to $a(\cdot,\cdot)$.*

Proof: Let $u, v \in V$. Using the symmetry of $a(\cdot,\cdot)$, $a_i(\cdot,\cdot)$, and the fact that $P_i u$, $P_i v \in V_i$ we obtain, making use of (14.23),[73]

$$a(P_i u, v) = a(v, P_i u) = a_i(P_i v, P_i u) = a_i(P_i u, P_i v) = a(u, P_i v)$$

For P_{AS} the result follows by summation. □

[73] In matrix formulation: $(P_i')^A = A^{-1}(\hat{B}_i A)^T A = A^{-1} A \hat{B}_i A = P_i'$ (analogously as in Remark 14.1).

Lemma 14.3 *The operators P_i, $i = 0 \ldots N$ and P_{AS} are coercive (positive definite) with respect to $a(\cdot,\cdot)$, i.e.,*

$$a(P_i u, u) > 0 \quad \text{for all} \quad 0 \neq u \in V \tag{14.24a}$$

and they satisfy the Cauchy-Schwarz inequality

$$a(P_i u, v) \leq \big(a(P_i u, u)\big)^{\frac{1}{2}} \big(a(P_i v, v)\big)^{\frac{1}{2}} \tag{14.24b}$$

Proof: From the definition (14.21) of the P_i we have

$$a(P_i u, u) = (A E_i A_i^{-1} E_i^{\mathrm{T}} A u', u') = (A_i^{-1} E_i^{\mathrm{T}} A u', E_i^{\mathrm{T}} A u') > 0$$

since the A_i are SPD. Together with Lemma 14.2 this shows that $a(P_i u, v)$ is a symmetric and coercive bilinear form on V, which implies (14.24b). □

Lemma 14.1 shows that B is SPD. Our goal is to find constants γ, Γ such that the estimate

$$\gamma B \leq A \leq \Gamma B \qquad \ldots \quad \text{'spectral equivalence'}$$

is valid. If these constants are available then, as in Exercise 12.4, we infer for the matrix $P'_{\mathrm{AS}} = B^{-1}A$:

$$\kappa = \kappa_\sigma(P'_{\mathrm{AS}}) = \frac{\lambda_{max}(P'_{\mathrm{AS}})}{\lambda_{min}(P'_{\mathrm{AS}})} \leq \frac{\Gamma}{\gamma} \tag{14.25}$$

In view of Exercise 12.4, the quantity κ allows us to assess the convergence behavior of the corresponding preconditioned CG method (PCG). The goal is the design of preconditioners B that are cheap (i.e., $r \mapsto B^{-1} r$ is simple to evaluate), while the ratio Γ/γ is as small as possible.

Example 14.1 Let $A \in \mathbb{R}^{N\times N}$ be SPD. Denote by e_i, $i = 1 \ldots N$, the Euclidean unit vectors in $\mathbb{R}^N$. Let $V = \mathbb{R}^N$, $V_i = \mathrm{span}\{e_i\}$, and let the bilinear forms $a_i(\cdot,\cdot)$ be obtained by restricting $a(\cdot,\cdot)$ to the one-dimensional spaces V_i, i.e., $a_i(u,v) = a(u,v)$ for $u, v \in V_i$. The corresponding subspace solution $u_i \in V_i = \mathrm{span}\{e_i\}$ is the solution of (14.12b),

$$a_i(u_i, v_i) = f(v_i) \quad \forall\, v \in V_i$$

Each single correction affects the i-th solution component only, and all these corrections are added up. In matrix terminology (cf. Lemma 14.1) we have, with respect to the Euclidean basis,

$$E_i = \big(0, \ldots, 0, 1, 0, \ldots, 0\big)^{\mathrm{T}}, \quad A_i = \big(a(e_i, e_i)\big) = \big(a_{i,i}\big)$$

From (14.17a) we obtain

$$B^{-1} = \mathrm{diag}\big(a_{1,1}^{-1}, \ldots, a_{N,N}^{-1}\big)$$

The corresponding AS preconditioner is precisely the Jacobi preconditioner, i.e., the preconditioning matrix B is the diagonal $D = \mathrm{diag}(A)$ of the stiffness matrix A.

In FEM terminology, for the Poisson equation with Dirichlet boundary conditions, discretized over mesh points x_i, this means that each single correction $u_i \in V_i$ is a multiple of the i-th basis function (hat function) over the local patch Ω_i around x_i, and u_i is the solution of a local discrete Poisson problem with *homogeneous* Dirichlet boundary condition on $\partial\Omega_i$. All these local solutions added up give rise to the value of the Jacobi preconditioner. In terms of degrees of freedom, this is a 'non-overlapping' method; however, from a geometric point of view, the local subdomains (patches) Ω_i are of course overlapping.

In this interpretation, Jacobi is a simple but ineffective example of a domain decomposition technique. More advanced techniques for domain decomposition are considered in Sec. 14.4. ∎

This example shows that AS is a generalization of the classical Jacobi preconditioner in the sense: 'Compute the actual residual, choose a family of subspaces, compute the Galerkin correction in each individual subspace, and sum up the corrections'. In general we also allow that the subproblems are only approximately solved and that the subspaces may be overlapping.

Exercise 14.1 Provide an interpretation of Jacobi line relaxation (block relaxation, where the blocks of variables are associated with grid lines of a regular mesh) in the spirit of Example 14.1. ∎

Abstract theory for AS.

In the following we characterize the AS preconditioner B by three parameters: C_0, $\rho(\mathcal{E})$, and ω, which enter via assumptions on the subspaces V_i and the bilinear forms $a_i(\cdot,\cdot)$ representing the approximate local problems.

Assumption 14.1 (stable decomposition)
There exists a constant $C_0 > 0$ such that every $u \in V$ admits a decomposition $u = \sum_{i=0}^{N} u_i$ with $u_i \in V_i$ such that

$$\sum_{i=0}^{N} a_i(u_i, u_i) \le C_0^2 \, a(u,u) \tag{14.26}$$

Assumption 14.2 (strengthened Cauchy-Schwarz inequality)
For $i, j = 1 \dots N$, let $\mathcal{E}_{i,j} = \mathcal{E}_{j,i} \in [0,1]$ be defined by the inequalities

$$|a(u_i, u_j)|^2 \le \mathcal{E}_{i,j}^2 \, a(u_i,u_i)\, a(u_j,u_j) \quad \forall\ u_i \in V_i,\ u_j \in V_j \tag{14.27}$$

By $\rho(\mathcal{E})$ we denote the spectral radius of the symmetric matrix $\mathcal{E} = (\mathcal{E}_{i,j}) \in \mathbb{R}^{N\times N}$. The particular assumption is that we have a nontrivial bound for $\rho(\mathcal{E})$ to our disposal.

Note that due to $\mathcal{E}_{i,j} \le 1$ (Cauchy-Schwarz inequality), the trivial bound $\rho(\mathcal{E}) = \|\mathcal{E}\|_2 \le \sqrt{\|\mathcal{E}\|_1 \, \|\mathcal{E}\|_\infty} \le N$ always holds. For particular Schwarz-type methods one aims at bounds for $\rho(\mathcal{E})$ which are *independent of* N.

Assumption 14.3 (local stability)
There exists $\omega > 0$ such that for all $i = 1 \dots N$:

$$a(u_i, u_i) \le \omega \, a_i(u_i, u_i) \quad \forall\ u_i \in V_i \tag{14.28}$$

Remark 14.2 The space V_0 is not included in the definition of $\mathcal{E}$; as we will see below, this space is allowed to play a special role.

$\mathcal{E}_{i,j} = 0$ means that the spaces V_i and V_j are orthogonal (in the $a(\cdot,\cdot)$-inner product). We will see below that small $\rho(\mathcal{E})$ is desirable. We will also see that a small C_0 is desirable.

The parameter ω represents a one-sided measure of the approximation properties of the inexact solvers a_i. If the local solver is of exact Galerkin type, i.e, $a_i(u,v) \equiv a(u,v)$ for $u, v \in V_i$, then $\omega = 1$. However, this does not necessarily imply that Assumptions 14.1 and 14.2 are also satisfied. ∎

Lemma 14.4 *[P. L. Lions]*

Let P_{AS} be defined by (14.19b) *resp.* (14.20)*. Then, under Assumption 14.1,*

(i) $P_{AS}\colon V \to V$ is a bijection, and

$$a(u,u) \le C_0^2\, a(P_{AS}u,u) = C_0^2 \sum_{i=0}^{N} a(P_i u,u) \quad \forall\ u \in V \tag{14.29}$$

(ii) Characterization of $b(u,u)$:

$$b(u,u) = a(P_{AS}^{-1}u,u) = \min\Big\{ \sum_{i=0}^{N} a_i(u_i,u_i)\colon\ u = \sum_{i=0}^{N} u_i,\ \ u_i \in V_i \Big\} \tag{14.30}$$

Proof: We make use of (14.22), the fundamental identity (14.23), and Cauchy-Schwarz inequalites.

(i): Let $u \in V$ and $u = \sum_i u_i$ be a decomposition of stable type guaranteed by Assumption 14.1. Then, by means of the fundamental identity (14.23) and the Cauchy-Schwarz inequality we obtain

$$\begin{aligned} a(u,u) &= a(u,\textstyle\sum_i u_i) = \sum_i a(u,u_i) = \sum_i a_i(P_i u,u_i) \le \sum_i \sqrt{a_i(P_i u,P_i u)\, a_i(u_i,u_i)} \\ &= \textstyle\sum_i \sqrt{a(u,P_i u)\, a_i(u_i,u_i)} \le \sqrt{\sum_i a(u,P_i u)}\,\sqrt{\sum_i a_i(u_i,u_i)} \\ &= \sqrt{a(u,P_{\text{AS}}u)}\,\sqrt{\textstyle\sum_i a_i(u_i,u_i)} \le \sqrt{a(u,P_{\text{AS}}u)}\, C_0 \sqrt{a(u,u)} \end{aligned}$$

This implies the estimate (14.29). In particular, it follows that P_{AS} is injective, because with (14.29), $P_{\text{AS}}u = 0$ implies $a(u,u) = 0$, hence $u = 0$. Due to finite dimension, we conclude that P_{AS} is bijective.

(ii): We first show that the minimum on the right-hand side of (14.30) cannot be smaller than $a(P_{\text{AS}}^{-1}u,u)$. To this end, we consider an arbitrary decomposition $u = \sum_i u_i$ with $u_i \in V_i$ and estimate

$$\begin{aligned} a(P_{\text{AS}}^{-1}u,u) &= \textstyle\sum_i a(P_{\text{AS}}^{-1}u,u_i) = \sum_i a_i(P_i P_{\text{AS}}^{-1}u,u_i) \\ &\le \sqrt{\textstyle\sum_i a_i(P_i P_{\text{AS}}^{-1}u,P_i P_{\text{AS}}^{-1}u)}\,\sqrt{\textstyle\sum_i a_i(u_i,u_i)} \\ &= \sqrt{\textstyle\sum_i a(P_{\text{AS}}^{-1}u,P_i P_{\text{AS}}^{-1}u)}\,\sqrt{\textstyle\sum_i a_i(u_i,u_i)} \\ &= \sqrt{a\big(P_{\text{AS}}^{-1}u,\textstyle\sum_i P_i P_{\text{AS}}^{-1}u\big)}\,\sqrt{\textstyle\sum_i a_i(u_i,u_i)} = \sqrt{a(P_{\text{AS}}^{-1}u,u)}\,\sqrt{\textstyle\sum_i a_i(u_i,u_i)} \end{aligned}$$

(Here we have used the identity $\sum_i P_i P_{\text{AS}}^{-1} = I$, see (14.22)). This shows $a(P_{\text{AS}}^{-1}u,u) \le \sum_i a_i(u_i,u_i)$. In order to see that $a(P_{\text{AS}}^{-1}u,u)$ is indeed the minimum of the right-hand side of (14.30), we define $u_i = P_i P_{\text{AS}}^{-1}u$. Then, $u_i \in V_i$ and $\sum_i u_i = u$, and

$$\begin{aligned} \textstyle\sum_i a_i(u_i,u_i) &= \textstyle\sum_i a_i(P_i P_{\text{AS}}^{-1}u,P_i P_{\text{AS}}^{-1}u) = \sum_i a(P_{\text{AS}}^{-1}u,P_i P_{\text{AS}}^{-1}u) \\ &= a(P_{\text{AS}}^{-1}u,\textstyle\sum_i P_i P_{\text{AS}}^{-1}u) = a(P_{\text{AS}}^{-1}u,u) \end{aligned}$$

We see that the minimum is attained for these special u_i, and the value of the minimum is $a(P_{\text{AS}}^{-1}u,u)$. This concludes the proof. □

The matrix $P'_{\text{AS}} = B^{-1}A$ from (14.19b) is the matrix representation of the operator P_{AS}. Since P_{AS} is selfadjoint in the A-inner product (see Lemma 14.2), we can estimate the smallest and the largest eigenvalue of $B^{-1}A$ by

$$\lambda_{min}(B^{-1}A) = \inf_{0\neq u\in V} \frac{a(P_{\text{AS}}u,u)}{a(u,u)}, \qquad \lambda_{max}(B^{-1}A) = \sup_{0\neq u\in V} \frac{a(P_{\text{AS}}u,u)}{a(u,u)} \tag{14.31}$$

Lemma 14.4, (i) in conjunction with Assumption 14.1 readily yields the lower bound

$$\lambda_{min}(B^{-1}A) \ge \frac{1}{C_0^2}$$

An upper bound for $\lambda_{max}(B^{-1}A)$ is obtained with the help of the following lemma.

Lemma 14.5 *Under Assumptions 14.2 and 14.3 we have*

$$\|P_i\|_A \leq \omega, \quad i = 0 \ldots N \tag{14.32a}$$

$$a(P_{AS} u, u) \leq \omega\,(1 + \rho(\mathcal{E}))\, a(u, u) \quad \textit{for all} \quad u \in V \tag{14.32b}$$

Proof:

– Proof of (14.32a): From Assumption 14.3 (inequality (14.28)) and again using the fundamental inequality (14.23) we infer for all $u \in V$:

$$\|P_i u\|_A^2 = a(P_i u, P_i u) \leq \omega\, a_i(P_i u, P_i u) = \omega\, a(u, P_i u) \leq \omega\, \|u\|_A \, \|P_i u\|_A \tag{14.33}$$

which implies (14.32a).

– For the proof of (14.32b), we observe that the space V_0 is assumed to play a special role. We define

$$\hat{P} = \sum_{i=1}^{N} P_i = P_{\text{AS}} - P_0 \tag{14.34}$$

Assumptions 14.2 and 14.3 then allows us to bound[74]

$$\begin{aligned} a(\hat{P} u, \hat{P} u) &= \sum_{i,j=1}^{N} a(P_i u, P_j u) \leq \sum_{i,j=1}^{N} \mathcal{E}_{i,j} \sqrt{a(P_i u, P_i u)} \sqrt{a(P_j u, P_j u)} \\ &\leq \omega \sum_{i,j=1}^{N} \mathcal{E}_{i,j} \sqrt{a_i(P_i u, P_i u)} \sqrt{a_j(P_j u, P_j u)} \leq \omega\, \rho(\mathcal{E}) \sum_{i=1}^{N} a_i(P_i u, P_i u) \\ &= \omega\, \rho(\mathcal{E}) \sum_{i=1}^{N} a(u, P_i u) = \omega\, \rho(\mathcal{E})\, a(u, \hat{P} u) \leq \omega\, \rho(\mathcal{E})\, \|u\|_A \, \|\hat{P} u\|_A \end{aligned} \tag{14.35}$$

from which we extract

$$\|\hat{P} u\|_A \leq \omega\, \rho(\mathcal{E})\, \|u\|_A \tag{14.36}$$

From (14.32a) we have $\|P_0 u\|_A \leq \omega \|u\|_A$. Combining this with (14.36) gives

$$a(P_{\text{AS}} u, u) = a(\hat{P} u, u) + a(P_0 u, u) \leq \|\hat{P} u\|_A \, \|u\|_A + \|P_0 u\|_A \, \|u\|_A \leq \omega\,(1 + \rho(\mathcal{E}))\, \|u\|_A^2$$

which is the desired estimate. □

Theorem 14.1 *Let C_0, ω, $\rho(\mathcal{E})$ be defined by Assumptions 14.1–14.3. Then,*

$$\lambda_{min}(P'_{AS}) \geq \frac{1}{C_0^2} \quad \textit{and} \quad \lambda_{max}(P'_{AS}) \leq \omega\,(1 + \rho(\mathcal{E}))$$

Proof: Follows from Lemmas 14.4 and 14.5 in conjunction with (14.31). □

Theorem 14.1 permits us to conclude spectral equivalence, i.e., we obtain a bound the spectral condition number of $B^{-1}A \leftrightsquigarrow P_{\text{AS}}$ (cf. (14.25)) in terms of the parameters C_0, ω and $\rho(\mathcal{E})$,

$$\kappa = \kappa_\sigma(P'_{\text{AS}}) \leq C_0^2\, \omega\,(1 + \rho(\mathcal{E})) \tag{14.37}$$

[74] In the second line of (14.35), a quadratic form in terms of $\mathcal{E}$ is estimated from above in terms of $\|\mathcal{E}\|_2$. Note that $\mathcal{E}$ is symmetric, such that $\|\mathcal{E}\|_2 = \rho(\mathcal{E})$.

The error amplification operator associated with the AS preconditioner (in the sense of one step of a stationary iteration scheme) is given by

$$G_{\text{AS}} := I - P_{\text{AS}} \quad \leftrightsquigarrow \quad I - P'_{\text{AS}} = I - B^{-1}A$$

Note that the above results to not imply $\|G_{\text{AS}}\|_A < 1$, i.e., AS as a stationary iteration is not guaranteed to be convergent under the above assumptions.

Example 14.2 (cf. Example 14.1) Let $A \in \mathbb{R}^{(N-1)\times(N-1)}$ be the SPD stiffness matrix for the 1D Poisson model problem,

$$A = \frac{1}{h}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad h = \frac{1}{N}$$

and

$$B = \operatorname{diag}(A) = \tfrac{1}{h}\operatorname{diag}\big(2,\ldots,2\big)$$

corresponding to the Jacobi preconditioner.

Denote by e_i, $i = 1\ldots N-1$, the Euclidean unit vectors in $\mathbb{R}^{N-1}$. Let $V = \mathbb{R}^{N-1}$, $V_0 = \{\}$ (!), and $V_i = \operatorname{span}\{e_i\}$, $i = 1\ldots N-1$. The bilinear form associated with A is

$$a(u,u) = (Au,u)$$

The bilinear forms $a_i(u,v) = (A_i u, v)$ are the restrictions of $a(\cdot,\cdot)$ to the (one-dimensional) spaces $V_i = \operatorname{span}\{e_i\}$. With

$$E_i = \big(0,\ldots,0,1,0,\ldots,0\big)^{\mathrm{T}}$$

we have

$$A_i = E_i^{\mathrm{T}} A E_i = \big(a_{i,i}\big) = \Big(\frac{2}{h}\Big)$$

Let us check Assumptions 14.1–14.3:

1. Stable decomposition: For $u = (u_1,\ldots,u_{N-1})^{\mathrm{T}} = \sum_{i=1}^{N-1} u_i e_i$ we have

$$\sum_{i=0}^{N-1} a_i(u_i e_i, u_i e_i) = \frac{2}{h}\sum_{i=0}^{N-1} u_i^2 = (Bu,u) = \frac{2}{h}(u,u) \le \frac{2}{h}\frac{1}{\lambda_{min}(A)}(Au,u)$$

   with $\lambda_{min}(A) = \frac{4}{h}\sin^2(\pi h) \approx 4\pi^2 h$. Thus, Assumption 14.1 is satisfied with the h-dependent constant

$$C_0^2 \doteq \frac{1}{2\pi^2 h^2}$$

   or equivalently

$$C_0^2 = \frac{1}{\lambda_{min}(B^{-1}A)}$$

2. Strengthened Cauchy-Schwarz inequality: With

$$a(u_i e_i, u_j e_j) = u_i u_j (Ae_i, e_j) = a_{i,j} u_i u_j$$

   and $a(u_i e_i, u_i e_i) = a_{i,i} u_i^2$ we have

$$|a(u_i e_i, u_j e_j)|^2 \le \mathcal{E}_{i,j}^2\, a(u_i e_i, u_i e_i)\, a(u_j e_j, u_j e_j) \quad \text{with} \quad \mathcal{E}_{i,j}^2 = \frac{|a_{i,j}|^2}{a_{i,i}\, a_{j,j}}$$

In particular, $\mathcal{E}_{i,i} = 1$, $\mathcal{E}_{i,j} = \frac{1}{2}$ for $|i-j| = 1$, and $\mathcal{E}_{i,j} = 0$ for $|i-j| > 1$, which gives

$$\rho(\mathcal{E}) \le 2 \quad \text{independent of} \quad h$$

3. Local stability: Due to

$$a(u_i e_i, u_i e_i) = u_i^2\,(A e_i, e_i) = u_i^2\,(B e_i, e_i) = u_i^2\, \tfrac{2}{h} = a_i(u_i e_i, u_i e_i)$$

the local stability estimate (14.28) is satisfied with stability constant $\omega = 1$.

Now, application of Theorem 14.1 shows that for the Jacobi preconditioner we have

$$\lambda_{min}(P'_{\text{AS}}) = \lambda_{min}(B^{-1}A) \gtrsim 2\pi^2 h^2 \quad \text{and} \quad \lambda_{max}(P'_{\text{AS}}) = \lambda_{max}(B^{-1}A) \le 1 \cdot (1+2) = 3$$

In this simple example, these parameters can of course also be computed directly:

$$\lambda_{min}(B^{-1}A) = \frac{h}{2}\,\lambda_{min}(A) \doteq 2\pi^2 h^2 \quad \text{and} \quad \lambda_{max}(B^{-1}A) = \frac{h}{2}\,\lambda_{max}(A) < 2$$

We see that

$$\kappa_\sigma(B^{-1}A) = \frac{\lambda_{max}(B^{-1}A)}{\lambda_{min}(B^{-1}A)} = \mathcal{O}(h^{-2})$$

and therefore the Jacobi preconditioner has no effect for the Poisson problem. This is also obvious due to the fact that the diagonal of A is constant. In other words: Jacobi is equivalent to a 'naive' substructuring method which does not yield a useful preconditioner. ∎

14.3 Multiplicative Schwarz (MS) methods

In multiplicative versions of Schwarz methods, subspace corrections are immediately applied, and the new residual is evaluated before proceeding – analogously as in the Gauss-Seidel method compared to Jacobi. With the same denotation as for AS methods, this amounts to the following procedure, starting from some initial u_0 with (weak) residual $r_0(\cdot) = f(\cdot) - a(u_0, \cdot)$:

$$\begin{aligned} &\text{Find } \delta_0 \in V_0 \text{ such that } a_0(\delta_0, v_0) = r_0(v_0) \quad \forall\, v_0 \in V_0 \\ &\text{Update approximation: } u_{0,0} = u_0 + \delta_0 \\ &\text{Update residual: } r_{0,0}(\cdot) = f(\cdot) - a(u_{0,0}, \cdot) = r_0(\cdot) - a(\delta_0, \cdot) \end{aligned} \tag{14.38a}$$

and continuing this over the subspaces $V_1, V_2, \ldots$: For $i = 1, 2, \ldots$,

$$\begin{aligned} &\text{Find } \delta_i \in V_i \text{ such that } a_i(\delta_i, v_i) = r_{0,i-1}(v_i) \quad \forall\, v_i \in V_i \\ &\text{Update approximation: } u_{0,i} = u_{0,i-1} + \delta_i \\ &\text{Update residual: } r_{0,i}(\cdot) = f(\cdot) - a(u_{0,i}, \cdot) = r_{0,i-1}(\cdot) - a(\delta_i, \cdot) \end{aligned} \tag{14.38b}$$

Remark 14.3 This procedure is analogous to, or rather a generalization of a Gauss-Seidel sweep, or preconditioner, applied to a linear system $Au = f$ starting from some initial u_0 with residual r_0. For a better understanding, let us reformulate such a Gauss-Seidel sweep in the following way equivalent to (5.8b). We start with[75]

$$u_0 = (u_0^1, \ldots, u_0^n)^{\mathrm{T}}$$

and associated residual

$$r_0 = f - A u_0 = (r_0^1, \ldots, r_0^n)^{\mathrm{T}}$$

[75] We use conventional matrix/vector notation and indexing beginning with 1 instead of 0; however, vector indices are denoted by superscripts for the moment.

1. Solve
$$a_{11}\,\delta^1 = r_0^1$$
and update u_0,
$$u_{0,1} = u_0 + (\delta^1, 0, \ldots, 0)^{\mathrm{T}}$$
and update the residual,
$$r_{0,1} = f - A\,u_{0,1} = r_0 - \begin{pmatrix} a_{11}\,\delta^1 \\ a_{21}\,\delta^1 \\ \vdots \\ a_{n1}\,\delta^1 \end{pmatrix} = \begin{pmatrix} r_{0,1}^1 \\ \underline{r_{0,1}^2} \\ \vdots \\ r_{0,1}^n \end{pmatrix}$$

2. Solve
$$a_{22}\,\delta^2 = \underline{r_{0,1}^2}$$
and update $u_{0,1}$,
$$u_{0,2} = u_{0,1} + (0, \delta^2, 0, \ldots, 0)^{\mathrm{T}} = u_0 + (\delta^1, \delta^2, 0, \ldots, 0)^{\mathrm{T}}$$
and update the residual,
$$r_{0,2} = f - A\,u_{0,2} = r_{0,1} - \begin{pmatrix} a_{12}\,\delta^2 \\ a_{22}\,\delta^2 \\ a_{32}\,\delta^2 \\ \vdots \\ a_{n2}\,\delta^2 \end{pmatrix} = \begin{pmatrix} r_{0,2}^1 \\ r_{0,2}^2 \\ \underline{\underline{r_{0,2}^3}} \\ \vdots \\ r_{0,2}^n \end{pmatrix}$$

3. Solve
$$a_{33}\,\delta^3 = \underline{\underline{r_{0,2}^3}}$$
and update $u_{0,2}$,
$$u_{0,3} = u_{0,2} + (0, 0, \delta^3, 0, \ldots, 0)^{\mathrm{T}} = u_0 + (\delta^1, \delta^2, \delta^3, 0, \ldots, 0)^{\mathrm{T}}$$
and update the residual,
$$r_{0,3} = f - A\,u_{0,3} = r_{0,2} - \begin{pmatrix} a_{13}\,\delta^3 \\ a_{23}\,\delta^3 \\ a_{33}\,\delta^3 \\ \vdots \\ a_{n3}\,\delta^3 \end{pmatrix} = \begin{pmatrix} r_{0,3}^1 \\ r_{0,3}^2 \\ \vdots \\ \vdots \\ r_{0,3}^n \end{pmatrix}$$

and so on. Formally, the full residuals $r_{0,i} = f - A\,u_{0,i}$ are involved in each step, but only the $\underline{\underline{\text{underlined}}}$ residual components enter the local (componentwise) corrections. ∎

Turning back to the general MS scheme in coordinate representation we obtain the following iteration for $i = 0, 1, \ldots$. As for AS, for simplicity of notation we formulate the 'subspace solution variant'[76] analogous to the solution of $(L + D)\,u = f$ with Gauss-Seidel preconditioner $L + D$.

[76] If MS is used as a preconditioner, f is again replaced by the residual r_0 of a given u_0, and the result of an MS sweep is the preconditioned residual; see (14.38).

Let $u'_{0,-1} = 0,\ r'_{0,-1} := f'$. For $i = 0, 1, 2, \ldots,$

$$\begin{aligned}
&\text{Solve} \quad A_i\,\delta''_i = E_i^{\mathrm{T}} r'_{0,i-1} \\
&\text{Let} \quad \delta'_i = E_i\,\delta''_i, \quad \text{update} \quad u'_{0,i} = u'_{0,i-1} + \delta'_i \\
&\text{Update the residual:} \quad r'_{0,i} = r'_{0,i-1} - A\,\delta'_i
\end{aligned}$$

After $N+1$ steps, we see that the new approximation $u'_1 = u'_{0,N}$ is given by

$$u'_1 = u'_0 + \sum_{i=0}^{N} \delta'_i$$

with

$$\begin{aligned}
\delta'_i &= E_i\,A_i^{-1} E_i^{\mathrm{T}} r'_{0,i-1}, \qquad i = 0, 1, \ldots \\
r'_{0,i} &= r'_{0,i-1} - A\,\delta'_i, \qquad i = 0, 1, \ldots
\end{aligned}$$

i.e.,

$$\begin{aligned}
\delta'_0 &= E_0\,A_0^{-1} E_0^{\mathrm{T}} r'_0 \\
\delta'_1 &= E_1\,A_1^{-1} E_1^{\mathrm{T}} r'_{0,1} \\
&= E_1\,A_1^{-1} E_1^{\mathrm{T}} (r'_0 - A\,\delta'_0) \\
&= E_1\,A_1^{-1} E_1^{\mathrm{T}} (I - E_0\,A_0^{-1} E_0^{\mathrm{T}} A)\, r'_0 \\
&\ldots
\end{aligned}$$

With $P'_i = \hat{B}_i\,A = E_i\,A_i^{-1} E_i^{\mathrm{T}} A$ (the matrix representation of the projection-like operators (14.21)) an induction argument leads us to the representation

$$\delta'_i = \hat{B}_i\,(I - P'_i) \cdots (I - P'_0)\, r'_0$$

For the successive residuals we obtain, with $Q'_i = A\,E_i\,A_i^{-1} E_i^{\mathrm{T}}$:

$$\begin{aligned}
r'_{0,0} &= (I - Q'_0)\, r'_0 \\
r'_{0,1} &= (I - Q'_1)\, r'_{0,0} = (I - Q'_1)(I - Q'_0)\, r'_0 \\
&\ldots
\end{aligned}$$

After a complete sweep over all subspaces V_i, we end up with

$$r'_1 = r'_{0,N} = (I - Q'_N) \cdots (I - Q'_0)\, r'_0$$

For the error $e'_1 = -A\,r'_1$ we obtain[77]

$$e'_1 = (I - P'_N) \cdots (I - P'_0)\, e'_0 =: G'_{\mathrm{MS}}\, e'_0$$

which motivates the name 'multiplicative method'. The operator $G_{\mathrm{MS}} \colon V \to V$,

$$G_{\mathrm{MS}} = (I - P_N) \cdots (I - P_0), \quad \text{with} \quad P_i \quad \leftrightsquigarrow \quad P'_i = \hat{B}_i\,A = E_i\,A_i^{-1} E_i^{\mathrm{T}} A \tag{14.39}$$

represents error amplification operator of one sweep of MS.

The abstract theory from the preceding section can be extended to the MS case. In particular, repeated application of MS sweeps yields a convergent stationary iteration scheme, provided the parameter ω from Assumption 14.3 satisfies $\omega \in (0, 2)$. The following Theorem 14.2 can be interpreted as s generalization of Theorem 5.3 on the convergence of the SOR scheme.

[77] Note that $A^{-1}(I - Q'_N) \cdots (I - Q'_0)A = (I - P'_N) \cdots (I - P'_0)$ – an exercise.

Lemma 14.6 *Let*

$$G_{-1} = I \quad \textit{and} \quad G_j := (I - P_j) \cdots (I - P_0), \quad j = 0 \ldots N$$

Then,

$$I - G_j = \sum_{i=0}^{j} P_i G_{i-1}, \quad j = 1 \ldots N \tag{14.40a}$$

and for $\omega \in (0, 2)$,

$$(2 - \omega) \sum_{i=0}^{N} a(P_i\, G_{i-1}\, u, G_{i-1}\, u) \leq \|u\|_A^2 - \|G_{MS} u\|_A^2 \quad \forall\, u \in V \tag{14.40b}$$

Proof: Due to

$$G_i = G_{i-1} - P_i\, G_{i-1} \tag{14.41}$$

the sum in (14.40a) is telescoping, and the result immediately follows.

– In order to prove (14.40b) we proceed as follows: Using (14.41) and Lemma 14.2 (A-selfadjointness of the P_i), a short calculation shows

$$\begin{aligned}
\|G_{i-1} u\|_A^2 - \|G_i u\|_A^2 &= a(G_{i-1} u, G_{i-1} u) - a(G_i u, G_i u) \\
&= a(G_{i-1} u, G_{i-1} u) - a(G_{i-1} u - P_i\, G_{i-1} u, G_{i-1} u - P_i\, G_{i-1} u) \\
&= 2\, a(P_i\, G_{i-1} u, G_{i-1} u) - a(P_i\, G_{i-1} u, P_i\, G_{i-1} u) \\
&\geq (2 - \omega)\, a(P_i\, G_{i-1} u, G_{i-1} u)
\end{aligned}$$

where the last inequality follows from Assumption 14.3 in combination with the fundamental inequality (14.23). Summing up these inequalities, with $G_N = G_{\text{MS}}$, gives (14.40b). □

Theorem 14.2 *Let* C_0, ω, $\rho(\mathcal{E})$ *be defined by Assumptions 14.1–14.3, and suppose* $\omega \in (0, 2)$. *Then, the MS error amplification operator* (14.39) *satisfies*[78]

$$\|G_{MS}\|_A^2 \leq 1 - \frac{2 - \omega}{C_0^2 \left(1 + \omega^2 \rho(\mathcal{E})\right)^2} < 1 \tag{14.42}$$

Proof: We use the denotation and the assertions from Lemma 14.6.

Estimate (14.42) is equivalent to

$$\|u\|_A^2 \leq \frac{1}{\delta} \left(\|u\|_A^2 - \|G_{\text{MS}}\, u\|_A^2 \right) \quad \forall\, u \in V, \quad \text{with} \quad \delta = \frac{2 - \omega}{C_0^2 \left(1 + \omega^2 \rho(\mathcal{E})\right)^2}$$

In view of Lemma 14.6, (14.40b), it is sufficient to show (again, $j = 0$ plays a special role)

$$\|u\|_A^2 = a(u, u) \leq \underbrace{\frac{2 - \omega}{\delta}}_{= C_0^2 (1 + \omega^2 \rho(\mathcal{E}))^2} \left\{ a(P_0\, u, u) + \sum_{j=1}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \right\} \quad \forall\, u \in V$$

Due to Lemma 14.4, (14.29), it thus remains to show that

[78] The right-hand side of (14.42) can be forced to be < 1 by choosing C_0 in Assumption 14.1 large enough. The 'critical' condition $\omega \in (0, 2)$ originates from Lemma 14.6; it is reminiscent of the condition on the relaxation parameter in the SOR method.

$$C_0^2\, a(P_0\, u, u) + C_0^2 \sum_{j=1}^{N} a(P_j\, u, u) \le C_0^2 \left(1 + \omega^2 \rho(\mathcal{E})\right)^2 \left\{ a(P_0\, u, u) + \sum_{j=0}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \right\} \quad \forall\, u \in V$$

or equivalently,

$$\sum_{j=1}^{N} a(P_j\, u, u) \le \left(2\omega^2 \rho(\mathcal{E}) + \omega^4 \rho(\mathcal{E})^2\right) a(P_0\, u, u) + \left(1 + \omega^2 \rho(\mathcal{E})\right)^2 \sum_{j=1}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \quad \forall\, u \in V \tag{14.43}$$

In order to show (14.43), we rewrite the terms on the left-hand side using Lemma 14.6, (14.40a),

$$\begin{aligned} a(P_j\, u, u) &= a(P_j\, u, G_{j-1}\, u) + a(P_j\, u, (I - G_{j-1})\, u) \\ &= \underbrace{(P_j\, u, G_{j-1}\, u)}_{\text{(i)}} + \underbrace{\sum_{i=0}^{j-1} a(P_j\, u, P_i G_{i-1}\, u)}_{\text{(ii)}}, \quad j = 1 \ldots N \end{aligned} \tag{14.44}$$

and estimate the sums of both contributions (i), (ii). For (14.44 (i)), the Cauchy-Schwarz inequality (14.24b) gives

$$\sum_{j=1}^{N} a(P_j\, u, G_{j-1}\, u) \le \left(\sum_{j=1}^{N} a(P_j\, u, u) \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \right)^{\frac{1}{2}} \tag{14.45a}$$

Concerning (14.44 (ii)), isolating the terms with $i = 0$ and estimating the terms with $i > 0$ in the same way as in the proof of Lemma 14.5 gives

$$\begin{aligned} \sum_{j=1}^{N} \sum_{i=0}^{j-1} a(P_j\, u, P_i G_{i-1}\, u) &= \underbrace{\sum_{j=1}^{N} a(P_j\, u, P_0\, u)}_{= a(\hat{P}\, u, P_0\, u) \text{ (see (14.34))}} + \sum_{j=1}^{N} \sum_{i=1}^{j-1} a(P_j\, u, P_i G_{i-1}\, u) \\ &\le a(\hat{P}\, u, P_0\, u) + \omega\, \rho(\mathcal{E}) \left(\sum_{j=1}^{N} a(P_j\, u, u) \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \right)^{\frac{1}{2}} \end{aligned} \tag{14.45b}$$

Combining (14.44) with (14.45) leads to

$$\sum_{j=1}^{N} a(P_j\, u, u) \le a(\hat{P}\, u, P_0\, u) + \left(1 + \omega\, \rho(\mathcal{E})\right) \left(\sum_{j=1}^{N} a(P_j\, u, u) \right)^{\frac{1}{2}} \left(\sum_{j=1}^{N} a(P_j\, G_{j-1}\, u, G_{j-1}\, u) \right)^{\frac{1}{2}}$$

The first term on the right-hand side can be estimated by (see (14.36),(14.33))

$$a(\hat{P}\, u, P_0\, u) \le \|\hat{P}\|_A\, a(P_0\, u, P_0\, u) \le \omega\, \rho(\mathcal{E})\, a(P_0\, u, P_0\, u) \le \omega^2 \rho(\mathcal{E})\, a(P_0\, u, u)$$

and finally, (14.43) follows by applying the arithmetic/geometric mean inequality to the second term and simple rearranging. □

In a sense, Theorem 14.2 is a stronger result as obtained above for the AS case. It also implies a bound for the condition number of $P_{\text{MS}} := I - G_{\text{MS}}$ (the analog of P_{AS} for the AS case): Let $q < 1$ denote the bound for $\|G_{\text{MS}}\|_A$ from Theorem 14.2. Then,

$$\|P_{\text{MS}}\|_A \le 1 + q < 2$$

and P_{MS} is invertible, with

$$\|P_{\text{MS}}^{-1}\|_A = \|(I - G_{\text{MS}})^{-1}\|_A \le \frac{1}{1 - q}$$

hence

$$\kappa_A(P_{\text{MS}}) \leq \frac{2}{1-q}$$

We see that $q \approx 1$ leads to an unfavorable behavior, which is typical for the standard Gauss-Seidel case (see Example 14.3 below).

Note that, since P_{MS} is not symmetric, it cannot be used in a straightforward way as for preconditioning the CG method. A symmetric version can be obtained in the same way as for the symmetric Gauss-Seidel method by repeating the multiplicative correction procedure, starting with V_N, down to V_0.

Example 14.3 We revisit Example 14.2, but consider Gauss-Seidel iteration, which corresponds to MS with the same partitioning. In Example 14.2 we have seen that

$$C_0^2 \doteq \frac{1}{2\,\pi^2\,h^2}\,, \quad \rho(\mathcal{E}) \leq 2, \quad \omega = 1$$

Thus, Theorem 14.2 shows

$$\|G_{\text{MS}}\|_A^2 = \|(L+D)^{-1}A\|_2 = 1 - \mathcal{O}(h^2)$$

and thus,

$$\kappa_A(P_{\text{MS}}) = \mathcal{O}(h^{-2})$$

In spite of this (well-known) 'negative' result (analogous to the Jacobi case), Gauss-Seidel preconditioning usually has a positive effect on the convergence behavior of Krylov solvers, but it deteriorates with increasing dimension similarly as the unpreconditioned version. ∎

Remark 14.4 Line relaxation techniques (block-Jacobi, block-Gauss-Seidel and similar techiques) naturally fit into the AS/MS framework, with appropriate subspaces V_i representing the respective blocking, i.e., the agglomeration of variables.

Multigrid methods can also be formulated in the context of subspace correction methods, based on hierarchical decompositions of the given space V. ∎

Now the question is what type of substructuring yields to an efficient AS or MS preconditioner. It turns out that, in the context of domain decomposition (Sec. 14.4), overlapping domains are advantageous in this respect, but less simple to realize computationally. In any case, including a basic global approximation on a coarse mesh (associated with the subspace V_0) ensures stability.

14.4 Introduction to domain decomposition techniques

The most important practical realization of the abstract idea of additive or multiplicative Schwarz methods is *domain decomposition*. A standard text on this topic is the book [21]. The abstract theory from Sections 14.2 and 14.3 provides a framework which has proven quite helpful in the design and analysis of a number of old and new domain decomposition techniques.

Often, the approximate local solvers are realized in terms of MG cycles, or sometimes special direct solvers (e.g., based on FFT techniques). In this section we discuss two typical domain decomposition techniques for elliptic problems. In particular, we consider the 2D Poisson equation on a domain $\Omega \subseteq \mathbb{R}^2$ as a prominent example.

An overlapping additive two-level method.

Let $\mathcal{T}_h$ be a quasi-uniform mesh (consisting of triangles), with mesh size h, of $\Omega \subset \mathbb{R}^2$. We consider the Dirichlet problem

$$-\Delta u = f \quad \text{on } \Omega, \quad u = 0 \quad \text{on } \partial\Omega \tag{14.46}$$

discretized by FEM with piecewise linear functions from the FEM space $V_h \subseteq H_0^1(\Omega)$, giving rise to a discrete system $A_h u_h' = f_h'$.

To generate a preconditioner for the resulting linear system we employ the AS framework. To fix ideas and to keep the exposition simple, we assume that a second, coarse triangulation $\mathcal{T}_H$ with mesh size H is chosen. The FEM space on this mesh is $V_H \subseteq H_0^1(\Omega)$. Furthermore we assume $V_H \subseteq V_h$, i.e., the mesh lines of $\mathcal{T}_H$ are also mesh lines of $\mathcal{T}_h$.

For simplicity of exposition we assume that the local solvers are exact, i.e., the FEM systems on the subdomains are solved exactly, such that Assumption 14.3 is trivially satisfied with $\omega = 1$.

The space V_H is associated with the special subspace V_0 from our abstract AS setting. Its purpose is to provide an initial, coarse approximation which enables us to start the AS procedure.

We assume that N subdomains Ω_i, $i = 1 \dots N$, of Ω are given which consist of unions of elements (triangles). We set

$$V_i = V_h \cap H_0^1(\Omega_i), \quad i = 1 \dots N$$

We assume that the subdomains Ω_i satisfy:

- $\Omega = \bigcup_{i=1}^N \Omega_i$
- The Ω_i are assumed to be overlapping; in particular, for the AS convergence theory the 'amount of overlap' is an essential parameter.
- Not more than M subdomains are simultaneously overlapping, i.e., $\sup_{i=1\dots N} \left|\{j\colon \Omega_i \cap \Omega_j \neq \emptyset\}\right| \leq M$.
- The fine mesh $\mathcal{T}_h$ is compatible with the decomposition of Ω, giving rise to N local meshes $\mathcal{T}_{h,i}$ on the subdomains Ω_i, such that $V_h = \sum_{i=1}^N V_i$. In particular, the boundaries $\partial\Omega_i$ consist of mesh lines from $\mathcal{T}_h$.

One step of our AS preconditioner amounts to the following procedure:

1. Solve the FEM equations $A_H u_H' = f_H'$ on the coarse mesh $\mathcal{T}_H$, giving $u_H \in V_H$.
2. Embedding u_H into the fine mesh gives rise to $u_0 \in V_0$, where $V_0 \subseteq V_h$ is the space of all interpolants from V_H to V_h (described by the range of the corresponding interpolation operator).
3. For $i = 1 \dots N$:

   Solve the subproblems $A_{h,i} u_{h,i}'' = f_{h,i}''$, where the local stiffness matrices $A_{h,i}$ refer to the local meshes over the subdomains Ω_i, with the appropriately restricted versions $f_{h,i}''$ of f_h' and with *homogeneous* Dirichlet boundary conditions on the boundaries $\partial\Omega_i$.
4. Extend (prolongate) the $u_{h,i}$ to functions u_i defined on the overall grid $\mathcal{T}_h$ (zero outside $\bar{\Omega}_i$), and set $\tilde{u} = u_0 + \sum_{i=1}^N u_i$.

A formally complete description involves the precise definition of prolongation operators E_i and their transposes E_i^{T}, such that the method exactly fits into our abstract AS framework.

One may also think of omitting the coarse space V_0. However, convergence theory tells us that this has an unfavorable effect on the condition number of the preconditioner for $h \to 0$, similarly as for the Jacobi

case. One should also bear in mind that, if the method is used as a preconditioner, e.g., for CG, then the right hand side of the given problem f is actually some intermediate residual.

Usually, the cost for performing a preconditioning step is significantly smaller than solving the full system exactly, at least if the coarser problem is also treated by some substructuring technique.

Apart from the acceleration effect, the most favorable feature of such a preconditioner is given by the fact that complex geometries can be reduced to simpler subgeometries, and the fact that individual subdomain corrections can be computed in parallel. This is the major advantage of additive compared to multiplicative approaches, and this is more important for practice than the observation that multiplicative preconditioners are usually more accurate.

However, due to the overlapping domains this requires some local communication, e.g., if each domain is mapped to an individual processor. This motivated the search for non-overlapping techniques; a typical example for such a technique is discussed later in this section.

A bit of theory. Application of Theorems 14.1.

The assumption on the maximal amount of simultaneous overlap of the subdomains implies that the spectral radius $\rho(\mathcal{E})$ appearing in Theorems 14.1 and 14.2 is bounded by M, the maximal number of simultaneous overlaps. We prove this with the help of the following lemma, which can be regarded as a sharpened version of the standard inequality $\|\mathcal{E}\|_2 \le \|\mathcal{E}\|_F$.

Lemma 14.7 *Let $\mathcal{E} \in \mathbb{R}^{N\times N}$ be a symmetric matrix with $M \le N$ non-zero entries per row and column. Then,*

$$\|\mathcal{E}\|_2 \le \sqrt{M} \max_{j=1\ldots N} \|\mathcal{E}_j\|_2$$

where $\mathcal{E}_j$ denotes the j-th column of $\mathcal{E}$.

In particular, if $|\mathcal{E}_{i,j}| \le 1$ for all $i,j \in \{1,\ldots,N\}$, then

$$\|\mathcal{E}\|_2 \le M$$

Proof: For each i, let $J(i) \subseteq \{1,\ldots,N\}$ denote the set of indices j with $\mathcal{E}_{i,j} \neq 0$. Then, for any $x \in \mathbb{R}^N$ application of the Cauchy-Schwarz inequality yields

$$\begin{aligned}
\|\mathcal{E}x\|_2^2 = \sum_i \Big| \sum_{j\in J(i)} \mathcal{E}_{i,j}\, x_j \Big|^2 &\le \sum_i \Big(\sum_{j\in J(i)} x_j^2 \cdot \sum_{j\in J(i)} {\mathcal{E}_{i,j}}^2 \Big) \\
&\le \Big(\sum_i \Big| \sum_{j\in J(i)} x_j^2 \Big| \Big) \cdot \max_i \sum_{j\in J(i)} {\mathcal{E}_{i,j}}^2 \\
&\le \Big(\sum_{\substack{i,j \\ \mathcal{E}_{i,j}\neq 0}} x_j^2 \Big) \cdot \max_j \|\mathcal{E}_j\|_2^2 \le M \,\|x\|_2^2 \cdot \max_j \|\mathcal{E}_j\|_2^2
\end{aligned}$$

by assumption on the sparsity structure of $\mathcal{E}$. This implies the first assertion, and the second assertion follows from the fact that for $|\mathcal{E}|_{i,j} \le 1$ we have $\|\mathcal{E}_j\|_2 \le \sqrt{M}$ for all j. □

Now we consider the matrix

$$\mathcal{E} = \big(\mathcal{E}_{i,j}\big), \quad \text{with} \quad \mathcal{E}_{i,j} = \sup_{\substack{0\neq u_i \in V_i \\ 0\neq u_j \in V_j}} \frac{|a(u_i,u_j)|^2}{a(u_i,u_i)\,a(u_j,u_j)} \le 1$$

involved in Assumption 14.2. For $\Omega_i \cap \Omega_j = \emptyset$ we have $\mathcal{E}_{i,j} = 0$ because (using the notation of Sec. 14.2)

$$a(u_i,u_j) = (E_i^{\mathrm{T}} A_i\, u_i'', E_j^{\mathrm{T}} A_j\, u_j'') = (A_i\, u_i'', E_i\, E_j^{\mathrm{T}} A_j\, u_j'') \quad \text{with} \quad E_i\, E_j^{\mathrm{T}} = 0$$

Therefore, due to our assumption on the maximal simultaneous overlap, the matrix $\mathcal{E}$ satisfies the assumptions of Lemma 14.7. This shows $\rho(\mathcal{E}) \leq M$.

In order to apply Theorem 14.1 it remains to verify that the decomposition ist stable in the sense of Assumption 14.1. This is the topic of the following theorem.

Theorem 14.3 *[AS with 'generous' overlap] Given the above hypotheses on the problem and the subdomains, there exists $C > 0$ such that any $u \in V_h$ can be decomposed as $u = \sum_{i=0}^{N} u_i$ with $u_i \in V_i$ and*

$$\sum_{i=0}^{N} \|u_i\|^2_{H^1(\Omega)} \leq C \left(1 + \frac{H^2}{\delta^2}\right) \|u\|^2_{H^1(\Omega)}$$

where $\delta > 0$ is a parameter characterizing the 'amount of overlap'. (δ can be defined in terms of the characteristic functions of the subdomains Ω_i.)

Proof: See [21]. The 1D version is an exercise. □

The boundedness and ellipticity of $a(\cdot,\cdot)$ implies $a(u_i, u_i) \leq C \|u_i\|^2_{H^1(\Omega)}$ and $\|u\|^2_{H^1(\Omega)} \leq C\, a(u,u)$. This shows that Theorem 14.3 implies that Assumption 14.1 is satisfied with

$$C_0^2 = C\left(1 + \frac{H^2}{\delta^2}\right).$$

Together with Theorem 14.1 shows that the condition number of the AS preconditioner grows at most like $\mathcal{O}(1 + H^2/\delta^2)$. Thus, the condition number is bounded uniformly in h. Furthermore, if the overlap is 'generous', i.e., $\delta \geq cH$, then the condition number is bounded uniformly in both h and H.

Remark 14.5 At first sight, the two-level approach might appear unnatural, namely to add all the local solutions to a coarse approximation u_0. In fact, in a classical iterative Schwarz procedure this step would not be included. However, this is nothing but an additional, global overlap, and the role of the coarse space V_0 is to ensure that information obtained by the subdomain solves is communicated to the other subdomains. ∎

The spectral condition number for the MS preconditioner can be estimated in an analogous way on the basis of Theorem 14.2.

Exercise 14.2 Realize the AS preconditioner for the 1D Poisson example on $\Omega = [0,1]$ (FD discretization) with a coarse and a fine mesh and two overlapping subdomains (subintervals) $\Omega_1 = [0,a]$, $\Omega_2 = [b,1]$ with $a > b$, $a - b = \delta = H$. Choose, e.g., $H = 0.1$ (fixed), $h = 0.01$, and reduce the size of h. Observe the performance of `pcg` preconditioned in this way. ∎

Exercise 14.3 Devise the multiplicative (MS) analog of the two-level AS procedure introduced above.

Hint: Compute the residual after each single step before proceeding. ∎

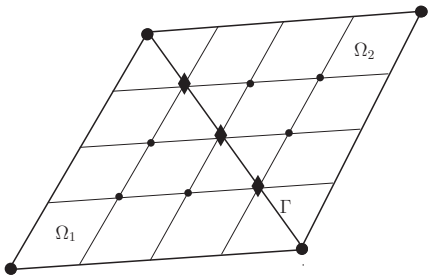


Figure 14.1: An edge patch $\Omega_\Gamma = \Omega_1 \cup \Omega_2 \cup \Gamma$, with nodes and mesh lines.

A non-overlapping additive two-level method.

We again consider the problem (14.46). We introduce a non-overlapping decomposition method involving appropriate subdomain solvers. We will concentrate on the conceptual and algorithmic aspects, assuming $a_i \equiv a$ ('exact solvers') for simplicity. To fix ideas for the definition of the subspaces involved in the non-overlapping approach, we assume:

- The triangulation $\mathcal{T}_h$ is a refinement of a coarse triangulation $\mathcal{T}_H$ (consisting of triangles).
- The subdomains[79] Ω_i are taken exactly as the triangles of the coarse triangulation $\mathcal{T}_H$.
- The method works on the union of the patches of neighboring triangles; we will consider a version where these patches are chosen as $\Omega_{i,j} = \Omega_i \cup \Omega_j$, all unions of pairs of neighboring triangles Ω_i, Ω_j sharing a common edge $\Gamma_{i,j}$.

Of course, the triangulations $\mathcal{T}_H$ and $\mathcal{T}_h$ are assumed to be admissible in the sense of Sec. A.5.

See Fig. 14.1 for the simplest case of a decomposition into two subdomains only.

Before we proceed with the formal definition of the subspaces that comprise the splitting of V_h for this method, we recall that the unknowns (degrees of freedom) in the original discrete problem correspond in a one-to-one fashion to the nodes of the fine triangulation. We split the set of nodes $\mathcal{N}$ of the fine triangulation $\mathcal{T}_h$ into:

- $\mathcal{N}_0$: the nodes of the coarse triangulation $\mathcal{T}_H$,
- $\mathcal{N}_i$: the nodes in the interior of the subdomains Ω_i (= triangles of the coarse triangulation),
- $\mathcal{N}_{i,j}$: the nodes on the edges $\Gamma_{i,j} = \bar{\Omega}_i \cap \bar{\Omega}_j$ of the coarse triangulation which are not contained in $\mathcal{N}_0$.

Again, V_h denotes the FEM space associated with $\mathcal{T}_h$. We construct a non-overlapping splitting of V_h in the form

$$V_h = V_0 + \sum_i V_i + \sum_{i,j} V_{i,j} \tag{14.47}$$

Of course, the patches $\Omega_{i,j}$ are overlapping to some extent; however, the term 'non-overlapping' refers to the fact that we design (14.47) as a *direct sum.* To this end we associate the subspaces V_0, V_i and $V_{i,j}$ with the nodal sets $\mathcal{N}_0$, $\mathcal{N}_i$ and $\mathcal{N}_{i,j}$, i.e., V_0 is associated with $\mathcal{N}_0$, each V_i with the nodes of $\mathcal{N}_i$ (i.e., with the interior of the subdomain Ω_i), and each $V_{i,j}$ with the set $\mathcal{N}_{i,j}$ (i.e., with the edge $\Gamma_{i,j}$). That (14.47) is indeed a direct sum will follow from the fact that, for each of the sets V_0, V_i, $V_{i,j}$, we will construct a basis which has a 'Kronecker δ-property' for the nodes of the corresponding nodal set $\mathcal{N}_0$, $\mathcal{N}_i$, or $\mathcal{N}_{i,j}$, respectively.

In detail, the spaces that make up the splitting of V_h are defined as follows:

[79] The splitting of V_h will be based on the subdomains Ω_i as well as so-called edge patches $\Omega_{i,j}$ to be defined below.

- V_0 is the space of piecewise linears on $\mathcal{T}_H$: $V_0 = V_H$.
- For each subdomain Ω_i we set $V_i = V_h \cap H_0^1(\Omega_i)$.
- For each edge $\Gamma_{i,j}$ of the coarse triangulation ($\Gamma_{i,j}$ denoting the edge shared by the subdomains Ω_i and Ω_j), we define the 'edge space' $V_{i,j}$ as the set of functions from V_h,

  (i) which are supported by the edge patch $\overline{\Omega}_{i,j} = \overline{\Omega_i \cup \Omega_j \cup \Gamma_{i,j}}$,

  (ii) and which are *discrete harmonic*, i.e.,

$$u \in V_{i,j} \;\Leftrightarrow\; \begin{cases} \operatorname{supp} u \subseteq \Omega_{i,j} & \text{and} \\ a(u,v) = 0 \quad \forall\, v \in V_i \oplus V_j \end{cases} \tag{14.48}$$

  Note that $\dim(V_{i,j}) = |\mathcal{N}_{ij}|$.

In this way we obtain a splitting

$$V_h = V_0 \oplus \sum_{i=1}^{N} V_i \oplus \sum_{\substack{i,j \\ \Gamma_{i,j} \neq \emptyset}} V_{i,j} \tag{14.49}$$

and a dimension argument shows that this is in fact a direct sum.

In the notation of Sec. 14.2, the non-overlapping AS preconditioner is now defined according to (cf. (14.17a)):

$$B^{-1} = \sum_{i=0}^{N} E_i\, A_i^{-1}\, E_i^{\mathrm{T}} + \sum_{i,j} E_{i,j}\, A_{i,j}^{-1}\, E_{i,j}^{\mathrm{T}} \tag{14.50}$$

where E_i and $E_{i,j}$ are the matrix representations of the embeddings $V_i \subseteq V_h$ and $V_{i,j} \subseteq V_h$; the matrices A_i and $A_{i,j}$ are the stiffness matrices for the subspaces V_i and $V_{i,j}$.

The stiffness matrices A_i and the matrices E_i corresponding to the problems based on the spaces V_i and the coarse space V_0 are defined in a straightforward way: As for the overlapping method from Sec. 14.4, A_0 corresponds to the stiffness matrix for the approximation of the given problem on the coarse mesh $\mathcal{T}_H$, and the subproblems to be solved on the V_i, $i = 1 \ldots N$, are local discrete Dirichlet problems on the subdomains Ω_i, with homogeneous boundary conditions on $\partial\Omega_i$.

The appropriate interpretation of the stiffness matrices $A_{i,j}$ for the edge spaces $V_{i,j}$ is more subtle. These are defined over the edge patches $\Omega_{i,j} = \Omega_i \cup \Omega_j \cup \Gamma_{i,j}$ with internal interface $\Gamma_{i,j}$. The correct interpretation of these subproblems is essential for understanding the nature of the non-overlapping AS, which is defined by (14.50) in a purely formal way. For the case where the stiffness matrix A_h is not formed explicitly but only the matrix-vector multiplication $u_h' \mapsto A_h u_h'$ is available, this will also show how the data for the local 'edge problems' have to be assembled and how these subproblems are to be solved.

The appropriate interpretation of the local edge problems is the topic of the following considerations, where we are ignoring discretization issues for a moment. To fix ideas, we consider a single edge Γ that is shared by two subdomains Ω_1 and Ω_2, i.e., Γ is an interior interface between Ω_1 and Ω_2, see Fig. 14.2. We denote the corresponding edge patch by $\Omega = \Omega_\Gamma = \Omega_1 \cup \Omega_2 \cup \Gamma$.

- We interpret the edge subproblems in the following way: Consider the local solutions u_1, u_2 of $-\Delta u = f$ on Ω_1, Ω_2, with homogeneous Dirichlet boundary conditions on $\partial\Omega_1, \partial\Omega_2$ (including Γ). These solutions are independent of each other. Now, evaluate the jump[80]

$$[\partial_n u_I] := \partial_{n_1} u_1 + \partial_{n_2} u_2 \tag{14.51}$$

[80] $[\partial_n u_I] = \partial_{n_1} u_1 + \partial_{n_2} u_2$ denotes the *jump* of the normal derivative over the interface Γ, where the $\partial_{n_i} u_i$ are oriented outward Ω_i.

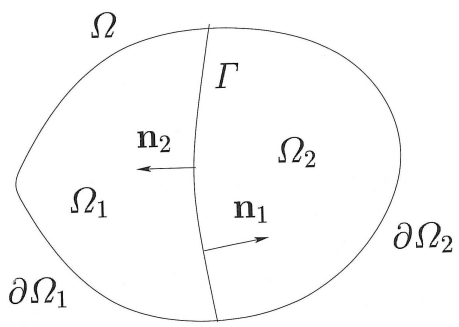


Figure 14.2: A combined domain $\Omega_1 \cup \Omega_2 \cup \Gamma$ with interface Γ.

across the interface Γ, and consider the continuous solution w of the 'interface problem'

$$-\Delta w = 0 \ \text{ on } \Omega_\Gamma, \quad w = 0 \ \text{ on } \partial\Omega_\Gamma, \quad [\partial_n w] = g_\Gamma := -[\partial_n u_I] \ \text{ on } \Gamma \tag{14.52}$$

(observe the minus sign with the jump term). We call w the *harmonic extension* of its trace $w|_\Gamma$ to Ω_Γ. Then, by construction, the linear combination $u = u_1 + u_2 + w$ is the solution of[81]

$$-\Delta u = f \ \text{ on } \Omega_\Gamma, \quad u = 0 \ \text{ on } \partial\Omega_\Gamma$$

with $[\partial_n u] = 0$ on Γ. In this way, we have solved the Dirichlet problem on Ω_Γ by means of solving two independent problems in Ω_1, Ω_2 and the additional interface problem (14.52).

At the discrete level, the solution of the subproblems on Ω_1 and Ω_2 is straightforward and gives rise to discrete approximations for u_1 and u_2. We now investigate the nature of the auxiliary interface problem (14.52),

- Problem (14.52) is of the type

$$-\Delta w = 0 \ \text{ on } \Omega_\Gamma, \quad w = 0 \ \text{ on } \partial\Omega_\Gamma, \quad [\partial_n w] = g_\Gamma \ \text{ on } \Gamma \tag{14.53a}$$

with a prescribed jump g_Γ over the interface. (In (14.52) we have $g_\Gamma = -[\partial_n u_I]$.)

To obtain the weak form of (14.53a) we use partial integration (A.7) on both subdomains Ω_i, with test functions $v \in H_0^1(\Omega_\Gamma)$:

$$\begin{aligned}\int_{\Omega_1} \Delta w\, v &= \int_{\partial\Omega_1} \partial_{n_1} w\, v - \int_{\Omega_1} \nabla w \cdot \nabla v \\ &= \int_{\partial\Omega_1 \setminus \Gamma} \partial_n w\, v + \int_\Gamma \partial_{n_1} w\, v - \int_{\Omega_1} \nabla w \cdot \nabla v \\ &= 0 + \int_\Gamma \partial_{n_1} w\, v - \int_{\Omega_1} \nabla w \cdot \nabla v\end{aligned}$$

and analogously,

$$\int_{\Omega_2} \Delta w\, v = 0 + \int_\Gamma \partial_{n_2} w\, v - \int_{\Omega_2} \nabla w \cdot \nabla v$$

Adding up these identities yields

$$\int_{\Omega_\Gamma} \Delta w\, v = \int_\Gamma [\partial_n w]\, v - \int_{\Omega_\Gamma} \nabla w \cdot \nabla v$$

[81] Here u_1 is extended by 0 to Ω_2, and vice versa.

This leads us to the weak form of (14.53a):

$$\text{Find } w \in H_0^1(\Omega_\Gamma) \text{ such that } \int_{\Omega_\Gamma} \nabla w \cdot \nabla v = \int_\Gamma g_\Gamma\, v \quad \forall\, v \in H_0^1(\Omega_\Gamma) \tag{14.53b}$$

- Now we consider the discretized version of (14.53),[82]

$$\begin{pmatrix} A_{\Gamma,\Gamma} & A_{\Gamma,I} \\ A_{I,\Gamma} & A_{I,I} \end{pmatrix} \begin{pmatrix} w'_\Gamma \\ w'_I \end{pmatrix} = \begin{pmatrix} g'_\Gamma \\ 0' \end{pmatrix} \tag{14.54}$$

with the appropriately blocked local stiffness matrix over Ω_Γ and the appropriate coordinate vectors. The indices Γ and I refer to the interface and interior components, respectively, and we have combined the interior nodes from $\mathcal{N}_1$ and $\mathcal{N}_2$ into the set $\mathcal{N}_I$ of all interior nodes. The block structure of the sparse stiffness matrix in (14.54) has to be interpreted accordingly.

Note: Of course, we may now simply solve the system (14.54). However, having the case of a larger number of subdomains in mind (with overlapping patches) we are interested in a inexact but 'localized' procedure to be used as a preconditioner, and which also lends itself better to parallelization. This can be achieved in the following way:

Elimination of the variable w'_I from (14.54) gives rise to a linear system for the edge component w'_Γ:

$$S\, w'_\Gamma = g'_\Gamma \tag{14.55a}$$

where S is the *Schur complement* of $A_{\Gamma,\Gamma}$,

$$S = A_{\Gamma,\Gamma} - A_{\Gamma,I}\, A_{I,I}^{-1}\, A_{I,\Gamma} \tag{14.55b}$$

As soon as w'_Γ is determined, the component w'_I is obtained as

$$w'_I = -A_{I,I}^{-1}\, A_{I,\Gamma}\, w'_\Gamma \tag{14.56}$$

We can interpret these operations in the following way:

- The action of S has the nature of a (discrete) *Dirichlet-to-Neumann map:* It transfers Dirichlet (i.e., pointwise) data w'_Γ on Γ to Neumann data g'_Γ on Γ.
- The action of S^{-1} has the nature of a (discrete) *Neumann-to-Dirichlet map:* The Dirichlet data w'_Γ on the interface Γ are computed from the solution of (14.55a) with a Neumann condition on Γ.
- Furthermore, w'_I is the *discrete harmonic extension* of w'_Γ to the interior of Ω_Γ, as explained below.

In order to understand the computational realization of (14.55),(14.56), the notion of discrete harmonic extension and the action of S and S^{-1} has to be understood more precisely. This is the topic of the following considerations.

- **Discrete harmonic extension:** Let Dirichlet data w'_Γ on Γ be given. We denote $-A_{I,I}^{-1}\, A_{I,\Gamma}$ by E and consider (cf. (14.56))

$$w'_I = E\, w'_\Gamma = -A_{I,I}^{-1}\, A_{I,\Gamma}\, w'_\Gamma$$

w'_I is the discrete harmonic extension of w'_Γ to the interior nodes in Ω_Γ. To see what this means, insert w'_Γ and w'_I into (14.54) to obtain

[82] Think of a straightforward Galerkin/FEM approximation on Ω_Γ with piecewise linear elements and appropriate numerical quadrature for setting up the system. For evaluating the right hand side, this involves square and line quadratures yielding f'_I and g'_Γ.

$$\begin{pmatrix} A_{\Gamma,\Gamma} & A_{\Gamma,I} \\ A_{I,\Gamma} & A_{I,I} \end{pmatrix} \begin{pmatrix} w'_\Gamma \\ w'_I \end{pmatrix} = \begin{pmatrix} S\,w'_\Gamma \\ 0' \end{pmatrix} \tag{14.57}$$

This means that

$$w' = \begin{pmatrix} w'_\Gamma \\ w'_I \end{pmatrix} = \begin{pmatrix} w'_\Gamma \\ E\,w'_\Gamma \end{pmatrix} \tag{14.58}$$

can be interpreted as the solution of a problem of the form (14.54), the discrete version of (14.53), with Neumann data Sw'_Γ on Γ. Thus, w' is the coordinate representation of an element w from the edge space $V_\Gamma := V_{1,2}$ (see (14.48)). The interior component w'_I is uniquely determined by w'_Γ. We call $E = -A_{I,I}^{-1}\,A_{I,\Gamma}$ the (discrete) harmonic extension operator.

- **Action of** S: For any given (discrete) Dirichlet data w'_Γ on the interface Γ, consider its discrete harmonic extension (14.58). Relation (14.57) leads to the following interpretation:

$$g'_\Gamma = S\,w'_\Gamma \quad \text{is a discretized version of the jump } [\partial_n w] \text{ across the interface } \Gamma \tag{14.59}$$

  i.e., a discrete Dirichlet-to-Neumann map.

- **Action of** S^{-1}: For any given (discrete) Neumann g'_Γ on the interface Γ, relation (14.57) leads to the following interpretation:

$$w'_\Gamma = S^{-1}\,g'_\Gamma \quad \text{is the solution of (14.54) evaluated at the interface } \Gamma \tag{14.60}$$

  i.e., a discrete Neumann-to-Dirichlet map. The inner component w'_I of the solution w' is the discrete harmonic extension of w'_Γ.

For the solution of the original, continuous problem (14.53), a similar interpretation in terms of a harmonic extension can be given. The corresponding Neumann-to-Dirichlet map is called the *Poincaré-Steklov operator*. For the precise theoretical foundation of this clever construction in the PDE context, see [17]; it involves a multi-domain formulation using local Poincaré-Steklov operators and harmonic extensions.

Exercise 14.4 Show that, with appropriate subblocking,

$$\begin{pmatrix} A_{\Gamma,\Gamma} & A_{\Gamma,I} \\ A_{I,\Gamma} & A_{I,I} \end{pmatrix} = \begin{pmatrix} A_{\Gamma,\Gamma} & A_{\Gamma,1} & A_{\Gamma,2} \\ A_{1,\Gamma} & A_{1,1} & 0 \\ A_{2,\Gamma} & 0 & A_{2,2} \end{pmatrix} \tag{14.61a}$$

the Schur complement (14.55b) can be expressed as

$$S = A_{\Gamma,\Gamma} - A_{\Gamma,1}\,A_{1,1}^{-1}\,A_{1,\Gamma} - A_{\Gamma,2}\,A_{2,2}^{-1}\,A_{2,\Gamma} \tag{14.61b}$$

■

In the case of a single domain $\Omega = \Omega_\Gamma$ with two subdomains Ω_1 and Ω_2, we see that, from the computational point of view, the solution of the original problem

$$-\Delta u = f \text{ on } \Omega_\Gamma, \quad u = 0 \text{ on } \partial\Omega_\Gamma \tag{14.62}$$

amounts, after discretization, to the following steps:

- Assemble the stiffness matrices in (14.61a).
- For $i = 1, 2$, compute the inverse matrices $A_{i,i}^{-1}$, and determine the discrete approximations u'_i of the subdomain problems

$$-\Delta u_i = f \text{ on } \Omega_I, \quad u = 0 \text{ on } \partial\Omega_I$$

- Extend u_1' by 0 to Ω_2 and vice versa.
- Evaluate the jump terms
$$g_\Gamma' = ([\partial_n u_I])' = (\partial_{n_1} u_1)' + (\partial_{n_2} u_2)'$$
(see (14.51)).
- Assemble the Schur complement (see (14.61b))
$$S = A_{\Gamma,\Gamma} - A_{\Gamma,1}\, A_{1,1}^{-1}\, A_{1,\Gamma} - A_{\Gamma,2}\, A_{2,2}^{-1}\, A_{2,\Gamma}$$
- Solve the Schur complement system, i.e., the discrete interface problem (14.55a),
$$S\, w_\Gamma' = g_\Gamma'$$
- Extend w_Γ' to w' via discrete harmonic extension (14.56),
$$w_I' = E\, w_\Gamma' = -A_{I,I}^{-1}\, A_{I,\Gamma}\, w_\Gamma'$$
- Determine u' on Ω_Γ as
$$u' = u_1' + u_2' + w'$$
By construction, this gives rise to the exact discrete solution.

Exercise 14.5 Reproduce the above considerations, at the continuous and the discrete level, for the 1D Poisson problem with $\Omega_1 = (0, c)$, $\Omega_2 = (c, 1)$, and $\Gamma = \{c\}$. ∎

The full preconditioner:

Returning to the case of a general decomposition into an arbitrary number of subdomains Ω_i, we can now describe our non-overlapping AS preconditioner, which is formally defined by (14.50), in precise detail:

- Solve the given problem on the coarse triangulation $\mathcal{T}_H$, giving rise to u_0.
- Apply the local solution procedure described above, over all subdomains Ω_i and Ω_j and all edge patches $\Omega_{i,j}$. This gives local contributions
  - u_i defined over Ω_i and extended by 0 outside Ω_i, and
  - $w_{i,j}$ defined over all edge patches $\Omega_{i,j}$ after local discrete harmonic extension, which are also extended by 0 outside $\Omega_{i,j}$.
- Adding up defines the action of the preconditioner (cf. 14.50)):
$$\tilde{u} = u_0 + \sum_{i=1}^{N} u_i + \sum_{\substack{i,j \\ \Gamma_{i,j} \neq \emptyset}} w_{i,j} \tag{14.63}$$

In particular, to determine the $w_{i,j}$, subproblems of the type (14.53b) are solved, in the way as described above, for all edge patches $\Omega_\Gamma = \Omega_{i,j}$. All interior local subproblems are of the original type $-\Delta u = f$, with local homogeneous boundary conditions as in the overlapping case. However, as we have seen, the edge subproblems are set up in a more subtle way described above, using discrete harmonic extensions. This patching procedure realizes an alternative way of communication between local subdomains, compared to overlapping.

In fact, this non-overlapping method has an 'overlapping taste', because the edge patches $\Omega_{i,j}$ have some overlap from a geometric point of view. However, the method is non-overlapping in an algebraic sense; it can more easily be parallelized than the an overlapping method. In particular, with subblocking as in Exercise 14.4, assembling the Schur complement matrices for the edge systems can be readily parallelized over the subdomains.

Theory shows that the combined strategy involving a globally defined, coarse 'skeleton' approximation u_0 (as for the overlapping AS) is essential for the successful performance of the preconditioner; see [17],[21].

Remark 14.6 Of course, for the case of more than $N > 2$ subdomains, this preconditioner is *not* exact (in contrast to the simplest case $N = 2$ considered above).

Concerning the concrete implementation of the preconditioner (14.63), realizing it by explicitly computing all the (rather small) local inverses $A_{i,i}$ and using this for setting up the Schur complement matrices (14.61b) is a common technique. This can be interpreted in the sense that, in a first step of the process, all interior nodes are eliminated, which is called *static condensation*. However, from (14.61b) we also see that, actually, not the 'complete' inverses $A_{i,i}^{-1}$ are required but only $A_{i,i}^{-1} A_{i,\Gamma}$, which has some potential for further optimization of the computational effort.

Explicit inversion of the $A_{i,i}$ is avoided, if iterative approximate local solvers are to be applied for the Schur complement systems: Each evaluation of a matrix-vector product $u \mapsto S u$ involves the invocation of two subdomain solvers, which can be performed in parallel. ∎

In the following we consider one possible simplification of our preconditioner.

A Neumann-Neumann preconditioner.

A common technique is to approximate the Schur complement system (14.55) in the following way. Consider (14.55a),

$$S w'_\Gamma = g'_\Gamma$$

with S written in the form (14.61b),

$$S = A_{\Gamma,\Gamma} - A_{\Gamma,1} A_{1,1}^{-1} A_{1,\Gamma} - A_{\Gamma,2} A_{2,2}^{-1} A_{2,\Gamma} \tag{14.64}$$

For any pair of nodes (x, x') on the interface Γ, the corresponding entry $(A_{\Gamma,\Gamma})_{x,x'}$ of $A_{\Gamma,\Gamma}$ is given by

$$(A_{\Gamma,\Gamma})_{x,x'} = \int_{\Omega_\Gamma} \nabla v_{h_x} \cdot \nabla v_{h,x'} = \int_{\Omega_1} \nabla v_{h_x} \cdot \nabla v_{h,x'} + \int_{\Omega_2} \nabla v_{h_x} \cdot \nabla v_{h,x'} \tag{14.65}$$

where $v_{h,x}$ and $v_{h,x'}$ denote the nodal basis functions (hat functions) associated with these nodes. In (14.65) we have split the integral into two contributions associated with the subdomains Ω_i, giving rise to a splitting

$$A_{\Gamma,\Gamma} = A_{\Gamma,\Gamma;1} + A_{\Gamma,\Gamma;2}$$

and

$$S = S_1 + S_2, \quad \text{with} \quad S_i = A_{\Gamma,\Gamma;i} - A_{\Gamma,i} A_{i,i}^{-1} A_{i,\Gamma}$$

We are aiming for a simplified approximation of the Schur complement S from (14.64), i.e., a preconditioner $\tilde{S} \approx S$. One possible choice is $\tilde{S} = S_1$ (or S_2). Consider the action of S_1^{-1}, i.e., the computation of $w'_{\Gamma,1} = S_1^{-1} g'_\Gamma$: Analogously as in (14.54)ff., this is equivalent to solving the system[83] with mixed boundary conditions,

$$\begin{pmatrix} A_{\Gamma,\Gamma;1} & A_{\Gamma,1} \\ A_{1,\Gamma} & A_{1,1} \end{pmatrix} \begin{pmatrix} w'_{\Gamma,1} \\ w'_1 \end{pmatrix} = \begin{pmatrix} g'_\Gamma \\ 0' \end{pmatrix} \tag{14.66}$$

[83] See Exercise 14.6.

and extracting the interface part $w'_{\Gamma,1}$ from the solution vector $(w'_{\Gamma,1}, w'_1)^{\mathrm{T}}$. This system is similar to (14.54), but smaller, and only the subdomain Ω_1 is involved. This is a localization property which is useful in view of capability for parallelization.

The action of this preconditioner is equivalent to the solution of $S_1\, w'_{\Gamma,1} = g'_\Gamma$. The so-called Neumann-Neumann preconditioner is obtained if Ω_1 and Ω_2 are treated in a way having equal rights, i.e., we solve

$$S_1\, w'_{\Gamma,1} = \omega_1\, g'_\Gamma \quad \text{and} \quad S_2\, w'_{\Gamma,1} = \omega_2\, g'_\Gamma$$

and compute $\tilde{w}'_\Gamma = w'_{\Gamma,1} + w'_{\Gamma,2}$. Here it appears natural to choose $\omega_1 + \omega_2 = 1$, but the concrete choice of the ω_i is not straightforward. This means that S^{-1} is approximated by $\omega_1\, S_1^{-1} + \omega_2\, S_2^{-1}$.

Exercise 14.6 In an analogous way as (14.54) is related to problem (14.53), the system (14.66) is related to a Poisson problem posed on Ω_1 with a Neumann condition on the interface Γ,

$$-\Delta w = 0 \ \text{ on } \Omega_1, \quad w = 0 \ \text{ on } \partial\Omega_1 \backslash \Gamma, \quad \partial_n w = g_\Gamma \ \text{ on } \Gamma \tag{14.67}$$

Verify the relationship between (14.67) and (14.66) via consideration of the weak form of (14.67), with test functions $v \in H_0^1(\Omega_1)$. (Cf. (14.53)–(14.54).) ∎

The full preconditioner:

Returning again to the case of a general decomposition into an arbitrary number of subdomains Ω_i, a Neumann-Neumann preconditioner is described in the literature as follows. First of all, the subdomain solutions $u_0, u_1, \ldots, u_N$ are determined in the same way as before (see (14.63)), and the associated Neumann jumps over all interfaces are evaluated.

Then, instead of computing the $w_{i,j}$ from (14.63) by considering all possible edge patches, it is more practical to consider again all subdomains Ω_i (here we are assuming triangles) with boundaries consisting of several edges Γ_k. Let A_h denote the global stiffness matrix. Now, with appropriate numbering,

- each block-row of A_h associated with the interior nodes in some the subdomains Ω_i consists of a quadratic subblock $A_{i,i}$ and one or several non-vanishing block $A_{i,\gamma}$, where the index γ refers to an edge which is part of $\partial\Omega_i$;
- each block-row of A_h associated with nodes along some interior edge Γ_γ consists of a quadratic subblock $A_{\gamma,\gamma}$ and at least two non-vanishing subblocks $A_{\gamma,i}$, where the index i refers to subdomains neighboring Γ_γ.

For the simple setting illustrated in Fig. 14.3 for instance, with subdomains numbered by $1, 2, 3$ and inner edges numbered by a, b, the global stiffness matrix has the block structure

$$A_h = \begin{pmatrix} A_{1,1} & A_{1,a} & & & \\ A_{a,1} & \boldsymbol{A_{a,a}} & \boldsymbol{A_{a,2}} & & \\ & \boldsymbol{A_{2,a}} & \boldsymbol{A_{2,2}} & \boldsymbol{A_{2,b}} & \\ & & \boldsymbol{A_{b.2}} & \boldsymbol{A_{b,b}} & A_{b,3} \\ & & & A_{3,b} & A_{3,3} \end{pmatrix} \tag{14.68}$$

Here the subblocks associated with subdomain Ω_2 have been highlighted, and we exemplify the procedure (which is applied to all Ω_i) for subdomain Ω_2: 'Cut out' and solve the highlighted subsystem, i.e., solve the localized system

$$A_h^{(2)}\, w'_2 = \begin{pmatrix} g'_a \\ 0 \\ g'_b \end{pmatrix} \tag{14.69a}$$

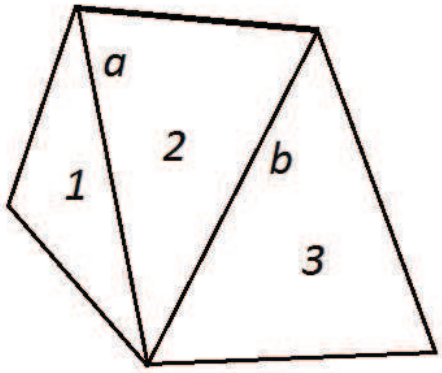


Figure 14.3: Decomposition of a polygonal domain into three triangles. In this simple example the boundaries of all subdomains meet the boundary of the full domain, i.e., there are no 'strictly interior' subdomains.

with stiffness matrix

$$A_h^{(2)} = \begin{pmatrix} A_{a,a} & A_{a,2} & \\ A_{2,a} & A_{2,2} & A_{2,b} \\ & A_{b.2} & A_{b,b} \end{pmatrix} \tag{14.69b}$$

and where g'_a, g'_b stem from the Neumann jumps at the interfaces a and b. This system can be solved using the Schur complement of $A_{2,2}$,

$$S_{2,2} = \begin{pmatrix} A_{a,a} & \\ & A_{b,b} \end{pmatrix} - \begin{pmatrix} A_{a,2} \\ A_{b,2} \end{pmatrix} \cdot A_{2,2}^{-1} \cdot \begin{pmatrix} A_{2,a} & A_{2,b} \end{pmatrix} = \begin{pmatrix} A_{a,a} - A_{a,2}A_{2,2}^{-1}A_{2,a} & -A_{a,2}A_{2,2}^{-1}A_{2,b} \\ -A_{b,2}A_{2,2}^{-1}A_{2,a} & A_{b,b} - A_{b,2}A_{2,2}^{-1}A_{2,b} \end{pmatrix}$$

to obtain the interface parts of the solution w'_2.

In the general case, this procedure is executed for all subdomains Ω_i, the resulting interface (edge) contributions are summed up. Finally, the discrete harmonic extensions over all subdomains associated with all the edges are computed (again by solving local mixed boundary value problems on all subdomains), and they are added to $\sum_{i=0}^{N} u_i$. This includes extension by zero and appropriate weighting. Usually, the weight for an interface node is chosen as the inverse of the number of $\bar{\Omega}_i$ sharing this node.

However, we are confronted with the fact that for a *strictly interior* subdomain Ω_i the associated localized system has Neumann data on the complete boundary $\partial\Omega_i$, which means the the corresponding local stiffness matrix (the analog of (14.69b)) is singular. It therefore is a common technique to apply a pseudo-inverse. The classical Moore-Penrose inverse bases on the singular value decomposition is generally applicable to this purpose but computationally rather expensive. Depending on the problem at hand, cheaper and more specific techniques van be used, see [21].

Remark 14.7 The localized system of the type (14.69) are 'memory-overlapping', because each edge is subset of several $\partial\Omega_i$. In a parallel implementation this requires some moderate amount of local communication, but the solution of the localized systems can be performed in parallel. ∎

Remark 14.8 For theoretical discussion of several preconditioning techniques in the FEM context based on substructuring see for instance [18]. In particular, for the Poisson problem, spectral equivalence of the preconditioners considered here can be proved; see [18, Section 4.4]. ∎

A FETI preconditioner.

Various domain decomposition methods for constructing efficient preconditioners, in particular for complex geometries are the topic of many recent research activities, see [17], [21]. A prominent class are the so-called FETI techniques ('Finite Element Tearing and Interconnecting').

To discuss the basic idea, we again consider the case of two subdomains Ω_1, Ω_2 with an interface Γ, as in Fig. 14.2. Instead of the full problem posed on $\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma$, we consider two local mixed Dirichlet/Neumann boundary value problems posed on the subdomains Ω_i; after discretization, these take the form

$$\begin{pmatrix} A_{\Gamma,\Gamma}^{(i)} & A_{\Gamma,I}^{(i)} \\ A_{I,\Gamma}^{(i)} & A_{I,I}^{(i)} \end{pmatrix} \begin{pmatrix} u_\Gamma'^{(i)} \\ u_I'^{(i)} \end{pmatrix} = \begin{pmatrix} f_\Gamma'^{(i)} + g_\Gamma'^{(i)} \\ f_I'^{(i)} \end{pmatrix}, \quad i = 1, 2 \tag{14.70}$$

If we find solutions of these two subdomain problems with continuous normal derivative along the interface, i.e.,

$$g_\Gamma' := g_\Gamma'^{(1)} = -g_\Gamma'^{(2)} \tag{14.71}$$

then we have solved the full problem on Ω. But the correct value of g_Γ' is of course unknown.

The interface parts of the two subdomain solutions are given by

$$u_\Gamma'^{(i)} = S^{(i)^{-1}} \big(s_\Gamma'^{(i)} + g_\Gamma'^{(i)} \big), \quad \text{with} \quad s_\Gamma'^{(i)} = f_\Gamma'^{(i)} - A_{\Gamma,I}^{(i)} A_{I,I}^{(i)^{-1}} f_I'^{(i)}$$

where the $S^{(i)}$ are the Schur complements of the $A_{I,I}^{(i)}$. We require $u_\Gamma' := u_\Gamma'^{(1)} = u_\Gamma'^{(2)}$ (continuity along the interface). This gives the equation

$$\big(S^{(1)^{-1}} + S^{(2)^{-1}} \big) g_\Gamma' = S^{(2)^{-1}} s_\Gamma'^{(2)} - S^{(1)^{-1}} s_\Gamma'^{(1)} =: d_\Gamma' \tag{14.72}$$

for the determination of the normal derivative g_Γ' (see (14.71)) along the interface, which was sought for in order to obtain the solution of the full problem on Ω.

For preconditioning, the matrix $S^{(1)^{-1}} + S^{(2)^{-1}}$ in (14.72) is approximated by $(S^{(1)} + S^{(2)})^{-1}$, which gives the directly computable approximation

$$(S^{(1)} + S^{(2)})\, d_\Gamma' \approx g_\Gamma' \tag{14.73}$$

for the normal derivative along the interface. Evaluating (14.73) and solving the resulting pair of subdomain problems (14.70) with the corresponding approximation g_Γ' defines a preconditioner for the full problem posed on Ω.

This special way of using Dirichlet-Neumann maps for defining a preconditioner is the basis for general FETI-type multi-domain preconditioners.

Remark 14.9 The terminology 'FETI' stems from the fact that inexact solution of (14.72), i.e., using the preconditioner (14.73) instead, leads to a solution which is discontinuous along the interface ('tearing'). If we use the preconditioner to define a stationary iteration scheme, on convergence we have 'interconnected' this along the interface, giving the exact continuous solution. ∎

Remark 14.10 Various domain decomposition methods for constructing efficient preconditioners, in particular for complex geometries are the topic of many recent research activities, see [17],[18],[21]. The so-called BPX-preconditioner, for instance, is based on hierarchical domain decomposition, which is formally similar to multigrid techniques. ∎

15 Krylov Subspace Methods for Eigenvalue Problems

We consider the eigenvalue problem for a (large, sparse) symmetric[84] matrix $A \in \mathbb{R}^{n\times n}$. We assume that the eigenvalues $\lambda_i \in \mathbb{R}$ are ordered according to

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$

The Lanczos process (see Sec. 9.3) starting from some initial vector $z_0 \in \mathbb{R}^n$ yields an orthonormal basis $V_m \in \mathbb{R}^{n\times m}$ of the Krylov space $\mathcal{K}_m = \mathcal{K}_m(A, z_0)$ and the symmetric tridiagonal matrix

$$T_m = V_m^{\mathrm{T}} A V_m \in \mathbb{R}^{m\times m} \tag{15.1}$$

For m not too large, the eigenvalues $\mu_i^{(m)}$ of T_m, the so-called *Ritz values*, can be computed by standard methods, e.g., the QR algorithm. The Ritz values serve as approximations to the eigenvalues λ_i of A.

15.1 Auxiliary results

We make use of the following classical results.

Lemma 15.1 *Let $A \in \mathbb{R}^{n\times n}$ be symmetric. If for $\mu \in \mathbb{R}$ and $x \in \mathbb{R}^n$ there holds*

$$\|Ax - \mu x\|_2 = \varepsilon \|x\|_2, \quad \varepsilon > 0 \tag{15.2}$$

then there exists an eigenvalue λ of A with

$$|\lambda - \mu| \leq \varepsilon$$

Proof: W.l.o.g. we assume that $\mu \notin \sigma(A)$. Then, $A - \mu I$ is symmetric and invertible, and we have

$$\max_{\lambda\in\sigma(A)} \frac{1}{|\lambda - \mu|} = \|(A - \mu I)^{-1}\|_2 = \sup_{y\neq 0} \frac{\|(A - \mu I)^{-1} y\|}{\|y\|_2}$$

For $y = Ax - \mu x$ this gives

$$\max_{\lambda\in\sigma(A)} \frac{1}{|\lambda - \mu|} \geq \frac{\|(A - \mu I)^{-1} y\|_2}{\|y\|_2} = \frac{\|x\|_2}{\|Ax - \mu x\|_2} = \frac{1}{\varepsilon}$$

Thus, there exists $\lambda \in \sigma(A)$ with

$$\frac{1}{|\lambda - \mu|} \geq \frac{1}{\varepsilon}$$

which concludes the proof. □

Remark 15.1 A pair (μ, x) satisfying (15.2) is called a ε*-pseudo-eigenpair* (pseudo-eigenvalue, pseudo-eigenvector). ∎

Theorem 15.1 *[Courant-Fischer; variational characterization of eigenvalues] Let $A \in \mathbb{R}^{n\times n}$ be symmetric and $\{z_1, \ldots, z_n\} \subseteq \mathbb{R}^n$ be an orthonormal system, $\mathcal{Z}_k = \operatorname{span}\{z_1, \ldots, z_k\}$. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ denote the eigenvalues of A, with associated orthonormal eigenvectors $x_1, \ldots, x_n$. Then, for $k = 1 \ldots n$,*

(i)

$$\min_{0\neq z\in\mathcal{Z}_k} \frac{z^T A z}{z^T z} \leq \lambda_k$$

and (ii) equality holds for $\mathcal{Z}_k = \mathcal{X}_k = \operatorname{span}\{x_1, \ldots, x_k\}$, i.e., the maximum of all possible minima in (i) is λ_k.

Proof: See, e.g., [14]. □

[84] All assertions are also valid for $A \in \mathbb{C}^{n\times n}$ Hermitian.

15.2 Monotonicity of approximation

Theorem 15.2 *Let $\mu_1^{(k)} \geq \mu_2^{(k)} \geq \ldots \geq \mu_k^{(k)}$ be the eigenvalues of the matrices $T_k \in \mathbb{R}^{k\times k}$, $k = 1, 2, \ldots$. Then, for $i = 1 \ldots k$ the inequalities*

$$\lambda_{n-i+1} \leq \mu_{k+1-i+1}^{(k+1)} \leq \mu_{k-i+1}^{(k)} \qquad \text{and} \qquad \mu_i^{(k)} \leq \mu_i^{(k+1)} \leq \lambda_i \tag{15.3}$$

are valid.

Proof: Let

$$z_i^{(k)} = V_k\, u_i^{(k)}, \quad i = 1 \ldots k, \quad \text{where } u_i^{(k)} = \text{eigenvector of } T_k \text{ associated with eigenvalue } \mu_i^{(k)}, \ \|u_i^{(k)}\|_2 = 1$$

Since $\{u_1^{(k)}, \ldots, u_k^{(k)}\} \subseteq \mathbb{R}^k$ is an orthonormal system, the same is true for $\{z_1^{(k)}, \ldots, z_k^{(k)}\} \subseteq \mathbb{R}^n$ (recall that $V_k^{\mathrm{T}} V_k = I_k$). Let us denote $\mathcal{U}_i^{(k)} = \mathrm{span}\{u_1^{(k)}, \ldots, u_i^{(k)}\}$.

– *Step 1.* Claim:

$$\mu_i^{(k)} \leq \lambda_i, \quad i = 1 \ldots k$$

To prove this, we apply Theorem 15.1, (ii) to T_k:

$$\mu_i^{(k)} = \min_{0\neq u\in\mathcal{U}_i^{(k)}} \frac{u^{\mathrm{T}} T_k\, u}{u^{\mathrm{T}} u} \tag{15.4}$$

For $z = V_k\, u$ we have $z^{\mathrm{T}} z = u^{\mathrm{T}} u$, $z \in \mathcal{Z}_i^{(k)} = \mathrm{span}\{z_1^{(k)}, \ldots, z_i^{(k)}\}$, and $u^{\mathrm{T}} T_k\, u = u^{\mathrm{T}} V_k^{\mathrm{T}} A V_k\, u = z^{\mathrm{T}} A\, z$. Using (15.4) and applying Theorem 15.1, (i) to A we thus obtain

$$\mu_i^{(k)} = \min_{0\neq z\in\mathcal{Z}_i^{(k)}} \frac{z^{\mathrm{T}} A\, z}{z^{\mathrm{T}} z} \leq \lambda_i$$

for $i = 1 \ldots k$, as asserted.

– *Step 2.* Claim:

$$\mu_i^{(k)} \leq \mu_i^{(k+1)}, \quad i = 1 \ldots k$$

The proof uses a similar argument as in step 1. We observe that $V_k\, u = V_{k+1}\, \hat{u}$ with $\hat{u} = \begin{pmatrix} u \\ 0 \end{pmatrix} \in \mathbb{R}^{k+1}$.

We consider the corresponding set $\{\hat{u}_1, \ldots, \hat{u}_i\} \subseteq \mathbb{R}^{k+1}$, $\hat{\mathcal{U}}_i = \mathrm{span}\{\hat{u}_1, \ldots, \hat{u}_i\}$. From (15.4) we have

$$\mu_i^{(k)} = \min_{0\neq u\in\mathcal{U}_i^{(k)}} \frac{u^{\mathrm{T}} V_k^{\mathrm{T}} A V_k\, u}{u^{\mathrm{T}} u} = \min_{0\neq \hat{u}\in\hat{\mathcal{U}}_i} \frac{\hat{u}^{\mathrm{T}} V_{k+1}^{\mathrm{T}} A V_{k+1}\, \hat{u}}{\hat{u}^{\mathrm{T}} \hat{u}} = \min_{0\neq \hat{u}\in\hat{\mathcal{U}}_i} \frac{\hat{u}^{\mathrm{T}} T_{k+1}\, \hat{u}}{\hat{u}^{\mathrm{T}} \hat{u}} \leq \mu_i^{(k+1)}$$

where the last inequality again holds due to Theorem 15.1, (i). This concludes the proof of step 2.

Combining step 1 and step 2 shows the second assertion in (15.3). The first assertion is obtained from applying the same arguments to the matrix $-A$. □

Theorem 15.2 says that, for increasing dimension k of $\mathcal{K}_k$,

- the leftmost (smallest) eigenvalue of T_k is monotonously ↓ and ≥ the smallest eigenvalue of A,
- the second-leftmost (second-smallest) eigenvalue of T_k is monotonously ↓ and ≥ the second-smallest eigenvalue of A,

  ...

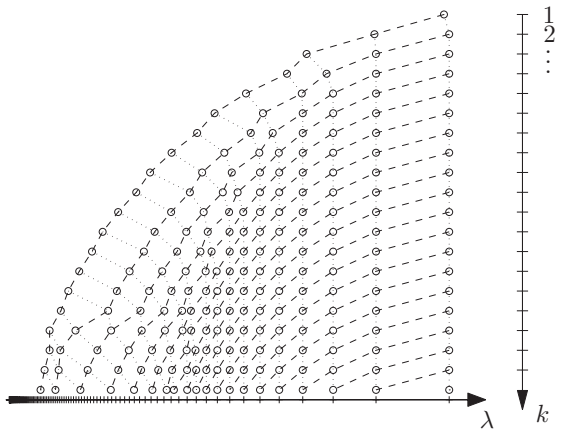


Figure 15.1: Ritz values and exact spectrum (Figure from [11]).

- the rightmost (largest) eigenvalue of T_k is monotonously $\uparrow$ and $\leq$ the largest eigenvalue of A,
- the second-rightmost (second-largest) eigenvalue of T_k is monotonously $\uparrow$ and $\leq$ the second-largest eigenvalue of A,
- ...

The theorem does not include an assertion concerning convergence. The accuracy of the Ritz values can be assessed in an a posteriori sense, however; see Theorem 15.3.

Example 15.1 Figure 15.1 illustrates the behavior of the Ritz values for $k = 1, 2, \ldots$ compared to the exact spectrum for the symmetric matrix $A \in \mathbb{R}^{100\times 100}$ with coefficients $a_{ij} = \pi^2 i(n-j+1)/(n+1)^3$ $(i \leq j)$. We see that the convergence behavior is quite good for the dominant, largest eigenvalues.

15.3 An a posteriori error estimate

Computable a posteriori estimates are of major practical relevance, in particular if a priori bounds are not available or too pessimistic. For the Lanzcos approximation to eigenvalues the following estimate is valid:

Theorem 15.3 *[a posteriori estimate] Let (μ, w) be an eigenpair of T_m with $\|w\|_2 = 1$, and let ω_m denote the last component of w, $\omega_m = e_m^T w$. Then, A has an eigenvalue λ satisfying*

$$|\lambda - \mu| \leq |\beta_{m+1}|\,|\omega_m|$$

where $\beta_{m+1}(= h_{m+1,m})$ is the lower right entry in the matrix $\bar{T}_m \in \mathbb{R}^{(m+1)\times m}$.

Proof: From identity (9.7) we have, in the notation from Sec. 9.3,

$$AV_m - V_m T_m = h_{m+1,m}\, v_{m+1}\, e_m^{\mathrm{T}} = \beta_{m+1}\, v_{m+1}\, e_m^{\mathrm{T}}$$

For $x = V_m w$ with $\|x\|_2 = \|w\|_2 = 1$ this gives[85]

$$Ax - \mu x = AV_m w - V_m(\mu w) = (AV_m - V_m T_m)\, w = \beta_{m+1}\, v_{m+1}\, e_m^{\mathrm{T}} w = \beta_{m+1}\, \omega_m\, v_{m+1}$$

Together with $\|v_{m+1}\|_2$ and $\beta_{m+1} > 0$ this leads to

$$\|Ax - \mu x\|_2 \leq |\beta_{m+1}|\,|\omega_m| = |\beta_{m+1}|\,|\omega_m|\,\|x\|_2$$

and the result follows from Lemma 15.1. □

[85] Like in other cases connected with Krylov methods, the residual is a multiple of v_{m+1}.

A The Finite Element Method (FEM) in a Nutshell

Starting from an abstract setting we will then mainly focus on the 2D Poisson equation (14.46) as a model problem. Most proofs are omitted or just indicated. A standard text on the topic is [3], for instance.

A.1 Elliptic bilinear forms and abstract variational problems

Let H denote a real Hilbert space with inner product $(\cdot,\cdot)$ and associated norm $\|\cdot\|$.

Definition A.1 *A* continuous (= bounded) bilinear form *on a Hilbert space H is a mapping $a(\cdot,\cdot): H\times H \to \mathbb{R}$ with the properties*

(i) *$a(u,\cdot)$ and $a(\cdot,u)$ is a continuous linear functional for each fixed $u \in H$,*

(ii) *$a(\cdot,\cdot)$ is bounded, i.e.*

$$|\,a(u,v)\,| \le M\,\|\,u\,\|\,\|\,v\,\| \qquad \textit{for all} \quad u,v \in H \tag{A.1}$$

A bilinear form $a(\cdot,\cdot)$ is called elliptic, *or* coercive, *if there exists a constant $\gamma>0$ with*

$$a(u,u) \ge \gamma\,\|\,u\,\|^2 \qquad \textit{for all} \quad u \in H \tag{A.2}$$

Evidently, $\gamma \le M$ holds true.

Lemma A.1 *Let the continuous bilinear form $a(\cdot,\cdot)$ be symmetric, i.e.,*

$$a(u,v) \equiv a(v,u) \tag{A.3}$$

Furthermore, let g be a linear functional on H. Then, $u_* \in H$ is a solution of the minimization problem

$$J(u) = \tfrac{1}{2}a(u,u) - g(u) \;\to\; \textit{min!} \tag{A.4}$$

iff u_* is the solution of the variational equation

$$a(u,v) = g(v) \qquad \textit{for all} \quad v \in H \tag{A.5}$$

Remark A.1

- For the finite-dimensional version of this assertions see (7.1).
- For the non-symmetric case, the *Lax-Milgram Lemma* states that the variational equation (A.5) still has a unique solution u_* satisfying

$$\|u_*\| \le \frac{1}{\gamma}\,\|g\|' \tag{A.6}$$

  where $\|\cdot\|'$ denotes the dual norm. See (1.2) for the finite-dimensional analogon of this assertion.

  However, this is not directly related to a minimization problem.

■

A.2 Weak derivatives and Sobolev spaces

Let[86] $\Omega \subseteq \mathbb{R}^2$ be an open domain with piecewise smooth boundary. The space $L^2(\Omega)$ of square-integrable functions on Ω is a Hilbert space with inner product and norm

$$(u,v)_{L^2(\Omega)} = \int_\Omega u\,v, \qquad \|u\|_{L^2(\Omega)} = \sqrt{(u,u)_{L^2(\Omega)}}$$

For derivatives of functions $u : \Omega \to \mathbb{R}$ we adopt the usual denotation in terms of a double index $\alpha = (\alpha_1, \alpha_2)$, $\alpha_i \geq 0$, with $|\alpha| = \alpha_1 + \alpha_2$. A function $D^\alpha u \in L^2(\Omega)$ is called the *weak derivative* of order α if

$$\int_\Omega D^\alpha u\, w = (-1)^{|\alpha|} \int_\Omega u\, D^\alpha w \quad \text{for all} \quad w \in C_0^\infty(\Omega)$$

The space of all functions u for which the weak derivatives up to order $|\alpha| = p$ exist is denoted by $H^p(\Omega)$ and is called a Sobolev space. It is a Hilbert space with inner product and norm

$$(u,v)_{H^p(\Omega)} = \sum_{|\alpha| \leq p} (D^\alpha u, D^\alpha v)_{L^2(\Omega)} \qquad \|u\|_{H^p(\Omega)} = (u,u)^{\frac{1}{2}}_{H^p(\Omega)}$$

For $p = 1$, for instance $H^1(\Omega)$ is equipped with inner product and norm

$$(u,v)_{H^1(\Omega)} = (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)}, \qquad \|u\|_{H^1(\Omega)} = (u,u)^{\frac{1}{2}}_{H^1(\Omega)},$$

where ∇ is to be understood in the weak sense.

The space of H^1-functions with zero trace on the boundary $\partial\Omega$ is denoted by $H_0^1(\Omega)$. On this space, the H^1-norm and the H^1-seminorm

$$|u|_{H^1(\Omega)} = \|\nabla u\|_{L^2(\Omega)}$$

can show to be equivalent (a consequence of the so-called Poincaré inequality).

The seminorm $|u|_{H^2(\Omega)}$ is defined in an analogous way.

Partial integration

Lemma A.2 *[partial integration in $\mathbb{R}^2$ (Green's formula)] For arbitrary $u \in C^2(\bar\Omega)$, $v \in C^1(\bar\Omega)$,*

$$\int_\Omega \Delta u\, v = \int_{\partial\Omega} \partial_n u\, v - \int_\Omega \nabla u \cdot \nabla v \tag{A.7}$$

where $\partial_n u$ denotes the normal derivative of u along $\partial\Omega$ with outward orientation.

A.3 The 2D Poisson equation; weak and variational formulation

Wir consider the homogeneous Dirichlet boundary value problem

$$-\Delta u(x) = f(x), \quad x \in \Omega, \tag{A.8a}$$

$$u(x) = 0, \qquad x \in \partial\Omega \tag{A.8b}$$

with $f \in L^2(\Omega)$. For the associated bilinear form

$$a(u,v) = (-\Delta u, v)_{L^2(\Omega)} = \int_\Omega -\Delta u\, v \tag{A.9}$$

[86] Analogous definitions in $\mathbb{R}^d$ for d arbitrary.

and sufficiently smooth u, v with zero boundary values, partial integration (A.7) gives

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \tag{A.10}$$

This motivates the more general, *weak formulation* of the given boundary value problem:

$$\textit{Find } u \in H_0^1(\Omega) \textit{ such that } \underbrace{\int_\Omega \nabla u \cdot \nabla v}_{a(u,v)} = \underbrace{\int_\Omega f\, v}_{(f,v)} \quad \textit{for all } v \in H_0^1(\Omega) \tag{A.11}$$

The following theorem is based on the fact that for the bilinear form (A.10) is elliptic and bounded (see Sec. A.1) on the space $H_0^1(\Omega)$ with norm $\|\cdot\|_{H^1(\Omega)}$; cf. e.g. [3].

Theorem A.1 *Problem* (A.11) *admits a unique solution* $u_* \in H_0^1(\Omega)$.

Moreover, since the bilinear form (A.10) is also symmetric, it satisfies the assumptions of Lemma A.1 with $H = H_0^1(\Omega)$ and $g(u) = (f,u)_{L^2(\Omega)}$.

Theorem A.2 *[Dirichlet principle] The unique solution* $u_* \in H_0^1(\Omega)$ *of* (A.11) *is characterized by the property that it is also the unique minimizer of the energy functional*

$$J(u) = \tfrac{1}{2} a(u,u) - (f,u)_{L^2(\Omega)} = \tfrac{1}{2} \int_\Omega \nabla u \cdot \nabla u - \int_\Omega f\, u \tag{A.12}$$

Regularity

Theorem A.3 *[H^2-regularity] If the domain* Ω *is convex or if the boundary* $\partial\Omega$ *is of class* C^2*, then the solution* u_* *of* (A.11) *is contained in* $H_0^1(\Omega) \cap H^2(\Omega)$ *and satisfies*

$$|u_*|_{H^2(\Omega)} \le C \|f\|_{L^2(\Omega)} \tag{A.13}$$

with a constant C *depending on* Ω.

A.4 Variational methods: Ritz, Galerkin

Consider the setting of Sec. A.1. Variational methods seek for an approximation of the solution u_* an n-dimensional linear subspace[87] $V_h \subseteq H$. For concrete realization one chooses a basis $\{v_1 \dots v_n\}$ of V_h, such that each $v_h \in V_h$ is uniquely represented as

$$v_h = \xi_1 v_1 + \ldots + \xi_n v_n$$

[87] In view of application to PDEs in the FEM context to be discussed in the sequel, the index h reprepresents the mesh width of the underlying discretization of the domain Ω, with $h \to 0$ as $n \to \infty$.

The Ritz approach.

Let $a(\cdot,\cdot)$ be symmetric. We seek for the minimizer of the energy functional $J(u)$ from (A.4) over V_h, and the corresponding element $u_h \in V_h$ satisfying

$$J(u_h) = \min_{v_h \in V_h} J(v_h) \tag{A.14}$$

is taken as an approximation for the exact solution u_*.

Due to

$$J(v_h) = J(\xi_1 v_1 + \ldots + \xi_n v_n) = \tfrac{1}{2} a\Big(\sum_{i=1}^n \xi_i v_i, \sum_{j=1}^n \xi_j v_j\Big) - g\Big(\sum_{j=1}^n \xi_j v_j\Big)$$
$$= \tfrac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \xi_i \xi_j\, a(v_i, v_j) - \sum_{j=1}^n \xi_j\, g(v_j).$$

Thus, minimization of $J(\cdot)$ over V_h is equivalent to minimization of the quadratic functional $\phi\colon \mathbb{R}^n \to \mathbb{R}$,

$$\phi(u) := \tfrac{1}{2}(Au, u) - (b, u) \tag{A.15}$$

over all[88] $u \in \mathbb{R}^n$, where

$$A = \begin{pmatrix} a(v_1,v_1) & \cdots & a(v_1,v_n) \\ \vdots & & \vdots \\ a(v_n,v_1) & \cdots & a(v_n,v_n) \end{pmatrix}, \qquad b = \begin{pmatrix} g(v_1) \\ \vdots \\ g(v_n) \end{pmatrix} \tag{A.16a}$$

with the so-called *stiffness matrix* $A \in \mathbb{R}^{n\times n}$. Due to the symmetry and ellipticity of $a(\cdot,\cdot)$, the stiffness matrix is SPD.

The unique minimizer of (A.15) is obtained as the solution of the SPD system

$$Au = b \tag{A.16b}$$

see (7.1). This approach is addressed as the *Ritz method.*

The Galerkin approach.

More generally, with $a(\cdot,\cdot)$ not necessarily symmetric, one starts from the variational problem (A.5) (in the PDE case this corresponds to the weak formulation, see (A.11)). We seek for $u_h \in V_h$ satisfying

$$a(u_h, v_h) = g(v_h) \quad \text{for all} \quad v_h \in V_h \tag{A.17}$$

This is equivalent to

$$a(u_h, v_j) = g(v_j), \quad j = 1 \ldots n, \tag{A.18a}$$

or with $u_h = \xi_1 v_1 + \ldots + \xi_n v_n$,

$$a\Big(\sum_{i=1}^n \xi_i v_i, v_j\Big) = \sum_{i=1}^n \xi_i\, a(v_i, v_j) = g(v_j), \quad j = 1 \ldots n \tag{A.18b}$$

In matrix-vector notation this results in the same system of equations as in (A.16) for the coefficient vector $u = (\xi_1, \ldots, \xi_n)^{\mathrm{T}}$. If $a(\cdot,\cdot)$ is not symmetric then A will also be unsymmetric. In the general case, the unique solvability of the system $Ax = b$ is a consequence of the Lax-Milgram Lemma (see Remark A.1); see also Assertion 8 from Sec. 1.3.

The following lemma relates the approximation error $u_h - u_*$ to the accuracy in which elements of H can be approximated by elements of the subspace V_h.

[88] Coefficient vectors in $\mathbb{R}^n$ associated with u_h are simply denoted by u.

Lemma A.3 *[Céa] Let $a(\cdot,\cdot)$ be a (not necessarily symmetric) bounded and elliptic bilinear form (see Def. A.1) on a Hilbertspace H and $V_h \subseteq H$ a linear subspace of H. Then the difference between the solutions $u_* \in H$ and $u_h \in V_h$ of the variational problems (A.17) and (A.5) satisfies*

$$\| u_h - u_* \| \leq \frac{M}{\gamma} \inf_{v_h \in V_h} \| v_h - u_* \| \tag{A.19}$$

Proof: With $V_h \subseteq H$ we have

$$a(u_*, v_h) = g(v_h) \quad \text{for all } v_h \in V_h$$

Due to (A.17) and with the linearity of $a(\cdot, v_h)$ this gives

$$a(u_h - u_*, v_h) = 0 \quad \text{for all } v_h \in V_h \qquad (\textit{Galerkin orthogonality}) \tag{A.20}$$

Furthermore,

$$a(u_h - u_*, u_h) = 0$$

This implies

$$\begin{aligned} a(u_h - u_*, u_h - u_*) &= a(u_h - u_*, u_h) - a(u_h - u_*, u_*) = \\ &= \quad 0 \quad - a(u_h - u_*, u_*) = \\ &= a(u_h - u_*, v_h) - a(u_h - u_*, u_*) = a(u_h - u_*, v_h - u_*) \end{aligned}$$

for all $v_h \in V_h$. Due to the continuity and ellipticity of $a(\cdot,\cdot)$ this implies

$$\gamma \| u_h - u_* \|^2 \leq M \| u_h - u_* \| \| v_h - u_* \| \quad \text{for all } v_h \in V_h$$

from which the estimate (A.19) immediately follows. □

A.5 FEM illustrated for the case of the 2D Poisson equation

We consider the Dirichlet problem for the 2D Poisson equation, see Sec. A.3. For the moment we assume that $\Omega = (0,1)^2$ is the unit square, and we consider the simplest variational approximation based on piecewise linear functions over a uniform *triangulation* $\mathcal{T}_h$ of Ω associated with a regular grid with meshwidth h and consisting of elements in the form of triangles, see Fig. A.1.

As approximating space we choose the finite-dimensional subspace $V_h \subseteq H_0^1(\Omega)$ which consists of continuous functions $v_h(x)$ linear[89] on each triangle and vanish on the boundary $\partial\Omega$. Each inner grid point $x_{i,j} = (i\,h, j\,h)$ is associated with a linear *hat function* $v_{i,j}(x)$ satisfying $v_{i,j}(x_{i,j}) = 1$, and $v_{i,j}(x_{k,\ell}) = 0$ for $(k,\ell) \neq (i,j)$. The support of $v_{i,j}$ is a so-called *patch* $\Omega_{i,j}$ consisting of 6 neighboring triangles.

[89] 'Higher-order elements' based on higher-degree polynomial ansatz functions are not considered here.

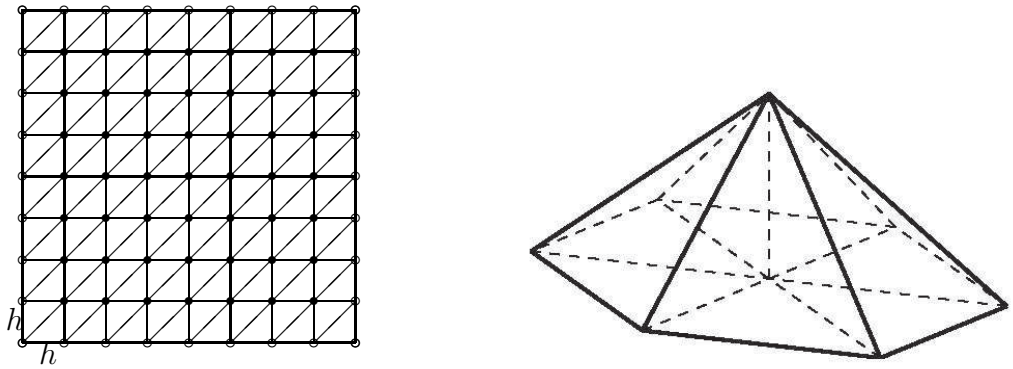


Figure A.1: Triangulation of a square; basis function (hat function)

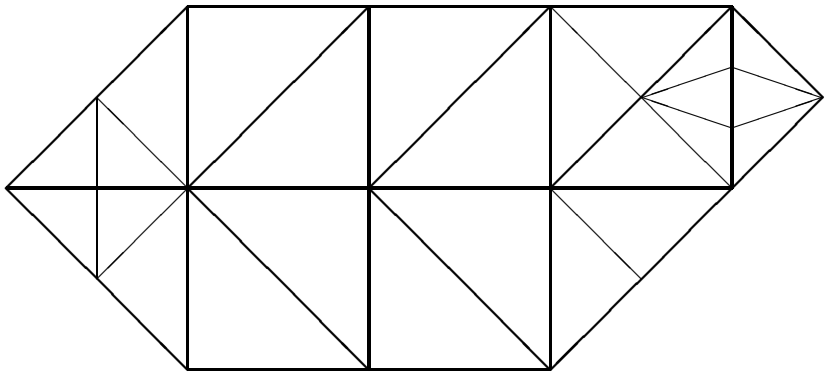


Figure A.2: Nonuniform admissible triangulation of a polygonal domain, with local refinement

These hat functions $v_{i,j}$ form a basis of V_h, the so-called *nodal basis.*

In this way we obtain a Galerkin system according to Sec. A.4. The coefficients $a(v_{i,j}, v_{k,\ell})$ in the 'Galerkin matrix' A from (A.16a) are given by

$$a(v_{i,j}, v_{k,\ell}) = \int_{\Omega_{i,j} \cap \Omega_{k,\ell}} \nabla v_{i,j} \cdot \nabla v_{k,\ell} \tag{A.21}$$

with piecewise constant integrands which are $\mathcal{O}(h^{-2})$ since the gradients involved are $\mathcal{O}(h^{-1})$. Due to local support with measure $\mathcal{O}(h^{-2})$ this results in a matrix A of small bandwith with entries of size $\mathcal{O}(1)$. In fact, after lexicographic ordering of grid points, up to a factor h^2 it is exactly the same matrix as for the standard FD discretization, see Example 2.2.[90]

The coefficients $b_{i,j}$ of the right-hand side b are obtained as

$$(f, v_{i,j})_{L^2(\Omega)} = \int_{\Omega} f\, v_{i,j} = \int_{\Omega_{i,j}} f\, v_{i,j} \tag{A.22}$$

In practice this has to be approximated by an appropriate quadrature formula.

The same procedure can be applied to the case of a general domain, and triangles (or other types of elements) of flexible size and shape. In general, an *admissible* triangulation $\mathcal{T}_h$ of means that[91] Ω is subdivided into a finite number of triangles in a way such that two triangles have either an empty intersection, a common edge, or a common vertex; see Fig. A.2. For such an admissible triangulation, the approximating subspace V_h and the associated nodal basis are well-defined.

The approximation results presented in the following section depend on the solution behavior as well as certain characteristics of the triangulation. For a triangle $T \subseteq \mathcal{T}_h$, let $\varrho(T)$ and $\varsigma(T)$ denote the diameters of its subscribed and superscribed circles, respectively. The parameter h is to be associated with $\max_{t \in \mathcal{T}_h} \varsigma(T)$. Furthermore, the condition

$$\max_{T \in \mathcal{T}_h} \frac{\varsigma(T)}{\varrho(T)} \leq C_{angle} \tag{A.23}$$

with a moderate-sized constant C, which can be interpreted as a *minimal angle condition*, plays an essential role. A triangulation for which this condition is satisfied is called *quasi-uniform.*

[90] Note the dependence on dimension: For problem dimension d the elements of the the stiffness matrix are $\mathcal{O}(h^{d-2})$.

[91] See Fig. Approximating curvilinear boundaries is an additional aspect.

A.6 FEM approximation properties

We assume that Ω is a polygonal domain and that $\mathcal{T}_h$ is an admissible triangulation of Ω such that $\Omega = \bigcup\{T : T \in \mathcal{T}_h\}$. The first step in a FEM convergence analysis ist to study the approximation properties of the subspace V_h associated with $\mathcal{T}_h$. The following theorem is based a scaling argument combined with an analysis of the corresponding approximation properties on a simple reference triangle.

Theorem A.4 *[local error estimate for piecewise linear FEM-interpolation in 2D] Let $T \in \mathcal{T}_h$ be a triangle, and let $\mathcal{I}_h u$ denote the piecewise linear interpolant of a function $u \in H^2(T)$ (interpolation an the vertices of T). Then the interpolation error $\mathcal{I}_h u - u$ satisfies the estimates*

$$\|\mathcal{I}_h u - u\|_{L^2(T)} \leq C h^2 |u|_{H^2(T)} \tag{A.24a}$$

$$|\mathcal{I}_h u - u|_{H^1(T)} \leq C h |u|_{H^2(T)} \tag{A.24b}$$

with a constant C independent of u but affected by the constant C_{angle} from the minimum angle condition (A.23).

Theorem A.4 implies the following global error estimate:

Corollary A.1 *[global error estimate for piecewise linear FEM-interpolation in 2D] Let $\Omega \subseteq \mathbb{R}^2$ be a bounded polygonal domain and $\mathcal{T}_h$ an admissible quasi-uniform triangulation of Ω. For $u \in H^2(\Omega)$, let $\mathcal{I}_h u$ denote its continuous piecewise linear interpolant over $\mathcal{T}_h$ (interpolation an the vertices of T). Then the interpolation error $\mathcal{I}_h u - u$ satisfies the estimate*

$$\|\mathcal{I}_h u - u\|_{H^1(\Omega)} \leq C h |u|_{H^2(\Omega)} \tag{A.25}$$

with a constant C independent of u but affected by the constant C_{angle} from the minimum angle condition (A.23).

Combination with the Céa Lemma (Lemma A.3) leads to the following convergence result. (Cf. Theorem A.3 for the underlying regularity requirement $u_* \in H^2(\Omega)$.)

Theorem A.5 *[FEM-convergence in 2D, a-priori error estimate for the Poisson equation] Let $\Omega \subseteq \mathbb{R}^2$ a convex polygonal domain and $\mathcal{T}_h$ an admissible, quasi-uniform triangulation of Ω. Then the FEM apporoximation error $u_h - u_*$ satisfies the estimate*

$$\|u_h - u_*\|_{H^1(\Omega)} \leq C h |u_*|_{H^2(\Omega)} \tag{A.26}$$

with a constant C independent of u but affected by the constant C_{angle} from the minimum angle condition (A.23).

By a more refined analysis based on a duality argument (the so-called Aubin-Nitsche Lemma) it also can be shown that an L^2 estimate of the form

$$\|u_h - u_*\|_{L^2(\Omega)} \leq C h^2 |u_*|_{H^2(\Omega)} \tag{A.27}$$

also holds true.

References

[1] P.R. Amestoy, T.A. Davis, and I.S. Duff: An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal.*, 17(4):886–905, 1996.

[2] W. Auzinger, O. Koch, D. Praetorius: Numerische Mathematik. Lecture Notes, TU Wien, 2016.

[3] S.C. Brenner and L.R. Scott: *The Mathematical Theory of Finite Element Methods*. Springer, 2002.

[4] W.K. Briggs, V.E. Henson, and S.F. McCormick: *A Multigrid Tutorial*. SIAM, 2000.

[5] I. Duff, A.M. Erisman, and J.K. Reid: *Direct Methods for Sparse Matrices*. Oxford Clarendon Press, 1992.

[6] M. Eiermann and O. Ernst: Geometric aspects of the theory of Krylov subspace methods. In A. Iserles, editor, *Acta Numerica 2001*, pages 251–312. Cambridge University Press, 2001.

[7] A. George: Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[8] A. George and J.W.H. Liu: *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[9] G.H. Golub and C.F. Van Loan: *Matrix Computations*. 4thed., John Hopkins University Press, 2013.

[10] W. Hackbusch: *Iterative Solution of Large Sparse Systems of Equations*, volume 95 of *Applied Mathematical Sciences*. Springer, 1994.

[11] M. Hanke-Bourgeois: *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, 3rd ed., Vieweg+Teubner Studium, 2009.

[12] A. Iserles: *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, 2nd ed., 2008.

[13] C. Kanzow: *Numerik linearer Gleichungssysteme. Direkte und iterative Verfahren*, Springer-Lehrbuch, 2005.

[14] P. Lancaster and M. Tismenetsky: *The Theory of Matrices*, Academic Press, 1984.

[15] C.D. Meyer: *Matrix Analysis and Applied Linear Algebra*. SIAM, 2000.

[16] G. Meurant: Gaussian elimination for the solution of linear systems of equations. In *Handbook of Numerical Analysis, Vol. VII*, pages 3–172. North-Holland, 2000.

[17] A. Quarteroni and A. Valli: *Domain Decomposition Methods for Partial Differential equations*. Oxford Science Publications, 1999.

[18] M.A. Olshanski and E.E. Tyrtyshnikov: *Iterative Methods for Linear Systems. Theory and Applications*. SIAM, 2014.

[19] Y. Saad: *Iterative Methods for Sparse Linear Systems*. 2nd ed., SIAM, 2003.

[20] W.F. Tinney and J.W. Walker: Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proc. of the IEEE*, 55:1801–1809, 1967.

[21] A. Toselli and O. Widlund: *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005.

[22] L.N. Trefethen, D. Bau: *Numerical Linear Algebra*. SIAM, 1997.

[23] W. Zulehner: *Numerische Mathematik. Eine Einführung anhand von Differentialgleichungsproblemen. Band 1: Stationäre Probleme*. Birkhäuser, 2008.