Iterative Methods and Preconditioners for Systems of Linear Equations

Gabriele Ciaramella and Martin J. Gander

September 24, 2020

Contents

Chapter 1

Introduction

Now, three-dimensional models are commonplace and iterative methods are almost mandatory. The memory and the computational requirements for solving three-dimensional Partial Differential Equations, or two-dimensional ones involving many degrees of freedom per point, may seriously challenge the most efficient direct solvers available today. Also, iterative methods are gaining ground because they are easier to implement efficiently on high-performance computers than direct methods.

Yousef Saad, Iterative Methods for Sparse Linear Systems, 2000.

There has been tremendous development in iterative methods and preconditioning over the past decades, and research in this area is more active than ever, with many conference series dedicated entirely to this subject, like the international conference on domain decomposition methods, the Copper Mountain multigrid and iterative methods conferences, the preconditioning conference series, and the European multigrid conferences. Iterative methods and preconditioning also form an important part of almost all conferences on numerical analysis these days. This is due to the tremendously increased computing power and the advent of massively parallel computers, and multi-core processors even in devices used in our daily lives, which permit simulations of unprecedented quality already on such systems.

There are many excellent books and monographs on iterative methods for linear systems of equations, from which we benefited tremendously in our research and preparing the courses we taught on this subject. An invaluable source is the reference text by *Yousef Saad* [111], who has himself made many seminal contributions to modern iterative methods, and who we quote frequently in this book. An outstanding reference for Krylov methods is the book by *Jörg Liesen* and *Zdeněk Strakoš* [85], with a focus on understanding the deep mathematics of the approximation properties of these methods. The book of *Anne Greenbaum* [64] has also been of great help preparing and understanding material for our lectures. We would finally also like to mention the templates book [2] which motivated us to give implementations of all the algorithms we discuss, in our case directly runnable in MATLAB.

Our book is an introduction to the historical development of iterative meth-

ods methods, up to the state of the art today, and it includes convergence analyses of all the methods we discuss. The material is based on the authors own experience and research with these methods, and permits a rapid entry in this exciting field of research. To limit the scope, the focus is entirely on iterative methods for the solution of linear systems, with the approximate solution of discretized partial differential equations in mind.

1.1 Motivation

> *We move from direct methods, a classical topic that is rather thoroughly understood, to the relatively untamed territory of iterative methods. These are the methods that seem likely to dominate the large-scale computations of the future.*
>
> Lloyd N. Trefethen and David Bau, Numerical Linear Algebra, Lecture 32, SIAM 1997.

With the advent of modern computers, the fundamental difficulty of problems in mathematics has changed dramatically, because of algorithms that compute solutions iteratively. Let us consider two simple examples:

1. For a linear system of equations

$$A\boldsymbol{u} = \boldsymbol{f}, \quad \text{with } A \in \mathbb{R}^{n\times n} \text{ and } \boldsymbol{f} \in \mathbb{R}^n \text{ given},$$

we want to compute the solution $\boldsymbol{u} \in \mathbb{R}^n$. The classical approach is to use *Gaussian elimination* named after *Carl Friedrich Gauss* from 1798[1], which systematically eliminates the first unknown in the second up to the last equation, and then the second unknown in the third to last equation and so on. This leads to the *LU-factorization* of the matrix A, see for example [57, Chapter 3], which was also already discovered by Gauss in 1820[2]. Instead of solving the linear system, one can then solve in two steps $LU\boldsymbol{u} = \boldsymbol{f}$, see also [57, Chapter 3]: setting $\boldsymbol{v} := U\boldsymbol{u}$, one first solves using *forward substitution* the triangular system

$$L\boldsymbol{v} = \boldsymbol{f},$$

for $\boldsymbol{v}$, followed by solving by *backward substitution*

$$U\boldsymbol{u} = \boldsymbol{v},$$

two triangular solves. For a matrix of size $n \times n$, the cost of computing the LU-factorization is $O(n^3)$, and the forward and backward substitutions cost $O(n^2)$ floating point operations. This method gives in principle the exact solution (up to round-off error) to the linear system of equations in a finite number of steps; one could not imagine a more simple problem in mathematics.

[1] Gaussian elimination was already known to Chinese mathematicians as early as 179 CE, and can be found in Newton's notes from 1670, and in Lagrange's work on Lagrange multipliers from 1788.

[2] Note that it took Gauss more than twenty years to realize this important and non-trivial interpretation!

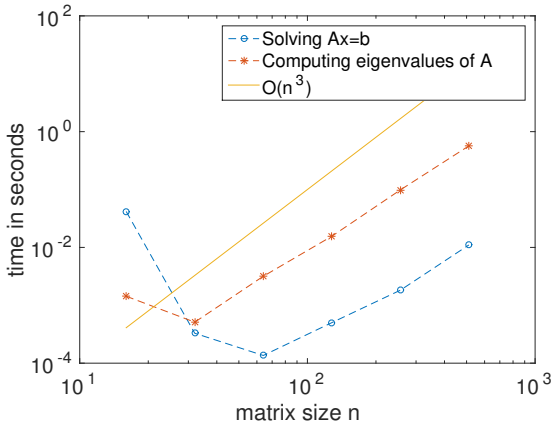


Figure 1.1: Comparison of computing times for solving linear systems and obtaining the eigenvalues of the associated matrix, as a function of the matrix size

2. Let us now consider the problem of finding the *eigenvalues* $\lambda_j \in \mathbb{C}$ and *eigenvectors* $\boldsymbol{\phi}_j \in \mathbb{C}^n$ of a given matrix $A \in \mathbb{R}^{n\times n}$,

$$A\boldsymbol{\phi}_j = \lambda_j \boldsymbol{\phi}_j, \quad j = 1, 2, \ldots, n.$$

Here there is no algorithm which can give us the exact solution after a finite number of steps, since the eigenvalues are the roots of the *characteristic polynomial* $p(\lambda) := \det(A - \lambda I)$, I the identity matrix, and it is well known by Galois theory that there exists no closed formula to express the roots of a polynomial of degree $n > 4$. Nevertheless, one can obtain the eigenvalues and eigenvectors to arbitrary accuracy in $O(n^3)$ operations using an iterative method, namely the *QR algorithm*, see for example [57, Chapter 7].

So while from a pure mathematical standpoint of view, solving a system of linear equations is trivial, and obtaining the eigenvalues of a matrix is impossible in closed form, there is little difference to obtain the actual solution. We compare in Figure 1.1 the cost in MATLAB to compute the solution of random linear systems and the eigenvalues of the associated matrix. The figure was obtained with the MATLAB script

```
n=2.^(4:9);                                          % matrix sizes
T1=zeros(1,length(n)); T2=zeros(1,length(n));        % for storing times
for j=1:10                                           % do 10 experiments
  for i=1:length(n)                                  % to average times
    A=rand(n(i));
    t0=clock; B=lu(A); t1(i)=etime(clock,t0);        % compute LU
    t0=clock; E=eig(A); t2(i)=etime(clock,t0);       % compute eigenvalues
  end;
  T1=T1+t1; T2=T2+t2;                                % sum times
end
```

```
loglog(n,T1/length(n),'--o',n,T2/length(n),'--*',n,n.^3/10000000,'-');
legend('Solving Ax=b','Computing eigenvalues of A','O(n^3)',2);
xlabel('matrix size n');
ylabel('time in seconds');
set(gca,'fontsize',20);                                  % bigger font size
```

We see that indeed both problems have the same computational complexity $O(n^3)$, the eigenvalue problem just has a larger constant.

1.2 Gauss and Jacobi

> *Schwerlich werden Sie je wieder direct eliminieren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.*
>
> Carl F. Gauss, in a letter to his friend Christian L. Gerling, 1823.

> *Die Beschwerlichkeit der strengen Auflösung einer grösseren Zahl linärer Gleichungen, auf welche in vielen Fällen die Methode der kleinsten Quadrate führt, hat an die Anwendung von Näherungsmethoden denken lassen.*
>
> Carl. G. J. Jacobi, Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden lineären Gleichungen, 1845.

Iterative methods for linear systems of equations have a long history. An overview of selected key events is shown in Figure 1.2, and we start with the first main inventions by *Carl Friedrich Gauss* and *Carl Gustav Jacob Jacobi*.

On December 26, 1823, Gauss sent a letter to his friend Gerling [58] to explain how he computed an approximate *least squares* solution based on angle measurements between the locations Berger Warte, Johannisberg, Taufstein and Milseburg. The system is symmetric, see Figure 1.3; it comes from the normal equations, and Gauss explains (translation by *Forsythe* [40]):

> "In order to eliminate indirectly, I note that, if 3 of the quantities a, b, c, d are set to 0, the fourth gets the largest value when d is chosen as the fourth. Naturally, every quantity must be determined from its own equation, and hence d from the fourth. I therefore set $d = -201$ and substitute this value. The absolute terms then become: $+5232$, -6352, $+1074$, $+46$; the other terms remain the same."

In this first step, Gauss sets as approximation $a = b = c = 0$ and then uses the fourth equation to compute an approximation to d,

$$d = -\frac{22156}{110} \approx -201.$$

Assuming that $d = -201$ now, he can compute an updated right-hand side which he calls "absolute terms",

$$\begin{aligned} 6 - 26d &\approx 5232, \\ -7558 - 6d &\approx -6352, \\ -14604 - 78d &\approx 1074, \\ 22156 + 110d &\approx 46, \end{aligned}$$

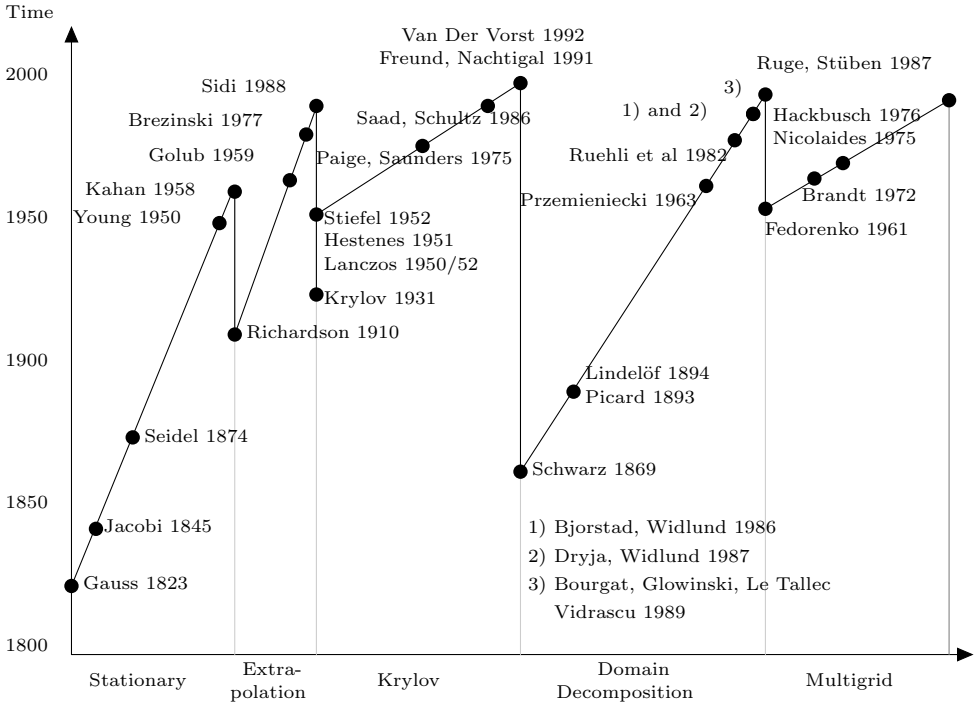


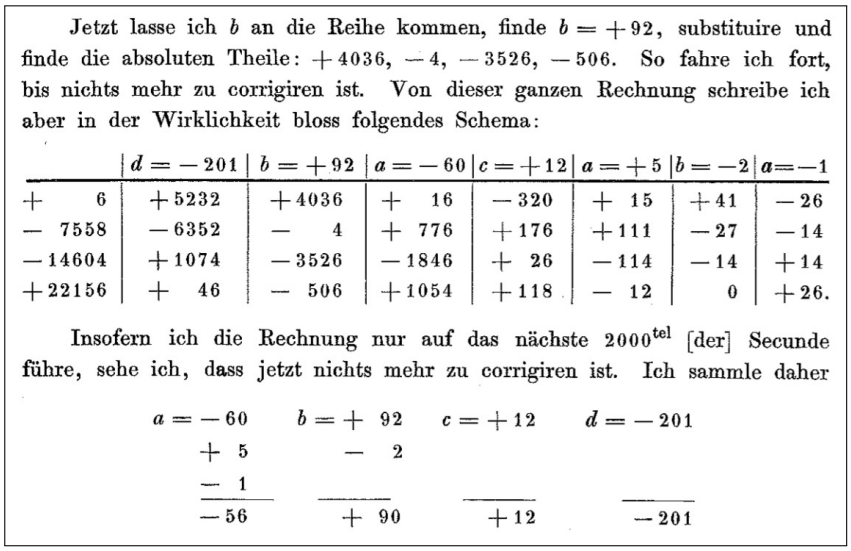
Figure 1.2: An overview of key events in the historical development of iterative methods for linear systems of equations, and how they will appear throughout the text.

Die Bedingungsgleichungen sind also:

$$
\begin{aligned}
0 &= +\ 6 + 67a - 13b - 28c - 26d \\
0 &= -\ 7558 - 13a + 69b - 50c - 6d \\
0 &= -\ 14604 - 28a - 50b + 156c - 78d \\
0 &= +\ 22156 - 26a - 6b - 78c + 110d; \\
&\text{Summe} = 0.
\end{aligned}
$$

Um nun indirect zu eliminiren, bemerke ich, dass, wenn 3 der Grössen a, b, c, d gleich 0 gesetzt werden, die vierte den grössten Werth bekommt, wenn d dafür gewählt wird. Natürlich muss jede Grösse aus ihrer eigenen Gleichung, also d aus der vierten, bestimmt werden. Ich setze also $d = -201$ und substituire diesen Werth. Die absoluten Theile werden dann: $+5232$, -6352, $+1074$, $+46$; das Übrige bleibt dasselbe.

Figure 1.3: Letter of Gauss from 1823 explaining the first step of his new method to solve linear equations by the indirect method.



Jetzt lasse ich b an die Reihe kommen, finde $b = +92$, substituire und finde die absoluten Theile: $+4036$, -4, -3526, -506. So fahre ich fort, bis nichts mehr zu corrigiren ist. Von dieser ganzen Rechnung schreibe ich aber in der Wirklichkeit bloss folgendes Schema:

	$d = -201$	$b = +92$	$a = -60$	$c = +12$	$a = +5$	$b = -2$	$a = -1$
$+6$	$+5232$	$+4036$	$+16$	-320	$+15$	$+41$	-26
-7558	-6352	-4	$+776$	$+176$	$+111$	-27	-14
-14604	$+1074$	-3526	-1846	$+26$	-114	-14	$+14$
$+22156$	$+46$	-506	$+1054$	$+118$	-12	0	$+26.$

Insofern ich die Rechnung nur auf das nächste 2000^{tel} [der] Secunde führe, sehe ich, dass jetzt nichts mehr zu corrigiren ist. Ich sammle daher

$a = -60$	$b = +92$	$c = +12$	$d = -201$
$+5$	-2		
-1			
-56	$+90$	$+12$	-201

Figure 1.4: Continuation of the letter of Gauss from 1823 to his friend Gerling.

and we see that also the last equation does not have a right-hand side zero, it is not necessary to compute d exactly. Gauss now continues as described in Figure 1.4. With the system with the new right-hand side, he searches again for the variable with the biggest contribution when setting the other ones to zero, and finds that it is b in this case, which leads to $b = \frac{6352}{69} \approx 92$. He computes again a new right-hand side and continues the procedure, saying that of all these computations he only keeps the table in Figure 1.4. He stops computing corrections with $a \approx -1$, and then sums for each variables all correction steps, which gives as approximate solution $a \approx -56$, $b \approx 90$, $c \approx 12$, $d \approx -201$. We can check these computations using the MATLAB script

```
A=[67 -13 -28 -26                  % example of Gauss
   -13 69 -50 -6
   -28 -50 156 -78
   -26 -6 -78 110];
f=-[6                              % original right hand side
  -7558
  -14604
   22156];
id=[4 2 1 3 1 2 1];                % sequence of variables used
D=diag(A);
fn=f;
u=zeros(4,1);                      % to sum corrections
F=[];
for i=1:length(id)                 % compute correction steps
  d=zeros(4,1);
  d(id(i))=round(fn(id(i))/D(id(i)))
  u(id(i))=u(id(i))+d(id(i));      % add correction
```

```
  fn=fn-A*d;                    % update the right-hand side
  F=[F -fn]                     % collect updated right-hand sides
  pause
end
u                               % approximate solution of Gauss
uexact=A\f                      % compare with Matlab solution
[A*u-f A*uexact-f]              % Gauss and Matlab solution residual
fm=f-u(4)*A(:,4);               % compute Matlab solution with same 4th
uc=A(1:3,:)'\fm;                % component as Gauss' solution
uc(4)=u(4);
[u uc]                          % compare again
```

The solution computed with backslash in MATLAB gives however the warning

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate. RCOND =  2.277381e-17.
> In exampleGauss at 26
xexact =
   17.5190
  163.3949
   85.1300
 -128.0000
```

and it is quite different from the solution computed by Gauss. So what happened here? We did so far not pay attention to an important comment of Gauss in Figure 1.3, namely 'Summe=0', which means that summing all equations we obtain zero, the system is *singular*, but consistent, and thus has many solutions. The MATLAB script above also computes the solution with the same last component as the one Gauss found, and we see then that the solution of Gauss is very accurate from the MATLAB output

```
ans =
  -56.0000  -55.4810
   90.0000   90.3949
   12.0000   12.1300
 -201.0000 -201.0000
```

Gauss concludes his letter with the statement in Figure 1.5 (translation by Forsythe [40]):

> "Almost every evening I make a new edition of the tableau, wherever there is easy improvement. Against the monotony of the surveying business, this is always a pleasant entertainment; one can also see immediately whether anything doubtful has crept in, what still remains to be desired, etc. I recommend this method to you for imitation. You will hardly ever again eliminate directly, at least not when you have more than 2 unknowns. The indirect procedure can be done while half asleep, or while thinking about other things."

> Fast jeden Abend mache ich eine neue Auflage des Tableaus, wo immer leicht nachzuhelfen ist. Bei der Einförmigkeit des Messungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man sieht dann auch immer gleich, ob etwas zweifelhaftes eingeschlichen ist, was noch wünschenswerth bleibt, etc. Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.

Figure 1.5: Gauss explains how relaxing these relaxations are.

So instead of watching TV or surfing on the Internet, it would be better for all of us to do some relaxing relaxations in the evening to find approximate solutions to linear systems of equations[3]!

The following MATLAB function attempts to solve a general linear system of equations precisely using the method described by Gauss, i.e. always choosing the most promising variable to do an update.

```
function u=Gauss(A,f,maxiter,tol,Rnd)
% GAUSS original iterative method by Gauss
%   u=Gauss(A,f,iter,tol,Rnd) attempts to solve the linear
%   system A*u=f using the iterative method originally proposed by
%   Gauss. The parameter maxiter limits the maximum number of
%   iterations, if the norm of the right hand side obtained compared
%   to the original one does not go below the parameter tol before.
%   Rnd is a flag to turn the rounding Gauss used on (Rnd=1) or off
%   (Rnd=0).

m=length(f);
D=diag(A);
fn=f;                              % keep original right hand side
u=zeros(m,1);                      % to sum corrections
i=1;
while norm(fn)/norm(f)>tol & i<=maxiter
  [du,id]=max(abs(fn./D));         % find biggest contribution
  d=zeros(m,1);
  d(id)=fn(id)/D(id);
  if Rnd, d(id)=round(d(id)); end; % round if desired
  u=u+d;                           % add correction
  fn=fn-A*d;                       % update the right-hand side
  i=i+1;
end
i
```

[3]The first author, assistant of the course "Méthodes Itératives" in Geneva, in an email to the second author, two weeks into the course: "Following the suggestion "do some relaxation calculus to relax in the evening", :-) I computed the iteration matrix of the alternating Schwarz method for three holes, or in another picture the iteration matrix of a modified method of reflections".

This function allows us to reproduce the example of Gauss, and also to test the method for arbitrary linear problems $A\boldsymbol{u} = \boldsymbol{f}$. To reproduce the Gauss example, we can now simply call `Gauss(A,f,7,1e-6,1)` with the original matrix A and right hand side vector $\boldsymbol{f}$ from the Gauss example, to find the same result. If we try more iterations, for example `Gauss(A,f,10,1e-6,1)`, the result however does not become more accurate, because we reached the level of accuracy one can obtained using rounding, the method *stagnates*. Turning rounding off, `Gauss(A,f,10,1e-6,0)`, we obtain

```
ans =
  -56.0383
   89.8419
   11.5599
 -201.5456
```

and we can indeed reach a more and more accurate solution by reducing the tolerance and increasing the maximum number of iterations, e.g.

```
u=Gauss(A,f,10,1e-12,0);
norm(f-A*u)
u=Gauss(A,f,20,1e-12,0);
norm(f-A*u)
u=Gauss(A,f,40,1e-12,0);
norm(f-A*u)
```

which leads to the residual norms `3.9411, 0.0080, 4.8106e-08`. As a further example, we consider the symmetric positive definite matrix

$$A = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \quad \text{and} \quad \boldsymbol{f} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Using the MATLAB commands

```
A=[4 -1 0 0;
   -1 4 -1 0;
   0 -1 4 -1;
   0 0 -1 4];
f=ones(4,1);
Gauss(A,f,30,1e-6,0)
A\f
```

we see that the method works very well. Trying however random matrices,

```
A=rand(5);
f=ones(5,1);
u=Gauss(A,f,100,1e-12,0)
u-A\f
```

the method almost always fails, the difference between the approximation `u` obtained and the exact solution `A\f` is very large. If we try however symmetric

positive definite matrices, which we can construct from random matrices by multiplying by their transpose, the method seems to always work, provided we do enough iterations,

```
A=rand(5);
A=A'*A;
f=ones(5,1);
u=Gauss(A,f,10000,1e-12,0)
u-A\f
```

About half a century after Gauss, in 1874, a general description of this iterative method was given by *Ludwig von Seidel* [114] for the normal equations, who also proved convergence of the method in this case[4], and proposed to do the relaxations cyclically, instead of always choosing the equation that gives the largest contribution. Seidel finally even suggested to distribute the computations to two computers (humans) to do parallel computing[5]. We will study the convergence properties of the Gauss method in Section 2.7.

Before Seidel's general description of the method of Gauss however, *Jacobi* presented in 1845 a variant of the Gauss method now known as the Jacobi method [77][6]. We show in Figure 1.6 how Jacobi imagined the first iteration step of the method. Note how modern the notation already is: matrix entries we denote today by a_{ij} are denoted by (ij), and the vector of unknowns is $\boldsymbol{x} = (x, x_1, x_2, \ldots)$, where we now use $\boldsymbol{x} = (x_1, x_2, x_3, \ldots)$. Also the right-hand side is using vector notation, $\boldsymbol{b} = (0m, 1m, 2m, \ldots)$, where we nowadays use $\boldsymbol{b} = (b_1, b_2, b_3, \ldots)$. So in modern notation, the system in Figure 1.6 reads

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots &= b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots &= b_3, \\ &\vdots \end{aligned}$$

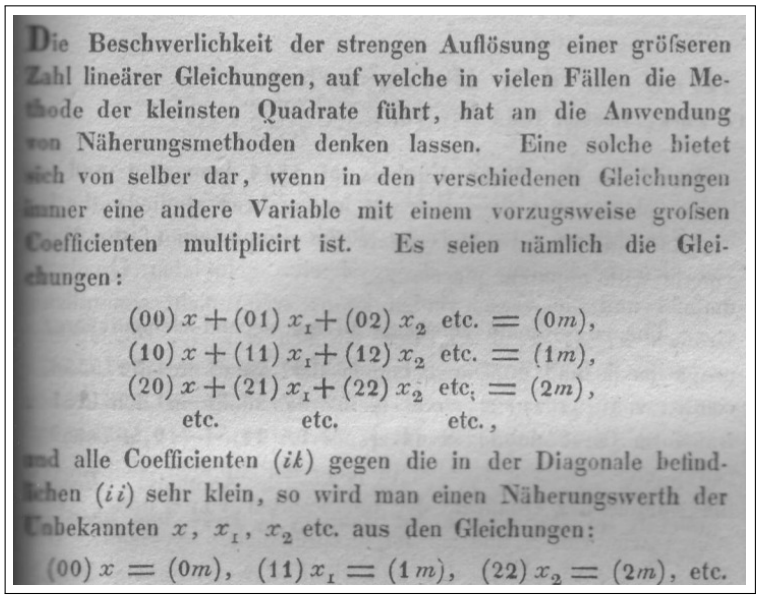
Jacobi then says right below the linear system in Figure 1.6 that if the off diagonal coefficients a_{ik} are all small compared to the a_{ii} on the diagonal, then a good approximation to the solution of the linear system can already be obtained by just solving the diagonal system, i.e.

$$x_1 \approx \frac{b_1}{a_{11}},\ x_2 \approx \frac{b_2}{a_{22}},\ x_3 \approx \frac{b_3}{a_{33}},\ \ldots\ .$$

[4] and considered this to be the end of research needed on iterative methods, because all systems can be transformed into a system of normal equations: "Wohl aber kann man auch jedes beliebige System linearer Gleichungen mit gleich vielen Unbekannten in die Normalform (B) bringen, genau nach der auf die Gleichungen (A) angewendeten Vorschrift, und aus dieser Form ist es nach unserer Methode ganz ebenso auflösbar, wie die aus Beobachtungen abgeleiteten Normalgleichungen."

[5] "... sich unter zwei Rechner so vertheilen lässt ..."

[6] Seidel was a student of Jacobi, and Jacobi thanks him for computations he had performed for him: "Man wird dort aus den von einem meiner gelehrten Freunde, Herrn Dr. Seidl in München, mit grosser Sorgfalt geführten Rechnungen ersehen,..."



Die Beschwerlichkeit der strengen Auflösung einer gröfseren Zahl lineärer Gleichungen, auf welche in vielen Fällen die Methode der kleinsten Quadrate führt, hat an die Anwendung von Näherungsmethoden denken lassen. Eine solche bietet sich von selber dar, wenn in den verschiedenen Gleichungen immer eine andere Variable mit einem vorzugsweise grofsen Coefficienten multiplicirt ist. Es seien nämlich die Gleichungen:

$$(00)\,x + (01)\,x_1 + (02)\,x_2 \text{ etc.} = (0m),$$
$$(10)\,x + (11)\,x_1 + (12)\,x_2 \text{ etc.} = (1m),$$
$$(20)\,x + (21)\,x_1 + (22)\,x_2 \text{ etc.} = (2m),$$
$$\text{etc.} \quad \text{etc.} \quad \text{etc.},$$

und alle Coefficienten (ik) gegen die in der Diagonale befindlichen (ii) sehr klein, so wird man einen Näherungswerth der Unbekannten x, x_1, x_2 etc. aus den Gleichungen:

$$(00)\,x = (0m), \quad (11)\,x_1 = (1m), \quad (22)\,x_2 = (2m), \text{ etc.}$$

Figure 1.6: Main idea of Jacobi inventing a different iterative method to solve linear systems of equations.

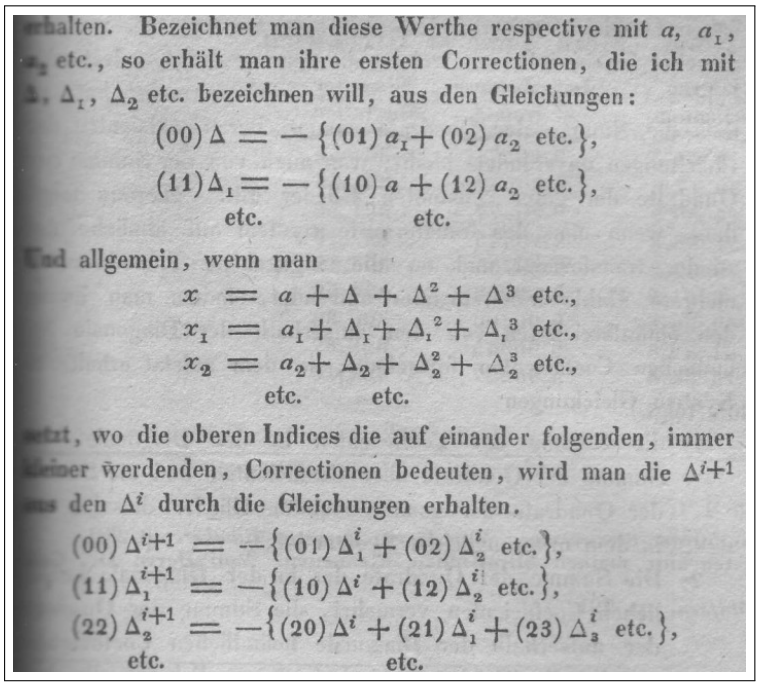
If these approximations are not good enough, Jacobi then explains, see Figure 1.7, how one can obtain the first corrections to improve the approximations, by computing

$$\begin{aligned}\Delta_1 &= \tfrac{1}{a_{11}}(-a_{12}x_2 - a_{13}x_3 - \ldots),\\ \Delta_2 &= \tfrac{1}{a_{22}}(-a_{21}x_1 - a_{23}x_3 - \ldots),\\ &\vdots\end{aligned}$$

and in general better and better approximations can be obtained by setting

$$\begin{aligned}x_1 &\approx \tfrac{b_1}{a_{11}} + \Delta_1 + \Delta_1^2 + \Delta_1^3 + \ldots,\\ x_2 &\approx \tfrac{b_2}{a_{22}} + \Delta_2 + \Delta_2^2 + \Delta_2^3 + \ldots,\\ x_3 &\approx \tfrac{b_3}{a_{33}} + \Delta_3 + \Delta_3^2 + \Delta_3^3 + \ldots,\\ &\vdots\end{aligned}$$

see again Figure 1.7, where now the corrections Δ_j^i are computed by the itera-



halten. Bezeichnet man diese Werthe respective mit a, a_1, a_2 etc., so erhält man ihre ersten Correctionen, die ich mit Δ, Δ_1, Δ_2 etc. bezeichnen will, aus den Gleichungen:

$$(00)\,\Delta = -\{(01)\,a_1 + (02)\,a_2 \text{ etc.}\},$$
$$(11)\,\Delta_1 = -\{(10)\,a + (12)\,a_2 \text{ etc.}\},$$
etc. etc.

Und allgemein, wenn man

$$x = a + \Delta + \Delta^2 + \Delta^3 \text{ etc.},$$
$$x_1 = a_1 + \Delta_1 + \Delta_1^2 + \Delta_1^3 \text{ etc.},$$
$$x_2 = a_2 + \Delta_2 + \Delta_2^2 + \Delta_2^3 \text{ etc.},$$
etc. etc.

setzt, wo die oberen Indices die auf einander folgenden, immer kleiner werdenden, Correctionen bedeuten, wird man die Δ^{i+1} aus den Δ^i durch die Gleichungen erhalten.

$$(00)\,\Delta^{i+1} = -\{(01)\,\Delta_1^i + (02)\,\Delta_2^i \text{ etc.}\},$$
$$(11)\,\Delta_1^{i+1} = -\{(10)\,\Delta^i + (12)\,\Delta_2^i \text{ etc.}\},$$
$$(22)\,\Delta_2^{i+1} = -\{(20)\,\Delta^i + (21)\,\Delta_1^i + (23)\,\Delta_3^i \text{ etc.}\},$$
etc. etc.

Figure 1.7: Jacobi writing the method directly as a modern iterative method with iteration index i.

tion[7]

$$\begin{aligned}
\Delta_1^{i+1} &= -\tfrac{1}{a_{11}}(a_{12}\Delta_2^i + a_{13}\Delta_3^i - \ldots),\\
\Delta_2^{i+1} &= -\tfrac{1}{a_{22}}(a_{21}\Delta_1^i + a_{23}\Delta_3^i - \ldots),\\
\Delta_3^{i+1} &= -\tfrac{1}{a_{33}}(a_{31}\Delta_1^i + a_{32}\Delta_2^i - \ldots),\\
&\vdots
\end{aligned}$$

Note in Figure 1.7 how groundbreaking modern notation Jacobi used, writing a general iterative method with iteration index i for the corrections.

Even though Gauss said that one will hardly ever use direct elimination for systems bigger than two by two, nowadays direct elimination is commonly used for linear systems with many thousands of unknowns. Linear systems which really benefit from iterative methods are systems where the size is such that it is an approximation to infinity, and we present now a typical example of such a system.

[7]Notice that Jacobi stated his algorithm in a difference form: the iterations are defined in terms of Δ_j^k that can be computed as $\Delta_j^k = x_j^k - x_j^{k-1}$, where the approximation of the solution at the iteration k is $x_j^k = \frac{b_j}{a_{jj}} + \sum_{\ell=1}^{k} \Delta_j^\ell$. The difference form of an iterative method is discussed in Section 2.1.

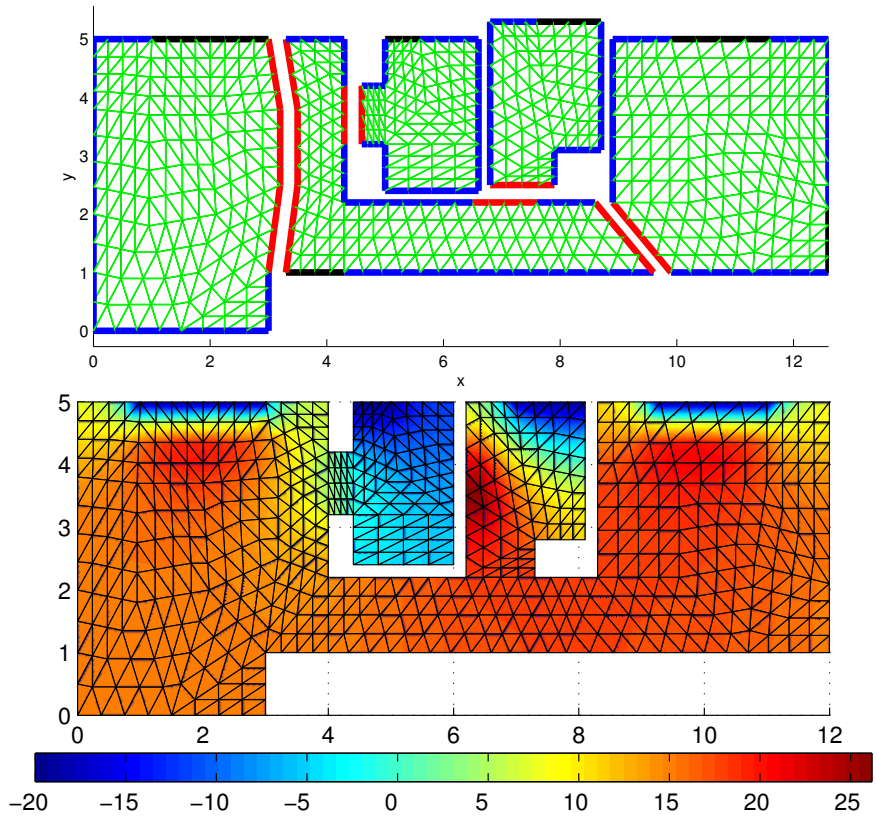


Figure 1.8: Top: discretization of the apartment 208 on 3421 Durocher, Montreal with the rooms slightly separated. Bottom: temperature distribution on a cold winter day.

1.3 Laplace's equation as a typical example

> *The title of this book, Matrix Iterative Analysis, suggests that we might consider here all matrix numerical methods which are iterative in nature. However, such an ambitious goal is in fact replaced by the more practical one where we seek to consider in some detail that smaller branch of numerical analysis concerned with the efficient solution, by means of iteration, of matrix equations arising from discrete approximations to partial differential equations.*
>
> *Richard S. Varga*, Matrix Iterative Analysis, 1962

The solution of *partial differential equations* by *discretization* often leads to large and *sparse linear systems* of equations, and these are ideally suited for iterative methods, as Varga already emphasized in his seminal book [124], see also the quote above. A first example is shown in Figure 1.8, from [43]. In the top panel, we see a two dimensional cross section of the apartment 208 on 3421 Durocher, Montreal, where the Ganders were living. The floor plan is shown with a finite element discretization, see e.g. [50] for more on the finite element method, and a decomposition into the different rooms (see Section 4.6

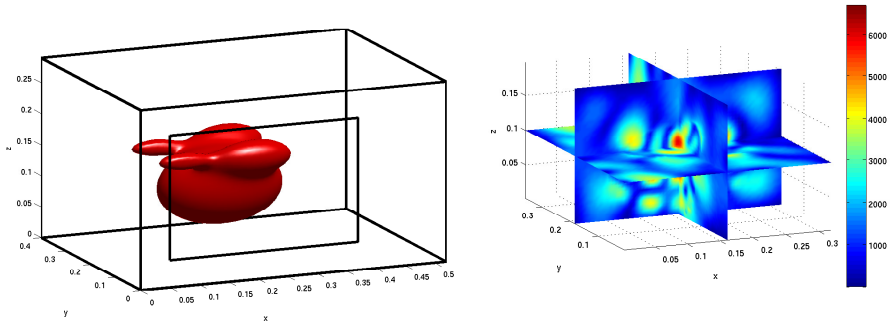


Figure 1.9: Heating a chicken in our Whirlpool Talent Combi 4 microwave oven from our times in Paris. Left: the geometry of the microwave and the chicken. Right: the electric field intensity in the chicken.

for more on domain decomposition): on the left is the living room, connected to the kitchen and with a long hallway to the bathroom and bedroom on the right. Insulated walls are shown in blue, the windows on top are shown in black, where we assume -20 degrees Celsius for a regular Montreal winter day, and the doors at the bottom and on the right are also shown in black. They lead to a heated public hallway, at about 15 degrees Celsius. In the bottom panel, we show the solution of the corresponding steady state heat equation we will introduce shortly, representing the temperature distribution: one can see that while the heaters in the living room on the left and the bedroom on the right are well placed to block the cold from the windows, the heater on the left wall in the bathroom is not effective to keep the room warm, a fact the Ganders strongly felt in winter. Also, the kitchen is not heated and stays cold, except when cooking and baking.

A second example, trying to cook a chicken in a microwave oven [34], is show in Figure 1.9. From the magnetic field intensity computed on the right by solving the *Maxwell's equations* in 3d, one can see why a turntable is so important in a microwave oven: there are hot spots, where the intensity of the standing wave is high in the chicken, and other areas, where there is very little heating happening, because the electric field is close to zero. Using a turntable which turns the food steadily can avoid hot spots and lead to an approximately even heating of the chicken. Another possibility in modern microwave ovens is to change the frequency of the source periodically, which also moves the hot spots around.

Let us now get back to the *heat equation* of the first example, and let us consider the propagation of heat in an enclosed space, e.g., a room in a building. The temperature is a function of time t and space $\boldsymbol{x}$, which we denote by $u(t, \boldsymbol{x})$. Now how does the total amount of heat in a given volume V evolve? The *Fourier law of heat transfer*, which was discovered experimentally by *Jean-Baptiste Joseph Fourier* in 1822, and was known to Jean-Baptiste Biot math-

ematically already in 1804, says that the heat flux $\boldsymbol{F}$ is proportional to the gradient of the temperature, $\boldsymbol{F} = -\nu\nabla u$ for some constant ν we will assume for simplicity to equal one, $\nu = 1$, and the minus sign is because heat always moves from warm to cold, not the other way round. If in the volume V there is no heat source or sink, then the total amount of heat in the volume can only be increased or diminished by the heat which flows through its boundary ∂V, and we thus must have

$$\frac{\partial}{\partial t}\int_V u(\boldsymbol{x},t)d\boldsymbol{x} = -\int_{\partial V}\boldsymbol{F}(\boldsymbol{x},t)\cdot\boldsymbol{n}\,ds = \int_{\partial V}\nabla u(\boldsymbol{x},t)\cdot\boldsymbol{n}\,ds, \tag{1.1}$$

where $\boldsymbol{n}$ is the unit outward normal on the boundary ∂V of the volume V, and $d\boldsymbol{s}$ denotes a surface element for the integration on ∂V. Using now the *Divergence Theorem of Gauss* from 1813 to transform the surface integral on the right into a volume integral, we get

$$\frac{\partial}{\partial t}\int_V u(\boldsymbol{x},t)d\boldsymbol{x} = \int_V \nabla\cdot\nabla u(\boldsymbol{x},t)d\boldsymbol{x}, \tag{1.2}$$

and noticing that the divergence of the gradient is the *Laplacian*[8],

$$\nabla\cdot\nabla = \begin{bmatrix}\partial_x\\ \partial_y\\ \partial_z\end{bmatrix}\cdot\begin{bmatrix}\partial_x\\ \partial_y\\ \partial_z\end{bmatrix} = \partial_x^2+\partial_y^2+\partial_z^2 = \Delta,$$

and exchanging time differentiation and integration, we obtain

$$\int_V \frac{\partial}{\partial t}u(\boldsymbol{x},t) - \Delta u(\boldsymbol{x},t)d\boldsymbol{x} = 0. \tag{1.3}$$

Equation (1.3) must hold for any arbitrary volume V, and thus the term under the integral sign must vanish identically, which leads to the *heat equation*,

$$u_t = \Delta u. \tag{1.4}$$

This is the equation which describes the evolution of temperature in a domain Ω, and to solve it, one needs to know what the initial temperature is, $u(\boldsymbol{x},0) = u^0(\boldsymbol{x})$, and also the temperature on the boundary of the domain, $u(\boldsymbol{x},t) = g(\boldsymbol{x},t)$ for $\boldsymbol{x}\in\partial\Omega$. If we are only interested in the stationary equilibrium, then for $t\to\infty$ we have $u_t = 0$ and the equation simplifies to

$$\begin{array}{rcll}\Delta u &=& 0 & \text{in } \Omega,\\ u &=& g & \text{on } \partial\Omega.\end{array} \tag{1.5}$$

Equation (1.5) is called *Laplace's equation*, and we derived it considering heat transfer like Fourier in 1822. *Laplace* however discovered this equation already in 1785 considering the attraction of celestial bodies, when he generalized Newton's inverse square law from 1687 for point masses to objects having non-zero volume. Laplace's equation turns out to be fundamental in many areas of physics and mathematics:

[8] We use for partial derivatives interchangeably $\frac{\partial}{\partial x}u = \partial_x u = u_x$.

- theory of *the attraction of celestial bodies* (Laplace 1785);
- theory of *stationary heat transfer* (Fourier 1822);
- theory of *magnetism* (Gauss and Weber in Göttingen 1839);
- theory of *electric fields* (W. Thomson, later Lord Kelvin 1847, Liouville 1847);
- *conformal mappings* (Gauss 1825);
- irrotational *fluid motion* in 2D (Helmholtz 1858);
- in *complex analysis* (Cauchy 1825, Riemann Thesis 1851).

To solve Laplace's equation (1.5), one needs to find a function such that it equals g on the boundary $\partial\Omega$, and when one computes the second partial derivatives with respect to each spatial variable and sums them, one obtains zero. It is very difficult to find such a function, and in general one can only obtain an approximation. To do so, let us consider a simple two dimensional example, where the domain Ω is the unit square, the stationary heat distribution is a function of two variables, $u(x, y)$, and the boundary function g is as given in Figure 1.10. To obtain an approximate solution, we introduce a grid with grid size $h = \frac{1}{m+1}$, for a $m \in \mathbb{N}^+$, and search for an approximation u_{ij} of $u(x, y)$ only at the grid points. Since it is natural to think in column vectors when solving linear systems, we order the unknowns column-wise, so $u_{ij} \approx u(jh, ih)$. The solution stored in the matrix u_{ij} can be pictured like it is shown in Figure 1.10, one only has to remember that the first element in the top left corner of the matrix u_{ij} is in the bottom left corner of the figure, and the y coordinate is increasing from bottom to top, like one is used to. This ordering is also used in Matlab for its plotting and meshing commands like `mesh`, `surf`, `reshape` and `numgrid`, and will greatly simplify its use. The *Laplace operator* in two dimensions,

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

is discretized using finite differences, see Problem 1:

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

and similarly

$$\frac{\partial^2 u}{\partial y^2}(x, y) \approx \frac{u(x, y + h) - 2u(x, y) + u(x, y - h)}{h^2}.$$

Introducing these approximations into (1.5) and multiplying by the common factor h^2, we obtain for the approximation at point (jh, ih) the equation

$$u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1} = 0. \tag{1.6}$$

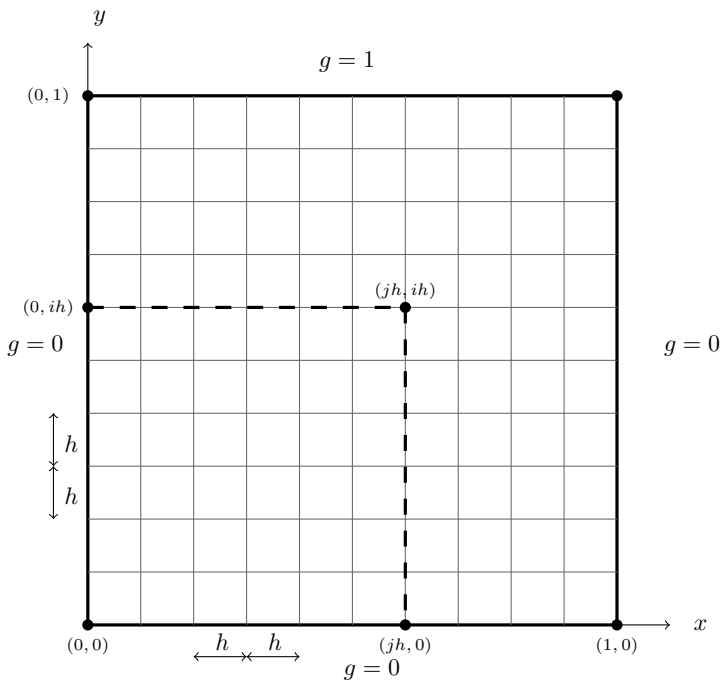


Figure 1.10: Discretization grid used to approximate the Laplace equation (1.5). Notice that the domain is discretized with $m+2$ points in both x and y directions. The grid size is $h = \frac{1}{m+1}$ and each point of the grid is of the form $(x_j, y_i) = (jh, ih)$ for $j, i = 0, 1, \ldots, m, m+1$. The function g defined on the boundary $\partial\Omega$ represents the boundary function that we use as Dirichlet boundary condition.

This equation reveals an interesting property of the approximate solution: at each point of the grid, it is precisely the average of the values at its four neighboring points,

$$u_{i,j} = \frac{1}{4}\big(u_{i,j-1} + u_{i-1,j} + u_{i+1,j} + u_{i,j+1}\big), \tag{1.7}$$

a very remarkable fact of the discrete Laplacian in (1.6). This implies for example that the approximation inside the domain Ω can neither have a local maximum, nor a local minimum, since otherwise it could not be the average of its neighbors. This is known as the *discrete maximum principle* for this so called five-point finite difference discretization of the Laplace equation. We now introduce the vectors

$$\boldsymbol{u}_j = \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{m,j} \end{bmatrix}, \quad j = 1, 2, \ldots, m,$$

which contain the values $u_{i,j}$ for $i = 1, \ldots, m$ on a vertical column corresponding

to j in Figure 1.10. Writing the equation (1.6) for each point i in the column, we obtain

$$\begin{array}{rcll} -4u_{1,j} + u_{2,j} + u_{1,j-1} + u_{1,j+1} &=& 0, & i = 1, \\ u_{1,j} - 4u_{2,j} + u_{3,j} + u_{2,j-1} + u_{2,j+1} &=& 0, & i = 2, \\ & \vdots & & \vdots \\ u_{m-2,j} - 4u_{m-1,j} + u_{m,j} + u_{m-1,j-1} + u_{m-1,j+1} &=& 0, & i = m-1, \\ u_{m-1,j} - 4u_{m,j} + u_{m,j-1} + u_{m,j+1} &=& -u_{m+1,j}, & i = m, \end{array}$$

where we put the value $u_{m+1,j}$ into the right-hand side, because its value is known to equal 1 from the boundary condition imposed at the top. Writing this for $j = 1, \ldots, m$ as a vector equation, we get

$$\boldsymbol{u}_{j-1} + T\boldsymbol{u}_j + \boldsymbol{u}_{j+1} = -\boldsymbol{e}_m, \tag{1.8}$$

where the unit vector $\boldsymbol{e}_m = \begin{bmatrix} 0, \ldots, 0, 1 \end{bmatrix}^\top$ appears because of the top boundary condition, where $u = 1$, and the tridiagonal matrix T is given by

$$T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{bmatrix}.$$

If we now introduce the vector of all unknowns

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_m \end{bmatrix}$$

and write the equations (1.8) all stacked on top of one another, then we obtain a large *sparse linear system* of equations with $n = m^2$ equations for n unknowns,

$$A\boldsymbol{u} = \boldsymbol{f},$$

with the block tridiagonal matrix

$$A = \begin{bmatrix} T & I & & \\ I & T & \ddots & \\ & \ddots & \ddots & I \\ & & I & T \end{bmatrix}, \tag{1.9}$$

and the right-hand side vector $\boldsymbol{f}$ containing the boundary conditions. We show in Figure 1.11 the solution we obtain using the MATLAB script

```
m=20;                          % number of gridpoints
A=Laplacian(m,2);              % five point Laplacian
```

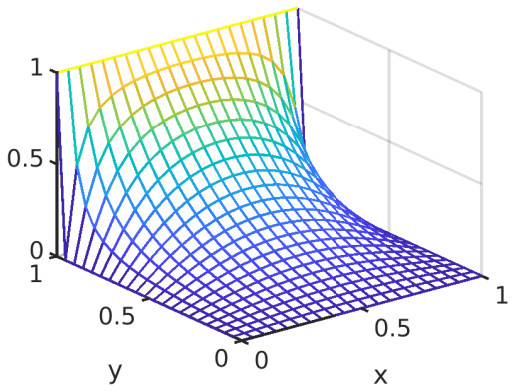


Figure 1.11: Approximate solution of Laplace's equation with Dirichlet boundary condition equal 1 at the top and 0 on the other boundaries.

```
f=zeros(m*m,1);                    % construct right hand side
f(m:m:end)=-1;
u=A\f;                             % solve by sparse Gaussian elimination
U=zeros(m+2);                      % fill solution into 2d
U(2:m+1,2:m+1)=reshape(u,m,m);     % fills matrix column-wise
U(end,1:m+2)=1;                    % add boundary condition
x=0:1/(m+1):1; y=x;                % mesh point vectors
mesh(x,y,U);
xlabel('x'); ylabel('y');          % labeling
```

in which we use the following function `Laplacian` that needs to be placed in the file `Laplacian.m`,

```
function A=Laplacian(m,d)
% LAPLACIAN compute a finite difference Laplacian
%   A=Laplacian(m,d) computes a finite difference Laplacian on the
%   unit interval/square/cube (d=1,2,3) using m interior points

if d==2
  G=numgrid('S',m+2); A=-delsq(G); % Matlab way to construct 2D Laplacian
else
  error('not implemented yet, see Problem 2')
end;
```

and which generates the finite difference matrix from our construction. This function will be completed in Problem 2 to also include discrete Laplacians in other spatial dimensions. We see in Figure 1.10 how nicely smooth the approximate temperature distribution links the boundary temperature 1 to the boundary temperature 0. Note also that the `reshape` command in Matlab automatically puts the long solution vector `u` in the correct ordering into the matrix `U` in the form u_{ij} as we designed it, and the `mesh` command plots it in

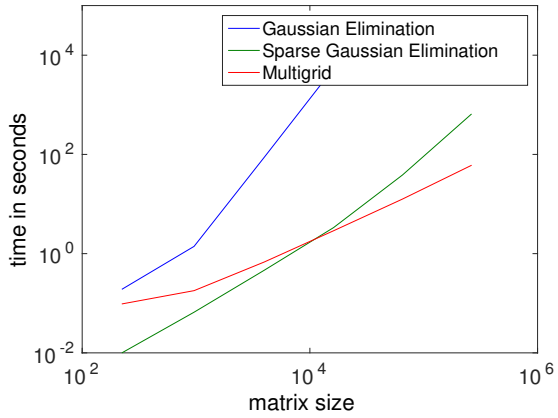


Figure 1.12: Solving Laplace's equation discretized by finite differences using Gaussian elimination, a sparse variant of Gaussian elimination, and multigrid.

the orientation of the plot in Figure 1.11 we are used to see. Every other choice of ordering would make this reshape and plotting very cumbersome.

The matrix A is sparse, with only 5 nonzero elements per row, and has a regular structure. If we assume that $m = 100$, then we have $n = 10'000$ unknowns and 100 million matrix entries, of which only $50'000$ (0.05%) are nonzero. This is precisely such a system where the size n is basically an approximation to infinity, an infinitely fine grid. We show in Figure 1.12 a comparison of the solution cost of such systems using *Gaussian elimination*, a sparse variant thereof, and the best currently known iterative method for such type of problems, called *multigrid*. We see that as soon as the problem becomes large, the iterative multigrid algorithm is faster, and we also observe that asymptotically the growth of the iterative method cost is much slower than the growth for the direct methods based on Gaussian elimination.

Multigrid is based on a very simple iterative method and a fundamental observation we can already show now: from the fact we observed in (1.7) that the approximation at each point is the average of its neighbors, a simple iterative method is to start with an arbitrary initial guess for the solution, and then at each point to replace the value by the average of its neighbors, i.e. starting with some $u^0_{i,j}$, we compute for all points simultaneously at each iteration $k = 1, 2, \ldots$

$$u^k_{i,j} = \frac{1}{4}\big(u^{k-1}_{i,j-1} + u^{k-1}_{i-1,j} + u^{k-1}_{i+1,j} + u^{k-1}_{i,j+1}\big), \tag{1.10}$$

which is in fact precisely the method Jacobi had proposed. We show in Figure 1.13 the first four iterations for grid sizes $m = 2, 4, 8, 16$. These results were obtained with the simple MATLAB script

```
m=2;                                % setup Laplace equation as before
A=Laplacian(m,2);
f=zeros(m*m,1); f(m:m:end)=-1;
```

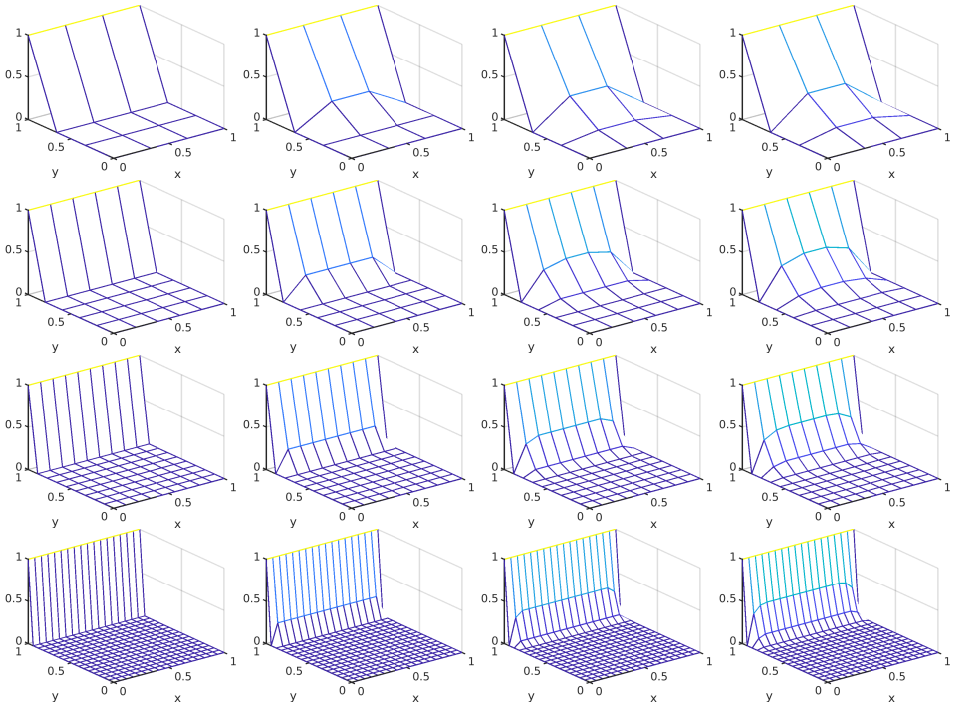


Figure 1.13: Initial guess and first three iteration of Jacobi's method for Laplace's equation for increasing mesh resolution.

```
x=0:1/(m+1):1;
u=zeros(size(f)); U=zeros(m+2);
U(end,1:m+2)=1;
for k=1:4                              % do four Jacobi steps
  U(2:m+1,2:m+1)=reshape(u,m,m);
  mesh(x,x,U);                         % plot current approximation
  xlabel('x');ylabel('y');
  u=u+(f-A*u)./diag(A);                % perform one Jacobi step
  pause
end
```

We see that for few grid points, $m = 2$, after just a few steps the approximate solution has the correct shape, as the high resolution approximation in Figure 1.11. When m becomes larger however, it seems to take more and more steps of the Jacobi iteration to produce a good approximation of the solution, and for $m = 16$, after 3 steps one still has the approximate solution zero in most of the domain. This is not surprising, since starting with a zero initial guess $u^0_{i,j} = 0$ at all points, the average of the neighboring points will remain zero as long as a point does not touch the boundary where the solution equals 1. The key idea of the multigrid method is to use a very coarse grid, e.g. $m = 2$, to get very quickly a good first approximation, and to then use this as an initial guess on

a finer grid, e.g. $m = 4$, to do a few corrections, and so on. Doing this leads to the very fast iterative method tested in Figure 1.12.

1.4 Problems

Problem 1. *Consider a uniform grid Ω_h in $\mathbb{R}^2$ with mesh size h such that $(x_j, y_i) \in \Omega_h$ with $x_{j+1} = x_j + h$ and $y_{i+1} = y_i + h$. Using Taylor series, show that the scheme of finite differences for the discrete Laplacian defined on Ω_h given by*

$$\frac{u(x_{j+1}, y_i) + u(x_j, y_{i+1}) - 4u(x_j, y_i) + u(x_{j-1}, y_i) + u(x_j, y_{i-1})}{h^2},$$

is a second-order approximation of the Laplace operator.

Problem 2. *Write a* MATLAB *function that takes as input the dimension of the grid and the spatial dimension of the domain and returns the matrix corresponding to the discrete Laplacian. Use the interface and header*

```
function A=Laplacian(m,d)
% LAPLACIAN compute a finite difference Laplacian.
%   A=Laplacian(m,d) computes a finite difference Laplacian on the
%   unit interval/square/cube (d=1,2,3) using m interior points in
%   each direction.
```

Hint: for two spatial dimensions, you could use solution shown already in this chapter, but it is better to develop a solution using the Kronecker product and the matlab function `kron`*, because then the generalization to higher dimensions is very easy.*

Problem 3. *Consider the Laplace equation*

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega = (0,1) \times (0,1), \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

where $g(0, y) = \sin(\pi y)$, $g(x, 0) = 0$, $g(x, 1) = 0$, $g(1, y) = 0$.

- *Prove that the solution to this problem is $u(x, y) = \sin(\pi y)\frac{e^{-\pi x} - e^{\pi(x-2)}}{1-e^{-2\pi}}$.*
- *Discretize the above Laplace problem using finite differences.*
- *Write a* MATLAB *code to solve the obtained discrete Laplace problem.*
- *Solve the discrete Laplace problem for several h and plot the L^2-error to obtain the Figure 1.14.*

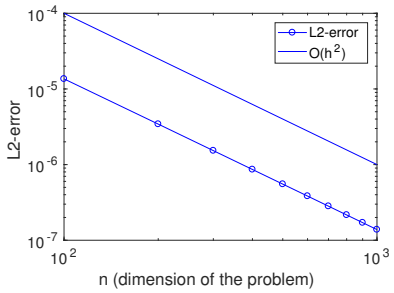


Figure 1.14: Convergence of the finite-differences scheme in the discrete L^2-norm.

Chapter 2

Stationary Iterative Methods

> *Let us pass on to methods which have this property in common: that starting from a table of numbers, correct at the boundary, but otherwise merely as near as one can guess, one proceeds by definite methods to modify this table and thereby to cause it to approach without limit towards the true finite-difference integral.*
>
> Lewis F. Richardson, The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam, 1911

Richardson describes above how he imagined iterative methods for a discretized partial differential equations to function: they start with some given approximation, and then proceed by applying precisely prescribed rules to improve this approximation until a good enough approximation is achieved. A very general way to describe such an iterative method for solving the linear system of equations

$$A\boldsymbol{u} = \boldsymbol{f}, \quad A \in \mathbb{R}^{n\times n},\ \boldsymbol{f} \in \mathbb{R}^n,$$

is to *split the matrix* into $A = M - N$. If M is invertible, this splitting induces the *stationary iteration*

$$M\boldsymbol{u}_{k+1} = N\boldsymbol{u}_k + \boldsymbol{f}, \quad k = 0, 1, 2, \ldots, \tag{2.1}$$

where an initial guess $\boldsymbol{u}_0 \in \mathbb{R}^n$ is needed to start the iteration. The method is called stationary, because neither M nor N depends on the iteration count k.

To obtain an efficient method, the *matrix splitting* must be such that solving linear systems with the matrix M is cheap, and the iteration (2.1) converges fast. These are unfortunately in general conflicting requirements, as one can see from the extreme case where we chose $M = A$, and thus $N = 0$, and we converge in one iteration to the solution, but this iteration is as expensive as solving the underlying system.

2.1 Error, residual, and difference of iterates

To each of the three iterative methods described, we can associate error vectors $\boldsymbol{e}^{(m)}$ *defined by ...*
Richard S. Varga, Matrix Iterative Analysis, 1962

Using the matrix splitting $A = M - N$, the *standard form* of the stationary iteration is

$$M\boldsymbol{u}_{k+1} = N\boldsymbol{u}_k + \boldsymbol{f} \iff \boldsymbol{u}_{k+1} = M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f}. \tag{2.2}$$

Since the right hand side in (2.2) satisfies

$$M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f} = M^{-1}(M - A)\boldsymbol{u}_k + M^{-1}\boldsymbol{f} = \boldsymbol{u}_k + M^{-1}(\boldsymbol{f} - A\boldsymbol{u}_k),$$

we can write the stationary iteration also in the so called *correction form*,

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + M^{-1}\boldsymbol{r}_k, \tag{2.3}$$

where we introduced the *residual* $\boldsymbol{r}_k := \boldsymbol{f} - A\boldsymbol{u}_k$, which is a measure of how good the approximation $\boldsymbol{u}_k$ is. The matrix M is called a *preconditioner*, since by iterating with (2.3), if $\boldsymbol{u}_k$ converges to $\boldsymbol{u}_\infty$, then $\boldsymbol{u}_\infty$ is solution of the *preconditioned system*

$$M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f}.$$

If we subtract the iteration from the split system, we get

$$\left.\begin{array}{rcl} M\boldsymbol{u} & = & N\boldsymbol{u} + \boldsymbol{f} \\ M\boldsymbol{u}_{k+1} & = & N\boldsymbol{u}_k + \boldsymbol{f} \end{array}\right\} \Rightarrow M(\boldsymbol{u} - \boldsymbol{u}_{k+1}) = N(\boldsymbol{u} - \boldsymbol{u}_k).$$

Introducing the *error* $\boldsymbol{e}_k := \boldsymbol{u} - \boldsymbol{u}_k$, we obtain an *iteration for the error*,

$$M\boldsymbol{e}_{k+1} = N\boldsymbol{e}_k \iff \boldsymbol{e}_{k+1} = M^{-1}N\boldsymbol{e}_k. \tag{2.4}$$

Multiplying the correction form (2.3) with A and subtracting the result from the right hand side $\boldsymbol{f}$, we obtain

$$\underbrace{\boldsymbol{f} - A\boldsymbol{u}_{k+1}}_{\boldsymbol{r}_{k+1}} = \underbrace{\boldsymbol{f} - A\boldsymbol{u}_k}_{\boldsymbol{r}_k} - AM^{-1}\boldsymbol{r}_k,$$

and an *iteration for the residual vectors*,

$$\boldsymbol{r}_{k+1} = (I - AM^{-1})\boldsymbol{r}_k = (I - AM^{-1})^k\boldsymbol{r}_0. \tag{2.5}$$

Therefore, recalling that $A = M - N$, we obtain

$$I - AM^{-1} = NM^{-1} = M(M^{-1}N)M^{-1},$$

which shows that the iteration matrices $I - AM^{-1}$ in (2.5) and $M^{-1}N$ in (2.4) are similar, and hence must have the same eigenvalues.

We now introduce also the *difference of consecutive iterates*,

$$\boldsymbol{d}_k := \boldsymbol{u}_{k+1} - \boldsymbol{u}_k.$$

Using the standard form of the stationary iteration (2.2), we get the *iteration for the differences*

$$\begin{aligned} \boldsymbol{d}_k &= \boldsymbol{u}_{k+1} - \boldsymbol{u}_k = M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f} - M^{-1}N\boldsymbol{u}_{k-1} - M^{-1}\boldsymbol{f} \\ &= M^{-1}N(\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \\ &= M^{-1}N\boldsymbol{d}_{k-1}. \end{aligned}$$

Hence the differences of consecutive iterates satisfy the same iteration as the error. Notice that an example of an iterative method written in *difference form* is the method proposed by Jacobi in 1845 and shown in Figures 1.6 and 1.7. Furthermore, from

$$\begin{aligned} \boldsymbol{d}_k &= M^{-1}N(\boldsymbol{u}_k - \boldsymbol{u} + \boldsymbol{u} - \boldsymbol{u}_{k-1}) \\ &= M^{-1}N(-\boldsymbol{e}_k + \boldsymbol{e}_{k-1}) = -M^{-1}N\boldsymbol{e}_k + \boldsymbol{e}_k \\ &= (I - M^{-1}N)\boldsymbol{e}_k, \end{aligned}$$

we obtain a relation between the difference of consecutive iterates and the error:

$$\boldsymbol{d}_k = M^{-1}A\boldsymbol{e}_k.$$

Finally, from the stationary iteration in correction form (2.3) we have $M\boldsymbol{d}_k = \boldsymbol{r}_k$, which relates the difference of consecutive iterates to the residual. We have thus proved the following

Theorem 1 (Error, residual, and difference recurrences)**.** *For a non-singular matrix $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{f} \in \mathbb{R}^n$, let $\boldsymbol{u} \in \mathbb{R}^n$ be the solution of $A\boldsymbol{u} = \boldsymbol{f}$, and let $A = M - N$ be a splitting with M non-singular. Choose $\boldsymbol{u}_0 \in \mathbb{R}^n$ and consider the sequence of iterates $\boldsymbol{u}_{k+1} = M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f}$. Let $\boldsymbol{e}_k := \boldsymbol{u} - \boldsymbol{u}_k$ be the error and $\boldsymbol{r}_k := \boldsymbol{f} - A\boldsymbol{u}_k$ be the residual at step k, and let $\boldsymbol{d}_k := \boldsymbol{u}_{k+1} - \boldsymbol{u}_k$ be the difference of two consecutive iterates. Then these vectors can be computed by the recurrences*

$$\boldsymbol{e}_{k+1} = M^{-1}N\boldsymbol{e}_k, \tag{2.6}$$

$$\boldsymbol{d}_{k+1} = M^{-1}N\boldsymbol{d}_k, \tag{2.7}$$

$$\boldsymbol{r}_{k+1} = (I - AM^{-1})\boldsymbol{r}_k = NM^{-1}\boldsymbol{r}_k, \tag{2.8}$$

where $(I - AM^{-1})$ and $M^{-1}N$ are similar matrices. Moreover, we have the relation

$$M\boldsymbol{d}_k = \boldsymbol{r}_k = A\boldsymbol{e}_k. \tag{2.9}$$

The last relation (2.9) in Theorem 1 has a nice interpretation: the solution of the linear system $A\boldsymbol{u} = \boldsymbol{f}$ can be obtained by solving the correction equation $A\boldsymbol{e}_k = \boldsymbol{r}_k$ and then adding the correction $\boldsymbol{u} = \boldsymbol{u}_k + \boldsymbol{e}_k$. However, we cannot

afford to solve systems with the matrix A because the matrix A is in general too large to be stored and factored. Therefore we replace the original problem by an easier one and solve instead $M\boldsymbol{d}_k = \boldsymbol{r}_k$, and then we iterate $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{d}_k$.

It is clear from (2.6)-(2.7)-(2.8) that the behavior of a stationary method is related to the matrix $M^{-1}N$. For this reason we have the following definition:

Definition 1 (Iteration matrix)**.** *For any matrix splitting $A = M - N$ with M invertible, the matrix $G := M^{-1}N$ is called the* iteration matrix*.*

In the next section we show that the convergence behavior of a stationary iteration based on the splitting $A = M - N$ is governed by the spectrum of the iteration matrix G.

2.2 Convergence analysis

> *The convergence of these methods is rarely guaranteed for all matrices, but a large body of theory exists for the case where the coefficient matrix arises from the finite difference discretization of elliptic partial differential equations.*
>
> *Yousef Saad*, Iterative Methods for Sparse Linear Systems, 2000.

We now study under which conditions a stationary iteration of the form (2.2) or (2.3) converges. We assume that the linear system $A\boldsymbol{u} = \boldsymbol{f}$ has a unique solution and consider the matrix splitting $A = M - N$ with an invertible matrix M. A first convergence result is obtained by using any vector norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}^+$ and the corresponding *induced matrix norm* $\|A\| := \sup_{\|\boldsymbol{u}\|=1} \|A\boldsymbol{u}\|$: taking norms on both sides of the iteration for the error (2.4), we obtain

$$\|\boldsymbol{e}_{k+1}\| \leq \|M^{-1}N\| \, \|\boldsymbol{e}_k\| \leq \ldots \leq \|M^{-1}N\|^{k+1} \|\boldsymbol{e}_0\|.$$

We therefore have convergence if $\|M^{-1}N\| < 1$ for the chosen norm. This is however a sufficient, but not a necessary condition for convergence: as a counterexample, consider a triangular matrix R with zero diagonal. This matrix can have a norm $\|R\| > 1$, but since R is nilpotent, we have $R^k \to 0$ for $k \to \infty$. Therefore we need another quantity that describes convergence more accurately.

Definition 2 (Spectral radius)**.** *The* spectral radius *of a matrix $A \in \mathbb{R}^{n\times n}$ is*

$$\rho(A) := \max_{j=1,\ldots,n} |\lambda_j(A)|,$$

where $\lambda_j(A)$ denotes the j-th eigenvalue of A.

One can wonder whether there is a relation between the spectral radius and the norm of a matrix. We have the following results:

Lemma 1 (Norm and spectral radius)**.** *For all induced matrix norms, $\rho(A) \leq \|A\|$ holds.*

Proof. Let $\lambda, \boldsymbol{v}$ be an eigenpair of A. From the eigenvector-eigenvalue relation $A\boldsymbol{v} = \lambda\boldsymbol{v}$, we obtain that

$$|\lambda|\|\boldsymbol{v}\| = \|\lambda\boldsymbol{v}\| = \|A\boldsymbol{v}\| \leq \|A\|\,\|\boldsymbol{v}\|.$$

Now since $\|\boldsymbol{v}\| \neq 0$, we obtain $|\lambda| \leq \|A\|$, and since this holds for any eigenvalue λ, we also get $\rho(A) = \max_{j=1\ldots n} |\lambda_j(A)| \leq \|A\|$. □

For symmetric matrices, the two concepts of norm and spectral radius are definitely related:

Lemma 2 (Norm and spectral radius for symmetric matrices)**.** *For symmetric matrices $A \in \mathbb{R}^{n\times n}$, the spectral radius equals the 2-norm, $\rho(A) = \|A\|_2$.*

Proof. Using the definition of the 2-norm, we obtain

$$\|A\|_2^2 = \lambda_{\max}(A^\top A) = \lambda_{\max}(A^2) = \max|\lambda(A)|^2 = \rho(A)^2.$$

□

However the spectral radius is not a norm in general. For norms, $\|A\| = 0$ implies that $A = 0$, which does not hold for the spectral radius: for an upper triangular matrix R with zero diagonal (thus all eigenvalues are zero) we have $\rho(R) = 0$ but $R \neq 0$. Furthermore, the triangle inequality

$$\rho(A + B) \leq \rho(A) + \rho(B)$$

also does not hold in general: take for example

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

Then $\rho(A + B) = 1$, but $\rho(A) = \rho(B) = 0$. Nonetheless, the following result holds:

Theorem 2 (Norm and spectral radius)**.** *Let $A \in \mathbb{R}^{n\times n}$. Then for any given $\epsilon > 0$, there exists a norm $\|\cdot\|$, which depends on A and ϵ, such that*

$$\rho(A) \leq \|A\| \leq \rho(A) + \varepsilon.$$

Proof. See Problem 6. □

We are now ready to prove convergence of stationary iterative methods based on the matrix splitting $A = M - N$.

Theorem 3 (Convergence of stationary methods)**.** *Let $A \in \mathbb{R}^{n\times n}$ be non-singular, $A = M - N$ with M non-singular and $\boldsymbol{f} \in \mathbb{R}^n$. The stationary iterative method*

$$M\boldsymbol{u}_{k+1} = N\boldsymbol{u}_k + \boldsymbol{f}$$

converges for any initial vector $\boldsymbol{u}_0 \in \mathbb{R}^n$ to the solution $\boldsymbol{u}$ of the linear system $A\boldsymbol{u} = \boldsymbol{f}$ if and only if $\rho(M^{-1}N) < 1$.

Proof. We start by showing the "only if" part with a proof by contraposition: assume that $|\lambda_m| = \rho(M^{-1}N) \geq 1$. Choosing $\boldsymbol{u}_0$ such that $\boldsymbol{e}_0 = \boldsymbol{u} - \boldsymbol{u}_0$ is a corresponding eigenvector, and applying the error recurrence (2.4), we get

$$\boldsymbol{e}_{k+1} = M^{-1}N\boldsymbol{e}_k = \cdots = (M^{-1}N)^{k+1}\boldsymbol{e}_0 = \lambda_m^{k+1}\boldsymbol{e}_0.$$

Thus, if $|\lambda_m| > 1$, then $|\lambda_m^{k+1}| \to \infty$, so the error cannot converge to zero. If $|\lambda_m| = 1$, then we also have no convergence since the error does not decrease.

For the "if" part, we assume that $\rho(M^{-1}N) < 1$. We then consider the *Jordan decomposition* (see, e.g., [62])

$$M^{-1}N = VJV^{-1}, \quad \text{with } V, J \in \mathbb{C}^{n\times n} \text{ and } V \text{ nonsingular.}$$

The matrix J is block-diagonal

$$J = \begin{bmatrix} J_{m_1}(\lambda_1) & 0 & 0 & \cdots & 0 \\ 0 & J_{m_2}(\lambda_2) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & J_{m_{s-1}}(\lambda_{s-1}) & 0 \\ 0 & \cdots & \cdots & 0 & J_{m_s}(\lambda_s) \end{bmatrix}$$

with

$$J_{m_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_i & 1 \\ 0 & 0 & \cdots & 0 & \lambda_i \end{bmatrix} \in \mathbb{C}^{m_i\times m_i}, \quad i = 1, \ldots, s,$$

where s is the number of Jordan blocks, λ_i are the eigenvalues of $M^{-1}N$, and m_i is the dimension of the i-th block. Now notice that the matrix $J_{m_i}(\lambda_i)$ can be written as the sum of a diagonal matrix and a nilpotent matrix,

$$J_{m_i}(\lambda_i) = \big(\lambda_i I + \widetilde{J}\big),$$

where I is the $m_i \times m_i$ identity and

$$\widetilde{J} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Now, since $\lambda_i I$ and $\widetilde{J}$ commute, we can use the *binomial theorem for matrices*[1] [125, Section 21, Theorem 21.1] to deduce that

$$\left(J_{m_i}(\lambda_i)\right)^k = \left(\lambda_i I + \widetilde{J}\right)^k = \sum_{r=0}^{k} \binom{k}{r} \lambda_i^{k-r} \widetilde{J}^r = \sum_{r=0}^{\min(k,m_i-1)} \binom{k}{r} \lambda_i^{k-r} \widetilde{J}^r, \quad (2.10)$$

where we used that

$$\widetilde{J}^2 = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}, \quad \ldots, \quad \widetilde{J}^{m_i-1} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

Notice that $(M^{-1}N)^k = VJ^kV^{-1}$, and since J is block diagonal, we get

$$J^k = \begin{bmatrix} (J_{m_1}(\lambda_1))^k & 0 & 0 & \cdots & 0 \\ 0 & (J_{m_2}(\lambda_2))^k & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & (J_{m_{s-1}}(\lambda_{s-1}))^k & 0 \\ 0 & \cdots & \cdots & 0 & (J_{m_s}(\lambda_s))^k \end{bmatrix}.$$

Using (2.10) we obtain the well-known expression for the powers of a *Jordan block*

$$J_{m_i}^k(\lambda_i) = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \cdots & \binom{k}{m_i-1}\lambda_i^{k-m_i+1} \\ 0 & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \cdots & \binom{k}{m_i-2}\lambda_i^{k-m_i+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} \\ 0 & 0 & \cdots & 0 & \lambda_i^k \end{bmatrix}.$$

Therefore, if $\rho(M^{-1}N) < 1$, then $|\lambda_i| < 1$ for all i, so that (recall Problem 8)

$$\lim_{k\to\infty} J_{m_i}^k(\lambda_i) = 0$$

[1] The binomial theorem for matrices states: Let A and B be two commuting matrices, then $(A+B)^k = \sum_{r=0}^{k} \binom{k}{r} A^r B^{k-r}$.

for all Jordan blocks. It follows that $\lim_{k\to\infty} J^k = 0$. This implies that

$$\lim_{k\to\infty} (M^{-1}N)^k = \lim_{k\to\infty} VJ^kV^{-1} = V(\lim_{k\to\infty} J^k)V^{-1} = 0.$$

□

2.3 Convergence factor and convergence rate

> *The convergence rate is the (natural) logarithm of the inverse of the convergence factor.*
>
> *Yousef Saad*, Iterative Methods for Sparse Linear Systems, 2000.

If the second author had carefully read Saad's book, in particular the quote above, he would not have created such a confusion in his early publications, until a careful reviewer finally straightened things out between the two notions of convergence factor and convergence rate. In order to understand the convergence behavior, we take the iteration for the error (2.4), and obtain by induction

$$\boldsymbol{e}_k = (M^{-1}N)^k \boldsymbol{e}_0.$$

Taking norms on both sides gives a bound for the error reduction over k iteration steps,

$$\frac{\|\boldsymbol{e}_k\|}{\|\boldsymbol{e}_0\|} \leq \|(M^{-1}N)^k\|.$$

If we want to know how many iterations are needed for the error measured in this norm to be reduced to a given tolerance ε, it suffices to impose

$$\frac{\|\boldsymbol{e}_k\|}{\|\boldsymbol{e}_0\|} \leq \|(M^{-1}N)^k\| < \varepsilon.$$

Rewriting the right-hand side in the form

$$\|(M^{-1}N)^k\| = \left(\left\|(M^{-1}N)^k\right\|^{\frac{1}{k}}\right)^k < \varepsilon$$

and taking the logarithm, we get

$$k > \frac{\ln(\varepsilon)}{\ln\left(\left\|(M^{-1}N)^k\right\|^{\frac{1}{k}}\right)}. \tag{2.11}$$

Since the number of necessary iterations k also appears on the right-hand side, equation (2.11) does not seem very useful at first glance to determine the number of iterations to take. However, for large k we can get a good estimate using the following lemma:

Lemma 3 (Gelfand's formula)**.** *For any matrix $G \in \mathbb{R}^{n\times n}$ with spectral radius $\rho(G)$ and any induced matrix norm $\|\cdot\|$, we have*

$$\lim_{k\to\infty} \|G^k\|^{\frac{1}{k}} = \rho(G).$$

Proof. See Problem 7. □

Definition 3 (Convergence factor)**.** *The* mean convergence factor *of an iteration matrix G is the number*

$$\rho_k(G) = \|G^k\|^{\frac{1}{k}}. \tag{2.12}$$

The asymptotic convergence factor *is the spectral radius*

$$\rho(G) = \lim_{k\to\infty} \rho_k(G). \tag{2.13}$$

Definition 4 (Convergence rate)**.** *The* mean convergence rate *of an iteration matrix G is the number*

$$R_k(G) = -\ln\left(\|G^k\|^{\frac{1}{k}}\right) = -\ln(\rho_k(G)). \tag{2.14}$$

The asymptotic convergence rate *is*

$$R_\infty(G) = -\ln(\rho(G)). \tag{2.15}$$

We can now say how many iteration steps k are necessary until the error reduction reaches a given tolerance ε, or until the error decreases by a factor δ, with $\varepsilon = 1/\delta$, namely

$$k \approx -\frac{\ln \varepsilon}{R_\infty(M^{-1}N)} = \frac{\ln \delta}{R_\infty(M^{-1}N)}.$$

Example 1. *Consider the matrix*

$$A = \begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

and the splitting $A = M - N$, where

$$M = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \quad \textit{and} \quad N = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The iteration matrix is given by $G = M^{-1}N$. The spectral radius of G (computed by the next MATLAB *script) is $\rho(M^{-1}N) \approx 0.35$. Consider a vector $\boldsymbol{e}_0 = [1\,1\,1]^\top$. Then, we can estimate the number of iterations needed to obtain a reduction of the error of a factor of $\delta = 1000$ ($\varepsilon = 1/\delta$):*

$$K = \frac{\ln \varepsilon}{-\ln \rho(M^{-1}N)} \approx 6.64.$$

Thus, we need about 7 iterations. The following MATLAB *script performs 7 iterations of the type $\boldsymbol{e}_{k+1} = G\boldsymbol{e}_k$.*

```
A=[4 -1 0; -1 4 -1; 0 -1 4];
M=diag(diag(A));                        % decomposition A=M-N
N=M-A;
G=M\N;                                  % iteration matrix
rho=max(abs(eig(G)))                    % spectral radius of G
R_inf=-log(rho)                         % asymptoric convergence rate
delta=1000;                             % reduction factor
epsi=1/delta;                           % tolerance
K=-log(epsi)/R_inf                      % estimate of the iterations "k"
e=ones(3,1);                            % initial error
nor(1)=norm(e,2);
for j=1:round(K)                        % approximate K by the closest integer
  e=G*e;                                % iteration step
  nor(j+1)=norm(e,2)/nor(1);
  disp(['j=' num2str(j) ' | norm=' num2str(nor(j+1))]);
end
v=0:round(K);
semilogy(v,nor,'ob-',v,10*rho.^v,'b');
legend('error','asymptotic convergence');
xlabel('iteration');
ylabel('error');
set(gca,'fontsize',16);                  % bigger font size
```

The results we obtain are

```
rho = 0.35355
R_inf = 1.0397
K = 6.6439
j=1 | norm=0.35355
j=2 | norm=0.125
j=3 | norm=0.044194
j=4 | norm=0.015625
j=5 | norm=0.0055243
j=6 | norm=0.0019531
j=7 | norm=0.00069053
```

and we can see that after 7 *iterations the reduction of the error in the norm* $\|\cdot\|_\infty$ *is about* 10^{-3} *as expected. Moreover, the convergence behavior is shown in Figure 2.1 where the error* $\frac{\|e_k\|_2}{\|e_0\|_2}$ *is compared with the asymptotic convergence rate.*

Notice that the convergence rate $R_\infty(G)$ represents the slope that the convergence curve of the norm of the error sequence should have asymptotically (for k sufficiently large). One may wonder whether the convergence of the error sequence is monotone for all k and whether it becomes asymptotically a straight line (as show in Figure 2.1). The answer is negative in both cases, and we show in the following example that one can construct a matrix G that gives rise to a

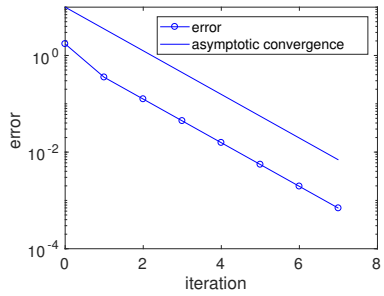


Figure 2.1: Error $\frac{\|\boldsymbol{e}_k\|_2}{\|\boldsymbol{e}_0\|_2}$ and asymptotic convergence rate.

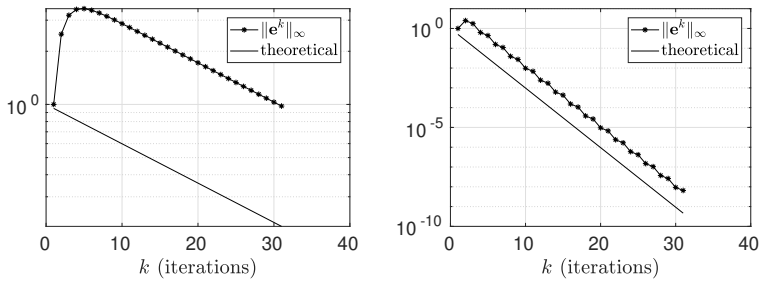


Figure 2.2: Convergence behavior discussed in Example 2.

convergence curve that is not monotonically decreasing for k "small" and that is not a straight line even for "large" values of k.

Example 2. *Consider the iteration matrix*

$$G = \begin{bmatrix} a & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & \frac{1}{2} \end{bmatrix}$$

where $a, b \in (0,1)$. *It holds that* $\rho(G) = \max(a, \frac{1}{2})$ *and* $R_\infty(G) = -\ln(\max(a, \frac{1}{2}))$. *If we set* $a = \frac{1.9}{2}$, *an initial vector* $\boldsymbol{e}^0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top$ *and evaluate the infinity norm of the sequence* $\boldsymbol{e}_{k+1} := G\boldsymbol{e}_k$ *for* $k = 0, 1, 2 \ldots$, *we obtain that* $\|\boldsymbol{e}_k\|_\infty \geq \|\boldsymbol{e}_{k-1}\|_\infty$ *for* $k = 1, 2, 3, 4$. *Therefore, the error in the infinity norm increases in the first iterations and then starts to decay. This behavior is clearly shown in Figure 2.2 (left), where we observe a significant growth of the quantity* $\|\boldsymbol{e}_k\|_\infty$ *in the first iteration. For* k *large the error decays with the expected asymptotic rate* $R_\infty(G)$.

Next, we consider $a = -\frac{1}{2}$. *This means that the iteration matrix* G *has two eigenvalues with the same modulus and opposite signs. In this case we observe the behavior shown in Figure 2.2 (right), where the convergence curve decays monotonically with an averaged slope equal to* $R_\infty(G)$. *However, the*

convergence curve is not a straight line. This is due to the two eigenvalues of G with opposite sign, that are $\frac{1}{2}$ and $-\frac{1}{2}$.

2.4 Regular splittings and M-matrices

> *Interestingly enough, the basis for the analysis of such modern cyclic iterative methods can be traced back to fundamental research by Perron and Frobenius on non-negative matrices, and our first aim is more nearly to survey the basic results on cyclic iterative methods, using the Perron-Frobenius theory as a basis.*
>
> *Richard S. Varga*, Matrix Iterative Analysis, 1962

When computers started to become more and more available, stationary iterations became an attractive alternative to direct solvers for linear systems, and it was important to find general criteria for matrices and associated splittings that lead to convergent stationary iterations. We follow in this subsection the pioneering work of Varga [124], which led to many interesting results on properties of matrices in general, and sparked the more important later research field of how to design rapidly converging stationary iterations and preconditioners.

Definition 5 (Non-negative matrix)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is said to be* non-negative *(*non-positive*) if $a_{ij} \geq 0$ (respectively $a_{ij} \leq 0$) for $i, j = 1, \ldots, n$. To denote a non-negative (non-positive) matrix we use the symbol $A \geq 0$ (respectively $A \leq 0$).*[2]

Perron and Frobenius discovered an interesting property of non-negative matrices:

Theorem 4 (Perron-Frobenius (1907/1912))**.** *Let $A \in \mathbb{R}^{n\times n}$ be a non-negative matrix. Then A has a non-negative real eigenvalue λ which equals the spectral radius of A, $\lambda = \rho(A)$, and a corresponding eigenvector which is non-negative.*

Proof. See, e.g., [64, Theorem 10.2.4] and [124, Theorem 2.7]. □

The original result of Perron and Frobenius is slightly stronger: it requires also that the matrix A be *irreducible* (i.e. there exists no permutation matrix P such that $P^\top AP$ is block upper triangular), and then 'non-negative' can be replaced by 'positive' in Theorem 4.

We now introduce a particular class of splittings for matrices, for which one can obtain very general convergence results.

Definition 6 (Regular splitting)**.** *A splitting $A = M - N$ is said to be* regular *if M is invertible and if both M^{-1} and N are non-negative.*

Theorem 5 (Convergence of regular splitting stationary methods)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a given matrix, and $A = M - N$ be a regular splitting. Then*

$$\rho(M^{-1}N) < 1 \quad \Longleftrightarrow \quad A \text{ is invertible and } A^{-1} \text{ non-negative.}$$

[2] The notation $A \geq 0$ is also used in functional analysis to denote a positive-semidefinite operator; see, e.g., [81, Section 9.3]. However, we do not adopt such notation in the present manuscript.

Proof. If $\rho(M^{-1}N) < 1$, then 1 is not an eigenvalue of $M^{-1}N$. Hence, the determinant of $(I - M^{-1}N)$ is non-zero, which means that $(I - M^{-1}N)$ is invertible. Therefore, we recall that M is invertible and write $A = M - N = M(I - M^{-1}N)$ to deduce that A is invertible as well. Furthermore, using the Neumann series (see Problem 9), we obtain

$$A^{-1} = (I - M^{-1}N)^{-1}M^{-1} = \sum_{j=0}^{\infty}(M^{-1}N)^j M^{-1},$$

which shows that A^{-1} is non-negative since products and sums of non-negative matrices are non-negative matrices as well.

On the other hand, suppose A is invertible and A^{-1} is non-negative. Since M is invertible, we obtain from $A = M-N = M(I-M^{-1}N)$ that also $(I-M^{-1}N)$ is invertible. Thus

$$A^{-1}N = (M(I - M^{-1}N))^{-1}N = (I - M^{-1}N)^{-1}M^{-1}N. \tag{2.16}$$

By assumption, both matrices M^{-1} and N are non-negative, and thus their product is also non-negative. Using Theorem 4, there exists a non-negative vector $\boldsymbol{v}$ such that

$$M^{-1}N\boldsymbol{v} = \lambda\boldsymbol{v}, \quad \lambda = \rho(M^{-1}N).$$

Using (2.16), we obtain

$$A^{-1}N\boldsymbol{v} = (I - M^{-1}N)^{-1}M^{-1}N\boldsymbol{v} = \rho(M^{-1}N)(I - M^{-1}N)^{-1}\boldsymbol{v}. \tag{2.17}$$

Now, since $(I-M^{-1}N)$ is invertible, 1 is not an eigenvalue of $M^{-1}N$. Moreover, since the eigenvalue λ that corresponds to $\rho(M^{-1}N)$ is real and non-negative, we have that $\rho(M^{-1}N) \neq 1$. Therefore, we can write that

$$(I - M^{-1}N)\boldsymbol{v} = (1 - \rho(M^{-1}N))\boldsymbol{v} \;\Leftrightarrow\; (I - M^{-1}N)^{-1}\boldsymbol{v} = \frac{1}{1 - \rho(M^{-1}N)}\boldsymbol{v}.$$

Replacing this equation into (2.17), we obtain

$$A^{-1}N\boldsymbol{v} = \frac{\rho(M^{-1}N)}{1 - \rho(M^{-1}N)}\boldsymbol{v}.$$

Now since A^{-1}, N, and $\boldsymbol{v}$ are non-negative, we get

$$\frac{\rho(M^{-1}N)}{1 - \rho(M^{-1}N)} \geq 0 \quad \Longrightarrow \quad 1 - \rho(M^{-1}N) \geq 0 \quad \Longrightarrow \quad 0 \leq \rho(M^{-1}N) \leq 1.$$

We have already seen that the non-singularity of $I - M^{-1}N$ and Theorem 4 ensures that $\rho(M^{-1}N) \neq 1$. Hence, we must have $\rho(M^{-1}N) < 1$. □

An example of a regular splitting is given by the matrices A, M, and N in Example 1. However, a regular splitting of an arbitrary matrix does not necessary exists, as we show in the next example.

Example 3. *Take any matrix $A \in \mathbb{R}^{2\times 2}$ such that $a_{j,k} > 0$ for all $j,k = 1,2$. We write that $A > 0$, where this inequality is understood in a component-wise manner. Assume that a regular splitting $A = M - N$ exists. Since $A > 0$ and $N \geq 0$, we get $M = A + N \geq A > 0$. Therefore, all the entries of M must be positive, that is $m_{j,k} > 0$ for all $j,k = 1,2$. Since the splitting is regular, the matrix M must be invertible, which means that its determinant is non-zero, that is $m_{1,1}m_{2,2} - m_{2,1}m_{1,2} \neq 0$, and we get $M^{-1} = \frac{1}{m_{1,1}m_{2,2}-m_{2,1}m_{2,2}} \begin{bmatrix} m_{2,2} & -m_{1,2} \\ -m_{2,1} & m_{1,1} \end{bmatrix}$. However, this matrix is not non-negative, because all of its entries are non-zero and some of them must be necessarily negative. This contradicts our hypothesis.*

Definition 7 (M-matrix)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is an* M-matrix *, if*

1. *$a_{ii} > 0$ for $i = 1,\ldots,n$,*
2. *$a_{ij} \leq 0$ for $i \neq j$, $\quad i,j = 1,\ldots,n$,*
3. *A is invertible,*
4. *$A^{-1} \geq 0$.*

Example 4. *Consider the boundary value problem*

$$-u''(x) = f(x) \text{ for } x \in (0,1), \quad u(0) = u(1) = 0.$$

After discretizing with step-size $h = 1/(n+1)$, $x_j = jh$, $u_j \approx u(x_j)$ and approximating

$$u''(x_j) \approx \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2},$$

we obtain the linear system

$$\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

Since the matrix of the system A has the non-negative inverse $B = A^{-1}$ where B is computed explicitly with the MATLAB *statements (see Problem 23)*

```
L=tril(ones(n)); v=ones(n,1);
B=L'*(eye(n)-1/(n+1)*(v*v'))*L;
```

we conclude that A is an M-matrix. This can also be proved analytically, see Problem 10.

As a consequence of Theorem 5 we obtain the following general convergence result.

Corollary 1. *Let $A \in \mathbb{R}^{n\times n}$ be an M-matrix and $A = M - N$ be a regular splitting. Then the stationary iteration $M\boldsymbol{u}_{k+1} = N\boldsymbol{u}_k + \boldsymbol{f}$ converges to the solution of $A\boldsymbol{u} = \boldsymbol{f}$.*

A further result, which is attributed to Householder [75] and John [78] is the following theorem (see also [7]).

Theorem 6 (Householder-John (1955/1956))**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and non-singular. Let $A = M - N$ be a splitting with a real, non-singular matrix M and assume that $N + M^\top$ is symmetric positive definite. Then*

$$\rho(M^{-1}N) < 1 \iff A \text{ is positive definite.}$$

Proof. We prove first that if A is positive definite, then $\rho(M^{-1}N) < 1$. Let $(\lambda, \boldsymbol{v})$ be a possibly complex eigenpair of the matrix $M^{-1}N$, $M^{-1}N\boldsymbol{v} = \lambda\boldsymbol{v}$. Then it holds that

$$A = M(I - M^{-1}N) \quad \Longrightarrow \quad A\boldsymbol{v} = (1-\lambda)M\boldsymbol{v},$$

and multiplying from the left with $\boldsymbol{v}^*$ we obtain

$$\boldsymbol{v}^*A\boldsymbol{v} = (1-\lambda)\boldsymbol{v}^*M\boldsymbol{v}, \tag{2.18}$$

which shows that $\lambda \neq 1$, since A is positive definite. If we take the conjugate transpose of the last equation, we get

$$(1-\bar{\lambda})\boldsymbol{v}^*M^*\boldsymbol{v} = (\boldsymbol{v}^*A\boldsymbol{v})^* = \boldsymbol{v}^*A^*\boldsymbol{v} = \boldsymbol{v}^*A\boldsymbol{v}. \tag{2.19}$$

Since M is real we have $M^* = M^\top$. Let $Q = N + M^\top = M + M^\top - A$, which is by assumption positive definite. Dividing (2.18) and (2.19) respectively by $(1-\lambda)$ and $(1-\bar{\lambda})$ (which cannot vanish, since we have seen $\lambda \neq 1$) and adding them, we get

$$\underbrace{\left(\frac{1}{1-\lambda} + \frac{1}{1-\bar{\lambda}}\right)}_{2\,\mathrm{Re}\,\frac{1}{1-\lambda}} \boldsymbol{v}^*A\boldsymbol{v} = \boldsymbol{v}^*(\underbrace{M + M^\top}_{Q+A})\boldsymbol{v}.$$

This implies that

$$2\,\mathrm{Re}\,\frac{1}{1-\lambda} = \frac{\boldsymbol{v}^*Q\boldsymbol{v} + \boldsymbol{v}^*A\boldsymbol{v}}{\boldsymbol{v}^*A\boldsymbol{v}} = 1 + \frac{\boldsymbol{v}^*Q\boldsymbol{v}}{\boldsymbol{v}^*A\boldsymbol{v}} > 1,$$

since both Q and A are positive definite. We therefore have

$$2\,\mathrm{Re}\,\frac{1}{1-\lambda} > 1,$$

which, using $\lambda = \alpha + i\beta$ with $\alpha, \beta \in \mathbb{R}$, gives

$$2\,\mathrm{Re}\,\frac{1}{1-\lambda} = \frac{2(1-\alpha)}{(1-\alpha)^2 + \beta^2} > 1 \iff |\lambda|^2 = \alpha^2 + \beta^2 < 1 \implies \rho(M^{-1}N) < 1.$$

We have then obtained that if A is positive definite, then $\rho(M^{-1}N) < 1$.

To prove the other direction, namely that if $\rho(M^{-1}N) < 1$ then A is positive definite, we first need the following lemma:

Lemma 4. *Under the same conditions as in Theorem 6, the following identity holds*

$$A - (M^{-1}N)^\top A(M^{-1}N) = (I - M^{-1}N)^\top (M^\top + N)(I - M^{-1}N). \quad (2.20)$$

Proof. To prove this lemma, we replace $A = M - N$, expand the left- and the right-hand sides of (2.20), and show that they are equal. The expansion of the right-hand side gives

$$\begin{aligned} & (I - M^{-1}N)^\top (M^\top + N)(I - M^{-1}N) \\ = \; & (M^\top + N - N^\top - N^\top M^{-\top} N)(I - M^{-1}N). \end{aligned}$$

Because of the symmetry $A^\top = A \iff M^\top - N^\top = M - N$ we get

$$\begin{aligned} & (I - M^{-1}N)^\top (M^\top + N)(I - M^{-1}N) \\ = \; & (M - N^\top M^{-\top} N)(I - M^{-1}N) \\ = \; & M - N^\top M^{-\top} N - N + N^\top M^{-\top} N M^{-1} N. \end{aligned}$$

Expanding the left-hand side, we obtain

$$\begin{aligned} & A - (M^{-1}N)^\top A(M^{-1}N) \\ = \; & M - N - N^\top M^{-\top}(M - N)M^{-1}N \\ = \; & M - N - (N^\top M^{-\top} M - N^\top M^{-\top} N)M^{-1}N \\ = \; & M - N - (N^\top M^{-\top} N - N^\top M^{-\top} N M^{-1} N) \\ = \; & M - N^\top M^{-\top} N - N + N^\top M^{-\top} N M^{-1} N, \end{aligned}$$

the same expression as for the right-hand side, which concludes this proof. □

Continuing with the proof of Theorem 6, we now prove the second part by contraposition. We suppose that A is not positive definite and show that this implies $\rho(M^{-1}N) \geq 1$. Let $\boldsymbol{e}_0 \in \mathbb{R}^n$ be given. We consider the sequence of vectors

$$\boldsymbol{e}_{k+1} = M^{-1}N\boldsymbol{e}_k.$$

Applying the lemma, we obtain

$$A\boldsymbol{e}_k - (M^{-1}N)^\top A \underbrace{(M^{-1}N)\boldsymbol{e}_k}_{\boldsymbol{e}_{k+1}} = (I - M^{-1}N)^\top \underbrace{(M^\top + N)}_{Q} \underbrace{(I - M^{-1}N)\boldsymbol{e}_k}_{\boldsymbol{e}_k - \boldsymbol{e}_{k+1}}.$$

Multiplying the last equation from the left with $\boldsymbol{e}_k^\top$ yields

$$\boldsymbol{e}_k^\top A\boldsymbol{e}_k - \boldsymbol{e}_{k+1}^\top A\boldsymbol{e}_{k+1} = (\boldsymbol{e}_k - \boldsymbol{e}_{k+1})^\top Q(\boldsymbol{e}_k - \boldsymbol{e}_{k+1}) \geq 0,$$

since Q is positive definite. Therefore the sequence $\boldsymbol{e}_k^\top A\boldsymbol{e}_k$ satisfies

$$\boldsymbol{e}_k^\top A\boldsymbol{e}_k \geq \boldsymbol{e}_{k+1}^\top A\boldsymbol{e}_{k+1},$$

and is thus non-increasing. Since A is assumed to be non-singular but not positive definite, we can find an initial vector $\boldsymbol{e}_0$ such that

$$0 > \boldsymbol{e}_0^\top A\boldsymbol{e}_0 \geq \boldsymbol{e}_1^\top A\boldsymbol{e}_1 \geq \cdots .$$

This means that $\boldsymbol{e}_k$ is not convergent to 0. Thus $\rho(M^{-1}N) \geq 1$. □

We will now get back to the classical, historical stationary iterations of Jacobi, Gauss and Seidel we have seen in Chapter 1. These methods can be written by splitting the matrix A into the strictly lower-triangular part L, the diagonal part $D = \mathrm{diag}(A)$, and the strictly upper-triangular part U,

$$A = L + D + U.$$

2.5 Jacobi

> *Eine solche [Näherungsmethode] bietet sich von selber dar, wenn in den verschiedenen Gleichungen immer eine andere Variable mit einem vorzugsweise grossen Coefficienten multiplicirt ist.*
>
> Carl. G. J. Jacobi, Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden lineären Gleichungen, 1845.

The *Jacobi method* is based on solving for every variable locally, with the other variables frozen at their old values, as we have seen in Section 1.2. This corresponds to the iteration

$$(\boldsymbol{u}_{k+1})_i = \frac{1}{a_{ii}}\Big[\boldsymbol{f}_i - \sum_{j=1, j\neq i}^{n} a_{ij}(\boldsymbol{u}_k)_j\Big], \tag{2.21}$$

which can be performed in parallel for all i. This iteration can be written as a matrix splitting $A = M - N$, with

$$M = D, \quad N = -L - U \quad \Longrightarrow \quad D\boldsymbol{u}_{k+1} = -(L+U)\boldsymbol{u}_k + \boldsymbol{f}. \tag{2.22}$$

With the following MATLAB program one can test the Jacobi method on our Laplace model problem from Section 1.3:

```
m=15;                              % number of gridpoints
A=Laplacian(m,2);                  % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;     % put bc into the rhs
u=A\f;                             % solve by sparse Gaussian elimination
x=0:1/(m+1):1; y=x;                % mesh point vectors
uj=zeros(size(f));                 % initial guess
U=zeros(m+2); U(end,1:m+2)=1;      % for plotting
for k=0:10                         % do 10 Jacobi steps
  err(k+1)=max(max(abs(u-uj)));    % compute error
  U(2:m+1,2:m+1)=reshape(uj,m,m);
  mesh(x,y,U);                     % plot current approximation
  xlabel('x');ylabel('y');
  uj=uj+(f-A*uj)./diag(A);         % perform one Jacobi step
  pause
end
```

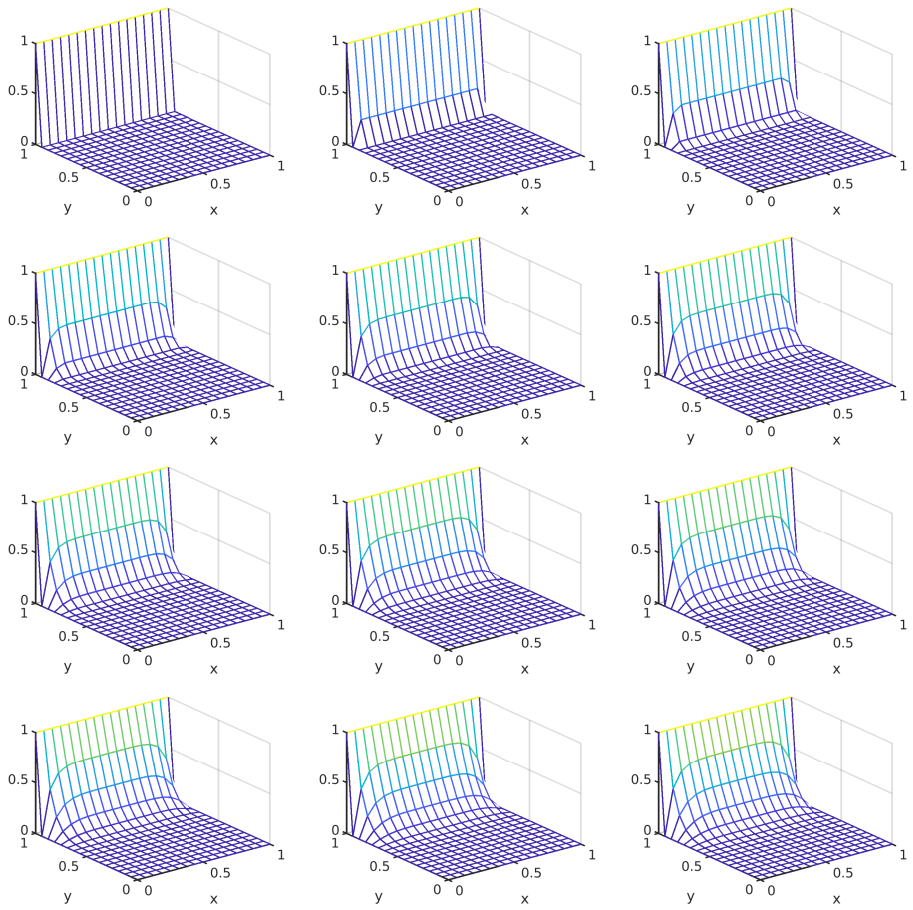


Figure 2.3: Initial guess and first iterations of the Jacobi method (2.22) applied to our Laplace model problem from Section 1.3.

We show in Figure 2.3 the first few approximations computed by the Jacobi method starting with a zero initial guess. Comparing with the shape of the solution shown in Figure 1.11, we see that the method apparently converges, but rather slowly. A classical convergence result for the Jacobi method, which motivated Jacobi to actually consider the method, see the quote above, is given in the following theorem:

Theorem 7 (Convergence of Jacobi's method for diagonally dominant matrices)**.** *If the matrix $A \in \mathbb{R}^{n\times n}$ is strictly diagonally dominant, i.e.*

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}| \quad \textit{for } i = 1, \ldots, n, \tag{2.23}$$

then the Jacobi iteration (2.22) *converges.*

Proof. The iteration matrix of the Jacobi method is $G_{\mathrm{J}} = -D^{-1}(L+U)$. The condition (2.23) allows us to estimate that

$$\|G_{\mathrm{J}}\|_\infty = \max_{i\in\{1,\dots,n\}} \frac{1}{|a_{ii}|}\sum_{j\neq i}|a_{ij}| < 1.$$

Using Lemma 1, we obtain

$$\rho_{G_{\mathrm{J}}} \leq \|G_{\mathrm{J}}\|_\infty < 1,$$

which together with Theorem 3 concludes the proof. □

We now apply Theorem 6 to the method of Jacobi.

Theorem 8 (Convergence of Jacobi's method for symmetric matrices)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and non-singular and consider the splitting $A = D+L+L^\top$ with L strictly lower triangular and D diagonal and positive definite, $D_{ii} > 0$ for $i = 1,\dots,n$. Then the Jacobi iteration converges if and only if A and $2D-A$ are positive definite.*

Proof. If A and $Q := 2D - A$ are positive definite, then according to Theorem 6 (Householder-John) Jacobi converges. In fact, it suffices to set $M = D$ and $N = -(L+L^\top)$, which implies that $N + M^\top = 2D - A$.

For the reverse, we assume now that Jacobi is convergent, and we want to show that A and Q are positive definite. Consider an eigenpair $(\lambda, \boldsymbol{v})$ of the iteration matrix $G_{\mathrm{J}} := -D^{-1}(L^\top + L)$,

$$M^{-1}N\boldsymbol{v} = \underbrace{-D^{-1}(L+L^\top)}_{G_{\mathrm{J}}}\boldsymbol{v} = \lambda\boldsymbol{v}. \tag{2.24}$$

Since D is positive definite, we can take its square-root, $D = D^{\frac{1}{2}}D^{\frac{1}{2}}$. The matrix G_{J} is then similar to the symmetric matrix

$$D^{\frac{1}{2}}G_{\mathrm{J}}D^{-\frac{1}{2}} = -D^{-\frac{1}{2}}(L^\top + L)D^{-\frac{1}{2}},$$

therefore the eigenvalues of the iteration matrix are real, and since we suppose that Jacobi is convergent, we have $|\lambda| < 1$ (see Theorem 3). Furthermore, from the eigenvalue equation (2.24), we have

$$(L+L^\top)\boldsymbol{v} = -\lambda D\boldsymbol{v}, \tag{2.25}$$

and adding on both sides $D\boldsymbol{v}$ gives

$$\begin{aligned} A\boldsymbol{v} &= (1-\lambda)D\boldsymbol{v} \\ \iff \underbrace{D^{-\frac{1}{2}}AD^{-\frac{1}{2}}}_{\widetilde{A}}\underbrace{D^{\frac{1}{2}}\boldsymbol{v}}_{\boldsymbol{y}} &= (1-\lambda)\underbrace{D^{\frac{1}{2}}\boldsymbol{v}}_{\boldsymbol{y}}. \end{aligned}$$

This means that $1-\lambda > 0$ is an eigenvalue of $\widetilde{A}$ with the corresponding eigenvector $\boldsymbol{y}$. Thus $\widetilde{A}$ is positive definite, which means that $0 < \boldsymbol{y}^\top\widetilde{A}\boldsymbol{y} = \boldsymbol{v}^\top A\boldsymbol{v}$ for

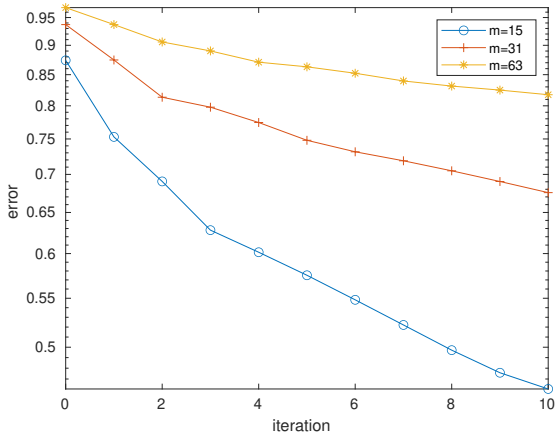


Figure 2.4: Convergence of the Jacobi method for different mesh sizes $h = \frac{1}{m+1}$.

all $\boldsymbol{y} \neq 0$ and therefore also for all $\boldsymbol{v} = D^{-\frac{1}{2}}\boldsymbol{y} \neq 0$. Therefore also A is positive definite.

A similar argument holds for Q. Multiplying (2.25) by -1 and adding $D\boldsymbol{v}$ on both sides yields

$$\underbrace{(D - (L + L^\top))}_{Q}\boldsymbol{v} = (1 + \lambda)D\boldsymbol{v}.$$

As before

$$D^{-\frac{1}{2}}QD^{-\frac{1}{2}}\boldsymbol{y} = (1 + \lambda)\boldsymbol{y},$$

and since $1 + \lambda > 0$ we conclude that Q is also positive definite. □

Using Theorem 8, one can show that Jacobi's method applied to the discrete Laplace problem converges, see Problem 13.

Proving convergence was the dominant task early on when studying iterative methods, but when using such methods to solve practical problems, one is really interested in the convergence speed. In particular, when solving discretized partial differential equations, one would like to solve problems with finer and finer mesh sizes to get more and more accurate solutions, and thus is interested in how convergence depends on the mesh size. We show in Figure 2.4 how the error decreases as the iterations progress when Jacobi is used to solve our Laplace model problem from Section 1.3 for the mesh we used for Figure 2.3 with $m = 15$ interior mesh points, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points. We see that convergence strongly depends on the mesh parameter h, and deteriorates when the mesh is refined, a very undesirable property. In this particular case where A is equal to the negative two-dimensional finite-difference Laplacian, it is possible to characterize precisely the deterioration of the Jacobi method with respect to the grid size h. A

well known fact is that the eigenvalues of A are given by (see Problem 10 and the corresponding hints)

$$\lambda_{j,k}(A) = 4 + 2\cos\Big(\frac{k\pi}{m+1}\Big) + 2\cos\Big(\frac{j\pi}{m+1}\Big),$$

for $j, k = 1, \ldots, m$. Since the Jacobi iteration matrix is $G_{\mathrm{J}} = -D^{-1}(L + L^\top) = I - \frac{1}{4}A$, the corresponding eigenvalues are

$$\lambda_{j,k}(G_{\mathrm{J}}) = 1 - \frac{1}{4}\lambda_{j,k}(A).$$

Hence, the spectral radius of G_{J} (as a function of the grid size h) is

$$\begin{aligned} \rho_{G_{\mathrm{J}}}(h) &= \max_{k,j}\Big|1 - \frac{1}{4}\lambda_{k,j}(A)\Big| \\ &= \frac{1}{2}\max_{k,j}\Big|\cos\Big(\frac{k\pi}{m+1}\Big) + \cos\Big(\frac{j\pi}{m+1}\Big)\Big| = \cos\big(h\pi\big), \end{aligned} \tag{2.26}$$

where we used that $h = \frac{1}{m+1}$. A Taylor expansion of $\rho_{G_{\mathrm{J}}}(h)$ around zero leads to

$$\rho_{G_{\mathrm{J}}}(h) = 1 - \frac{\pi^2}{2}h^2 + O(h^4),$$

which shows precisely the deterioration of $\rho_{G_{\mathrm{J}}}(h)$ as $h \to 0$. This deterioration is related to the fact that Jacobi only treats neighboring mesh points in each iteration, and as Figure 2.3 indicates, for finer and finer meshes, approximations will take longer and longer to reach grid points with small y coordinate.

A similar convergence result as in Theorem 8 can be proved in the case that the decomposition is obtained by block matrices. For example, consider the block-tridiagonal matrix A given in the introduction in (1.9),

$$A = \begin{bmatrix} T & I & & \\ I & T & \ddots & \\ & \ddots & \ddots & I \\ & & I & T \end{bmatrix}.$$

It is natural to use a block splitting $A = D + L + L^\top$, where

$$D = \begin{bmatrix} T & & & \\ & T & & \\ & & \ddots & \\ & & & T \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 0 & & & \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}.$$

We then get a *block-Jacobi Iteration*.

Theorem 9 (Convergence of the block-Jacobi method)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and non-singular and consider the splitting $A = D + L + L^\top$, where*

$$D = \begin{bmatrix} \ddots & & \\ & D_{jj} & \\ & & \ddots \end{bmatrix} \tag{2.27}$$

with the blocks D_{jj} positive definite, and L is strictly lower block-triangular. Then the Jacobi iteration converges if and only if A and $2D - A$ are positive definite.

Proof. The proof is identical to the proof of Theorem 8: we only need to note that since the blocks D_{jj} are positive definite, D is positive definite as well. Hence, we have the decomposition $D = D^{\frac{1}{2}} D^{\frac{1}{2}}$. □

This slight modification of Theorem 8 can be used to show that block Jacobi converges for the matrix A given in (1.9). We know that $-A$ is a positive definite matrix, see Problem 10. According to Theorem 9, it suffices to show that

$$A - 2D = \begin{bmatrix} -T & I & & \\ I & -T & \ddots & \\ & \ddots & \ddots & I \\ & & I & -T \end{bmatrix}$$

is also positive definite; see Problem 16. Finally, we wish to remark that block-Jacobi is very much related to domain decomposition methods. In particular, it is equivalent to a discrete parallel Schwarz method with minimal overlap; see Chapter 4 and Problem 39.

2.6 Gauss-Seidel

> *Wenn man also, von irgend welchem Systeme von Anfangswerthen ausgehend, und in irgend welcher Aufeinanderfolge der Unbekannten (wobei es nicht nöthig ist, den ganzen Cyklus derselben durchzugehen, ehe man wieder auf eine schon verbesserte zurückkommt), successive Correctionen an den Unbekannten anbringt, indem man Sorge trägt, die jedesmalige Verbesserung einer jeden immer so zu bestimmen, dass durch dieselbe diejenige Normalgleichung erfüllt wird, in der die betreffende Unbekannte die ausgezeichnete Stellung in der Diagonale einnimmt, so verringert man Schritt für Schritt die Summe der Fehlerquadrate, solange an ihr noch etwas zu verringern ist.*
>
> *Ludwig Seidel*, Über ein Verfahren, die Gleichungen, auf welche die Methode der kleinsten Quadrate führt, sowie lineäre Gleichungen überhaupt, durch successive Annäherung aufzulösen, 1874.

Today, the *Gauss-Seidel method* is the original Gauss method we have seen in Section 1.2, and more generally described by Seidel, see the quote above, but simply cycling through all the variables sequentially, i.e.

$$(\boldsymbol{u}_{k+1})_i = \frac{1}{a_{ii}}\Big[\boldsymbol{f}_i - \sum_{j=1}^{i-1} a_{ij}(\boldsymbol{u}_{k+1})_j - \sum_{j=i+1}^{n} a_{ij}(\boldsymbol{u}_k)_j\Big]. \tag{2.28}$$

This corresponds to the matrix splitting

$$M = D + L, \quad N = -U \quad \Rightarrow \quad (D + L)\boldsymbol{u}_{k+1} = -U\boldsymbol{u}_k + \boldsymbol{f}. \tag{2.29}$$

With the following MATLAB program one can test the Gauss-Seidel method on our Laplace model problem from Section 1.3:

```
m=15;                              % number of gridpoints
A=Laplacian(m,2);                  % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;     % put bc into the rhs
u=A\f;                             % solve by sparse Gaussian elimination
x=0:1/(m+1):1; y=x;                % mesh point vectors
ugs=zeros(size(f));                % initial guess
U=zeros(m+2); U(end,1:m+2)=1;      % for plotting
LD=tril(A);                        % L+D
for k=0:10                         % do 10 Gauss-Seidel steps
  err(k+1)=max(max(abs(u-ugs)));   % compute error
  U(2:m+1,2:m+1)=reshape(ugs,m,m);
  mesh(x,y,U);                     % plot current approximation
  xlabel('x');ylabel('y');
  ugs=ugs+LD\(f-A*ugs);            % perform one Gauss-Seidel step
  pause
end
```

We show in Figure 2.5 the first few approximations computed by the Gauss-Seidel method starting with a zero initial guess. Comparing with the iterates of the Jacobi method in Figure 2.3, we see that Gauss-Seidel converges a bit faster, see also Figure 2.9 later which shows error curves. This seems to be natural, since Gauss-Seidel uses the updated values as soon as they are available. Faster convergence is often the case, e.g. for our Laplace model problem $\Delta u = 0$, for which one can show that only half of the iteration steps are needed for the same accuracy; see, e.g., Problem 18. However, here is an example where Jacobi converges and Gauss-Seidel does not.

Example 5. *Consider the matrix*

$$A = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 1 & 0 \\ 1 & 2 & -3 \end{bmatrix}.$$

The Jacobi iteration matrix is

$$G_{\mathrm{J}} = -D^{-1}(L + U) = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ \frac{1}{3} & \frac{2}{3} & 0 \end{bmatrix},$$

whose eigenvalues are $0.373\cdots \pm i0.867\ldots$ *and* $-0.747\ldots$ *Thus* $\rho_{G_{\mathrm{J}}} = 0.944\cdots < 1$ *and the iteration converges. On the other hand, the iteration matrix of Gauss-Seidel is*

$$G_{\mathrm{GS}} = -(D + L)^{-1}U = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{bmatrix},$$

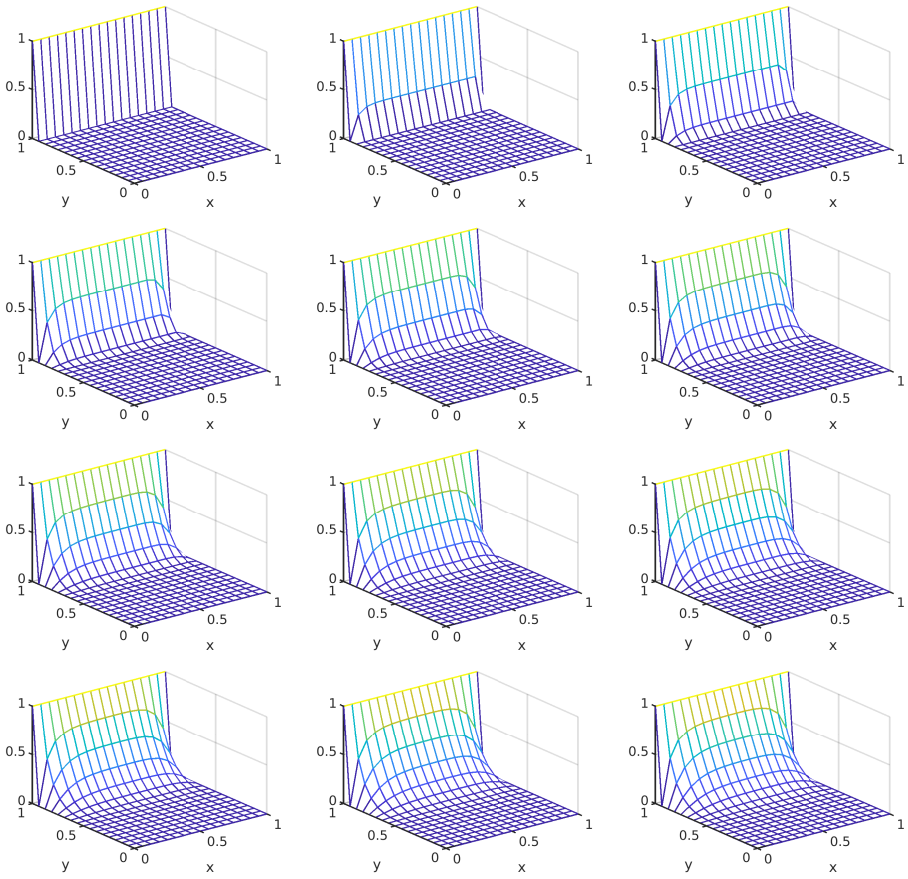


Figure 2.5: Initial guess and first iterations of the Gauss Seidel method (2.29) applied to our Laplace model problem from Section 1.3.

which has the eigenvalues 0, 0, -1 *with* $\rho_{G_{\mathrm{GS}}} = 1$. *The iteration therefore does not converge in general.*

In the next example, we show a case where the Gauss-Seidel method converges, while the Jacobi method can diverge.

Example 6. *Consider the matrix*

$$A = \begin{bmatrix} 1 & t & t \\ t & 1 & t \\ t & t & 1 \end{bmatrix},$$

where t *is a positive arbitrary parameter in* $(0, 1)$. *The Gauss-Seidel and Jacobi*

iteration matrices are

$$G_{\mathrm{GS}} = \begin{bmatrix} 0 & -t & -t \\ 0 & t^2 & t^2 - t \\ 0 & t^2 - t^3 & 2t^2 - t^3 \end{bmatrix} \quad \textit{and} \quad G_{\mathrm{J}} = \begin{bmatrix} 0 & -t & -t \\ -t & 0 & -t \\ -t & -t & 0 \end{bmatrix}.$$

The eigenvalues of these matrices are in absolute value given by

$$|\lambda(G_{\mathrm{GS}})| = \left\{ \frac{|t^3 + \sqrt{t^2 - 4t}(t^2 - t) - 3t^2|}{2}, \frac{|t^3 - \sqrt{t^2 - 4t}(t^2 - t) - 3t^2|}{2}, 0 \right\}$$

and

$$|\lambda(G_{\mathrm{J}})| = \{2|t|, |t|, |t|\}.$$

If we plot $\rho_{G_{\mathrm{GS}}}$ as a function of t, we obtain Figure 2.6, where it is clear that $\rho_{G_{\mathrm{GS}}} < 1$ for any $t \in (0, 1)$. On the other hand, it is obvious that $\rho_{G_{\mathrm{J}}} = 2|t|$.

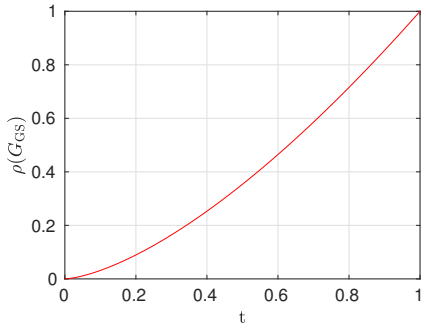


Figure 2.6: $\rho_{G_{\mathrm{GS}}}$ as function of t for $t \in [0, 1]$.

Hence, $\rho_{G_{\mathrm{J}}} > 1$ for all $t > \frac{1}{2}$.

Like Jacobi, Gauss-Seidel also only works on neighboring mesh points when solving discretized partial differential equations, and one can expect that convergence also deteriorates when the mesh is refined. We show in Figure 2.7 how the error decreases as the iterations progress when Gauss-Seidel is used to solve our Laplace model problem from Section 1.3 for the mesh we used for Figure 2.5 with $m = 15$ interior mesh points, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points. We see that like for Jacobi, convergence strongly depends on the mesh parameter h, and deteriorates when the mesh is refined. So even though Gauss-Seidel is a bit faster than Jacobi (note the different scale in Figure 2.7 compared to Figure 2.4), Gauss-Seidel has the same problems as Jacobi to solve discretized partial differential equations: for finer and finer meshes, approximations will take longer and longer to travel across the grid. In particular, similarly as for Jacobi one can prove that

$$\rho_{G_{\mathrm{GS}}}(h) = 1 - \pi^2 h^2 + O(h^4),$$

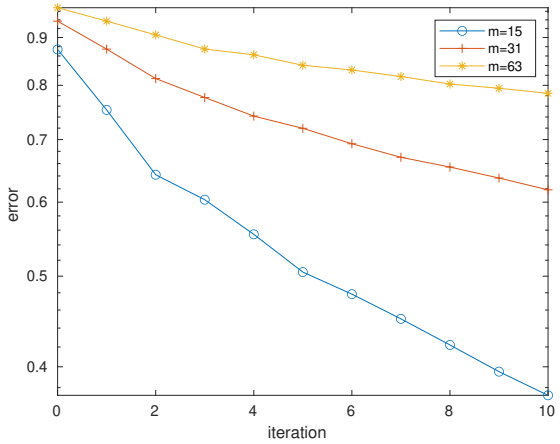


Figure 2.7: Convergence of the Gauss-Seidel method for different mesh sizes $h = \frac{1}{m+1}$.

which clearly shows the deterioration of the convergence factor of Gauss-Seidel for $h \to 0$. To obtain this result, a convergence analysis is required. However, instead of analyzing the convergence of the Gauss-Seidel method, we now show an important generalization, with improved convergence behavior. The convergence analysis of the generalization then also contains Gauss-Seidel as a special case.

2.7 Successive over-relaxation: SOR

> *One can use a fixed relaxation factor, and, if this single factor is suitably chosen, a large gain in the rate of convergence is possible.*
>
> David Young, PhD Thesis, 1950.

The *successive over-relaxation method* (SOR) is derived from the Gauss-Seidel method by introducing a relaxation parameter ω, and was the core subject of the PhD thesis of *David Young* [128], who managed to determine the optimal choice of the parameter for a large class of problems, see the quote above. This was the first attempt to design a rapidly converging stationary iteration, not just to obtain a convergent one, and the ideas of David Young still influence research on stationary iterations and preconditioning today, as we will see in Section 4.6 on domain decomposition methods. In SOR, the component $(\boldsymbol{u}_{k+1})_i$ is computed as for Gauss-Seidel but then averaged with its value from the previous iteration,

$$(\boldsymbol{u}_{k+1})_i = (1-\omega)(\boldsymbol{u}_k)_i + \frac{\omega}{a_{ii}}\Big[\boldsymbol{f}_i - \sum_{j=1}^{i-1} a_{ij}(\boldsymbol{u}_{k+1})_j - \sum_{j=i+1}^{n} a_{ij}(\boldsymbol{u}_k)_j\Big]. \quad (2.30)$$

One can clearly see that $\omega = 1$ leads to the Gauss-Seidel method (2.28). In fact, SOR can also be derived by considering the Gauss-Seidel iteration form

$$(D+L)\boldsymbol{u} = -U\boldsymbol{u} + \boldsymbol{f}. \quad (2.31)$$

Multiplying (2.31) by ω and adding on both sides the expression $(1-\omega)D\boldsymbol{u}$, we obtain the SOR iteration

$$(D+\omega L)\boldsymbol{u}_{k+1} = (-\omega U + (1-\omega)D)\boldsymbol{u}_k + \omega \boldsymbol{f}. \tag{2.32}$$

SOR is therefore based on the splitting

$$A = M - N \text{ with } M = \frac{1}{\omega}D + L \text{ and } N = -U + \left(\frac{1}{\omega} - 1\right) D, \tag{2.33}$$

where we divided (2.32) by ω in order to remove the factor ω in front of the right-hand side term $\boldsymbol{f}$ and obtain the stationary iterative method in standard form (2.2). We can again clearly see that for $\omega = 1$ we get as special case the Gauss-Seidel method.

It is very easy to adapt the Gauss-Seidel MATLAB program from Section 2.6 to perform SOR iterations. Testing the convergence with a suitable choice of the parameter ω that we will see later, we obtain the first iterates shown in Figure 2.8. Comparing with the Gauss-Seidel iterates in Figure 2.5, we see that SOR is substantially faster. How much faster the convergence truly is is illustrated in Figure 2.9 which shows the errors measured: SOR is much faster than Jacobi or Gauss-Seidel; using an optimized relaxation parameter makes all the difference!

To start our investigation of SOR, we now show that one cannot choose the relaxation parameter arbitrarily if one wants to obtain a convergent method. This general, very elegant result is due to William Kahan from his PhD thesis [79].

Theorem 10 (Kahan (1958))**.** *Let $A \in \mathbb{R}^{n\times n}$ and $A = L + D + U$ with D diagonal and invertible, and L and U strictly lower- and upper triangular. If*

$$G_{\text{SOR}} = (D+\omega L)^{-1}(-\omega U + (1-\omega)D) \tag{2.34}$$

is the SOR iteration matrix, then

$$\rho_{G_{\text{SOR}}} \geq |\omega - 1|, \ \forall \omega \in \mathbb{R}. \tag{2.35}$$

Proof. The key idea is to insert DD^{-1} between the factors of G_{SOR},

$$\begin{aligned} G_{\text{SOR}} &= (D+\omega L)^{-1} D D^{-1} (-\omega U + (1-\omega)D) \\ &= (I + \omega D^{-1} L)^{-1} (-\omega D^{-1} U + (1-\omega) I). \end{aligned}$$

Now the determinant of $(I + \omega D^{-1}L)$ equals 1, since this matrix is lower triangular with unit diagonal, which implies that the determinant of its inverse also equals 1. Therefore

$$\det(G_{\text{SOR}}) = \det(-\omega D^{-1}U + (1-\omega)I) = (1-\omega)^n,$$

since this second factor is upper triangular with $1-\omega$ on the diagonal. The determinant of a matrix is equal to the product of its eigenvalues, which in our case yields

$$\prod_{j=1}^{n} \lambda_j(G_{\text{SOR}}) = (1-\omega)^n \quad \Longrightarrow \quad |1-\omega|^n \leq \left(\max_j |\lambda_j(G_{\text{SOR}})|\right)^n = \rho_{G_{\text{SOR}}}^n,$$

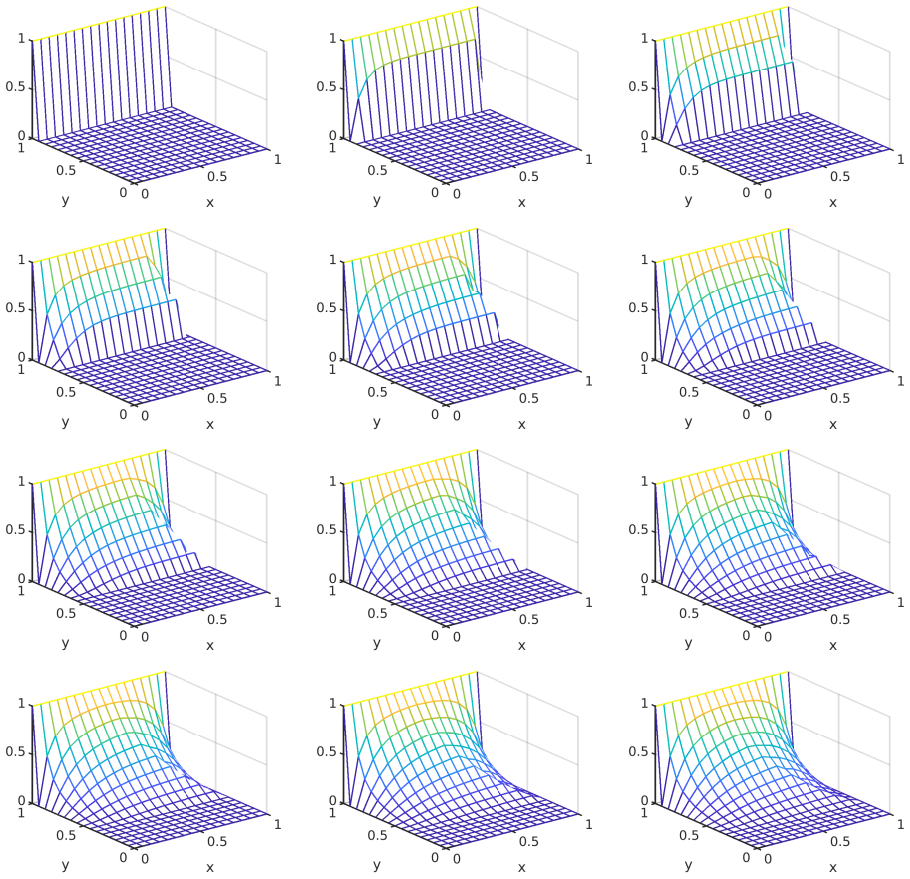


Figure 2.8: Initial guess and first iterations of the SOR method (2.32) applied to our Laplace model problem from Section 1.3.

and thus implies the result after taking the n-th root. □

From this elegant result, we can conclude that

$$\rho_{G_{\text{SOR}}} < 1 \quad \Longrightarrow \quad 0 < \omega < 2,$$

and thus for convergence of SOR it is necessary to choose $0 < \omega < 2$.

The next result is a general convergence result for SOR, due to Ostrowski and Reich[3]. It is for a restricted class of matrices, and does not answer the question yet on how to choose ω in order to obtain a fast method.

Theorem 11 (Ostrowski-Reich)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and invertible. Consider the usual splitting $A = D + L + L^\top$ with D diagonal and positive*

[3]Reich established this result for Gauss-Seidel in [104], and Ostrowski for the general SOR case in [95].

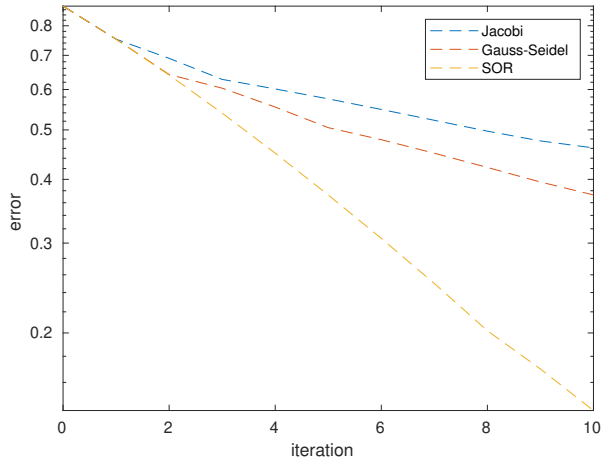


Figure 2.9: Errors measured for the iterates shown for Jacobi in Figure 2.3, Gauss-Seidel in Figure 2.5 and SOR in Figure 2.8.

definite, $D_{jj} > 0$ for $j = 1, \ldots, n$. Then SOR converges for all $0 < \omega < 2$ if and only if A is positive definite.

Proof. Since A is symmetric, recalling (2.33) the SOR iteration is

$$\underbrace{\left(\frac{1}{\omega}D + L\right)}_{M}\boldsymbol{u}_{k+1} = \underbrace{\left(\frac{1-\omega}{\omega}D - L^\top\right)}_{N}\boldsymbol{u}_k + \boldsymbol{f}.$$

M is non-singular, since $D_{jj} > 0$. Consider

$$Q = N + M^\top = \frac{2-\omega}{\omega}D.$$

Q is positive definite because $D_{jj} > 0$ and $0 < \omega < 2$. Now we can apply Theorem 6 (Householder-John) to conclude the proof. □

Theorem 11 implies that Gauss-Seidel and SOR applied to the discrete Laplace problem converge, as we observed already in Figures 2.8 and 2.9.

An important question, which remained unanswered so far, is how to choose the parameter ω in SOR in order to obtain a fast method. The pioneer in this area was David Young, who answered this question for a large class of matrices in his PhD thesis [128][4].

Definition 8 (Property A)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ has* Property A *if there exists a permutation matrix P such that*

$$P^\top AP = \begin{bmatrix} D_1 & F \\ E & D_2 \end{bmatrix}, \text{ with } D_1 \text{ and } D_2 \text{ diagonal.} \tag{2.36}$$

[4] An electronic version is available at `http://www.ma.utexas.edu/CNA/DMY/`

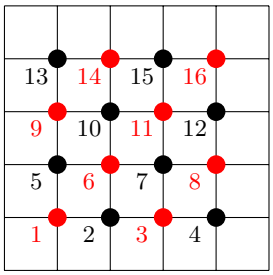


Figure 2.10: Discretization grid and chessboard used to define the red-black ordering.

Example 7. *If we discretize the one-dimensional Laplacian* $\frac{\partial^2}{\partial x^2}$ *by finite differences on a regular grid, the discrete operator becomes*

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

If n *is even, then* A *can be permuted as required for Property A,*

$$P^\top AP = \frac{1}{h^2}\left[\begin{array}{cccc|cccc} -2 & & & & 1 & & & \\ & \ddots & & & 1 & \ddots & & \\ & & \ddots & & & \ddots & \ddots & \\ & & & -2 & & & 1 & 1 \\ \hline 1 & 1 & & & -2 & & & \\ & \ddots & \ddots & & & \ddots & & \\ & & \ddots & 1 & & & \ddots & \\ & & & 1 & & & & -2 \end{array}\right],$$

with P *constructed by ordering first the odd canonical vectors* $\boldsymbol{e}_j$ *and then the even ones as follows:*

$$P = \begin{bmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_3 & \boldsymbol{e}_5 & \cdots & \boldsymbol{e}_{n-1} & \boldsymbol{e}_2 & \boldsymbol{e}_4 & \boldsymbol{e}_6 & \cdots & \boldsymbol{e}_n \end{bmatrix}.$$

To see Property A for the 2-dimensional discrete Laplace it is possible to use the so-called red-black ordering. To do so, we consider a grid of m *interior points in each direction to approximate the domain* Ω*, see Figure 1.10, and then assign a red or black color to each interior node of the grid in a chessboard fashion; see Figure 2.10. The red-black ordering consists in collecting in a vector first the red elements and then the black ones. For example, for the red-black chess board given in Figure 2.10 we have*

$$\boldsymbol{u}_{\mathrm{RB}} = \begin{bmatrix} u_1 & u_3 & u_6 & \cdots & u_{16} & u_2 & u_4 & u_5 & \cdots & u_{15} \end{bmatrix}.$$

To obtain the vector $\boldsymbol{u}_{\mathrm{RB}}$*, one can consider the permutation matrix*

$$P = \begin{bmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_3 & \boldsymbol{e}_6 & \cdots & \boldsymbol{e}_{16} & \boldsymbol{e}_2 & \boldsymbol{e}_4 & \boldsymbol{e}_5 & \cdots & \boldsymbol{e}_{15} \end{bmatrix}.$$

This matrix allows us to obtain $\boldsymbol{u}_{\mathrm{RB}} = P^\top \boldsymbol{u}$ *and a matrix* $P^\top A P$ *having the same structure as in* (2.36)*.*

For matrices with Property A, a useful relation exists between the eigenvalues of the associated Jacobi and SOR iteration matrices. We first need

Lemma 5. *Let* B *be a matrix having a zero diagonal and the block structure*

$$B = \begin{bmatrix} 0 & F \\ E & 0 \end{bmatrix} = L + U,$$

where L *and* U *are the strictly lower- and upper-triangular parts. If* μ *is an eigenvalue of* B*, then so is* $-\mu$*. Furthermore,* B *and* $\alpha L + \frac{1}{\alpha} U$ *are similar, and thus have the same eigenvalues for all* $\alpha \neq 0$*.*

Proof. Consider the diagonal matrix

$$S = \begin{bmatrix} I & 0 \\ 0 & \alpha I \end{bmatrix} \quad \Longrightarrow \quad SBS^{-1} = \begin{bmatrix} 0 & \frac{1}{\alpha} F \\ \alpha E & 0 \end{bmatrix} = \alpha L + \frac{1}{\alpha} U.$$

Furthermore, if $\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}$ is an eigenvector of B,

$$B \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = \mu \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} \;\Rightarrow\; \begin{bmatrix} F\boldsymbol{v} \\ E\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \mu \boldsymbol{u} \\ \mu \boldsymbol{v} \end{bmatrix},$$

then $\begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{v} \end{bmatrix}$ is an eigenvector with eigenvalue $-\mu$, because

$$B \begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{v} \end{bmatrix} = \begin{bmatrix} -F\boldsymbol{v} \\ E\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} -\mu \boldsymbol{u} \\ \mu \boldsymbol{v} \end{bmatrix} = -\mu \begin{bmatrix} \boldsymbol{u} \\ -\boldsymbol{v} \end{bmatrix}.$$

□

Theorem 12 (Relation between eigenvalues of G_{SOR} and G_{J})**.** *Let* $A \in \mathbb{R}^{n \times n}$ *have Property A, and let*

$$\widetilde{A} = P^\top A P = \begin{bmatrix} D_1 & F \\ E & D_2 \end{bmatrix} = L + D + U,$$

with all diagonal elements of D_1 *and* D_2 *nonzero. Let*

$$G_{\mathrm{SOR}} = (D + \omega L)^{-1}(-\omega U + (1 - \omega) D)$$

be the SOR iteration matrix for $\widetilde{A}$ *and*

$$G_{\mathrm{J}} = -D^{-1}(L + U)$$

the Jacobi iteration matrix. Assume that $\omega \neq 0$ and take a $\lambda \neq 0$. Then we have that λ is an eigenvalue of G_{SOR} if and only if μ, solution of

$$(\lambda + \omega - 1)^2 = \lambda \omega^2 \mu^2,$$

is an eigenvalue of G_{J}.

Proof. As before, we have

$$\begin{aligned} G_{\mathrm{SOR}} &= (D + \omega L)^{-1} D D^{-1} (-\omega U + (1-\omega) D) \\ &= (I + \omega D^{-1} L)^{-1} (-\omega D^{-1} U + (1-\omega) I), \end{aligned}$$

therefore λ is an eigenvalue if and only if

$$\begin{aligned} & \det((I + \omega D^{-1} L)^{-1} (-\omega D^{-1} U + (1-\omega) I) - \lambda I) = 0 \\ \iff & \det(-\omega D^{-1} U + (1-\omega) I - \lambda (I + \omega D^{-1} L)) = 0 \\ \iff & \det\left((\lambda + \omega - 1) I + \omega D^{-1} (\lambda L + U)\right) = 0. \end{aligned} \tag{2.37}$$

Now factoring out $\omega\sqrt{\lambda}$ (which is non-zero by assumption) in the determinant, we get

$$(2.37) \iff \det\left(\frac{\lambda + \omega - 1}{\omega \sqrt{\lambda}} I + D^{-1} (\sqrt{\lambda} L + \frac{1}{\sqrt{\lambda}} U) \right) = 0. \tag{2.38}$$

This equation means that $\frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}$ is an eigenvalue of $-D^{-1}(\sqrt{\lambda}L + \frac{1}{\sqrt{\lambda}}U)$. Using Lemma 5, we obtain

$$-D^{-1}(\sqrt{\lambda}L + \frac{1}{\sqrt{\lambda}}U) = -(\sqrt{\lambda} D^{-1} L + \frac{1}{\sqrt{\lambda}} D^{-1} U) \text{ is similar to } G_{\mathrm{J}} = -D^{-1}(L+U),$$

therefore they have the same eigenvalues. Since μ and $-\mu$ are eigenvalues of G_{J}, we get

$$\pm\mu = \frac{\lambda + \omega - 1}{\omega \sqrt{\lambda}} \quad \iff \quad (\lambda + \omega - 1)^2 = \lambda \omega^2 \mu^2. \tag{2.39}$$

□

We are now ready to prove the most important result for SOR methods: the optimal choice of the relaxation parameter ω, that is the $\omega \in (0, 2)$ that minimizes $\rho_{G_{\mathrm{SOR}}}(\omega)$, given by Young in his thesis [128].

Theorem 13 (Optimal choice of ω - David Young 1950)**.** *Let A, $\widetilde{A}$ and G_{J} be defined as in Theorem 12. If the eigenvalues $\mu(G_{\mathrm{J}})$ are real and $\rho_{G_{\mathrm{J}}} < 1$, then the optimal SOR parameter ω for $\widetilde{A}$ is*

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho_{G_{\mathrm{J}}}^2}}.$$

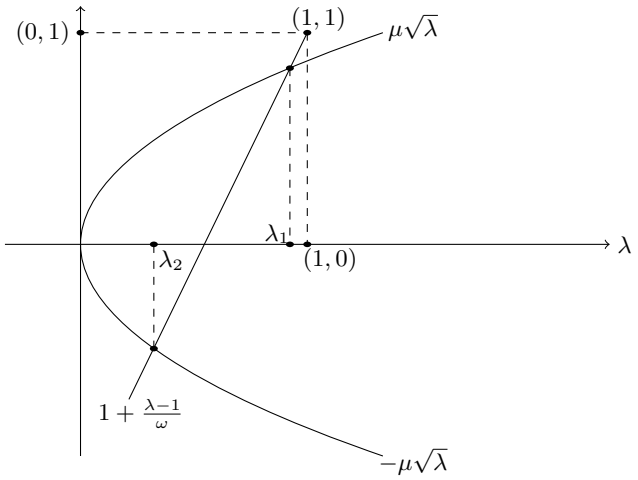


Figure 2.11: Relation between the roots of Jacobi and SOR. Assuming that $\lambda \in \mathbb{R}$, the straight line $\lambda \mapsto 1 + \frac{\lambda-1}{\omega}$ intersects the parabola $\pm\mu\sqrt{\lambda}$ in two points corresponding to λ_1 and λ_2.

Proof. Consider the SOR iteration matrix G_{SOR} of $\widetilde{A}$ as in Theorem 12, our goal is to solve the problem

$$\min_{\omega\in(0,2)} \rho_{G_{\mathrm{SOR}}}(\omega).$$

From (2.39), we obtain

$$\pm\mu\sqrt{\lambda} = \frac{\lambda + \omega - 1}{\omega}. \tag{2.40}$$

For μ fixed, the left-hand side of (2.40) is the equation of a parabola, whereas the right-hand side is that of a straight line passing through the point (1,1) with slope $1/\omega$. Figure 2.11 shows the roots of this equation. We see that the roots can be real or complex, depending on ω. If the roots are real, λ_1 is always bigger in modulus than λ_2, and λ_1 is increasing in μ, and decreasing in ω. Solving for λ, we obtain the quadratic equation

$$\lambda^2 + (2(\omega - 1) - \omega^2\mu^2)\lambda + (\omega - 1)^2 = 0 \tag{2.41}$$

with the solutions

$$\lambda_{1,2} = \frac{1}{2}\left(\omega^2\mu^2 - 2(\omega - 1) \pm \sqrt{\omega^2\mu^2(\omega^2\mu^2 - 4(\omega - 1))}\right).$$

With the discriminant $d(\omega, \mu) := \omega^2\mu^2 - 4(\omega - 1)$, we have that $d(0, \mu) = 4$ and $d(2, \mu) = -4 + 4\mu^2 < 0$, and d becomes zero if

$$4 - 4\omega + \omega^2\mu^2 = 0 \iff \omega_{1,2} = \frac{2 \pm 2\sqrt{1 - \mu^2}}{\mu^2}.$$

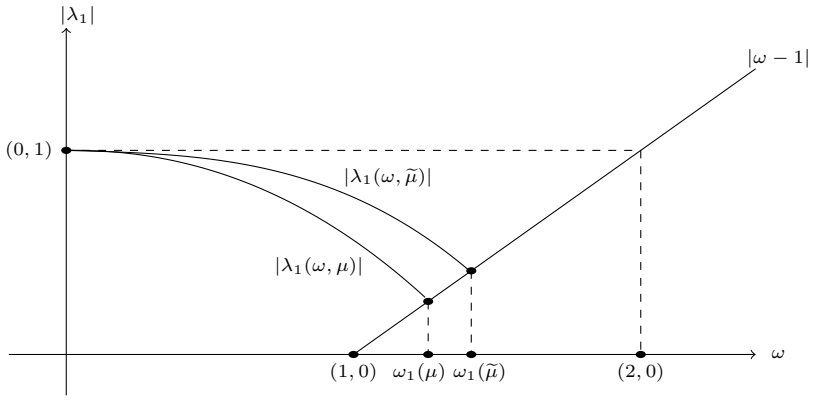


Figure 2.12: Value of $|\lambda_1|$ as functions of ω. The two curves correspond to different values μ and $\widetilde{\mu}$ such that $\mu < \widetilde{\mu}$.

Of the two possible values for ω, we only have to consider the smaller one,

$$\omega_1(\mu) = \frac{2 - 2\sqrt{1-\mu^2}}{\mu^2} = \frac{2}{1+\sqrt{1-\mu^2}},$$

since the second value $\omega_2 > 2$ can not lead to a convergent method, see Theorem 10. Thus for $\omega \in (0, \omega_1)$, the eigenvalues are real and λ_1 is the bigger one in modulus.

For larger values $\omega \in (\omega_1, 2)$, the discriminant is negative and $\lambda_{1,2}$ are complex conjugates (hence $|\lambda_1| = |\lambda_2|$). From (2.41), we see that[5] $\lambda_1 \lambda_2 = (\omega - 1)^2$, so that in the complex case we have $|\lambda_1| = |\omega - 1|$. We thus obtain for each eigenvalue μ of the Jacobi method the corresponding curve shown in Figure 2.12, and since in the real case λ_1 is increasing with μ, we obtain

$$\rho_{G_{\mathrm{SOR}}} = \begin{cases} \frac{1}{2}\left(\omega^2\rho_{G_{\mathrm{J}}}^2 - 2(\omega-1) + \sqrt{\omega^2\rho_{G_{\mathrm{J}}}^2(\omega^2\rho_{G_{\mathrm{J}}}^2 - 4(\omega-1))}\right), \\ \qquad\qquad \text{for } \omega \in (0,\omega_1), \text{ decreasing}, \\ |\omega - 1|, \qquad \text{for } \omega \in (\omega_1, 2), \text{ linearly increasing}. \end{cases}$$

The minimum of $\rho_{G_{\mathrm{SOR}}}(\omega)$ is thus reached for $\omega^* = \omega_1(\rho_{G_{\mathrm{J}}})$. □

For ω^*, the optimized convergence factor of SOR becomes

$$\rho_{G_{\mathrm{SOR}}}^* = \omega^* - 1 = \frac{1 - \sqrt{1-\rho_{G_{\mathrm{J}}}^2}}{1+\sqrt{1-\rho_{G_{\mathrm{J}}}^2}} = \left(\frac{\rho_{G_{\mathrm{J}}}}{1+\sqrt{1-\rho_{G_{\mathrm{J}}}^2}}\right)^2. \tag{2.42}$$

In general, we do not know in advance the spectral radius $\rho_{G_{\mathrm{J}}}$. So we cannot compute ω^* for SOR and have to rely on estimates. The rule of thumb is to try

[5] To see this, notice that $\lambda_1 + \lambda_2 = \omega^2\mu^2 - 2(\omega - 1)$. Hence, from (2.41) we have $\lambda_1^2 + -(\lambda_1 + \lambda_2)\lambda_1 + (\omega-1)^2 = 0$, which implies that $\lambda_1\lambda_2 = (\omega-1)^2$.

to overestimate ω, because of the steeper slope on the left, see Figure 2.13. To illustrate this, we consider the MATLAB function

```
function r=Rho(omega,mu);
% RHO computes spectral radius for SOR
%   r=Rho(omega,mu) computes the spectral radius of the iteration
%   matrix for SOR, given the spectral radius mu of the Jacobi
%   iteration matrix and the relaxation parameter omega

if omega<2/(1+sqrt(1-mu^2))
  r=((omega*mu)^2-2*(omega-1)+sqrt((omega*mu)^2*...
    ((omega*mu)^2-4*(omega-1))))/2;
else
  r=abs(omega-1);
end;
```

and plot it for various values of μ using the commands

```
axis([0,2,0,1])
hold on
xx=[0:0.01:2];
for mu=0.1:0.1:0.9
  yy=[];
  for x=xx
    yy=[yy rho(x,mu)];
  end
  plot(xx,yy)
end
```

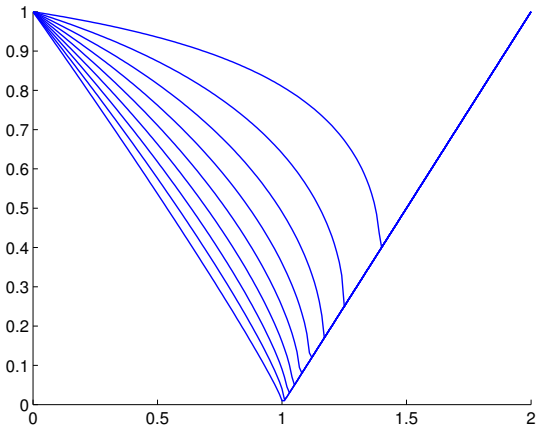


Figure 2.13: $\rho_{G_{\mathrm{SOR}}}$ as function of ω for $\mu = 0.1 : 0.1 : 0.9$. Notice that $\rho_{G_{\mathrm{SOR}}}$ increases with μ.

We can see that for $\mu = 0.9$, the choice of $\omega \approx 1.4$ yields a convergence factor

of about 0.4. This is a drastic improvement over Jacobi, leading to about 8 to 9 times fewer iterations, since $\mu^{8.5} \approx 0.4$.

The optimal choice of the relaxation parameter ω also improves the convergence factor asymptotically: one gains a square root, as one can see when setting $\rho_{G_J} = 1 - \varepsilon$, and then expanding the corresponding optimized $\rho(G_{SOR})$ for small ε. Using the MAPLE commands

```
rho:=mu^2/(1+sqrt(1-mu^2))^2;
mu:=1-epsilon;
series(rho,epsilon,1);
```

we obtain $\rho(G_{SOR}) = 1 - 2\sqrt{2\varepsilon} + O(\varepsilon)$, which shows that indeed the improvement is a square root.

Notice that Theorem 13 allows us to compute (or estimate) the optimal parameter ω^* that minimizes the spectral radius of the iteration matrix $\widetilde{G}_{\mathrm{SOR}}(\omega)$ corresponding to the matrix $\widetilde{A} := P^\top A P$, and not to A. Consider the splittings $\widetilde{A} = \widetilde{D} + \widetilde{L} + \widetilde{U}$ and $A = D + L + U$, where $\widetilde{D}$ and D are diagonal, L and $\widetilde{L}$ strictly lower-triangular and U and $\widetilde{U}$ strictly upper-triangular. The Jacobi and SOR iteration matrices corresponding to $\widetilde{A}$ are

$$\widetilde{G}_{\mathrm{J}} = -\widetilde{D}^{-1}(\widetilde{L} + \widetilde{U}) \quad \text{and} \quad \widetilde{G}_{\mathrm{SOR}}(\omega) = (\widetilde{D} + \omega\widetilde{L})^{-1}(-\omega\widetilde{U} + (1-\omega)\widetilde{D}).$$

On the other hand, the Jacobi and SOR iteration matrices corresponding to A are

$$G_{\mathrm{J}} = -D^{-1}(L + U) \quad \text{and} \quad G_{\mathrm{SOR}}(\omega) = (D + \omega L)^{-1}(-\omega U + (1-\omega)D).$$

Now notice that the relations $\widetilde{D} = P^\top D P$ and $\widetilde{L} + \widetilde{U} = P^\top (L + U) P$ hold. Hence, one can show that

$$\widetilde{G}_{\mathrm{J}} = -\widetilde{D}^{-1}(\widetilde{L} + \widetilde{U}) = -P^\top D^{-1} P P^\top (L + U) P = P^\top G_{\mathrm{J}} P,$$

that means that G_{J} and $\widetilde{G}_{\mathrm{J}}$ are similar. However, the same similarity does not hold in general for $G_{\mathrm{SOR}}(\omega)$ and $\widetilde{G}_{\mathrm{SOR}}(\omega)$, as the following example shows.

Example 8. *Consider the matrix*

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & \frac{1}{2} & 1 \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix}$$

and the matrix

$$\widetilde{A} := P_{14} A P_{14}^\top = \begin{bmatrix} 1 & \frac{1}{2} & 0 & 0 \\ 1 & 1 & \frac{1}{2} & 0 \\ 0 & 0 & 1 & \frac{1}{2} \\ 1 & 0 & 1 & 1 \end{bmatrix}, \text{ where } P_{14} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Consider the splittings $A = D + L + U$ and $\widetilde{A} = \widetilde{D} + \widetilde{L} + \widetilde{U}$, where D and $\widetilde{D}$ are diagonal, L and $\widetilde{L}$ are strictly lower triangular, and U and $\widetilde{U}$ are strictly

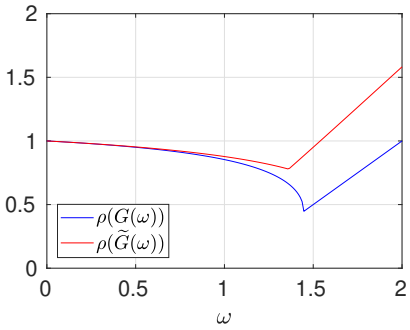


Figure 2.14: Spectral radii $\rho(G(\omega))$ and $\rho(\widetilde{G}(\omega))$ corresponding to the SOR iteration matrices constructed in Example 8.

upper triangular. The corresponding SOR iteration matrices are $G_{\mathrm{SOR}}(\omega) = (D+\omega L)^{-1}(-\omega U + (1-\omega)D)$ *and* $\widetilde{G}_{\mathrm{SOR}}(\omega) = (\widetilde{D}+\omega\widetilde{L})^{-1}(-\omega\widetilde{U} + (1-\omega)\widetilde{D})$. *Their spectral radii are shown in Figure 2.14 as functions of* $\omega \in [0,2]$. *It is clear that* $\rho(G(\omega))$ *and* $\rho(\widetilde{G}(\omega))$ *do not coincide, most notably for* $\omega > 1$. *Therefore the two matrices* $G(\omega)$ *and* $\widetilde{G}(\omega)$ *are not similar. Notice also that* $\rho(G(\omega))$ *and* $\rho(\widetilde{G}(\omega))$ *attain their minima at different* ω.

Example 8 shows that $G_{\mathrm{SOR}}(\omega)$ and $\widetilde{G}_{\mathrm{SOR}}(\omega)$ are not in general similar. Therefore, the optimal parameter ω^* for $\widetilde{G}_{\mathrm{SOR}}(\omega)$ does not necessarily coincide with the optimal parameter for $G_{\mathrm{SOR}}(\omega)$. However, if A is the discrete Laplace operator obtained by a finite-difference discretization, ω^* for $\widetilde{G}_{\mathrm{SOR}}(\omega)$ coincides with the optimal parameter for $G_{\mathrm{SOR}}(\omega)$. To see it, we need to introduce the following property:

Definition 9 (Property B)**.** *A matrix* $A \in \mathbb{R}^{n\times n}$ *is said to have Property B (or "Banholzer Property")*[6]*, if*

$$\det\left(\alpha D + \beta L + \beta^{-1}U\right) = \det\left(\alpha D + L + U\right),$$

for the usual splitting $A = D + L + U$ *and all* $\alpha, \beta \in \mathbb{R}\setminus\{0\}$.

The next theorem shows that Property A and Property B are related; see [64, Theorem 10.1.5] and [76, Lemma 10.6].

Theorem 14 (Property A and Property B)**.** *If a matrix* A *has Property A, then there exists a permutation matrix* P *such that* $P^\top AP$ *has Property B.*

Proof. The proof can be found in [76, Lemma 10.6]. □

[6]This property has been already discussed in the literature; see, e.g., [64, 76, 127]. However, the term "Property B" was invented by Stefan Banholzer, assistant of the course in Konstanz. His idea was to mimic the very neutral name "Property A" introduced by David Young. The poor Stefan didn't realize that the "B" could stay for "Banholzer"! Hence, we also called it the "Banholzer Property" and wish him all the best!

Theorem 15. *Assume that the matrix $A \in \mathbb{R}^{n\times n}$ has Property B with all diagonal elements of D being nonzero and let G_{J} and $G_{\mathrm{SOR}}(\omega)$ be the iteration matrices of the Jacobi method and the SOR method for the matrix A. Assume that $\omega \neq 0$ holds and take a $\lambda \neq 0$. Then λ is an eigenvalue of $G_{\mathrm{SOR}}(\omega)$ if and only if the solution μ of*

$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2 \tag{2.43}$$

is an eigenvalue of G_{J}.

Proof. Following exactly the lines of the proof of Theorem 12 one gets that λ is an eigenvalue of $G_{\mathrm{SOR}}(\omega)$ if and only if (compare (2.38))

$$\det\left(\frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}I + D^{-1}\left(\sqrt{\lambda}L + \frac{1}{\sqrt{\lambda}}U\right)\right) = 0. \tag{2.44}$$

Since the diagonal matrix D is invertible by assumption, we have $\det(D) \neq 0$, and hence (2.44) is equivalent to

$$\det\left(\frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}D + \sqrt{\lambda}L + \frac{1}{\sqrt{\lambda}}U\right) = 0. \tag{2.45}$$

Using Property B yields

$$(2.45) \iff \det\left(\frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}D + L + U\right) = 0. \tag{2.46}$$

Finally, since $\det(D^{-1}) \neq 0$ we can conclude that

$$(2.46) \iff \det\left(\frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}I - G_{\mathrm{J}}\right) = 0, \tag{2.47}$$

which is equivalent to $\mu = \frac{\lambda+\omega-1}{\omega\sqrt{\lambda}}$ being an eigenvalue of the matrix G_{J}. □

With this result we can reformulate Theorem 13 to show the optimal choice of the parameter ω for the SOR iteration matrix of the matrix A.

Theorem 16 (Optimal choice of ω for matrices having Property B)**.** *Assume that the matrix $A \in \mathbb{R}^{n\times n}$ has Property B with all diagonal elements of D being nonzero and let G_{J} and $G_{\mathrm{SOR}}(\omega)$ be the iteration matrices of the Jacobi method and the SOR method for the matrix A. If all eigenvalues $\mu(G_{\mathrm{J}})$ are real and $\rho_{G_{\mathrm{J}}} < 1$, then the optimal SOR parameter ω for $G_{\mathrm{SOR}}(\omega)$ is*

$$\omega^* = \frac{2}{1+\sqrt{1-\rho_{G_{\mathrm{J}}}^2}}.$$

Proof. The proof is exactly the same as the one of Theorem 13, since we only use formula (2.43), which relates the eigenvalues of G_{J} with the eigenvalues of $G_{\mathrm{SOR}}(\omega)$. □

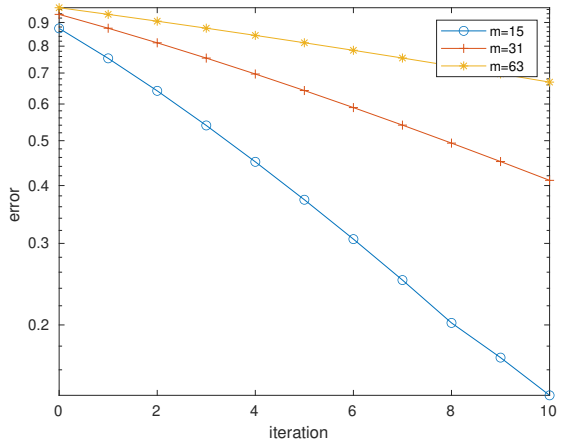


Figure 2.15: Convergence of the SOR method for different mesh sizes $h = \frac{1}{m+1}$.

Theorems 13 and 16 imply that, if a matrix A has both Property A and Property B, then the optimal parameters for the SOR iterations matrices corresponding to A and $\widetilde{A}$ coincide. This is exactly the case of the discrete Laplace operator:

Theorem 17 (Property B of the discrete Laplace operator)**.** *Let A be the discrete negative Laplace-operator in any dimension $d \in \mathbb{N}^+$ obtained by the second-order finite-difference scheme. Then A has Property B.*

Proof. A proof is given in [127, Lemma 1]. □

So can the optimized parameter in SOR remove the dependence on the mesh size of the convergence when solving discretized partial differential equations? We show in Figure 2.15 how the error decreases as the iterations progress when SOR is used to solve our Laplace model problem from Section 1.3 for the mesh we used for Figure 2.8 with $m = 15$ interior mesh points, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points. We see that like for Jacobi and Gauss-Seidel, although SOR is substantially faster, convergence still depends on the mesh parameter h, also when the optimized over-relaxation parameter is used. Like Jacobi and Gauss-Seidel, for finer and finer meshes, approximations will take longer and longer to travel across the grid. This can be seen very clearly if one replaces the expression (2.26), that is $\rho_{G_{\mathrm{J}}}(h) = \cos(\pi h)$, into formula (2.42) and computes the Taylor expansion:

$$\rho_{G_{\mathrm{SOR}}}(h) = \frac{1 - \sqrt{1 - \rho_{G_{\mathrm{J}}}(h)^2}}{1 + \sqrt{1 - \rho_{G_{\mathrm{J}}}(h)^2}} = \frac{1 - \sin(\pi h)}{1 + \sin(\pi h)} = 1 - 2\pi h + O(h^2).$$

This expansion shows that, on the one hand, the convergence of SOR deteriorates for $h \to 0$, but on the other hand the deterioration is slower compared to Jacobi and Gauss-Seidel. Therefore, the optimization of the parameter ω leads

to a substantial improvement: SOR is not only faster, but its convergence factor deteriorates significantly slowlier than the ones of Jacobi and Gauss-Seidel.

2.8 Richardson

> *It follows that by judiciously spacing the α's along the horizontal axis and by taking a sufficient number of such points (that is of approximations) the successive maxima and minima can be made all less, in absolute value, than any finite quantity ϵ however small.*
>
> Lewis F. Richardson, The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam, 1911

In order to explain the Richardson iteration, we consider the correction form (2.3) of the stationary iterative method,

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + M^{-1}\boldsymbol{r}_k.$$

If we choose $M^{-1} = \alpha I$, which corresponds to the splitting $M = \frac{1}{\alpha}I$ and $N = \frac{1}{\alpha}I - A$, we obtain what is often called the *Method of Richardson*,

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha(\boldsymbol{f} - A\boldsymbol{u}_k) = (I - \alpha A)\boldsymbol{u}_k + \alpha\boldsymbol{f}. \tag{2.48}$$

Lewis Fry Richardson however considered a much more interesting method, namely

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k(\boldsymbol{f} - A\boldsymbol{u}_k), \tag{2.49}$$

as one can see from the quote above and in Figure 2.16 from his original publication [105][7]. This is however a non-stationary method, and we will get back to such methods in Chapter 3. For the simple, stationary form of Richardson's method in (2.48), we have the following convergence theorem:

Theorem 18 (Convergence of Richardson's method)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then*

a) *Richardson converges if and only if* $0 < \alpha < \dfrac{2}{\rho(A)}$.

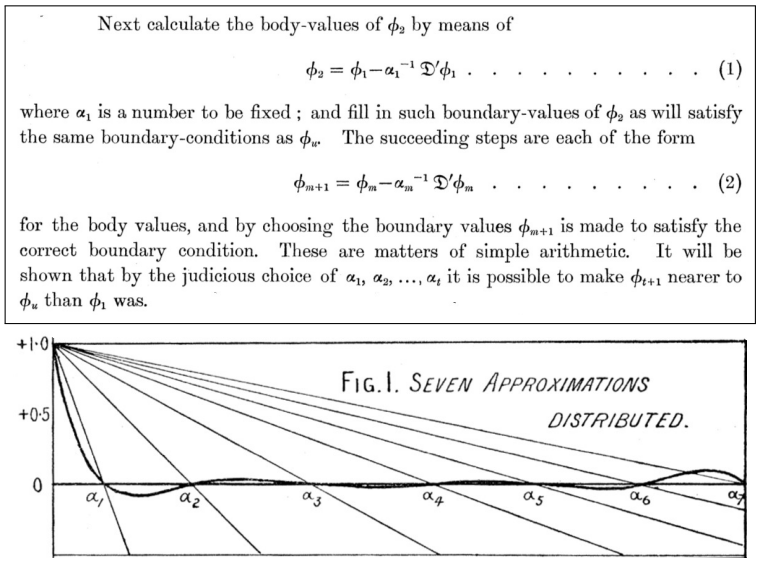
b) *The convergence is optimal for* $\alpha^* = \dfrac{2}{\lambda_{\max}(A) + \lambda_{\min}(A)}$.

c) *The convergence factor is* $\rho(I - \alpha^* A) = \dfrac{\kappa(A) - 1}{\kappa(A) + 1}$ *with* $\kappa(A) := \dfrac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$.

Proof. Since A is positive definite, we have

$$0 < \lambda_{\min}(A) \le \lambda_j(A) \le \lambda_{\max}(A) = \rho(A),$$

[7] This publication is a masterpiece of a scientific publication in the field of numerical analysis: it contains discretization techniques for partial differential equations, the invention of Richardson extrapolation to decrease the truncation error, a groundbreaking new iterative method with a polynomial approximation optimized in the sense of Chebyshev without knowing Chebyshev theory, and the application of all these inventions to simulate a masonry dam!



Next calculate the body-values of ϕ_2 by means of

$$\phi_2 = \phi_1 - \alpha_1^{-1}\,\mathfrak{D}'\phi_1 \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (1)$$

where α_1 is a number to be fixed; and fill in such boundary-values of ϕ_2 as will satisfy the same boundary-conditions as ϕ_u. The succeeding steps are each of the form

$$\phi_{m+1} = \phi_m - \alpha_m^{-1}\,\mathfrak{D}'\phi_m \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

for the body values, and by choosing the boundary values ϕ_{m+1} is made to satisfy the correct boundary condition. These are matters of simple arithmetic. It will be shown that by the judicious choice of $\alpha_1, \alpha_2, \ldots, \alpha_t$ it is possible to make ϕ_{t+1} nearer to ϕ_u than ϕ_1 was.

Figure 2.16: The true Richardson iterative method with a relaxation parameter which depends on the iteration, $\alpha = \alpha_k$, and a manually optimized choice by Richardson himself trying to minimize the *residual polynomial* without using Chebyshev theory.

for $j = 1, \ldots, n$. The iteration matrix $M^{-1}N = I - \alpha A$ has the eigenvalues $1 - \alpha\lambda_j(A)$, hence

$$\rho(I - \alpha A) < 1 \quad \Longleftrightarrow \quad 1 - \alpha\lambda_{\min}(A) < 1 \;\text{ and }\; 1 - \alpha\lambda_{\max}(A) > -1.$$

Thus we have convergence for

$$0 < \alpha < \frac{2}{\lambda_{\max}(A)}.$$

To minimize $\rho(I - \alpha A)$, we consider Figure 2.17, which shows the curves $|1 - \alpha\lambda_{\max}|$ and $|1 - \alpha\lambda_{\min}|$. We see that the optimal α is determined by the intersection point between the two lines, since the curves $|1 - \alpha\lambda_j(A)|$ for other j lie in between these two lines. This means

$$\alpha\lambda_{\max}(A) - 1 = 1 - \alpha\lambda_{\min}(A) \quad \Longrightarrow \quad \alpha^* = \frac{2}{\lambda_{\max}(A) + \lambda_{\min}(A)}.$$

For the optimal α, we get

$$\rho_{\text{opt}} = \rho(I - \alpha^* A) = \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)} = \frac{\kappa(A) - 1}{\kappa(A) + 1} \tag{2.50}$$

with $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$. □

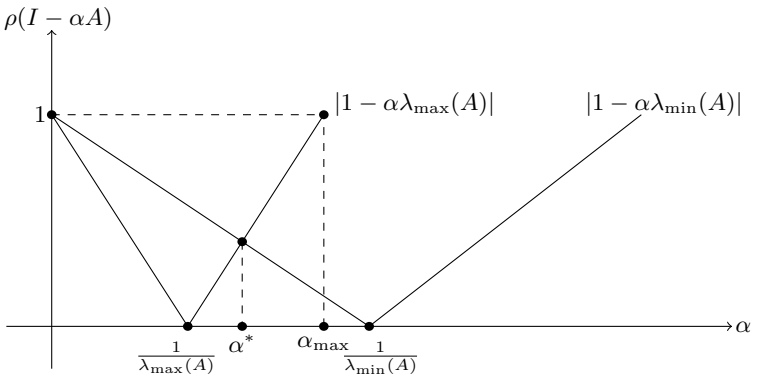


Figure 2.17: Determining an optimal α for the Richardson iteration.

For our Laplace model problem, the simplified Richardson iteration with the optimized relaxation parameter is equivalent to Jacobi's method, since the diagonal is simply a constant in this case, see Problem 21. We show in Figure 2.18 the first iterations of the true Richardson method using five different relaxation parameters α_k determined as the roots of appropriate Chebyshev polynomials, see Problem 22. We see a very interesting convergence behavior: depending on which range of frequencies the parameter α_k addresses, the approximation converges with different components in the solution. In Figure 2.19 we show the errors measured for simplified Richardson with one optimized parameter α, compared to the true method of Richardson with variable α_k optimized for 5 or 10 precomputed values. We see that the true Richardson method is much faster, and convergence looks super-linear in intervals corresponding to the length of the number of optimized α_k values. The Krylov methods in the next chapter will also achieve this, and surprisingly in an automated fashion, so no tuning of parameters will be needed!

So can an optimized sequence of parameters in the true Richardson method remove the dependence on the mesh size of the convergence when solving discretized partial differential equations? We show in Figure 2.20 how the error decreases as the iterations progress when Richardson is used to solve our Laplace model problem from Section 1.3 for $m = 15$, 31 and 63 interior mesh points, and we see that still convergence depends on the mesh parameter h. Like for the other stationary iterative methods, for finer and finer meshes, approximations will take longer and longer to travel across the grid.

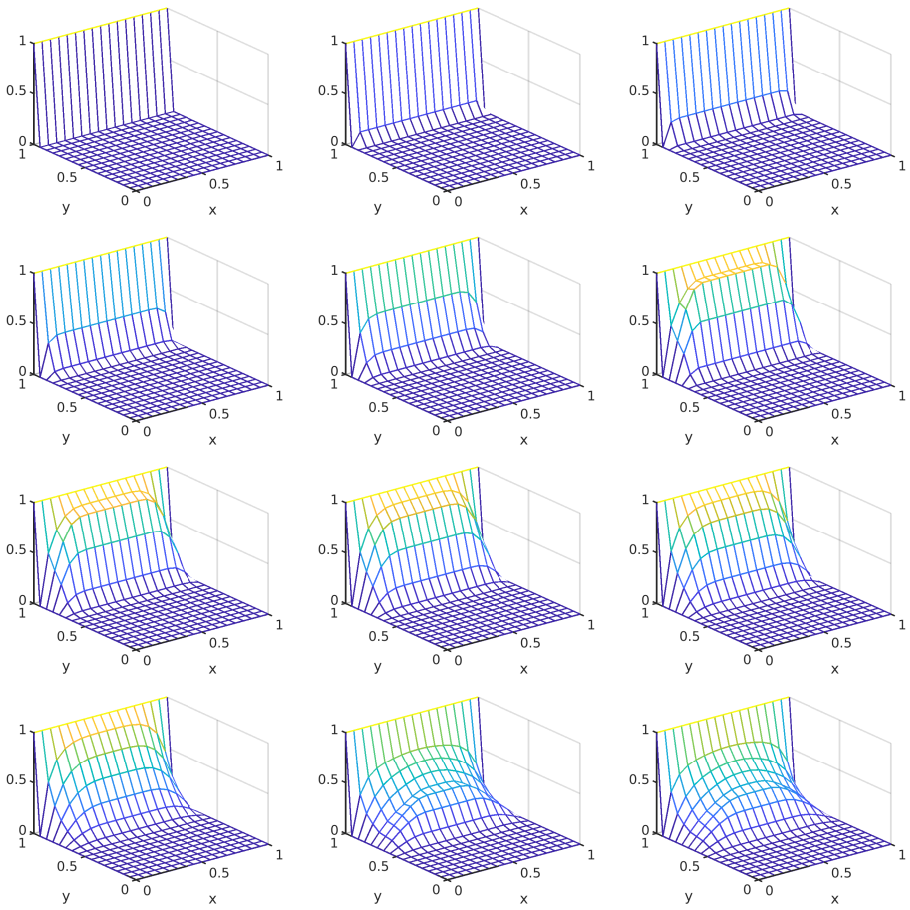


Figure 2.18: Initial guess and first iterations of the true Richardson method (2.49) with optimized α_k for 5 precomputed choices applied to our Laplace model problem from Section 1.3

2.9 Problems

Problem 4. *For $\boldsymbol{x} \in \mathbb{C}^m$ and $1 \leq p < \infty$, the $p-$norms and the $\infty-$norm are defined by*

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}}, \quad \|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq m} \{|x_i|\}.$$

For a matrix $A \in \mathbb{C}^{n \times m}$, the corresponding induced matrix norms are

$$\|A\|_{pq} = \sup_{\boldsymbol{x} \neq 0} \frac{\|A\boldsymbol{x}\|_p}{\|\boldsymbol{x}\|_q}, \quad \textit{for } 1 \leq p, q \leq \infty.$$

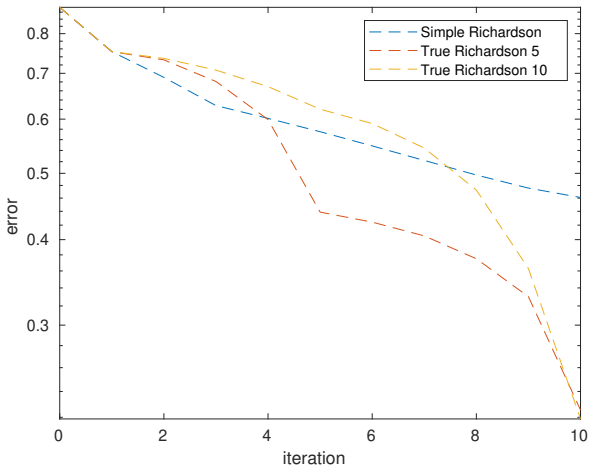


Figure 2.19: Errors measured for the simplified Richarson iteration with one optimized parameter compared to the true Richardson method (2.49) with optimized α_k for 5 and 10 precomputed choices applied to our Laplace model problem from Section 1.3.

In the particular case $p = q$, we consider the notation $\|\cdot\|_{pp} = \|\cdot\|_p$. A further common matrix norm is the Frobenius norm,

$$\|A\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2.$$

Prove the following four equalities, where $\rho(A) := \max\{|\lambda_i|\}$ is the spectral radius and λ_i are the eigenvalues of A.

$$\|A\|_1 = \max_{1\le j\le m}\sum_{i=1}^{n}|a_{ij}|, \quad \|A\|_\infty = \max_{1\le i\le n}\sum_{j=1}^{m}|a_{ij}|,$$
$$\|A\|_2 = (\rho(A^*A))^{\frac{1}{2}}, \qquad \|A\|_F = (\mathrm{trace}(A^*A))^{\frac{1}{2}} = (\mathrm{trace}(AA^*))^{\frac{1}{2}}.$$

Problem 5. *Prove that the Frobenius norm satisfies the sub-multiplicative property*

$$\|AB\|_F \le \|A\|_F\|B\|_F.$$

Is the Frobenius norm an induced norm? Why?

Problem 6. *Let $A \in \mathbb{C}^{n\times n}$. Then for any given $\epsilon > 0$, there exists a (vector) norm for $\mathbb{C}^n$ such that the induced matrix norm $\|\cdot\|$, which depends on A and ϵ, satisfies*

$$\rho(A) \le \|A\| \le \rho(A) + \varepsilon.$$

Problem 7. *For any matrix $G \in \mathbb{R}^{n\times n}$ with spectral radius $\rho(G)$ and any induced matrix norm, we have*

$$\lim_{k\to\infty} \|G^k\|^{\frac{1}{k}} = \rho(G).$$

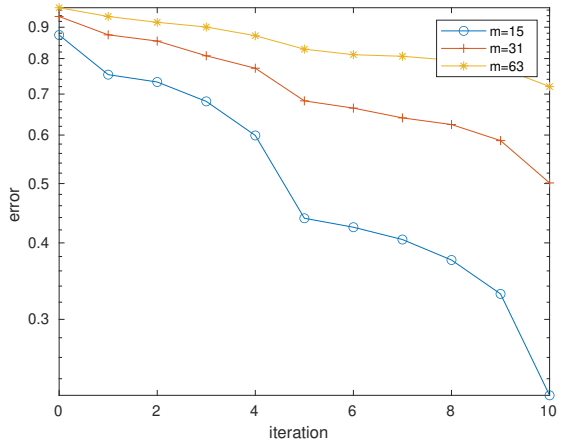


Figure 2.20: Convergence of the true Richardson method for different mesh sizes $h = \frac{1}{m+1}$.

Problem 8. *Prove that for a given positive integer p,*

$$\lim_{k\to\infty} \binom{k}{p}^{\frac{1}{k}} = 1,$$

and for $0 < \rho < 1$,

$$\lim_{k\to\infty} \binom{k}{p} \rho^k = 0.$$

Problem 9. *Let $B \in \mathbb{C}^{n\times n}$ be a square matrix with a spectral radius $\rho(B) < 1$. Prove that $I - B$ is invertible and that*

$$(I - B)^{-1} = \sum_{j=0}^{\infty} B^j.$$

This series is a particular case of the Neumann series.

Problem 10. *Show that the matrix A representing the discrete negative Laplacian obtained by the standard second-order finite-difference discretization in dimensions 1 and 2 is an M-matrix.*
Hint in 1D: consider a splitting $A = M - N$, show that $\rho(M^{-1}N) < 1$, and use Theorem 5. Notice that for a tridiagonal matrix $A \in \mathbb{R}^{n\times n}$ with Töplitz structure,

$$A = \begin{bmatrix} a & b & & \\ c & a & b & \\ & c & a & b \\ & & & \ddots \end{bmatrix}$$

the eigenvalues are given by $\lambda_k(A) = a + 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right)$ for $k = 1, \ldots, n$.
Hint in 2D:

- *Let A_x and A_y be the discrete matrices corresponding to $-\frac{d^2}{dx^2}$ and $-\frac{d^2}{dy^2}$. Then, the matrix A can be obtained as $A = A_x \otimes I + I \otimes A_y$, where $\otimes$ denotes the Kronecker product.*

- *Let $(\lambda_x, \boldsymbol{v}_x)$ and $(\lambda_y, \boldsymbol{v}_y)$ be the eigenpairs of A_x and A_y. Prove that the eigenvalues of A are given by $\lambda = \lambda_x + \lambda_y$. To show this property, consider the vector $\boldsymbol{w} := \boldsymbol{v}_x \otimes \boldsymbol{v}_y$ and the product $A\boldsymbol{w} = \big(A_x \otimes I + I \otimes A_y\big)\big(\boldsymbol{v}_x \otimes \boldsymbol{v}_y\big)$ using the formula $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$.*

- *Consider a splitting $A = M - N$, show that $\rho(M^{-1}N) < 1$, and use Theorem 5.*

Problem 11. *Write the iterative methods Jacobi, Gauss-Seidel, and SOR in their standard, correction, residual, and differences forms.*

Problem 12. *Write a* MATLAB *script for the solution of the Laplace equation*

$$\begin{aligned} \Delta u &= f \quad \text{in } \Omega = (0,1) \times (0,1), \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

by using the methods of Jacobi, Gauss-Seidel and SOR. The three iterative solvers have to be implemented in three independent functions, so they can be reused for arbitrary linear systems of equations.

Problem 13. *Use Theorem 8 to prove that the Jacobi method applied to the discrete Laplace problem converges.*

Problem 14. *Find an example of matrix A such that the Gauss-Seidel method converges.*

Problem 15. *Consider the iteration matrix $G = \begin{bmatrix} \frac{5}{2} & -1 \\ 1 & 0 \end{bmatrix}$. Consider the initial vector $\boldsymbol{v}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, compute the iterations $\boldsymbol{v}_1 = G\boldsymbol{v}_0$, $\boldsymbol{v}_2 = G\boldsymbol{v}_1$ and their norms. What do you observe? Repeat the experiment for the initial vector $\boldsymbol{w}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. Do you observe the same behavior? Why? Hint: compute eigenvalues and eigenvectors of G and recall Theorem 3.*

Problem 16. *Let A be the standard discrete five point Laplace matrix in dimension 2,*

$$A = -\begin{bmatrix} T & I & & \\ I & T & \ddots & \\ & \ddots & \ddots & I \\ & & I & T \end{bmatrix} \quad \text{with} \quad T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & 1 & \\ & & \ddots & \\ & & 1 & -4 \end{bmatrix}.$$

Consider the block-Jacobi splitting $A = D + L + L^\top = M - N$ with

$$M = D = -\begin{bmatrix} T & & & \\ & T & & \\ & & \ddots & \\ & & & T \end{bmatrix} \quad \text{and} \quad N = -\big(L + L^\top\big) = \begin{bmatrix} 0 & I & & \\ I & 0 & \ddots & \\ & \ddots & \ddots & I \\ & & I & 0 \end{bmatrix}.$$

Using Theorem 9, prove that the block-Jacobi method converges.
Hint: Prove that $-T$, A and $2D - A$ are positive definite.

Problem 17. *Compute the optimal parameter of the SOR method for the discrete negative Laplacian in dimensions 2. Do the same for the Richardson method.*

Problem 18. *Consider a matrix A having Property A. Prove that the method of Gauss-Seidel converges twice as fast as the method of Jacobi in this case.*
Hint: Use Theorem 12 and the formula $(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2$, where λ is an eigenvalue of the SOR iteration matrix, μ is an eigenvalue of the Jacobi iteration matrix, and ω is the relaxation parameter of the SOR method.

Problem 19. *Recall that the SOR iteration matrix is*

$$G_{\mathrm{SOR}} = \big(D + \omega L\big)^{-1}\Big(-\omega U + (1-\omega)D\Big).$$

If $\lambda = 0$ is an eigenvalue of G_{SOR}, what is the value of ω?

Problem 20. *Write a function in* MATLAB *that solves a linear system $A\boldsymbol{u} = \boldsymbol{f}$ by means of the SOR method. Test this function solving the problem*

$$\begin{aligned} -\Delta u &= 1 \ \text{ in } \Omega = (0,1) \times (0,1), \\ u &= 0 \ \text{ on } \partial\Omega. \end{aligned}$$

Moreover, using also the methods of Jacobi and Gauss-Seidel (Problem 12) try to reproduce Figure 2.21.

Problem 21. *Consider A the discrete matrix of $-\Delta$ in dimension 2. Show that the method of Jacobi and the method of Richardson (with the optimal parameter α_{opt} computed in Problem 17) have the same convergence behavior.*

Problem 22. *Implement the true Richardson method with a variable relaxation parameter α_k and a general symmetric and positive definite matrix. Experiment with different choices of α_k. Can you determine, for a fixed number of different relaxation parameters, an optimized choice of their values using Chebyshev polynomials, in the same spirit Richardson did this by hand for seven parameters in Figure 2.16?*

Problem 23. *Consider the 1D (negative) Laplacian matrix $A \in \mathbb{R}^{n\times n}$ given, e.g., in Example 4. Using the Sherman-Morrison formula, prove that*

$$A^{-1} = L^\top\Big(I - \frac{1}{n+1}(\boldsymbol{v}\boldsymbol{v}^\top)\Big)L,$$

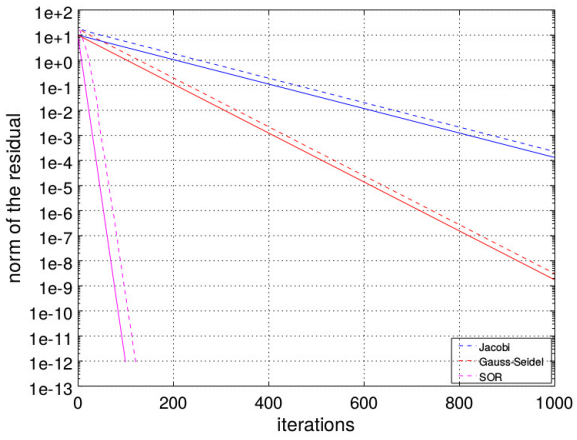


Figure 2.21: The blue, red, and magenta lines correspond to the methods of Jacobi, Gauss-Seidel, and SOR. In particular, the dashed lines correspond to the numerical experiments, whereas the solid lines represents the theoretical convergence estimates of the three methods.

where $\boldsymbol{v}^\top = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$ *and*

$$L = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & & \cdots & 1 \end{bmatrix}.$$

Chapter 3

Krylov Methods

Because of their overwhelming success in applications, Krylov subspace methods are counted among the "Top 10 Algorithms" of the 20th century.

Jörg Liesen and *Zdeněk Strakoš*, Krylov Subspace Methods, Principles and Analysis, 2013.

We have seen that, in his original method, Richardson used a different parameter at each iteration, namely

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k(\boldsymbol{f} - A\boldsymbol{u}_k), \tag{3.1}$$

and he tried to find a good sequence of parameters $\{\alpha_k\}_k$ to speed up the convergence, see Figure 2.16. This is an excellent idea, but it is much harder than the one parameter α we optimized resulting in Theorem 18. Richardson's dream was realized when *Gene Golub* invented the Chebyshev semi-iterative method in his PhD thesis in 1959 [61]. Using the roots of the Chebyshev polynomials, which are the smallest polynomials in a given interval in the $\|\cdot\|_{L^\infty}$-norm, and their recurrence relation, Gene managed to derive a method where the optimal α_k for a symmetric positive definite matrix with a given spectral interval estimate are derived on the fly using the recurrence relation of the Chebyshev polynomials. A precise description of this invention can be found in [57, Section 11.5]. There is however an even better approach, which does not require the estimate of the spectral interval of the matrix, which leads to the so called Krylov methods which we will study in this chapter, and which were selected among the top 10 algorithms of the 20th century, see the quote above.

We follow in the beginning the groundbreaking work of Eduard Stiefel in [117], where he describes for the first time the algorithm of conjugate gradients, the first Krylov method dating from 1951. These methods are called Krylov methods because they involve what is called a Krylov space, which also appear in a paper by *Nikolay Mitrofanovich Krylov* from 1931, who studied vibration phenomena arising in linear systems of second-order ordinary differential equations [82]. As one can see from the quotes in Section 3.2, Stiefel had already emphasized that the method of conjugate gradients both gives successive approximations to the solution as well as the solution after a finite number of steps.

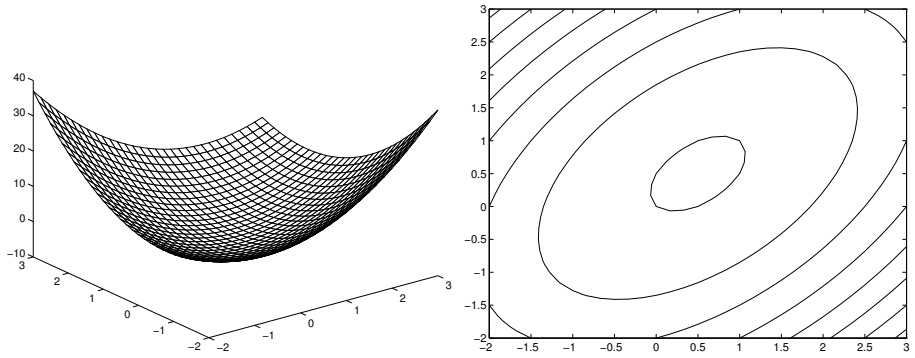


Figure 3.1: Example of a function $F(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^\top A\boldsymbol{u} - \boldsymbol{u}^\top \boldsymbol{f}$ which is minimized to find a solution to the linear system $A\boldsymbol{u} = \boldsymbol{f}$.

In the publication with Hestenes then the finite number of steps were more emphasized. This was misleading for the algorithm, which in finite-precision arithmetic rarely converges in a finite number of steps, and so it took almost a generation of numerical analysts to expand research into more general Krylov methods like GMRES [112], which we will also see in this chapter. We start however first with a simple idea coming from optimization.

3.1 Steepest descent

> *Es handelt sich hier im Grunde genommen um eine Anwendung des Ritzschen Gedankens, ein Minimum angenähert dadurch zu bestimmen, dass man die Konkurrenz auf eine leicht zu handhabende lineare Schar beschränkt.*
>
> *Eduard Stiefel*, Über einige Methoden der Relaxationsrechnung, 1952.

If the matrix A is symmetric and positive definite, then instead of solving the linear system of equations $A\boldsymbol{u} = \boldsymbol{f}$, we can minimize $F(\boldsymbol{u}) := \frac{1}{2}\boldsymbol{u}^\top A\boldsymbol{u} - \boldsymbol{u}^\top \boldsymbol{f}$ to find the solution. To see that, note that for a minimum of the function $F(\boldsymbol{u})$ we need that the gradient $\nabla F(\boldsymbol{u})$ is zero and the second derivative is positive. Now the gradient is, using that $A^\top = A$,

$$\nabla F(\boldsymbol{u}) = \frac{1}{2}A\boldsymbol{u} + \frac{1}{2}A^\top \boldsymbol{u} - \boldsymbol{f} = A\boldsymbol{u} - \boldsymbol{f}.$$

Setting the gradient to zero we see that the minimizer solves the linear system of equations $A\boldsymbol{u} = \boldsymbol{f}$. Since the Hessian matrix $\nabla^2 F(\boldsymbol{u}) = A$ and A is by assumption positive definite, we have indeed a minimum of $F(\boldsymbol{u})$ as the solution of $A\boldsymbol{u} = \boldsymbol{f}$[1]. Figure 3.1 shows a two-dimensional example of such a function $F(\boldsymbol{u})$.

[1]Since the Hessian of F coincides with A, which is positive definite, the Hessian is positive definite and thus F is strictly convex. Hence a necessary and sufficient condition for $\boldsymbol{u}$ to be the unique global minimum is $\nabla F(\boldsymbol{u}) = 0$; see, e.g., [12].

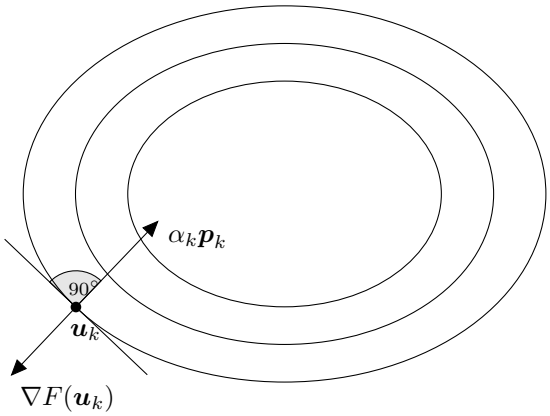


Figure 3.2: The method of steepest descent: the black curves are the level sets of F; the black dot represents the approximate solution $\boldsymbol{u}_k$ at the kth iterate; the two vectors represent the gradient of F at $\boldsymbol{u}_k$ and the new search direction $\boldsymbol{p}_k$ multiplied by the step-length α_k.

To find the minimum of $F(\boldsymbol{u})$, one can start at some initial guess $\boldsymbol{u}_0$ and then compute for $k = 0, 1, 2, \ldots$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k.$$

The question is how to choose the search directions $\boldsymbol{p}_k$ and the distance to go in that search direction, α_k.

Since we want to find a minimum, we could intuitively go into the direction where the function $F(\boldsymbol{u})$ decreases most, which is a reasonable tactic[2]. This would be in the opposite direction of the gradient of $F(\boldsymbol{u})$, as shown in Figure 3.2. The direction opposite to the gradient of $F(\boldsymbol{u}_k)$ is

$$-\nabla F(\boldsymbol{u}_k) = -(A\boldsymbol{u}_k - \boldsymbol{f}) = \boldsymbol{f} - A\boldsymbol{u}_k = \boldsymbol{r}_k,$$

and hence the search direction $\boldsymbol{p}_k$ equals the residual at step k, $\boldsymbol{r}_k$. Doing this, we find surprisingly again the method of Richardson (3.1)! But now, to determine how far we want to go in that direction, we can employ the tactic of going as long as it goes down, since we want to find a minimum, which is precisely the 'Ritzscher Gedanke' in the quote above. This is equivalent to minimizing the function $F(\boldsymbol{u}_k + \alpha_k \boldsymbol{r}_k)$ with respect to the parameter α_k to find the new iterate $\boldsymbol{u}_{k+1}$. Performing this minimization we obtain

$$\frac{d}{d\alpha_k} F(\boldsymbol{u}_{k+1}) = \frac{d}{d\alpha_k} F(\boldsymbol{u}_k + \alpha_k \boldsymbol{r}_k) = (\nabla F(\boldsymbol{u}_{k+1}))^\top \boldsymbol{r}_k = 0,$$

which means that we have to choose $\boldsymbol{u}_{k+1}$ such that $\nabla F(\boldsymbol{u}_{k+1})$ is orthogonal to the current residual $\boldsymbol{r}_k$. But $\nabla F(\boldsymbol{u}_{k+1}) \equiv -\boldsymbol{r}_{k+1}$ and thus we can compute the

[2]Stiefel was a colonel in the Swiss army, and carefully distinguished between a tactic, which is locally best, and a strategy to win globally.

parameter α_k as follows:

$$\begin{aligned} 0 &= \boldsymbol{r}_{k+1}^\top \boldsymbol{r}_k \\ &= (\boldsymbol{f} - A\boldsymbol{u}_{k+1})^\top \boldsymbol{r}_k \\ &= (\boldsymbol{f} - A(\boldsymbol{u}_k + \alpha_k \boldsymbol{r}_k))^\top \boldsymbol{r}_k \\ &= (\boldsymbol{f} - A\boldsymbol{u}_k - \alpha_k A\boldsymbol{r}_k)^\top \boldsymbol{r}_k \\ &= (\boldsymbol{r}_k - \alpha_k A\boldsymbol{r}_k)^\top \boldsymbol{r}_k \\ &= \boldsymbol{r}_k^\top \boldsymbol{r}_k - \alpha_k \boldsymbol{r}_k^\top A\boldsymbol{r}_k \end{aligned}$$

and hence, solving for α_k, we get

$$\alpha_k = \frac{\boldsymbol{r}_k^\top \boldsymbol{r}_k}{\boldsymbol{r}_k^\top A\boldsymbol{r}_k}. \tag{3.2}$$

So the algorithm of steepest descent for the linear system $A\boldsymbol{u} = \boldsymbol{f}$, where A is symmetric and positive definite is given in MATLAB as follows:

```
function [u,uk,res]=SteepestDescent(A,f,u0,tol,m)
% STEEPESTDESCENT solves Au=f using the steepest descent method
%   [u,uk,res]=SteepestDescent(A,f,u0,tol,m); solves Au=f using steepest
%   descent starting at the initial guess u0 up to a tolerance tol using
%   at most m iterations. A has to be symmetric positive definite.
%   SteepestDescent returns in the matrix uk the iterates, in u the
%   solution computed, and in res the history of the norm of the residuals.

if nargin<5, m=100; end          % default values
if nargin<4, tol=1e-6; end
r=f-A*u0;
res(1)=norm(r);
uk(:,1)=u0;
k=0;
while norm(r)/norm(f)>tol && k<=m
  k=k+1;
  al=r'*r/(r'*A*r);
  uk(:,k+1)=uk(:,k)+al*r;
  r=f-A*uk(:,k+1);
  res(k+1)=norm(r);
end
u=uk(:,k+1);
```

Applying Steepest Descent to our Laplace model problem from Section 1.3, we obtain for the first iterates the approximations shown in Figure 3.3. We see that the method converges, comparable to the convergence of Jacobi in Figure 2.3. We next prove that Steepest Descent is converging for such type of problems, and give a convergence estimate.

Theorem 19 (Convergence of steepest descent)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then the method of steepest descent converges, and the*

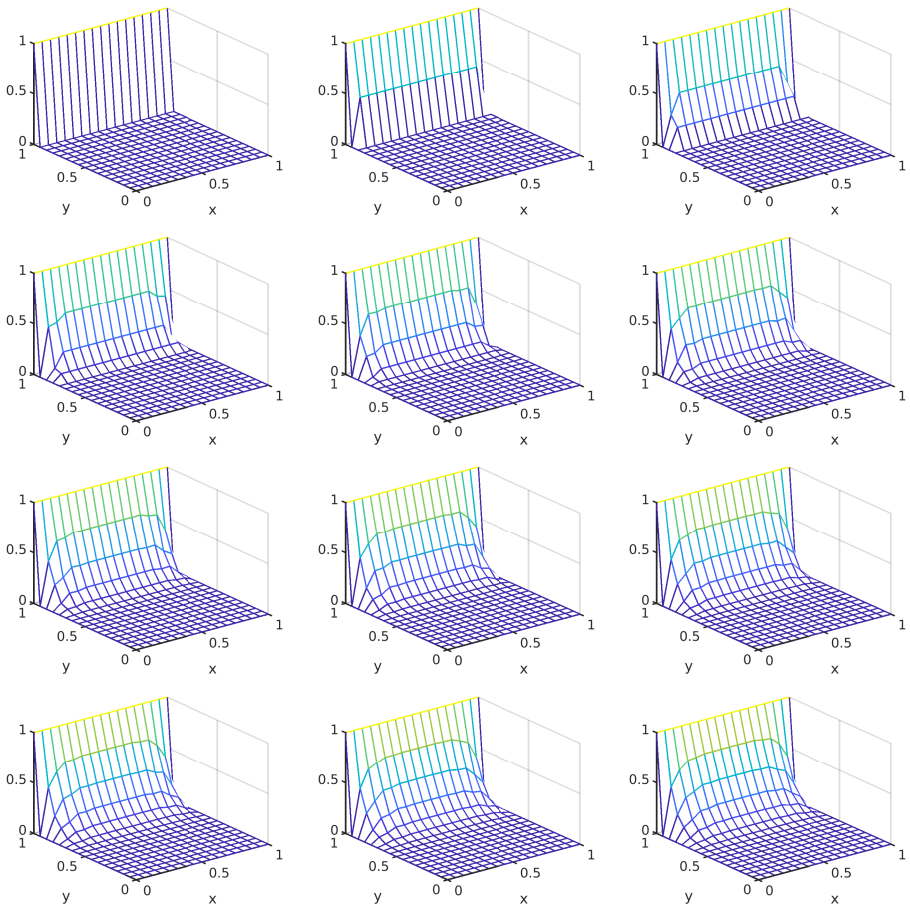


Figure 3.3: Initial guess and first iterations of Steepest Descent (2.49) applied to our Laplace model problem from Section 1.3.

errors satisfy the convergence estimate

$$\|\boldsymbol{e}_k\|_A \leq \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^k \|\boldsymbol{e}_0\|_A, \tag{3.3}$$

where $\kappa(A) := \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ *is the* spectral condition number *of* A*, and* $\|\boldsymbol{v}\|_A := \sqrt{\boldsymbol{v}^\top A \boldsymbol{v}}$.

Proof. To study if the sequence $\|\boldsymbol{e}_k\|_A$ converges to zero as k goes to infinity, recall that

$$\boldsymbol{e}_{k+1} = \boldsymbol{u} - \boldsymbol{u}_{k+1} = \boldsymbol{u} - \boldsymbol{u}_k - \alpha_k \boldsymbol{r}_k = \boldsymbol{e}_k - \alpha_k \boldsymbol{r}_k,$$

and, using that $A = A^\top$, compute

$$\begin{aligned}\|\boldsymbol{e}_{k+1}\|_A^2 &= \boldsymbol{e}_{k+1}^\top A\boldsymbol{e}_{k+1}\\ &= (\boldsymbol{e}_k - \alpha_k\boldsymbol{r}_k)^\top A(\boldsymbol{e}_k - \alpha_k\boldsymbol{r}_k)\\ &= \boldsymbol{e}_k^\top A\boldsymbol{e}_k - 2\alpha_k\boldsymbol{r}_k^\top A\boldsymbol{e}_k + \alpha_k^2\boldsymbol{r}_k^\top A\boldsymbol{r}_k.\end{aligned}$$

Since $A\boldsymbol{e}_k = A\boldsymbol{u} - A\boldsymbol{u}_k = \boldsymbol{f} - A\boldsymbol{u}_k = \boldsymbol{r}_k$, we obtain that

$$\begin{aligned}\|\boldsymbol{e}_{k+1}\|_A^2 &= \boldsymbol{e}_k^\top A\boldsymbol{e}_k - 2\alpha_k\boldsymbol{r}_k^\top\boldsymbol{r}_k + \alpha_k^2\boldsymbol{r}_k^\top A\boldsymbol{r}_k\\ &= \|\boldsymbol{e}_k\|_A^2 - 2\alpha_k\|\boldsymbol{r}_k\|_2^2 + \alpha_k^2\|\boldsymbol{r}_k\|_A^2\\ &= \|\boldsymbol{e}_k\|_A^2 - \alpha_k\|\boldsymbol{r}_k\|_2^2,\end{aligned}$$

where we used (3.2). Inserting again the value of α_k given by (3.2) and dividing by $\|\boldsymbol{e}_k\|_A^2$, we get

$$\frac{\|\boldsymbol{e}_{k+1}\|_A^2}{\|\boldsymbol{e}_k\|_A^2} = 1 - \frac{\|\boldsymbol{r}_k\|_2^4}{(\boldsymbol{r}_k^\top A\boldsymbol{r}_k)(\boldsymbol{e}_k^\top A\boldsymbol{e}_k)} = 1 - \frac{\|\boldsymbol{r}_k\|_2^4}{(\boldsymbol{r}_k^\top A\boldsymbol{r}_k)(\boldsymbol{r}_k^\top A^{-1}\boldsymbol{r}_k)},$$

where we used that $A\boldsymbol{e}_k = \boldsymbol{r}_k$ and $\boldsymbol{e}_k^\top = (A^{-1}\boldsymbol{r}_k)^\top = \boldsymbol{r}_k^\top A^{-1}$, since A is symmetric. By the Kantorovitch inequality, see Theorem 20 below, the right-hand side can be bounded using $\kappa := \kappa(A)$, the condition number of A, and we obtain

$$\frac{\|\boldsymbol{e}_{k+1}\|_A^2}{\|\boldsymbol{e}_k\|_A^2} \le 1 - \frac{4}{(\sqrt{\kappa} + \frac{1}{\sqrt{\kappa}})^2} = \left(\frac{\kappa - 1}{\kappa + 1}\right)^2 < 1,$$

which concludes the proof using induction. □

Note that since

$$\begin{aligned}\|\boldsymbol{r}_k\|_{A^{-1}}^2 &= (\boldsymbol{f} - A\boldsymbol{u}_k)^\top A^{-1}(\boldsymbol{f} - A\boldsymbol{u}_k)\\ &= (A^{-1}(\boldsymbol{f} - A\boldsymbol{u}_k))^\top A(A^{-1}(\boldsymbol{f} - A\boldsymbol{u}_k))\\ &= (\boldsymbol{u} - \boldsymbol{u}_k)^\top A(\boldsymbol{u} - \boldsymbol{u}_k)\\ &= \|\boldsymbol{u} - \boldsymbol{u}_k\|_A^2,\end{aligned}$$

Theorem 19 also gives immediately an estimate for the residual $\boldsymbol{r}_k$ in the norm $\|\boldsymbol{v}\|_{A^{-1}} = \sqrt{\boldsymbol{v}^\top A^{-1}\boldsymbol{v}}$.

Theorem 20 (Kantorovitch inequality)**.** *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Then for all $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{v} \neq 0$, we have*

$$1 \le \frac{(\boldsymbol{v}^\top A\boldsymbol{v})(\boldsymbol{v}^\top A^{-1}\boldsymbol{v})}{(\boldsymbol{v}^\top\boldsymbol{v})^2} \le \frac{\left(\sqrt{\kappa(A)} + \left(\sqrt{\kappa(A)}\right)^{-1}\right)^2}{4}, \tag{3.4}$$

where $\kappa(A) := \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ is the spectral condition number *of A.*

Proof. Let $A = Q\Lambda Q^\top$ be the eigen-decomposition of A, with Q orthogonal and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, where $0 < \lambda_1 \leq \cdots \leq \lambda_n$ because A is positive definite. Then defining $\Lambda^{1/2} = \operatorname{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$, we can write

$$\begin{aligned}
\boldsymbol{v}^\top \boldsymbol{v} &= \boldsymbol{v}^\top Q Q^\top \boldsymbol{v} && (Q \text{ is orthogonal}) \\
&= \boldsymbol{v}^\top Q \Lambda^{1/2} \Lambda^{-1/2} Q^\top \boldsymbol{v} \\
&\leq \|\Lambda^{1/2} Q^\top \boldsymbol{v}\|_2 \|\Lambda^{-1/2} Q^\top \boldsymbol{v}\|_2 && (\text{Cauchy–Schwarz}) \\
&= (\boldsymbol{v}^\top Q \Lambda Q^\top \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top Q \Lambda^{-1} Q^\top \boldsymbol{v})^{1/2} \\
&= (\boldsymbol{v}^\top A \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top A^{-1} \boldsymbol{v})^{1/2}.
\end{aligned}$$

Squaring and dividing by $(\boldsymbol{v}^\top \boldsymbol{v})^2$ then yields the first inequality.

For the second inequality, let $c > 0$ be a constant (to be chosen later) and let $\widetilde{A} := cA$. Then

$$\begin{aligned}
(\boldsymbol{v}^\top A \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top A^{-1} \boldsymbol{v})^{1/2} &= (\boldsymbol{v}^\top (cA) \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top (cA)^{-1} \boldsymbol{v})^{1/2} \\
&\leq \frac{1}{2} (\boldsymbol{v}^\top \widetilde{A} \boldsymbol{v} + \boldsymbol{v}^\top \widetilde{A}^{-1} \boldsymbol{v}) \\
&= \frac{1}{2} \boldsymbol{v}^\top Q (\widetilde{\Lambda} + \widetilde{\Lambda}^{-1}) Q^\top \boldsymbol{v},
\end{aligned}$$

where we have used the arithmetic-geometric mean inequality $\sqrt{ab} \leq \frac{1}{2}(a+b)$ for $a, b \geq 0$. If we now define the function $f(\lambda) := \lambda + \lambda^{-1}$, then the inequality above becomes

$$\begin{aligned}
(\boldsymbol{v}^\top A \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top A^{-1} \boldsymbol{v})^{1/2} &\leq \frac{1}{2} (Q^\top \boldsymbol{v})^\top f(\widetilde{\Lambda}) (Q^\top \boldsymbol{v}) \\
&\leq \frac{1}{2} \|f(\widetilde{\Lambda})\|_2 \|Q^\top \boldsymbol{v}\|_2^2 = \frac{1}{2} \max_j |f(\widetilde{\lambda}_j)| \boldsymbol{v}^\top \boldsymbol{v}.
\end{aligned}$$

But $f(\lambda) > 0$ and $f''(\lambda) = 2\lambda^{-3} > 0$ for $\lambda > 0$, so f is a positive convex function over the interval $\left[\widetilde{\lambda}_1, \widetilde{\lambda}_n\right]$ (where $\widetilde{\lambda}_j = c\lambda_j$). Thus, we have

$$\max_j |f(\widetilde{\lambda}_j)| = \max\{f(\widetilde{\lambda}_1), f(\widetilde{\lambda}_n)\},$$

so picking $c = 1/\sqrt{\lambda_1 \lambda_n}$ gives

$$\widetilde{\lambda}_1 = \sqrt{\frac{\lambda_1}{\lambda_n}}, \qquad \widetilde{\lambda}_n = \sqrt{\frac{\lambda_n}{\lambda_1}},$$

which in turn yields

$$\max_j |f(\widetilde{\lambda}_j)| = f(\widetilde{\lambda}_1) = f(\widetilde{\lambda}_n) = \sqrt{\frac{\lambda_1}{\lambda_n}} + \sqrt{\frac{\lambda_n}{\lambda_1}}.$$

Finally, noting that $\kappa(A) = \lambda_n / \lambda_1$, we obtain

$$(\boldsymbol{v}^\top A \boldsymbol{v})^{1/2} (\boldsymbol{v}^\top A^{-1} \boldsymbol{v})^{1/2} \leq \frac{1}{2} \left(\sqrt{\kappa(A)} + \frac{1}{\sqrt{\kappa(A)}} \right) \boldsymbol{v}^\top \boldsymbol{v},$$

which, upon squaring and dividing by $(\boldsymbol{v}^\top \boldsymbol{v})^2$, gives the second inequality. $\square$

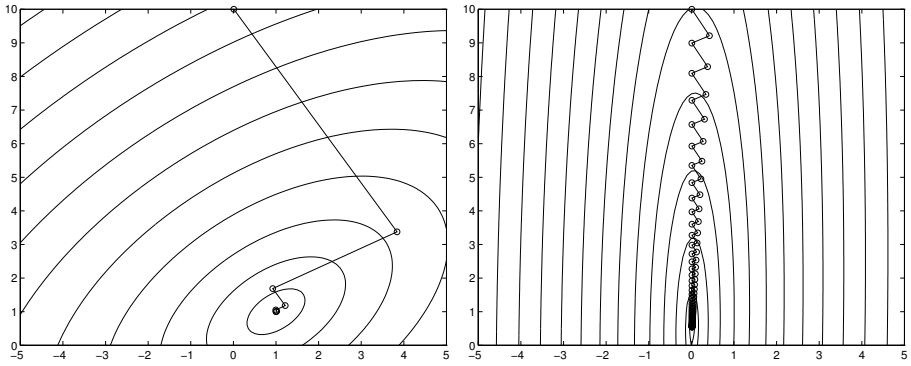


Figure 3.4: Good convergence of steepest descent on the left and a steep valley which slows down the convergence dramatically on the right.

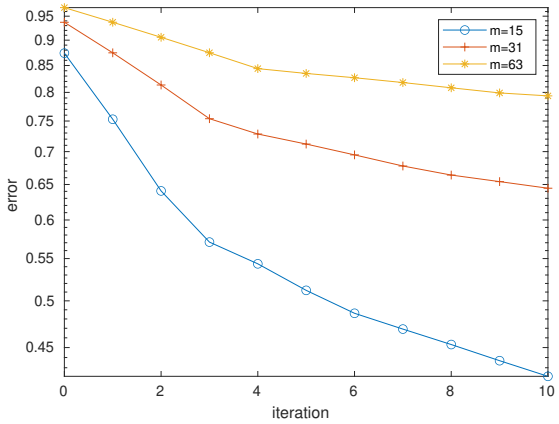


Figure 3.5: Convergence of the steepest descent method for different mesh sizes $h = \frac{1}{m+1}$.

Theorem 19 shows that the method of steepest descent always converges, but the convergence may be very slow. To see why, consider the situation where the function $F(\boldsymbol{u})$ represents a narrow valley, like shown in Figure 3.4 on the right. Steepest descent is in that case zigzagging through the valley and therefore approaching the minimum very slowly.

Let us now study the convergence of steepest descent for decreasing values of the grid size h, as we did for all the methods of Chapter 2. We show in Figure 3.5 how the error decreases as the iterations progress when steepest descent is used to solve our Laplace model problem (see Section 1.3) for different values of interior grid points: $m = 15$, $m = 31$ and $m = 63$. We observe that the convergence strongly depends on the grid size and deteriorates quite fast for decreasing h. This behavior can be also seen by studying the theoretical convergence bound of Theorem 19 as a function of h and expanding it in zero.

This expansion leads to

$$\frac{\kappa(A)-1}{\kappa(A)+1} = 1 - \frac{\pi^2 h^2}{2} + O(h^4),$$

which similar to the results obtained for Jacobi and Gauß-Seidel. Moreover, all the depicted convergence curves decay faster at the first iterations, but then the convergence becomes slower. This corresponds to the highly zigzagging behavior shown in Figure 3.4 (right). Such behavior was already recognized by Stiefel in [117][3].

3.2 The conjugate gradient method

> *Nach dem bekannten Verfahren des stärksten Abstiegs und der Iteration in Gesamtschritten wird in Abschnitt 5 ein 'n-Schritt-Verfahren' angegeben, welches sowohl eine sukzessive Approximation an die Lösung, als auch deren exakte Bestimmung in endlich vielen Schritten liefert.*
>
> Eduard Stiefel, Über einige Methoden der Relaxationsrechnung, 1952.

> *An iterative algorithm is given for solving a system $A\boldsymbol{x} = \boldsymbol{k}$ of n linear equations in n unknowns. The solution is given in n steps.*
>
> Magnus R. Hestenes and Eduard Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, 1952.

The problem of steepest descent is that the same search directions are used over and over again. A better idea is to use each search direction only once and go far enough so that one does never have to go into that direction again. If we do this in a two-dimensional problem, there are only two independent directions, so the algorithm must converge in two steps, and more generally for n-dimensional problems in n steps, see the quotes above. But how can this be achieved? Consider the example given in Figure 3.6. The two independent search directions are given by $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. First we go into the direction $\boldsymbol{p}_0$. How far do we have to go, if we never want to go again into this direction? We have to go until the search direction is orthogonal to the direction where the solution lies, i.e. $\boldsymbol{p}_k$ is orthogonal to the new error vector $\boldsymbol{e}_{k+1}$ at the next step,

$$\boldsymbol{p}_k^\top \boldsymbol{e}_{k+1} = 0.$$

From this equation we can derive the parameter α_k by

$$\begin{aligned} \boldsymbol{p}_k^\top \boldsymbol{e}_{k+1} &= 0 \\ \boldsymbol{p}_k^\top (\boldsymbol{u} - \boldsymbol{u}_{k+1}) &= 0 \\ \boldsymbol{p}_k^\top (\boldsymbol{u} - \boldsymbol{u}_k - \alpha_k \boldsymbol{p}_k) &= 0 \\ \boldsymbol{p}_k^\top \boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k^\top \boldsymbol{p}_k &= 0 \end{aligned}$$

[3] Stiefel writes exctly "Das Auftreten von Käfigen ist eine allgemeine Erscheinung bei Relaxationsverfahren und sehr unerwünscht. Es bewirkt, dass die Relaxation am Anfang flott vorwärtsgeht, aber dann immer weniger ausgiebig wird", see Figure 3.5.

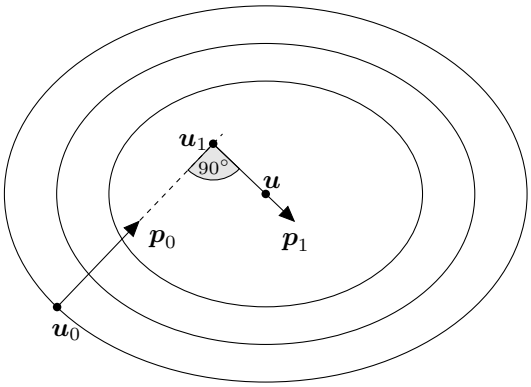


Figure 3.6: Optimal method: $\boldsymbol{u}$ is the minimum of F; the black curves are the level sets of F; the search directions $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are orthogonal and lead to an optimal method that converges in two steps starting from $\boldsymbol{u}_0$ and passing through the first approximation $\boldsymbol{u}_1$.

and thus

$$\alpha_k = \frac{\boldsymbol{p}_k^\top \boldsymbol{e}_k}{\boldsymbol{p}_k^\top \boldsymbol{p}_k}.$$

Hence given n orthogonal search directions $\boldsymbol{p}_k$ this algorithm finds the solution to $A\boldsymbol{u} = \boldsymbol{f}$ in at most n steps for A symmetric positive definite. To see this, consider any initial guess $\boldsymbol{u}_0 \in \mathbb{R}^n$ and the corresponding error $\boldsymbol{e}_0 = \boldsymbol{u} - \boldsymbol{u}_0$. Since the n search directions are linearly independent they form a basis in $\mathbb{R}^n$ and we can expand the error $\boldsymbol{e}_0$ as $\boldsymbol{e}_0 = \sum_{j=0}^{n-1} c_j \boldsymbol{p}_j$ for some coefficients c_j. Now, recalling that $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k$ and working recursively we get

$$\begin{aligned}
\boldsymbol{e}_{k+1} &= \boldsymbol{u} - \boldsymbol{u}_{k+1} = \boldsymbol{u} - \boldsymbol{u}_k - \alpha_k \boldsymbol{p}_k \\
&= \boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k = \boldsymbol{e}_{k-1} - \alpha_{k-1}\boldsymbol{p}_{k-1} - \alpha_k \boldsymbol{p}_k \\
&= \cdots = \boldsymbol{e}_0 - \sum_{j=0}^{k} \alpha_j \boldsymbol{p}_j = \sum_{j=0}^{n-1} c_j \boldsymbol{p}_j - \sum_{j=0}^{k} \alpha_j \boldsymbol{p}_j.
\end{aligned}$$

Using the orthogonality condition $\boldsymbol{p}_k^\top \boldsymbol{e}_{k+1} = 0$ together with the orthogonality of the $\boldsymbol{p}_j$s, the previous equality allows us to compute

$$0 = \boldsymbol{p}_k^\top \boldsymbol{e}_{k+1} = (c_k - \alpha_k)\|\boldsymbol{p}_k\|_2^2,$$

which implies that $\alpha_k = c_k$ for $k = 0, \ldots, n-1$. Therefore, we have that

$$\begin{aligned}
\boldsymbol{e}_n &= \boldsymbol{e}_{n-1} - \alpha_{n-1}\boldsymbol{p}_{n-1} = \boldsymbol{e}_{n-2} - \alpha_{n-2}\boldsymbol{p}_{n-2} - \alpha_{n-1}\boldsymbol{p}_{n-1} \\
&= \cdots = \boldsymbol{e}_0 - \sum_{j=0}^{n-1} \alpha_j \boldsymbol{p}_j = \sum_{j=0}^{n-1} (c_j - \alpha_j)\boldsymbol{p}_j = 0.
\end{aligned}$$

This shows that this algorithm converges theoretically in at most n steps. However there is a big problem in practice: to compute the distance how far we have

to go along a search direction, we need to know already where the solution is, since we have to enforce $\boldsymbol{p}_k^\top \boldsymbol{e}_{k+1} = 0$, and to compute the error vector $\boldsymbol{e}_{k+1}$ we need to know $\boldsymbol{u}$, since $\boldsymbol{e}_{k+1} = \boldsymbol{u} - \boldsymbol{u}_{k+1}$. So in this form the algorithm is not practical.

The key idea now is to note that even without knowing $\boldsymbol{e}_{k+1}$ we know the value of $A\boldsymbol{e}_{k+1}$ because

$$A\boldsymbol{e}_{k+1} = A\boldsymbol{u} - A\boldsymbol{u}_{k+1} = \boldsymbol{f} - A\boldsymbol{u}_{k+1} = \boldsymbol{r}_{k+1},$$

the residual at step $k+1$. This suggests to use a different inner product to enforce orthogonality in. Instead of using the Euclidean inner product of two vectors $\boldsymbol{u}^\top \boldsymbol{v}$ we will use the weighted inner product $\boldsymbol{u}^\top A\boldsymbol{v}$. Suppose we have a set of search directions $\boldsymbol{p}_k$ which is A-orthogonal, namely $\boldsymbol{p}_j^\top A\boldsymbol{p}_k = 0$ for $j \neq k$. Then we perform the same iteration as before,

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k,$$

but instead of choosing α_k so that the new error is orthogonal to the current search direction $\boldsymbol{p}_k$, we move into that direction until the new error is A-orthogonal to the current search direction, namely

$$\boldsymbol{p}_k^\top A\boldsymbol{e}_{k+1} = 0.$$

Note that now we can determine the parameter α_k without knowing the exact solution, by using

$$\begin{aligned} 0 &= \boldsymbol{p}_k^\top A\boldsymbol{e}_{k+1} \\ &= \boldsymbol{p}_k^\top A(\boldsymbol{u} - \boldsymbol{u}_{k+1}) \\ &= \boldsymbol{p}_k^\top A(\boldsymbol{u} - \boldsymbol{u}_k - \alpha_k \boldsymbol{p}_k) \\ &= \boldsymbol{p}_k^\top A\boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k^\top A\boldsymbol{p}_k \\ &= \boldsymbol{p}_k^\top \boldsymbol{r}_k - \alpha_k \boldsymbol{p}_k^\top A\boldsymbol{p}_k. \end{aligned}$$

Hence the optimal choice is

$$\alpha_k = \frac{\boldsymbol{p}_k^\top \boldsymbol{r}_k}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k}, \tag{3.5}$$

which is explicitly known since the residual $\boldsymbol{r}_k$ is given by $\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u}_k$. Thus given a set of n A-orthogonal search directions $\boldsymbol{p}_k$, such an algorithm finds the solution to $A\boldsymbol{u} = \boldsymbol{f}$ in n steps for A symmetric and positive definite.

One could in principle construct an A-orthogonal basis of search directions in advance and make this algorithm practical, but as we will see now, this is not necessary, because it is possible to construct such A-orthogonal search directions on the fly. This leads to the famous *conjugate gradient method (CG)* invented independently by Stiefel [117] and Forsythe, Hestenes and Rosser [41], and led to the famous joint paper of Hestenes and Stiefel on the method [72]. In fact,

Rosser and Stiefel attended the same symposium held at the INA in August 1951, and presented both the same algorithm.

So how is it possible to generate A-orthogonal search directions on the fly? Consider as initial search direction the residual as in steepest descent,

$$\boldsymbol{p}_0 := \boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0. \tag{3.6}$$

Then we obtain for the distance to go along that direction according to (3.5)

$$\alpha_0 := \frac{\boldsymbol{r}_0^\top \boldsymbol{p}_0}{\boldsymbol{p}_0^\top A\boldsymbol{p}_0}, \tag{3.7}$$

and the new approximation is

$$\boldsymbol{u}_1 = \boldsymbol{u}_0 + \alpha_0\boldsymbol{p}_0 = \boldsymbol{u}_0 + \frac{\boldsymbol{r}_0^\top \boldsymbol{p}_0}{\boldsymbol{p}_0^\top A\boldsymbol{p}_0}\boldsymbol{p}_0. \tag{3.8}$$

Now the new residual

$$\boldsymbol{r}_1 = \boldsymbol{f} - A\boldsymbol{u}_1 = \boldsymbol{f} - A\boldsymbol{u}_0 - \alpha_0 A\boldsymbol{p}_0 = \boldsymbol{r}_0 - \alpha_0 A\boldsymbol{p}_0$$

satisfies

$$\boldsymbol{p}_0^\top \boldsymbol{r}_1 = \boldsymbol{p}_0^\top(\boldsymbol{r}_0 - \alpha_0 A\boldsymbol{p}_0) = \boldsymbol{p}_0^\top \boldsymbol{r}_0 - \frac{\boldsymbol{r}_0^\top \boldsymbol{p}_0}{\boldsymbol{p}_0^\top A\boldsymbol{p}_0}\boldsymbol{p}_0^\top A\boldsymbol{p}_0 = 0, \tag{3.9}$$

which means it is orthogonal to the first search direction $\boldsymbol{p}_0$ and thus a good candidate for the new search direction $\boldsymbol{p}_1$ if we want to avoid searching into the same direction again. We just need to A-orthogonalize it with respect to the previous search direction $\boldsymbol{p}_0$ to obtain a method of conjugate search directions. This is achieved by choosing a scalar β_0 such that $\boldsymbol{p}_1 = \boldsymbol{r}_1 + \beta_0\boldsymbol{p}_0$ is A-orthogonal to $\boldsymbol{p}_0$, i.e.

$$0 = \boldsymbol{p}_0^\top A\boldsymbol{p}_1 = \boldsymbol{p}_0^\top A(\boldsymbol{r}_1 + \beta_0\boldsymbol{p}_0) = \boldsymbol{p}_0^\top A\boldsymbol{r}_1 + \beta_0\boldsymbol{p}_0^\top A\boldsymbol{p}_0, \tag{3.10}$$

which implies

$$\beta_0 = -\frac{\boldsymbol{p}_0^\top A\boldsymbol{r}_1}{\boldsymbol{p}_0^\top A\boldsymbol{p}_0}. \tag{3.11}$$

The amazing discovery of Stiefel and Hestenes is that this step is also sufficient for the following iterations:

Theorem 21 (The conjugate gradient method, Hestenes and Stiefel 1952)**.** *For a given linear system $A\boldsymbol{u} = \boldsymbol{f}$ with symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$, and initial approximation $\boldsymbol{u}_0 \in \mathbb{R}^n$, we set $\boldsymbol{p}_0 := \boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$ and compute for $k = 0, 1, 2, \ldots$ the approximate solutions*

$$\boldsymbol{u}_{k+1} := \boldsymbol{u}_k + \alpha_k\boldsymbol{p}_k, \quad \alpha_k = \frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k}, \tag{3.12}$$

and search directions

$$\boldsymbol{p}_{k+1} := \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k, \quad \beta_k = -\frac{\boldsymbol{p}_k^\top A \boldsymbol{r}_{k+1}}{\boldsymbol{p}_k^\top A \boldsymbol{p}_k}. \tag{3.13}$$

Then the residuals $\boldsymbol{r}_j := \boldsymbol{f} - A\boldsymbol{u}_j$ *are orthogonal*

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_j = 0 \quad j = 0, 1, \ldots, k, \tag{3.14}$$

the residuals are orthogonal to the search directions,

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_j = 0 \quad j = 0, 1, \ldots, k, \tag{3.15}$$

and the search directions are A-orthogonal,

$$\boldsymbol{p}_{k+1}^\top A \boldsymbol{p}_j = 0 \quad j = 0, 1, \ldots, k. \tag{3.16}$$

Proof. The proof is by induction, and we have already seen that the result holds for $k = 0$, see (3.9), and (3.10) with (3.11). So we assume that the result holds for k, and show that it then still must hold for $k + 1$. For the orthogonality of the residuals, we compute

$$\begin{aligned} \boldsymbol{r}_{k+1}^\top \boldsymbol{r}_j &= \boldsymbol{r}_j^\top \boldsymbol{r}_{k+1} \\ &= \boldsymbol{r}_j^\top (\boldsymbol{f} - A\boldsymbol{u}_{k+1}) \\ &= \boldsymbol{r}_j^\top \big(\boldsymbol{f} - A(\boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k)\big) \\ &= \boldsymbol{r}_j^\top \boldsymbol{r}_k - \alpha_k \boldsymbol{r}_j^\top A \boldsymbol{p}_k, \end{aligned}$$

and using from the first equation in (3.13) that $\boldsymbol{r}_j = \boldsymbol{p}_j - \beta_{j-1}\boldsymbol{p}_{j-1}$ to replace the second occurrence of $\boldsymbol{r}_j$ on the right leads to

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_j = \boldsymbol{r}_j^\top \boldsymbol{r}_k - \alpha_k (\boldsymbol{p}_j^\top A \boldsymbol{p}_k - \beta_{j-1} \boldsymbol{p}_{j-1}^\top A \boldsymbol{p}_k) = \boldsymbol{r}_j^\top \boldsymbol{r}_k - \alpha_k \boldsymbol{p}_j^\top A \boldsymbol{p}_k, \tag{3.17}$$

since $\boldsymbol{p}_{j-1}^\top A \boldsymbol{p}_k = 0$ for $j \leq k$ by induction hypothesis. Now we distinguish two cases, $j = k$ and $j < k$: if $j < k$, then also $\boldsymbol{r}_j^\top \boldsymbol{r}_k = 0$ and $\boldsymbol{p}_j^\top A \boldsymbol{p}_k = 0$ by induction hypothesis, and thus $\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_j = 0$ for $j < k$. Now for $j = k$, we obtain from (3.17) using the definition of α_k in (3.12) that

$$\begin{aligned} \boldsymbol{r}_{k+1}^\top \boldsymbol{r}_k &= \boldsymbol{r}_k^\top \boldsymbol{r}_k - \frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top A \boldsymbol{p}_k} \boldsymbol{p}_k^\top A \boldsymbol{p}_k \\ &= \boldsymbol{r}_k^\top \boldsymbol{r}_k - \boldsymbol{r}_k^\top \boldsymbol{p}_k \\ &= \boldsymbol{r}_k^\top \boldsymbol{r}_k - \boldsymbol{r}_k^\top (\boldsymbol{r}_k + \beta_{k-1} \boldsymbol{p}_{k-1}) \\ &= -\beta_{k-1} \boldsymbol{r}_k^\top \boldsymbol{p}_{k-1}, \end{aligned}$$

where we used $\boldsymbol{p}_k = \boldsymbol{r}_k + \beta_{k-1}\boldsymbol{p}_{k-1}$ from the first equation in (3.13). By induction hypothesis we have that $\boldsymbol{r}_k^\top \boldsymbol{p}_{k-1} = 0$, which concludes the proof of orthogonality of the residuals in (3.14).

Next, we prove the orthogonality relation (3.15). To do so, we consider the product $\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_j$ and notice that

$$\boldsymbol{r}_{k+1} = \boldsymbol{f} - A\boldsymbol{u}_{k+1} = \boldsymbol{f} - A(\boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k) = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k.$$

Hence, we compute

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_j = \boldsymbol{r}_k^\top \boldsymbol{p}_j - \alpha_k \boldsymbol{p}_k^\top A\boldsymbol{p}_j,$$

where we used that $A = A^\top$. Now we again distinguish two cases, $j < k$ and $j = k$: if $j < k$, then $\boldsymbol{r}_k^\top \boldsymbol{p}_j = 0$ and $\boldsymbol{p}_k^\top A\boldsymbol{p}_j = 0$ by induction hypothesis. If $j = k$, then using (3.12) we have

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_k = \boldsymbol{r}_k^\top \boldsymbol{p}_k - \alpha_k \boldsymbol{p}_k^\top A\boldsymbol{p}_k = \boldsymbol{r}_k^\top \boldsymbol{p}_k - \frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k} \boldsymbol{p}_k^\top A\boldsymbol{p}_k = 0,$$

which concludes the proof of (3.15).

To prove the A-orthogonality of the search directions, $\boldsymbol{p}_{k+1}^\top A\boldsymbol{p}_j = 0$ for $j = 0, 1, \ldots, k$, we first note that by construction this holds already for $j = k$ by the definition of β_k. For $j < k$, we first compute

$$\boldsymbol{r}_{j+1} = \boldsymbol{f} - A\boldsymbol{u}_{j+1} = \boldsymbol{f} - A(\boldsymbol{u}_j + \alpha_j \boldsymbol{p}_j) = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$$

from which we obtain that

$$A\boldsymbol{p}_j = \frac{1}{\alpha_j}(\boldsymbol{r}_j - \boldsymbol{r}_{j+1}). \tag{3.18}$$

Now using again $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$ from (3.13), and then inserting (3.18), we get

$$\boldsymbol{p}_{k+1}^\top A\boldsymbol{p}_j = (\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k)^\top A\boldsymbol{p}_j = \frac{1}{\alpha_j} \boldsymbol{r}_{k+1}^\top (\boldsymbol{r}_j - \boldsymbol{r}_{j+1}) + \beta_k \boldsymbol{p}_k^\top A\boldsymbol{p}_j = 0,$$

since we proved already that $\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_j = 0$ and $\boldsymbol{r}_{k+1}^\top \boldsymbol{r}_{j+1} = 0$ for $j < k$, and by induction hypothesis also $\boldsymbol{p}_k^\top A\boldsymbol{p}_j = 0$ for $j < k$. This concludes the proof. □

It appears at first sight that one iteration of the conjugate gradient algorithm in Theorem 21 requires three matrix-vector products: $A\boldsymbol{p}_k$ in (3.12) to compute $\alpha_k = \frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k}$, $A\boldsymbol{r}_{k+1}$ in (3.13) to compute $\beta_k = -\frac{\boldsymbol{p}_k^\top A\boldsymbol{r}_{k+1}}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k}$, and then the residual $\boldsymbol{r}_{k+1} = \boldsymbol{f} - A\boldsymbol{u}_{k+1}$ also needs to be computed. This can however be reduced to the single matrix-vector product $A\boldsymbol{p}_k$ as follows: for the residual, we can compute

$$\boldsymbol{r}_{k+1} = \boldsymbol{f} - A\boldsymbol{u}_{k+1} = \boldsymbol{f} - A(\boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k) = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k \tag{3.19}$$

and thus only $A\boldsymbol{p}_k$ is needed. To compute β_k, we first compute

$$\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_{k+1} = (\boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k)^\top \boldsymbol{p}_{k+1} = \boldsymbol{r}_k^\top \boldsymbol{p}_{k+1} = \boldsymbol{r}_k^\top (\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k) = \beta_k \boldsymbol{r}_k^\top \boldsymbol{p}_k,$$

where we used the A-orthogonality of the search directions $\boldsymbol{p}_k$ and orthogonality of the residuals $\boldsymbol{r}_k$. Solving for β_k, and using again that the residuals are orthogonal to the search directions, we get

$$\beta_k = \frac{\boldsymbol{r}_{k+1}^\top \boldsymbol{p}_{k+1}}{\boldsymbol{r}_k^\top \boldsymbol{p}_k} = \frac{\boldsymbol{r}_{k+1}^\top (\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k)}{\boldsymbol{r}_k^\top (\boldsymbol{r}_k + \beta_{k-1} \boldsymbol{p}_{k-1})} = \frac{\|\boldsymbol{r}_{k+1}\|_2^2}{\|\boldsymbol{r}_k\|_2^2}.$$

Because of $\boldsymbol{r}_k^\top \boldsymbol{p}_k = \|\boldsymbol{r}_k\|_2^2$, the computation of α_k simplifies also to

$$\alpha_k = \frac{\|\boldsymbol{r}_k\|_2^2}{\boldsymbol{p}_k^\top A \boldsymbol{p}_k}. \tag{3.20}$$

With these observations, we obtain a version of CG which uses only one matrix-vector multiplication per iteration step:

```
function [u,uk,res]=CG(A,f,u0,tol,m)
% CG conjugate gradient method
%   [u,uk,res]=CG(A,f,u0,tol,m) solves Au=f using the conjugate gradient
%   method starting at the initial guess u0 up to a tolerance tol using
%   at most m iterations. The matrix A has to be symmetric positive
%   definite. CG returns in the matrix uk the iterates, in u the solution
%   computed, and in res the history of the norm of the residuals.

if nargin<5, m=100; end             % default values
if nargin<4, tol=1e-6; end
uk(:,1)=u0;
r=f-A*uk(:,1);
p=r;
oldrho=r'*r;
res(1)=sqrt(oldrho);
k=0;
while sqrt(oldrho)/norm(f)>tol && k<=m
  k=k+1;
  Ap=A*p;
  alpha=oldrho/(p'*Ap);
  uk(:,k+1)=uk(:,k)+alpha*p;
  r=r-alpha*Ap;
  rho=r'*r;
  beta=rho/oldrho; oldrho=rho;
  res(k+1)=sqrt(oldrho);
  p=r+beta*p;
end
u=uk(:,k+1);
```

Applying the conjugate gradient method to our Laplace model problem from Section 1.3, we obtain for the first iterates the approximations shown in Figure 3.7. We see that the method converges rather quickly, comparable to SOR in Figure 2.8 and the true Richardson's method in Figure 2.18. We have seen by construction that CG converges in exact arithmetic after at most n iterations for

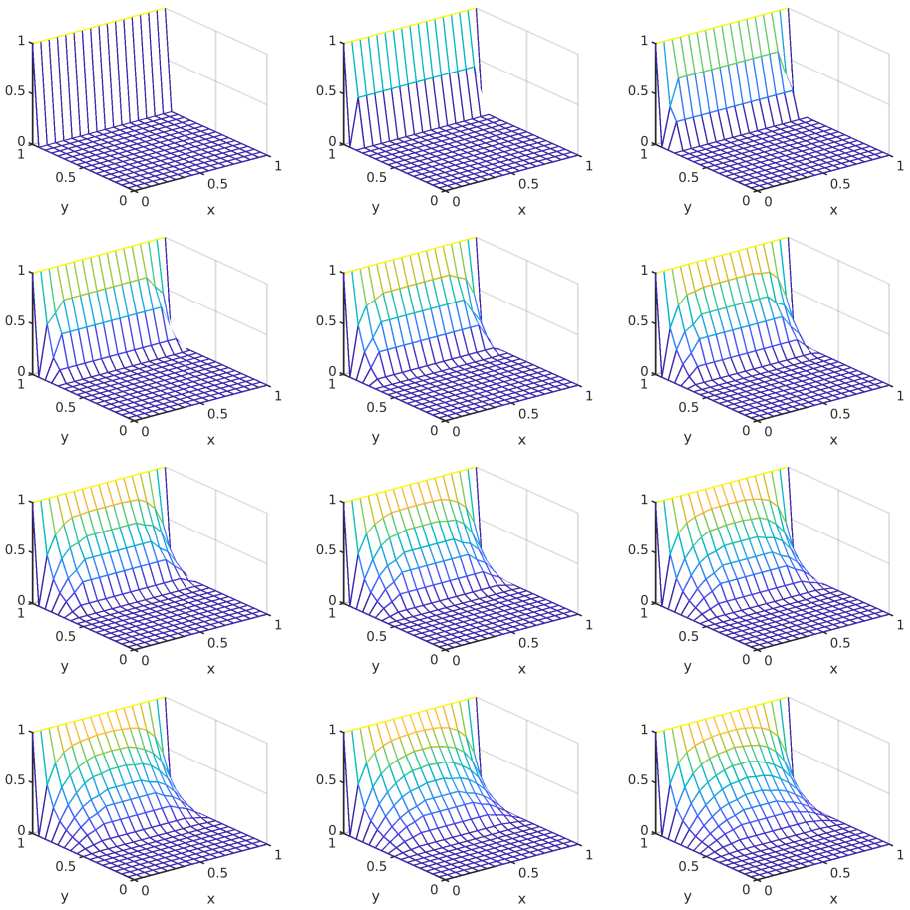


Figure 3.7: Initial guess and first iterations of the conjugate gradient method applied to our Laplace model problem from Section 1.3.

a problem with n unknowns[4], and this was emphasized in the early days as one of the main features of CG. Figure 3.7 indicates however that CG has an even more important convergence property: one often needs much fewer iterations to obtain already very good approximations to the solution. To see this, we now introduce the Krylov space that appears in the work of Krylov from 1931 when he studied the solution of second order linear systems of ordinary differential equation [82]. Krylov spaces play a major role in the conjugate gradient method and its generalizations we will see later.

Definition 10 (Krylov space)**.** *For given* $A \in \mathbb{R}^{n\times n}$ *and* $\boldsymbol{v} \in \mathbb{R}^n$ *the* Krylov

[4]Convergence in floating point arithmetic is more delicate, but well understood since the seminal work of Chris Paige [96], see also [85].

space *(of order k) is defined as*

$$\mathcal{K}_k(A, \boldsymbol{v}) := \text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \dots, A^{k-1}\boldsymbol{v}\}. \tag{3.21}$$

For our purposes, we consider the Krylov space associated with a linear system of equations $A\boldsymbol{u} = \boldsymbol{f}$ with $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^n$ and initial guess $\boldsymbol{u}_0 \in \mathbb{R}^n$, that is

$$\mathcal{K}_k(A, \boldsymbol{r}_0) = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \dots, A^{k-1}\boldsymbol{r}_0\},$$

where $\boldsymbol{r}_0 := \boldsymbol{f} - A\boldsymbol{u}_0$ denotes the residual of the initial guess.

Before we relate the Krylov space $\mathcal{K}_k(A, \boldsymbol{r}_0)$ with the residuals $\boldsymbol{r}_k$ and the search directions $\boldsymbol{p}_k$ of the CG method, we study some properties of $\mathcal{K}_k(A, \boldsymbol{r}_0)$. To do so, we recall the definition of the minimal polynomial of a given matrix A; see, e.g., [108, Section 6.1].

Definition 11 (Minimal polynomial)**.** *The minimal polynomial of a matrix $A\in \mathbb{R}^{n\times n}$ is the nonzero monic polynomial $p_{\min}$ of lowest degree such that $p_{\min}(A) = 0$. Moreover, we define the minimal polynomial of a pair $(A, \boldsymbol{v})\in \mathbb{R}^{n\times n} \times \mathbb{R}^n$ as the nonzero monic polynomial $p^{\boldsymbol{v}}_{\min}$ of lowest degree such that $p^{\boldsymbol{v}}_{\min}(A)\boldsymbol{v} = 0$, and it is clear that $\deg(p^{\boldsymbol{v}}_{\min}) \le \deg(p_{\min})$.*

We recall a few important properties of the minimal polynomial in

Lemma 6 (Properties of the minimal polynomial)**.** *Let $p_{\min}$ be the minimal polynomial of a matrix A of size $n \times n$. Then:*

(a) *$p_{\min}$ is unique.*

(b) *If A is diagonalizable and has $m \le n$ distinct eigenvalues, then $p_{\min}$ has degree m.*

(c) *If A is invertible, then the constant coefficient of $p_{\min}$ is necessarily nonzero.*

Proof. To prove (a), take any polynomial p such that $p(A) = 0$. According to the standard Euclidean division of polynomials, if we divide p by $p_{\min}$ we get $p = qp_{\min} + r$, where q and r are the quotient and the remainder with $\deg(r) < \deg(p_{\min})$. Hence, either $r = 0$ or $r \neq 0$ with degree smaller than $p_{\min}$. Now, we have $0 = p(A) = q(A)p_{\min}(A) + r(A) = r(A)$. Therefore p and r represent two possible candidates to be (or to construct by dividing them by their highest-degree coefficient) a minimal polynomial. If $r = 0$, then $p(x) = q(x)p_{\min}(x)$ and hence p has degree higher or equal to the degree of $p_{\min}$, and if p and $p_{\min}$ have the same degree, then $q(x) = q$ is constant and $p(x)/q = p_{\min}(x)$. Now if $r \neq 0$, then $\deg(r) < \deg(p)$. So it is sufficient to study r. Denote by γ the highest-degree coefficient of r and notice that $\widetilde{r} := \frac{r}{\gamma}$ is a monic polynomial of the same degree of r and $\widetilde{r}(A) = 0$. Therefore, $\widetilde{r}$ is a possible candidate to be a minimal polynomial, but since $\deg(\widetilde{r}) < \deg(p_{\min})$, we get a contradiction to the fact that $p_{\min}$ has minimal degree among the monic polynomials that are zero in A.

Now, we prove (b). To do so, we recall that the minimal polynomial of a diagonalizable matrix is the product of distinct monic factors [73, Corollary 3.3.8]. Hence, since A has m distinct eigenvalues its minimal polynomial has degree m.

Finally, we prove (c). Assume that A is invertible, $p_{\min}$ has degree n, and denote its coefficients by γ_j for $j = 0, \dots, n$. Seeking a contradiction, assume that $\gamma_0 = 0$. Therefore, we have that $p_{\min}(A) = A^n + \gamma_{n-1}A^{n-1} + \cdots + \gamma_1 A$. Since A is invertible, we can define the monic polynomial $\widetilde{p}(A) := A^{-1}p_{\min}(A)$. Notice that $\widetilde{p}(A) = A^{n-1} + \gamma_{n-1}A^{n-2} + \cdots + \gamma_1$ and $\widetilde{p}(A) = 0$, which contradicts the fact that $p_{\min}$ has minimal degree. □

The minimal polynomial of A allows us to characterize the Krylov space; see, e.g., [108].

Theorem 22 (Invariance of a Krylov space)**.** *Consider a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{v} \in \mathbb{R}^n$. Let $p^{\boldsymbol{v}}_{\min}$ be the minimal polynomial of $(A, \boldsymbol{v})$ and assume that $p^{\boldsymbol{v}}_{\min}$ has degree k. Then the Krylov space $\mathcal{K}_k(A, \boldsymbol{v})$ is invariant under A, that is for any $\boldsymbol{w} \in \mathcal{K}_k(A, \boldsymbol{v})$ it holds that $A\boldsymbol{w} \in \mathcal{K}_k(A, \boldsymbol{v})$. Similarly, if the minimal polynomial $p_{\min}$ of A has degree k, then $\mathcal{K}_k(A, \boldsymbol{v})$ is invariant under A for any vector $\boldsymbol{v}$.*

Proof. Consider any vector $\boldsymbol{w} \in \mathcal{K}_k(A, \boldsymbol{v})$. Then $\boldsymbol{w} = \sum_{j=0}^{k-1} \gamma_j A^j \boldsymbol{v}$ for some coefficients γ_j. Now, assume that $\gamma_{k-1} = 0$, then $A\boldsymbol{w} \in \mathcal{K}_k(A, \boldsymbol{v})$. In the case $\gamma_{k-1} \neq 0$, we have to show that $A(A^{k-1}\boldsymbol{v}) = A^k\boldsymbol{v} \in \mathcal{K}_k(A, \boldsymbol{v})$ as well. To do so, consider the minimal polynomial $p^{\boldsymbol{v}}_{\min}(x) = x^k + c_{k-1}x^{k-1} + \cdots + c_0$. Since $p^{\boldsymbol{v}}_{\min}(A)\boldsymbol{v} = 0$, we get

$$0 = p^{\boldsymbol{v}}_{\min}(A)\boldsymbol{v} = A^k\boldsymbol{v} + c_{k-1}A^{k-1}\boldsymbol{v} + \cdots + c_1 A\boldsymbol{v} + c_0\boldsymbol{v},$$

which implies that

$$A^k\boldsymbol{v} = -[c_{k-1}A^{k-1}\boldsymbol{v} + \cdots + c_1 A\boldsymbol{v} + c_0\boldsymbol{v}] \in \mathcal{K}_k(A, \boldsymbol{v}).$$

Hence, it holds that $A\boldsymbol{w} \in \mathcal{K}_k(A, \boldsymbol{v})$. The second statement follows from similar arguments and our proof is complete. □

Theorem 23 (Dimension of a Krylov space)**.** *Consider a nonsingular matrix $A \in \mathbb{R}^{n\times n}$ and a nonzero vector $\boldsymbol{v} \in \mathbb{R}^n$. The Krylov space $\mathcal{K}_k(A, \boldsymbol{v})$ is of dimension k if and only if the degree of the minimal polynomial $p^{\boldsymbol{v}}_{\min}$ of the pair $(A, \boldsymbol{v})$ is larger than $k-1$, that is* $\deg(p^{\boldsymbol{v}}_{\min}) \geq k$.

Proof. Since A is invertible, its kernel is $\ker(A) = \{0\}$, and recalling that $\boldsymbol{v}$ is nonzero, it holds that $A^j\boldsymbol{v}$ is nonzero for any positive integer $j < \infty$. Now, the nonzero vectors $\boldsymbol{v}, A\boldsymbol{v}, \dots, A^{k-1}\boldsymbol{v}$ form a basis of $\mathcal{K}_k(A, \boldsymbol{v})$ if and only if for any complex k-tuple γ_j for $j = 0, \dots, k-1$, where at least one γ_j is nonzero, the linear combination $\sum_{j=0}^{k-1} \gamma_j A^j\boldsymbol{v}$ is nonzero. This condition is equivalent to the condition that there is no nonzero polynomial p of degree lower than or equal to $k-1$ such that $p(A)\boldsymbol{v} = 0$. □

It is clear from Theorem 23 that the vector $\boldsymbol{v}$ influences the dimension of $\mathcal{K}_k(A, \boldsymbol{v})$; see also the following example.

Example 9. *Consider the matrix*

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

and the three canonical vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ *and* $\boldsymbol{e}_3$. *The corresponding Krylov spaces are*

$$\mathcal{K}_3(A, \boldsymbol{e}_1) = \operatorname{span}\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ -4 \\ 1 \end{bmatrix} \right\} \quad \textit{with } \dim \mathcal{K}_3(A, \boldsymbol{e}_1) = 3,$$

$$\mathcal{K}_3(A, \boldsymbol{e}_2) = \operatorname{span}\left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} -4 \\ 6 \\ -4 \end{bmatrix} \right\} \quad \textit{with } \dim \mathcal{K}_3(A, \boldsymbol{e}_2) = 2,$$

$$\mathcal{K}_3(A, \boldsymbol{e}_3) = \operatorname{span}\left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -4 \\ 5 \end{bmatrix} \right\} \quad \textit{with } \dim \mathcal{K}_3(A, \boldsymbol{e}_3) = 3.$$

It is clear that the choice of the vector $\boldsymbol{v}$ *influences the dimension of* $\mathcal{K}_3(A, \boldsymbol{v})$. *Notice also that the minimal polynomial of* A *is* $p_{\min}(x) = x^3 - 6x^2 + 10x - 4$, *while the minimal polynomial of the pair* $(A, \boldsymbol{e}_2)$ *is* $p^{\boldsymbol{e}_2}_{\min}(x) = x^2 - 4x + 2$.

An interesting property of the Krylov space $\mathcal{K}_k(A, \boldsymbol{f})$ associated to the system $A\boldsymbol{u} = \boldsymbol{f}$ is given by the following theorem.

Theorem 24 (Krylov space and solution of $A\boldsymbol{u} = \boldsymbol{f}$)**.** *If the minimal polynomial* $p_{\min}$ *of the invertible matrix* A *has degree* m, *then the solution to* $A\boldsymbol{u} = \boldsymbol{f}$ *belongs to the space* $\mathcal{K}_m(A, \boldsymbol{f})$. *Moreover, let* $\boldsymbol{u}_0 \in \mathbb{R}^n$ *be a given vector and* $\boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$. *Then* $\boldsymbol{u}$ *belongs to the* affine Krylov space $\boldsymbol{u}_0 + \mathcal{K}_m(A, \boldsymbol{r}_0)$.

Proof. Since the minimal polynomial has degree m, it holds that

$$0 = p_{\min}(A) = \gamma_0 \Big(I + \sum_{j=1}^{m} \frac{\gamma_j}{\gamma_0} A^j \Big),$$

where $\gamma_0 \neq 0$ because A is invertible (Lemma 6). This implies that

$$A^{-1} = -\sum_{j=0}^{m-1} \frac{\gamma_{j+1}}{\gamma_0} A^j,$$

and we conclude by noticing that

$$\boldsymbol{u} = A^{-1}\boldsymbol{f} = -\sum_{j=0}^{m-1} \frac{\gamma_{j+1}}{\gamma_0} A^j \boldsymbol{f}$$

and

$$\boldsymbol{u} = A^{-1}\boldsymbol{f} = A^{-1}(\boldsymbol{f} - A\boldsymbol{u}_0) + \boldsymbol{u}_0 = \boldsymbol{u}_0 + A^{-1}\boldsymbol{r}_0 = \boldsymbol{u}_0 - \sum_{j=0}^{m-1} \frac{\gamma_{j+1}}{\gamma_0} A^j \boldsymbol{r}_0.$$

□

Notice also that, in general, the Krylov space $\mathcal{K}_m(A, \boldsymbol{f})$ and its affine counterpart $\boldsymbol{u}_0 + \mathcal{K}_m(A, \boldsymbol{r}_0)$ do not coincide. Consider as an example the first two canonical vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ of $\mathbb{R}^3$, the matrix A of Example 9, the vector $\boldsymbol{f} = \boldsymbol{e}_1$, the vector $\boldsymbol{u}_0 = \frac{1}{8}[5\ -2\ 1]^\top$ (which gives $\boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0 = \boldsymbol{e}_2$), and the spaces $\mathcal{K}_3(A, \boldsymbol{e}_1)$ and $\boldsymbol{u}_0 + \mathcal{K}_3(A, \boldsymbol{e}_2)$. A simple argument allows one to see that $\frac{1}{8}\boldsymbol{e}_1 \in \mathcal{K}_3(A, \boldsymbol{e}_1)$ but $\frac{1}{8}\boldsymbol{e}_1 \notin \boldsymbol{u}_0 + \mathcal{K}_3(A, \boldsymbol{e}_2)$.

Theorem 24 anticipates an important feature of Krylov methods that we will see in this chapter: if $\boldsymbol{u}_0$ is the initialization vector, Krylov methods construct the solution $\boldsymbol{u}$ to $A\boldsymbol{u} = \boldsymbol{f}$ exploiting the structure of affine Krylov spaces $\boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$, where k is the iteration count. Indeed, different Krylov methods exploit these spaces in different manners as we will see in Section 3.6.

Let us now continue with the analysis of the conjugate gradient method by relating residuals and search directions constructed by CG to the Krylov space.

Lemma 7 (Residuals, search directions and Krylov space - 1)**.** *The residual vectors $\boldsymbol{r}_k$ and the search directions $\boldsymbol{p}_k$ generated by the conjugate gradient algorithm are contained in the Krylov space,*

$$\boldsymbol{r}_k, \boldsymbol{p}_k \in \mathcal{K}_{k+1}(A, \boldsymbol{r}_0). \tag{3.22}$$

Proof. The proof is by induction. The result clearly holds initially for $k = 0$, because by definition $\boldsymbol{r}_0 = \boldsymbol{p}_0 \in \mathcal{K}_1(A, \boldsymbol{r}_0) = \text{span}\{\boldsymbol{r}_0\}$. So we assume that (3.22) holds for k, and we show that then the same also holds for $k+1$. Recalling (3.19), that is

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k,$$

and using the induction hypothesis $\boldsymbol{r}_k \in \mathcal{K}_{k+1}(A, \boldsymbol{r}_0) \subseteq \mathcal{K}_{k+2}(A, \boldsymbol{r}_0)$, and $\boldsymbol{p}_k \in \mathcal{K}_{k+1}(A, \boldsymbol{r}_0)$, which implies that $A\boldsymbol{p}_k \in \mathcal{K}_{k+2}(A, \boldsymbol{r}_0)$, we obtain the first claim.

Now for the search directions, using (3.13) and the induction hypothesis on $\boldsymbol{p}_k$, we obtain that

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \in \mathcal{K}_{k+2}(A, \boldsymbol{r}_0),$$

because we have already shown that $\boldsymbol{r}_{k+1} \in \mathcal{K}_{k+2}(A, \boldsymbol{r}_0)$. This concludes the proof. □

Note that the previous result implies that $\boldsymbol{r}_{j-1} \in \mathcal{K}_j(A, \boldsymbol{r}_0) \subseteq \mathcal{K}_k(A, \boldsymbol{r}_0)$ for $j = 1, \ldots, k$. Therefore, we have that

$$\text{span}\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1}\} \subset \mathcal{K}_k(A, \boldsymbol{r}_0) \tag{3.23}$$

and

$$\text{span}\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{k-1}\} \subset \mathcal{K}_k(A, \boldsymbol{r}_0). \tag{3.24}$$

However, equality holds in both cases. To see this, it is sufficient to notice that, according to Theorem 21, residuals and search directions generated by the conjugate gradient algorithm are linearly independent, hence

$$\dim(\text{span}\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1}\}) = k \text{ and } \dim(\text{span}\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{k-1}\}) = k.$$

Hence equality in (3.23) and (3.24) follows by noticing that $\dim(\mathcal{K}_k(A, \boldsymbol{r}_0)) \leq k$ (see also Theorem 23). We prove this property, independently of Theorem 21, in the next lemma; see, e.g. [93].

Lemma 8 (Residuals, search directions and Krylov space - 2)**.** *Consider the vectors $\boldsymbol{r}_j$ and $\boldsymbol{p}_j$ for $j = 0, \ldots, k-1$ generated by the conjugate gradient iterations till the kth iterate. We have*

$$\text{span}\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1}\} = \mathcal{K}_k(A, \boldsymbol{r}_0) \tag{3.25}$$

and

$$\text{span}\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{k-1}\} = \mathcal{K}_k(A, \boldsymbol{r}_0). \tag{3.26}$$

Proof. We proceed by induction. The equalities (3.25) and (3.26) are trivially satisfied for $k = 1$. We assume by induction that they are satisfied for some k and we prove that they hold for $k+1$. By Lemma 7 we have that $\boldsymbol{r}_k, \boldsymbol{p}_k \in \mathcal{K}_{k+1}(A, \boldsymbol{r}_0)$ for any k, and since $\boldsymbol{r}_{j-1}, \boldsymbol{p}_{j-1} \in \mathcal{K}_j(A, \boldsymbol{r}_0) \subset \mathcal{K}_{k+1}(A, \boldsymbol{r}_0)$ for $j = 1, \ldots, k+1$ we obtain that (3.23) and (3.24) hold for $k+1$. Hence we need to show that the reverse inclusions hold as well. To do so, we first notice that by the induction hypothesis (3.26) we have

$$A^k \boldsymbol{r}_0 = A(A^{k-1}\boldsymbol{r}_0) \in \text{span}\{A\boldsymbol{p}_0, A\boldsymbol{p}_1, \ldots, A\boldsymbol{p}_{k-1}\}. \tag{3.27}$$

Now, since by (3.18) it holds that $A\boldsymbol{p}_j = \frac{1}{\alpha_j}(\boldsymbol{r}_j - \boldsymbol{r}_{j+1})$ for $j = 0, \ldots, k-1$, we obtain from (3.27) that

$$A^k \boldsymbol{r}_0 \in \text{span}\{\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k-1}, \boldsymbol{r}_k\}. \tag{3.28}$$

By combining (3.28) with the induction hypothesis (3.25), we get

$$\mathcal{K}_{k+1}(A, \boldsymbol{r}_0) = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0, A^k\boldsymbol{r}_0\} \subset \text{span}\{\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k-1}, \boldsymbol{r}_k\}.$$

Therefore, (3.25) holds also for $k+1$. Next, we show that (3.26) continues to hold when k is replaced by $k+1$. Indeed,

$$\begin{aligned}
&\text{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k-1}, \boldsymbol{p}_k\} \\
&\quad = \text{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k-1}, \boldsymbol{r}_k\} && \text{(by (3.13))} \\
&\quad = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0, \boldsymbol{r}_k\} && \text{(by the induction hypothesis (3.26))} \\
&\quad = \text{span}\{\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k-1}, \boldsymbol{r}_k\} && \text{(by (3.25) for } k\text{)} \\
&\quad = \text{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0, A^k\boldsymbol{r}_0\} && \text{(by (3.25) for } k+1\text{)},
\end{aligned}$$

which is (3.26) for $k+1$. □

Remark 1. *Note that recalling $\boldsymbol{r}_k \perp \boldsymbol{r}_j$ for $j = 0, 1, \ldots, k-1$, and (3.25), that is $\mathcal{K}_k(A, \boldsymbol{r}_0) = \operatorname{span}\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1}\}$, we have that*

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A, \boldsymbol{r}_0).$$

We next show that the conjugate gradient algorithm finds at iteration k the best approximation of the unknown solution $\boldsymbol{u}$ of the linear system $A\boldsymbol{u} = \boldsymbol{f}$ in the *affine Krylov space* $\boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$ (recall also Theorem 24) in the energy norm $\|\boldsymbol{v}\|_A := \sqrt{\boldsymbol{v}^\top A \boldsymbol{v}}$, also called the A-norm:

Lemma 9 (Best approximation property of CG)**.** *Let $\boldsymbol{u}$ be the solution of $A\boldsymbol{u} = \boldsymbol{f}$, with $A \in \mathbb{R}^{n\times n}$ symmetric and positive definite. Then, at iteration k, the approximation $\boldsymbol{u}_k$ computed by CG minimizes the A-norm of the error,*

$$\boldsymbol{u}_k = \underset{\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)}{\arg\min} \|\boldsymbol{u} - \widetilde{\boldsymbol{u}}\|_A. \tag{3.29}$$

Proof. We first show that $\boldsymbol{u}_k \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$:

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \alpha_{k-1}\boldsymbol{p}_{k-1} = \boldsymbol{u}_{k-2} + \alpha_{k-2}\boldsymbol{p}_{k-2} + \alpha_{k-1}\boldsymbol{p}_{k-1} = \cdots = \boldsymbol{u}_0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{p}_j,$$

and $\boldsymbol{p}_{j-1} \in \mathcal{K}_j(A, \boldsymbol{r}_0) \subseteq \mathcal{K}_k(A, \boldsymbol{r}_0)$ for $j = 1, 2, \ldots, k$, see Lemma 7, and hence

$$\boldsymbol{u}_k \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0).$$

Now let $\widetilde{\boldsymbol{u}} := \boldsymbol{u}_k + \delta\boldsymbol{u}$ for some $\delta\boldsymbol{u} \in \mathcal{K}_k(A, \boldsymbol{r}_0)$. This implies that $\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$, and we obtain

$$\begin{aligned}
\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}\|_A^2 &= (\boldsymbol{u} - \widetilde{\boldsymbol{u}})^\top A(\boldsymbol{u} - \widetilde{\boldsymbol{u}}) \\
&= (\boldsymbol{u} - \boldsymbol{u}_k - \delta\boldsymbol{u})^\top A(\boldsymbol{u} - \boldsymbol{u}_k - \delta\boldsymbol{u}) \\
&= (\boldsymbol{u} - \boldsymbol{u}_k)^\top A(\boldsymbol{u} - \boldsymbol{u}_k) - \delta\boldsymbol{u}^\top A(\boldsymbol{u} - \boldsymbol{u}_k) \\
&\quad -(\boldsymbol{u} - \boldsymbol{u}_k)^\top A\delta\boldsymbol{u} + \delta\boldsymbol{u}^\top A\delta\boldsymbol{u} \\
&= \|\boldsymbol{u} - \boldsymbol{u}_k\|_A^2 - 2\delta\boldsymbol{u}^\top A(\boldsymbol{u} - \boldsymbol{u}_k) + \|\delta\boldsymbol{u}\|_A^2 \\
&= \|\boldsymbol{u} - \boldsymbol{u}_k\|_A^2 + \|\delta\boldsymbol{u}\|_A^2,
\end{aligned}$$

where we used that $A^\top = A$ for the second last line, and for the last line the orthogonality of the residuals to the Krylov space from Remark 1,

$$A(\boldsymbol{u} - \boldsymbol{u}_k) = \boldsymbol{f} - A\boldsymbol{u}_k = \boldsymbol{r}_k \perp \mathcal{K}_k(A, \boldsymbol{r}_0),$$

which implies that the mixed term $2\delta\boldsymbol{u}^\top A(\boldsymbol{u} - \boldsymbol{u}_{k+1}) = 0$, since $\delta\boldsymbol{u} \in \mathcal{K}_k(A, \boldsymbol{r}_0)$. Therefore $\|\boldsymbol{u} - \widetilde{\boldsymbol{u}}\|_A^2$ is minimal if $\delta\boldsymbol{u} = 0$, which means that $\boldsymbol{u}_k = \widetilde{\boldsymbol{u}}$, the solution of the best approximation problem (3.29). □

We next show that the best approximation problem can be interpreted as a polynomial approximation problem.

Lemma 10 (Polynomial best approximation property of CG)**.** *Under the same hypotheses as in Lemma 9, we have*

$$\|\boldsymbol{u} - \boldsymbol{u}_k\|_A = \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A,$$

where $\mathcal{P}_k$ denotes the set of polynomials[5] *of degree lower than or equal to k and $\boldsymbol{u}_0$ is a given initial guess.*

Proof. Recalling Definition 10, we have

$$\boldsymbol{u}_k \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0) = \boldsymbol{u}_0 + \operatorname{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0\},$$

which implies that there exist coefficients γ_j such that

$$\boldsymbol{u}_k = \boldsymbol{u}_0 + \sum_{j=0}^{k-1} \gamma_j A^j \boldsymbol{r}_0 = \boldsymbol{u}_0 + q_{k-1}(A)\boldsymbol{r}_0$$

for the polynomial $q_{k-1}(x) := \sum_{j=0}^{k-1} \gamma_j x^j$ with $q_{k-1} \in \mathcal{P}_{k-1}$. Similarly, for any $\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$ we have that $\widetilde{\boldsymbol{u}} = \boldsymbol{u}_0 + \widetilde{q}(A)\boldsymbol{r}_0$ for some polynomial $\widetilde{q} \in \mathcal{P}_{k-1}$. Now from Lemma 9, we know that $\boldsymbol{u}_k$ minimizes the error in the A-norm,

$$\begin{aligned}
\|\boldsymbol{u} - \boldsymbol{u}_k\|_A &= \min_{\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)} \|\boldsymbol{u} - \widetilde{\boldsymbol{u}}\|_A \\
&= \min_{\widetilde{q} \in \mathcal{P}_{k-1}} \|\boldsymbol{u} - (\boldsymbol{u}_0 + \widetilde{q}(A)\boldsymbol{r}_0)\|_A \\
&= \min_{\widetilde{q} \in \mathcal{P}_{k-1}} \|\boldsymbol{u} - (\boldsymbol{u}_0 + \widetilde{q}(A)(\boldsymbol{f} - A\boldsymbol{u}_0))\|_A \\
&= \min_{\widetilde{q} \in \mathcal{P}_{k-1}} \|\boldsymbol{u} - (\boldsymbol{u}_0 + \widetilde{q}(A)A(\boldsymbol{u} - \boldsymbol{u}_0))\|_A \\
&= \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A,
\end{aligned}$$

where on the last line the new polynomial $p(x) = 1 - \widetilde{q}(x)x$ is in $\mathcal{P}_k$ since $\widetilde{q}(x)$ is in $\mathcal{P}_{k-1}$, and $p(0) = 1$. □

Corollary 2 (Convergence of CG in finite number of iterations)**.** *If the (symmetric positive definite) matrix A has $m \leq n$ distinct eigenvalues, then CG converges in at most m steps.*

Proof. Since A is diagonalizable and has m distinct eigenvalues, Lemma 6 (b) implies that the minimal polynomial $p_{\min}$ of A has degree m. Moreover, since A is invertible, Lemma 6 ensures that the constant coefficient c_0 of the minimal polynomial $p_{\min}$ is non-zero. Hence we can define the polynomial $p^*_{\min} := p_{\min}/c_0$ that satisfies $p^*_{\min}(0) = 1$ and $p^*_{\min}(A) = 0$, and has degree m. Therefore, by Lemma 10 at the mth iterate we have

$$\|\boldsymbol{u} - \boldsymbol{u}_m\|_A = \min_{\substack{p \in \mathcal{P}_m \\ p(0)=1}} \|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A \leq \|p^*_{\min}(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A = 0,$$

which implies the claim. □

[5] With the condition $p(0) = 1$ these are called residual polynomials.

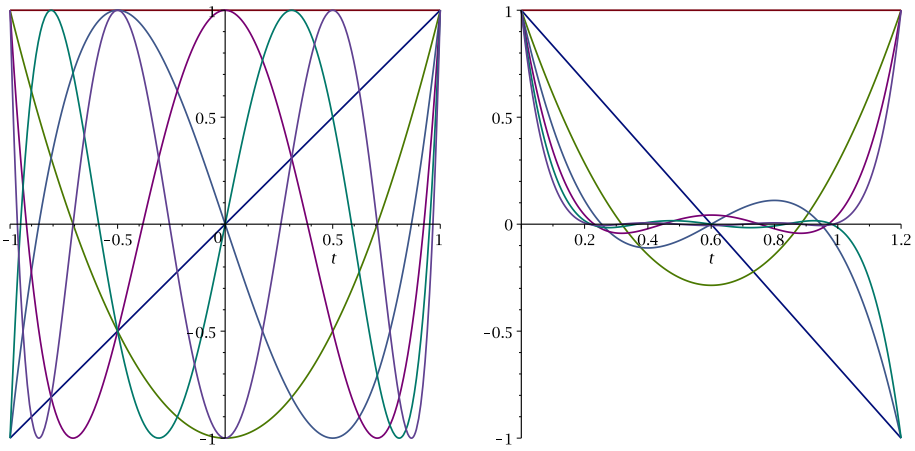


Figure 3.8: The first seven Chebyshev polynomials on the left, and the shifted scaled ones on the right.

In order to obtain a convergence estimate for the conjugate gradient method, we need the *Chebyshev polynomials*, which go back to the work by Chebyshev on minimizing the wear and tear of the transmission used in steam engines from the cylinder to the wheels [18].

Definition 12 (Chebyshev Polynomials)**.** *The* Chebyshev polynomials $T_k(t)$ *are defined by*

$$T_k(t) := \cos(k \arccos t), \quad -1 \le t \le 1, \quad k = 0, 1, 2, \ldots \tag{3.30}$$

We show in Figure 3.8 the first seven Chebyshev polynomials, which we plotted using the MAPLE commands

```
> with(orthopoly);
> plot([seq(T(i,t),i=0..6)],t=-1..1);
```

The definition of T_k does not readily reveal that it is a polynomial, but when evaluating the first ones, we find

$$\begin{aligned} T_0(t) &= \cos(0 \arccos t) = \cos(0) = 1, \\ T_1(t) &= \cos(1 \arccos t) = t \\ T_2(t) &= \cos(2 \arccos t) = \cos(\arccos t)^2 - \sin(\arccos t)^2 = t^2 - (1 - t^2) = 2t^2 - 1, \end{aligned}$$

where we used on the last line the trigonometric identities $\cos(a+b) = \cos a \cos b - \sin a \sin b$ and $\sin^2 x + \cos^2 x = 1$. It is however not convenient to continue in this fashion. A better approach is to sum the trigonometric identities

$$\cos((k+1)\alpha) = \cos(k\alpha + \alpha) = \cos(k\alpha)\cos\alpha - \sin(k\alpha)\sin\alpha,$$

and

$$\cos((k-1)\alpha) = \cos(k\alpha - \alpha) = \cos(k\alpha)\cos\alpha + \sin(k\alpha)\sin\alpha,$$

leading to

$$\cos((k+1)\alpha) + \cos((k-1)\alpha) = 2\cos(k\alpha)\cos\alpha,$$

which translates by inserting $\alpha = \arccos t$ into the well-known *three-term recurrence relation* for Chebyshev polynomials,

$$T_0(t) = 1, \quad T_1(t) = t, \quad T_{k+1}(t) = 2tT_k(t) - T_{k-1}(t), \quad k = 1, 2, \ldots \tag{3.31}$$

From this recurrence relation, we can clearly see now that the $T_k(t)$ are polynomials of degree k.

For k even, T_k is an even function, i.e., $T_k(t) = T_k(-t)$, for all t. This can be proved by induction from the recurrence relation. Similarly, for k odd, we have $T_k(t) = -T_k(-t)$. These properties can also be observed in Figure 3.8.

The important result related to Chebyshev polynomials we will need for the convergence analysis is that the polynomial which solves the min-max problem

$$\widetilde{p} = \underset{\substack{p\in\mathcal{P}_k \\ p(0)=1}}{\arg\min} \max_{\substack{t\in[\alpha,\beta] \\ 0\notin[\alpha,\beta]}} |p(t)|, \tag{3.32}$$

i.e. the smallest polynomial on the interval $[\alpha, \beta]$ which equals one at zero, is the following shifted and scaled Chebyshev polynomial; see, e.g., [108, Section 6.11].

$$\widetilde{p}(t) = \frac{T_k(1 - 2\frac{\beta - t}{\beta - \alpha})}{T_k(1 - 2\frac{\beta}{\beta - \alpha})}. \tag{3.33}$$

The first few such polynomials are shown in Figure 3.8 on the right, plotted with the MAPLE commands

```
> with(orthopoly);
> a:=0.2;b:=1;
> plot([seq(T(i,1-2*(b-t)/(b-a))/T(i,1-2*b/(b-a)),i=0..6)],t=0..1.2);
```

We can nicely see how they become small very rapidly in the interval $[\alpha, \beta]$ as the degree increases, and how they equioscillate around the maximum on the interval $[\alpha, \beta]$, and grow rapidly outside the interval. These would have been the polynomials Richardson could have used in the true Richardson's method (3.1), see also Figure 2.16 where he tried by trial and error to find a good such polynomial. Before we show that the conjugate gradient method converges at least as fast as such an optimized true Richardson's method, which was realized by the Chebyshev semi-iterative method of *Gene Golub* [61], we need a final property of the Chebyshev polynomials, which appears naturally when we generalize the argument $t \in \mathbb{R}$ to a complex argument $z \in \mathbb{C}$:

Lemma 11 (Complex representation of Chebyshev polynomials)**.** *The Chebyshev polynomials can also be expressed as*

$$T_k(z) = \frac{w^k + w^{-k}}{2}, \quad z = (w + w^{-1})/2, \quad \textit{with } w, z \in \mathbb{C}. \tag{3.34}$$

Proof. We verify by induction that the quantities $T_k(z)$ defined by (3.34) obey the three-term recurrence relation (3.31). For $k = 0$ we have $T_0(z) = (w^0 + w^{-0})/2 = 1$, and it is clear that $T_1(z) = (w + w^{-1})/2 = z$. So we assume now that (3.34) holds for $j \leq k$. Then

$$\begin{aligned} T_{k+1}(z) &= 2zT_k(z) - T_{k-1}(z) \\ &= (w + w^{-1})\frac{w^k + w^{-k}}{2} - \frac{w^{k-1} + w^{-(k-1)}}{2} \\ &= \frac{w^{k+1} + w^{-k+1} + w^{k-1} + w^{-k-1} - w^{k-1} - w^{-(k-1)}}{2} \\ &= \frac{w^{k+1} + w^{-(k+1)}}{2}, \end{aligned}$$

which concludes the proof. □

Note that if we solve the equation

$$z = (w + w^{-1})/2 \quad \Longleftrightarrow \quad w^2 - 2zw + 1 = 0$$

for w, we get the two solutions

$$w_1 = z + \sqrt{z^2 - 1}, \quad w_2 = z - \sqrt{z^2 - 1} = \frac{1}{w_1}.$$

Hence, we have $w_1 = \frac{1}{w_2}$. Thus $T_k(z)$ does not depend on which root is chosen, and we find in either case

$$T_k(z) = \frac{\left(z + \sqrt{z^2 - 1}\right)^k + \left(z - \sqrt{z^2 - 1}\right)^k}{2}. \tag{3.35}$$

We are now ready to prove a convergence estimate for the conjugate gradient method.

Theorem 25 (Convergence estimate for CG)**.** *Let $\boldsymbol{u} \in \mathbb{R}^n$ be the solution of $A\boldsymbol{u} = \boldsymbol{f}$, with $A \in \mathbb{R}^{n\times n}$ symmetric and positive definite with $\lambda_{\max} > \lambda_{\min}$, and $\boldsymbol{f} \in \mathbb{R}^n$ with a given initial guess $\boldsymbol{u}_0 \in \mathbb{R}^n$. Then the approximation $\boldsymbol{u}_k$ computed by CG satisfies the estimate*

$$\|\boldsymbol{u} - \boldsymbol{u}_k\|_A \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k \|\boldsymbol{u} - \boldsymbol{u}_0\|_A, \tag{3.36}$$

where $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ is the condition number of the matrix A.

Proof. From Lemma 10 we have that

$$\|\boldsymbol{u} - \boldsymbol{u}_k\|_A = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A. \tag{3.37}$$

Since A is symmetric, its Schur decomposition is $A = Q\Lambda Q^\top$, with

$$Q = \begin{bmatrix} \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n \end{bmatrix} \quad \text{and} \quad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix},$$

where Q is orthogonal and $\boldsymbol{q}_j$ and λ_j are the eigenvectors and the eigenvalues of A. Recall that since $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, its eigenvectors form a basis for $\mathbb{R}^n$. Therefore, we can decompose $\boldsymbol{u} - \boldsymbol{u}_0$ into eigenvectors of A and obtain

$$\boldsymbol{u} - \boldsymbol{u}_0 = \sum_{j=1}^{n} \gamma_j \boldsymbol{q}_j,$$

for appropriate scalars γ_j. This implies that

$$p(A)(\boldsymbol{u} - \boldsymbol{u}_0) = \sum_{j=1}^{n} \gamma_j p(A)\boldsymbol{q}_j = \sum_{j=1}^{n} \gamma_j p(\lambda_j)\boldsymbol{q}_j, \tag{3.38}$$

where we used that $\boldsymbol{q}_j$ are eigenvectors of A. Moreover, using that $Q^\top(\boldsymbol{u}-\boldsymbol{u}_0) = Q^\top \sum_{j=1}^n \gamma_j \boldsymbol{q}_j = \sum_{j=1}^n \gamma_j \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ denotes the jth canonical vector, we get

$$\begin{aligned} \|\boldsymbol{u} - \boldsymbol{u}_0\|_A^2 &= (\boldsymbol{u} - \boldsymbol{u}_0)^\top A(\boldsymbol{u} - \boldsymbol{u}_0) \\ &= (\boldsymbol{u} - \boldsymbol{u}_0)^\top Q\Lambda Q^\top(\boldsymbol{u} - \boldsymbol{u}_0) \\ &= (Q^\top(\boldsymbol{u} - \boldsymbol{u}_0))^\top \Lambda (Q^\top(\boldsymbol{u} - \boldsymbol{u}_0)) \\ &= \sum_{j=1}^{n} \gamma_j^2 \lambda_j. \end{aligned} \tag{3.39}$$

Using (3.38) and (3.39), we obtain

$$\begin{aligned} \|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A^2 &= \Big(p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\Big)^\top A\Big(p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\Big) \\ &= \left(\sum_{j=1}^{n} \gamma_j p(\lambda_j)\boldsymbol{q}_j\right)^\top Q\Lambda Q^\top \left(\sum_{j=1}^{n} \gamma_j p(\lambda_j)\boldsymbol{q}_j\right) \\ &= \sum_{j=1}^{n} (p(\lambda_j))^2 \gamma_j^2 \lambda_j \le \Big(\max_{j=1,2,\ldots} |p(\lambda_j)|^2\Big) \sum_{j=1}^{n} \gamma_j^2 \lambda_j \\ &= \Big(\max_{j=1,2,\ldots} |p(\lambda_j)|^2\Big) \|\boldsymbol{u} - \boldsymbol{u}_0\|_A^2. \end{aligned}$$

Next, we denote by $\lambda_{\max}$ and $\lambda_{\min}$ the maximum and minimum eigenvalues of A. By taking the square root, and estimating over the entire interval that contains the positive eigenvalues of A instead of the discrete set, we get

$$\|p(A)(\boldsymbol{u} - \boldsymbol{u}_0)\|_A \le \max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} |p(\lambda)| \|\boldsymbol{u} - \boldsymbol{u}_0\|_A. \tag{3.40}$$

Replacing (3.40) into (3.37) leads to

$$\|\boldsymbol{u}-\boldsymbol{u}_k\|_A \leq \min_{\substack{p\in\mathcal{P}_k\\ p(0)=1}} \max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} |p(\lambda)|\|\boldsymbol{u}-\boldsymbol{u}_0\|_A. \tag{3.41}$$

Now, the polynomial that solves the min-max problem on the right-hand side of (3.41), that is the smallest on the interval $[\lambda_{\min},\lambda_{\max}]$ with $p(0)=1$, is the real, scaled, and shifted Chebyshev polynomial

$$\widetilde{p}(\lambda) := \frac{T_k\Big(1-2\frac{\lambda_{\max}-\lambda}{\lambda_{\max}-\lambda_{\min}}\Big)}{T_k\Big(1-2\frac{\lambda_{\max}}{\lambda_{\max}-\lambda_{\min}}\Big)},$$

where T_k is the Chebyshev polynomial defined on the interval $[-1,1]$. Recall that $\lambda_{\max}>\lambda_{\min}$. We have

$$\begin{aligned}\max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} |\widetilde{p}(\lambda)| &= \frac{1}{\Big|T_k\Big(1-2\frac{\lambda_{\max}}{\lambda_{\max}-\lambda_{\min}}\Big)\Big|} = \frac{1}{\Big|T_k\Big(\frac{\lambda_{\max}-\lambda_{\min}-2\lambda_{\max}}{\lambda_{\max}-\lambda_{\min}}\Big)\Big|}\\ &= \frac{1}{\Big|T_k\Big(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\Big)\Big|},\end{aligned}$$

where we used the properties of T_k, see Figure 3.8. Recalling that $\kappa(A)=\frac{\lambda_{\max}}{\lambda_{\min}}$, we obtain

$$\max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} |\widetilde{p}(\lambda)| = \frac{1}{\Big|T_k\Big(\frac{\kappa(A)+1}{\kappa(A)-1}\Big)\Big|}. \tag{3.42}$$

Using the complex representation (3.34) given in Lemma 11, that is

$$T_k(z)=\frac{w^k+w^{-k}}{2}, \quad z=\frac{w+w^{-1}}{2}, \quad \text{with } w,z\in\mathbb{C},$$

we find (for $z=\frac{\kappa(A)+1}{\kappa(A)-1}$) that

$$w^2-2zw+1 = w^2-2\frac{\kappa(A)+1}{\kappa(A)-1}w+1=0,$$

which is solved by

$$\begin{aligned} w &= \frac{\kappa(A)+1}{\kappa(A)-1}+\sqrt{\Big(\frac{\kappa(A)+1}{\kappa(A)-1}\Big)^2-1}\\ &= \frac{\kappa(A)+1+\sqrt{(\kappa(A)+1)^2-(\kappa(A)-1)^2}}{\kappa(A)-1}\\ &= \frac{\kappa(A)+1+\sqrt{4\kappa(A)}}{\kappa(A)-1}\\ &= \frac{(\sqrt{\kappa(A)}+1)^2}{(\sqrt{\kappa(A)}+1)(\sqrt{\kappa(A)}-1)}\\ &= \frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}.\end{aligned}$$

Therefore, we have that

$$T_k(z) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}\right)^k + \left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}\right)^{-k}\right]. \tag{3.43}$$

Finally, by combining (3.41), (3.42), and (3.43), we obtain

$$\begin{aligned}
\|\boldsymbol{u}-\boldsymbol{u}_k\|_A &\leq \frac{2}{\left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}\right)^k + \left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}\right)^{-k}}\|\boldsymbol{u}-\boldsymbol{u}_0\|_A \\
&= \frac{2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k}{1+\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^{2k}}\|\boldsymbol{u}-\boldsymbol{u}_0\|_A \\
&\leq 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|\boldsymbol{u}-\boldsymbol{u}_0\|_A,
\end{aligned}$$

which is our claim. □

Note that in the last step of the proof, we neglected the additional term in the denominator, which becomes small when the iteration number k becomes large, so the estimate remains sharp for k large. For k small however, especially if the condition number $\kappa(A)$ is large, this term compensates almost the additional factor 2 in the estimate. Notice that, using (3.36) a similar estimate can be obtained in the norm $\|\cdot\|_2$; see Problem 30.

We show in Figure 3.9 a comparison of the convergence of steepest descent, SOR with the optimally chosen parameter, and the conjugate gradient algorithm applied to the discretized Laplace problem on the unit square with 16 grid points, whose solution was shown in Figure 1.11. We measured the error in the A-norm, and also plot the convergence estimate we proved in Theorem 19 for Steepest Descent, the asymptotic contraction factor of SOR from (2.42), and the convergence estimate from Theorem 25 for the conjugate gradient method. We see that steepest descent is clearly the slowest method, a good local tactic is not necessarily good for an efficient global strategy, and the convergence estimate we proved is accurate. The conjugate gradient method on the other hand is by far the fastest asymptotically. It only initially follows the pessimistic convergence estimate from Theorem 25, and then speeds up after about 30 iterations. SOR with optimized parameter is initially as fast as CG, but is then beaten after about 30 iterations in this example. This shows that well tuned stationary iterative methods can still be interesting, and we will see in Chapter 4 that it is precisely such methods that lead to the best preconditioners when they are used combined with Krylov acceleration.

Preconditioners are necessary, since the conjugate gradient method, and Krylov methods in general, can not remove the mesh dependence of their convergence, since they still use only local connections between the grid points

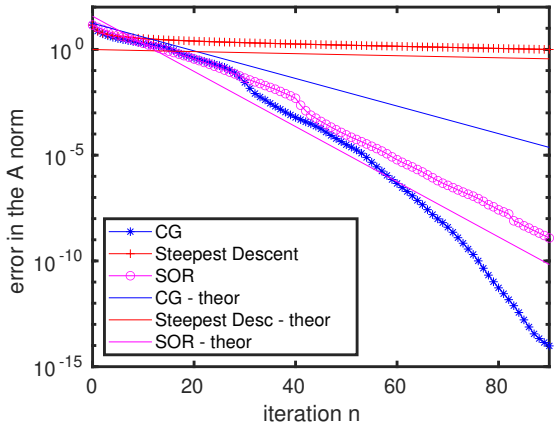


Figure 3.9: Decay of the error when using Steepest Descent, SOR with optimized parameter and the conjugate gradient method, together with the theoretical convergence estimates.

in the discretization matrix when performing the matrix vector products, even though an optimal polynomial combination is then used to approximate the solution. This we can see already in Figure 3.7, where still unknowns with small y coordinate are not improving their approximation and remain at zero. We illustrate this in Figure 3.10 where we see how the error decreases as the iterations progress when CG is used to solve our Laplace model problem from Section 1.3 for the mesh we used for Figure 3.7 with $m = 15$ interior mesh points, and two refined meshes with $m = 31$ and $m = 63$ interior mesh points. The convergence deterioration of CG can be also seen by studying the convergence factor estimated in Theorem 25 as a function of the grid size h. The expansion of this convergence factor around zero leads to

$$\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} = 1 - \pi h + O(h^2),$$

which is similar to the result obtained for the SOR method (with optimal parameter) and shows clearly that CG behaves better than Jacobi, Gauß-Seidel and steepest descent. Moreover, notice that asymptotically CG converges faster than the predicted theoretical upper bound.

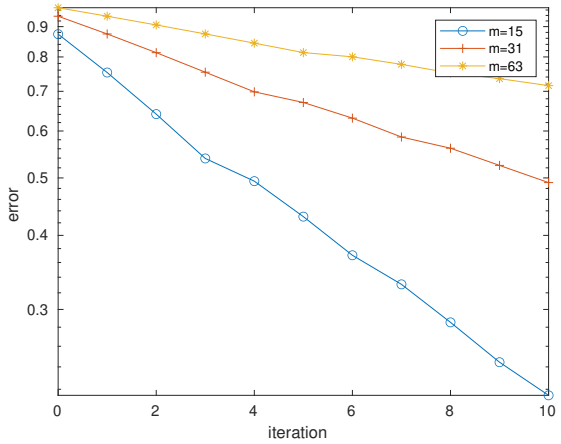


Figure 3.10: Convergence of the CG method for different mesh sizes $h = \frac{1}{m+1}$.

3.3 The Arnoldi iteration

> *An interpretation of Dr. Cornelius Lanczos' iteration method, which he has named "minimized iterations", is discussed in this article, expounding the method as applied to the solution of the characteristic matrix equations both in homogeneous and non-homogeneous form. This interpretation leads to a variation of the Lanczos procedure which may frequently be advantageous by virtue of reducing the volume of numerical work in practical applications. Both methods employ essentially the same algorithm, requiring the generation of a series of orthogonal functions through which a simple matrix equation of reduced order is established.*
>
> Walter E. Arnoldi, The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem, 1951.

The *Arnoldi iteration* was invented by *Walter E. Arnoldi* in 1951 [1] following the invention of the Lanczos algorithm by *Cornelius Lanczos* in 1950 [83], see the quote above, but we start here with the Arnoldi iteration, because then the Lanczos process treated in the next section becomes just a special case for symmetric positive definite matrices.

Each matrix $A \in \mathbb{R}^{n\times n}$ can be transformed by an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ into an upper Hessenberg matrix $H \in \mathbb{R}^{n\times n}$,

$$A = QHQ^\top, \tag{3.44}$$

where the upper Hessenberg matrix has only zero entries in the lower triangular part except for next to the diagonal, i.e. $H_{i,j} = 0$ for $i > j + 1$. This can be achieved for example using Givens rotations, or Householder reflections, see [57, Section 7.5]. Arnoldi first considered only the first k columns of the matrix Q,

$$Q_k := [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k] .$$

From (3.44), we have

$$AQ = QH,$$

and if we only consider the first k columns, we get

$$A\,[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k] = \big[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{k+1}\big] \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ & \ddots & \ddots & \vdots \\ & & \ddots & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix},$$

or

$$AQ_k = Q_{k+1}H_k.$$

The kth column in this matrix identity states

$$A\boldsymbol{q}_k = \sum_{j=1}^{k+1} h_{jk}\boldsymbol{q}_j \quad \Longleftrightarrow \quad h_{k+1,k}\boldsymbol{q}_{k+1} = A\boldsymbol{q}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{q}_j. \tag{3.45}$$

Hence, if we know the first k vectors $\boldsymbol{q}_1$, $\boldsymbol{q}_1$, $\ldots$, $\boldsymbol{q}_k$, we can compute $\boldsymbol{q}_{k+1}$. In fact, (3.45) is just an orthogonalization of a new vector $A\boldsymbol{q}_k$ with respect to the already known vectors $\boldsymbol{q}_j$, $j = 1, 2, \ldots, k$. If one uses the (modified) Gram-Schmidt orthogonalization process to do this, we obtain, starting with an arbitrary vector $\boldsymbol{b}$ the Arnoldi iteration

```
function [H,Q,v]=Arnoldi(A,b,k)
% ARNOLDI Arnoldi iteration
%   [H,Q,v]=Arnoldi(A,b,k) applies k<=n steps of the Arnoldi iteration to
%   the matrix A starting with the vector b. Computes Q orthogonal and
%   H upper Hessenberg such that AQ=QH+ve_k^T, with Q(:,1)=b/norm(b).

Q(:,1)=b/norm(b);
for j=1:k
  v=A*Q(:,j);
  for i=1:j
    H(i,j)=Q(:,i)'*v;
    v=v-H(i,j)*Q(:,i);
  end
  if j<k
    H(j+1,j)=norm(v);
    if H(j+1,j)<1e-15, disp('lucky breakdown'); break; end
    Q(:,j+1)=v/H(j+1,j);
  end
end
```

Notice that in the previous algorithm the vector `v` given as an output is $\mathtt{v} = A\boldsymbol{q}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{q}_j$. The output matrices of the algorithm are $\mathtt{Q} = Q_k \in \mathbb{R}^{n\times k}$ and $\mathtt{H} \in \mathbb{R}^{k\times k}$, and observe that

$$\begin{aligned} AQ_k = Q_{k+1}H_k &= \begin{bmatrix} Q_k & \boldsymbol{q}_{k+1} \end{bmatrix} \begin{bmatrix} \mathtt{H} \\ h_{k+1,k}\boldsymbol{e}_k^\top \end{bmatrix} = Q_k\mathtt{H} + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^\top \\ &= Q_k\mathtt{H} + \Big(A\boldsymbol{q}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{q}_j\Big)\boldsymbol{e}_k^\top = \mathtt{QH} + \mathtt{v}\boldsymbol{e}_k^\top, \end{aligned} \tag{3.46}$$

where $\boldsymbol{e}_k$ denotes the kth canonical vector and we used (3.45). Notice that $\mathtt{H} \in \mathbb{R}^{k\times k}$ is the matrix $H_k \in \mathbb{R}^{k+1\times k}$ without the last row.

Moreover, we remark that $h_{j+1,j}$ is posed to be equal to $\|\mathtt{v}\|_2$. This can be explained by noticing that $\mathtt{v} = A\boldsymbol{q}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{q}_j = h_{k+1,k}\boldsymbol{q}_{k+1}$ (by (3.45)). Hence, we have that $\boldsymbol{q}_{k+1} = \frac{1}{h_{k+1,k}}\mathtt{v}$. Since $\boldsymbol{q}_{j+1}$ has to be normalized, if follows that $h_{j+1,j} = \|\mathtt{v}\|_2$.

The following lemma shows that the vectors $\boldsymbol{q}_j$ generated by the Arnoldi iterations form a basis of the Krylov space $\mathcal{K}_k(A, \boldsymbol{q}_1)$; see, e.g., [108].

Lemma 12 (Arnoldi as span of Krylov spaces)**.** *Assume that the Arnoldi procedure generates the vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ with $h_{j+1,j} \neq 0$ for $j = 1, \ldots, k$. Then, the vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ form an orthonormal basis of the Krylov space*

$$\mathcal{K}_k(A, \boldsymbol{q}_1) = \operatorname{span}\{\boldsymbol{q}_1, A\boldsymbol{q}_1, \ldots, A^{k-1}\boldsymbol{q}_1\}.$$

Proof. The orthonormality of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ follows by their construction in the Arnoldi procedure.

We want to show that each vector $\boldsymbol{q}_j$ can be written as $q_{j-1}(A)\boldsymbol{q}_1$, where q_{j-1} is a polynomial of degree $j-1$. This is obtained by induction. The result trivially holds for $j = 1$ because $\boldsymbol{q}_1 = q_0(A)\boldsymbol{q}_1$ with $q_0(x) := 1$. Now, we assume that $\boldsymbol{q}_\ell = q_{\ell-1}(A)\boldsymbol{q}_1$ for $\ell = 1, \ldots, j$, and we want to show that the same holds for $\ell = j+1$. Recalling (3.45) and using the induction hypothesis, we obtain

$$h_{j+1,j}\boldsymbol{q}_{j+1} = A\boldsymbol{q}_j - \sum_{\ell=1}^{j} h_{\ell,j}\boldsymbol{q}_\ell = Aq_{j-1}(A)\boldsymbol{q}_1 - \sum_{\ell=1}^{j} h_{\ell,j}q_{\ell-1}(A)\boldsymbol{q}_1,$$

which means that $\boldsymbol{q}_{j+1}$ can be written as $q_j(A)\boldsymbol{q}_1$, where q_j is a polynomial of degree j.

Since each $\boldsymbol{q}_j$ has the form of $q_{j-1}(A)\boldsymbol{q}_1$, it holds that

$$\operatorname{span}\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k\} \subset \mathcal{K}_k(A, \boldsymbol{q}_1).$$

The claim follows recalling the orthonormality of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$ and the fact that $\dim \mathcal{K}_k(A, \boldsymbol{q}_1) \leq k$. □

The following lemma explains the lucky breakdown in the Arnoldi algorithm presented above.

Lemma 13 (Arnoldi breakdown)**.** *Let A be invertible. Arnoldi's method stops at step j, that is $h_{j+1,j} = 0$, if and only if the minimal polynomial $p_{\min}^{\boldsymbol{q}_1}$ of the pair $(A, \boldsymbol{q}_1)$ is of degree j. In this case, the space $\mathcal{K}_j(A, \boldsymbol{q}_1)$ is invariant under A.*

Proof. First we show that if $\deg(p_{\min}^{\boldsymbol{q}_1}) = j$, then $h_{j+1,j} = 0$. Assume by contradiction that $h_{j+1,j} \neq 0$, then $\boldsymbol{q}_{k+1}$ can be defined by (3.45). By Lemma 12 the Krylov space $\mathcal{K}_{j+1}(A, \boldsymbol{q}_1)$ would be of dimension $j+1$. Since $\deg(p_{\min}^{\boldsymbol{q}_1}) = j < j+1$, we get a contradiction to Theorem 23. Hence $h_{j+1,j} = 0$.

To show the reverse, assume that $h_{j+1,j} = 0$, that is $\boldsymbol{q}_{j+1} = 0$. Then Lemma 12 (together with the orthogonality of the $\boldsymbol{q}_j$) implies that $\dim \mathcal{K}_j(A, \boldsymbol{q}_1) = j$. Therefore, Theorem 23 ensures that $\deg(p_{\min}^{\boldsymbol{q}_1}) \geq j$. Moreover, since $\boldsymbol{q}_{j+1} = 0$ we have $\dim \mathcal{K}_{j+1}(A, \boldsymbol{q}_1) < j + 1$, otherwise there would be a contradiction to Lemma 12. Since $\dim \mathcal{K}_{j+1}(A, \boldsymbol{q}_1) < j + 1$ we cannot have that $\deg(p_{\min}^{\boldsymbol{q}_1}) > j$ according to Theorem 23. Therefore we obtain that $\deg(p_{\min}^{\boldsymbol{q}_1}) = j$.

The fact that $\mathcal{K}_j(A, \boldsymbol{q}_1)$ is invariant under A follows then from Theorem 22. □

Remark 2. *The eigenvalues of the matrix H_k without the last line (that is the matrix `H` in the above algorithm) are good approximations to the extremal eigenvalues of A, which was the main motivation of Arnoldi for this iteration. This is suggested by the formula* (3.46)*. In fact, by multiplying by $Q_k^\top$, we get $Q_k^\top A Q_k =$ `H`. We refer to [120, Lecture 34].*

Remark 3. *Notice that the previous algorithm uses the modified Gram-Schmidt orthogonalization procedure. However, there are other more efficient practical implementations of the Arnoldi procedure based, e.g., on Householder reflections; see, e.g., [108, Section 6.3.2].*

3.4 The Lanczos algorithm

> *Moreover, the method leads to a well convergent successive approximation procedure by which the solution of integral equations of the Fredholm type and the solution of the eigenvalue problem of linear differential and integral operators may be accomplished.*
>
> Cornelius Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, 1950.

If we apply the Arnoldi iteration to a symmetric matrix, $A = A^\top$, then the decomposition (3.44) can be simplified. Since

$$QHQ^\top = A = A^\top = (QHQ^\top)^\top = QH^\top Q^\top,$$

the Hessenberg matrix must also be symmetric, $H^\top = H$, and we obtain a tridiagonal matrix $T := H = H^\top$. This also simplifies the recurrence relation (3.45), namely

$$t_{k+1,k}\boldsymbol{q}_{k+1} = A\boldsymbol{q}_k - \sum_{j=k-1}^{k} t_{jk}\boldsymbol{q}_j,$$

because all the remaining terms are zero, or

$$\beta_k \boldsymbol{q}_{k+1} = A\boldsymbol{q}_k - \beta_{k-1}\boldsymbol{q}_{k-1} - \alpha_k \boldsymbol{q}_k, \tag{3.47}$$

if the tridiagonal matrix entries are named as

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

We thus obtain the Lanczos algorithm

```
function [alpha,beta,Q,v]=Lanczos(A,b,k)
% LANCZOS Lanczos algorithm
%   [alpha,beta,Q,v]=Lanczos(A,b,k) applies k<=n steps of the Lanczos
%   algorithm to the symmetric matrix A starting with the vector b.
%   Computes Q orthogonal and a symmetric tridiagonal matrix given
%   by the diagonal in the vector alpha, and the super- and
%   subdiagonal in the vector beta.

Q(:,1)=b/norm(b);
if k==1, beta=[]; end;
for j=1:k
  v=A*Q(:,j);
  alpha(j)=Q(:,j)'*v;
  v=v-alpha(j)*Q(:,j);
  if j>1
    v=v-beta(j-1)*Q(:,j-1);
  end
  if j<k
    beta(j)=norm(v);
    Q(:,j+1)=v/beta(j);
  end
end
```

The choice of naming the entries α_j and β_j in the tridiagonal matrix T is not a coincidence, the Lanczos algorithm is the same process as the one used by the conjugate gradient algorithm, see Theorem 21, there is just a scaling difference, see Problem 31. Since the Lanczos algorithm is the underlying method for the conjugate gradient method to solve symmetric linear systems, one could use the Arnoldi iteration to obtain a Krylov method for a general, non-symmetric matrix. GMRES is precisely realizing this idea but in a different way, as we show in the next section.

3.5 Generalized minimal residual: GMRES

> *We present an iterative method for solving linear systems, which has the property of minimizing at every step the norm of the residual vector over a Krylov subspace. The algorithm is derived from the Arnoldi process for constructing an l_2-orthogonal basis of Krylov subspaces. It can be considered as a generalization of Paige and Saunders'* MINRES *algorithm and is theoretically equivalent to the* Generalized Conjugate Residual *(*GCR*) method and to* ORTHODIR.
>
> *Yousef Saad* and *Martin H. Schultz*, GMRES: a Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, 1986.

The *Generalized Minimal Residual method* (GMRES), proposed by Saad and Schultz [112], constructs at iteration k an approximation $\boldsymbol{u}_k$ in the affine Krylov space $\boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$ not by making the new residual orthogonal to the current Krylov subspace like in the conjugate gradient method, but by making the new residual norm $\|\boldsymbol{r}_k\|_2 = \|\boldsymbol{f} - A\boldsymbol{u}_k\|_2$ as small as possible, see the quote

above. To simplify the notation, but without loss of generality, we will assume in the following that we start the iteration with $\boldsymbol{u}_0 = 0$. Then $\boldsymbol{r}_0 = \boldsymbol{f}$ and $\boldsymbol{u}_k \in \mathcal{K}_k(A, \boldsymbol{f})$.

Theorem 26 (Foundation of GMRES)**.** *Let $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{f} \in \mathbb{R}^n$ and let $AQ_k = Q_{k+1}H_k$ be the decomposition computed by the Arnoldi (initialized by $\boldsymbol{r}_0 = \boldsymbol{f}$) iteration at step k. Then the minimizer*

$$\boldsymbol{u}_k = \underset{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{f})}{\arg\min} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2$$

is given by $\boldsymbol{u}_k = Q_k\boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^k$ is the solution of the least squares problem

$$\boldsymbol{v} = \underset{\boldsymbol{w}\in\mathbb{R}^k}{\arg\min} \big\| \|\boldsymbol{f}\|_2\boldsymbol{e}_1 - H_k\boldsymbol{w}\big\|_2. \tag{3.48}$$

Proof. According to Lemma 12, Q_k from the Arnoldi iteration contains in its columns an orthonormal basis of the Krylov space $\mathcal{K}_k(A, \boldsymbol{f})$. Hence one can express $\widetilde{\boldsymbol{u}} \in \mathcal{K}_k(A, \boldsymbol{f})$ as $\widetilde{\boldsymbol{u}} = Q_k\boldsymbol{v}$ for some appropriate vector $\boldsymbol{v} \in \mathbb{R}^k$. Using also from the Arnoldi iteration that $AQ_k = Q_{k+1}H_k$, we thus get

$$\begin{aligned}\min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{f})} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 &= \min_{\boldsymbol{v}\in\mathbb{R}^k} \|\boldsymbol{f} - AQ_k\boldsymbol{v}\|_2 \\ &= \min_{\boldsymbol{v}\in\mathbb{R}^k} \|\boldsymbol{f} - Q_{k+1}H_k\boldsymbol{v}\|_2.\end{aligned}$$

Now since $\boldsymbol{q}_1 = \boldsymbol{f}/\|\boldsymbol{f}\|_2$ from the Arnoldi process,

$$\boldsymbol{f} = \|\boldsymbol{f}\|_2 Q_{k+1}\boldsymbol{e}_1,$$

with $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^\top \in \mathbb{R}^{k+1}$, and since for any vector $\boldsymbol{w}$ we have by the orthogonality of the columns of Q_{k+1} that

$$\|Q_{k+1}\boldsymbol{w}\|_2^2 = \boldsymbol{w}^\top Q_{k+1}^\top Q_{k+1}\boldsymbol{w} = \boldsymbol{w}^\top\boldsymbol{w} = \|\boldsymbol{w}\|_2^2,$$

we obtain

$$\begin{aligned}\min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{f})} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 &= \min_{\boldsymbol{v}\in\mathbb{R}^k} \big\|Q_{k+1}(\|\boldsymbol{f}\|_2\boldsymbol{e}_1 - H_k\boldsymbol{v})\big\|_2 \\ &= \min_{\boldsymbol{v}\in\mathbb{R}^k} \big\| \|\boldsymbol{f}\|_2\boldsymbol{e}_1 - H_k\boldsymbol{v}\big\|_2,\end{aligned}$$

which concludes the proof. □

Remark 4 (Foundation of GMRES for $\boldsymbol{u}_0 \neq 0$)**.** *Notice that Theorem 26 can be easily generalized for $\boldsymbol{u}_0 \neq 0$. In this case, we write*

$$\begin{aligned}\boldsymbol{u}_k &= \underset{\widetilde{\boldsymbol{u}}\in\boldsymbol{u}_0+\mathcal{K}_k(A,\boldsymbol{r}_0)}{\arg\min} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 = \boldsymbol{u}_0 + \underset{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{r}_0)}{\arg\min} \|\boldsymbol{f} - A\boldsymbol{u}_0 - A\widetilde{\boldsymbol{u}}\|_2 \\ &= \boldsymbol{u}_0 + \underset{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{r}_0)}{\arg\min} \|\boldsymbol{r}_0 - A\widetilde{\boldsymbol{u}}\|_2.\end{aligned}$$

Therefore, using Theorem 26 for the problem $\arg\min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{r}_0)} \|\boldsymbol{r}_0 - A\widetilde{\boldsymbol{u}}\|_2$ *we get that* $\boldsymbol{u}_k = \boldsymbol{u}_0 + Q_k\boldsymbol{v}$*, where*

$$\boldsymbol{v} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^k} \big\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - H_k\boldsymbol{w}\big\|_2,$$

and one clearly has that

$$\|\boldsymbol{r}_k\|_2 = \min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{r}_0)} \|\boldsymbol{r}_0 - A\widetilde{u}\|_2 = \min_{\boldsymbol{w}\in\mathbb{R}^k} \big\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - H_k\boldsymbol{w}\big\|_2.$$

The least squares problem (3.48) has a special structure: the matrix H_k is upper Hessenberg and there are $k+1$ equations and k unknowns. Such problems are best solved by applying k Givens rotations to reduce the system to an equivalent system with an upper triangular matrix; see Problem 32.

GMRES solves implicitly a polynomial approximation problem. Since $\boldsymbol{u}_k \in \mathcal{K}_k(A,\boldsymbol{f})$, it is a linear combination of the basis vectors $A^j\boldsymbol{f}$, $j = 0,\ldots,k-1$,

$$\boldsymbol{u}_k = \sum_{j=0}^{k-1} \gamma_j A^j \boldsymbol{f},$$

for some coefficients γ_j. For the residual $\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u}_k$, we get

$$\boldsymbol{r}_k = \boldsymbol{f} - \sum_{j=0}^{k-1} \gamma_j A^{j+1}\boldsymbol{f} = \sum_{j=0}^{k} \zeta_j A^j \boldsymbol{f}, \quad \zeta_0 = 1,\ \zeta_j = -\gamma_{j-1}.$$

With the *residual polynomial*

$$p_k(A) = \sum_{j=0}^{k} \zeta_j A^j, \quad p_k(0) = 1,$$

and denoting the set of all polynomials of degree lower than or equal to k by $\mathcal{P}_k$, the minimization problem becomes

$$\min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{f})} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 = \min_{\widetilde{\zeta}_1,\ldots,\widetilde{\zeta}_{k-1}} \|\boldsymbol{f} + \sum_{j=1}^{k} \widetilde{\zeta}_j A^j \boldsymbol{f}\|_2 = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{f}\|_2. \quad (3.49)$$

Notice that the polynomial approximation problem (3.49) seems similar to the one that we obtained in Lemma 10 for CG. However, there is a big difference between these two problems. It is possible to prove (see Problem 29) that at each iteration CG, for A symmetric and positive definite, minimizes the residual in the A^{-1}-norm ($\|\boldsymbol{z}\|_{A^{-1}} = \sqrt{\boldsymbol{z}^\top A^{-1}\boldsymbol{z}}$) in the sense that

$$\|\boldsymbol{r}_k\|_{A^{-1}} = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{r}_0\|_{A^{-1}}. \quad (3.50)$$

Recalling that $\boldsymbol{u}_0 = 0$ and hence $\boldsymbol{r}_0 = \boldsymbol{f}$, the GMRES polynomial approximation problem gives

$$\|\boldsymbol{r}_k\|_2 = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{r}_0\|_2, \quad (3.51)$$

and we clearly see that while GMRES minimizes the residual in the 2-norm, CG minimizes the residual in the A^{-1} norm.

The polynomial approximation problem (3.49) is crucial for the convergence analysis of GMRES: it allows one to obtain different convergence bounds.

The first bound is based on the spectrum of the matrix A, which is assumed to be diagonalizable, and on the condition number of the corresponding eigenvector matrix. This result is derived in Theorem 27 and leads to the famous convergence estimate given in Theorem 28, where the eigenvalues of A are assumed to lie in an ellipse that does not contain the origin.

A second class of convergence estimates is based on the so-called *numerical range* of the matrix A, see Definition 13, which leads to different convergence bounds that do not require A to be diagonalizable.

Finally, a recent third class of convergence estimates is based on the pseudo-spectrum of the matrix A; see, e.g., [64, page 57], [121] and [36].

Theorem 27 (Convergence bound for GMRES)**.** *Let $A \in \mathbb{R}^{n\times n}$ be diagonalizable, $A = S\Lambda S^{-1}$, with the diagonal matrix $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ and $\boldsymbol{f} \in \mathbb{R}^n$. Then at the kth step of the GMRES iteration, we have*

$$\frac{\|\boldsymbol{f} - A\boldsymbol{u}_k\|_2}{\|\boldsymbol{f}\|_2} \leq \kappa(S) \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \max_{\lambda\in\{\lambda_1,\ldots,\lambda_n\}} |p(\lambda)|,$$

where $\kappa(S) := \|S\|_2\|S^{-1}\|_2$ is the condition number of the matrix S.

Proof. The approximation problem (3.49) is

$$\begin{aligned}
\min_{\widetilde{\boldsymbol{u}}\in\mathcal{K}_k(A,\boldsymbol{f})} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 &= \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{f}\|_2 = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(S\Lambda S^{-1})\boldsymbol{f}\|_2 \\
&= \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|S\,p(\Lambda)\,S^{-1}\boldsymbol{f}\|_2 \\
&\leq \underbrace{\|S\|_2\|S^{-1}\|_2}_{\kappa(S)} \|\boldsymbol{f}\|_2 \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \max_{\lambda\in\{\lambda_1,\ldots,\lambda_n\}} |p(\lambda)|.
\end{aligned}$$

□

Since A is not assumed to be symmetric, its eigenvectors S and eigenvalues λ_j may be complex. Therefore, in order to estimate the convergence rate of GMRES, we need to find polynomials that are small on a given set of complex numbers. The following lemma will be helpful in finding such polynomials:

Lemma 14. *Let $C(0, r)$ denote the circle of radius r centered at the origin. Let $\gamma \in \mathbb{C}$ with $|\gamma| > r$. Then*

$$\min_{\substack{p\in\mathcal{P}_k \\ p(\gamma)=1}} \max_{z\in C(0,r)} |p(z)| = \left(\frac{r}{|\gamma|}\right)^k, \tag{3.52}$$

and the minimum is attained for $\widehat{p}(z) = \left(\frac{z}{\gamma}\right)^k$.

Proof. The polynomial $\widehat{p}(z) = \left(\frac{z}{\gamma}\right)^k$ satisfies $\widehat{p}(\gamma) = 1$. Moreover, since any $z \in C(0,r)$ is of the form $re^{i\vartheta}$, it holds that

$$|\widehat{p}(z)| = \left(\frac{r}{|\gamma|}\right)^k \quad \forall z \in C(0,r), \tag{3.53}$$

and hence

$$\max_{z \in C(0,r)} |\widehat{p}(z)| = \left(\frac{r}{|\gamma|}\right)^k. \tag{3.54}$$

Thus it is sufficient to show that no polynomial with the same normalization is smaller. To do so, consider any polynomial p_k of degree k and its normalized version,

$$g_k(z) = \frac{p_k(z)}{p_k(\gamma)}.$$

Seeking a contradiction, we assume that

$$|g_k(z)| = \left|\frac{p_k(z)}{p_k(\gamma)}\right| < |\widehat{p}(z)| = \left|\frac{r}{\gamma}\right|^k \qquad \forall z \in C(0,r) \tag{3.55}$$

holds, and notice that if we contradict (3.55), then recalling (3.53) and (3.54) we obtain that

$$|g_k(z)| \geq |\widehat{p}(z)| = \left(\frac{r}{|\gamma|}\right)^k = \max_{\widetilde{z} \in C(0,r)} |\widehat{p}(\widetilde{z})|,$$

for any $z \in C(0,r)$, which implies that $\widehat{p}$ solves (3.52).

By the *Rouché theorem*[6], see, e.g., [116, Chapter 3, Theorem 4.3] and [118, Theorem 3.42], if for two analytic functions f, g we have $|g(z)| < |f(z)|$ for all $z \in C(0,r)$, then f and $f-g$ have the same number of zeros inside $C(0,r)$. Now $\left(\frac{z}{\gamma}\right)^k$ has a zero of multiplicity k at 0, and the difference $(\frac{z}{\gamma})^k - \frac{p_k(z)}{p_k(\gamma)}$ vanishes at $z = \gamma$, which implies that

$$\left(\frac{z}{\gamma}\right)^k - \frac{p_k(z)}{p_k(\gamma)} = (z-\gamma)q_{k-1}(z),$$

with some polynomial q_{k-1} of degree lower than or equal to $k-1$ with at most $k-1$ zeros inside $C(0,r)$. Since γ is a root of $\left(\frac{z}{\gamma}\right)^k - \frac{p_k(z)}{p_k(\gamma)}$ outside of $C(0,r)$, we have a contradiction to the Rouché theorem, and therefore no such polynomial exists. □

[6]Rouché's theorem: Consider any two functions $f, g : \Omega \to \mathbb{C}$ that are analytic in a bounded, open, and simply connected domain $\Omega \subset \mathbb{C}$ whose boundary $\partial\Omega$ is a closed contour. If $|g(z)| < |f(z)|$ for any $z \in \partial\Omega$, then $f(z)$ and $f(z)+g(z)$ have the same number of zeros in Ω. Notice that the function g in this statement is the function $-g$ in our proof; in fact a direct inspection of the proof of the Rouché theorem reveals that the sign of g does not affect the result; see, e.g., [116, Chapter 3, Theorem 4.1 and Theorem 4.3]

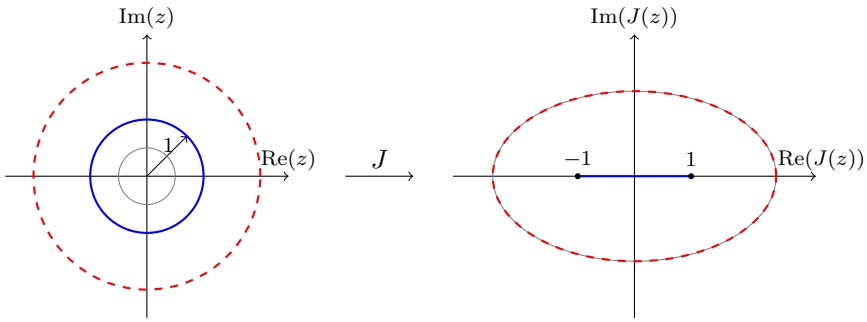


Figure 3.11: The Joukowski transformation J. Left: three circles of radius $r = 1$ (blue), $r = 2$ (red-dashed), and $r = \frac{1}{2}$ (gray). Right: the three ellipses obtained by the Joukowski transformation J applied to the three circles. The blue line is the (degenerate) ellipses that corresponds to the (blue) circle of radius $r = 1$. The two ellipses (gray and red-dashed) are obtained transforming the gray and dashed red circles (notice that these two ellipses coincide). The two points are the foci of the gray and red-dashed ellipses.

Note that with a change of variables $\widetilde{z} = z + c$, Lemma 14 can be applied to a circle centered at c. We have

$$\min_{\substack{p\in\mathcal{P}_k\\ p(\gamma)=1}} \max_{\widetilde{z}\in C(c,r)} |p(\widetilde{z})| = \min_{\substack{p\in\mathcal{P}_k\\ p(\gamma-c)=1}} \max_{z\in C(0,r)} |p(z)| = \left(\frac{r}{|\gamma - c|}\right)^k,$$

where the second equality follows by Lemma 14.

As a next ingredient in our search for small polynomials, we will need the conformal mapping

$$J(w) = \frac{w + w^{-1}}{2},$$

which is called the *Joukowski transformation*. It maps the circle $C(0, r)$ defined by $w = re^{i\vartheta}$ to

$$\begin{aligned} z &= \frac{1}{2}\left(re^{i\vartheta} + \frac{1}{r}e^{-i\vartheta}\right) = \frac{1}{2}\left(r\cos\vartheta + ir\sin\vartheta + \frac{1}{r}\cos\vartheta - \frac{i}{r}\sin\vartheta\right) \\ &= \frac{1}{2}\left(\left(r + \frac{1}{r}\right)\cos\vartheta + i\left(r - \frac{1}{r}\right)\sin\vartheta\right), \end{aligned}$$

an ellipse with foci at -1 and 1 and principal axes $a = \left(r + \frac{1}{r}\right)/2$ and $b = \left(r - \frac{1}{r}\right)/2$. Without loss of generality, we may assume $r > 1$, since r and $1/r$ produce the same ellipse. For $r = 1$ the ellipse is degenerate: the circle is mapped to the real interval $[-1, 1]$. See Figure 3.11.

The Joukowski transformation can be used for expressing the Chebyshev polynomials defined by the recursion (3.31), as we have already seen in Lemma 11, which led to the representation formula (3.35) for the Chebyshev polynomials.

Lemma 15. *Let E_J denote the ellipse obtained from the circle $C(0,r)$ using the Joukowski transformation $J(w)$, and let γ be a point outside E_J. Let w_γ be the solution with larger modulus of $J(w)=\gamma$. Then*

$$\frac{r^k}{|w_\gamma|^k} \le \min_{\substack{p\in\mathcal{P}_k\\ p(\gamma)=1}} \max_{z\in E_J} |p(z)| \le \frac{r^k + r^{-k}}{|w_\gamma^k + w_\gamma^{-k}|}. \tag{3.56}$$

Proof. Every polynomial $p \in \mathcal{P}_k$ with $p(\gamma)=1$ can be written as

$$p(z) = \frac{\sum_{j=0}^k \zeta_j z^j}{\sum_{j=0}^k \zeta_j \gamma^j}.$$

Using the Joukowski transformation $z = J(w)$, this becomes

$$p(z) = \frac{\sum_{j=0}^k \widetilde{\zeta}_j (w^j + w^{-j})}{\sum_{j=0}^k \widetilde{\zeta}_j (w_\gamma^j + w_\gamma^{-j})}. \tag{3.57}$$

In particular, for $\widetilde{\zeta}_k = 1$ and $\widetilde{\zeta}_j = 0$, $j = 0, 1, \ldots, k-1$, we obtain a normalized Chebyshev polynomial,

$$p^*(z) = \frac{w^k + w^{-k}}{w_\gamma^k + w_\gamma^{-k}} = \frac{T_k(z)}{T_k(\gamma)}.$$

Letting $w = re^{i\vartheta}$, we get

$$\left| r^k e^{ik\vartheta} + \frac{1}{r^k} e^{-ik\vartheta} \right| \le \left| r^k e^{ik\vartheta} \right| + \left| \frac{1}{r^k} e^{-ik\vartheta} \right| = r^k + \frac{1}{r^k},$$

and the maximum of $p^*(z)$ on $J(C(0,r)) = E_J$ is attained for $\vartheta = 0$. Therefore, we have found that

$$\max_{z\in E_J} |p^*(z)| = \frac{r^k + r^{-k}}{|w_\gamma^k + w_\gamma^{-k}|},$$

which implies

$$\min_{\substack{p\in\mathcal{P}_k\\ p(\gamma)=1}} \max_{z\in E_J} |p(z)| \le \max_{z\in E_J} |p^*(z)| = \frac{r^k + r^{-k}}{|w_\gamma^k + w_\gamma^{-k}|}.$$

Thus the upper bound in (3.56) is proved.

In order to prove the lower bound, we rewrite (3.57) as

$$p(z) = \frac{w^{-k}}{w_\gamma^{-k}} \frac{\sum_{j=0}^k \widetilde{\zeta}_j (w^{k+j} + w^{k-j})}{\sum_{j=0}^k \widetilde{\zeta}_j (w_\gamma^{k+j} + w_\gamma^{k-j})}.$$

The absolute value becomes

$$|p(z)| = \frac{r^{-k}}{|w_\gamma|^{-k}} |q_{2k}(w)|,$$

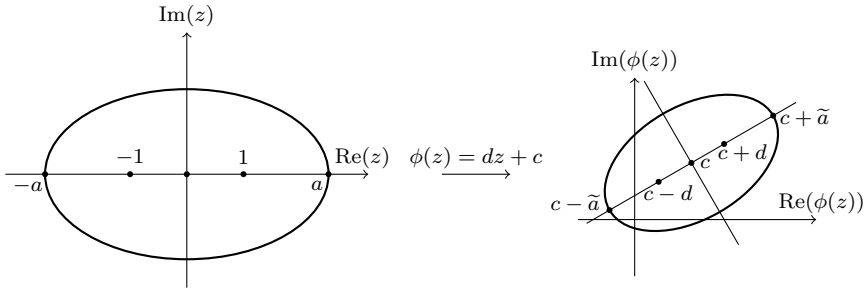


Figure 3.12: Joukowski ellipse $E_J = J(C(0,r))$ (left) and its transformation $\phi(E_J)$ (right).

where q_{2k} is a polynomial of degree lower than or equal to $2k$ with $q_{2k}(w_\gamma) = 1$. By Lemma 14, we have

$$|q_{2k}(w)| \geq \frac{r^{2k}}{|w_\gamma|^{2k}} \quad \Longrightarrow \quad \max_{z \in E_J} |p(z)| \geq \frac{r^{-k}}{|w_\gamma|^{-k}} \frac{r^{2k}}{|w_\gamma|^{2k}} = \frac{r^k}{|w_\gamma|^k},$$

which is the lower bound. □

Corollary 3. *The upper bound estimate* (3.56) *in Lemma 15 also holds in the interior of the ellipse.*

Proof. This follows directly from an application of the maximum principle for the modulus of analytic (holomorphic) functions (see, e.g., [118, Chapter 5] and [63, Chapter 6]), which says that functions that are analytic in a bounded and simply-connected domain $\Omega \subset \mathbb{C}$ attain their maximum in modulus on the boundary $\partial\Omega$. □

Since the difference between the two expressions

$$\frac{r^k}{|w_\gamma|^k} \quad \text{and} \quad \frac{r^k + r^{-k}}{|w_\gamma^k + w_\gamma^{-k}|}$$

converges to zero for $k \to \infty$, we conclude that for large k the complex Chebyshev polynomial

$$p^*(z) = \frac{w^k + w^{-k}}{w_\gamma^k + w_\gamma^{-k}} = \frac{T_k(z)}{T_k(\gamma)}$$

is optimal. But $p^*(z)$ is only optimal asymptotically for k large, which is different from the real case, where the Chebyshev polynomials are optimal for all k.

Lemma 15 can be generalized to any ellipse in the complex plane. To do so, consider the transformation $z \mapsto \phi(z) := dz + c$. Notice that c, $\mathrm{Re}(d)$, $\mathrm{Im}(d)$ are translation, scaling, and rotation parameters and we define $\widetilde{a} = ad$. So we denote by $E(c, d, \widetilde{a})$ the ellipse $\phi(E_J)$ (see Figure 3.12), where c, $c+d$, and $c+\widetilde{a}$ are the positions of the center, and the focal point and the extrema shown in Figure

3.12. Notice that $\frac{\widetilde{a}}{d} = \frac{ad}{d} = a > 1$. As before, we denote by $\mathrm{int}E(c,d,\widetilde{a})$ the set of points inside $E(c,d,\widetilde{a})$, and we define $\overline{E(c,d,\widetilde{a})} := \mathrm{int}E(c,d,\widetilde{a}) \cup E(c,d,\widetilde{a})$. Now, recalling the proof of Lemma 15, we have that (with $\widetilde{z} = \phi(z)$)

$$\min_{\substack{p\in\mathcal{P}_k \\ p\left(\frac{\gamma-c}{d}\right)=1}} \max_{z\in E_J} |p(z)| = \min_{\substack{p\in\mathcal{P}_k \\ p(\gamma)=1}} \max_{\widetilde{z}\in\overline{E(c,d,\widetilde{a})}} |p(\widetilde{z})|$$

$$\leq \max_{\widetilde{z}\in\overline{E(c,d,\widetilde{a})}} |p^*(\widetilde{z})| = \max_{\widetilde{z}\in\overline{E(c,d,\widetilde{a})}} |\widehat{T}_k(\widetilde{z})|,$$

where $\widehat{T}_k(\widetilde{z}) = \frac{T_k\left(\frac{\widetilde{z}-c}{d}\right)}{T_k\left(\frac{\gamma-c}{d}\right)}$. The maximum is attained at $\widetilde{z} = c + \widetilde{a}$, as we have seen in the proof of Lemma 15, and hence

$$\max_{\widetilde{z}\in\overline{E(c,d,\widetilde{a})}} |\widehat{T}_k(\widetilde{z})| = \left| \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{T_k\left(\frac{\gamma-c}{d}\right)} \right| = \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{|T_k\left(\frac{\gamma-c}{d}\right)|},$$

where we removed the modulus in the numerator, since $\frac{\widetilde{a}}{d} > 1$, which means $T_k(\frac{\widetilde{a}}{d}) > 0$.

Theorem 28 (Convergence estimate for GMRES)**.** *Let $A \in \mathbb{R}^{n\times n}$ be diagonalizable, $A = S\Lambda S^{-1}$ with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and assume that all $\lambda_j \in \overline{E(c,d,\widetilde{a})}$ and that the origin is not contained in the ellipse. Then at the kth iteration of GMRES, we have*

$$\frac{\|\boldsymbol{f} - A\boldsymbol{u}_k\|_2}{\|\boldsymbol{f}\|_2} \leq \kappa(S) \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{|T_k\left(\frac{c}{d}\right)|}.$$

Proof. From Theorem 27 we know that

$$\frac{\|\boldsymbol{f} - A\boldsymbol{u}_k\|_2}{\|\boldsymbol{f}\|_2} \leq \kappa(S) \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \max_i |p(\lambda_i)|.$$

Now, we recall that $\{\lambda_1, \ldots, \lambda_n\} \subset \overline{E(c,d,\widetilde{a})}$. Then Lemma 15, the change of variables, and setting $\gamma = 0$ show that

$$\min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \max_{\widetilde{z}\in\overline{E(c,d,\widetilde{a})}} |p(\widetilde{z})| \leq \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{|T_k\left(-\frac{c}{d}\right)|} = \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{|T_k\left(\frac{c}{d}\right)|},$$

since $|T_k(-z)| = |T_k(z)|$. □

If $a, c, d \in \mathbb{R}$, then for $c > \widetilde{a} > d$, we get

$$\begin{aligned} \frac{T_k\left(\frac{\widetilde{a}}{d}\right)}{|T_k\left(\frac{c}{d}\right)|} &= \frac{\left(\frac{\widetilde{a}}{d} + \sqrt{\left(\frac{\widetilde{a}}{d}\right)^2 - 1}\right)^k + \left(\frac{\widetilde{a}}{d} + \sqrt{\left(\frac{\widetilde{a}}{d}\right)^2 - 1}\right)^{-k}}{\left|\left(\frac{c}{d} + \sqrt{\left(\frac{c}{d}\right)^2 - 1}\right)^k + \left(\frac{c}{d} + \sqrt{\left(\frac{c}{d}\right)^2 - 1}\right)^{-k}\right|} \\ &\approx \left(\frac{\frac{\widetilde{a}}{d} + \sqrt{\left(\frac{\widetilde{a}}{d}\right)^2 - 1}}{\frac{c}{d} + \sqrt{\left(\frac{c}{d}\right)^2 - 1}}\right)^k = \left(\frac{\widetilde{a} + \sqrt{\widetilde{a}^2 - d^2}}{c + \sqrt{c^2 - d^2}}\right)^k, \end{aligned} \tag{3.58}$$

for k sufficiently large.

If the spectrum of GMRES does not lie in an ellipse excluding the origin, or if the matrix is not diagonalizable, different techniques need to be used to study the convergence of GMRES, and this is still an active area of research, see, e.g., the monograph [85, Section 5.7] and references therein. If the matrix A is nonnormal, but its eigenvector matrix S is well-conditioned, then the convergence bounds obtained in Theorems 27 and 28 give reasonable estimates of the norm of the residual, even if they are not necessarily sharp. The distribution of the eigenvalues of the matrix A is then essentially sufficient to describe the GMRES behavior [64]. In general, however, the behavior of GMRES can not be determined from the eigenvalues alone. To overcome this problem, the numerical range of the matrix A is important.

Definition 13 (Numerical range and numerical radius)**.** *Consider a matrix $A \in \mathbb{C}^{n\times n}$. The set*

$$\mathcal{F}(A) := \{\boldsymbol{y}^* A\boldsymbol{y} \,:\, \boldsymbol{y} \in \mathbb{C}^n, \boldsymbol{y}^*\boldsymbol{y} = 1\}$$

is called numerical range *of A. The value*

$$\nu(A) := \max\{|z| \,:\, z \in \mathcal{F}(A)\}$$

is called numerical radius *of A.*

In the literature, the numerical range is also often called the *field of values*, *range of values*, *Hausdorff domain*, and *Wertvorrat*; see, e.g., [102]. Some of the main properties of the numerical range and the numerical radius are summarized in the following theorem. Other properties can be found in, e.g., [74, 71, 102].

Theorem 29 (Properties of $\mathcal{F}(A)$ and $\nu(A)$)**.** *For any $A, B \in \mathbb{C}^{n\times n}$ and any $\alpha, \beta \in \mathbb{C}$ it holds:*

(a) $\mathcal{F}(A)$ is a compact subset of $\mathbb{C}$.

(b) $\mathcal{F}(\alpha A + \beta I) = \alpha\mathcal{F}(A) + \beta$ (I is the identity matrix).

(c) $\mathcal{F}(A)$ is convex.

(d) The spectrum of A is contained in $\mathcal{F}(A)$ and $\rho(A) \le \nu(A)$.

(e) $\nu(A + B) \le \nu(A) + \nu(B)$.

(f) $\nu(\alpha A) = |\alpha|\nu(A)$.

(g) $\frac{1}{2}\|A\|_2 \le \nu(A) \le \|A\|_2$.

(h) $\nu(A^m) \le [\nu(A)]^m$ for $m = 1, 2, \ldots$

Proof. The property (a) clearly holds, since $\mathcal{F}(A)$ is the image of the unit sphere (a compact set) under the continuous map $\boldsymbol{y} \mapsto \boldsymbol{y}^* A\boldsymbol{y}$. The statement (c) is the famous Toeplitz-Hausdorff theorem, we refer to [102] for a proof. The proofs of

the statements (b), (d), (e), and (f) are not difficult and just use the definitions of $\mathcal{F}(A)$ and $\nu(A)$; they are left as an exercises to the reader, see Problem 33. More interesting are the proofs of (g) and (h), which we now give, adapted from [64] and [98].

Let us begin with (g). For any $\boldsymbol{y} \in \mathbb{C}^n$ such that $\|\boldsymbol{y}\|_2 = 1$, we use the Cauchy-Schwarz inequality to write

$$|\boldsymbol{y}^* A \boldsymbol{y}| \leq \|\boldsymbol{y}\|_2 \|A\boldsymbol{y}\|_2 \leq \|A\|_2,$$

since $\boldsymbol{y}$ is normalized, which implies the right inequality of (g), that is $\nu(A) \leq \|A\|_2$. To prove the left inequality, we use the standard decomposition $A = S + H$, where $H = \frac{1}{2}(A + A^*)$ is the Hermitian part of A and $S = \frac{1}{2}(A - A^*)$ is the skew-Hermitian part of A. Since H and S are normal matrices, one can show that $\rho(H) = \|H\|_2 = \nu(H)$ and $\rho(S) = \|S\|_2 = \nu(S)$ (see Problem 34). Using these relations, the triangle inequality, and the fact that $\nu(A) = \nu(A^*)$, we obtain

$$\begin{aligned}
\|A\|_2 &\leq \|H\|_2 + \|S\|_2 = \nu(H) + \nu(S) \\
&= \frac{1}{2}\Big[\max_{\|\boldsymbol{y}\|_2^2=1} |\boldsymbol{y}^*(A + A^*)\boldsymbol{y}| + \max_{\|\boldsymbol{y}\|_2^2=1} |\boldsymbol{y}^*(A - A^*)\boldsymbol{y}|\Big] \\
&\leq \frac{1}{2}\Big[2 \max_{\|\boldsymbol{y}\|_2^2=1} |\boldsymbol{y}^* A \boldsymbol{y}| + 2 \max_{\|\boldsymbol{y}\|_2^2=1} |\boldsymbol{y}^* A^* \boldsymbol{y}|\Big] \\
&\leq \nu(A),
\end{aligned}$$

which implies $\|A\|_2 \leq \nu(A)$.

Let us now prove (h). To do so, we make use of the two polynomial identities

$$1 - z^m = \prod_{k=1}^{m} (1 - w^k z) \tag{3.59}$$

and

$$1 = \sum_{k=1}^{m} \frac{1}{m} \prod_{\ell=1, \ell \neq k}^{m} (1 - w^\ell z), \tag{3.60}$$

where $w = \exp(\frac{2\pi i}{m})$, i is the imaginary unit, and m is an arbitrary positive integer. The polynomial identities (3.59) and (3.60) are proved in Theorem 45 in the Appendix. Now, consider any matrix $C \in \mathbb{C}^{n \times n}$ and any vector $\boldsymbol{x} \in \mathbb{C}^n$ such that $\|\boldsymbol{x}\|_2 = 1$, and define the vectors

$$\boldsymbol{x}_k := \prod_{\ell=1, \ell \neq k}^{m} [I - w^\ell C]\boldsymbol{x}, \tag{3.61}$$

for $k = 1, \ldots, m$. Using the notation $\langle \boldsymbol{y}, \boldsymbol{x} \rangle := \boldsymbol{x}^* \boldsymbol{y}$, we write

$$\begin{aligned}
1 - \langle C^m \boldsymbol{x}, \boldsymbol{x} \rangle &= \langle [I - C^m] \boldsymbol{x}, \boldsymbol{x} \rangle \\
&= \langle [I - C^m] \boldsymbol{x}, \sum_{k=1}^{m} \frac{1}{m} \prod_{\ell=1, \ell \neq k}^{m} (1 - w^\ell C) \boldsymbol{x} \rangle \qquad \text{(using (3.60))} \\
&= \frac{1}{m} \sum_{k=1}^{m} \langle [I - C^m] \boldsymbol{x}, \boldsymbol{x}_k \rangle \qquad \text{(using (3.61))} \\
&= \frac{1}{m} \sum_{k=1}^{m} \langle \prod_{\ell=1}^{m} (I - w^\ell C) \boldsymbol{x}, \boldsymbol{x}_k \rangle \qquad \text{(using (3.59))} \\
&= \frac{1}{m} \sum_{k=1}^{m} \langle (I - w^k C) \boldsymbol{x}_k, \boldsymbol{x}_k \rangle \qquad \text{(using (3.61))} \\
&= \frac{1}{m} \sum_{k=1}^{m} \|\boldsymbol{x}_k\|_2^2 \Big[1 - w^k \langle C \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2}, \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2} \rangle \Big],
\end{aligned}$$

which gives us the relation

$$1 - \langle C^m \boldsymbol{x}, \boldsymbol{x} \rangle = \frac{1}{m} \sum_{k=1}^{m} \|\boldsymbol{x}_k\|_2^2 \Big[1 - w^k \langle C \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2}, \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2} \rangle \Big], \tag{3.62}$$

for any normalized $\boldsymbol{x}$ and any matrix C. If we choose $C = \exp(i\vartheta)B$, for an arbitrary matrix $B \in \mathbb{C}^{n \times n}$ and $\vartheta \in \mathbb{R}$, and replace it into (3.62), we get

$$1 - \exp(im\vartheta) \langle B^m \boldsymbol{x}, \boldsymbol{x} \rangle = \frac{1}{m} \sum_{k=1}^{m} \|\boldsymbol{x}_k\|_2^2 \Big[1 - w^k \exp(i\vartheta) \langle B \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2}, \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2} \rangle \Big]. \tag{3.63}$$

If $\nu(B) \leq 1$ holds, then the term $1 - w^k \exp(i\vartheta) \langle B \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2}, \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2} \rangle$ has non-negative real part for any $\vartheta \in \mathbb{R}$:

$$\mathrm{Re} \Big[1 - w^k \exp(i\vartheta) \langle B \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2}, \frac{\boldsymbol{x}_k}{\|\boldsymbol{x}_k\|_2} \rangle \Big] \geq 0 \quad \forall \vartheta \in \mathbb{R},$$

for $k = 1, \ldots, m$. Therefore, the left-hand side of (3.63) has also non-negative real part. This is implies that

$$\mathrm{Re} \Big[\exp(im\vartheta) \langle B^m \boldsymbol{x}, \boldsymbol{x} \rangle \Big] \leq 1 \quad \forall \vartheta \in \mathbb{R},$$

which then gives us the relation $|\langle B^m \boldsymbol{x}, \boldsymbol{x} \rangle| \leq 1$, for any normalized vector $\boldsymbol{x}$.[7] Therefore, we obtain

$$\nu(B^m) \leq 1. \tag{3.64}$$

Hence, if $\nu(B) \leq 1$ holds, then $\nu(B^m) \leq 1$.

[7] To see this, one can write $1 \geq \mathrm{Re}\big(e^{im\vartheta} \langle B^m \boldsymbol{x}, \boldsymbol{x} \rangle\big) = \mathrm{Re}\big(e^{i(m\vartheta + \delta)} |\langle B^m \boldsymbol{x}, \boldsymbol{x} \rangle|\big)$ and then choose $\theta = -\delta/m$.

Now, let us consider any non zero matrix $A \in \mathbb{C}^{n\times n}$. We define $B := \frac{1}{\nu(A)}A$ and, using (f), compute

$$\nu(B) = \nu\Big(\frac{1}{\nu(A)}A\Big) = \frac{1}{\nu(A)}\nu(A) = 1.$$

Therefore (3.64) holds and we get

$$1 \geq \nu(B^m) = \nu\Big(\frac{1}{\nu(A)^m}A^m\Big) = \frac{1}{\nu(A)^m}\nu(A^m),$$

which implies that $\nu(A)^m \geq \nu(A^m)$. In the special case $A = 0$, we obviously have that $\nu(A) = 0$ and $\nu(A^m) = 0$. Hence the result follows. □

We can now prove the following convergence estimate. Recall that we assumed in our analysis that $\boldsymbol{u}_0 = 0$, i.e. $\boldsymbol{r}_0 = \boldsymbol{f}$.

Theorem 30 (Convergence bound for GMRES)**.** *Consider a matrix $A \in \mathbb{R}^{n\times n}$, a vector $\boldsymbol{f} \in \mathbb{R}^n$ and an initial guess $\boldsymbol{u}_0 = 0$. Suppose that the numerical range of A is contained in a disk of radius $s \in \mathbb{R}$ and center $c \in \mathbb{C}$ that does not contain the origin (hence $s/|c| < 1$):*

$$0 \notin \{z \in \mathbb{C} : |z - c| \leq s\} \supset \mathcal{F}(A).$$

Then the residual $\boldsymbol{r}_k$ of GMRES is bounded for any k by

$$\|\boldsymbol{r}_k\|_2 \leq 2\Big(\frac{s}{|c|}\Big)^k \|\boldsymbol{f}\|_2. \tag{3.65}$$

Proof. Using (3.49) we get

$$\|\boldsymbol{r}_k\|_2 = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{f}\|_2 \leq \|\widetilde{p}_k(A)\boldsymbol{f}\|_2, \tag{3.66}$$

with $\widetilde{p}_k(z) = (1 - z/c)^k$. Theorem 29 allows us to obtain

$$\begin{aligned}
\|\widetilde{p}_k(A)\boldsymbol{f}\|_2 \leq \|\widetilde{p}_k(A)\|_2\|\boldsymbol{f}\|_2 &= \Big\|\Big(I - \frac{1}{c}A\Big)^k\Big\|_2\|\boldsymbol{f}\|_2 \\
&\leq 2\nu\Big(\Big(I - \frac{1}{c}A\Big)^k\Big)\|\boldsymbol{f}\|_2 && \text{(by Theorem 29 (g))} \\
&\leq 2\Big[\nu\Big(I - \frac{1}{c}A\Big)\Big]^k\|\boldsymbol{f}\|_2. && \text{(by Theorem 29 (h))}
\end{aligned}$$

Moreover, using Theorem 29 (b) we get[8]

$$\mathcal{F}\Big(I - \frac{1}{c}A\Big) = 1 - \frac{1}{c}\mathcal{F}(A) \subset \Big\{z \in \mathbb{C} : |z| \leq \frac{s}{|c|}\Big\},$$

which implies that $\nu\Big(I - \frac{1}{c}A\Big) \leq \frac{s}{|c|}$. Hence the claim follows. □

[8]Notice that $1 - \frac{1}{c}\mathcal{F}(A)$ is a simple transformation that scales, translate and rotate $\mathcal{F}(A)$. If one applies the same transformation to the circle containing $\mathcal{F}(A)$, then this circle is mapped into another circle of radius $s/|c|$, centered in the origin, and containing $1 - \frac{1}{c}\mathcal{F}(A)$.

The result obtained in Theorem 30 is quantitative and requires the numerical range to lie in a disk bounded away from the origin. However, by exploiting further properties of the numerical range it is possible to obtain a general residual bound for GMRES. This is a very intriguing research area, which is still very active and whose charm is heightened by the famous *Crouzeix's conjecture*. This fascinating story began with the work [33] of Bernard and Francois Delyon, who proved in 1999 that, given a smooth, bounded and convex domain $\Omega \subset \mathbb{C}$ that contains the (closure of the) numerical range $\mathcal{F}(A)$, there exists a best constant $C_\Omega > 0$ such that

$$\|f(A)\|_2 \leq C_\Omega \sup_{z\in\Omega} |f(z)| \tag{3.67}$$

for any rational function f[9]. This work inspired *Michel Crouzeix*, who in 2004 stated the following conjecture [29].

Conjecture 1 (Crouzeix's conjecture, 2004)**.** *Let* $\mathcal{Q} := \sup_\Omega C_\Omega$*, then* $\mathcal{Q} = 2$.

This conjecture is still unsolved. Crouzeix proved a few years later in [30] that $2 \leq \mathcal{Q} \leq 11.81$. Very recently, in 2017 in the manuscript [32], Crouzeix and Palencia strongly improved his bound to $2 \leq \mathcal{Q} \leq 1 + \sqrt{2}$, which is very close to the conjectured bound! This result has been generalized in 2019 by Crouzeix and Greenbaum for more general regions in the complex plane [31].

Moreover, it is known that the conjecture holds for tridiagonal 3×3 matrices with elliptic field of values centered at an eigenvalue [60] and for general $n \times n$ matrices that are nearly Jordan blocks [20]. Furthermore, Greenbaum and Overton provided numerical support for Crouzeix's conjecture [65]. Despite all these very recent improvements, the conjecture is still unsolved in the general case.

Let us return to the convergence bound for GMRES. As shown in [32] (and recalling that $\mathcal{F}(A)$ is compact, see Theorem 29), it holds that

$$\|f(A)\|_2 \leq \widetilde{\mathcal{Q}} \max_{z\in\mathcal{F}(A)} |f(z)|, \tag{3.68}$$

where $\widetilde{\mathcal{Q}} = 1+\sqrt{2}$ (and $\widetilde{\mathcal{Q}} = 2$ if Crouzeix's conjecture holds), using the relation (3.49) one can get the famous convergence bound

$$\|\boldsymbol{r}_k\|_2 \leq \widetilde{\mathcal{Q}} \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \max_{z\in\mathcal{F}(A)} |p(z)|\, \|\boldsymbol{f}\|_2,$$

i.e. one has to find residual polynomials that are small on the numerical range of A.

Let us now briefly discuss a third approach that permits to get a convergence bound for GMRES using the notion of the ϵ-pseudospectrum of A. This bound is based on an idea presented in [121], see also [64, page 57] and [36] for more details. The ϵ-pseudospectrum of A is defined as

$$\Lambda_\epsilon := \{z \in \mathbb{C} : \|(zI - A)^{-1}\|_2 \geq \epsilon^{-1}\},$$

[9] Notice that this result is proved in a general Hilbert space setting [33].

where $\epsilon < 0$, and we denote by Γ_ϵ the boundary of Λ_ϵ. For any polynomial p we can use the Cauchy integral to write

$$p(A) = \frac{1}{2\pi i} \int_\Gamma p(z)(zI - A)^{-1} dz,$$

where Γ is any simple closed curve (or the union of simple closed curves) in $\mathbb{C}$ containing the spectrum of A. We denote by $L(\Gamma)$ the length of this curve. If one chooses $\Gamma = \Gamma_\epsilon$ (which clearly contains the spectrum of A) and uses the previous equality, we can obtain the bound

$$\|p(A)\|_2 \leq \frac{L(\Gamma_\epsilon)}{2\pi} \max_{z \in \Gamma_\epsilon} \|p(z)(zI - A)^{-1}\|_2 \leq \frac{L(\Gamma_\epsilon)}{2\pi\epsilon} \max_{z \in \Gamma_\epsilon} |p(z)|. \tag{3.69}$$

Now, using the bound (3.49) and the estimate (3.69) we obtain

$$\|\boldsymbol{r}_k\|_2 \leq \frac{L(\Gamma_\epsilon)}{2\pi\epsilon} \max_{z \in \Gamma_\epsilon} |p(z)| \|\boldsymbol{f}\|_2 \tag{3.70}$$

The accuracy of the estimate (3.5) strongly depends on the choice of the parameter ϵ, which is unfortunately often not so easy in practice.

In general, as we have seen, the convergence analysis of GMRES is not an easy task. Any of the bounds obtained above are in some cases far from being sharp; see, e.g., [64, 36]. Greenbaum, Pták and Strakoš [66, 67] have shown that any nonincreasing curve represents a plot of residual norm versus iteration number for GMRES applied to some problem, and this for an arbitrary set of chosen eigenvalues. Hence, for example, eigenvalues tightly clustered around 1 are not necessarily leading to good convergence of Krylov methods when the matrix is highly nonnormal. To see this, consider a matrix A of the form [64]

$$A = \begin{bmatrix} 0 & \times & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \times & \times & \cdots & \times & 0 \\ 0 & \times & \times & \cdots & \times & \times \\ \times & \times & \times & \cdots & \times & \times \end{bmatrix}. \tag{3.71}$$

Recalling our assumption $\boldsymbol{u}_0 = 0$, if $\boldsymbol{r}_0 = \boldsymbol{f}$ is a multiple of the first canonical vector $\boldsymbol{e}_1$, then $A\boldsymbol{f}$ is a multiple of $\boldsymbol{e}_n$, $A^2\boldsymbol{f}$ is a linear combination of $\boldsymbol{e}_n$ and $\boldsymbol{e}_{n-1}$, $A^3\boldsymbol{f}$ is a linear combination of $\boldsymbol{e}_n$, $\boldsymbol{e}_{n-1}$ and $\boldsymbol{e}_{n-2}$, etc[10]. This means that $\boldsymbol{r}_0 = \boldsymbol{f}$ is orthogonal to the set $\text{span}\{A\boldsymbol{f}, \ldots, A^{n-1}\boldsymbol{f}\}$, and hence

$$\boldsymbol{u}_k = \underset{\widetilde{\boldsymbol{u}} \in \mathcal{K}_k(A, \boldsymbol{f})}{\arg\min} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 = 0,$$

[10]This information propagation mechanism is similar to the one we observed in all the methods we tested so far for Laplace's equation: because of the local connectivity only in the discrete Laplace matrix, the approximation computed by all the methods remained zero in the part of the domain where y is small, both for stationary and Krylov methods. The main goal of preconditioners in Chapter 4 is to change this!

for $k = 1, \ldots, n-1$. This means that GMRES makes no progress until step n! Now, the class of matrices of the form (3.71) includes all $n \times n$ companion matrices

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ c_0 & c_1 & c_2 & \cdots & c_{n-2} & c_{n-1} \end{bmatrix},$$

whose eigenvalues are the roots of the (characteristic and minimal) polynomial $p(z) = z^n - \sum_{j=0}^{n-1} c_j z^j$, and the coefficients $c_0, \ldots, c_{n-1}$ can be chosen to make this matrix have any desired eigenvalues! Let us consider two examples.

Example 10 (Companion matrix with eigenvalues equal to 1)**.** *Consider the polynomial $p(z) = (z-1)^3$ and the corresponding companion matrix*

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -3 & 3 \end{bmatrix},$$

which has all eigenvalues equal to 1. However, if we consider $\boldsymbol{u}_0 = 0$ and $\boldsymbol{f} = \boldsymbol{e}_1$, the first canonical vector in $\mathbb{R}^3$, then GMRES will produce approximations $\boldsymbol{u}_k$ such that $\|\boldsymbol{r}_k\|_2 = 1$ for $k = 0, 1, 2$, and only at the third iteration one gets $\|\boldsymbol{r}_3\|_2 = 0$, as illustrated by the MATLAB *script*

```
A=[0 1 0; 0 0 1; 1 -3 3];              % companion matrix, minimal polynomial
                                       % is p(z)=(z-1)^3=z^3-3z^2+3z-1
f=zeros(3,1); f(1)=1;                  % right-hand side vector equal to e1
u0=zeros(3,1);                         % initial guess for gmres
[u,~,~,~,resvec]=gmres(A,f,[],1e-1,3,[],[],u0);
plot(0:3,resvec,'o-b','Linewidth',2); % plot gmres residuals
xlabel('iterations'); ylabel('residual'); grid on;
```

which produces the graph shown in Figure 3.13 (left). This example clearly shows that, even though all the eigenvalues of A are equal to 1, the GMRES residual remains constant for the first two iterations.

Example 11 (Companion matrix with random eigenvalues)**.** *In this example, we construct a companion matrix A in $\mathbb{R}^{n \times n}$ corresponding to a minimal polynomial with randomly chosen roots, and then solve the problem $A\boldsymbol{u} = \boldsymbol{f}$, with $\boldsymbol{f} = \boldsymbol{e}_1$, the first canonical vector in $\mathbb{R}^n$ starting from an initial guess $\boldsymbol{u}_0 = 0$. This is done by the* MATLAB *script*

```
n=10;                                  % dimension of the problem
rootp=1-2*rand(n,1);                   % generate a random vector of roots
coeff=poly(rootp);                     % find the corresponding coefficients
coeff=coeff./coeff(1);                 % make the polynomial monic
coeff=fliplr(coeff);                   % flip the vector of coefficients
```

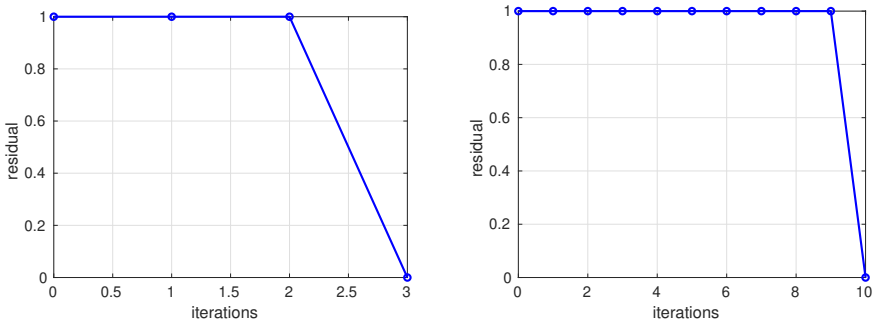


Figure 3.13: GMRES iterations corresponding to Example 10 (left) and Example 11 (right).

```
A=diag(ones(n-1,1),1);                  % construct companion matrix
A(end,:)=-coeff(1:end-1);
u0=zeros(n,1);                          % initial guess for gmres
f=zeros(n,1); f(1)=1;                   % right-hand side vector equal to e1
[u,~,~,~,resvec]=gmres(A,f,[],1e-1,n,[],[],u0);
plot(0:n,resvec,'o-b','Linewidth',2);   % plot gmres residuals
xlabel('iterations'); ylabel('residual'); grid on;
```

which produces the graph shown in Figure 3.13 (right). It is clear from this graph that the GMRES residuals are equal to 1 *for all* $k = 1, \ldots, n-1$ *and only at the last iteration the residual drops to zero.*

These examples are not in contradiction with the convergence estimate for GMRES in Theorem 28, since the condition number $\kappa(S)$ of the corresponding eigenvector matrix that appears in the estimate is quite large here (just try `[S,E]=eig(A); cond(S)`), so the theoretical estimate is not useful anymore. One probably would not use GMRES to solve a linear system with the sparsity pattern in (3.71), but the same holds for any matrix that is unitarily similar to one of the form (3.71). Note also that (3.71) is simply a permuted triangular matrix. Every matrix is similar to a lower triangular matrix (recall the Schur decomposition), but fortunately most matrices are not unitarily similar to one of the form (3.71).

In exact arithmetic, we can make another statement about the convergence of GMRES. To do so, we make use of the minimal polynomial introduced in Definition 11 and characterized in Lemma 6. We have

Theorem 31 (Convergence of GMRES in a finite number of iterations)**.** *Let* $A \in \mathbb{R}^{n\times n}$ *be invertible,* $\boldsymbol{f} \in \mathbb{R}^n$*, and assume that the minimal polynomial* $p_{\min}$ *of* A *has degree* m*. Then GMRES applied to the linear system* $A\boldsymbol{u} = \boldsymbol{f}$ *converges to the exact solution in at most* m *iterations*[11]*.*

[11]Notice that the assumption that $p_{\min}$ has degree m is satisfied for a matrix A that is diagonalizable and that has m distinct eigenvalues, see Lemma 6.

Proof. Since A is invertible, the minimal polynomial $p_{\min}(A)$ has the constant coefficient $\alpha_0 \neq 0$. Thus the polynomial

$$p^*(A) = \frac{1}{\alpha_0} p_{\min}(A)$$

satisfies $p^*(0) = 1$ and $p^*(A) = 0$. GMRES minimizes the residual, which is equivalent to the polynomial approximation problem

$$\min_{\boldsymbol{u}_k \in \mathcal{K}_k(A,\boldsymbol{f})} \|\boldsymbol{f} - A\boldsymbol{u}_k\|_2 = \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{f}\|_2 \leq \|p^*(A)\boldsymbol{f}\|_2 = 0, \quad \text{when } k = m.$$

□

```
function [u,uk,res]=GMRES(A,f,u0,tol,m)
% GMRES Generalized Minimal Residual
%   [u,uk,res]=GMRES(A,f,u0,tol,m) solves Au=f using the GMRES
%   method starting at the initial guess u0 up to a tolerance tol using
%   at most m iterations. It computes u in the affine Krylov space
%   u0+(r0,Ar0...,A^(k-1)r0) by minimizing the norm ||f-A*u||_2 where k is
%   the smallest integer such that ||f-A*u||_2<tol.
%   GMRES returns in the matrix uk the iterates, in u the solution
%   computed, and in res the history of the norm of the residuals.

if nargin<5, m=100; end            % default values
if nargin<4, tol=1e-6; end
k=1;                               % Initialize the iteration index
r0=f-A*u0;                         % Compute the initial residual
res=norm(r0);                      % Initialize the vercor of residuals
rhs=res;                           % Initialize the right-hand side
Q(:,1)=r0/res;                     % First column of Q
while res(end)/norm(f)>tol && k<=m % GMRES iterations
  Q(:,k+1)=A*Q(:,k);               % PART 1: Arnoldi: (A Q_k = Q_{k+1} H_k)
  for j=1:k                        % modified Gram-Schmidt on v=A*Q(:,k)
    H(j,k)=Q(:,k+1)'*Q(:,j);
    Q(:,k+1)=Q(:,k+1)-H(j,k)*Q(:,j);
  end
  H(k+1,k)=norm(Q(:,k+1));         % Lower-diagonal element
  Q(:,k+1)=Q(:,k+1)/H(k+1,k);      % Compute the vector Q(:,k+1)
  R(k+1,k)=0;                      % PART 2: QR by Givens rotations
  R(:,k)=H(:,k);                   % New column of R
  for j=1:k-1                      % Former Givens rotations on last column
    Rk=c(j)*R(j,k)+s(j)*R(j+1,k);
    R(j+1,k)=-s(j)*R(j,k)+c(j)*R(j+1,k);
    R(j,k)=Rk;
  end                              % Next apply the new Givens rotation
  c(k)=R(k,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  s(k)=R(k+1,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  R(k,k)=c(k)*R(k,k)+s(k)*R(k+1,k);
```

```
  R(k+1,k)=0;
  rhs(k+1)=-s(k)*rhs(k);          % PART 3: Update rhs and residuals
  rhs(k)=c(k)*rhs(k);             % New Givens rotation on rhs
  res(k+1)=abs(rhs(k+1));         % Compute the new norm of the residual
  k=k+1;                          % Update iteration index
  if nargout>1
    y=R\rhs';                     % Compute the coefficients y_j
    uk(:,k)=u0+Q(:,1:end-1)*y;    % Compute u as sum_j y_j q_j
  end
end
y=R\rhs';                         % Compute the coefficients y_j
u=u0+Q(:,1:end-1)*y;              % Compute u as u0+sum_j y_j q_j
```

Next, we explain some details of this implementation. At each iteration k, the GMRES loop is given by three main parts:

- Part 1: One step of the Arnoldi method is performed to compute the new column $\boldsymbol{q}_{k+1}$ of Q_{k+1} and the kth columns of $H_k \in \mathbb{R}^{k+1\times k}$. This is obtained by a classical Gram-Schmidt orthogonalization process applied to the vector $\boldsymbol{v} = A\boldsymbol{q}_k$. Notice that Q_{k+1} has size $n \times k+1$, where n is the length of $\boldsymbol{f}$.

- Part 2: The matrix H_k is QR-factorized as $H_k = \widetilde{Q}R_k$ using Givens rotations. In particular, $\widetilde{Q} \in \mathbb{R}^{k+1\times k+1}$ is an orthogonal matrix obtained as the product $\widetilde{Q} = (G_k G_{k-1} \cdots G_1)^\top$, where each Givens matrix G_j has the form

$$G_j = \begin{bmatrix} I_{j-1} & & & \\ & \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} & & \\ & & & I_{k-j-1} \end{bmatrix},$$

  with I_{j-1} and I_{k-j-1} identities of dimension $j-1$ and $k-j-1$. The scalars c_k and s_k are computed to obtain that

$$[G_k(G_{k-1}\cdots G_1 H_k)]_{j+1,j} = 0.$$

  Notice that the matrices G_j are not explicitly constructed. At the iteration k, first the matrix $(G_{k-1}\cdots G_1 H_k)$ is assembled, and then the action of G_k on $(G_{k-1}\cdots G_1 H_k)$ is computed. Using the structure of the matrices G_j, this is achieved in $O(n)$ operations, see Problem 32 for more details.

- Part 3: The Givens matrix G_k is applied to the vector

$$\texttt{rhs} = G_{k-1}\cdots G_1(\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1) = \|\boldsymbol{r}_0\|_2\widetilde{Q}^\top\boldsymbol{e}_1 =: \boldsymbol{w}_{k-1}.$$

  This is necessary to solve the least-squares problem (3.48). To see this, recall the QR-factorization of H_k (Part 2) and observe that

$$\boldsymbol{w}_k = G_k G_{k-1}\cdots G_1(\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1) = G_k\boldsymbol{w}_{k-1}.$$

Hence, at iteration k, the vector $\boldsymbol{w}_k$ is obtained by applying G_k to $\boldsymbol{w}_{k-1}$, constructed at the previous iteration. Now, we denote $\boldsymbol{w}_k = \begin{bmatrix} \boldsymbol{w} \\ w_{k+1,k} \end{bmatrix}$ and $R_k = \begin{bmatrix} R \\ 0 \end{bmatrix}$, where $R \in \mathbb{R}^{k \times k}$ is upper triangular, and write

$$\begin{aligned} \left\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - H_k \boldsymbol{y} \right\|_2 &= \left\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - \widetilde{Q} R_k \boldsymbol{y} \right\|_2 = \left\| \widetilde{Q} (\|\boldsymbol{r}_0\|_2 \widetilde{Q}^\top \boldsymbol{e}_1 - R_k \boldsymbol{y}) \right\|_2 \\ &= \left\| \|\boldsymbol{r}_0\|_2 \widetilde{Q}^\top \boldsymbol{e}_1 - R_k \boldsymbol{y} \right\|_2 = \left\| \boldsymbol{w}_k - R_k \boldsymbol{y} \right\|_2 \\ &= \sqrt{\left\| \boldsymbol{w} - R\boldsymbol{y} \right\|_2^2 + |w_{k+1,k}|^2}. \end{aligned}$$

Since we assume that $\boldsymbol{y} = R^{-1}\boldsymbol{w}$, Theorem 26 (together with Remark 4) allows us to compute the norm of the residual as follows:

$$\begin{aligned} \|\boldsymbol{r}_k\|_2 = \|\boldsymbol{f} - A\boldsymbol{u}_k\|_2 &= \left\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - H_k \boldsymbol{y} \right\|_2 \\ &= \sqrt{\left\| \boldsymbol{w} - R\boldsymbol{y} \right\|_2^2 + |w_{k+1,k}|^2} \\ &= |w_{k+1,k}|. \end{aligned}$$

Even though GMRES is for general linear systems, and does too much work for symmetric positive definite problems, we now test it on our Laplace model problem from Section 1.3 using the analogous MATLAB script we have shown already for testing the more suitable CG algorithm for the Laplace model problem in Section 3.2. The first iterates we obtain from GMRES are shown in Figure 3.14. We see that GMRES converges rather quickly, comparable to CG in Figure 3.7. Since GMRES minimized the residual, and CG the A-norm of the error, we plot these quantities for comparison for GMRES and CG in Figure 3.15, together with the l^2-norm of the error. We clearly see that the A-norm of the error is smaller for CG than for GMRES, and also the l^2-norm of the error, but the residual is smaller for GMRES than for CG. Since the problem is symmetric, one should have used *MINRES* for these computations, which gives the same result as GMRES, but for much lower computational cost, comparable to CG, see also Section 3.6. Finally, we test the behavior of GMRES for different values of the grid size h. If we solve our Laplace test problem for different values of mesh size, we obtain the result depicted in Figure 3.16, which shows the deterioration of the convergence properties of GMRES for decreasing h (compare with Figure 3.10).

Remark 5 (Restarted GMRES)**.** *Each iteration generates a new vector, which has to be stored in memory. For large (but sparse) matrices, this can quickly use up the available memory and lead to problems. Therefore, one often uses GMRES with restarts: after m iterations, the vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m$ are discarded, the best approximation $\boldsymbol{u}_m$ found so far is retained and GMRES is restarted in order to solve for the corrections $\widetilde{\boldsymbol{u}}$ the system*

$$A\widetilde{\boldsymbol{u}} = \boldsymbol{f} - A\boldsymbol{u}_m = \boldsymbol{r}_m.$$

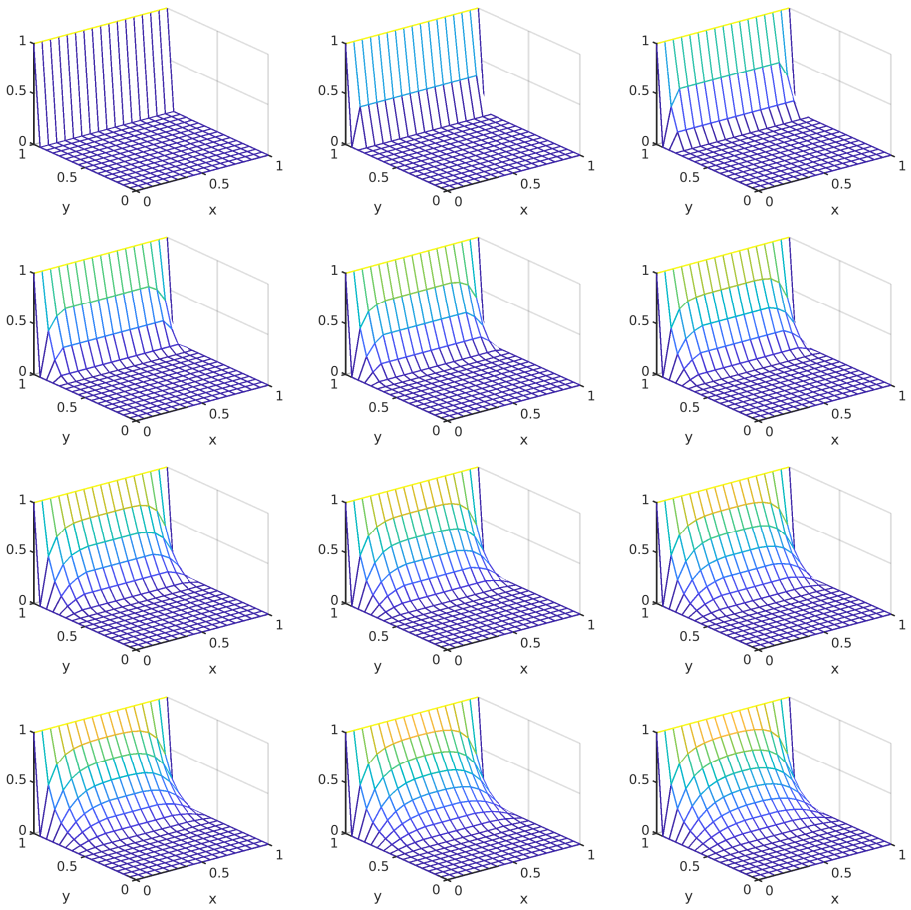


Figure 3.14: Initial guess and first iterations of GMRES applied to our Laplace model problem from Section 1.3.

Remark 6 (Lucky breakdown of GMRES)**.** *If it happens that $h_{k+1,k} = 0$, then the GMRES iteration breaks down. This breakdown of GMRES is also known as lucky breakdown , since if $h_{k+1,k} = 0$ holds, then the approximation $\boldsymbol{u}_k$ computed by GMRES at the kth iteration coincides with the exact solution $\boldsymbol{u}$. To see this, we recall the relation* (3.46) *and denote by $\widetilde{H}_k$ the $k \times k$ matrix obtained from H_k by deleting the last row. Since $h_{k+1,k} = 0$, we have that $Q_k \widetilde{H}_k = AQ_k$. Therefore, the invertibility of A and the fact that* $\operatorname{rank}(Q_k) = k$ *allow us to write*

$$k \geq \operatorname{rank}(\widetilde{H}_k) = \operatorname{rank}(Q_k \widetilde{H}_k) = \operatorname{rank}(AQ_k) = \operatorname{rank}(Q_k) = k,$$

which means that $\widetilde{H}_k$ (and hence also H_k) has full rank. Hence, using Theorem 26 and Remark 4 the residual norm $\|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 = \left\| \|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1 - \widetilde{H}_k \boldsymbol{w} \right\|_2$ is mini-

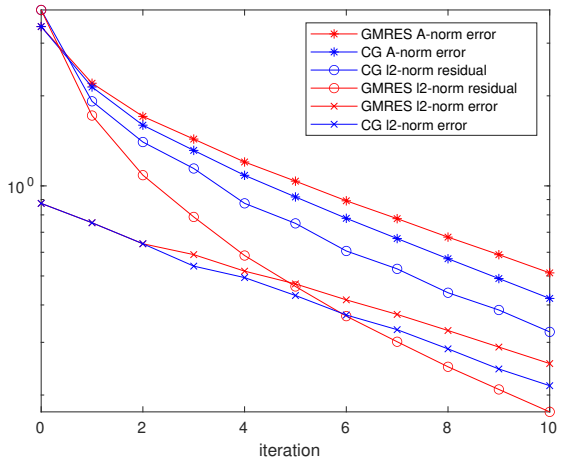


Figure 3.15: Comparison of CG and GMRES applied to our Laplace model problem from Section 1.3.

mized by $\boldsymbol{w} = \widetilde{H}_k^{-1}(\|\boldsymbol{r}_0\|_2 \boldsymbol{e}_1)$*, which obviously gives residual zero. This means that a breakdown of GMRES occurs if the exact solution has been found.*

Notice that this situation coincides with the breakdown of the Arnoldi procedure, which we discussed in Lemma 13. This lemma can be used together with Theorem 31 to obtain that the breakdown condition $h_{k+1,k} = 0$ *implies that GMRES converged to the exact solution.*

Remark 7 (Modified Gram-Schmidt and Householder Arnoldi)**.** *In Remark 3 we have briefly mentioned that, while the original Arnoldi algorithm uses the modified Gram-Schmidt (MGS) orthogonalization procedure, there exist more efficient implementations that cure the possible loss of orthogonality that characterize the numerical behavior of MGS. One of these implementations is based on an Arnoldi Householder process, which guarantees more robust numerical orthogonalization. Similarly also the GMRES implementation can be changed by using an Arnoldi Householder process; see, e.g., [108], and the* MATLAB *implementation* `gmres`*. However, Liesen and Strakoš clarify in their manuscript [85] that*

> *In conclusion, unless the matrix A is close to singular, MGS GMRES provides, despite the (gradual) loss of orthogonality among the computed MGS Arnoldi vectors, an approximate solution with normwise relative backward error comparable to the Householder Arnoldi GMRES.*

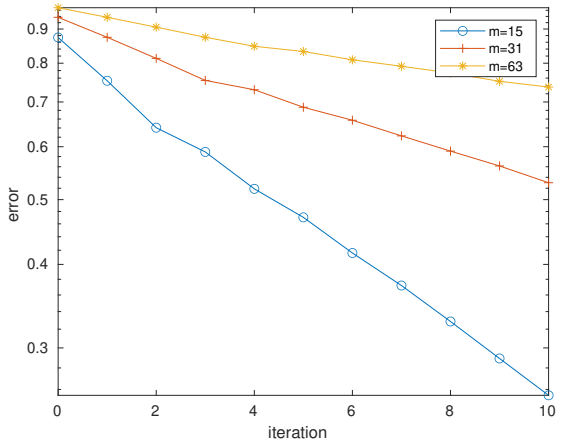


Figure 3.16: Convergence of GMRES for different gird sizes $h = \frac{1}{m+1}$.

3.6 Two families of Krylov methods

The following theorem gives the mathematical description of several important Krylov subspace methods in terms of the search and constraints spaces, and it shows the corresponding optimality properties.

Jörg Liesen and *Zdeněk Strakoš*, Krylov Subspace Methods, Principles and Analysis, 2013.

We have seen two main ideas to find an approximate solution of a given linear system $A\boldsymbol{u} = \boldsymbol{f}$ in the affine Krylov space $\boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$:

1. Methods based on the orthogonalization of the residual with respect to the Krylov space $\mathcal{K}_k(A, \boldsymbol{r}_0)$. These methods find an approximate solution $\boldsymbol{u}_k \in \boldsymbol{u}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$ such that

$$\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u}_k \perp \mathcal{K}_k(A, \boldsymbol{r}_0).$$

   Methods in this class are

   - CG, the conjugate gradient method, for symmetric positive definite matrices A, which was the first method of this type, invented independently by *David Hestenes* and *Eduard Stiefel* in 1952, see their joint paper [72].
   - SymmLQ, for symmetric but indefinite matrices A, invented by *Chris Paige* and *Michael Saunders* in 1975. This method is based on the Lanczos process and an LQ factorization of the obtained tridiagonal matrix and thus has a short recurrence with low storage requirements similar to CG, see [97]. The LQ factorization is the analog of the QR factorization, but with a lower triangular matrix L instead of the upper triangular matrix R. If the matrix is positive definite, SymmLQ generates the same iterates as CG.

- FOM, the Full Orthogonalization Method, which works for arbitrary matrices A, and was invented by *Yousef Saad* in 1981. The method uses Arnoldi, and thus requires substantially more storage, like GMRES, see [107].
- BiCGStab, the Bi-Conjugate Gradient method with stabilization, which is also a method for general matrices A, invented by *Henk A. Van Der Vorst* in 1992. The method constructs two bi-orthogonal sequences of vectors, one based on A and one based on $A^\top$, see [122]. The method uses short recurrences requiring therefore less storage than FOM, but it does not fully solve the problem of orthogonalization like in FOM.

2. Methods based on the minimization of the residual:

   - MINRES, the Minimum Residual method, for symmetric but possibly indefinite matrices A. This method was also invented by Paige and Saunders in 1975, in the same paper as SymmLQ, see [97], and uses a short recurrence with storage requirements similar to CG.
   - GMRES, the Generalized Minimum Residual method, for arbitrary matrices A, invented by Saad and Schultz in 1986, based on the Arnoldi process, see [112]. Even though this method needs a lot of storage, it is very popular for testing preconditioners, since it really minimizes the residual.
   - QMR, the Quasi-Minimum Residual method, also for general matrices A, and using a short recurrence with storage requirements similar to CG. This method was invented by *Roland W. Freund* and *Noël M. Nachtigal* in 1991, and only approximately solves the minimization problem, see [42].

Krylov methods are still a very active area of research, see the recent monograph [85] and references therein, and their convergence properties are far from being completely understood, even without taking into account round-off error. For normal matrices A however, i.e. matrices such that $AA^\top = A^\top A$, all these methods have a good convergence behavior if their spectrum $\{\lambda_j(A)\}$ is close to 1 in the complex plane. For most matrices coming from applications however, this is unfortunately not the case, especially for discretizations of partial differential equations, and the system first needs to be transformed into a new system which has a more favorable spectrum. This process is called preconditioning, and is the subject of the next chapter.

3.7 Problems

Problem 24. *Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and denote by $\nabla f(\boldsymbol{v})$ the gradient of f in $\boldsymbol{v} \in \mathbb{R}^n$ obtained by the usual scalar product for $\mathbb{R}^n$. Prove that $\nabla f(\boldsymbol{v})$ is orthogonal to the level set of f in $\boldsymbol{v}$ and that $-\nabla f(\boldsymbol{v})$ defines the direction of steepest descent of f in $\boldsymbol{v}$.*

Problem 25. *Implement the steepest descent method to solve a linear system* $A\boldsymbol{u} = \boldsymbol{f}$. *Test your codes to solve the Laplace equation.*

Problem 26. *In this exercise we prove the Kantorovitch inequality using the Wielandt inequality [73]. Consider a Hermitian positive definite matrix* $A \in \mathbb{C}^{n\times n}$ *having eigenvalues* $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. *The Wielandt inequality is*

$$|\boldsymbol{u}^* A\boldsymbol{v}|^2 \leq \left(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}\right)^2 (\boldsymbol{u}^* A\boldsymbol{u})(\boldsymbol{v}^* A\boldsymbol{v}) \text{ for all orthogonal } \boldsymbol{u}, \boldsymbol{v} \in \mathbb{C}^n, \quad (3.72)$$

and the Kantorovitch inequality is

$$(\boldsymbol{u}^* A\boldsymbol{u})(\boldsymbol{u}^* A^{-1}\boldsymbol{u}) \leq \frac{(\lambda_1 + \lambda_n)^2}{4\lambda_1\lambda_n}\|\boldsymbol{u}\|_2^4 \text{ for all } \boldsymbol{u} \in \mathbb{C}^n. \quad (3.73)$$

The Kantorovitch inequality can be obtained by the Wielandt inequality following the next steps.

(a) Show that (3.72) *and* (3.73) *are both satisfied if* $\boldsymbol{u} \in \mathbb{C}^n$ *is an eigenvector of* A. *Hint: use the arithmetic-geometric mean inequality* $\sqrt{ab} \leq \frac{1}{2}(a+b)$ *for any* $a, b > 0$.

(b) Show that if $\boldsymbol{u} \in \mathbb{C}^n$ *is not an eigenvector of* A, *then* $A^{-1}x - (\boldsymbol{u}^* A^{-1}\boldsymbol{u})\boldsymbol{u} \neq 0$ *and* $x - (\boldsymbol{u}^* A^{-1}\boldsymbol{u})A\boldsymbol{u} \neq 0$.

(c) Show that if $\boldsymbol{u} \in \mathbb{C}^n$ *is a unit vector, that is* $\|\boldsymbol{u}\|_2 = 1$, *then* $(\boldsymbol{u}^* A\boldsymbol{u})(\boldsymbol{u}^* A^{-1}\boldsymbol{u}) \geq 1$, *with strict inequality if* $\boldsymbol{u}$ *is not an eigenvector of* A. *Hint: begin with* $1 = (\boldsymbol{u}^*\boldsymbol{u})^2 = (\boldsymbol{u}^* A^{1/2} A^{-1/2}\boldsymbol{u})^2$ *and use the Cauchy-Schwarz inequality.*

(d) Consider a unit vector $\boldsymbol{u} \in \mathbb{C}^n$ *which is not an eigenvector of* A. *Define* $\boldsymbol{v} := A^{-1}\boldsymbol{u} - (\boldsymbol{u}^* A^{-1}\boldsymbol{u})\boldsymbol{u}$. *Show that*

(1) $\boldsymbol{v}^*\boldsymbol{u} = 0$.
(2) $A\boldsymbol{v} \neq 0$.
(3) $\boldsymbol{u}^* A\boldsymbol{v} = 1 - (\boldsymbol{u}^* A\boldsymbol{u})(\boldsymbol{u}^* A^{-1}\boldsymbol{u}) < 0$.
(4) $\boldsymbol{v}^* A\boldsymbol{v} = -(\boldsymbol{u}^* A^{-1}\boldsymbol{u})(\boldsymbol{v}^* A\boldsymbol{u})$.

Hint: use the results proved in (b) and (c).

(e) Use the Wielandt inequality (3.72) *and the results proved above to obtain the Kantorovitch inequality* (3.73).

Problem 27. *Implement the conjugate gradient method to solve a linear system* $A\boldsymbol{u} = \boldsymbol{f}$. *Test your codes to solve the Laplace equation.*

Problem 28. *Show that the step-length* $\alpha_k = \frac{\boldsymbol{r}_k^\top \boldsymbol{p}_k}{\boldsymbol{p}_k^\top A\boldsymbol{p}_k}$ *is optimal in the sense that it minimizes the function*

$$\alpha \mapsto \phi(\alpha) := \frac{1}{2}(\boldsymbol{u}_k + \alpha\boldsymbol{p}_k)^\top A(\boldsymbol{u}_k + \alpha\boldsymbol{p}_k) - \boldsymbol{f}^\top(\boldsymbol{u}_k + \alpha\boldsymbol{p}_k).$$

Show that α_k *is also a minimizer for*

$$\alpha \mapsto \varphi(\alpha) := \|\boldsymbol{u}_k + \alpha\boldsymbol{p}_k - \boldsymbol{u}\|_A^2.$$

Problem 29. *Using Lemma 10, prove that CG minimizes at each iteration the residual in the A^{-1}-norm ($\|\boldsymbol{z}\|_{A^{-1}} = \sqrt{\boldsymbol{z}^\top A^{-1}\boldsymbol{z}}$), that is*

$$\|\boldsymbol{r}_k\|_{A^{-1}} = \min_{\substack{p\in\mathcal{P}_k \\ p(0)=1}} \|p(A)\boldsymbol{r}_0\|_{A^{-1}}.$$

Problem 30. *Let A be a symmetric and positive definite matrix. Using the estimate* (3.36)*, prove that*

$$\|\boldsymbol{u}-\boldsymbol{u}_k\|_2 \le 2\sqrt{\kappa(A)}\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|\boldsymbol{u}-\boldsymbol{u}_0\|_2.$$

Hint: first show that $\lambda_{\min}(A)\|\boldsymbol{x}\|_2^2 \le \|\boldsymbol{x}\|_A^2 \le \lambda_{\max}(A)\|\boldsymbol{x}\|_2^2$ for any vector $\boldsymbol{x}$, and then use (3.36)*.*

Problem 31. *In this exercise you will derive the conjugate gradient method, for the solution of $A\boldsymbol{u} = \boldsymbol{f}$ ($A \in \mathbb{R}^{n\times n}$ symmetric positive definite), from the Lanczos method. To do so, consider the following steps.*

1. *Recall the decomposition $AQ_k = Q_{k+1}H_k$, where $Q_k = \begin{bmatrix}\boldsymbol{q}_1 & \cdots & \boldsymbol{q}_k\end{bmatrix} \in \mathbb{R}^{n\times k}$ and assume that $\boldsymbol{q}_1 = \frac{1}{\|\boldsymbol{f}\|_2}\boldsymbol{f}$. The matrix H_k has the form $H_k = \begin{bmatrix}T_k \\ \boldsymbol{v}\end{bmatrix}$ where*

$$T_k = \begin{bmatrix}\alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & & \ddots & & \\ & & & \beta_{k-1} & \alpha_k\end{bmatrix},$$

and $\boldsymbol{v} = \begin{bmatrix}0 & \cdots & 0 & \beta_k\end{bmatrix}$.

   a) *Show that $Q_k^\top A Q_k = T_k$.*

   b) *Show that*

$$\boldsymbol{u}_k := \operatorname*{arg\,min}_{\boldsymbol{x}_k\in\operatorname{span}\{\boldsymbol{q}_1\cdots\boldsymbol{q}_k\}} \frac{1}{2}\boldsymbol{x}_k^\top A\boldsymbol{x}_k - \boldsymbol{x}_k^\top\boldsymbol{f} = Q_k T_k^{-1}\big(\|\boldsymbol{f}\|_2\boldsymbol{e}_1\big).$$

   Hint: Introduce a vector $\boldsymbol{y}\in\mathbb{R}^k$ such that $\boldsymbol{x}_k = Q_k\boldsymbol{y}$ and prove that $\boldsymbol{y} = T_k^{-1}Q_k^\top\boldsymbol{f}$.

2. *Consider the LU-decomposition of T_k, that is $T_k = L_kU_k$, where*

$$L_k = \begin{bmatrix}1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & & \\ & & \gamma_{k-1} & 1\end{bmatrix} \quad \text{and} \quad U_k = \begin{bmatrix}\eta_1 & \beta_1 & & \\ & \eta_2 & \beta_2 & \\ & & \ddots & \\ & & & \eta_k\end{bmatrix}.$$

a) *Show that* $\gamma_{k-1} = \frac{\beta_{k-1}}{\eta_{k-1}}$ *and* $\eta_k = \alpha_k - \gamma_{k-1}\beta_{k-1}$.

b) *Define* $P_k := Q_k U_k^{-1}$ *and show that the columns* $\boldsymbol{p}_j$ *of* P_k *satisfy* $\boldsymbol{p}_j = \frac{1}{\eta_j}\big(\boldsymbol{q}_j - \beta_{j-1}\boldsymbol{p}_{j-1}\big)$ *for* $j = 1, \ldots, k-1$ *and* $\boldsymbol{p}_k = \frac{1}{\eta_k}\boldsymbol{q}_k$.

c) *Show that* $\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \zeta_k \boldsymbol{p}_k$ *for a* $\zeta_k \in \mathbb{R}$. *Hint: begin with* $\boldsymbol{u}_k = Q_k T_k^{-1}\big(\|\boldsymbol{f}\|_2 \boldsymbol{e}_1\big)$ *and use the LU-decomposition of* T_k.

3. *Prove that:*

- *the vectors* $\boldsymbol{p}_k$ *are A-orthogonal.*
- $\boldsymbol{r}_k = \sigma_k \boldsymbol{q}_{k+1}$ *for some* $\sigma_k \in \mathbb{R}$, *where* $\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u_k}$ *is the residual.*
- $\boldsymbol{r}_k \perp \boldsymbol{q}_j$ *for* $j = 1, \ldots, k-1$ *and that* $\boldsymbol{r}_k \perp \boldsymbol{r}_{k-1}$

4. *Take a vector* $\widetilde{\boldsymbol{p}}_{k-1}$ *parallel to* $\boldsymbol{p}_k$, *then we can rewrite* $\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \zeta_k \boldsymbol{p}_k$ *(2.c) as* $\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \widetilde{\alpha}_{k-1}\widetilde{\boldsymbol{p}}_{k-1}$ *for some* α_{k-1}.

- *Show that* $\widetilde{\boldsymbol{p}}_k = \boldsymbol{r}_k + \widetilde{\beta}_k \widetilde{\boldsymbol{p}}_{k-1}$ *for some* $\widetilde{\beta}_k \in \mathbb{R}$.
- *Compute* $\widetilde{\alpha}_k$ *by using the orthogonality* $\boldsymbol{r}_k \perp \boldsymbol{r}_{k-1}$.
- *Compute* $\widetilde{\beta}_k$ *by using the A-orthogonality of the vectors* $\widetilde{p}_k$. *Hint: you should obtain that* $\widetilde{\alpha}_k = \frac{\|\boldsymbol{r}_k\|_2^2}{\widetilde{\boldsymbol{p}}_k^{\top} A \widetilde{\boldsymbol{p}}_k}$ *and* $\widetilde{\beta}_k = \frac{\|\boldsymbol{r}_k\|_2^2}{\|\boldsymbol{r}_{k-1}\|_2^2}$.

5. *Summarize the obtained results and compare them with the CG algorithm.*

Problem 32. *Construct an algorithm that computes in* $O(n^2)$ *operations the QR decomposition of a Hessenberg matrix H of size* $n \times n$ *using Givens rotations. Next, suppose that the QR decomposition is already known for the matrix that corresponds to the first* $n-1$ *columns of H. Show how this information can be used to compute the QR decomposition in* $O(n)$ *operations.*

Problem 33. *Prove the statements (b), (d), (e), and (f) of Theorem 29.*

Problem 34. *Let* $H \in \mathbb{C}^{n\times n}$ *and* $S \in \mathbb{C}^{n\times n}$ *be Hermitian and skew-Hermitian, respectively. Prove that* $\rho(H) = \|H\|_2 = \nu(H)$ *and* $\rho(S) = \|S\|_2 = \nu(S)$, *where* $\nu(H)$ *and* $\nu(S)$ *are the numerical radii of H and S.*

Problem 35. *Consider the GMRES algorithm for the solution of* $A\boldsymbol{u} = \boldsymbol{f}$, *with the matrix* $A \in \mathbb{R}^{2n\times 2n}$ *given by*

$$A = \begin{bmatrix} I & B \\ 0 & I \end{bmatrix},$$

where I is the $n \times n$ *identity and* $B \in \mathbb{R}^{n\times n}$ *is an arbitrary matrix. How many iterations are at most performed (for an arbitrary choice of* $\boldsymbol{f} \in \mathbb{R}^n$*) by GMRES to converge?*

Chapter 4

Preconditioning

Es werden in der Regel doch mehrere der ausserhalb der Diagonale befindlichen Coefficienten so bedeutende Werthe annehmen, dass der Erfolg der soeben angegebenenen Näherungsmethode dadurch vereitelt wird. Man kann aber, wie ich im Folgenden zeigen will, durch Wiederholung einer leichten Rechnung die Gleichungen in andere umformen, in welchen der erwähnte Uebelstand immer weniger hervortritt, so dass zuletzt die Gleichungen eine Form erhalten, welche die Anwendung der obigen Näherungsmethode verstattet.

Jacobi, Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden lineären Gleichungen, 1845.

We have seen that Krylov methods are based on an optimality principle, minimizing for example the error or the residual for a given number of matrix vector multiplications, and it seems therefore that one can not do better than these methods. Nevertheless in practice, often the convergence of Krylov methods is also slow, and one has to first transform the linear system into a new one, such that the Krylov method converges better when applied to the transformed system. This process of transforming the original system into a new one is called *Preconditioning*. The first traces we know about preconditioning go back to the original paper of Jacobi [77], where he invented the Jacobi method, and the preconditioning step he uses makes the Jacobi method converge faster, see the quote above. With the advent of Krylov methods, preconditioning became a main research area, and intensive efforts are still ongoing today.

4.1 Stationary iterative methods and preconditioning

Thus, any iterative technique can be used as a preconditioner: block-SOR, SSOR, ADI, Multi-grid, etc. More interestingly, iterative procedures such as GMRES, CGNR, or CGS can also be used as preconditioners.

Yousef Saad, Iterative Methods for Sparse Linear Systems, 2000.

There is an important relation between stationary iterative methods and preconditioning, as Saad already emphasized in the quote above. To explain

this, we first have to introduce the idea of preconditioning: we consider a linear system $A\boldsymbol{u} = \boldsymbol{f}$ with $A \in \mathbb{R}^{n\times n}$, $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^n$. The Krylov methods discussed in Chapter 3 converge well if the spectrum of A is very close to 1, that is if A is close to the identity[1]. Most matrices A from applications are however far from the identity, and Krylov methods are converging very slowly, a typical example is the discretized Laplacian we have used as our guiding example in this book. One should then transform the system $A\boldsymbol{u} = \boldsymbol{f}$ in order to make it easier to solve with a Krylov method. To do so, let $M \in \mathbb{R}^{n\times n}$ be an invertible matrix, called preconditioner; the transformed (or preconditioned) system is then given by

$$M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f}. \tag{4.1}$$

From (4.1), we can see that

1. $M = A$ would be the best possible preconditioner, since the system (4.1) becomes
$$I\boldsymbol{u} = A^{-1}\boldsymbol{f},$$
with all eigenvalues equal to 1, not just close to one, and a Krylov method will converge in one step, since the solution lies in the initial affine Krylov space $\boldsymbol{u}_0 + \mathcal{K}_0(M^{-1}A, \boldsymbol{r}_0) = \boldsymbol{u}_0 + \text{span}\{A^{-1}\boldsymbol{f} - \boldsymbol{u}_0\}$.

2. The computation of M^{-1}, i.e. solving linear systems with system matrix M, should however be much less expensive than solving systems with the original matrix A, since otherwise one would better solve the system $A\boldsymbol{u} = \boldsymbol{f}$ directly.

In preconditioning one must therefore find a compromise: an M which is reasonably close to A, but easy and cheap to invert. The preconditioning in (4.1) is called *left-preconditioning*, since the preconditioner M acts on the left of A. *Right-preconditioning* is also possible: in this case one would solve

$$AM^{-1}\boldsymbol{v} = \boldsymbol{f} \quad \text{with} \quad \boldsymbol{u} = M^{-1}\boldsymbol{v}. \tag{4.2}$$

If M is symmetric and positive definite, then one can *precondition symmetrically*,

$$L^{-1}AL^{-\top}\boldsymbol{v} = L^{-1}\boldsymbol{f} \quad \text{with} \quad \boldsymbol{u} = L^{-\top}\boldsymbol{v}, \tag{4.3}$$

where $M = LL^\top$. The matrix L could be for example the lower-triangular Cholesky factor of M. Preconditioners can be based purely on algebraic information, i.e. the matrix, see the early contribution quoted above from [90], or on the underlying physical problem, which is done with the domain decomposition methods quoted above from [126], and multigrid methods.

In Chapter 2 we have studied stationary iterative methods based on the splitting $A = M - N$, and the conditions for obtaining a good method, i.e. a good matrix M, were

[1] This statement rigorously holds only for normal matrices. For non-normal matrices, convergence can still be very bad, see [85, Section 5.7.2], because the bounds based on spectral information contain the condition number of the eigenvectors which can make them useless for understanding convergence of the Krylov method. See also the example given in Section 3.5 after Theorem 30.

1. the spectral radius of the iteration matrix $I - M^{-1}A$ should be small, close to zero, for fast convergence of the stationary iterative method,

2. the computation of M^{-1} should be cheap, like with the diagonal M for Jacobi, or the triangular M for Gauss-Seidel.

Comparing these two conditions to the conditions for a good preconditioner for a Krylov method earlier, we see that they are very much related, and it is not a coincidence that these matrices are called M in both cases: the condition of easy invertibility of M is identical, and the condition that the spectral radius of the iteration matrix $I - M^{-1}A$ should be small implies that the eigenvalues of $M^{-1}A$ should be close to 1. Rewriting the stationary iteration (2.1), i.e.

$$M\boldsymbol{u}_{k+1} = N\boldsymbol{u}_k + \boldsymbol{f}, \tag{4.4}$$

in the form

$$\boldsymbol{u}_{k+1} = M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f} = (I - M^{-1}A)\boldsymbol{u}_k + M^{-1}\boldsymbol{f}, \tag{4.5}$$

we see that at convergence of the stationary iterative method, we obtain the system

$$\boldsymbol{u} = (I - M^{-1}A)\boldsymbol{u} + M^{-1}\boldsymbol{f} \quad \Longleftrightarrow \quad M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f},$$

which is precisely the preconditioned system (4.1). So one can either solve the preconditioned system (4.1) using the associated stationary iterative method (4.4), or using a Krylov method to solve (4.1). The following theorem indicates that one should always use a Krylov method!

Theorem 32 (Preconditioned Krylov methods versus stationary iterative methods)**.** *Consider a splitting $A = M - N$ with M invertible and the corresponding stationary method* (4.5) *and a Krylov method minimizing the residual applied to the preconditioned system* (4.1)*. Define the corresponding preconditioned residuals as $\boldsymbol{r}_k^{\text{stat}} := M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k^{\text{stat}}$ and $\boldsymbol{r}_k^{\text{kry}} := M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k^{\text{kry}}$. Then we have that $\|\boldsymbol{r}_k^{\text{kry}}\|_2 \leq \|\boldsymbol{r}_k^{\text{stat}}\|_2$ for any $k = 0, 1, 2, \ldots$. In other words, a stationary iterative method* (4.5) *based on M can never perform less iterations than a Krylov method minimizing the residual applied to* (4.1)*.*

Proof. A stationary iterative method based on the splitting $A = M - N$ computes at the iteration k the approximation

$$\boldsymbol{u}_k = (I - M^{-1}A)\boldsymbol{u}_{k-1} + M^{-1}\boldsymbol{f} = \boldsymbol{u}_{k-1} + \boldsymbol{r}_{k-1}^{\text{stat}},$$

where the *preconditioned residual* $\boldsymbol{r}_k^{\text{stat}}$ satisfies

$$\begin{aligned}\boldsymbol{r}_k^{\text{stat}} &= M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k \\ &= M^{-1}\boldsymbol{f} - M^{-1}A(\boldsymbol{u}_{k-1} + \boldsymbol{r}_{k-1}^{\text{stat}}) \\ &= \boldsymbol{r}_{k-1}^{\text{stat}} - M^{-1}A\boldsymbol{r}_{k-1}^{\text{stat}} = (I - M^{-1}A)\boldsymbol{r}_{k-1}^{\text{stat}}.\end{aligned}$$

Using this recursively we get

$$\boldsymbol{r}_k^{\text{stat}} = (I - M^{-1}A)\boldsymbol{r}_{k-1}^{\text{stat}} = (I - M^{-1}A)^2\boldsymbol{r}_{k-2}^{\text{stat}} = \cdots = (I - M^{-1}A)^k\boldsymbol{r}_0.$$

Therefore, we have with the residual polynomial $p_k^{\text{stat}}(x) = (1-x)^k$

$$\boldsymbol{r}_k^{\text{stat}} = p_k^{\text{stat}}(M^{-1}A)\boldsymbol{r}_0, \text{ with } p_k^{\text{stat}}(0) = 1.$$

A Krylov method on the other hand uses the Krylov space

$$\mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0) := \left\{\boldsymbol{r}_0, M^{-1}A\boldsymbol{r}_0, \ldots, (M^{-1}A)^{k-1}\boldsymbol{r}_0\right\}$$

to find an approximation $\boldsymbol{u}_k \in \boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0)$, that is

$$\boldsymbol{u}_k = \boldsymbol{u}_0 + \sum_{j=1}^{k} \gamma_j (M^{-1}A)^{j-1}\boldsymbol{r}_0,$$

for some coefficients γ_j. The *preconditioned residual* $\boldsymbol{r}_k^{\text{kry}}$ of the Krylov method thus satisfies

$$\boldsymbol{r}_k^{\text{kry}} = M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k = M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_0 - M^{-1}A\sum_{j=1}^{k}\gamma_j(M^{-1}A)^{j-1}\boldsymbol{r}_0,$$

and hence

$$\boldsymbol{r}_k^{\text{kry}} = p_k^{\text{kry}}(M^{-1}A)\boldsymbol{r}_0, \text{ with } p_k^{\text{kry}}(0) = 1,$$

where p_k^{kry} is a polynomial of degree k. Now a Krylov method minimizing the residual will find the polynomial p_k^{kry} such that $\|\boldsymbol{r}_k^{\text{kry}}\|_2$ is as small as possible, and thus at least as small as $\|\boldsymbol{r}_k^{\text{stat}}\|_2$ which used the simple polynomial $p_k^{\text{stat}}(x) = (1-x)^k$ of the associated stationary method. □

The previous result shows that one should never use a stationary iterative method in practice. Even though the Krylov method might have a small overhead, needing a few more scalar products and to store a few more vectors, the benefit of the optimized polynomial far out-weights this additional cost. Hence all the iterative methods studied in Chapter 2, like Jacobi, Gauss-Seidel, and SOR, should be used as preconditioners for a Krylov method. The Krylov method serves as an accelerator of convergence, and Krylov methods can be developed from this point of view, see [57, Chapter 11]. The importance of studying stationary iterative methods lies in the development of preconditioners.

4.2 Left and right preconditioning

> *The above transformation of the linear system $A \to M^{-1}A$ is often not what is used in practice.*
> Templates for the solution of linear systems: building blocks for iterative methods, 1994.

This section is motivated by the following questions: What is the difference between left and right preconditioning? Is there any relation between these two techniques? Do they lead to similar convergence behavior of Krylov methods?

A first simple remark is that $M^{-1}A$ and AM^{-1} share the same spectrum, since

$$M^{-1}A\boldsymbol{v} = \lambda\boldsymbol{v} \quad \Longleftrightarrow \quad M^{-1}AM^{-1}M\boldsymbol{v} = \lambda\boldsymbol{v} \quad \Longleftrightarrow \quad AM^{-1}(M\boldsymbol{v}) = \lambda(M\boldsymbol{v}).$$

Hence, one can expect that a Krylov method applied to the systems

$$M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f} \quad \text{and} \quad AM^{-1}(M\boldsymbol{u}) = \boldsymbol{f} \tag{4.6}$$

would show similar convergence behavior. However, as we have seen in Section 3.5, the spectrum does not necessarily govern the convergence of certain Krylov methods. For this reason, we compare here the optimality properties of GMRES applied to the left- and right-preconditioned systems (4.6).

In case of left preconditioning, GMRES minimizes the residual norm

$$\|M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k\|_2$$

over the affine Krylov space

$$\boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0^{\mathrm{p}}), \tag{4.7}$$

where $\boldsymbol{r}_0^{\mathrm{p}}$ is the *preconditioned residual*, that is $\boldsymbol{r}_0^{\mathrm{p}} = M^{-1}\boldsymbol{r}_0$, with $\boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$. Hence, we have that

$$\begin{aligned}\|M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_k\|_2 &= \|M^{-1}\boldsymbol{f} - M^{-1}A\boldsymbol{u}_0 - M^{-1}Ap_{k-1}(M^{-1}A)\boldsymbol{r}_0^{\mathrm{p}}\|_2 \\ &= \|\boldsymbol{r}_0^{\mathrm{p}} - M^{-1}Ap_{k-1}(M^{-1}A)\boldsymbol{r}_0^{\mathrm{p}}\|_2,\end{aligned}$$

where p_{k-1} is a polynomial that minimizes the residual norm over all the space of polynomials of degree lower or equal to $k-1$. We now perform the following simple calculation

$$\begin{aligned}\boldsymbol{r}_0^{\mathrm{p}} - M^{-1}Ap_{k-1}(M^{-1}A)\boldsymbol{r}_0^{\mathrm{p}} &= M^{-1}\big(\boldsymbol{r}_0 - Ap_{k-1}(M^{-1}A)M^{-1}\boldsymbol{r}_0\big) \\ &= M^{-1}\big(\boldsymbol{r}_0 - AM^{-1}p_{k-1}(AM^{-1})\boldsymbol{r}_0\big).\end{aligned} \tag{4.8}$$

Therefore, GMRES applied to a left-preconditioned system minimizes the norm

$$\|M^{-1}\big(\boldsymbol{r}_0 - AM^{-1}p_{k-1}(M^{-1}A)\boldsymbol{r}_0\big)\|_2$$

over the space of polynomials of degree lower or equal to $k-1$.

Consider now GMRES applied to the right-preconditioned system in (4.6). In this case GMRES minimizes the residual norm

$$\|\boldsymbol{f} - AM^{-1}\boldsymbol{v}_k\|_2$$

over the affine Krylov space space

$$\boldsymbol{v}_0 + \mathcal{K}_k(AM^{-1}, \boldsymbol{r}_0), \tag{4.9}$$

where $\boldsymbol{r}_0 = \boldsymbol{f} - AM^{-1}\boldsymbol{v}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$. If we express (4.9) in terms of the variable $\boldsymbol{u}$ (rather than $\boldsymbol{v}$), we obtain

$$\begin{aligned} M^{-1}\boldsymbol{v}_0 + M^{-1}\mathcal{K}_k(AM^{-1}, \boldsymbol{r}_0) &= \boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, M^{-1}\boldsymbol{r}_0) \\ &= \boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0^{\mathrm{p}}), \end{aligned}$$

which is exactly the Krylov space (4.7). In other words, GMRES applied to a right-preconditioned system computes an approximation

$$\boldsymbol{u}_k = \boldsymbol{u}_0 + p_{k-1}(M^{-1}A)M^{-1}\boldsymbol{r}_0,$$

where p_{k-1} is a polynomial obtained by minimizing the residual norm

$$\|\boldsymbol{f} - AM^{-1}\boldsymbol{v}_k\|_2 = \|\boldsymbol{f} - A\boldsymbol{u}_k\|_2 = \|\boldsymbol{r}_0 - AM^{-1}p_{k-1}(AM^{-1})\boldsymbol{r}_0\|_2$$

over the space of polynomials of degree lower or equal to $k-1$.

Summarizing our findings, we have that GMRES applied to the left-preconditioned system performs the minimization of the preconditioned residual,

$$\min_{\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0^{\mathrm{p}})} \|M^{-1}(\boldsymbol{f} - A\widetilde{\boldsymbol{u}})\|_2 \quad \Leftrightarrow \quad \min_{p_{k-1}} \|M^{-1}\big(\boldsymbol{r}_0 - AM^{-1}p_{k-1}(M^{-1}A)\boldsymbol{r}_0\big)\|_2,$$

while GMRES applied to the right-preconditioned system performs the minimization of the unpreconditioned residual,

$$\min_{\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \mathcal{K}_k(M^{-1}A, \boldsymbol{r}_0^{\mathrm{p}})} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2 \quad \Leftrightarrow \quad \min_{p_{k-1}} \|\boldsymbol{r}_0 - AM^{-1}p_{k-1}(M^{-1}A)\boldsymbol{r}_0\|_2,$$

but in both cases the minimization problem is posed on the same (Krylov or polynomial) space. This indicates that significant differences in the convergence behaviors are possible, and this is especially the case when M is ill-conditioned; see, e.g., [108, Section 9.3.4].

4.3 Preconditioning in practice

> *Remarkably, the splitting of M is in practice not needed.*
> Templates for the solution of linear systems: building blocks for iterative methods, 1994.

In many cases (e.g. multigrid methods and domain decomposition methods; see Sections 4.6, 4.7, 4.8 and 4.10) the explicit structure of M is not known or expensive to compute. Moreover, one would certainly not compute first the preconditioned system $M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f}$ and then use a Krylov method, because assembling the matrix $M^{-1}A$ could be too expensive! At each iteration of the Krylov method, we would like to solve a linear system $M\boldsymbol{v} = \boldsymbol{w}$ at most. This leads to a common question: how does a preconditioner have to be used in practice? Moreover, assuming that one wishes to use CG with a symmetric preconditioner M, how to handle the symmetric preconditioning (4.3), where the expensive decomposition $M = LL^{\top}$ is required? These questions are answered

in this section, see also the quote above which is promising. Let us begin with the first one.

A Krylov method applied to the system $A\boldsymbol{u} = \boldsymbol{f}$ constructs the Krylov subspaces $\mathcal{K}_k(A, \boldsymbol{p})$. This is in general obtained by a matrix-vector multiplication of the type $A\boldsymbol{p}$, see, e.g., the CG and GMRES algorithm given in Sections 3.2 and 3.5. Now, a Krylov method applied to the preconditioned system $M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f}$ will construct a Krylov space $\mathcal{K}_k(M^{-1}A, \boldsymbol{p})$ by computing at each step the vector $M^{-1}A\boldsymbol{p}$. Therefore, one can easily modify the Krylov algorithm by adding an extra step at each iteration: once the vector $\boldsymbol{v} = A\boldsymbol{p}$ is computed, one can then compute $\boldsymbol{w} = M^{-1}\boldsymbol{v}$ by solving the linear system $M\boldsymbol{w} = \boldsymbol{v}$. As an example, we modified the GMRES implementation given in Section 3.5 in order to add the possibility of preconditioning:

```
function [u,uk,res]=PGMRES(A,f,u0,M,tol,m)
% PGMRES: Preconditioned Generalized Minimal Residual
%   [u,uk,res]=PGMRES(A,f,u0,M,tol,m) solves M^-1Au=M^-1f using the GMRES
%   method starting at the initial guess u0 up to a tolerance tol using
%   at most m iterations. It computes u in the (affine) preconditioned
%   Krylov space by minimizing the norm ||M^-1f-M^-1A*u||_2 where k is
%   the smallest integer such that ||M^-1f-M^-1A*u||_2<tol.
%   PGMRES returns in the matrix uk the iterates, in u the solution
%   computed, and in res the history of the norm of the residuals.

if nargin<6, m=100; end            % default values
if nargin<5, tol=1e-6; end
if nargin<4, M=speye(length(f)); end
k=1;                               % Initialize the iteration index
r0=f-A*u0;                         % Compute the initial residual
r0=M\r0;                           % Apply the preconditioner to r0
res=norm(r0);                      % Initialize the vercor of residuals
rhs=res;                           % Initialize the right-hand side
Q(:,1)=r0/res;                     % First column of Q
while res(end)/norm(b)>tol && k<=m % GMRES iterations
  Q(:,k+1)=A*Q(:,k);               % PART 1: Arnoldi: (M\A*Q_k=Q_{k+1}*H_k)
  Q(:,k+1)=M\Q(:,k+1);             % Apply the preconditioner
  for j=1:k                        % modified Gram-Schmidt on v=M\A*Q(:,k)
    H(j,k)=Q(:,k+1)'*Q(:,j);
    Q(:,k+1)=Q(:,k+1)-H(j,k)*Q(:,j);
  end
  H(k+1,k)=norm(Q(:,k+1));         % Lower-diagonal element
  Q(:,k+1)=Q(:,k+1)/H(k+1,k);      % Compute the vector Q(:,k+1)
  R(k+1,k)=0;                      % PART 2: QR by Givens rotations
  R(:,k)=H(:,k);                   % New column of R
  for j=1:k-1                      % Former Givens rotations on last column
    Rk=c(j)*R(j,k)+s(j)*R(j+1,k);
    R(j+1,k)=-s(j)*R(j,k)+c(j)*R(j+1,k);
    R(j,k)=Rk;
  end                              % Apply the new Givens rotation
  c(k)=R(k,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  s(k)=R(k+1,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
```

```
  R(k,k)=c(k)*R(k,k)+s(k)*R(k+1,k);
  R(k+1,k)=0;                          % PART 3: Update rhs and residuals
  rhs(k+1)=-s(k)*rhs(k);               % New Givens rotation on rhs
  rhs(k)=c(k)*rhs(k);
  res(k+1)=abs(rhs(k+1));              % Compute the new norm of the residual
  k=k+1;                               % Update iteration index
  if nargout>1
    y=R\rhs';                          % Compute the coefficients y_j
    uk(:,k)=u0+Q(:,1:end-1)*y;         % Compute u as sum_j y_j q_j
  end
end
y=R\rhs';                              % Compute the coefficients y_j
u=u0+Q(:,1:end-1)*y;                   % Compute u as u0+sum_j y_j q_j
```

Comparing the MATLAB functions `GMRES` and `PGMRES` the reader can easily see the difference: the (inverse) preconditioner M^{-1} is applied to the vector $\boldsymbol{r}_0$ before the GMRES loop and to the vector $\boldsymbol{q}_{k+1}$ only once per iteration.

Consider now a case where we would like to use a stationary method

$$\boldsymbol{u}_{k+1} = M^{-1}N\boldsymbol{u}_k + M^{-1}\boldsymbol{f}$$

as preconditioner, but we do not know explicitly M[2]. Using a stationary method means however that one knows at least (in form of a program) the function

$$(A, \boldsymbol{f}, \boldsymbol{u}_0, k) \mapsto \boldsymbol{u}_k = stationary(A, \boldsymbol{f}, \boldsymbol{u}_0, k)$$

that for a given matrix A, right-hand side $\boldsymbol{f}$ and initial guess $\boldsymbol{u}_0$ allows us to compute the approximation $\boldsymbol{u}_k$ at iteration k of $A^{-1}\boldsymbol{f}$. In MATLAB, this means that a function of the form

```
function uk=stationary(A,f,u0,k)
```

is available. Notice that `stationary` can represent any stationary method, e.g., Jacobi or Gauss-Seidel. Now, one can easily define the function $\boldsymbol{w} \mapsto g(\boldsymbol{w})$ as

```
g(w)=@(w) stationary(A,w,0,1),
```

which means that

$$g(\boldsymbol{w}) = stationary(A, \boldsymbol{w}, 0, 1) = M^{-1}\boldsymbol{w},$$

exactly the action of M^{-1} on $\boldsymbol{w}$! This means that providing the function g to the Krylov method is enough for the preconditioning, and the explicit matrix M is not needed! The same can be done also for the matrix A: it is sufficient to have a function $\boldsymbol{w} \mapsto g_A(\boldsymbol{w}) = A\boldsymbol{w}$. In other words, we can use the Krylov method in a *matrix-free* way. A matrix-free implementation of GMRES can be easily obtained by modifying few lines in the previous `PGMRES` function:

[2]This arises e.g. in domain decomposition methods and multigrid methods, where we have a code that computes $\boldsymbol{u}_{k+1}$ from $\boldsymbol{u}_k$, but no direct access to the matrix components involved in these computations, see Sections 4.6, 4.7, 4.8, and 4.10.

```
function [u,uk,res]=PGMRESMFREE(gA,f,u0,g,tol,m)
% PGMRESMFREE: Matrix-free Preconditioned Generalized Minimal Residual
%   [u,uk,res]=PGMRESMFREE(gA,f,u0,g,tol,m) solves g(gA(u))=g(f) using the
%   GMRES method starting at the initial guess u0 up to a tolerance tol
%   using at most m iterations.
%   gA and g are two functions that correspond to the actions of the
%   matrices A and M^-1 on a vector v: gA(v)=A*v and g(v)=M^-1*v.
%   PGMRESMFREE computes u in the (affine) preconditioned
%   Krylov space by minimizing the norm ||M^-1f-M^-1A*u||_2 where k is
%   the smallest integer such that ||M^-1f-M^-1A*u||_2<tol.
%   PGMRESMFREE returns in the matrix uk the iterates, in u the solution
%   computed, and in res the history of the norm of the residuals.

if nargin<6, m=100; end            % default values
if nargin<5, tol=1e-6; end
if nargin<4, g=@(v) v; end
k=1;                               % Initialize the iteration index
r0=f-gA(u0);                       % Compute the initial residual
r0=g(r0);                          % Apply the preconditioner to r0
res=norm(r0);                      % Initialize the vercor of residuals
rhs=res;                           % Initialize the right-hand side
Q(:,1)=r0/res;                     % First column of Q
while res(end)/norm(b)>tol && k<=m % GMRES iterations
  Q(:,k+1)=gA(Q(:,k));             % PART 1: Arnoldi (M\A*Q_k=Q_{k+1}*H_k)
  Q(:,k+1)=g(Q(:,k+1));            % Apply the preconditioner
  for j=1:k                        % Gram-Schmidt on v=M\A*Q(:,k)
    H(j,k)=Q(:,k+1)'*Q(:,j);
    Q(:,k+1)=Q(:,k+1)-H(j,k)*Q(:,j);
  end
  H(k+1,k)=norm(Q(:,k+1));         % Lower-diagonal element
  Q(:,k+1)=Q(:,k+1)/H(k+1,k);      % Compute the vector Q(:,k+1)
  R(k+1,k)=0;                      % PART 2: QR by Givens rotations
  R(:,k)=H(:,k);                   % New column of R
  for j = 1:k-1                    % Former Givens rotations on last column
    Rk=c(j)*R(j,k)+s(j)*R(j+1,k);
    R(j+1,k)=-s(j)*R(j,k)+c(j)*R(j+1,k);
    R(j,k)=Rk;
  end                              % Apply the new Givens rotation
  c(k)=R(k,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  s(k)=R(k+1,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  R(k,k)=c(k)*R(k,k)+s(k)*R(k+1,k);
  R(k+1,k)=0;                      % PART 3: Update rhs and residuals
  rhs(k+1)=-s(k)*rhs(k);           % New Givens rotation on rhs
  rhs(k)=c(k)*rhs(k);
  res(k+1)=abs(rhs(k+1));          % Compute the new norm of the residual
  k=k+1;                           % Update iteration index
  if nargout>1
    y=R\rhs';                      % Compute the coefficients y_j
    uk(:,k)=u0+Q(:,1:end-1)*y;     % Compute u as sum_j y_j q_j
  end
```

```
end
y=R\rhs';                          % Compute the coefficients y_j
u=u0+Q(:,1:end-1)*y;               % Compute u as sum_j y_j q_j
```

One can also use GMRES to solve directly the preconditioned system. To see this, we recall that

$$stationary(A, 0, \boldsymbol{w}, 1) = M^{-1}N\boldsymbol{w},$$

and $M^{-1}A = M^{-1}(M - N) = I - M^{-1}N$, we can easily obtain the action of the preconditioned matrix as

$$g_{M^{-1}A}(\boldsymbol{w}) = \boldsymbol{w} - stationary(A, 0, \boldsymbol{w}, 1) = M^{-1}A\boldsymbol{w}.$$

Therefore, we can feed GMRES with $g_{M^{-1}A}$ (instead of g_A) and without any explicit preconditioning function. This trick allows us to apply GMRES 'directly' to the preconditioned system.

Let us now answer the second question posed at the beginning of this section. Assume that the matrix A and the preconditioner M are symmetric and positive definite. The following theorem shows that the explicit computation of the decomposition $M = LL^\top$ is not necessary in practice. The conjugate gradient method can be easily modified in a way that only the solution of a system $M\boldsymbol{v} = \boldsymbol{w}$ (hence the action of M^{-1} over a vector $\boldsymbol{w}$) is required.

Theorem 33 (Preconditioned conjugate gradient)**.** *Let A and M be symmetric and positive definite and assume that M is decomposed as $M = LL^\top$. Denote by $\{\widetilde{\boldsymbol{u}}_k\}_k$, $\{\widetilde{\boldsymbol{r}}_k\}_k$ and $\{\widetilde{\boldsymbol{p}}_k\}_k$ the sequences of approximations, residuals and directions generated by CG applied to the system $\widetilde{A}\widetilde{\boldsymbol{u}} = \widetilde{\boldsymbol{f}}$, where $\widetilde{A} = L^{-1}AL^{-\top}$, $\widetilde{\boldsymbol{u}} = L^\top\boldsymbol{u}$ and $\widetilde{\boldsymbol{f}} = L^{-1}\boldsymbol{f}$. Consider the sequences $\{\boldsymbol{u}_k\}_k$, $\{\boldsymbol{r}_k\}_k$ and $\{\boldsymbol{p}_k\}_k$ defined as $\boldsymbol{u}_k := L^{-\top}\widetilde{\boldsymbol{u}}_k$, $\boldsymbol{r}_k := L\widetilde{\boldsymbol{r}}_k$ and $\boldsymbol{p}_k := L^{-\top}\widetilde{\boldsymbol{p}}_k$, for $k = 0, 1, \ldots$ These sequences satisfy the relations*

$$\boldsymbol{p}_0 = M^{-1}\boldsymbol{r}_0, \tag{4.10}$$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \alpha_k\boldsymbol{p}_k, \tag{4.11}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k, \tag{4.12}$$

$$\boldsymbol{p}_{k+1} = M^{-1}\boldsymbol{r}_{k+1} + \beta_{k+1}\boldsymbol{p}_k, \tag{4.13}$$

$$\alpha_{k+1} = \frac{(M^{-1}\boldsymbol{r}_{k+1})^\top\boldsymbol{r}_{k+1}}{\|\boldsymbol{p}_{k+1}\|_A^2}, \tag{4.14}$$

$$\beta_{k+1} = \frac{(M^{-1}\boldsymbol{r}_{k+1})^\top\boldsymbol{r}_{k+1}}{(M^{-1}\boldsymbol{r}_k)^\top\boldsymbol{r}_k}, \tag{4.15}$$

for $k = 0, 1, \ldots$. The method obtained by (4.10)-(4.15) is called the preconditioned conjugate gradient method.

Proof. The CG method applied to the system $\widetilde{A}\widetilde{\boldsymbol{u}} = \widetilde{\boldsymbol{f}}$ produces the iterates

(see Theorem 21)

$$\widetilde{\boldsymbol{p}}_0 = \widetilde{\boldsymbol{r}}_0, \tag{4.16}$$

$$\widetilde{\boldsymbol{u}}_{k+1} = \widetilde{\boldsymbol{u}}_k + \alpha_k \widetilde{\boldsymbol{p}}_k, \tag{4.17}$$

$$\widetilde{\boldsymbol{r}}_{k+1} = \widetilde{\boldsymbol{r}}_k - \alpha_k A\widetilde{\boldsymbol{p}}_k, \tag{4.18}$$

$$\widetilde{\boldsymbol{p}}_{k+1} = \widetilde{\boldsymbol{r}}_{k+1} + \beta_{k+1}\widetilde{\boldsymbol{p}}_k, \tag{4.19}$$

$$\alpha_{k+1} = \frac{(\widetilde{\boldsymbol{r}}_{k+1})^\top \widetilde{\boldsymbol{r}}_{k+1}}{\|\widetilde{\boldsymbol{p}}_{k+1}\|_A^2}, \tag{4.20}$$

$$\beta_{k+1} = \frac{(\widetilde{\boldsymbol{r}}_{k+1})^\top \widetilde{\boldsymbol{r}}_{k+1}}{(\widetilde{\boldsymbol{r}}_k)^\top \widetilde{\boldsymbol{r}}_k}. \tag{4.21}$$

To get (4.10) we use the definitions of $\boldsymbol{r}_k$ and $\boldsymbol{p}_k$ together with (4.16),

$$\boldsymbol{p}_0 = L^{-\top}\widetilde{\boldsymbol{p}}_0 = L^{-\top}\widetilde{\boldsymbol{r}}_0 = L^{-\top}L^{-1}\boldsymbol{r}_0 = M^{-1}\boldsymbol{r}_0.$$

Using the definitions of $\boldsymbol{u}_k$ and $\boldsymbol{p}_k$ together with (4.17), we get

$$\boldsymbol{u}_{k+1} = L^{-\top}\widetilde{\boldsymbol{u}}_{k+1} = L^{-\top}\widetilde{\boldsymbol{u}}_k + \alpha_k L^{-\top}\widetilde{\boldsymbol{p}}_k = \boldsymbol{u}_k + \alpha_k \boldsymbol{p}_k,$$

which is (4.11). The relations (4.12), (4.13), (4.14) and (4.15) are obtained by similar calculations. □

For an implementation of the preconditioned CG method see Problem 36.

4.4 Flexible GMRES: FGMRES

> *In order to be able to enhance robustness of iterative solvers, we should be able to determine, e.g., by means of heuristics, whether or not a given preconditioner is suitable for the problem at hand. If not one can attempt another possible iterative method/preconditioner and switch periodically if necessary. It is desirable to be able to switch within the outer iteration instead of restarting.*
>
> *Yousef Saad*, A Flexible Inner-Outer Preconditioned GMRES Algorithm, 1993.

Yousef Saad, the inventor of GMRES, proposed in [109] a generalization of GMRES which permits the change of the preconditioner in each iteration, see also the quote above. Different preconditioners can have very different properties, which could be all desirable when attempting to solve efficiently a large linear system. The idea of Saad opened the path for a new field of research called *multi-preconditioning*, which is still an active area of investigation.

To allow the change of the preconditioner in the course of the iterations, Saad proposed in [109] a very interesting strategy, which is a simple modification of GMRES. To explain the *Flexible GMRES* method (FGMRES), let us consider a right-preconditioned system

$$AM^{-1}(M\boldsymbol{u}) = \boldsymbol{f},$$

where both the matrices A and M are assumed to be invertible. The GMRES method applied to this system is given by the algorithm:

1. Input $\boldsymbol{u}_0$ (initial guess), ϵ (tolerance)
2. Compute $\boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$ and $\boldsymbol{q}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$ and set $k = 1$.
3. Compute $\boldsymbol{z}_k = M^{-1}\boldsymbol{q}_k$.
4. Compute $\boldsymbol{w} = A\boldsymbol{z}_k$.
5. For $j = 1, \ldots, k$ do (modified Gram-Schmidt)
   - Compute $h_{j,k} = \boldsymbol{w}^\top \boldsymbol{q}_j$.
   - Update $\boldsymbol{w} = \boldsymbol{w} - h_{j,k}\boldsymbol{q}_j$.
6. Compute $h_{k+1,k} = \|\boldsymbol{w}\|_2$ and $\boldsymbol{q}_{k+1} = \boldsymbol{w}/h_{k+1,k}$.
7. Define $Q_k = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k]$.
8. Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - H_k\boldsymbol{y}\|_2$.
9. Compute $\boldsymbol{u}_k = \boldsymbol{u}_0 + M^{-1}Q_k\boldsymbol{y}_k$ and $\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u}_k$.
10. If $\|\boldsymbol{r}_k\|_2 > \epsilon$, update $k = k + 1$ and go to Step 3.

Since the idea of FMGRES is to change the preconditioner in the course of the iterations, the Step 3 in the algorithm above is replaced by

$$\boldsymbol{z}_k = M_k^{-1}\boldsymbol{q}_k,$$

where now the preconditioner is indexed by k. This simple modification allows one to change the preconditioner at each GMRES iteration. Moreover, together with storing the vectors $\boldsymbol{q}_j$, FGMRES stores also the vectors $\boldsymbol{z}_j$. Hence, we need to modify Step 7, by defining $Z_k = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k]$, and Step 9, by computing $\boldsymbol{u}_k = \boldsymbol{u}_0 + Z_k\boldsymbol{y}_k$. This leads to the FGMRES algorithm, where the three modifications are underlined:

1. Input $\boldsymbol{u}_0$ (initial guess), ϵ (tolerance)
2. Compute $\boldsymbol{r}_0 = \boldsymbol{f} - A\boldsymbol{u}_0$ and $\boldsymbol{q}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$ and set $k = 1$.
3. Compute $\underline{\boldsymbol{z}_k = M_k^{-1}\boldsymbol{q}_k}$.
4. Compute $\boldsymbol{w} = A\boldsymbol{z}_k$.
5. For $j = 1, \ldots, k$ do (modified Gram-Schmidt)
   - Compute $h_{j,k} = \boldsymbol{w}^\top \boldsymbol{q}_j$.
   - Update $\boldsymbol{w} = \boldsymbol{w} - h_{j,k}\boldsymbol{q}_j$.
6. Compute $h_{k+1,k} = \|\boldsymbol{w}\|_2$ and $\boldsymbol{q}_{k+1} = \boldsymbol{w}/h_{k+1,k}$.
7. Define $\underline{Z_k = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k]}$.
8. Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - H_k\boldsymbol{y}\|_2$.
9. Compute $\underline{\boldsymbol{u}_k = \boldsymbol{u}_0 + Z_k\boldsymbol{y}_k}$ and $\boldsymbol{r}_k = \boldsymbol{f} - A\boldsymbol{u}_k$.
10. If $\|\boldsymbol{r}_k\|_2 > \epsilon$, update $k = k + 1$ and go to Step 3.

An implementation of FGMRES is given by the MATLAB function

```
function [u,uk,res]=FGMRES(A,f,u0,M,tol,m)
% FGMRES: Flexible Generalized Minimal Residual
%   [u,uk,res]=FGMRES(A,f,u0,M,tol,m) solves Au=f using the FGMRES
%   method starting at the initial guess u0 up to a tolerance tol using
%   at most m iterations. It computes u by changing cyclically the
%   (right) preconditioner at each iteration. The preconditioners are
%   contained in a cell M. For example given a matrix A, one can have
%   M{1}=diag(diag(A))  (Jacobi preconditioner)
%   M{2}=tril(A)        (Gauss-Seidel preconditioner)
%   M{3}=eye(size(A,1)) (no preconditioner)
%   FGMRES returns in the matrix uk the iterates, in u the solution
%   computed, and in res the history of the norm of the residuals.

if nargin<6, m=100; end            % default values
if nargin<5, tol=1e-6; end
num_p=length(M);                   % Compute the number of preconditioners
k=1;                               % Initialize the iteration index
r0=f-A*u0;                         % Compute the initial residual
res=norm(r0);                      % Initialize the vercor of residuals
rhs=res;                           % Initialize the right-hand side
Q(:,1)=r0/res;                     % First column of Q
Z=[];                              % Initialize matrix Z
ind=0;                             % Set preconditioner index ind=0
while res(end)>tol && k<=m         % FGMRES iterations
  if ind>=num_p || ind==0          % Reset/increase the preconditioner index
      ind=1;
  else
      ind=ind+1;
  end
  Z(:,k)=M{ind}\Q(:,k);            % PART 1: Arnoldi-apply ind-th preconditioner
  Q(:,k+1)=A*Z(:,k);               % Apply the matrix A
  for j = 1:k                      % modified Gram-Schmidt
    H(j,k)=Q(:,k+1)'*Q(:,j);
    Q(:,k+1)=Q(:,k+1)-H(j,k)*Q(:,j);
  end
  H(k+1,k)=norm(Q(:,k+1));         % Lower-diagonal element
  Q(:,k+1)=Q(:,k+1)/H(k+1,k);      % Compute the vector Q(:,k+1)
  R(k+1,k)=0;                      % PART 2: QR by Givens rotations
  R(:,k)=H(:,k);                   % New column of R
  for j=1:k-1                      % Former Givens rotations on last column
    Rk=c(j)*R(j,k)+s(j)*R(j+1,k);
    R(j+1,k)=-s(j)*R(j,k)+c(j)*R(j+1,k);
    R(j,k)=Rk;
  end                              % Next apply the new Givens rotation
  c(k)=R(k,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  s(k)=R(k+1,k)/sqrt(R(k,k)^2+R(k+1,k)^2);
  R(k,k)=c(k)*R(k,k)+s(k)*R(k+1,k);
  R(k+1,k)=0;
  rhs(k+1)=-s(k)*rhs(k);           % PART 3: Update rhs and residuals
  rhs(k)=c(k)*rhs(k);              % New Givens rotation on rhs
```

```
  res(k+1)=abs(rhs(k+1));          % Compute the new norm of the residual
  k=k+1;                           % Update iteration index
  if nargout>1
    y=R\rhs';                      % Compute the coefficients y_j
    uk(:,k)=u0+Z*y;                % Compute u as u0+sum_j y_j z_j
  end
end
y=R\rhs';                          % Compute the coefficients y_j
u=u0+Z*y;                          % Compute u as u0+sum_j y_j z_j
end
```

The main important difference between the standard GMRES and FGMRES is that the classical Arnoldi-type relation

$$(AM^{-1})Q_k = Q_{k+1}H_k \tag{4.22}$$

is replaced by the more general equality

$$AZ_k = Q_{k+1}H_k. \tag{4.23}$$

Clearly, H_k is a $k+1 \times k$ upper Hessenberg matrix and the two above relations coincide if $M_k = M$ for all k. Similarly to (3.46) for (4.22), an alternative to (4.23) is

$$AZ_k = Q_k\widehat{H}_k + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^\top, \tag{4.24}$$

where $\widehat{H}_k$ is the $k \times k$ matrix obtained from H_k by deleting the last row.

FGMRES satisfies an optimality property similar to the one proved for GMRES in Theorem 26.

Theorem 34 (Optimality of FGMRES, Saad 1993)**.** *The approximation $\boldsymbol{u}_k$ computed by FGMRES at step k is the solution to the residual minimization problem*

$$\min_{\widetilde{\boldsymbol{u}} \in \boldsymbol{u}_0 + \operatorname{span}\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_k\}} \|\boldsymbol{f} - A\widetilde{\boldsymbol{u}}\|_2.$$

Proof. For any $\boldsymbol{w} \in \boldsymbol{u}_0 + \operatorname{span}\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$ we write

$$\begin{aligned} \boldsymbol{f} - A\boldsymbol{w} &= \boldsymbol{f} - A(\boldsymbol{u}_0 + Z_k\boldsymbol{y}) \\ &= \boldsymbol{r}_0 - AZ_k\boldsymbol{y} \\ &= \|\boldsymbol{r}_0\|_2\boldsymbol{q}_1 - Q_{k+1}H_k\boldsymbol{y} \\ &= Q_{k+1}\big(\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - H_k\boldsymbol{y}\big), \end{aligned} \tag{4.25}$$

where we used (4.23) to get the third equality. The result follows by recalling that $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - H_k\boldsymbol{y}\|_2$ in Step 8 of FGMRES. □

We now study the possible breakdown of the FGMRES method, which occurs if $h_{k+1,k} = 0$. This situation is not a problem for the classical GMRES method, because a breakdown occurs only if GMRES converged to the exact solution; see Remark 6. This is not in general the case for FGMRES, where the invertibility

of A does not necessarily imply that $\widehat{H}_k$ has full rank, since the matrix Z_k is not necessarily full rank[3].

Theorem 35 (Breakdown of FGMRES, Saad 1993)**.** *Assume that $\|\boldsymbol{r}_0\|_2 \neq 0$ and that FGMRES performed successfully the first $k-1$ iterations with $h_{j+1,j} \neq 0$ for $j < k$. In addition, assume that $\widehat{H}_k$ has full rank. Then $\boldsymbol{u}_k$ is the exact solution if and only if $h_{k+1,k} = 0$.*

Proof. Assume first that $h_{k+1,k} = 0$. Using the relation (4.24), we obtain that $AZ_k = Q_k\widehat{H}_k$. This and the equality (4.25) allow us to write

$$\begin{aligned}\|\boldsymbol{f} - A\boldsymbol{u}_k\|_2 &= \|\|\boldsymbol{r}_0\|_2\boldsymbol{q}_1 - AZ_k\boldsymbol{y}_k\|_2 \\ &= \|\|\boldsymbol{r}_0\|_2\boldsymbol{q}_1 - Q_k\widehat{H}_k\boldsymbol{y}_k\|_2 \\ &= \|\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - \widehat{H}_k\boldsymbol{y}_k\|_2.\end{aligned}$$

Since $\widehat{H}_k$ is invertible, the residual norm is minimized by $\boldsymbol{y}_k = \|\boldsymbol{r}_0\|_2\widehat{H}_k^{-1}\boldsymbol{e}_1$, which gives $\|\boldsymbol{f} - A\boldsymbol{u}_k\|_2 = 0$.

Assume now that $\boldsymbol{u}_k$ coincides with the exact solution $\boldsymbol{u}$, then

$$\begin{aligned}0 &= \boldsymbol{f} - A\boldsymbol{u}_k \\ &= Q_k\big[\|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 - \widehat{H}_k\boldsymbol{y}_k\big] + h_{k+1,k}\boldsymbol{q}_{k+1}\boldsymbol{e}_k^\top\boldsymbol{y}_k,\end{aligned} \tag{4.26}$$

where $\boldsymbol{e}_k$ is the kth canonical vector in $\mathbb{R}^k$. If the last entry of $\boldsymbol{y}_k$ is zero, then $\boldsymbol{e}_k^\top\boldsymbol{y}_k = 0$. This implies that $\widehat{H}_k\boldsymbol{y}_k = \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1$. Since $h_{j+1,j} \neq 0$ for $j < k$, then a simple back substitution starting from the last equation of $\widehat{H}_k\boldsymbol{y}_k = \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1$ would show that $\boldsymbol{y}_k = 0$, which would then imply $\|\boldsymbol{r}_0\|_2 = 0$, contradicting the assumption. Hence $\boldsymbol{e}_k^\top\boldsymbol{y}_k \neq 0$. Therefore, since $\boldsymbol{q}_{k+1}$ is orthogonal to $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k$, the only way in which (4.26) can be satisfied is that

$$\widehat{H}_k\boldsymbol{y}_k = \|\boldsymbol{r}_0\|_2\boldsymbol{e}_1 \text{ and } \boldsymbol{q}_{k+1} = 0,$$

which implies that $h_{k+1,k} = 0$. □

4.5 Algebraic preconditioning methods

A particular class of regular splittings of not necessarily symmetric M-matrices is proposed. If the matrix is symmetric, this splitting is combined with the conjugate-gradient method to provide a fast iterative solution algorithm.

J. A. Meijerink and Henk A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-Matrix, 1977.

[3]Consider a matrix $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2] = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}$ and the two preconditioners $M_1 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ and $M_2 = \begin{bmatrix}\frac{1}{2} & -\frac{1}{2}\\ 0 & 1\end{bmatrix}$. Then a simple calculation reveals that $Z = [M_1^{-1}\boldsymbol{q}_1, M_2^{-1}\boldsymbol{q}_2] = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$.

In Section 2.6 we have studied the classical version of the Gauss-Seidel method, also known as *forward Gauss-Seidel*, which is based on the splitting matrices $M = (D + L)$ and $N = -U$. It is also possible to consider a *backward Gauss-Seidel* method obtained by $M = (D+U)$ and $N = -L$. Combining these two steps, we obtain the *symmetric Gauss-Seidel* method,

$$(D + L)\boldsymbol{u}_{k+\frac{1}{2}} = -U\boldsymbol{u}_k + \boldsymbol{f}, \tag{4.27}$$

$$(D + U)\boldsymbol{u}_{k+1} = -L\boldsymbol{u}_{k+\frac{1}{2}} + \boldsymbol{f}, \tag{4.28}$$

which will lead us naturally to the so-called *Incomplete LU (ILU)* preconditioner [108]. Using the first equation (4.27), we have $\boldsymbol{u}_{k+\frac{1}{2}} = -(D+L)^{-1}U\boldsymbol{u}_k + (D+L)^{-1}\boldsymbol{f}$. Replacing this into the second equation (4.28), we obtain

$$\boldsymbol{u}_{k+1} = (D+U)^{-1}L(D+L)^{-1}U\boldsymbol{u}_k - (D+U)^{-1}L(D+L)^{-1}\boldsymbol{f} + (D+U)^{-1}\boldsymbol{f}.$$

Defining $M := \Big((D+U)^{-1} - (D+U)^{-1}L(D+L)^{-1}\Big)^{-1}$, we obtain the usual stationary iteration relation

$$\boldsymbol{u}_{k+1} = (I - M^{-1}A)\boldsymbol{u}_k + M^{-1}\boldsymbol{f},$$

and we have found the symmetric Gauss-Seidel preconditioner M, whose inverse can be rewritten in the form

$$\begin{aligned} M^{-1} &= (D+U)^{-1} - (D+U)^{-1}L(D+L)^{-1} \\ &= (D+U)^{-1} - (D+U)^{-1}(L+D-D)(D+L)^{-1} \\ &= (D+U)^{-1} - (D+U)^{-1}\big(I - D(D+L)^{-1}\big) \\ &= (D+U)^{-1}D(D+L)^{-1} = \widetilde{U}^{-1}\widetilde{L}^{-1}, \end{aligned}$$

where $\widetilde{U} := D^{-1}(D+U)$ is upper triangular, and $\widetilde{L} := D+L$ is lower triangular. We see that the entries $\widetilde{U}_{ij}$ and $\widetilde{L}_{ij}$ are zero if the corresponding entries A_{ij} of the original matrix are zero. We thus obtained a preconditioner $M = \widetilde{L}\widetilde{U}$ given by its LU-factorization. To be a good preconditioner, $M = \widetilde{L}\widetilde{U}$ should be a good approximation of A, and naturally the question arises if it is possible to replace the entries in $\widetilde{L}$ and $\widetilde{U}$ chosen by symmetric Gauss Seidel by entries which make $M = \widetilde{L}\widetilde{U}$ an even better approximation of A. This is the idea behind the very popular Incomplete LU (ILU) preconditioner developed and studied independently by Varga [123] and Buleev [14] in 1960 and by Meijerink and van der Vorst in 1977, see [90].

In the original version of ILU, one allowed only non-zero entries in $\widetilde{L}$ and $\widetilde{U}$ where A had a non-zero entry, the structure that was indicated by symmetric Gauss-Seidel. These non-zero entries in $\widetilde{L}$ and $\widetilde{U}$ are then computed using Gaussian elimination:

```
function [L,U]=ILU0(A)
% ILU0 incomplete ILU factorization with zero fill-in
```

```
%   [L,U]=ILU0(A); computes an ILU factorization of the matrix
%   A with zero fill-in.

n=size(A,1);
for k=1:n-1
  for i=k+1:n
    if A(i,k)~=0
      A(i,k)=A(i,k)/A(k,k);
      for j=k+1:n
        if A(i,j)~=0
          A(i,j)=A(i,j)-A(i,k)*A(k,j);
        end
      end
    end
  end
end
L=speye(size(A))+tril(A,-1);
U=triu(A,0);
```

If we apply ILU as a preconditioner in the stationary iteration to our Laplace model problem from Section 1.3[4], using the MATLAB statements

```
m=15;                                   % number of gridpoints
A=Laplacian(m,2);                       % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;          % put bc into the rhs
u=A\f;                                  % solve by sparse Gaussian elimination
h=1/(m+1); x=0:h:1; y=x;                % mesh point vectors
un=zeros(size(f));                      % initial guess
UU=zeros(m+2); UU(end,1:m+2)=1;         % for plotting
[L,U]=ILU0(A);
for n=0:10
  err(n+1)=max(max(abs(u-un)));         % compute error
  UU(2:m+1,2:m+1)=reshape(un,m,m);
  mesh(x,y,UU)
  xlabel('x');ylabel('y');
  pause
  un=un+U\(L\(f-A*un));
end
```

we get for the first few iterates the approximations shown in Figure 4.1. We see that ILU leads to a more rapidly converging stationary iterative method, substantially better than the other stationary methods we have seen so far, and even better than the conjugate gradient method in this example. Note that we

[4]We use a stationary iterative method here so we can compare to the other stationary iterative methods we tested earlier, even though originally ILU was suggested to be used as a preconditioner for a Krylov method. Theorem 32 shows that this would work even better, but then the properties of ILU are mixed with the properties of the Krylov method, and it becomes harder to precisely understand the convergence mechanisms. We thus advocate the principle to always investigate preconditioners first as stationary methods, before accelerating them by a Krylov method.

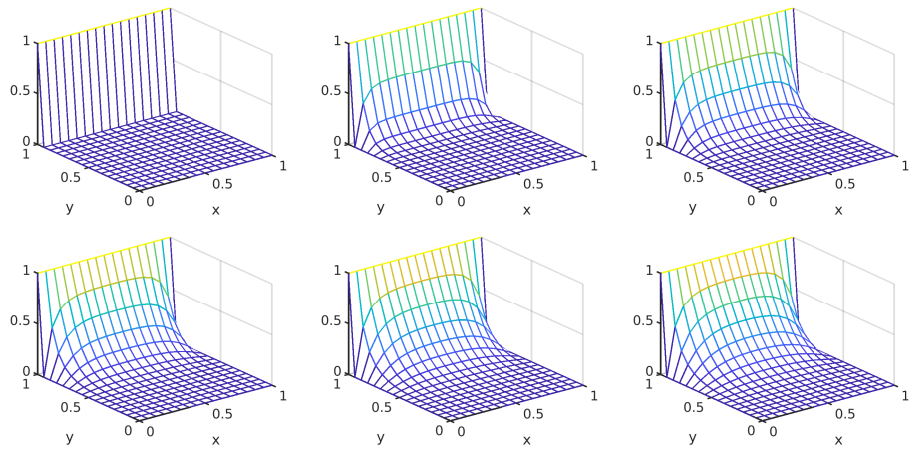


Figure 4.1: Initial guess and first iterations of ILU as a stationary iteration applied to our Laplace model problem from Section 1.3.

only show the first five iterates for ILU, as we will do for the other preconditioners in this chapter, in contrast to the earlier chapters where we showed the first 11 iterates. The reason for this superior performance of ILU is that all the earlier methods we studied are based on local operations: Jacobi, Gauss-Seidel, SOR and Richardson use only operations between neighboring grid points, and similarly the Krylov methods only use matrix vector multiplications, again interaction between neighboring grid points due to the sparsity of the finite difference Laplacian. This is why none of these methods could converge rapidly for the unknowns in the corresponding plots of the iterates for small y coordinates in Figures 2.3, 2.5, 2.8, 2.18, 3.3, and 3.7. ILU contains a true attempt to solve the problem by factorization, i.e. non-local information, and this is a key ingredient for a good preconditioner; all more sophisticated preconditioners like domain decomposition and multigrid in this chapter contain such components.

Nevertheless, also ILU leads to mesh dependent convergence, as we can see in Figure 4.2 where we see how the error decreases as the iterations progress when ILU is used as a stationary iteration to solve our Laplace model problem from Section 1.3 for the mesh we used for Figure 4.1 with $m = 15$ interior mesh points, and two refined meshes with $m = 31$ and $m = 63$ interior mesh points.

To improve ILU, one does not have to use the sparsity pattern of the underlying matrix to determine which positions to fill in: one can more generally first decide which entries of the matrices $\widetilde{L}$ and $\widetilde{U}$ can be non-zero, by means of the set

$$\mathcal{Z} := \{(i,j) \,:\, \widetilde{L}_{ij} \neq 0 \text{ or } \widetilde{U}_{ij} \neq 0\},$$

and then replace the corresponding condition on A in the algorithm by a condition using the set $\mathcal{Z}$. Meijerink and van der Vorst showed in [90] that if $\mathcal{Z}$ contains the diagonal, and the matrix A is an M-matrix, then the ILU algorithm

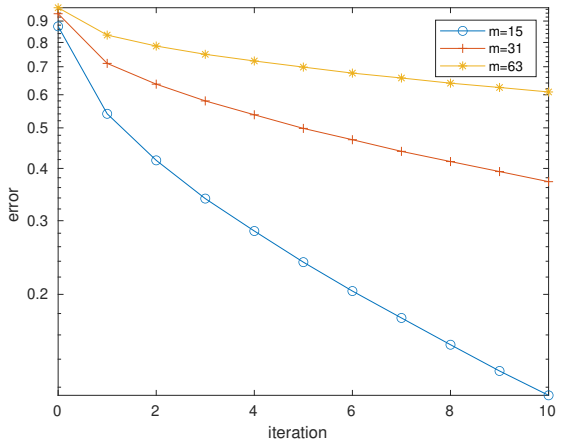


Figure 4.2: Convergence of the ILU stationary iteration for different mesh sizes $h = \frac{1}{m+1}$.

produces the incomplete factorization

$$A = \widetilde{L}\widetilde{U} - R,$$

which is a regular splitting of A, i.e. the splitting satisfies Definition 6. Therefore, the corresponding stationary iterative method converges according to Theorem 5.

In order to obtain an even better approximation of A, one can determine the elements of $\widetilde{L}$ and $\widetilde{U}$ to fill in during the factorization process using a tolerance. This is the idea behind ILUT(ϵ) which was introduced by Saad in 1994, see [110]. In this variant of ILU, Gaussian elimination is used, and elements are only stored in $\widetilde{L}$ and $\widetilde{U}$ if they are larger than the specified tolerance. There is no proof that ILUT(ϵ) terminates and leads to an approximate factorization, but it works very well in practice. All these variants of ILU are available in the MATLAB function `ilu`. We show in Figure 4.3 for our Laplace model problem from Section 1.3 how much fill-in is created in the L factor by the exact LU factorization, compared to ILU0 and ILUT(ϵ) with two values of ϵ. We can see that the exact LU factorization completely fills in the band between the tridiagonal blocks on the diagonal, and the off diagonal blocks of the discrete Laplacian, leading to 4111 non-zero entries in L, while ILU0 preserves the structure of the discrete Laplacian with only 736 non-zero elements in $\widetilde{L}$. With the tolerance 0.01, the band starts to fill-in a little, leading to 1326 non-zero elements in $\widetilde{L}$, and with the lower tolerance 0.002 the fill-in becomes more pronounced, with 2269 non-zero elements in $\widetilde{L}$, about half of the exact L. This fill-in has an important impact on the performance of the preconditioner obtained, as is illustrated in Figure 4.4. As expected, the exact LU factorization leads to a preconditioner such that the stationary iterative method converges in one iteration, but it is as expensive as solving the system directly using the LU factorization. ILU0

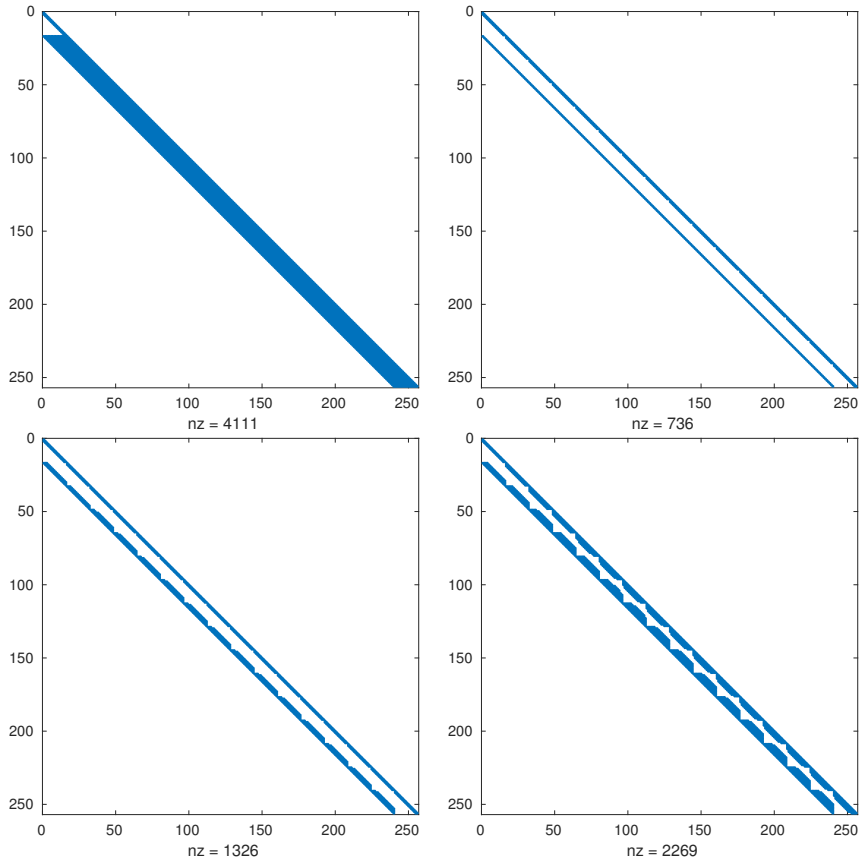


Figure 4.3: Fill-in in the L factor created by the exact LU factorization (top left), ILU0 (top right), and ILUT(0.01) (bottom left) and ILUT(0.002) (bottom right) for the five point finite difference Laplacian.

leads already to reasonably fast convergence, but with ILUT(ϵ) one can reach any convergence speed, if one is willing to pay enough with the fill-in. This is one of the great strength of ILUT: it is possible to obtain a preconditioner as close as one wants to the direct solver.

A slightly different, but related idea is to directly construct an approximate LU factorization of the inverse of the system matrix $A \in \mathbb{R}^{n\times n}$, which led to the *sparse Approximate Inverse (AINV) preconditioner* for symmetric positive definite matrices introduced by Benzi, Meyer and Tuma in 1996, see [8]. AINV is based on the fact that one can readily obtain a factorization of A^{-1} from a set of conjugate (A-orthogonal) search directions $\boldsymbol{z}_j$, $j = 1, 2, \ldots, n$: if one collects these search directions in the matrix $Z := [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n]$, then we, get because of A-orthogonality,

$$Z^\top AZ = \mathrm{diag}(p_1, \ldots, p_n), \quad p_i = \boldsymbol{z}_i^\top A\boldsymbol{z}_i,$$

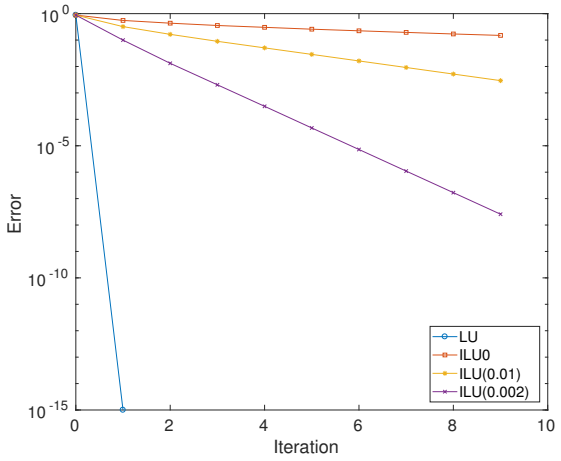


Figure 4.4: Performance of the different ILU preconditioners when used as stationary iterative solvers for our Laplace model problem.

and hence for the inverse

$$A^{-1} = ZD^{-1}Z^{\top}.$$

If one constructs the conjugate directions using the conjugate Gram-Schmidt process and starts with the canonical basis vectors $\boldsymbol{e}_i$, it turns out that the resulting Z matrix is upper triangular, $Z = L^{-\top}$ with $A = LDL^{\top}$ being the Cholesky factorization of A, see [8]. AINV is then obtained by only computing a sparse approximation of Z by either specifying a fill-in pattern or a drop tolerance like in ILU.

A different idea to obtain a preconditioner at the algebraic level is to construct directly a matrix M^{-1} which is a good approximation of A^{-1}. A well known such preconditioner is the *Sparse Approximate Inverse preconditioner (SPAI)*, which was introduced by Grote and Huckle in 1997[5], see [68]. Here, a sparse matrix M^{-1} is directly constructed by minimizing the Frobenius norm

$$\|I - AM^{-1}\|_F^2 := \sum_{i=1}^{n} \|(I - AM^{-1})\boldsymbol{e}_i\|_2^2,$$

which can be naturally done in parallel for each column, and leads to n independent least squares problems when a sparsity pattern is imposed on M^{-1}. Here is an implementation in MATLAB:

```
function M=SPAI(A,Z)
% SPAI computes a sparse approximate inverse
%   M=SPAI(A,Z); computes a sparse approximate inverse M of A with the
%   sparsity pattern in the matrix Z using the SPAI technique.
```

[5]The second author was sitting as a fresh graduate student with Marcus Grote and Thomas Huckle on the lawn at Stanford when SPAI was invented.

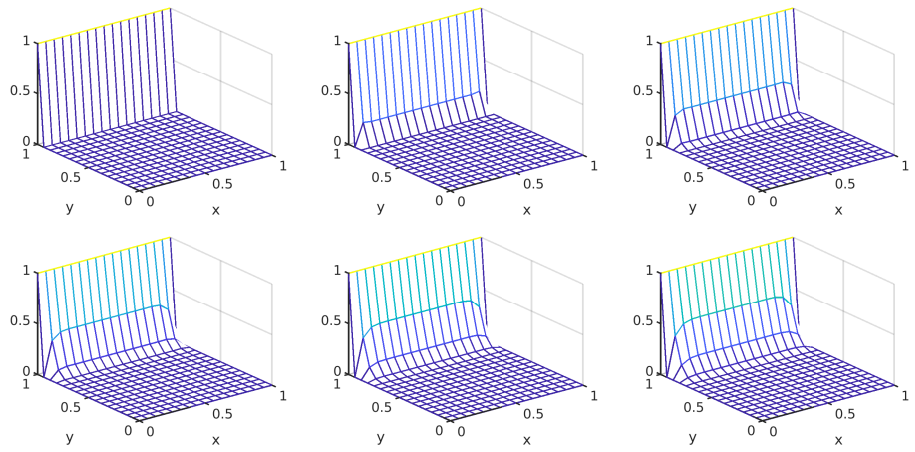


Figure 4.5: Initial guess and first iterations of the SPAI preconditioner with a diagonal sparsity pattern only, applied to our Laplace model problem from Section 1.3.

```
n=size(A,2);
I=speye(n);
M=Z;
for i=1:n
  id=find(Z(:,i)~=0);                    % find non-zero entries in Z
  M(id,i)=A(:,id)\I(:,i);                % fill them with least squares
end
```

If one puts as sparsity pattern `Z` a dense matrix, e.g. `M=SPAI(A,ones(size(A)))`, then SPAI computes actually the inverse of A, $M = A^{-1}$. If we give only a diagonal matrix, `M=SPAI(A,speye(size(A)))`, we obtain for our Laplace model problem when SPAI is used in a stationary iteration the first iterates in Figure 4.5. We see that even though this is the best possible diagonal approximation of the inverse in the SPAI sense, convergence is rather slow. To improve it, we give as sparsity pattern `Z` the matrix A, `M=SPAI(A,A)`. This leads to the iterates in Figure 4.6. We see that convergence is already better, and we can further improve it by giving a sparsity pattern which connects further neighbors, e.g. `M=SPAI(A,A^2)` with the results in Figure 4.7, and `M=SPAI(A,A^3)` with the results in Figure 4.8. We see that convergence is improving further and further, which is due to more and more fill-in created in the sparsity pattern by taking powers, as illustrated in Figure 4.9. Similar to ILU, one can thus obtain an arbitrarily good preconditioner, if one is willing to pay with more and more fill in. We illustrate this in Figure 4.10. As expected, using the completely dense pattern indicated by 'full', leads to a preconditioner such that the stationary iterative method converges in one iteration. Increasing the fill-in starting from the diagonal approximation with powers of the matrix improves the convergence, but comparing to the ILU approach in Figure 4.4 we see that convergence for a comparable fill-in in SPAI is much slower than for ILU (note the difference

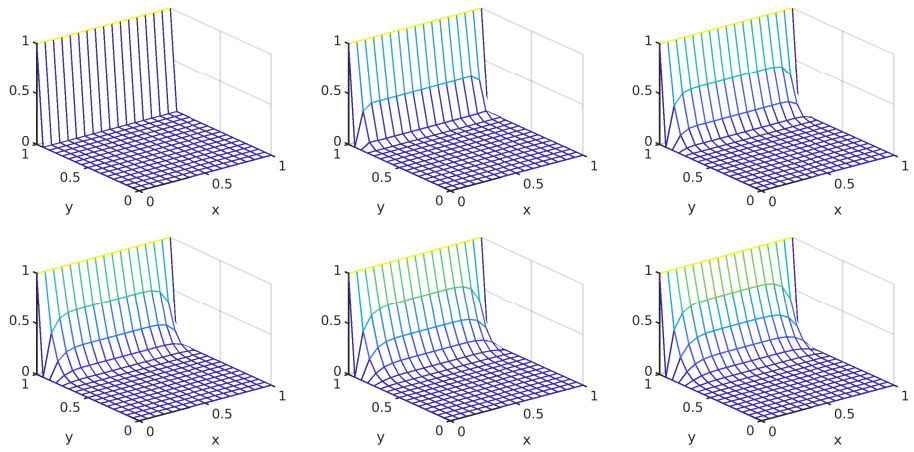


Figure 4.6: Initial guess and first iterations of the SPAI preconditioner with sparsity pattern corresponding to the sparsity pattern of the matrix A from our Laplace model problem in Section 1.3.

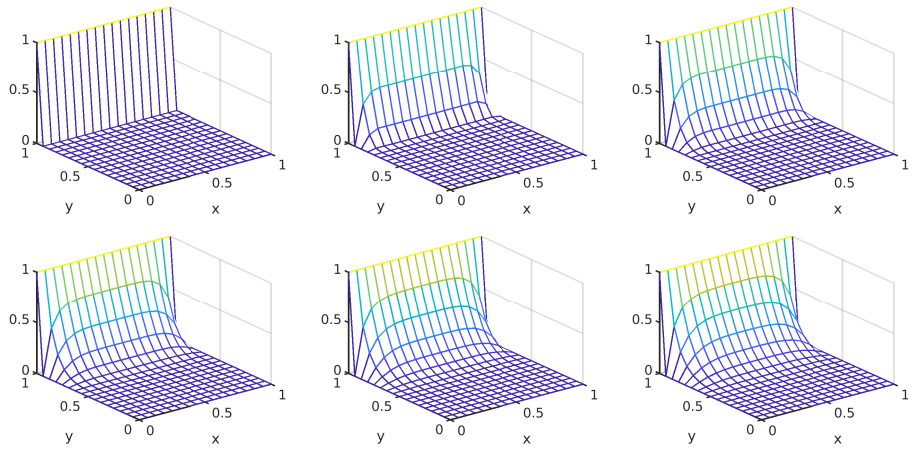


Figure 4.7: Initial guess and first iterations of the SPAI preconditioner with sparsity pattern corresponding to the sparsity pattern of the matrix A^2 from our Laplace model problem in Section 1.3.

in scale). This is because the exact inverse is fully dense, while the exact LU factorization only fills in the band width in our Laplace example, and so a much better approximation can be obtained in ILU when a similar number of fill-in elements is used compared to SPAI. SPAI convergence is also mesh dependent, as we can see in Figure 4.11, where we used as the fill-in pattern the sparsity pattern of A^3 when solving our Laplace model problem from Section 1.3 for the mesh we used for Figure 4.8 with $m = 15$ interior mesh points, and two refined

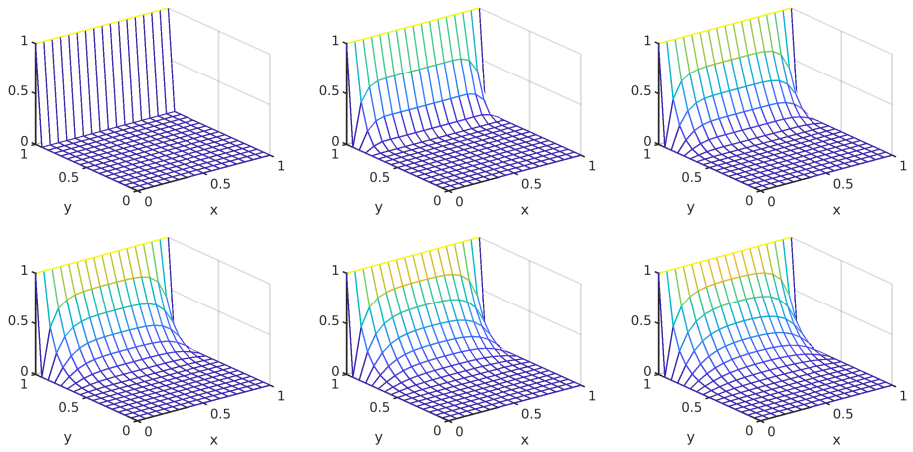


Figure 4.8: Initial guess and first iterations of the SPAI preconditioner with sparsity pattern corresponding to the sparsity pattern of the matrix A^3 from our Laplace model problem in Section 1.3.

meshes with $m = 31$ and $m = 63$ interior mesh points.

There is also an adaptive version of SPAI, where one starts with a given sparsity pattern, for example diagonal, and then one adds adaptively non-zero entries in M^{-1} in such a way that the residual is reduced as much as possible; see [68] and, e.g., [6].

4.6 Schwarz domain decomposition methods

> *Durch Fortsetzung einiger Untersuchungen, welche gewisse Arten von Abbildungsaufgaben betreffen, und von denen ein Theil im 70. Bande von Borchardt's Journal und in dem das Programm der eigenössischen poytechnischen Schule für das Wintersemester 1869-70 begleitenden Aufsatze: "Zur Theorie der Abbildung" veröffentlicht ist, bin ich auf ein Beweisverfahren geführt worden, durch welches, wie ich mich überzeugt zu haben glaube, alle Sätze, deren Beweis Riemann in seinen veröffentlichten Abhandlungen mittelst des Dirichletschen Princips zu führen gesucht hat, mit Strenge bewiesen werden können.*
>
> *Hermann Amandus Schwarz*, Ueber einen Grenzübergang durch alternierendes Verfahren, 1869.

We have seen in Chapter 1 that the linear system $A\boldsymbol{u} = \boldsymbol{f}$ often represents a discretization of a partial differential equation, e.g.,

$$\begin{aligned} \Delta u &= f \ \text{ in } \Omega := (0,1) \times (0,L), \\ u &= g \ \text{ on } \partial\Omega. \end{aligned} \tag{4.29}$$

This suggests that the continuous problem corresponding to $A\boldsymbol{u} = \boldsymbol{f}$ can be used to derive good preconditioners. In particular, if we can define a good stationary iterative method at the continuous level, we can construct a preconditioner by the corresponding discretization, and since the method already works at the

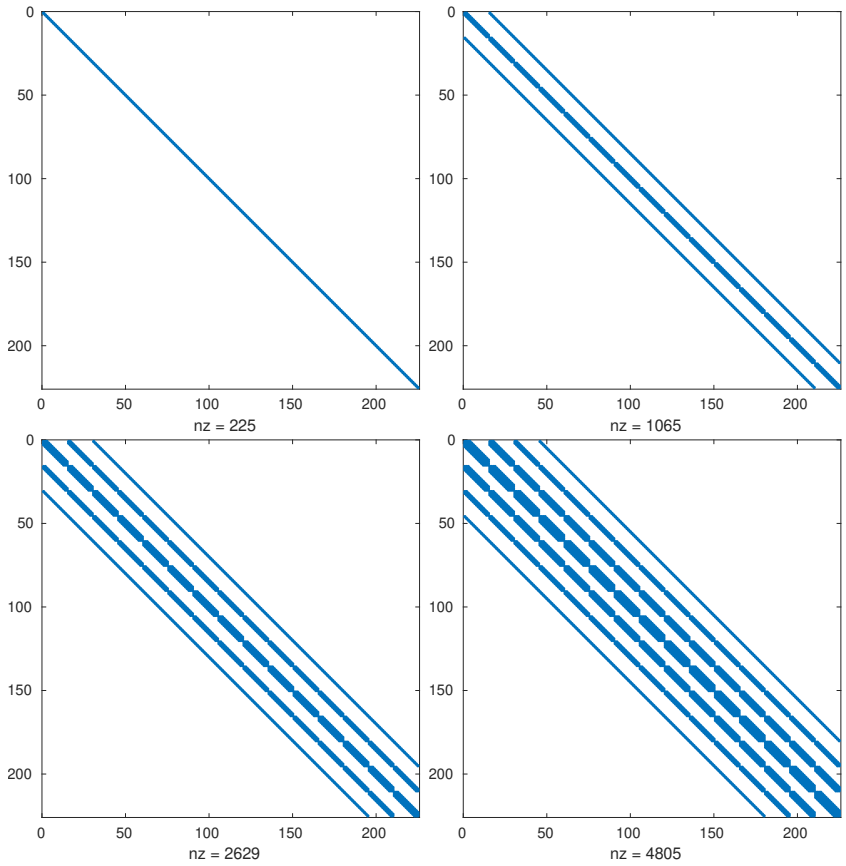


Figure 4.9: SPAI fill in patterns created by A^0, A^1, A^2 and A^3, where A is the five point finite difference stencil from our Laplace model problem in Section 1.3.

continuous level, it is very likely that the discretized method then converges independently of the mesh size, i.e. the number of iterations for a certain accuracy does not depend on the size of the problem. Domain decomposition methods follow exactly this idea.

Schwarz domain decomposition methods are the oldest domain decomposition methods, and in their classical variant they need overlapping subdomain decompositions to converge. Schwarz methods were first defined at the continuous level directly for the partial differential equation, and the goal of Schwarz was to close a gap in the proof of the Riemann mapping theorem, see the quote above, and [55] for more details on the historical context. This led to the *alternating Schwarz method*, which a century later was generalized by *Pierre-Louis Lions* to the *parallel Schwarz method*. These methods can be discretized and used as iterative solvers, but there are also discrete Schwarz methods, namely the *multiplicative Schwarz method*, the *additive Schwarz method*, and the *re-*

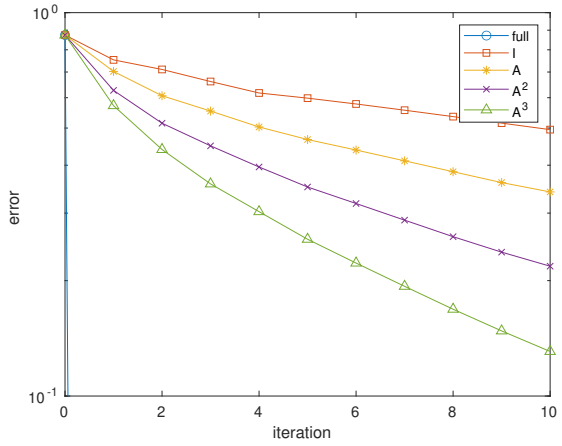


Figure 4.10: Performance of the different SPAI preconditioners when used as stationary iterative solvers for our Laplace model problem.

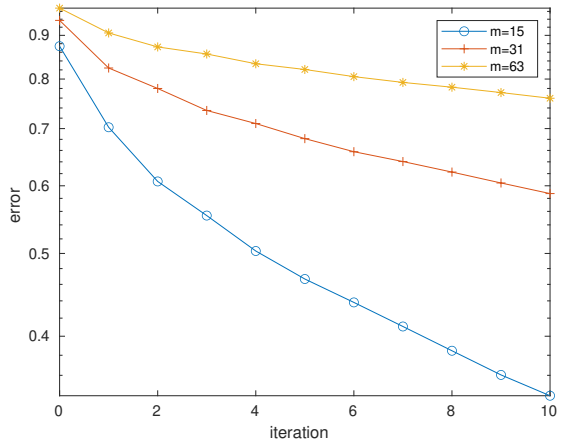


Figure 4.11: Convergence of the SPAI stationary iteration for different mesh sizes $h = \frac{1}{m+1}$.

stricted additive Schwarz method, and we will see why there are three discrete methods and only two classical continuous ones.

The classical Schwarz methods are overlapping domain decomposition methods, but there are also *non-overlapping domain decomposition* methods. Their roots lie in the substructuring methods of *Janusz S. Przemieniecki* [101], but they evolved rapidly into the Dirichlet-Neumann, Neumann-Neumann and FETI algorithms, and we will directly introduce and study the latter later in this section.

There are also domain decomposition methods for time-dependent problems, which have their roots in the work of *Emile Picard* [100] and *Ernest Lindelöf*

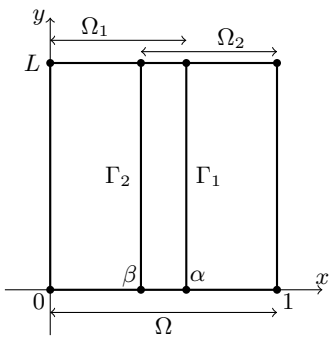


Figure 4.12: Decomposition of the domain $\Omega = (0,1) \times (0,L)$ into $\Omega = \Omega_1 \cup \Omega_2$, where $\Omega_1 := (0,\alpha) \times (0,L)$ and $\Omega_2 := (\beta,1) \times (0,L)$. The interfaces are denoted by Γ_1 and Γ_2.

[86], and became numerical solvers with the invention of *waveform relaxation methods* in the group of *Albert E. Ruehli* at IBM [84], but since we do not treat time-dependent problems in this book, we will not explain these methods further.

Classical alternating and parallel Schwarz methods

The first domain decomposition method in history is the alternating Schwarz method which was proposed by *Hermann Amandus Schwarz* in 1869, see [113]. To define the alternating Schwarz method for the solution to (4.29), we consider the simple domain decomposition $\Omega = \Omega_1 \cup \Omega_2$, where the two subdomains are $\Omega_1 := (0,\alpha) \times (0,L)$ and $\Omega_2 := (\beta,1) \times (0,L)$ with $\alpha > \beta$; see Figure 4.12. The two subdomains Ω_1 and Ω_2 have boundaries $\partial\Omega_1$ and $\partial\Omega_2$, and $\Gamma_1 := \partial\Omega_1 \backslash (\partial\Omega \cap \partial\Omega_1)$ and $\Gamma_2 := \partial\Omega_2 \backslash (\partial\Omega \cap \partial\Omega_2)$ are called interfaces. The *alternating Schwarz method* is defined by iteratively solving the two subproblems[6]

$$\begin{aligned} \Delta u_1^n &= f && \text{in } \Omega_1, & \Delta u_2^n &= f && \text{in } \Omega_2, \\ u_1^n &= u_2^{n-1} && \text{on } \Gamma_1, & u_2^n &= u_1^n && \text{on } \Gamma_2, \\ u_1^n &= g && \text{on } \partial\Omega \cap \partial\Omega_1, & u_2^n &= g && \text{on } \partial\Omega \cap \partial\Omega_2. \end{aligned} \tag{4.30}$$

In particular, the Schwarz iterative procedure starts with an initial guess u_2^0 on Γ_1, which is used to solve the first subproblem to get u_1^1. The function u_1^1 is then traced on Γ_2 to solve the second subproblem that gives u_2^1. Again, the function u_2^1 is traced on Γ_1 to solve the first subproblem and compute u_1^2. One can then trace u_1^2 on Γ_2, and solve the second subproblem to get u_2^2. Continuing in this way, it is possible to obtain two sequences $\{u_1^n\}_n$ and $\{u_2^n\}_n$ that approximate the solution u on Ω_1 and Ω_2. A small modification which leads to the *parallel*

[6] In contrast to the earlier chapters, it is more common to use n as the iteration index here, and to put it as superscript, a convention we follow throughout this chapter.

Schwarz method was introduced by Lions in [87]: instead of using on the second subdomain in (4.30) $u_2^n = u_1^n$, he proposed to use

$$u_2^n = u_1^{n-1} \qquad \text{on } \Gamma_2, \tag{4.31}$$

which then allows the two subdomain solves to be done in parallel, before exchanging information with the neighboring subdomain. In the case of many subdomains, this leads to a large scale parallel method which is of great interest. In the two subdomain case however, this modification just leads to a method which computes the original alternating Schwarz sequence $\{u_2^0, u_1^1, u_2^2, u_1^3, u_2^4, \dots\}$, and the analog one starting on the other subdomain $\{u_1^0, u_2^1, u_1^2, u_2^3, u_1^4, \dots\}$, so not much is gained then with this parallelism, and the convergence behavior is the same. We thus only study the convergence of the alternating Schwarz method (4.30) in what follows; for the well posedness of the Schwarz algorithm, see the Appendix, Section 5.1.

The convergence of classical alternating and parallel Schwarz methods can be proved by different techniques: an analysis based on variational arguments and orthogonal projection [87, 23], which provides convergence in the H^1 norm, an analysis based on maximum principle estimates, which allows one to obtain convergence in the maximum norm [88, 22, 24, 26, 25], and Fourier analysis [43, 21, 16]. The following theorem is based on Fourier analysis, and we suppose for the rest of this section that the assumptions in Theorem 43 (resp. Theorem 44) hold for the Schwarz algorithm to be well posed.

Theorem 36 (Convergence of the classical Schwarz method)**.** *Assume that the domain $\Omega = (0,1) \times (0,L)$ is decomposed as in Figure 4.12. For any sufficiently regular initialization u^0 such that $u^0 = g$ on $\partial\Omega$, the alternating Schwarz method* (4.30) *converges to the solution u of* (4.29)*, and satisfies the convergence estimate*

$$\|u - u_j^n\|_{L^2(\Omega_j)} \leq \rho_s^n \|u - u^0\|_{L^2(\Omega)}, \tag{4.32}$$

where $\|\cdot\|_{L^2}$ is the classical L^2-norm, and

$$\rho_s(\alpha, \beta, L) = \frac{\sinh(\frac{\pi}{L}\beta)}{\sinh(\frac{\pi}{L}\alpha)} \frac{\sinh(\frac{\pi}{L}(1-\alpha))}{\sinh(\frac{\pi}{L}(1-\beta))} < 1. \tag{4.33}$$

Proof. Define the errors $e_j^n := u - u_j^n$ for $j = 1, 2$. Since the Schwarz subproblems are linear, the errors e_j^n satisfy the so-called error equations

$$\begin{aligned} \Delta e_1^n &= 0 && \text{in } \Omega_1, & \qquad \Delta e_2^n &= 0 && \text{in } \Omega_2, \\ e_1^n &= e_2^{n-1} && \text{on } \Gamma_1, & \qquad e_2^n &= e_1^n && \text{on } \Gamma_2, \\ e_1^n &= 0 && \text{on } \partial\Omega \cap \partial\Omega_1, & \qquad e_2^n &= 0 && \text{on } \partial\Omega \cap \partial\Omega_2. \end{aligned} \tag{4.34}$$

Hence, by Weyl's theorem [59] they are harmonic functions. It is well known that a separation of variables ansatz permits to show that the solutions e_j^n to (4.34) have the form of the Fourier sine series [94, Section 4.3]

$$e_j^n(x, y) = \sum_{k \in K} \widehat{e}_j^n(x, k) \sin(ky), \tag{4.35}$$

with $K := \{\frac{\pi}{L}, \frac{2\pi}{L}, \frac{3\pi}{L}, \ldots\}$, and where $\widehat{e}_j^n(x,k)$ are the Fourier coefficients that solve the second-order ordinary differential equations

$$\begin{aligned} \partial_{xx}\widehat{e}_1^n(x,k) - k^2\widehat{e}_1^n(x,k) &= 0 \text{ for } x \in (0,\alpha), \\ \widehat{e}_1^n(0,k) &= 0, \\ \widehat{e}_1^n(\alpha,k) &= \widehat{e}_2^{n-1}(\alpha,k), \end{aligned}$$

and

$$\begin{aligned} \partial_{xx}\widehat{e}_2^n(x,k) - k^2\widehat{e}_2^n(x,k) &= 0 \text{ for } x \in (\beta,1), \\ \widehat{e}_2^n(\beta,k) &= \widehat{e}_1^n(\beta,k), \\ \widehat{e}_2^n(1,k) &= 0. \end{aligned}$$

The solutions to these two ordinary differential equations have the form

$$\widehat{e}_j^n(x,k) = A_j^n(k)e^{kx} + B_j^n(k)e^{-kx},$$

for $j = 1, 2$, where $A_j^n(k)$ and $B_j^n(k)$ are determined by imposing the boundary conditions. Using first the outer boundary conditions posed in the original problem we obtain

$$\begin{aligned} \widehat{e}_1^n(0,k) = 0 \quad &\Longrightarrow \quad B_1^n(k) = -A_1^n(k), \\ \widehat{e}_2^n(1,k) = 0 \quad &\Longrightarrow \quad B_2^n(k) = -A_2^n(k)e^{2k}, \end{aligned} \tag{4.36}$$

which implies that

$$\begin{aligned} \widehat{e}_1^n(x,k) &= A_1^n(k)(e^{kx} - e^{-kx}) = 2A_1^n(k)\sinh(kx), \\ \widehat{e}_2^n(x,k) &= A_2^n(k)(e^{kx} - e^{2k}e^{-kx}) = -2A_2^n(k)e^k\sinh(k(1-x)). \end{aligned} \tag{4.37}$$

Now, from the transmission condition in (4.34) we have that $\widehat{e}_1^n(\alpha,k) = \widehat{e}_2^{n-1}(\alpha,k)$, which implies

$$A_1^n(k)2\sinh(k\alpha) = A_2^{n-1}(k)\big(-2e^k\sinh(k(1-\alpha))\big), \tag{4.38}$$

and $\widehat{e}_2^n(\beta,k) = \widehat{e}_1^n(\beta,k)$ gives

$$A_1^n(k)2\sinh(k\beta) = A_2^n(k)\big(-2e^k\sinh(k(1-\beta))\big). \tag{4.39}$$

By combining (4.38) and (4.39), we obtain that

$$A_2^n(k) = \frac{\sinh(k\beta)}{\sinh(k(1-\beta))}\frac{\sinh(k(1-\alpha))}{\sinh(k\alpha)}A_2^{n-1}(k).$$

The convergence factor of this iteration is thus

$$\widehat{\rho}_s(\alpha,\beta,k) := \frac{\sinh(k\beta)}{\sinh(k(1-\beta))}\frac{\sinh(k(1-\alpha))}{\sinh(k\alpha)}, \tag{4.40}$$

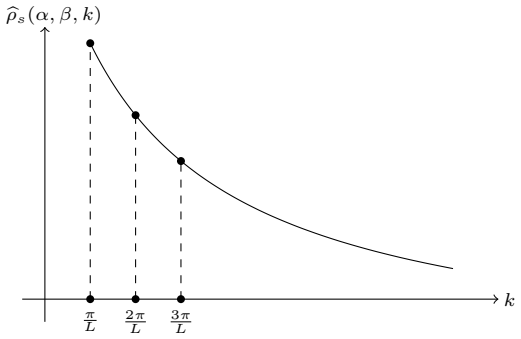


Figure 4.13: Qualitative behavior of the convergence factor $\widehat{\rho}_s(\alpha,\beta,k)$ as a function of k. Notice that we are interested in the values $k = \frac{\pi}{L}, \frac{2\pi}{L}, \frac{3\pi}{L}, \ldots$

and repeating a similar argument also for A_1^n, we obtain by induction

$$A_2^n(k) = \big(\widehat{\rho}_s(\alpha,\beta,k)\big)^n A_2^0(k) \quad \text{and} \quad A_1^n(k) = \big(\widehat{\rho}_s(\alpha,\beta,k)\big)^n A_1^0(k). \tag{4.41}$$

The qualitative behavior of the convergence factor $\widehat{\rho}_s(\alpha,\beta,k)$ as a function of k is shown in Figure 4.13. A direct calculation shows that the maximum of $\widehat{\rho}_s(\alpha,\beta,k)$ is attained for the lowest frequency $k = \frac{\pi}{L}$, $\max_{k\in K} \widehat{\rho}_s(\alpha,\beta,k) = \widehat{\rho}_s(\alpha,\beta,\frac{\pi}{L})$. Equation (4.41) implies that

$$\begin{aligned} e_1^n(x,y) &= \sum_{k\in K} \widehat{e}_1^n(x,k)\sin(ky) \\ &= \sum_{k\in K} A_1^n(k) 2\sinh(kx)\sin(ky) \\ &= \sum_{k\in K} \big(\widehat{\rho}_s(\alpha,\beta,k)\big)^n A_1^0(k) 2\sinh(kx)\sin(ky). \end{aligned}$$

We can now use the Parseval-Plancherel formula [28, Theorems 4.9-1 and 4.9-2] to estimate

$$\begin{aligned} \|e_1^n(x,\cdot)\|^2_{L^2(0,1)} &= \sum_{k\in K} \Big[\big(\widehat{\rho}_s(\alpha,\beta,k)\big)^n A_1^0(k) 2\sinh(kx)\Big]^2 \\ &\le \big(\widehat{\rho}_s(\alpha,\beta,\tfrac{\pi}{L})\big)^{2n} \sum_{k\in K} \Big[A_1^0(k) 2\sinh(kx)\Big]^2 \\ &= \big(\widehat{\rho}_s(\alpha,\beta,\tfrac{\pi}{L})\big)^{2n} \|e_1^0(x,\cdot)\|^2_{L^2(0,1)}, \end{aligned}$$

that is

$$\|e_1^n(x,\cdot)\|_{L^2(0,1)} \le \big(\widehat{\rho}_s(\alpha,\beta,\tfrac{\pi}{L})\big)^n \|e_1^0(x,\cdot)\|_{L^2(0,1)},$$

and hence

$$\|e_1^n\|_{L^2(\Omega_1)} \le \big(\widehat{\rho}_s(\alpha,\beta,\tfrac{\pi}{L})\big)^n \|u - u^0\|_{L^2(\Omega)}.$$

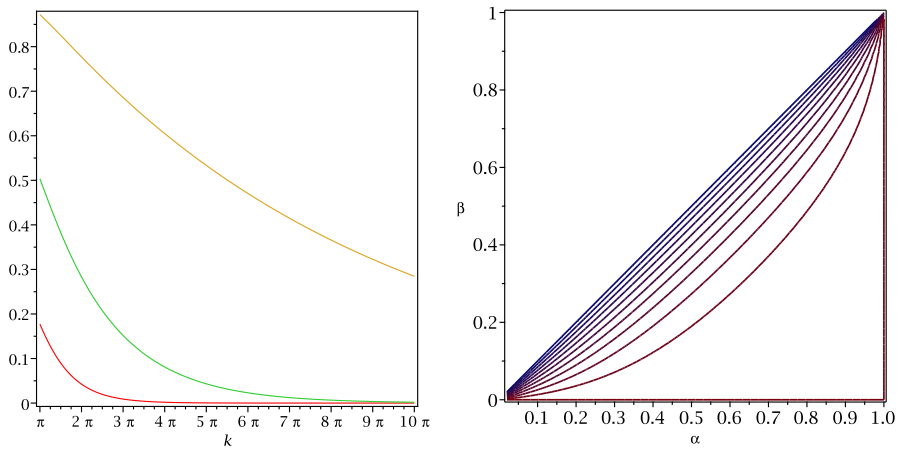


Figure 4.14: Left: dependence of the convergence factor $\widehat{\rho}_s(\alpha, \beta, k)$ of the alternating Schwarz method as a function of the frequency parameter k, from top to bottom for $\alpha \in \{0.51, 0.55, 0.625\}$ and $\beta \in \{0.49, 0.45, 0.375\}$. Right: level sets corresponding to $\{0, 0.1, \ldots, 1\}$ of the overall convergence factor $\rho_s(\alpha, \beta, L)$ from Theorem 36.

The result for e_2^n is obtained using the same arguments, and we obtain (4.33) with $\rho_s(\alpha, \beta, L) = \widehat{\rho}_s(\alpha, \beta, \frac{\pi}{L})$. This quantity is smaller than one, since the hyperbolic sine is a growing function, and the subdomains overlap, $\beta < \alpha$, and thus also $1 - \alpha < 1 - \beta$, which makes the two rearranged fractions in $\rho_s(\alpha, \beta, L)$ both smaller than one. □

The analysis above is typical for the analysis of iterative methods based on the underlying physics of the problem: it is performed whenever possible at the continuous level, without discretization. The convergence result we obtained in Theorem 36 shows that the convergence speed of the method depends on the overlap parameters α and β. In Figure 4.14, we show the convergence factor $\widehat{\rho}_s(\alpha, \beta, k)$ from (4.40) as a function of each Fourier frequency k, and also $\rho_s(\alpha, \beta, L)$ from (4.33) as a function of α and β for $L = 1$. The plots were obtained with the MAPLE commands

```
rho:=sinh(k*beta)/sinh(k*(1-beta))*sinh(k*(1-alpha))/sinh(k*alpha);
alpha:=0.625; beta:=0.375; rho1:=rho;
alpha:=0.55; beta:=0.45; rho2:=rho;
alpha:=0.51; beta:=0.49; rho3:=rho;
plot([rho1,rho2,rho3],k=Pi..10*Pi,axes=boxed);
alpha:='alpha';beta:='beta';k:=Pi;
plots[contourplot](rho,alpha=0..1,beta=0..alpha,contours=
  [seq(i/10,i=0..10)],axes=boxed);
```

We see on the left in Figure 4.14 that the convergence of the alternating Schwarz method is greatly influenced by the overlap size $\alpha - \beta$: if the overlap is large, convergence is fast, and if the overlap is small, convergence is slower. We also see that high frequencies, k large, converge much faster than low frequencies,

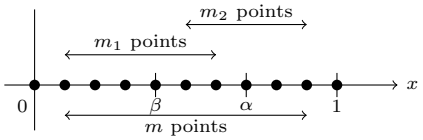


Figure 4.15: Discretization grid and discrete domain decomposition.

k small, so the classical Schwarz method is a smoother. On the right in Figure 4.14 we also see that the position of the interfaces defined by α and β, i.e. the subdomain geometry, has an influence on the convergence of the alternating Schwarz method: if one subdomain is small and the other one is big, the method also converges faster for the same overlap size than when both subdomains are of the same size. From the formula for $\rho_s(\alpha, \beta, L)$ from (4.33), we also see that the domain height or interface length L influences the convergence: for L large, the convergence is slow, and for L small, the convergence is fast. This influence of the geometry of the decomposition on the performance of the Schwarz method has only been investigated very recently, see for example [45, 3, 27].

Classical Schwarz preconditioners

To use the alternating Schwarz method as a computational tool and then as a preconditioner, we have to discretize the algorithm (4.30). We have seen in Chapter 1 that Laplace's equation (4.29) at the discrete level gives the linear system

$$A\boldsymbol{u} = \boldsymbol{f},$$

where $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^{m^2}$, $A \in \mathbb{R}^{m^2 \times m^2}$ is the discrete Laplacian matrix, and the domain $\overline{\Omega} = [0,1] \times [0,1]$[7] is discretized with a uniform grid of $(m+2) \times (m+2)$ points, including the boundary points.

The matrix A can be decomposed according to the two subdomain decomposition of the alternating Schwarz method in two different ways,

$$A = \begin{bmatrix} A_1 & B_{12} \\ \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times \\ B_{21} & A_2 \end{bmatrix}, \tag{4.42}$$

where $A_j = R_j A R_j^\top$, and R_1 and R_2 are rectangular restriction matrices given by

$$R_1 = \begin{bmatrix} I_{m_1 m} & 0 \end{bmatrix} \in \mathbb{R}^{m_1 m \times m^2} \quad \text{and} \quad R_2 = \begin{bmatrix} 0 & I_{m_2 m} \end{bmatrix} \in \mathbb{R}^{m_2 m \times m^2},$$

with $I_{m_1 m}$ and $I_{m_2 m}$ the identities of size $m_1 m$ and $m_2 m$, and the discretization points corresponding to m_1 and m_2 are shown in Figure 4.15. To write the alternating Schwarz method at the discrete level, we need to introduce in addition the matrices

$$\widetilde{B}_{12} = \begin{bmatrix} 0 & B_{12} \end{bmatrix} \in \mathbb{R}^{m_1 m \times m_2 m} \quad \text{and} \quad \widetilde{B}_{21} = \begin{bmatrix} B_{21} & 0 \end{bmatrix} \in \mathbb{R}^{m_2 m \times m_1 m},$$

[7] We chose for simplicity a square domain here, $L = 1$.

and the vectors $\boldsymbol{f}_j = R_j \boldsymbol{f}$ for $j = 1, 2$. The discrete version of the alternating Schwarz method (4.30) can then be expressed using these matrices by

$$\begin{aligned} A_1 \boldsymbol{u}_1^n &= \boldsymbol{f}_1 - \widetilde{B}_{12} \boldsymbol{u}_2^{n-1}, \\ A_2 \boldsymbol{u}_2^n &= \boldsymbol{f}_2 - \widetilde{B}_{21} \boldsymbol{u}_1^n, \end{aligned} \tag{4.43}$$

where $\boldsymbol{u}_1 \in \mathbb{R}^{m_1 m}$ and $\boldsymbol{u}_2 \in \mathbb{R}^{m_2 m}$. We see that in the first line of (4.43), there is a subdomain solve using the matrix A_1 corresponding to the discretized Laplacian on subdomain Ω_1, and at the interface, this subdomain solve uses data from the previous iteration on subdomain Ω_2 represented by the term $\widetilde{B}_{12} \boldsymbol{u}_2^{n-1}$. Similarly, in the second line of (4.43), there is a subdomain solve using the matrix A_2 corresponding to the discretized Laplacian on subdomain Ω_2, and at the interface, this subdomain solve uses data from the just completed solve on subdomain Ω_1 represented by the term $\widetilde{B}_{21} \boldsymbol{u}_1^n$. An example of matrices A_1, A_2, $\widetilde{B}_{12}$, and $\widetilde{B}_{21}$ for a one-dimensional Laplace problem is the following (see also Problem 39).

Example 12. *Consider the one-dimensional Laplace problem*

$$u_{xx} = f \text{ in } (0,1) \text{ with } u(0) = u(1) = 0,$$

and the alternating Schwarz method

$$\begin{aligned} &(u_1^n)_{xx} = f \text{ in } (0,\alpha) \text{ with } u_1^n(0) = 0,\ u_1^n(\alpha) = u_2^{n-1}(\alpha), \\ &(u_2^n)_{xx} = f \text{ in } (\beta,1) \text{ with } u_2^n(\beta) = u_1^n(\beta),\ u_2^n(1) = 0. \end{aligned} \tag{4.44}$$

We use a discretization mesh of $m = 9$ interior points with a mesh size $h = \frac{1}{m+1}$. The two subdomains are $\Omega_1 = (0, \alpha)$ and $\Omega_2 = (\beta, 1)$ with $\alpha = 7h$, $\beta = 4h$, and an overlap of $m_{\text{over}} h$ with $m_{\text{over}} = 3$; see Figure 4.15. In Ω_1 and Ω_2 we consider $m_1 = 6$ and $m_2 = 5$ interior mesh points. The discretization of (4.44) is then given by

$$A_1 \boldsymbol{u}_1^n = f_1 + \widetilde{B}_{12} \boldsymbol{u}_2^{n-1}, \quad A_2 \boldsymbol{u}_2^n = f_2 + \widetilde{B}_{21} \boldsymbol{u}_1^n,$$

where $\boldsymbol{u}_1 \in \mathbb{R}^{m_1}$, $\boldsymbol{u}_2 \in \mathbb{R}^{m_2}$,

$$A_1 = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{6\times 6}, \quad B_{12} = \frac{1}{h^2} \begin{bmatrix} & & \\ & & \\ & & \\ & & \\ & & \\ 1 & & \end{bmatrix} \in \mathbb{R}^{6\times 3},$$

$$A_2 = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{5\times 5}, \quad B_{21} = \frac{1}{h^2} \begin{bmatrix} & & & & 1 \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \in \mathbb{R}^{5\times 4}.$$

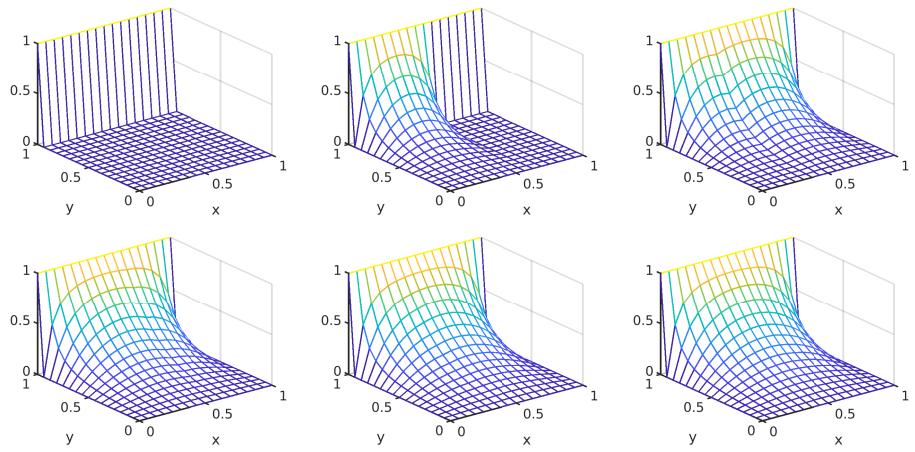


Figure 4.16: Initial guess and first iterations of the alternating Schwarz method applied to our Laplace model problem.

and the matrices $\widetilde{B}_{12}$ *and* $\widetilde{B}_{21}$ *are*

$$\widetilde{B}_{12} = \frac{1}{h^2}\begin{bmatrix} 0_{m_1 \times m_{\text{over}}-1} & B_{12} \end{bmatrix} \in \mathbb{R}^{6\times 5}, \quad \widetilde{B}_{21} = \frac{1}{h^2}\begin{bmatrix} B_{21} & 0_{m_2 \times m_{\text{over}}-1} \end{bmatrix} \in \mathbb{R}^{5\times 6}.$$

Notice that

$$\widetilde{B}_{12}\boldsymbol{u}_2^{n-1} = \frac{1}{h^2}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ (\boldsymbol{u}_2^{n-1})_3 \end{bmatrix}, \quad \widetilde{B}_{21}\boldsymbol{u}_1^{n} = \frac{1}{h^2}\begin{bmatrix} (\boldsymbol{u}_1^{n})_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

which clearly shows how the Dirichlet transmission condition is transferred from one subdomain to the other in the discretized alternating Schwarz method.

Getting back to the two-dimensional case, we show in Figure 4.16 the first few iterations of the alternating Schwarz method applied to our Laplace model problem (see also Problem 42). We used $m = 15$ interior mesh points, and an overlap of $4h$, $h = \frac{1}{m+1}$, and subdomains of the same size, which means that $\alpha = \frac{1}{2} + 2h = 0.625$ and $\beta = \frac{1}{2} - 2h = 0.375$. One can clearly see how effective subdomain solves are to give rapidly a very good approximate solution: only after solving on the left and right twice, we arrive basically at the solution.

For fixed positions of the interfaces α and β in physical space, this discretized alternating Schwarz method will converge as predicted by the continuous analysis when the mesh is refined, and the contraction factor will thus not depend on the mesh parameter h when h goes to zero. This is a first iterative solver for the linear system corresponding to the discretized Laplacian whose convergence

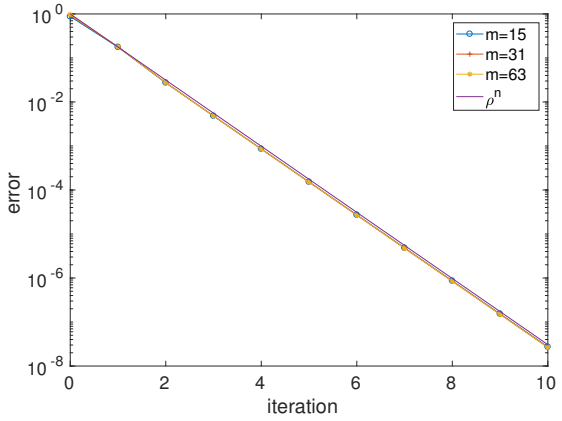


Figure 4.17: Convergence of the alternating Schwarz method for different mesh sizes $h = \frac{1}{m+1}$. For comparison also the theoretical convergence estimate based on the continuous analysis from Theorem 36 is shown.

does not depend on the mesh size! We illustrate this in Figure 4.17, where we show how the error decays in the discretized alternating Schwarz iteration for the mesh we used for Figure 4.16 with $m = 15$, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points, keeping the subdomains and hence the interfaces fixed at $\alpha = 0.625$ and $\beta = 0.375$. We clearly see that convergence is independent of the mesh parameter h, and very well predicted by the continuous analysis in Theorem 36.

Multiplicative Schwarz

There is an even more convenient way to implement the alternating Schwarz method than the discretized iteration (4.43), and which one can also more easily generalize to many subdomains. It is based on formulating the iteration for all the unknowns at once, and became known under the name *Multiplicative Schwarz Method*. For our two subdomain case, the method starts with an initial guess $\boldsymbol{u}^0$ for all the unknowns, and then computes for $n = 0, 1, \ldots$

$$\begin{aligned} \boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{u}^n + R_1^\top A_1^{-1} R_1 \big(\boldsymbol{f} - A\boldsymbol{u}^n\big), \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top A_2^{-1} R_2 \big(\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{1}{2}}\big). \end{aligned} \tag{4.45}$$

It is not immediately clear how the multiplicative Schwarz method (4.45) is related to the discretization of the alternating Schwarz method (4.43), since the subdomain solutions $\boldsymbol{u}_j^n$ do not even appear in the multiplicative Schwarz method (4.45). We show in the following theorem that (4.43) and (4.45) are in fact equivalent (for a proof in the general case of many subdomains, see [44]).

Theorem 37 (Alternating Schwarz = Multiplicative Schwarz)**.** *Assume that the initialization $\boldsymbol{u}^0$ of the multiplicative Schwarz method* (4.45)*, and the initializations $\boldsymbol{u}_1^0$ and $\boldsymbol{u}_2^0$ of the discretized alternating Schwarz method* (4.43) *satisfy the*

relation

$$\boldsymbol{u}^0 := (I - R_2^\top R_2) R_1^\top \boldsymbol{u}_1^0 + R_2^\top \boldsymbol{u}_2^0. \tag{4.46}$$

Then for $n \geq 1$ the iterates $\boldsymbol{u}^n$ of the multiplicative Schwarz method (4.45) *and the iterates $\boldsymbol{u}_1^0$ and $\boldsymbol{u}_2^0$ of the discretized alternating Schwarz method* (4.43) *remain related by the same relation,*

$$\boldsymbol{u}^n := (I - R_2^\top R_2) R_1^\top \boldsymbol{u}_1^n + R_2^\top \boldsymbol{u}_2^n. \tag{4.47}$$

Proof. We proceed by induction. Notice that for $n = 0$ (4.47) is equal to (4.46). Now, we assume that (4.47) holds for n and we show that it holds for $n+1$ as well. The first equation in (4.45) allows us to compute

$$\begin{aligned}
\boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{u}^n + R_1^\top A_1^{-1} R_1 \big(\boldsymbol{f} - A\boldsymbol{u}^n\big) \\
&= \boldsymbol{u}^n + R_1^\top A_1^{-1} \big(R_1 \boldsymbol{f} - R_1 A \boldsymbol{u}^n\big) \\
&= \boldsymbol{u}^n + R_1^\top A_1^{-1} \big(\boldsymbol{f}_1 - \begin{bmatrix} A_1 & B_{12} \end{bmatrix} \boldsymbol{u}^n\big) \\
&= \boldsymbol{u}^n + R_1^\top A_1^{-1} (\boldsymbol{f}_1 - A_1 R_1 \boldsymbol{u}^n - \widetilde{B}_{12} \boldsymbol{u}_2^n) \\
&= (I - R_1^\top A_1^{-1} A_1 R_1) \boldsymbol{u}^n + R_1^\top A_1^{-1} (\boldsymbol{f}_1 - \widetilde{B}_{12} \boldsymbol{u}_2^n) \\
&= (I - R_1^\top R_1) \boldsymbol{u}^n + R_1^\top \boldsymbol{u}_1^{n+1},
\end{aligned} \tag{4.48}$$

where the last equality follows from the first equation in (4.43). Similarly, using the second equation in (4.45), we get

$$\begin{aligned}
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top A_2^{-1} R_2 \big(\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{1}{2}}\big) \\
&= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top A_2^{-1} \big(R_2 \boldsymbol{f} - R_2 A \boldsymbol{u}^{n+\frac{1}{2}}\big) \\
&= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top A_2^{-1} \big(\boldsymbol{f}_2 - \begin{bmatrix} B_{21} & A_2 \end{bmatrix} \boldsymbol{u}^{n+\frac{1}{2}}\big) \\
&= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top A_2^{-1} (\boldsymbol{f}_2 - \widetilde{B}_{21} \boldsymbol{u}_2^{n+\frac{1}{2}} - A_2 R_2 \boldsymbol{u}^{n+\frac{1}{2}}) \\
&= (I - R_2^\top R_2) \boldsymbol{u}^{n+\frac{1}{2}} + R_2^\top \boldsymbol{u}_2^{n+1},
\end{aligned}$$

and using (4.48), we obtain

$$\begin{aligned}
\boldsymbol{u}^{n+1} &= (I - R_2^\top R_2)(I - R_1^\top R_1) \boldsymbol{u}^n + (I - R_2^\top R_2) R_1^\top \boldsymbol{u}_1^{n+1} + R_2^\top \boldsymbol{u}_2^{n+1} \\
&= (I - R_2^\top R_2) R_1^\top \boldsymbol{u}_1^{n+1} + R_2^\top \boldsymbol{u}_2^{n+1},
\end{aligned} \tag{4.49}$$

where we used that $(I - R_2^\top R_2)(I - R_1^\top R_1) = 0$. We have then obtained that (4.47) holds for $n+1$ and our proof is complete. □

We are now interested in finding the preconditioner M_{MS} that corresponds to the multiplicative Schwarz method. To do so, we replace the first equation

of (4.45) into the second one to get[8]

$$\begin{aligned}\boldsymbol{u}^{n+1} &= \boldsymbol{u}^n + R_1^\top A_1^{-1} R_1\big(\boldsymbol{f} - A\boldsymbol{u}^n\big) + R_2^\top A_2^{-1} R_2\big[\boldsymbol{f} - A\big(\boldsymbol{u}^n + R_1^\top A_1^{-1} R_1\big(\boldsymbol{f} - A\boldsymbol{u}^n\big)\big)\big] \\ &= \big(I - R_1^\top A_1^{-1} R_1 A - R_2^\top A_2^{-1} R_2 A + R_2^\top A_2^{-1} R_2 A R_1^\top A_1^{-1} R_1 A\big)\boldsymbol{u}^n \\ &\quad + \underbrace{\big(R_1^\top A_1^{-1} R_1 + R_2^\top A_2^{-1} R_2 - R_2^\top A_2^{-1} R_2 A R_1^\top A_1^{-1} R_1\big)}_{M_{\mathrm{MS}}^{-1}} \boldsymbol{f},\end{aligned}$$

where we identified the inverse of the *multiplicative Schwarz preconditioner* M_{MS}. The previous formula can be rewritten as

$$\boldsymbol{u}^{n+1} = \big(I - R_1^\top A_1^{-1} R_1 A\big)\big(I - R_2^\top A_2^{-1} R_2 A\big)\boldsymbol{u}^n + M_{\mathrm{MS}}^{-1}\boldsymbol{f}, \tag{4.50}$$

which is the equivalent preconditioned form of the multiplicative Schwarz method (4.45).

We can now give the program we used to produce the numerical experiments in Figure 4.16:

```
m=15;                                  % number of gridpoints
A=Laplacian(m,2);                      % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;         % put bc into the rhs
u=A\f;                                 % solve by sparse Gaussian elimination
ai=9; bi=7;                            % alpha=(ai+1)*h, beta=(bi-1)*h
R1=[speye(m*ai) sparse(m*ai,m*(m-ai))]; % construct restriction matrices
R2=[sparse(m*(m-bi+1),m*(bi-1)) speye(m*(m-bi+1))];
A1=R1*A*R1';                           % construct subdomain matrices
A2=R2*A*R2';
h=1/(m+1); x=0:h:1; y=x;               % mesh point vectors
udd=zeros(size(f));                    % initial guess
U=zeros(m+2); U(end,1:m+2)=1;          % for plotting
for n=0:10
  err(n+1)=max(max(abs(u-udd)));       % compute error
  U(2:m+1,2:m+1)=reshape(udd,m,m);
  mesh(x,y,U)
  xlabel('x');ylabel('y');
  pause
  udd=udd+R1'*(A1\(R1*(f-A*udd)));     % first subdomain solve
  U(2:m+1,2:m+1)=reshape(udd,m,m);
  mesh(x,y,U)
  xlabel('x');ylabel('y');
  pause
  udd=udd+R2'*(A2\(R2*(f-A*udd)));     % second subdomain solve
end
```

The multiplication of the two terms stemming from the subdomain solve in the multiplicative Schwarz preconditioner (4.50) is not convenient for parallel computing, since the operations need to be performed sequentially.

[8]This type of discrete analysis of Schwarz methods involves more and more complicated sequences of the R_j matrices, especially in the many subdomain case [44], and led Stefan Güttel during his postdoc in Geneva to show us the German tongue twister on 'Rhabarberbarbera', just search on the internet.

Additive Schwarz and Restricted Additive Schwarz

Multiplying the two terms in (4.50) out, we obtain

$$\begin{aligned}\big(I - R_1^\top A_1^{-1} R_1 A\big)\big(I - R_2^\top A_2^{-1} R_2 A\big) = \\ I - R_1^\top A_1^{-1} R_1 A - R_2^\top A_2^{-1} R_2 A + R_1^\top A_1^{-1} R_1 A R_2^\top A_2^{-1} R_2 A,\end{aligned}$$

which led Dryja and Widlund to just drop the last term in the preconditioner, which is the only sequential one [126], and led to the well known so-called *additive Schwarz method*,

$$\boldsymbol{u}^{n+1} = \big(I - R_1^\top A_1^{-1} R_1 A - R_2^\top A_2^{-1} R_2 A\big)\boldsymbol{u}^n + M_{\text{AS}}^{-1}\boldsymbol{f}, \tag{4.51}$$

where the additive Schwarz preconditioner M_{AS} is

$$M_{\text{AS}}^{-1} = R_1^\top A_1^{-1} R_1 + R_2^\top A_2^{-1} R_2.$$

It is very tempting to think that this modification is equivalent to the modification proposed by Lions in (4.31) to make the alternating Schwarz method parallel, but this is not so! The additive Schwarz method (4.51) only converges in the case of minimal overlap, that is $\alpha - \beta = h$, where h is the mesh size, which means the subdomains at the algebraic level have no interior unknowns in common, see [44]. However, it is a good preconditioner for Krylov methods even in the case of larger overlap, and it is symmetric if the underlying problem matrix A is symmetric, which then permits the use of preconditioned conjugate gradient method. The *Restricted Additive Schwarz method (RAS)*, which was discovered due to a programming error in [15], is the correct algebraic formulation of the Lions parallel Schwarz method , for a proof, see [44]. It is obtained by modifying the extension matrices R_j^T in M_{AS}^{-1} to new extension matrices $\widetilde{R}_j^T$,

$$M_{\text{RAS}}^{-1} = \widetilde{R}_1^\top A_1^{-1} R_1 + \widetilde{R}_2^\top A_2^{-1} R_2,$$

where the structure of the $\widetilde{R}_j$ is like the structure of the R_j, but the values in $\widetilde{R}_j$ constitute a partition of unity, so that they sum to the identity,

$$\sum_j \widetilde{R}_j^T R_j = I.$$

Unfortunately RAS is a non-symmetric preconditioner, even for symmetric matrices A, except in the case of minimal overlap, where RAS and Additive Schwarz become the same method; see Problem 41.

We show in Figure 4.18 the initial guess and the first few iterates when using restricted additive Schwarz for solving our Laplace model problem. We clearly see that this corresponds to a faithful implementation of the parallel Schwarz method: compared with the alternating Schwarz method in Figure 4.16, now both subdomains solve simultaneously and produce a new approximation, which is combined in the overlap using the partition of unity function in the $\widetilde{R}_j$. These results were obtained with the short MATLAB code

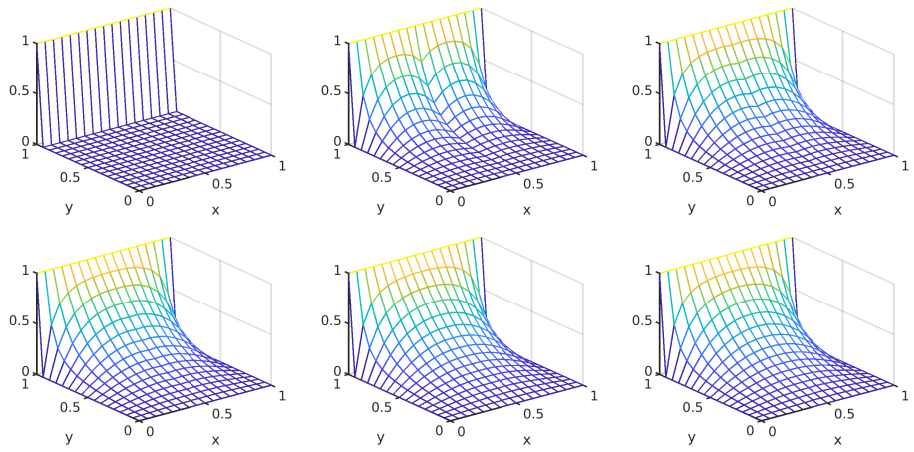


Figure 4.18: Initial guess and first iterations of the restricted additive Schwarz method, which is equivalent to the discretization of the parallel Schwarz method of Lions, applied to our Laplace model problem.

```
m=15;                                  % number of gridpoints
A=Laplacian(m,2);                      % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;         % put bc into the rhs
u=A\f;                                 % solve by sparse Gaussian elimination
ai=9; bi=7;                            % alpha=(ai+1)*h, beta=(bi-1)*h
R1=[speye(m*ai) sparse(m*ai,m*(m-ai))]; % construct restriction matrices
R2=[sparse(m*(m-bi+1),m*(bi-1)) speye(m*(m-bi+1))];
mi=floor((ai+bi)/2);
R1t=R1;R1t(m*mi+1:end,:)=0;            % up to the middle included
R2t=R2;R2t(1:(mi-bi+1)*m,:)=0;
A1=R1*A*R1';                           % construct subdomain matrices
A2=R2*A*R2';
h=1/(m+1); x=0:h:1; y=x;               % mesh point vectors
udd=zeros(size(f));                    % initial guess
U=zeros(m+2); U(end,1:m+2)=1;          % for plotting
for n=0:10
  err(n+1)=max(max(abs(u-udd)));       % compute error
  U(2:m+1,2:m+1)=reshape(udd,m,m);
  mesh(x,y,U)
  xlabel('x');ylabel('y');
  pause
  udd=udd+R1t'*(A1\(R1*(f-A*udd)))+R2t'*(A2\(R2*(f-A*udd)));
end
```

If we replace the $\widetilde{R}_j$ by the R_j in the sum in the last line of the loop to obtain the additive Schwarz method, we get the results shown in Figure 4.19. We clearly see that the method does not converge in the overlap, and can thus not be used as a stationary iterative solver, except when a suitable damping parameter is

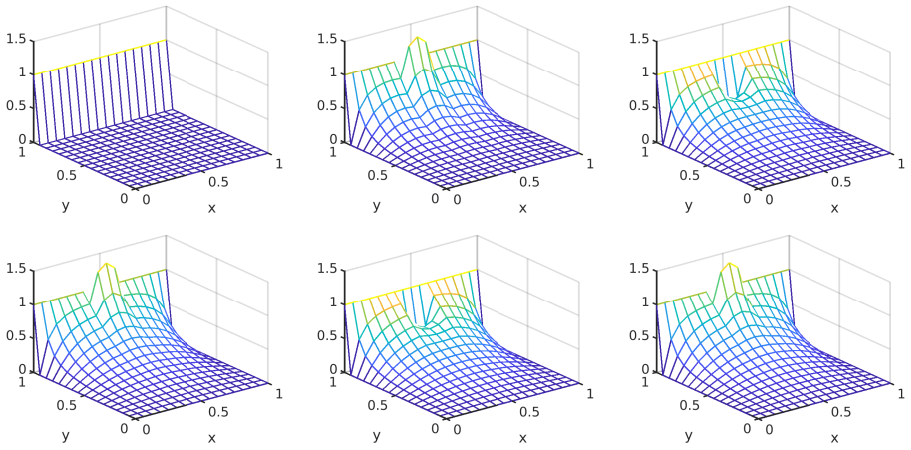


Figure 4.19: Initial guess and first iterations of the additive Schwarz method applied as stationary iteration to our Laplace model problem.

employed. When additive Schwarz is used as a preconditioner for a Krylov method however, the Krylov method takes care of the convergence problem. This corresponds in the classical abstract Schwarz theory to the number of colors appearing, which describes how many subdomains contain the same physical point, see for example [119].

Optimized Schwarz Methods

The classical alternating and parallel Schwarz methods discussed so far have two major drawbacks. On the one hand, the necessary overlap increases the computational effort needed for the solution of each subproblem, because it makes the subproblems bigger. On the other hand, the convergence of these methods deteriorates rapidly when the size of the overlap becomes small, and in the extreme case of non-overlapping subdomains, the convergence to the correct limit is lost: the method simply stagnates with the initial guess at the interface if there is no overlap.

For these reasons, Lions introduced in [89] a new domain decomposition method, by replacing the Dirichlet transmission conditions on the interfaces by Robin transmission conditions, namely

$$\begin{aligned}
\Delta u_1^n &= f && \text{in } \Omega_1, \\
\partial_{n_1} u_1^n + p_1 u_1^n &= \partial_{n_1} u_2^{n-1} + p_1 u_2^{n-1} && \text{on } \Gamma_1, \\
u_1^n &= g && \text{on } \partial\Omega \cap \partial\Omega_1, \\
\Delta u_2^n &= f && \text{in } \Omega_2, \\
\partial_{n_2} u_2^n + p_2 u_2^n &= \partial_{n_2} u_1^n + p_2 u_1^n && \text{on } \Gamma_2, \\
u_2^n &= g && \text{on } \partial\Omega \cap \partial\Omega_2,
\end{aligned} \tag{4.52}$$

where n_j are the outward normal vectors on the interfaces Γ_j, and the p_j are parameters or even operators that can be used to tune the convergence speed of the method. Lions designed this method as a variant of the Schwarz method without overlap, $\Omega_1 \cap \Omega_2 = \emptyset$ and $\Gamma_1 = \Gamma_2$, see [89], but it can naturally also be used with overlap, as suggested by Nataf and Rogier [91], and has a convergence rate for overlapping decompositions that is in general much faster than the convergence rate of the classical Schwarz methods, especially if the parameters p_j are well chosen, see, e.g., [43, 16]. The underlying principle of optimized Schwarz methods has led to a wealth of research and new algorithms, for a historical perspective, see [44], and for an overview of new wave propagation solvers, see [56].

To illustrate how much faster optimized Schwarz methods converge, we consider again the Poisson model problem on the rectangle decomposed into two rectangular subdomains, as shown in Figure 4.12. Then the normal derivatives in the optimized Schwarz method (4.52) become $\partial_{n_1} = \partial_x$ and $\partial_{n_2} = -\partial_x$, and like for the classical Schwarz method (4.30), we study its converge using Fourier series. The errors $e_j^n := u - u_j^n$ expanded in a Fourier series are of the same form (4.35) as for the classical Schwarz method,

$$e_j^n(x,y) = \sum_{k\in K} \widehat{e}_j^n(x,k)\sin(ky), \tag{4.53}$$

with $K := \{\frac{\pi}{L}, \frac{2\pi}{L}, \frac{3\pi}{L}, \ldots\}$ as before, and also the subdomain solutions for the Fourier coefficients are of the same form,

$$\begin{aligned} \widehat{e}_1^n(x,k) &= 2A_1^n(k)\sinh(kx), \\ \widehat{e}_2^n(x,k) &= -2A_2^n(k)e^k\sinh(k(1-x)). \end{aligned} \tag{4.54}$$

Using the *transmission conditions* in (4.52) for the errors, we get for the Fourier coefficients of the error using that $\partial_{n_1} = \partial_x$ and $\partial_{n_2} = -\partial_x$ the relations

$$\begin{aligned} &(\partial_x + p_1)\widehat{e}_1^n(\alpha,k) = 2A_1^n(k\cosh(k\alpha) + p_1\sinh(k\alpha)) = \\ &(\partial_x + p_1)\widehat{e}_2^{n-1}(\alpha,k) = -2A_2^{n-1}e^k(-k\cosh(k(1-\alpha)) + p_1\sinh(k(1-\alpha))) \end{aligned} \tag{4.55}$$

and

$$\begin{aligned} &(-\partial_x + p_2)\widehat{e}_2^n(\beta,k) = -2A_2^n e^k(k\cosh(k(1-\beta)) + p_2\sinh(k(1-\beta))) = \\ &(-\partial_x + p_2)\widehat{e}_1^n(\beta,k) = 2A_1^n e^k(-k\cosh(k\beta) + p_2\sinh(k\beta))). \end{aligned} \tag{4.56}$$

Introducing the second relation at iteration index $n-1$ into the first one, we find

$$A_1^n = \widehat{\rho}_{OSM}(\alpha,\beta,k,p_1,p_2)A_1^{n-1}$$

with the convergence factor of the optimized Schwarz method given by

$$\begin{aligned} &\widehat{\rho}_{OSM}(\alpha,\beta,k,p_1,p_2) = \\ &\frac{-k\cosh(k(1-\alpha)) + p_1\sinh(k(1-\alpha))}{k\cosh(k\alpha) + p_1\sinh(k\alpha)} \frac{-k\cosh(k\beta) + p_2\sinh(k\beta)}{k\cosh(k(1-\beta)) + p_2\sinh(k(1-\beta))}, \end{aligned} \tag{4.57}$$

and the same relation also holds for A_2^n. We see from this convergence factor that if the parameters p_1 and p_2 are large, we recover in the limit the convergence factor $\widehat{\rho}_s(\alpha, \beta, k)$ of the classical Schwarz method in (4.40), which is natural since the Robin transmission conditions in this case tend to the Dirichlet transmission conditions of the classical Schwarz method. We see however also that if we choose

$$\begin{aligned} p_1 = p_1(k) &= \frac{k\cosh(k(1-\alpha))}{\sinh(k(1-\alpha))} = k\coth(k\alpha) \\ p_2 = p_2(k) &= \frac{k\cosh(k\beta)}{\sinh(k\beta))} = k\coth(k\beta), \end{aligned} \tag{4.58}$$

then the convergence factor vanishes identically, $\widehat{\rho}_{OSM}(\alpha, \beta, k) \equiv 0$, and we obtain a direct solver for the kth error component, with convergence achieved in two iterations![9] In order to obtain the best performance, we want to make the contraction factor as small as possible for all the Fourier frequencies we are solving for, i.e. we should solve the *min-max problem*

$$\min_{p_1,p_2\in\mathbb{R}} \max_{k_{\min}\le k\le k_{\max}} |\widehat{\rho}_{OSM}(\alpha, \beta, k, p_1, p_2)|. \tag{4.59}$$

Here $k_{\min} = \frac{\pi}{L}$ and $k_{\max}$ depends on how many Fourier modes we are computing with. If the optimized Schwarz method is discretized on a grid with m grid points in the y direction of the interfaces, the mesh size h of a regular grid would be $h = \frac{L}{m+1}$, and we would have $m = \frac{L}{h} - 1$ discrete Fourier modes, which leads to

$$k_{\max} = \frac{m\pi}{L} = (\frac{L}{h} - 1)\frac{\pi}{L} \sim \frac{\pi}{h} \tag{4.60}$$

when the mesh size h becomes small. The min-max problem (4.59) can be easily solved numerically, using the following Matlab statements:

```
k=(1:15)*pi;                          % frequencies to be treated
alpha=0.625; beta=0.375;              % overlap parameters
rhoOSM2=@(p) (-k.*cosh(k*(1-alpha))+p(1)*sinh(k*(1-alpha)))...
  ./(k.*cosh(k*alpha)+p(1)*sinh(k*alpha))...
  .*(-k.*cosh(k*beta)+p(2)*sinh(k*beta))...
  ./(k.*cosh(k*(1-beta))+p(2)*sinh(k*(1-beta)));
rhoOSM=@(p) rhoOSM2([p p]);           % when using only one parameter
R2=@(p) max(abs(rhoOSM2(p)));         % functions giving the maximum
R=@(p) max(abs(rhoOSM(p)));           % to use in the minimization
[p2opt,R2opt]=fminsearch(R2,[1 2]) % minimize with p1 and p2
[popt,Ropt]=fminsearch(R,1)           % minimize with p1=p2=p
rhoS=sinh(k*beta)./sinh(k*(1-beta)).*sinh(k*(1-alpha))...
  ./sinh(k*alpha);                    % classical Schwarz
plot(k,rhoS,'-o',k,rhoOSM(popt),'-+',k,rhoOSM2(p2opt),'-*');
xlabel('k');
legend('Classical Schwarz','Optimized Schwarz','Optimized Schwarz 2p');
```

[9] In the parallel Schwarz variant of Lions, we need two iterations, in the alternating one the second subdomain converges already in the first iteration, see Problem 44.

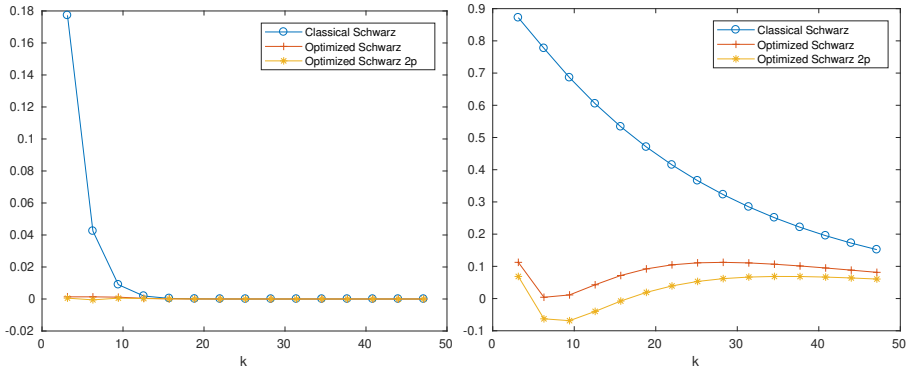


Figure 4.20: Convergence factors as function of the frequency parameter k, on the left for a large overlap, and on the right for a small overlap, both for classical and optimized Schwarz methods. Note the different scales in the figures.

Running the program for the given frequency range `k=(1:15)*pi`, MATLAB finds the best choice is $p_1^* = 3.8679$ and $p_2^* = 7.0775$, with a convergence factor bounded by $5.3847e-04$. If one uses only one parameter, MATLAB finds $p^* = 4.4760$, with a convergence factor bounded by 0.0014. Compared to the classical Schwarz method, which has already quite a good convergence factor of 0.1773, the optimized Schwarz method with two parameters is about 5 times faster, since $0.1773^5 = 1.7520e-04$. The convergence factors for the three methods is shown in Figure 4.20 on the left for each Fourier mode.

If we make the overlap much smaller, say $\alpha = 0.51$ and $\beta = 0.49$, we get $p_1^* = 4.3786$ and $p_2^* = 16.5309$, with a convergence factor bounded by 0.0686, and with one parameter $p^* = 7.2842$, with a convergence factor bounded by 0.1125. The classical Schwarz method has in with this overlap a convergence factor of 0.8719, which makes it about 20 times slower than the optimized Schwarz method with two parameters. So when the overlap becomes small, which is the case in practice where the overlap is only a few mesh sizes wide, the gap between the performance of the classical Schwarz method and the optimized Schwarz method widens.

To implement the optimized Schwarz method is not more difficult than to implement the classical Schwarz method, it suffices to make only a small change in the RAS implementation replacing the subdomain matrices to become matrices with Robin transmission conditions, see Problem 46 and also [35, 44]. This leads to the so-called *optimized restricted additive Schwarz method (ORAS)*,

```
m=15;                              % number of gridpoints
A=Laplacian(m,2);                  % five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1;     % put bc into the rhs
u=A\f;                             % solve by sparse Gaussian elimination
ai=10; bi=6;                       % alpha=ai*h, beta=bi*h, 2h less than RAS!
R1=[speye(m*ai) sparse(m*ai,m*(m-ai))];
R2=[sparse(m*(m-bi+1),m*(bi-1)) speye(m*(m-bi+1))];
mi=floor((ai+bi)/2);
```

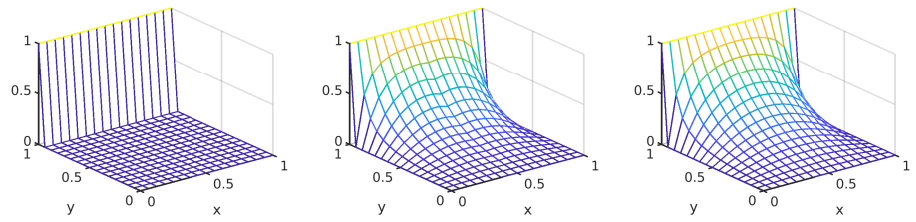


Figure 4.21: Initial guess and first two iterations of the optimized restricted additive Schwarz method, which is equivalent to the discretization of the optimized parallel Schwarz method, applied to our Laplace model problem.

```
R1t=R1;R1t(m*mi+1:end,:)=0;      % up to the middle included
R2t=R2;R2t(1:(mi-bi+1)*m,:)=0;
p1=4.4760;p2=p1;                 % put optimized values
p1=7.0775;p2=3.8679;             % two sided optimized values
A1=R1*A*R1';
A2=R2*A*R2';                     % change to Robin transmission conditions
h=1/(m+1);                       % mesh size scaling needed
A1(end-m+1:end,end-m+1:end)=0.5*A1(end-m+1:end,end-m+1:end)-p1*h*speye(m);
A2(1:m,1:m)=0.5*A2(1:m,1:m)-p2*h*speye(m);
x=0:h:1; y=x;                    % mesh point vectors
udd=zeros(size(f));              % initial guess
U=zeros(m+2); U(end,1:m+2)=1;    % for plotting
for n=0:10
  err(n+1)=max(max(abs(u-udd))); % compute error
  U(2:m+1,2:m+1)=reshape(udd,m,m);
  mesh(x,x,U)
  xlabel('x');ylabel('y');
  pause
  udd=udd+R1t'*(A1\(R1*(f-A*udd)))+R2t'*(A2\(R2*(f-A*udd)));
end
```

Running this code leads to the iterates shown in Figure 4.21. In contrast to the classical Schwarz case, we only show the first two iterates, since the following ones look identical, the method has visually converged already after two iterations! To see the important improvement in convergence more clearly, we show in Figure 4.22 the error reduction over the iterations for RAS and ORAS. We clearly see that ORAS converges much faster than RAS, but why is there a third convergence curve in Figure 4.22 labeled ORASa? This is because of an important difference when one uses the ORAS formulation to implement optimized Schwarz methods compared to RAS: as indicated in the program line defining the interfaces, these lie now exactly on the corresponding grid points, and not one outside, like in the Dirichlet case, since the Robin matrices also compute the solution along the interfaces. So to have the same *geometric overlap* in the ORAS implementation compared to RAS, one has to include one more grid line on each side. The additional convergence curve in Figure 4.22

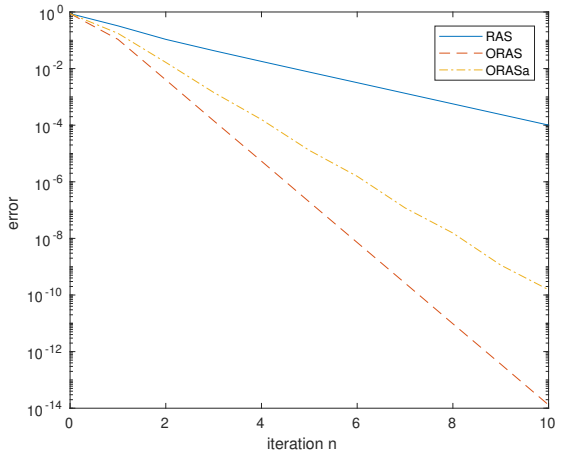


Figure 4.22: Error reduction as a function of the iteration index n comparing the classical parallel Schwarz method (RAS) and the optimized Schwarz method (ORAS) for our Laplace model problem, corresponding to the iterates shown in Figure 4.18 and 4.21 respectively.

corresponds to running the optimized Schwarz method with the same *algebraic overlap* as RAS, `ai=9` and `bi=7`, which means less geometric overlap for ORAS, namely $\alpha = 0.5625$, $\beta = 0.4375$, for which the optimized parameter becomes $p^* = 4.8185$. We see that convergence is still much faster than for classical RAS, so it is always worthwhile to use ORAS. An important consequence of this is that it does not make sense to run the ORAS code without algebraic overlap (just try it), i.e. `bi=ai+1`, since the geometric overlap would then be $-h$! But the ORAS implementation can also not be used to run a non-overlapping optimized Schwarz method, i.e. with `bi=ai+1`. A minimum overlap is needed since the subdomain solutions stored in a truncated way by the $\widetilde{R}_j$ operators in the global vector of unknowns must still retain enough information also to compute the normal derivative, see [115, Assumption 1] for the precise condition, and also Problem 45.

To illustrate how also the optimized Schwarz method converges independently of the mesh size when the overlap is held constant, independently of the mesh size, we show in Figure 4.23 how the error decays in the discretized optimized Schwarz iteration for the mesh we used in Figure 4.21 with $m = 15$ and $\alpha = 0.625$ and $\beta = 0.375$, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points, keeping the interfaces fixed at $\alpha = 0.625$ and $\beta = 0.375$. We see that convergence is independent of the mesh parameter h, and very well predicted by the continuous analysis for the optimized parameter p^*.[10]

To get a better theoretical understanding of the min-max problem that needs to be solved in optimized Schwarz methods to get this substantially faster con-

[10] Convergence flattens out at the roundoff error level for all mesh sizes.

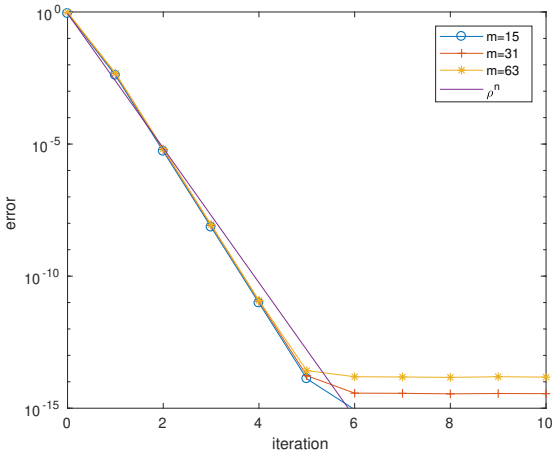


Figure 4.23: Convergence of the optimized Schwarz method for different mesh sizes $h = \frac{1}{m+1}$. For comparison also the theoretical convergence estimate (4.57) based on the continuous analysis is shown for the optimized $p^* = 4.4760$ solving the min-max problem (4.59).

vergence, it is convenient to introduce a simplification in our analysis, which is common in the analysis of optimized Schwarz methods, namely to make the subdomain sizes infinitely large on each side. This simplifies the formulas, but retains the mathematical essence of the problem for diffusive partial differential equations[11]. We thus consider now the two subdomains $\Omega_1 = (-\infty, \alpha) \times (0, 1)$ and $\Omega_2 = (\beta, \infty) \times (0, 1)$. Then the Fourier coefficients of the errors are not given by hyperbolic sines from (4.54) any more, but just exponential functions, namely

$$\begin{aligned} \widehat{e}_1^n(x,k) &= A_1^n(k)e^{kx}, \\ \widehat{e}_2^n(x,k) &= A_2^n(k)e^{-kx}, \end{aligned} \tag{4.61}$$

since they satisfy the error differential equation as well, and now as boundary conditions on the outer boundaries of the subdomains at $-\infty$ and ∞ that the errors must remain bounded there. Using the transmission conditions in (4.52) for these Fourier error coefficients, we get

$$\begin{aligned} (\partial_x + p_1)\widehat{e}_1^n(\alpha,k) &= A_1^n(k+p_1)e^{k\alpha} = \\ (\partial_x + p_1)\widehat{e}_2^{n-1}(\alpha,k) &= A_2^{n-1}(-k+p_1)e^{-k\alpha} \end{aligned} \tag{4.62}$$

and

$$\begin{aligned} (-\partial_x + p_2)\widehat{e}_2^n(\beta,k) &= A_2^n e^k(k+p_2)e^{k\beta} = \\ (-\partial_x + p_2)\widehat{e}_1^n(\beta,k) &= A_1^n e^k(-k+p_2)e^{-k\beta}. \end{aligned} \tag{4.63}$$

[11]This is not the case for wave propagation type problems, like the Helmholtz equation, where such a modification has an important influence on the min-max problem due to reflections from the boundaries.

Introducing again the second relation at iteration index $n-1$ into the first one, we find

$$A_1^n = \widehat{\rho}_{OSMs}(\alpha, \beta, k, p_1, p_2) A_1^{n-1},$$

with the simplified convergence factor of the optimized Schwarz method given by

$$\widehat{\rho}_{OSMs}(\alpha, \beta, k, p_1, p_2) = \frac{k - p_1}{k + p_1} \frac{k - p_2}{k + p_2} e^{-2k(\alpha-\beta)}, \tag{4.64}$$

and the same relation also holds for A_2^n. If we let p_1 and p_2 go to infinity, we also recover the simplified convergence factor for the classical Schwarz method, namely

$$\widehat{\rho}_s(\alpha, \beta, k) = e^{-2k(\alpha-\beta)}, \tag{4.65}$$

which now explicitly shows the typical exponential decay, depending on the size of the overlap $\delta := \alpha - \beta$, and the Fourier frequency k. We also see that the Robin transmission conditions add an extra factor in front of the classical Schwarz convergence factor, and this factor is smaller than one provided that $p_1, p_2 > 0$, so the optimized Schwarz method converges even without overlap.

We show now how one can analyze such *min-max problems* in optimized Schwarz methods, for the specific simplified case and one parameter, i.e. $p_1 = p_2 = p$, which leads to the min-max problem

$$\min_{p \in \mathbb{R}} \max_{k_{\min} \le k \le k_{\max}} \left(\frac{k - p}{k + p} \right)^2 e^{-2k\delta}. \tag{4.66}$$

There are three classical steps for solving such min-max problems, see e.g. [43]: first one finds the range for p in which the solution p^* must lie. Then one identifies the local maxima in k of the convergence factor, and finally one solves the min-max problem using this information. This leads to the following two lemmas and the final convergence theorem.

Lemma 16 (Range of p)**.** *If p^* is solution of the min-max problem* (4.66)*, then $p^* \in (k_{\min}, k_{\max})$.*

Proof. First we notice that p^* can not be negative, since if it were, changing the sign of p^* would make the convergence factor even smaller due to the fraction in the first factor and the fact that $k \ge 0$. Next we compute the partial derivative of $\rho(\delta, k, p) := \left(\frac{k-p}{k+p}\right)^2 e^{-2k\delta}$ with respect to p to find

$$\partial_p \rho(\delta, k, p) = \frac{4(p-k)k}{(p+k)^3} e^{-2k\delta}.$$

Hence, if $p < k_{\min}$, then $\partial_p \rho(\delta, k, p) < 0$, and thus increasing p decreases the convergence factor for all $k \in (k_{\min}, k_{\max})$. Similarly, if $p > k_{\max}$, then $\partial_p \rho(\delta, k, p) > 0$, and thus decreasing p decreases the convergence factor for all $k \in (k_{\min}, k_{\max})$. The optimum p^* must therefore lie in $(k_{\min}, k_{\max})$. □

Lemma 17 (Local maxima of ρ)**.** *For $p \in (k_{\min}, k_{\max})$, the convergence factor has two local maxima $k_{1,2} \in [k_{\min}, k_{\max}]$: one at $k_1 = k_{\min}$, and the second one at*

$$k_1 = \begin{cases} k_{\max} & \text{if } \delta = 0 \text{ or } k_{\max} < \bar{k} := \sqrt{p^2 + 2p/\delta}, \\ \bar{k} & \text{otherwise.} \end{cases} \tag{4.67}$$

Proof. If the overlap $\delta = 0$, the function $\rho(\delta, k, p)$ is convex in k and thus can only have maxima at the endpoints $k_{\min}$ and $k_{\max}$, since it is non-negative. If the overlap $\delta > 0$, computing the derivative of $\rho(\delta, k, p)$ with respect to k gives

$$\partial_k \rho(\delta, k, p) = \frac{2(k-p)(\delta p^2 + 2p - \delta k^2) e^{-2*k\delta}}{(k+p)^3}.$$

Therefore $\rho(\delta, k, p)$ has a minimum point at $k = p$ where its value is $\rho(\delta, p, p) = 0$, and one positive maximum point $\bar{k} = \sqrt{p^2 + 2p/\delta}$ (the other one with the minus sign is outside the range of interest $(k_{\min}, k_{\max})$). This maximum $\bar{k}$ is only of interest in the min-max problem if it lies inside $(k_{\min}, k_{\max})$, and by monotonicity otherwise the boundary point $k_{\max}$ is a local maximum. □

We can now obtain the best choice of the parameter p in the Robin transmission conditions of the optimized Schwarz method under the simplifying assumption. It is also possible to obtain this result on the bounded domain, but it is technically more involved.

Theorem 38 (Convergence of the Optimized Schwarz method)**.** *The solution p^* of the min-max problem* (4.66) *is given by* equioscillation *between the maximum points k_1 and k_2 of Lemma 17,*

$$\rho(\delta, k_1, p^*) = \rho(\delta, k_2, p^*). \tag{4.68}$$

Without overlap, $\delta = 0$, we have

$$p^* = \sqrt{k_{\min} k_{\max}}, \tag{4.69}$$

and the associated optimized convergence factor is then bounded by

$$\max_{k_{\min} \le k \le k_{\max}} \rho(0, k, p^*) = \left(\frac{\sqrt{\frac{k_{\max}}{k_{\min}}} - 1}{\sqrt{\frac{k_{\max}}{k_{\min}}} + 1} \right)^2 \sim 1 - 4\sqrt{\frac{k_{\min}}{k_{\max}}}, \tag{4.70}$$

when $k_{\max}$ is large. For positive overlap $\delta > 0$ small, and when $k_{\max} = \infty$, we have

$$p^* \sim \frac{k_{\min}^{\frac{2}{3}}}{2^{\frac{1}{3}} \delta^{\frac{1}{3}}}, \tag{4.71}$$

and the associated optimized convergence factor is then bounded by

$$\max_{k_{\min} \le k \le k_{\max}} \rho(\delta, k, p^*) \sim 1 - 2^{\frac{7}{3}} k_{\min}^{\frac{1}{3}} \delta^{\frac{1}{3}}. \tag{4.72}$$

Proof. Without overlap, $\delta = 0$, and $p \in (k_{\min}, k_{\max})$, when we increase p, $\rho(0, k_{\min}, p)$ is increasing, and $\rho(0, k_{\max}, p)$ is decreasing, and hence by continuity the solution p^* of (4.66) is attained when (4.68) holds, which leads to the solution p^* in (4.69). The asymptotic result for the convergence factor bound is obtained by expanding the expression in a Taylor series for $k_{\max} = \frac{1}{\varepsilon}$ for ε small. For positive overlap $\delta > 0$, the reasoning for equioscillation is analogous, and the asymptotic result is obtained making the ansatz $p^* \sim C_p \delta^{-\frac{1}{3}}$ and expanding (4.68) for δ small, which shows that the equioscillation equation has indeed a solution for this ansatz. Such computations are best performed in Maple, as we explain below. □

We now show how the asymptotic expansions needed for the proof of Theorem 38 can easily be done in Maple:

```
rho:=((k-p)/(k+p))^2*exp(-2*k*d):          # simplified convergence factor
solve(factor(diff(rho,k)),k);              # find local maximum
```

$$p, \frac{\sqrt{dp(dp+2)}}{d}, -\frac{\sqrt{dp(dp+2)}}{d}$$

```
k:=kmin;R1:=rho;k:=kb;R2:=rho;k:='k':       # local maxima
kb:=sqrt(d*p*(d*p+2))/d:
p:=Cp/d^(1/3):                             # ansatz based on numerics
se1:=series(R1,d,2);                       # series for d small
```

$$se1 := 1 - 4\,\frac{kmin\,d^{1/3}}{Cp} + 8\,\frac{kmin^2 d^{2/3}}{Cp^2} + \left(-2\,kmin - 12\,\frac{kmin^3}{Cp^3}\right) d + O\left(d^{4/3}\right)$$

```
se2:=series(R2,d,2);
```

$$se2 := 1 - 4\,\sqrt{2}\sqrt{Cp}\,d^{1/3} + O\left(d^{2/3}\right)$$

```
Cpsols:=solve(op(2,se1)=op(2,se2),Cp);     # determine constant Cp
Cp:=simplify(Cpsols[1],symbolic);          # choose real root
```

$$Cp := 1/2\,2^{2/3} kmin^{2/3}$$

```
simplify(p,symbolic);
```

$$1/2\,\frac{2^{2/3} kmin^{2/3}}{\sqrt[3]{d}}$$

We show in Table 4.1 a comparison of the optimal choice of the parameter given by the original min-max problem (4.59) and the simplified min-max problem (4.66) when the overlap $\delta = \alpha - \beta$ becomes small. We clearly see that the closed form solution from Theorem 38 of the simplified min-max problem gives a very good approximation of the best parameter p^* of the original min-max problem, especially when the overlap is becoming small, and the algorithm will perform almost as well as with the numerically optimized parameter. This is the case for many partial differential equations, and closed form formulas are available in the literature, see [43] for Laplace type problems, [47, 5, 4] for advection reaction diffusion problems, [52, 48] for Helmholtz equations, [49, 46]

		original min-max (4.59)		simplified min-max (4.66)	
α	β	p^*	$\rho_s(p^*)$	p^*	$\rho_s(p^*)$
0.625	0.375	4.4760	0.0014	2.7026	0.0072
0.51	0.49	7.2842	0.1125	6.2721	0.1320
0.501	0.499	14.6107	0.3793	13.5128	0.3937

Table 4.1: Comparison of the numerical solution of the min-max problem (4.59) and the asymptotic solution of the simplified min-max problem (4.66) from Theorem 38, and corresponding convergence factor of the optimized Schwarz method (not of the simplified problem).

for the wave equation, [34, 103, 10] for Maxwell's equations, and research is currently continuing for further models, especially time harmonic wave propagation [56], and also in the heterogeneous case where different models are used in different subdomains [54]. For further information, it is best to search for optimized Schwarz methods with the corresponding name of the partial differential equation, for example in the proceedings of the international conference series on Domain Decomposition Methods available on `www.ddm.org`.

All Schwarz methods can be generalized to the case of many subdomains, and their implementation does not present any significant difficulties, see for example Problem 47. When the number of subdomains becomes larger, convergence slows down in certain situations, and one needs a coarse correction to fix this, which is well understood in the domain decomposition literature, but goes beyond the introduction to domain decomposition methods here.

4.7 The Dirichlet-Neumann domain decomposition method

> *It is also interesting to note that if a symmetric region is cut in half, and treated fully symmetrically, then $S = 2S^{(1)}$ and the conjugate gradient iteration converges in one step.*
>
> *Petter E. Bjørstad* and *Olof B. Widlund*, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, 1986.

The *Dirichlet-Neumann method* is a *non-overlapping domain decomposition method* going back to the seminal work by Bjorstad and Widlund in 1986, see [9] and the quote above, and belongs to the class of iterative substructuring methods, for a historical review, see [53]. It uses a different convergence mechanism than the alternating Schwarz method, which used overlap in its classical form, and a combination of overlap and transmission conditions in the optimized form. As its name suggests, it uses on one side of the interface between subdomains a Dirichlet transmission condition, and on the other side a Neumann condition. For the simple two subdomain configuration indicated in Figure 4.24, the Dirichlet-Neumann algorithm[12] starts with an initial guess λ^0 along the in-

[12]The Dirichlet-Neumann algorithm was originally introduced directly as a preconditioner for the conjugate gradient method, but we describe the method here as a stationary iteration

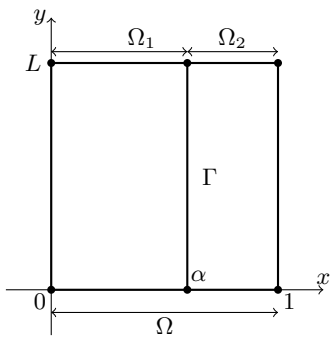


Figure 4.24: Decomposition of the domain $\Omega = (0,1) \times (0,L)$ into $\overline{\Omega} = \overline{\Omega}_1 \cup \overline{\Omega}_2$, where $\Omega_1 := (0,\alpha) \times (0,L)$ and $\Omega_2 := (\alpha,1) \times (0,L)$. The interface is denoted by Γ.

terface Γ, and then computes for our Poisson model problem first a Dirichlet solution on subdomain Ω_1,

$$\begin{array}{rcll} \Delta u_1^n & = & f & \text{in } \Omega_1, \\ u_1^n & = & \lambda^{n-1} & \text{on } \Gamma, \\ u_1^n & = & g & \text{on } \partial\Omega \cap \partial\Omega_1, \end{array} \tag{4.73}$$

followed by computing a Neumann solution on subdomain Ω_2 using as Neumann data the approximation just computed on the subdomain Ω_1,

$$\begin{array}{rcll} \Delta u_2^n & = & f & \text{in } \Omega_2, \\ \partial_{n_2} u_2^n & = & \partial_{n_2} u_1^n & \text{on } \Gamma, \\ u_2^n & = & g & \text{on } \partial\Omega \cap \partial\Omega_2. \end{array} \tag{4.74}$$

Again ∂_{n_2} denotes the unit outer normal derivative of subdomain Ω_2 along the interface Γ, and in our example, we could simply write $\partial_{n_2} = -\partial_x$. The minus sign has however no importance since it appears on both sides and can thus be canceled. Finally in the Dirichlet-Neumann algorithm, the interface approximation λ^n is updated using a *relaxation parameter* θ,

$$\lambda^n = \theta\lambda^{n-1} + (1-\theta)u_2^n \quad \text{on } \Gamma. \tag{4.75}$$

In order to understand the convergence properties of the Dirichlet-Neumann method, we proceed like in our analysis of the Schwarz methods: we introduce the equations satisfied by the errors $e_j^n := u - u_j^n$, $d^n := \lambda - \lambda^n$, where $\lambda := u|_\Gamma$, namely

$$\begin{array}{rcll} \Delta e_1^n & = & f & \text{in } \Omega_1, \\ e_1^n & = & d^{n-1} & \text{on } \Gamma, \\ e_1^n & = & g & \text{on } \partial\Omega \cap \partial\Omega_1, \end{array} \tag{4.76}$$

followed by

$$\begin{array}{rcll} \Delta e_2^n & = & f & \text{in } \Omega_2, \\ \partial_{n_2} e_2^n & = & \partial_{n_2} e_1^n & \text{on } \Gamma, \\ e_2^n & = & g & \text{on } \partial\Omega \cap \partial\Omega_2, \end{array} \tag{4.77}$$

like the alternating Schwarz method.

and with the final update

$$d^n = \theta d^{n-1} + (1-\theta)e_2^n \quad \text{on } \Gamma. \tag{4.78}$$

Using a Fourier sine expansion like in the analysis of the Schwarz method (4.35),

$$e_j^n(x,y) = \sum_{k\in K} \widehat{e}_j^n(x,k)\sin(ky), \tag{4.79}$$

with $K := \{\frac{\pi}{L}, \frac{2\pi}{L}, \frac{3\pi}{L}, \ldots\}$, we find that the Fourier coefficients are of the same form (4.37) as for the Schwarz method,

$$\begin{aligned}\widehat{e}_1^n(x,k) &= 2A_1^n(k)\sinh(kx),\\ \widehat{e}_2^n(x,k) &= -2A_2^n(k)e^k\sinh(k(1-x)).\end{aligned}$$

The Dirichlet transmission condition on subdomain Ω_1 then implies

$$\widehat{e}_1^n(\alpha,k) = \widehat{d}^{n-1}(k) = 2A_1^n(k)\sinh(k\alpha),$$

which gives for the constant

$$A_1^n(k) = \frac{\widehat{d}^{n-1}(k)}{2\sinh(k\alpha)}, \tag{4.80}$$

and hence for the error on Ω_1

$$\widehat{e}_1^n(x,k) = \frac{\sinh(kx)}{\sinh(k\alpha)}\widehat{d}^{n-1}(k). \tag{4.81}$$

Now using the Neumann transmission condition on subdomain Ω_2, we obtain

$$\partial_x\widehat{e}_2^n(x,k)|_{x=\alpha} = 2A_2^n(k)e^k\cosh(k(1-\alpha))k = \partial_x\widehat{e}_1^n(x,k)|_{x=\alpha} = 2A_1^n(k)\cosh(k\alpha)k,$$

which gives for the constant

$$A_2^n(k) = \frac{\cosh(k\alpha)e^{-k}}{\cosh(k(1-\alpha))}A_1^n(k),$$

and inserting (4.80) on the right, we find

$$A_1^n(k) = \frac{1}{2}\frac{\cosh(k\alpha)}{\cosh(k(1-\alpha))}\frac{e^{-k}}{\sinh(k\alpha)}\widehat{d}^{n-1}(k),$$

which finally leads to the error in subdomain Ω_2,

$$\widehat{e}_2^n(x,k) = -\frac{\cosh(k\alpha)}{\cosh(k(1-\alpha))}\frac{\sinh(k(1-x))}{\sinh(k\alpha)}\widehat{d}^{n-1}(k). \tag{4.82}$$

We can now trace this error on the interface Γ and introduce it into the update condition for $\widehat{d}^n$ in (4.78),

$$\widehat{d}^n(k) = \left(\theta - (1-\theta)\frac{\tanh(k(1-\alpha))}{\tanh(k\alpha)}\right)\widehat{d}^{n-1}(k) \quad \text{on } \Gamma.$$

Therefore, the *convergence factor of the Dirichlet-Neumann method* is

$$\widehat{\rho}_{DN}(\alpha,k,\theta)=\left(\theta-(1-\theta)\frac{\tanh(k(1-\alpha))}{\tanh(k\alpha)}\right), \tag{4.83}$$

and by induction the error on the interface satisfies

$$\widehat{d}^{n}(k)=\left(\widehat{\rho}_{DN}(\alpha,k,\theta)\right)^{n}\widehat{d}^{0}(k).$$

In order to understand the convergence behavior of the Dirichlet-Neumann method, we need to study $\widehat{\rho}_{DN}(\alpha,k,\theta)$, for which the following two Lemmas are useful.

Lemma 18 (Properties of F_{DN})**.** *For $\alpha\in(0,1)$ and $k>0$ the function*

$$F_{DN}(\alpha,k)=\frac{\tanh(k(1-\alpha))}{\tanh(k\alpha)} \tag{4.84}$$

has the following properties:

1. $\lim_{k\to\infty}F_{DN}(\alpha,k)=1$*;*
2. *for $\alpha=\frac{1}{2}$, $F_{DN}(\frac{1}{2},k)\equiv 1$;*
3. *for $\alpha>\frac{1}{2}$, $F_{DN}(\alpha,k)$ is growing in k, and $F_{DN}(\alpha,k)<1$;*
4. *for $\alpha<\frac{1}{2}$, $F_{DN}(\alpha,k)$ is decreasing in k, and $F_{DN}(\alpha,k)>1$.*

Proof. 1. We compute directly

$$\begin{aligned}\lim_{k\to\infty}F_{DN}(\alpha,k)&=\lim_{k\to\infty}\frac{e^{k(1-\alpha)}-e^{-k(1-\alpha)}}{e^{k(1-\alpha)}+e^{-k(1-\alpha)}}\frac{e^{k\alpha}+e^{-k\alpha}}{e^{k\alpha}-e^{-k\alpha}}\\&=\lim_{k\to\infty}\frac{1-e^{-2k(1-\alpha)}}{1+e^{-2k(1-\alpha)}}\frac{1+e^{-2k\alpha}}{1-e^{-2k\alpha}}=1.\end{aligned}$$

2. We simply insert $\alpha=\frac{1}{2}$, $F_{DN}(\frac{1}{2},k)=\frac{\tanh(k\frac{1}{2})}{\tanh(k\frac{1}{2})}=1$.

3. Since $\alpha>1/2$, we have that $1-\alpha<\alpha$, which implies that $\tanh(\alpha k)>\tanh((1-\alpha)k)$, where we used the strict monotonicity of the hyperbolic tangent function. Hence $\frac{\tanh((1-\alpha)k)}{\tanh(\alpha k)}<1$. The fact that $F_{DN}(\alpha,k)$ is growing in k follows by inspection, see also the plot in Figure 4.25, obtained with the Maple commands

```
FDN:=tanh(k*(1-alpha))/tanh(k*alpha);
plot3d(FDN,k=Pi..10*Pi,alpha=0..1,axes=boxed);
```

4. This result follows by similar arguments as in 3.

□

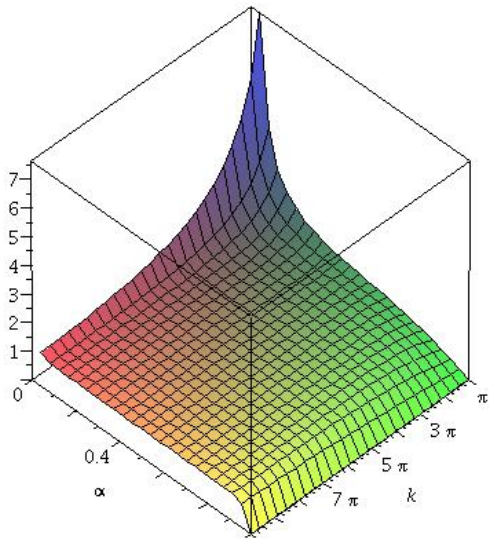


Figure 4.25: Plot of the function $F_{DN}(\alpha, k)$ defined in (4.84) from the Dirichlet-Neumann algorithm.

In order to get some intuition on how the relaxation parameter θ should be chosen in the Dirichlet-Neumann method, we first look at the plot of the convergence factor in modulus shown in Figure 4.26, which we obtained with the Maple commands

```
rhoDN:=theta-(1-theta)*tanh(k*(-alpha+1))/tanh(k*alpha);
alpha:=2/3;
plot([seq(abs(rhoDN),theta=[0.45,0.47,0.49])],k=Pi..5*Pi,
     axes=boxed,legend=['theta=0.45','theta=0.47','theta=0.49']);
```

We clearly see that there is an optimal choice for the relaxation parameter θ: if the left subdomain is bigger, then if θ is too small, low frequencies k converge faster than high frequencies, and if θ is too large, high frequencies converge faster than low frequencies. If the right subdomain is bigger, it is the opposite. From Lemma 18 we also know that if both subdomains have the same size, then the optimal choice is $\theta = \frac{1}{2}$, and the algorithm converges in one iteration, since $\widehat{\rho}_{DN}(\frac{1}{2}, k, \frac{1}{2}) \equiv 0$, for all frequencies k, it becomes a direct solver[13]. If the subdomains are of different size, the following lemma gives the optimal choice θ^* of the relaxation parameter.

Lemma 19 (Optimal relaxation parameter for Dirichlet-Neumann)**.** *The optimal choice θ^* for the relaxation parameter in the Dirichlet-Neumann method is given by*

$$\theta^* = \frac{F_{DN}(\alpha, k_{\max}) + F_{DN}(\alpha, k_{\min})}{2 + F_{DN}(\alpha, k_{\max}) + F_{DN}(\alpha, k_{\min})}, \tag{4.85}$$

[13]Note however that after one iteration, only the interface trace λ is known, one has to do a second solve in the subdomains to obtain the correct solution also inside the subdomains.

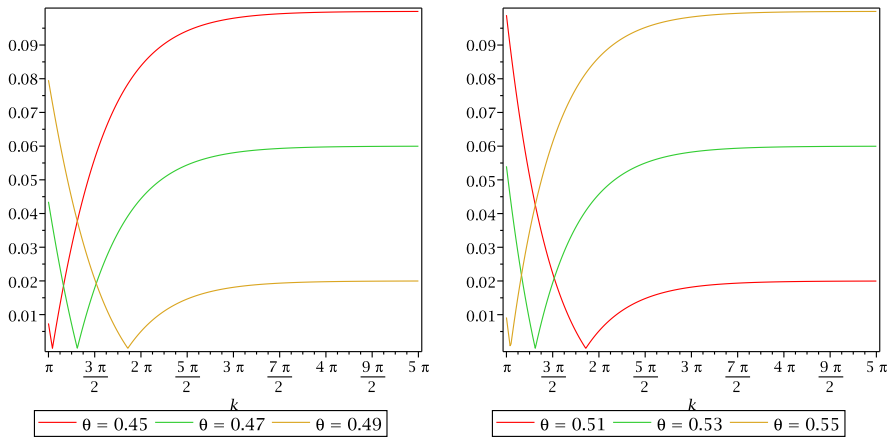


Figure 4.26: Convergence factor of the Dirichlet-Neumann method in modulus, $|\widehat{\rho}_{DN}(\alpha, k, \theta)|$, for interface position $\alpha = 2/3$ on the left (left subdomain bigger), and $\alpha = 1/3$ on the right (right subdomain bigger).

with the function $F_{DN}(\alpha, k)$ defined in (4.84)*. The associated convergence factor of the Dirichlet-Neumann method then satisfies*

$$\max_{k_{\min} \le k \le k_{\max}} |\widehat{\rho}_{DN}(\alpha, k, \theta^*)| = \left| \frac{F_{DN}(\alpha, k_{\max}) - F_{DN}(\alpha, k_{\min})}{2 + F_{DN}(\alpha, k_{\max}) + F_{DN}(\alpha, k_{\min})} \right|. \qquad (4.86)$$

Proof. If $\alpha > \frac{1}{2}$, then $F_{DN}(\alpha, k)$ is increasing in k according to Lemma 18, and hence the maximum of $\widehat{\rho}_{DN}(\alpha, k, \theta) = \theta - (1-\theta)F_{DN}(\alpha, k)$ is attained at $k = k_{\min}$, and the minimum at $k = k_{\max}$, and since $\widehat{\rho}_{DN}(\alpha, k, \theta)$ is a linear function in θ, one can equilibrate the maximum and the minimum to achieve that $\widehat{\rho}_{DN}(\alpha, k, \theta)$ is as close as possible to zero. If $\alpha < \frac{1}{2}$, then $F_{DN}(\alpha, k)$ is decreasing in k according to Lemma 18, and thus the maximum of $\widehat{\rho}_{DN}(\alpha, k, \theta)$ is attained at $k = k_{\max}$, and the minimum at $k = k_{\min}$, and one can again equilibrate the maximum and the minimum. Therefor in both cases, the optimal choice θ^* satisfies the *equioscillation* equation

$$\widehat{\rho}_{DN}(\alpha, k_{\min}, \theta^*) = -\widehat{\rho}_{DN}(\alpha, k_{\max}, \theta^*),$$

which is a linear equation for θ^*,

$$\theta^* - (1-\theta^*)F_{DN}(\alpha, k_{\min}) = -(\theta^* - (1-\theta^*)F_{DN}(\alpha, k_{\max}))$$

that leads to the solution given in (4.85). Inserting this choice into $\widehat{\rho}_{DN}(\alpha, k_{\min}, \theta^*)$ (or evaluated at $k_{\max}$ since their modulus is now equal), we get

$$\begin{aligned}
\widehat{\rho}_{DN}(\alpha, k_{\min}, \theta^*) &= \theta^* - (1-\theta^*)F_{DN}(\alpha, k_{\min}) \\
&= \tfrac{F_{DN}(\alpha,k_{\max})+F_{DN}(\alpha,k_{\min})}{2+F_{DN}(\alpha,k_{\max})+F_{DN}(\alpha,k_{\min})} - \left(1 - \tfrac{F_{DN}(\alpha,k_{\max})+F_{DN}(\alpha,k_{\min})}{2+F_{DN}(\alpha,k_{\max})+F_{DN}(\alpha,k_{\min})}\right) F_{DN}(\alpha, k_{\min}) \\
&= \tfrac{F_{DN}(\alpha,k_{\max})-F_{DN}(\alpha,k_{\min})}{2+F_{DN}(\alpha,k_{\max})+F_{DN}(\alpha,k_{\min})},
\end{aligned}$$

which concludes the proof. □

We can now show that with this optimized choice of the relaxation parameter θ, the Dirichlet-Neumann method is convergent.

Theorem 39 (Convergence of the Dirichlet-Neumann method)**.** *With the optimized choice of the relaxation parameter*

$$\theta^* = \frac{1 + \tilde{F}_{DN}(\alpha, L)}{3 + \tilde{F}_{DN}(\alpha, L)}, \tag{4.87}$$

where

$$\tilde{F}_{DN}(\alpha, L) := \frac{\tanh(\frac{\pi}{L}(1-\alpha))}{\tanh(\frac{\pi}{L}\alpha)}, \tag{4.88}$$

the Dirichlet-Neumann method (4.73)*,* (4.74) *and* (4.75) *converges for all initial guesses λ^0 along the interface Γ, and we have the linear convergence estimate*

$$||\lambda - \lambda^n||_2 \leq \rho_{DN}^n ||\lambda - \lambda^0||_2, \tag{4.89}$$

with the convergence factor

$$\rho_{DN} = \left| \frac{1 - \tilde{F}_{DN}(\alpha, L)}{3 + \tilde{F}_{DN}(\alpha, L)} \right| < 1. \tag{4.90}$$

Proof. When $k_{\max}$ goes to infinity, Lemma 18 shows that $F_{DN}(\alpha, k_{\max}) \to 1$, and inserting this into (4.86) of Lemma 19, we obtain

$$\max_{k_{\min} \leq k \leq \infty} |\widehat{\rho}_{DN}(\alpha, k, \theta^*)| = \left| \frac{1 - F_{DN}(\alpha, \frac{\pi}{L})}{3 + F_{DN}(\alpha, \frac{\pi}{L})} \right|,$$

where we also used that $k_{\min} = \frac{\pi}{L}$. Since $F_{DN}(\alpha, \frac{\pi}{L})$ is positive, this quantity is clearly less than 1, and we can then use the Parseval-Plancherel identity to obtain the L^2 error estimate (4.89). □

We show in Figure 4.27 how the convergence factor $\rho_{DN}(\alpha, L)$ of the Dirichlet-Neumann method from Theorem 4.89 depends on the interface position α and the domain height L. We see that convergence is better for small L than for large L like for the Schwarz methods, the lateral boundary conditions help convergence. We also see that a small Dirichlet subdomain is worse than a small Neumann subdomain, and comparable size is best for convergence, in contrast to the Schwarz method that converges better when one subdomain is small compared to the other, see the right plot in Figure 4.14. The convergence result in Theorem 4.89 shows that if the two subdomains are equal, $\alpha = \frac{1}{2}$, the Dirichlet-Neumann method is a direct solver, since $\tilde{F}_{DN}(\frac{1}{2}, L) = 1$, which implies $\theta^* = \frac{1}{2}$ as we have seen earlier and $\rho_{DN} = 0$, which is clearly visible in Figure 4.27. Note in addition also that for the choice $\theta = \frac{1}{2}$, we have

$$\lim_{k \to \infty} \widehat{\rho}_{DN}(\alpha, k, \frac{1}{2}) = \frac{1}{2} - \frac{1}{2} F_{DN}(\alpha, k) = 0$$

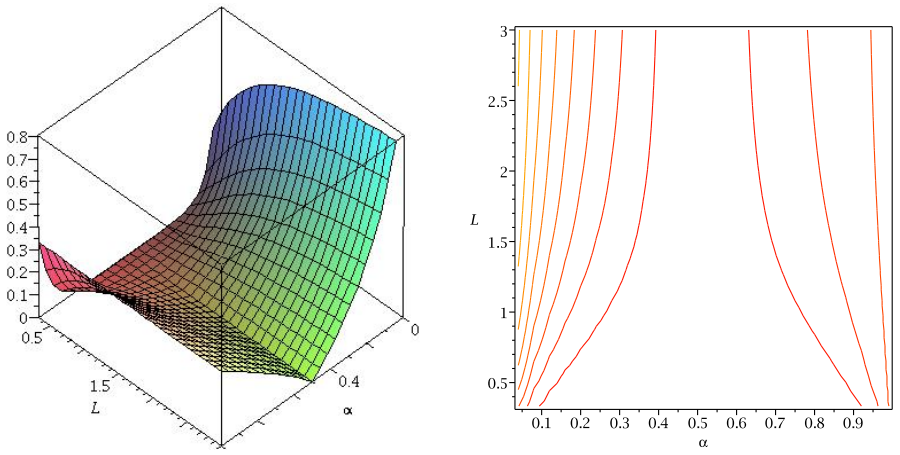


Figure 4.27: Dependence of the convergence factor $\rho_{DN}(\alpha, L)$ of the Dirichlet-Neumann method from Theorem 4.89 as a function of the interface position α, and the domain height L, with a 3D plot on the left, and the corresponding contour plot on the right, with the level sets corresponding to $\{0, 0.1, \ldots, 1\}$.

because of Lemma 18, and thus for high frequencies, the Dirichlet-Neumann method always converges very fast for the choice of relaxation parameter $\theta = \frac{1}{2}$, independently of the interface position α. This is because the damping nature of the Poisson equation does not allow high frequencies to propagate very far, and so they can not see that the subdomain decomposition is not symmetric. This is the fundamental reason why the Dirichlet-Neumann method converges robustly for all frequencies, even when $k_{\max}$ becomes large, and thus Dirichlet-Neumann converges independently of the mesh size h, which gives $k_{\max} \sim \frac{\pi}{h}$ as we have seen in (4.60). This is also the case for Schwarz methods, if the overlap size does not depend on the mesh size, as we have seen in Figure 4.17, but in practice often the overlap is proportional to the mesh size, since it is only a few mesh cells wide.

The Dirichlet-Neumann method can also be discretized to be used as a numerical solver. It is convenient for this to first implement a subdomain solver with general Robin boundary conditions on the left and right, as for example the one here in Matlab:

```
function U=Solve2dR(f,ai,bi,gl,gr,p1,p2)
% SOLVE2DR solves 2d Poisson problem using Robin conditions
%   u=Solve2dR(f,ai,bi,gl,gr,p1,p2) solves the two dimensional
%   Poisson equation Delta u=f on the domain Omega=(ai*h,bi*h)x(0,1) with
%   Robin boundary conditions (dn+p1)u=gl at x=ai*h and (dn+p2)u=gr at
%   x=bi*h and u=0 at y=0 and y=1 using a finite difference
%   approximation with interior grid points (bi-ai) times length(gl)
%   using the same mesh size h=1/(length(gl)+1) in both x and y.

nx=bi-ai+1;                                % number of point in x direction
```

```
ny=length(gl);                                  % number of point in y direction
h=1/(ny+1);                                     % mesh size
ex=ones(nx,1);                                  % construct 1d finite differences
Dxx=spdiags([ex/h^2 -2/h^2*ex ex/h^2],[-1 0 1],nx,nx);
ey=ones(ny,1);
Dyy=spdiags([ey/h^2 -2/h^2*ey ey/h^2],[-1 0 1],ny,ny);
A=kron(speye(size(Dxx)),Dyy)+kron(Dxx,speye(size(Dyy)));
A(1:ny,1:ny)=A(1:ny,1:ny)/2+p1/h*speye(ny);  % put Robin conditions
A(end-ny+1:end,end-ny+1:end)=A(end-ny+1:end,end-ny+1:end)/2+p2/h*speye(ny);
f(1:ny,1)=f(1:ny,1)/2+gl/h;                     % add boundary conditions into rhs
f(1:ny,end)=f(1:ny,end)/2+gr/h;
u=A\f(:);                                       % solve by sparse Gaussian el.
U=reshape(u,ny,nx);                             % put solution into matrix
```

We can then use this solver to solve both subdomain problems using Dirichlet conditions and Neumann conditions, as in the following implementation of the Dirichlet-Neumann method:

```
m=15;                                % number of gridpoints
h=1/(m+1);                           % include the h^2 scaling for
A=Laplacian(m,2)/h^2;                % the true five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1/h^2;   % include h^2 scaling also in rhs
u=A\f;                               % since true solver Solve2dR is used
a=6;                                 % alpha=(ai+1)*h
F1=zeros(m,a+1); F1(m,:)=-1/h^2;     % f for subdomain solves must
F2=zeros(m,m-a+2); F2(m,:)=-1/h^2;   % include Robin bc columns
x1=0:h:a*h; x2=a*h:h:1; y=0:h:1;     % finite difference meshes
z1=zeros(1,a+1);z2=zeros(1,m-a+2);   % for plotting purposes
o1=ones(1,a+1);o2=ones(1,m-a+2);     % for plotting purposes
e=ones(m,1);                         % construct normal derivative
Na=[-speye(m) spdiags([-e +4*e -e]/2,[-1 0 1],m,m)]/h;
la=zeros(m,1);                       % zero initial guess
U2=zeros(size(F2));
f1=@(al,L) tanh(pi/L*(1-al))/tanh(pi/L*al);
th=(1+f1(a*h,1))/(3+f1(a*h,1));      % optimized relaxation parameter
pe=1e10;                             % 1e10 to emulate a Dirichlet condition
g=zeros(m,1);                        % left and right Dirichlet condition
for n=1:10
  err(n)=norm(la-u((a-1)*m+1:a*m),'inf');
  U1=Solve2dR(F1,0,a,g*pe,la*pe,pe,pe);
  mesh(x1,y,[z1;U1;o1]); hold on; mesh(x2,y,[z2;U2;o2]); hold off
  xlabel('x'); ylabel('y'); zlabel('Dirichlet Step');
  axis([0 1 0 1 0 1])
  pause
  ta=Na*[U1(:,end-1);U1(:,end)]+F2(:,1)*h/2;
  U2=Solve2dR(F2,a,m+1,ta,g*pe,0,pe);
  la=th*la+(1-th)*U2(:,1);
  mesh(x1,y,[z1;U1;o1]); hold on; mesh(x2,y,[z2;U2;o2]); hold off
  xlabel('x'); ylabel('y'); zlabel('Neumann Step');
  axis([0 1 0 1 0 1])
  pause
```

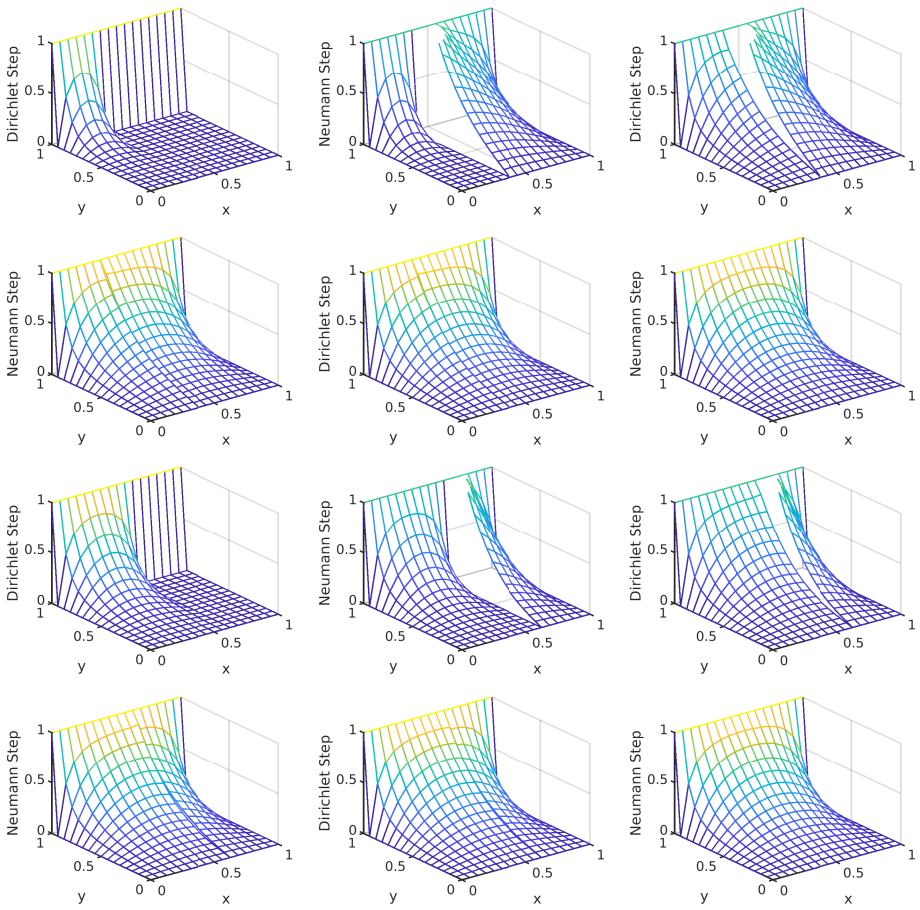


Figure 4.28: First iterations of the Dirichlet-Neumann method with optimized relaxation parameter θ^* from Theorem 39, applied to our Laplace model problem. Top 2 rows: $\alpha = 0.375$. Next two rows: $\alpha = 0.625$.

```
end
```

Note that to obtain Dirichlet conditions, we need to choose a very large Robin parameter in this implementation, which we call `pe`. Running this code, with different positions for the interface indicated by `a`, we get the results in Figure 4.28. We see that the Dirichlet-Neumann method converges for the optimized choice of the relaxation parameter, adapted to the interface position α. Note that for an arbitrary choice of the relaxation parameter, the Dirichlet-Neumann method can also diverge. If we put the interface into the middle, so that the two subdomains are identical in size, we get the results shown in Figure 4.29. We see that indeed as predicted, the Dirichlet-Neumann method becomes a direct solver: after the first iteration, the predicted interface value λ^1 coincides with

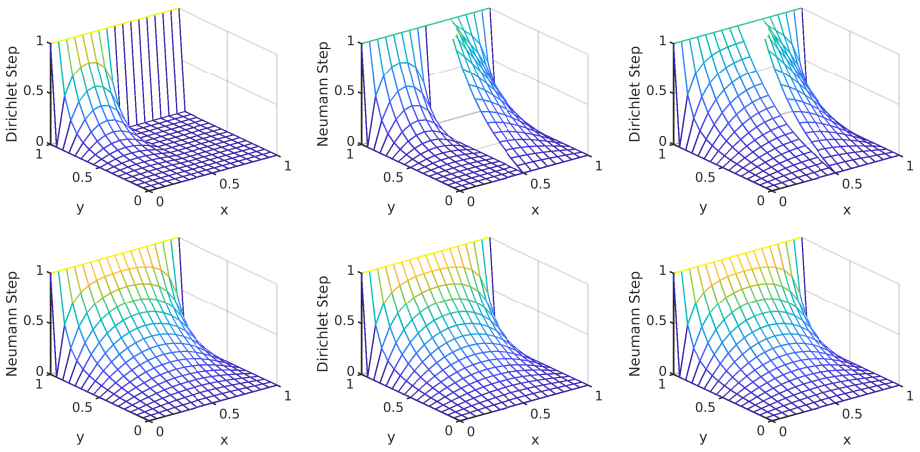


Figure 4.29: First iterations of the Dirichlet-Neumann method with symmetric subdomains, $\alpha = \frac{1}{2}$, where the optimized relaxation parameter $\theta^* = \frac{1}{2}$ from Theorem 39, applied to our Laplace model problem.

the solution of the problem, so that the next subdomain solves deliver the exact solution also in volume, and the algorithm has arrived at its fixed point in the second row of Figure 4.29.

To illustrate how the Dirichlet-Neumann method converges independently of the mesh size, we show in Figure 4.30 how the error decays in the discretized Dirichlet-Neumann iteration for the mesh we used for Figure 4.28 with $m = 15$ and $\alpha = 0.375$, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points, keeping the interface fixed at $\alpha = 0.375$. We clearly see that convergence is independent of the mesh parameter h, and very well predicted by the continuous analysis in Theorem 39 for the optimized parameter θ^*.[14]

There is an important comment here to make about the *relaxation parameter* , which seems so important for the Dirichlet-Neumann method used as a stationary iterative solver. As soon as one uses the method as a preconditioner for a Krylov method, this parameter becomes completely irrelevant! To see this, consider an arbitrary stationary iterative method, to which we add in the end a relaxation parameter,

$$\begin{aligned} \boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{u}^n + M^{-1}(\boldsymbol{f} - A\boldsymbol{u}^n) \\ \boldsymbol{u}^{n+1} &= \theta\boldsymbol{u}^n + (1-\theta)\boldsymbol{u}^{n+\frac{1}{2}} \\ &= \theta\boldsymbol{u}^n + (1-\theta)(\boldsymbol{u}^n + M^{-1}(f - A\boldsymbol{u}^n)). \end{aligned}$$

Using this method as a preconditioner for a Krylov method is equivalent to applying the Krylov method to the equation at the fixed point, where it reads

$$\boldsymbol{u} = \theta\boldsymbol{u} + (1-\theta)(\boldsymbol{u} + M^{-1}(\boldsymbol{f} - A\boldsymbol{u})) \iff (1-\theta)M^{-1}A\boldsymbol{u} = (1-\theta)M^{-1}\boldsymbol{f},$$

[14] Convergence flattens out at the level chosen to emulate the Dirichlet conditions, `1/pe=1e-10`.

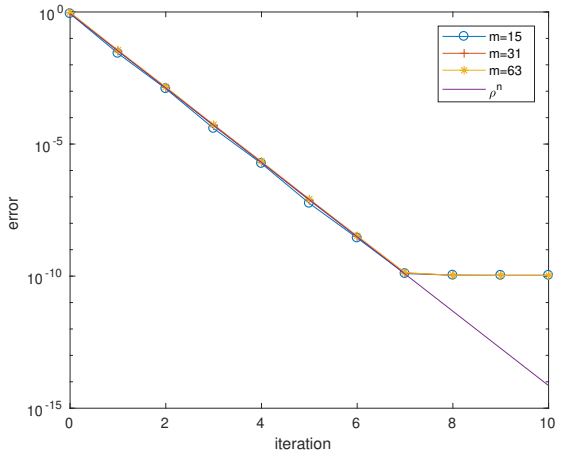


Figure 4.30: Convergence of the Dirichlet-Neumann method for different mesh sizes $h = \frac{1}{m+1}$. For comparison also the theoretical convergence estimate based on the continuous analysis from Theorem 39 is shown.

and we see that the multiplication of the preconditioned system by $(1 - \theta)$, which comes from the relaxation step, will have no effect whatsoever on the convergence properties of the Krylov method which is used to solve it. So the Krylov method takes care of any possible relaxation at the end of an iteration. This is very different for the optimization in the Robin transmission conditions of the optimized Schwarz method, which truly improves the preconditioner.

As a final remark, the Dirichlet-Neumann method can be generalized to decompositions into more than just two subdomains, otherwise it would not be interesting for parallel computing. One has however to take care when assigning which interface side takes Dirichlet data and which takes Neumann data, for an example, see Figure 4.31. There are only very few studies in the literature on how such assignments affect the convergence of the Dirichlet-Neumann method, and different assignments then require different schedulings for the order in which subdomain problems are solved, see for example [51]. These are precisely the difficulties which make the Dirichlet-Neumann method less popular in practice. The Neumann-Neumann method in the following section does not have this problem, but it shares a further drawback of these non-overlapping methods: in the presence of a cross point, the middle point in Figure 4.31 where more than two subdomains meet, the methods are in general not well defined at the continuous level, see for example [17].

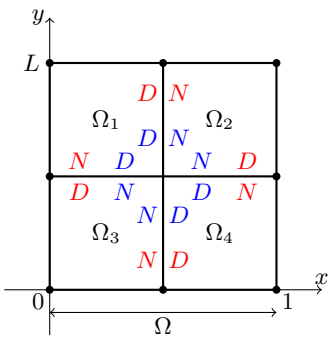


Figure 4.31: Decomposition of the domain $\Omega = (0,1) \times (0,L)$ into four subdomains with a cross point, and possible assignments of Dirichlet and Neumann conditions for the Dirichlet-Neumann algorithm, one in red and one in blue.

4.8 The Neumann-Neumann domain decomposition method

> *Ce préconditionneur agit sur l'opérateur de Steklov-Poincaré (représenté après discrétisation par la matrice complément de Schur) par un calcul de moyenne de traces et la résolution d'un problème de Neumann par sous-domaine.*
>
> *Jean-François Bourgat, Roland Glowinski, Patrick Le Tallec, Marina Vidrascu*, Variational Formulation and Algorithm for Trace Operator in Domain Decomposition Calculations, 1989.

The *Neumann-Neumann method* is also a *non-overlapping domain decomposition* method, invented by Bourgat, Glowinski, Le Tallec and Vidrascu in 1989, see [11] and the quote above; a brief historical review can be found in [53]. Its name can be a bit confusing, since at each iteration, first Dirichlet problems are solved on each subdomain, followed by Neumann correction problems, also on each subdomain, so each iteration is twice as expensive as the iterations of the earlier domain decomposition methods we have seen. If the order is reversed, and one starts with the Neumann problems, followed by Dirichlet correction problems, the algorithm became known under the name *FETI (Finite Element Tearing and Interconnect)*, invented by Farhat and Roux in 1991, see [38]. Since the two algorithms are very much related, we will only study the Neumann-Neumann method in detail here. For the simple two subdomain configuration shown in Figure 4.24, the Neumann-Neumann algorithm starts with an initial guess λ^0 along the interface Γ, and then computes for our Poisson model problem first two Dirichlet solutions, one on each subdomain Ω_j, $j = 1, 2$,

$$\begin{aligned} \Delta u_j^n &= f && \text{in } \Omega_j, \\ u_j^n &= \lambda^{n-1} && \text{on } \Gamma, \\ u_j^n &= g && \text{on } \partial\Omega \cap \partial\Omega_j, \end{aligned} \tag{4.91}$$

followed by computing two Neumann corrections, one on each subdomain, using

as Neumann data the jump remaining after the Dirichlet step[15],

$$\begin{array}{rcll} \Delta\psi_j^n &=& 0 & \text{in } \Omega_j, \\ \partial_{n_j}\psi_j^n &=& \partial_{n_1}u_1^n + \partial_{n_2}u_2^n & \text{on } \Gamma, \\ \psi_j^n &=& 0 & \text{on } \partial\Omega \cap \partial\Omega_j. \end{array} \tag{4.92}$$

Here ∂_{n_j}, $j = 1, 2$ denotes the unit outer normal derivative of subdomain Ω_j along the interface Γ, and in our example, we could simply write $\partial_{n_1} = \partial_x$, $\partial_{n_2} = -\partial_x$. Finally, like in the Dirichlet-Neumann algorithm, the interface approximation λ^n is updated using a *relaxation parameter* θ, traditionally written in the form

$$\lambda^n = \lambda^{n-1} - \theta(\psi_1^n + \psi_2^n) \quad \text{on } \Gamma. \tag{4.93}$$

To understand the convergence properties of the Neumann-Neumann method, we use again Fourier analysis for the *error equations*: with $e_j^n := u - u_j^n$, $d^n := \lambda - \lambda^n$, $\lambda := u|_\Gamma$, and leaving the ψ_j, which are already correction terms, these quantities satisfy for $j = 1, 2$

$$\begin{array}{rcll} \Delta e_j^n &=& 0 & \text{in } \Omega_j, \\ e_j^n &=& d^{n-1} & \text{on } \Gamma, \\ e_j^n &=& 0 & \text{on } \partial\Omega \cap \partial\Omega_j, \end{array} \tag{4.94}$$

and

$$\begin{array}{rcll} \Delta\psi_j^n &=& 0 & \text{in } \Omega_j, \\ \partial_{n_j}\psi_2^n &=& \partial_{n_1}e_1^n + \partial_{n_2}e_2^n & \text{on } \Gamma, \\ \psi_j^n &=& 0 & \text{on } \partial\Omega \cap \partial\Omega_j, \end{array} \tag{4.95}$$

followed by the final update

$$d^n = d^{n-1} - \theta(\psi_1^n + \psi_2^n) \quad \text{on } \Gamma. \tag{4.96}$$

Using a Fourier sine expansion,

$$\begin{aligned} e_j^n(x, y) &= \sum_{k\in K} \widehat{e}_j^n(x, k)\sin(ky), \\ \psi_j^n(x, y) &= \sum_{k\in K} \widehat{\psi}_j^n(x, k)\sin(ky), \end{aligned} \tag{4.97}$$

with $K := \{\frac{\pi}{L}, \frac{2\pi}{L}, \frac{3\pi}{L}, \ldots\}$, we find that the Fourier coefficients are of the form

$$\begin{aligned} \widehat{e}_1^n(x, k) &= \frac{\sinh(kx)}{\sinh(k\alpha)}\widehat{d}^{n-1}(k), \quad & \widehat{e}_2^n(x, k) &= \frac{\sinh(k(1-x))}{\sinh(k(1-\alpha))}\widehat{d}^{n-1}(k), \\ \widehat{\psi}_1^n(x, k) &= B_1^n(k)\frac{\sinh(kx)}{\sinh(k\alpha)}, & \widehat{\psi}_2^n(x, k) &= B_2^n(k)\frac{\sinh(k(1-x))}{\sinh(k(1-\alpha))}. \end{aligned} \tag{4.98}$$

[15]In the original presentation of the Neumann-Neumann algorithm, there is a factor one half in front of the sum of the Neumann derivatives, $\frac{1}{2}(\partial_{n_1}u_1^n + \partial_{n_2}u_2^n)$, but this just scales the relaxation parameter θ that follows, so we do not put this factor here.

To determine the remaining constants $B_j^n(k)$ in the $\widehat{\psi}_j^n$, we need to compute the right hand side in the Neumann boundary condition (4.95),

$$\begin{aligned}(\partial_{n_1}\widehat{e}_1^n + \partial_{n_2}\widehat{e}_2^n)|_{x=\alpha} &= (\partial_x\widehat{e}_1^n - \partial_x\widehat{e}_2^n)|_{x=\alpha} \\ &= \left(\frac{k\cosh(k\alpha)}{\sinh(k\alpha)} + \frac{k\cosh(k(1-\alpha))}{\sinh(k(1-\alpha))}\right)\widehat{d}^{n-1} \\ &= k\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1}.\end{aligned}$$

Imposing this as Neumann boundary condition on the corrections ψ_j^n leads to

$$\begin{aligned}B_1^n(k)\frac{k\cosh(k\alpha)}{\sinh(k\alpha)} &= k\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1}, \\ B_2^n(k)\frac{k\cosh(k(1-\alpha))}{\sinh(k(1-\alpha))} &= k\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1}.\end{aligned}$$

Solving for the $B_j^n(k)$, we get

$$\begin{aligned}B_1^n(k) &= \tanh(k\alpha)\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1}, \\ B_2^n(k) &= \tanh(k(1-\alpha))\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1},\end{aligned}$$

and inserting them into the $\widehat{\psi}_j^n$ leads to

$$\begin{aligned}\widehat{\psi}_1^n(x,k) &= \tanh(k\alpha)\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\frac{\sinh(kx)}{\sinh(k\alpha)}\widehat{d}^{n-1}, \\ \widehat{\psi}_2^n(x,k) &= \tanh(k(1-\alpha))\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\frac{\sinh(k(1-x))}{\sinh(k(1-\alpha))}\widehat{d}^{n-1}.\end{aligned}$$

We can then evaluate the updating formula (4.96) which contains the sum $\widehat{\psi}_1^n(\alpha,k)+\widehat{\psi}_2^n(\alpha,k)$ to find

$$\widehat{d}^n = \widehat{d}^{n-1} - \theta\big(\tanh(k\alpha)+\tanh(k(1-\alpha))\big)\big(\coth(k\alpha)+\coth(k(1-\alpha))\big)\widehat{d}^{n-1} \quad \text{on } \Gamma,$$

which implies that the *convergence factor of the Neumann-Neumann method* is given by

$$\widehat{\rho}_{NN}(\alpha,k,\theta) = 1 - \theta\big(\tanh(k\alpha)+\tanh(k(1-\alpha))\big)\big(\coth(k\alpha)+\coth(k(1-\alpha))\big). \tag{4.99}$$

By induction, the error on the interface satisfies

$$\widehat{d}^n(k) = \left(\widehat{\rho}_{NN}(\alpha,k,\theta)\right)^n \widehat{d}^0(k).$$

To understand the convergence behavior of the Neumann-Neumann method, we need to study $\widehat{\rho}_{NN}(\alpha,k,\theta)$, for which we use again two Lemmas.

Lemma 20 (Properties of F_{NN})**.** *The function*

$$F_{NN}(\alpha,k) = \big(\tanh(k\alpha)+\tanh(k(1-\alpha))\big)\big(\coth(k\alpha)+\coth(k(1-\alpha))\big) \tag{4.100}$$

has the following properties:

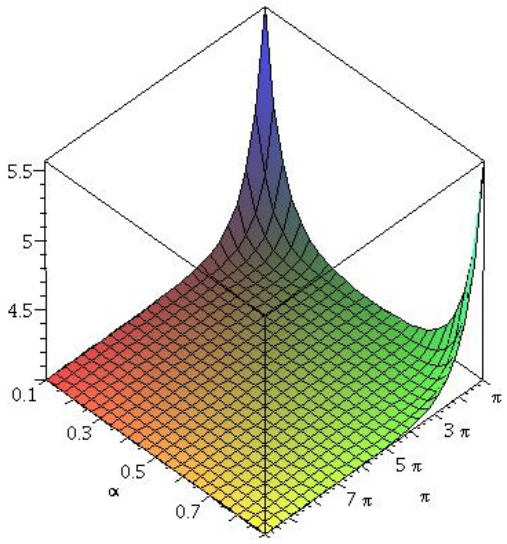


Figure 4.32: Plot of the function $F_{NN}(\alpha, k)$ defined in (4.100) from the Neumann-Neumann algorithm.

1. $\lim_{k=\infty} F_{NN}(\alpha, k) = 4$;
2. *for* $\alpha = \frac{1}{2}$, $F_{NN}(\frac{1}{2}, k) \equiv 4$;
3. $F_{NN}(\alpha, k)$ *is decreasing in* k.

Proof. These results can be obtained by direct computations, analogously to those in the proof of Lemma 18 for the Dirichlet-Neumann method, see also Figure 4.32. □

Like in the Dirichlet-Neumann method, there is an optimal choice for the relaxation parameter θ:

Lemma 21 (Optimal relaxation parameter for Neumann-Neumann)**.** *The optimal choice* θ^* *for the relaxation parameter in the Neumann-Neumann method is given by*

$$\theta^* = \frac{2}{F_{NN}(\alpha, k_{\max}) + F_{NN}(\alpha, k_{\min})}, \tag{4.101}$$

with the function $F_{NN}(\alpha, k)$ *defined in* (4.100)*. The associated convergence factor of the Neumann-Neumann method then satisfies*

$$\max_{k_{\min} \le k \le k_{\max}} |\widehat{\rho}_{NN}(\alpha, k, \theta^*)| = \left| \frac{F_{NN}(\alpha, k_{\max}) - F_{NN}(\alpha, k_{\min})}{F_{NN}(\alpha, k_{\max}) + F_{NN}(\alpha, k_{\min})} \right|. \tag{4.102}$$

Proof. Since $F_{NN}(\alpha, k)$ is decreasing in k according to Lemma 20, the maximum of $\widehat{\rho}_{NN}(\alpha, k, \theta) = 1 - \theta F_{NN}(\alpha, k)$ is attained at $k = k_{\max}$, and the minimum at

$k = k_{\min}$, and because $\widehat{\rho}_{NN}(\alpha, k, \theta)$ is a linear function in θ, one can equilibrate the maximum and the minimum to get $\widehat{\rho}_{NN}(\alpha, k, \theta)$ as close as possible to zero. The optimal choice θ^* therefore satisfies the equioscillation equation

$$\widehat{\rho}_{NN}(\alpha, k_{\min}, \theta^*) = -\widehat{\rho}_{NN}(\alpha, k_{\max}, \theta^*),$$

which gives (4.101), and a direct computation leads to (4.102). □

With this optimized choice of the relaxation parameter θ, the Neumann-Neumann method is convergent.

Theorem 40 (Convergence of the Neumann-Neumann method)**.** *With the optimized choice of the relaxation parameter*

$$\theta^* = \frac{2}{4 + \tilde{F}_{NN}(\alpha, L)}, \tag{4.103}$$

where

$$\tilde{F}_{NN}(\alpha, L) := \left(\tanh(\tfrac{\pi}{L}\alpha) + \tanh(\tfrac{\pi}{L}(1-\alpha)\right)\left(\coth(\tfrac{\pi}{L}\alpha) + \coth(\tfrac{\pi}{L}(1-\alpha)\right), \tag{4.104}$$

the Neumann-Neumann method (4.91)*,* (4.92) *and* (4.93) *converges for all initial guesses λ^0 along the interface Γ, and we have the linear convergence estimate*

$$\|\lambda - \lambda^n\|_2 \leq \rho_{NN}^n \|\lambda - \lambda^0\|_2, \tag{4.105}$$

with the convergence factor

$$\rho_{NN}(\alpha, L) = \left|\frac{4 - \tilde{F}_{NN}(\alpha, L)}{4 + \tilde{F}_{NN}(\alpha, L)}\right| < 1. \tag{4.106}$$

Proof. When $k_{\max}$ goes to infinity, Lemma 20 shows that $F_{NN}(\alpha, k_{\max}) \to 4$, and inserting this into (4.102) of Lemma 21, we obtain

$$\max_{k_{\min} \leq k \leq \infty} |\widehat{\rho}_{NN}(\alpha, k, \theta^*)| = \left|\frac{4 - F_{NN}(\alpha, \frac{\pi}{L})}{4 + F_{NN}(\alpha, \frac{\pi}{L})}\right|,$$

where we also used that $k_{\min} = \frac{\pi}{L}$. Since $F_{NN}(\alpha, \frac{\pi}{L})$ is positive, this quantity is clearly less than 1, and we can then use the Parseval-Plancherel identity to obtain the L^2 error estimate (4.105). □

We show in Figure 4.33 how the convergence factor $\rho_{NN}(\alpha, L)$ in (4.106) of the Neumann-Neumann method from Theorem 40 depends on the interface position α and the domain height L. We again see that convergence is better for small L than for large L like for the earlier domain decomposition methods. We also see that the convergence behavior is now symmetric in α, in contrast to the Dirichlet-Neumann method, but again convergence is worse when one subdomain is smaller than the other, in contrast to the Schwarz method, see the right plot in Figure 4.14. The convergence result in Theorem 40 also shows

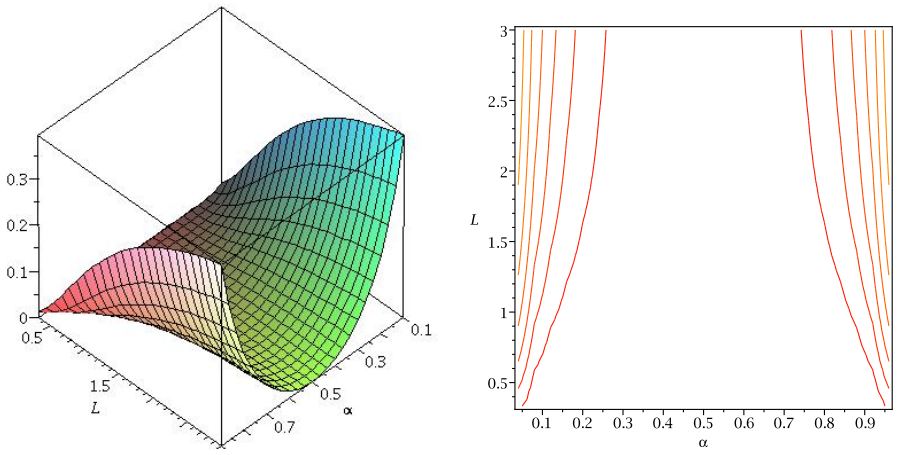


Figure 4.33: Dependence of the convergence factor $\rho_{NN}(\alpha, L)$ of the Neumann-Neumann method from Theorem 40 as a function of the interface position α, and the domain height L, with a 3D plot on the left, and the corresponding contour plot on the right, with the level sets corresponding to $\{0, 0.1, \ldots, 1\}$.

that if the two subdomains are equal, $\alpha = \frac{1}{2}$, the Neumann-Neumann method is a direct solver, since $\tilde{F}_{NN}(\frac{1}{2}, L) = 4$, which implies $\theta^* = \frac{1}{4}$ as we have seen earlier and $\rho_{NN} = 0$, which is clearly visible in Figure 4.33. Like for Dirichlet-Neumann, we also have for the special choice $\theta = \frac{1}{4}$ that

$$\lim_{k\to\infty} \widehat{\rho}_{NN}(\alpha, k, \frac{1}{2}) = 1 - \frac{1}{4} \lim_{k\to\infty} F_{NN}(\alpha, k) = 0$$

because of Lemma 20, and thus for high frequencies, the Neumann-Neumann method always converges very fast for the choice of relaxation parameter $\theta = \frac{1}{4}$, independently of the interface position α. This is again because of the damping nature of the Poisson equation, as in the case of Dirichlet-Neumann.

The Neumann-Neumann method can also be discretized to obtain a preconditioner, and it is again convenient to use the same solver `Solve2dR` with Robin boundary conditions as for Dirichlet-Neumann. We can then use this solver to solve both subdomain problems using Dirichlet conditions followed by both problems using Neumann conditions, as in the following implementation of the Neumann-Neumann method:

```
m=15;                                  % number of gridpoints
h=1/(m+1);                             % include the h^2 scaling for
A=Laplacian(m,2)/h^2;                  % the true five point Laplacian
f=zeros(m*m,1); f(m:m:end)=-1/h^2;     % include h^2 scaling also in rhs
u=A\f;                                 % since true solver Solve2dR is used
a=6;                                   % alpha=ai*h
F1=zeros(m,a+1); F1(m,:)=-1/h^2;       % f for subdomain solves must
F2=zeros(m,m-a+2); F2(m,:)=-1/h^2;     % include Robin bc columns
x1=0:h:a*h; x2=a*h:h:1; y=0:h:1;       % finite difference meshes
```

```
z1=zeros(1,a+1);z2=zeros(1,m-a+2);   % for plotting purposes
o1=ones(1,a+1);o2=ones(1,m-a+2);     % for plotting purposes
e=ones(m,1);                         % construct normal derivative
Na=[-speye(m) spdiags([-e +4*e -e]/2,[-1 0 1],m,m)]/h;
Nb=[spdiags([-e +4*e -e]/2,[-1 0 1],m,m) -speye(m)]/h;
la=zeros(m,1);                       % zero initial guess
U2=zeros(size(F2));
f1=@(al,L) (tanh(pi/L*al)+tanh(pi/L*(1-al)))*(coth(pi/L*al)+coth(pi/L*(1-al)));
th=2/(4+f1(a*h,1));                  % optimized relaxation parameter
pe=1e10;                             % 1e10 to emulate a Dirichlet condition
g=zeros(m,1);                        % left and right Dirichlet condition
for n=1:10
  err(n)=norm(la-u((a-1)*m+1:a*m),'inf');
  U1=Solve2dR(F1,0,a,g*pe,la*pe,pe,pe);
  U2=Solve2dR(F2,a,m+1,la*pe,g*pe,pe,pe);
  mesh(x1,y,[z1;U1;o1]); hold on;mesh(x2,y,[z2;U2;o2]);hold off
  xlabel('x'); ylabel('y'); zlabel('Neumann-Neumann Iterate');
  axis([0 1 0 1 0 1])
  pause
  tb=Nb*[U2(:,1);U2(:,2)]+F2(:,1)*h/2;
  ta=Na*[U1(:,end-1);U1(:,end)]+F1(:,end)*h/2;
  psi1=Solve2dR(zeros(size(F1)),0,a,pe*g,ta+tb,pe,0);
  psi2=Solve2dR(zeros(size(F2)),a,m+1,ta+tb,g*pe,0,pe);
  la=la+th*(psi1(:,end)+psi2(:,1));
  mesh(x1,y,[z1;psi1;z1]); hold on;mesh(x2,y,[z2;psi2;z2]);hold off
  xlabel('x'); ylabel('y'); zlabel('Neumann-Neumann Correction');
  pause
end
```

Running this code, with different positions for the interface indicated by `a`, we get the results in Figure 4.34. We see that the Neumann-Neumann method converges for the optimized choice of the relaxation parameter, adapted to the interface position α, and convergence is very fast, as one can see from the different scales in the plots of the corrections. If we put the interface into the middle, so that the two subdomains are identical in size, we get the results shown in Figure 4.35. We see that indeed as predicted, the Neumann-Neumann method becomes a direct solver: after the first iteration, following the correction, the predicted interface value λ^1 coincides with the solution of the problem, so that the next subdomain Dirichlet solves deliver the solution also in volume, and the algorithm has arrived at its fixed point in the second row of Figure 4.35, the corrections are numerically zero.

To illustrate how the Neumann-Neumann method converges independently of the mesh size, we show in Figure 4.36 how the error decays in the discretized Neumann-Neumann iteration for the mesh we used for Figure 4.34 with $m = 15$ and $\alpha = 0.375$, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points, keeping the interface fixed at $\alpha = 0.375$. We see that convergence is independent of the mesh parameter h, and very well predicted

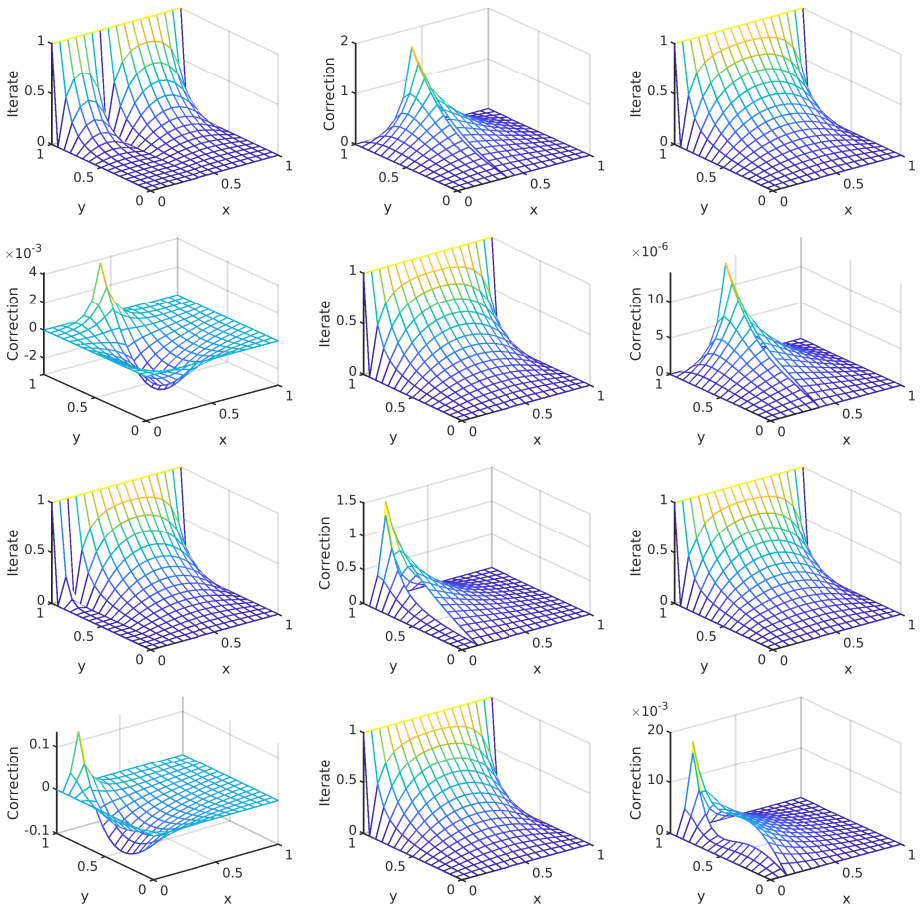


Figure 4.34: First iterations and corrections of the Neumann-Neumann method with optimized relaxation parameter θ^* from Theorem 39, applied to our Laplace model problem. Top 2 rows: $\alpha = 0.375$. Next two rows: $\alpha = 0.125$.

by the continuous analysis in Theorem 40 for the optimized parameter θ^*.[16]

As for Dirichlet-Neumann, the relaxation parameter is only important when Neumann-Neumann is used as an iterative solver, as a preconditioner this is not needed, for the same reasons as explained at the end of the Dirichlet-Neumann section.

The generalization to many subdomains is much easier than for Dirichlet-Neumann, since no decisions on the interfaces have to be made, all subdomains must solve a Dirichlet problem followed by a Neumann problem. This Neumann problem can however pose difficulties when a subdomain is completely

[16]Convergence flattens out again at the level chosen to emulate the Dirichlet conditions, `1/pe=1e-10`.

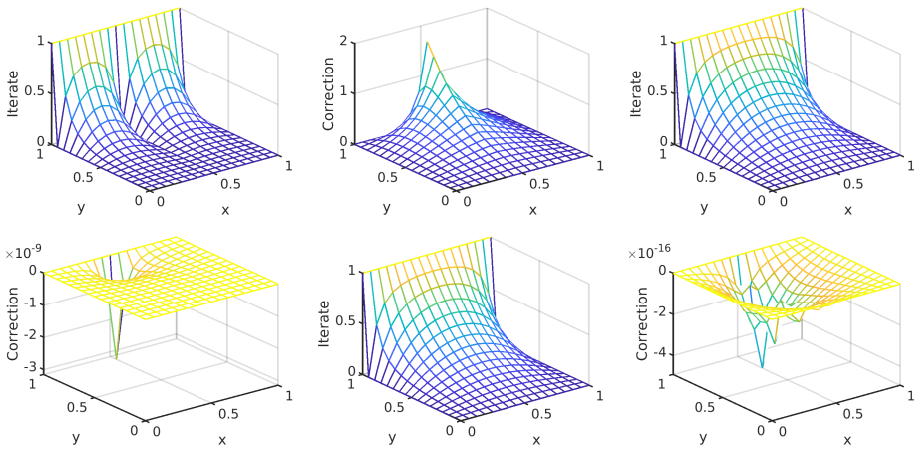


Figure 4.35: First iterations of the Neumann-Neumann method with symmetric subdomains, $\alpha = \frac{1}{2}$, where the optimized relaxation parameter $\theta^* = \frac{1}{2}$ from Theorem 39, applied to our Laplace model problem.

surrounded by other subdomains, so that it only has Neumann boundary conditions. It is then not invertible, the solution is only determined up to a constant. This was first considered as a disadvantage, but it turned out that one can use this constant to add a coarse correction to the method and make it scalable when many subdomains are used, leading to the Balancing Neumann-Neumann method: the bug became a feature, and the same happened for the FETI methods.

But like the Dirichlet-Neumann method, cross points make the Neumann-Neumann method not well posed at the continuous level, and so the analyses were so far only performed at the discrete level in the literature.

4.9 Comparison of Schwarz, Dirichlet-Neumann and Neumann-Neumann

To conclude this section about domain decomposition preconditioners, we show in Figure 4.37 a comparison of the convergence factors of the classical and optimized Schwarz methods, and the Dirichlet-Neumann and the Neumann-Neumann methods, as a function of the interface position α. For the Schwarz methods, we used an overlap $\delta = 0.1$, which we centered around the interface position. We clearly see that if the subdomains are close to be the same size, the Dirichlet-Neumann and Neumann-Neumann methods are hard to beat, since their convergence factor tends to zero when the subdomains are exactly symmetric. We also see that the Dirichlet-Neumann method converges in an asymmetric way, as expected from the asymmetric solution process. It also becomes evident that the classical Schwarz method is not competitive, it would need a lot more

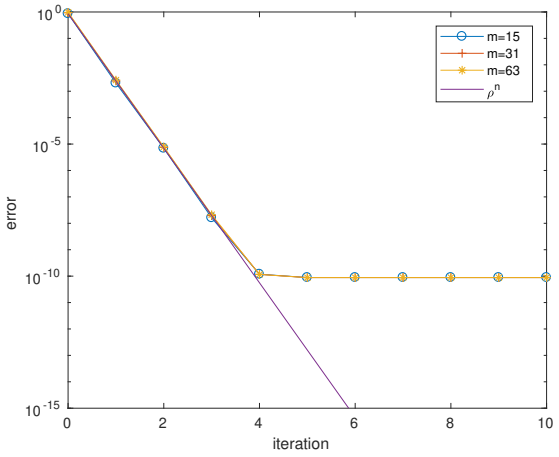


Figure 4.36: Convergence of the Neumann-Neumann method for different mesh sizes $h = \frac{1}{m+1}$. For comparison also the theoretical convergence estimate based on the continuous analysis from Theorem 40 is shown.

overlap to beat Dirichlet-Neumann and Neumann-Neumann when the subdomain size becomes quite different. The optimized Schwarz methods are however very competitive, and their performance does not deteriorate with the subdomain asymmetry, it becomes better. It is this robustness property of optimized Schwarz methods that makes them so interesting: they can even take advantage of heterogeneities in the PDE, and converge faster than without heterogeneity, whereas classical domain decomposition methods try to be only robust.

4.10 Multigrid methods

> *The characteristic feature of the multi-grid iteration is its fast convergence. The convergence speed does not deteriorate when the discretization is refined, whereas classical iterative methods slow down for decreasing grid size. As a consequence one obtains an acceptable approximation of the discrete problem at the expense of computational work proportional to the number of unknowns, which is also the number of the equations in the system. It is not only the complexity which is optimal, also the constant of proportionality is so small that other methods can hardly surpass the multi-grid efficiency.*
>
> Wolfgang Hackbusch, Multi-Grid Methods and Applications, 1985.

In this section, we present multigrid methods, which are among the most efficient methods to solve linear systems $A\boldsymbol{u} = \boldsymbol{f}$ stemming from discretizations of Laplace like boundary value problems. In contrast to the domain decomposition methods we have seen, which inherit their robust convergence when the mesh is refined from the fact that they were formulated at the continuous level, multigrid methods have mesh independent convergence for a different reason: they use many levels of approximation, which classical stationary iterative methods like Jacobi and Gauss-Seidel do not, see also the quote above.

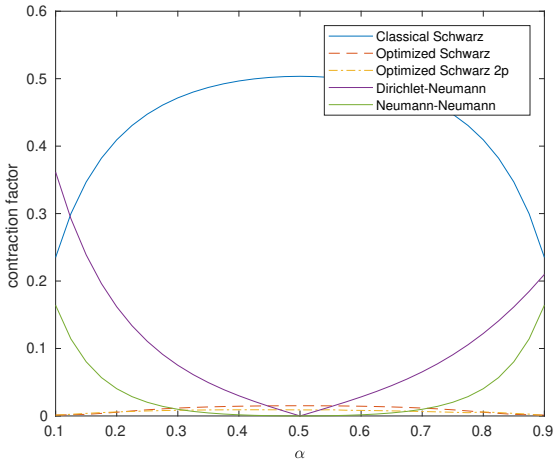


Figure 4.37: Convergence factors for the various domain decomposition methods and two subdomain decompositions, as a function of the interface position α.

Early important ideas for multigrid methods can already be found in the work of *Radii Petrovich Fedorenko* [39], but seminal contributions are due to *Achi Brandt* [13] and *Wolfgang Hackbusch* [69] in the seventies, who both greatly fostered the tremendous development of multigrid methods to come, and there are many variants now of multigrid methods. Further important contributions were also made by *Roy A. Nicolaides* [92], especially also for coarse spaces used in domain decomposition. To make multigrid methods more easily applicable to general matrix problems, algebraic multigrid methods were developped by *John W. Ruge* and *Klaus Stüben* [106], but we will not treat such methods in our simple first introduction to multigrid methods here.

To describe the basic multigrid method, we follow the analysis presented by Hackbusch in [70, Chapter 2] and use the simple one-dimensional Laplace equation

$$-u_{xx} = f \text{ in } \Omega = (0,1) \text{ with } u(0) = u(1) = 0. \tag{4.107}$$

To motivate this choice of a one-dimensional model, we recall the following excerpt from [70, Chapter 2]:

> A natural model of an elliptic equation is the two-dimensional Poisson equation in a square. Unfortunately, the analysis of the multigrid algorithm for this problem is still an involved exercise.

A two-dimensional Fourier analysis for the Poisson equation in a square is also possible and given in [70, Chapter 8]. However, the one-dimensional analysis we consider is simpler and then more appropriate to introduce the main ideas of multigrid methods. Notice also that other more general convergence analyses are possible, see, e.g., [70, Chapters 6 and 7]. However, they only lead in general to upper bounds on the contraction factor.

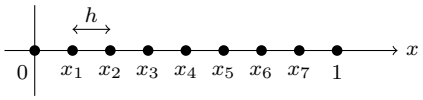


Figure 4.38: Grid of 7 points inside $(0,1)$.

To discretize (4.107), we use a uniform grid of $m = 2^\ell - 1$ gridpoints in $(0,1)$ with $h = \frac{1}{m+1}$, where ℓ is a given integer. For example if $\ell = 3$ we have a grid of $m = 7$ points inside $(0,1)$, see Figure 4.38. The corresponding linear system $A\boldsymbol{u} = \boldsymbol{f}$ is given by

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \tag{4.108}$$

where $f_j = f(x_j)$ for $j = 1, \ldots, m$ and $x_j = jh$.

If we use the Jacobi method in the correction form to solve this linear system, we get

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + D^{-1}(\boldsymbol{f} - A\boldsymbol{u}^n),$$

where $D = \frac{2}{h^2}I$. Now, recall that the eigenvectors and eigenvalues of A are given by

$$\varphi_{k,j} = \sin(k\pi x_j) \text{ and } \lambda_k = -\frac{4}{h^2}\sin^2\Big(k\pi\frac{h}{2}\Big),$$

for $k = 1, \ldots, m$. We call the frequencies k close to 1 low-frequencies, the ones close to m high-frequencies, and the ones close to $m/2$ medium frequencies, since this corresponds to how the associated eigenfunctions oscillate. If we consider the error at the iteration $n = 0$ to be $\boldsymbol{e}^0 = \boldsymbol{\varphi}_k$, then the Jacobi iteration gives after one iteration the error (recall Theorem 1)

$$\begin{aligned} \boldsymbol{e}^1 &= \boldsymbol{e}^0 - D^{-1}A\boldsymbol{e}^0 \\ &= \boldsymbol{\varphi}_k + \frac{h^2}{2}\lambda_k\boldsymbol{\varphi}_k \\ &= \Big[1 - \frac{h^2}{2}\frac{4}{h^2}\sin^2\Big(k\pi\frac{h}{2}\Big)\Big]\boldsymbol{\varphi}_k \\ &= \rho_k\boldsymbol{\varphi}_k, \end{aligned}$$

where the convergence factor is defined as $\rho_k := 1 - 2\sin^2\big(k\pi\frac{h}{2}\big)$. In Figure 4.39, we show a plot of the convergence factor ρ_k as a function of k, which illustrates that ρ_k assumes in modulus values close to 1 for low and high frequencies, and is close to 0 for medium frequencies. This means that the Jacobi method for

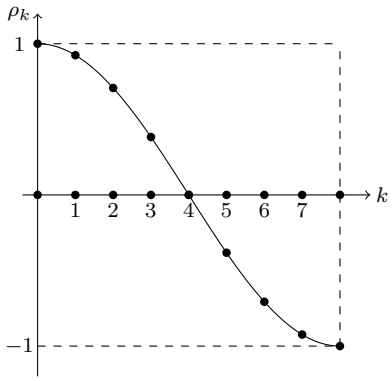


Figure 4.39: Convergence factor ρ_k: example for $m = 7$, which illustrates that $\rho_k \approx 1$ for $k \approx 1$ and $\rho_k \approx -1$ for $k \approx m = 7$.

$\boldsymbol{e}^0 = \boldsymbol{\varphi}_k$ converges very fast for a medium frequency k and very slowly for k close to 1 or to m.

This behavior of Jacobi can be controlled with a relaxation[17] (or damping) parameter $\omega \in (0, 1)$,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \omega D^{-1}(\boldsymbol{f} - A\boldsymbol{u}^n). \tag{4.109}$$

Computing the *convergence factor for this damped Jacobi iteration* as before, we obtain the convergence factor

$$\rho_{k,\omega} = 1 - 2\omega \sin^2\Big(k\pi\frac{h}{2}\Big).$$

We can now use the parameter ω to tune the convergence of the Jacobi method for different frequencies k. For example, with $\omega = \frac{1}{2}$ Jacobi converges very fast for high frequencies error components $\boldsymbol{e}^0 = \boldsymbol{\varphi}_k$ with k close to m, see Figure 4.40. An optimal choice to damp the upper half of the frequencies is $\omega = \frac{2}{3}$, which leads to the best overall damping of these frequencies. This value of ω is obtained by setting the convergence factor $\rho_{\frac{m}{2},\omega} = -\rho_{m,\omega}$ following the Chebyshev *equioscillation principle*[18] and solving for ω, see Figure 4.40. However, all the curves shown in Figure 4.40 are very close to 1 for low frequencies. This means that after ν iterations the error $\boldsymbol{e}^\nu$, which satisfies

$$A\boldsymbol{e}^\nu = A(\boldsymbol{u} - \boldsymbol{u}^\nu) = \boldsymbol{f} - A\boldsymbol{u}^\nu = \boldsymbol{r}^\nu, \tag{4.110}$$

contains mostly low frequencies, and we see that ω cannot be used effectively to improve the convergence of the method of Jacobi for low frequencies. The

[17] The method (4.109) is known as damped Jacobi method and corresponds to a splitting $A = M - N$ with $M = \frac{1}{\omega}D$ and $N = \frac{1}{\omega}[(1-\omega)D - \omega(L+U)]$.

[18] This principle goes back to the work of Chebyshev [19] and his discovery of the Chebyshev polynomials, which were motivated by minimizing the wear and tear of the force transmission mechanism for steam engins.

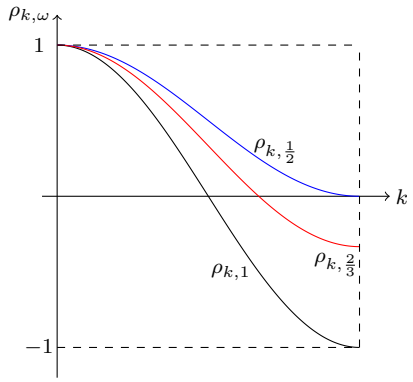


Figure 4.40: Convergence factor $\rho_{k,\omega}$: example for $\omega = 1$ corresponding to undamped Jacobi (black line), $\omega = \frac{1}{2}$ (blue line), and $\omega = \frac{2}{3}$ (red line).

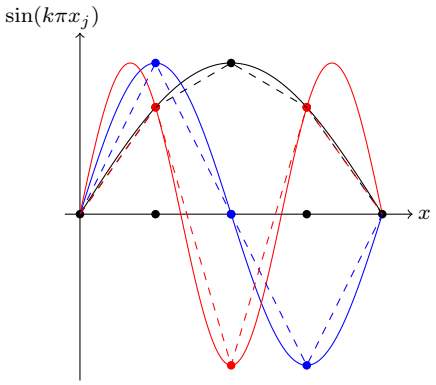


Figure 4.41: Low frequencies $k = 1$ (black), $k = 2$ (blue), and $k = 3$ (red) and their approximations on a coarse grid (dashed lines).

slow convergence of the low frequencies can however be addressed by a different mechanism: low frequencies can be well represented on a coarser grid using a smaller number of grid points, e.g. $M = 2^{\ell-1} - 1$[19] with coarse mesh size $H = \frac{1}{M+1}$. We show in Figure 4.41 the coarse grid for our initial example, where we only used 3 points instead of the original 7, for the three lowest frequencies. We see that the frequencies $k = 1$ and $k = 2$ are very well approximated with only 3 grid points; the approximation of the frequency $k = 3$ is not quite as good, but it is still oscillating correctly three times when represented by only 3 grid points.

Now, recalling that after ν iterations the error vector $\boldsymbol{e}^\nu$ contains mostly low

[19]This M has no relation to the symbol M we used for the preconditioning matrix.

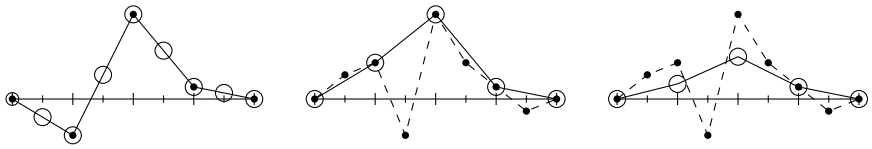


Figure 4.42: Interpolation and restriction by injection and full weighting.

frequencies, one can solve (4.110) only on a coarse grid:

$$\begin{aligned} A_{\mathrm{H}}\boldsymbol{e}_{\mathrm{H}} &= R(\boldsymbol{f} - A\boldsymbol{u}^{\nu}) = R\boldsymbol{r}^{\nu}, \\ \boldsymbol{u}^{new} &= \boldsymbol{u}^{\nu} + P\boldsymbol{e}_{\mathrm{H}}, \end{aligned} \tag{4.111}$$

where $P \in \mathbb{R}^{m\times M}$ is an *interpolation matrix* (or *prolongation operator*) that interpolates from an approximation on the coarse grid to the fine grid, given for linear interpolation in our example by

$$P = \begin{bmatrix} \frac{1}{2} & & \\ 1 & & \\ \frac{1}{2} & \frac{1}{2} & \\ & 1 & \\ & \frac{1}{2} & \frac{1}{2} \\ & & 1 \\ & & \frac{1}{2} \end{bmatrix}_{7\times 3},$$

and $R \in \mathbb{R}^{M\times m}$ is a *restriction matrix* which restricts a solution given on the fine grid to the coarse grid nodes. There are two common choices for R: the so-called injection, which simply selects the values on the fine grid where it coincides with the coarse grid, given in our example by

$$R := \begin{bmatrix} 0 & 1 & 0 & & & & \\ & & 0 & 1 & 0 & & \\ & & & & 0 & 1 & 0 \end{bmatrix}_{3\times 7},$$

and the *full weighting restriction matrix* $R := \frac{1}{2}P^{\top}$, which also takes into account values on the fine grid where there is no coarse grid node by averaging. We show in Figure 4.42 the action of the interpolation P and the restriction R graphically for our example. On the left, we see the action of the interpolation: the black points represent a vector $\boldsymbol{v} \in \mathbb{R}^3$ defined on the coarse grid, not including the boundary points where the function equals zero, and the circles represent the interpolated vector $\boldsymbol{w} = P\boldsymbol{v} \in \mathbb{R}^7$ on the fine grid, again not including the zero boundary points. In the middle of Figure 4.42, we show the action of the restriction operator R by injection: the black points represent a vector $\boldsymbol{w} \in \mathbb{R}^7$ defined on the fine mesh not including the zero boundary points, and the circles are the restricted vector $\boldsymbol{v} = R\boldsymbol{w} \in \mathbb{R}^3$ by injection, again without the zero boundary points. On the right in Figure 4.42, we show the action of the fully weighted restriction R: the black points represent a vector $\boldsymbol{w}$

defined on the fine mesh, and the circles are the fully weighted restricted vector $\boldsymbol{v} = R\boldsymbol{w}$.

Using the damped Jacobi method together with the coarse grid correction leads to one complete step of a two grid algorithm starting with an initial guess $\boldsymbol{u}_0$, namely

$$\begin{aligned} \boldsymbol{u}^n &= \boldsymbol{u}^{n-1} + \omega D^{-1}(\boldsymbol{f} - A\boldsymbol{u}^{n-1}) \quad \text{for } n = 1, 2, \ldots, \nu_1, \\ \boldsymbol{u}^0 &= \boldsymbol{u}^{\nu_1} + PA_{\mathrm{H}}^{-1}R(\boldsymbol{f} - A\boldsymbol{u}^{\nu_1}), \\ \boldsymbol{u}^n &= \boldsymbol{u}^{n-1} + \omega D^{-1}(\boldsymbol{f} - A\boldsymbol{u}^{n-1}) \quad \text{for } n = 1, 2, \ldots, \nu_2. \end{aligned} \tag{4.112}$$

The method is first performing ν_1 so-called *pre-smoothing steps* using damped Jacobi. Then the residual is computed and restricted to a coarse grid, where a coarse problem is solved, and the correction is prolongated to the fine grid and added to the current approximation. This is called the *coarse correction*. Then again ν_2 steps of damped Jacobi are applied, called *post-smoothing* . This represents one complete step of a stationary iterative method that can be written for the error in the form

$$\boldsymbol{e}^1 = G\boldsymbol{e}^0,$$

where the iteration matrix G is given by

$$G = \Big(I - \omega D^{-1}A\Big)^{\nu_2}\Big[I - PA_{\mathrm{H}}^{-1}RA\Big]\Big(I - \omega D^{-1}A\Big)^{\nu_1}, \tag{4.113}$$

as one can read off from (4.112): the first part of G from the right, i.e. $(I - \omega D^{-1}A)^{\nu_1}$ corresponds to the pre-smoothing, the second part $[I - PA_{\mathrm{H}}^{-1}RA]$ is the coarse-grid correction, and the last part $(I - \omega D^{-1}A)^{\nu_2}$ represents the post-smoothing.

Unfortunately the iteration matrix G of the two-grid method can not be diagonalized as easily as the Jacobi iteration. It is however possible to easily study the effect of G on two modes simultaneously, a low-frequency component and a corresponding high-frequency component, and we thus analyze the convergence of the two-grid method for an initial error given by

$$\boldsymbol{e}^0 = e_k\boldsymbol{\varphi}_k + e_{\widetilde{k}}\boldsymbol{\varphi}_{\widetilde{k}} = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}e_k \\ e_{\widetilde{k}}\end{bmatrix},$$

where $\widetilde{k} = m + 1 - k$. The following two Lemmas show that it is because of the restriction R and prolongation P that one has to use a low frequency mode and its high frequency companion together in the analysis.

Lemma 22 (Action of the fully weighted restriction operator)**.** *Let the eigenfunction of the coarse discrete Laplacian matrix A_H be denoted by $\phi_{k,j} := \sin(k\pi 2jh)$ for $k = 1, \ldots, m$ with m odd. Then the fully weighted restriction matrix R combines a low frequency mode $\boldsymbol{\varphi}_k$ together with its high frequency counterpart $\boldsymbol{\varphi}_{\widetilde{k}}$, with $\widetilde{k} = m + 1 - k$, into a corresponding coarse mode $\boldsymbol{\phi}_k$,*

$$R\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}e_k \\ e_{\widetilde{k}}\end{bmatrix} = R\big(e_k\boldsymbol{\varphi}_k + e_{\widetilde{k}}\boldsymbol{\varphi}_{\widetilde{k}}\big) = (e_kc_k^2 - e_{\widetilde{k}}s_k^2)\boldsymbol{\phi}_k = \boldsymbol{\phi}_k\begin{bmatrix}c_k^2 & -s_k^2\end{bmatrix}\begin{bmatrix}e_k \\ e_{\widetilde{k}}\end{bmatrix},$$

with the abbreviations $c_k := \cos(k\pi\frac{h}{2})$, $s_k := \sin(k\pi\frac{h}{2})$. *The middle mode is mapped to zero by the restriction,* $R\boldsymbol{\varphi}_{\frac{m+1}{2}} = 0$.

Proof. The proof is obtained by a direct calculation using trigonometric identities: for $k \leq \frac{m+1}{2}$, we obtain for $j \leq \frac{m+1}{2}$ that

$$\begin{aligned}\big[R\boldsymbol{\varphi}_k\big]_j &= \frac{1}{4}\big(\varphi_{k,2j-1} + 2\varphi_{k,2j} + \varphi_{k,2j+1}\big)\\ &= \frac{1}{4}\big(\sin(k\pi(2j-1)h) + 2\sin(k\pi 2jh) + \sin(k\pi(2j+1)h)\big)\\ &= \frac{1}{4}\big(2\sin(k\pi 2jh) + 2\cos(k\pi h)\sin(k\pi 2jh)\big),\end{aligned}$$

where we used the trigonometric identity $\sin(a-b) + \sin(a+b) = 2\sin a\cos b$. We can then further simplify to obtain

$$\begin{aligned}\big[R\boldsymbol{\varphi}_k\big]_j &= \frac{1}{4}\big(2\sin(k\pi 2jh) + 2\cos(k\pi h)\sin(k\pi 2jh)\big)\\ &= \frac{1}{2}\big(1+\cos(k\pi h)\big)\sin(k\pi 2jh)\\ &= \cos^2\Big(k\pi\frac{h}{2}\Big)\sin(k\pi 2jh)\\ &= c_k^2\phi_{k,j},\end{aligned} \tag{4.114}$$

where we used the identity $\cos^2\big(\frac{a}{2}\big) = \frac{1+\cos a}{2}$ and the definition of c_k and of the coarse mode $\boldsymbol{\phi}_k$.

For the middle mode $k = \frac{m+1}{2}$, one has that $\sin(k\pi 2jh) = \sin\big(\frac{m+1}{2}\pi 2j\frac{1}{m+1}\big) = 0$, which together with (4.114) implies that $R\boldsymbol{\varphi}_{\frac{m+1}{2}} = 0$.

Next, we consider the complementary high frequency mode $\boldsymbol{\varphi}_{\widetilde{k}}$, and notice that

$$\begin{aligned}\varphi_{\widetilde{k},j} &= \sin(\widetilde{k}\pi jh) = \sin((m+1-k)\pi jh)\\ &= \sin\Big((m+1)\pi j\frac{1}{m+1} - k\pi jh\Big) = -(-1)^j\sin(k\pi jh),\end{aligned} \tag{4.115}$$

where we used the trigonometric identity $\sin(a-b) = \sin a\cos b - \cos a\sin b$. We can now compute similarly the action of the restriction R on the complementary

mode, and using (4.115) we obtain

$$\begin{aligned}
\big[R\boldsymbol{\varphi}_{\widetilde{k}}\big]_j &= \frac{1}{4}\big(\varphi_{\widetilde{k},2j-1} + 2\varphi_{\widetilde{k},2j} + \varphi_{\widetilde{k},2j+1}\big) \\
&= \frac{1}{4}\big(\sin(\widetilde{k}\pi(2j-1)h) + 2\sin(\widetilde{k}\pi 2jh) + \sin(\widetilde{k}\pi(2j+1)h)\big) \\
&= \frac{1}{4}\Big[-(-1)^{2j-1}\sin(k\pi(2j-1)h) - 2(-1)^{2j}\sin(k\pi 2jh) - (-1)^{2j+1}\sin(k\pi(2j+1)h)\Big] \\
&= -\frac{1}{4}\big(2\sin(k\pi 2jh) - 2\cos(k\pi h)\sin(k\pi 2jh)\big) \\
&= -\frac{1}{2}\big(1-\cos(k\pi h)\big)\sin(k\pi 2jh) \\
&= -\sin^2\Big(k\pi\frac{h}{2}\Big)\sin(k\pi 2jh) \\
&= -s_k^2\phi_{k,j},
\end{aligned}$$

where we used the trigonometric identity $\sin^2\big(\frac{a}{2}\big) = \frac{1-\cos a}{2}$. This concludes our proof. □

Lemma 23 (Action of the prolongation operator)**.** *The prolongation operator P generates from a coarse eigenmode $\boldsymbol{\phi}_k$ of the coarse discretized Laplacian A_H a low frequency mode $\boldsymbol{\varphi}_k$ and the corresponding high frequency mode $\boldsymbol{\varphi}_{\widetilde{k}}$ according to*

$$P\boldsymbol{\phi}_k = (c_k^2\boldsymbol{\varphi}_k - s_k^2\boldsymbol{\varphi}_{\widetilde{k}}) = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix} c_k^2 \\ -s_k^2\end{bmatrix}.$$

Proof. Recalling the definition of P, we have to distinguish between the cases of j even and odd. For j odd, using that $H = 2h$, we obtain

$$\begin{aligned}
[P\boldsymbol{\phi}_k]_j &= \frac{1}{2}\big(\phi_{k,\frac{j-1}{2}} + \phi_{k,\frac{j+1}{2}}\big) \\
&= \frac{1}{2}\Big[\sin\big(k\pi\frac{j-1}{2}H\big) + \sin\big(k\pi\frac{j+1}{2}H\big)\Big] \\
&= \frac{1}{2}\Big[\sin\big(k\pi(j-1)h\big) + \sin\big(k\pi(j+1)h\big)\Big] \\
&= \cos(k\pi h)\sin(kj\pi h),
\end{aligned} \tag{4.116}$$

where we used the trigonometric identity $\sin(a-b) + \sin(a+b) = 2\sin a\cos b$. Now, we notice that

$$\begin{aligned}
&\frac{1}{2}\big(1-\cos(k\pi h)\big) = \sin^2\Big(k\pi\frac{h}{2}\Big) \\
\implies\quad & 1 - 2\sin^2\Big(k\pi\frac{h}{2}\Big) = \cos(k\pi h) \\
\implies\quad & -\sin^2\Big(k\pi\frac{h}{2}\Big) + \cos^2\Big(k\pi\frac{h}{2}\Big) = \cos(k\pi h) \\
\implies\quad & \cos(k\pi h) = c_k^2 - s_k^2,
\end{aligned} \tag{4.117}$$

and

$$\begin{aligned}\sin((m+1-k)j\pi h) &= \sin\Big((m+1)j\pi\frac{1}{m+1} - kj\pi h\Big)\\ &= \sin(j\pi)\cos(kj\pi h) - \cos(j\pi)\sin(kj\pi h)\\ &= \sin(kj\pi h),\end{aligned} \tag{4.118}$$

where we used that j is odd and the trigonometric identity $\sin(a-b) = \sin a\cos b - \cos a\sin b$. Replacing (4.117) into (4.116) and using (4.118), we obtain

$$\begin{aligned}[P\boldsymbol{\phi}_k]_j &= (c_k^2 - s_k^2)\sin(kj\pi h)\\ &= c_k^2\sin(kj\pi h) - s_k^2\sin((m+1-k)j\pi h)\\ &= c_k^2\varphi_{k,j} - s_k^2\varphi_{\widetilde{k},j}.\end{aligned}$$

Now for j even, we first notice similarly as in (4.118) that

$$\begin{aligned}\sin((m+1-k)j\pi h) &= \sin\Big((m+1)j\pi\frac{1}{m+1} - kj\pi h\Big)\\ &= \sin(j\pi)\cos(kj\pi h) - \cos(j\pi)\sin(kj\pi h)\\ &= -\sin(kj\pi h),\end{aligned}$$

and we therefore obtain

$$\begin{aligned}[P\boldsymbol{\phi}_k]_j &= \phi_{k,\frac{j}{2}} = \sin\big(k\pi\frac{j}{2}H\big) = \sin(k\pi jh)\\ &= (c_k^2 + s_k^2)\sin(k\pi jh)\\ &= c_k^2\sin(k\pi jh) - s_k^2\sin((m+1-k)j\pi k)\\ &= c_k^2\varphi_{k,j} - s_k^2\varphi_{\widetilde{k},j},\end{aligned}$$

which concludes our proof. □

Using Lemma 22 and Lemma 23, we can now precisely describe how the two-grid operator G acts on a low-frequency mode and its complementary high frequency mode:

Lemma 24 (Action of the two-grid operator)**.** *The action of the matrix G of the two-level method on a vector $e_k\boldsymbol{\varphi}_k + e_{\widetilde{k}}\boldsymbol{\varphi}_{\widetilde{k}}$ is given by*

$$G\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}e_k\\ e_{\widetilde{k}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}G_k\begin{bmatrix}e_k\\ e_{\widetilde{k}}\end{bmatrix}, \tag{4.119}$$

where

$$G_k := \begin{bmatrix}1-2\omega s_k^2 & 0\\ 0 & 1-2\omega c_k^2\end{bmatrix}^{\nu_2}\begin{bmatrix}s_k^2 & c_k^2\\ s_k^2 & c_k^2\end{bmatrix}\begin{bmatrix}1-2\omega s_k^2 & 0\\ 0 & 1-2\omega c_k^2\end{bmatrix}^{\nu_1}. \tag{4.120}$$

Proof. The proof is obtained in two steps. First, recalling that $D^{-1}A = -\frac{h^2}{2}A$ and the eigenvalues of A, that are $\lambda_k = -\frac{4}{h^2}s_k^2$, we compute for one Jacobi smoothing step

$$\begin{aligned}\Big(I - \omega D^{-1}A\Big)\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}e_k \\ e_{\tilde{k}}\end{bmatrix} &= \Big(I - \omega D^{-1}A\Big)\Big(e_k\boldsymbol{\varphi}_k + e_{\tilde{k}}\boldsymbol{\varphi}_{\tilde{k}}\Big)\\ &= (1-2\omega s_k^2)e_k\boldsymbol{\varphi}_k + (1-2\omega s_{\tilde{k}}^2)e_{\tilde{k}}\boldsymbol{\varphi}_{\tilde{k}}\\ &= (1-2\omega s_k^2)e_k\boldsymbol{\varphi}_k + (1-2\omega c_k^2)e_{\tilde{k}}\boldsymbol{\varphi}_{\tilde{k}},\end{aligned}$$

where the last equality is obtained because of

$$\begin{aligned}s_{\tilde{k}} &= \sin\Big((m+1-k)\pi\frac{h}{2}\Big)\\ &= \sin\Big((m+1)\pi\frac{1}{2(m+1)}\Big)\cos\Big(k\pi\frac{h}{2}\Big) - \cos\Big((m+1)\pi\frac{1}{2(m+1)}\Big)\sin\Big(k\pi\frac{h}{2}\Big)\\ &= \cos\Big(k\pi\frac{h}{2}\Big) = c_k,\end{aligned}$$

since $\sin(a-b) = \sin a\cos b - \cos a\sin b$. We thus have for one Jacobi smoothing step

$$\Big(I - \omega D^{-1}A\Big)\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}e_k \\ e_{\tilde{k}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}1-2\omega s_k^2 & 0\\ 0 & 1-2\omega c_k^2\end{bmatrix}\begin{bmatrix}e_k \\ e_{\tilde{k}}\end{bmatrix}. \tag{4.121}$$

Next, we notice that for the coarse eigenfunction $\boldsymbol{\phi}_k$ we have

$$\begin{aligned}A_H\boldsymbol{\phi}_k &= -\frac{4}{H^2}\sin\Big(k\pi\frac{H}{2}\Big)\boldsymbol{\phi}_k = -\frac{4}{(2h)^2}\sin\Big(k\pi h\Big)\boldsymbol{\phi}_k\\ &= -\frac{2}{h^2}\sin\Big(k\pi\frac{h}{2}\Big)\cos\Big(k\pi\frac{h}{2}\Big)\boldsymbol{\phi}_k = -\frac{2}{h^2}s_kc_k\boldsymbol{\phi}_k,\end{aligned} \tag{4.122}$$

where we used again that $\sin(a+b) = \sin a\cos b + \sin b\cos a$ to obtain that $\sin(k\pi h) = \sin(k\pi h/2 + k\pi h/2) = 2\cos(k\pi h/2)\sin(k\pi h/2)$. Now, we denote by $\lambda_{H,k}$ the eigenvalues of A_H and compute using Lemma 22 and Lemma 23

$$\begin{aligned}PA_H^{-1}RA\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix} &= PA_H^{-1}R\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}\lambda_k & 0\\ 0 & \lambda_{\tilde{k}}\end{bmatrix}\\ &= PA_H^{-1}\boldsymbol{\phi}_k\begin{bmatrix}c_k^2 & -s_k^2\end{bmatrix}\begin{bmatrix}\lambda_k & 0\\ 0 & \lambda_{\tilde{k}}\end{bmatrix}\\ &= P\boldsymbol{\phi}_k\frac{1}{\lambda_{H,k}}\begin{bmatrix}c_k^2 & -s_k^2\end{bmatrix}\begin{bmatrix}\lambda_k & 0\\ 0 & \lambda_{\tilde{k}}\end{bmatrix}\\ &= P\boldsymbol{\phi}_k\frac{1}{\lambda_{H,k}}\begin{bmatrix}c_k^2\lambda_k & -s_k^2\lambda_{\tilde{k}}\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}c_k^2\\ -s_k^2\end{bmatrix}\begin{bmatrix}c_k^2\frac{\lambda_k}{\lambda_{H,k}} & -s_k^2\frac{\lambda_{\tilde{k}}}{\lambda_{H,k}}\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\tilde{k}}\end{bmatrix}\begin{bmatrix}c_k^4\frac{\lambda_k}{\lambda_{H,k}} & -c_k^2s_k^2\frac{\lambda_{\tilde{k}}}{\lambda_{H,k}}\\ -c_k^2s_k^2\frac{\lambda_k}{\lambda_{H,k}} & s_k^4\frac{\lambda_{\tilde{k}}}{\lambda_{H,k}}\end{bmatrix}.\end{aligned}$$

We thus obtain for the coarse correction step

$$\left(I - PA_H^{-1}RA\right)\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}1 - c_k^4\frac{\lambda_k}{\lambda_{H,k}} & c_k^2 s_k^2\frac{\lambda_{\widetilde{k}}}{\lambda_{H,k}} \\ c_k^2 s_k^2\frac{\lambda_k}{\lambda_{H,k}} & 1 - s_k^4\frac{\lambda_{\widetilde{k}}}{\lambda_{H,k}}\end{bmatrix}. \quad (4.123)$$

Using (4.122) and that $H = 2h$, we can simplify the ratios of the fine and coarse eigenvalues appearing in the matrix,

$$\begin{aligned}\frac{\lambda_k}{\lambda_{H,k}} &= \frac{-\frac{4}{h^2}\sin^2(k\pi h/2)}{-\frac{4}{H^2}\sin^2(k\pi H/2)} = \frac{-\frac{1}{h^2}\sin^2(k\pi h/2)}{-\frac{1}{(2h)^2}\sin^2(k\pi H/2)} = \frac{4s_k^2}{(2s_kc_k)^2} = \frac{1}{c_k^2}, \\ \frac{\lambda_{\widetilde{k}}}{\lambda_{H,k}} &= \frac{-\frac{4}{h^2}\cos^2(k\pi h/2)}{-\frac{4}{H^2}\sin^2(k\pi H/2)} = \frac{4c_k^2}{(2s_kc_k)^2} = \frac{1}{s_k^2}.\end{aligned} \quad (4.124)$$

Combining (4.123) and (4.124), we obtain for the coarse correction the simple action

$$\left(I - PA_H^{-1}RA\right)\begin{bmatrix}\boldsymbol{\phi}_k & \boldsymbol{\phi}_{\widetilde{k}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\phi}_k & \boldsymbol{\phi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}1 - c_k^2 & c_k^2 \\ s_k^2 & 1 - s_k^2\end{bmatrix} = \begin{bmatrix}\boldsymbol{\phi}_k & \boldsymbol{\phi}_{\widetilde{k}}\end{bmatrix}\begin{bmatrix}s_k^2 & c_k^2 \\ s_k^2 & c_k^2\end{bmatrix}. \quad (4.125)$$

The result in the Lemma then follows from (4.125), (4.121) and recalling the structure of G given in (4.113). □

Lemma 24 implies that the subspace spanned by a low-frequency mode and its high-frequency companion, $\operatorname{span}\{\boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{\widetilde{k}}\}$, is an invariant subspace of the two-grid iteration matrix G. This implies, as the next lemma shows, that G is similar to a block-diagonal matrix.

Lemma 25 (Properties of the two-grid operator)**.** *The two-grid matrix G is similar to the block-diagonal matrix*

$$\widetilde{G} = \begin{bmatrix}G_1 & & & \\ & \ddots & & \\ & & G_{\frac{m+1}{2}-1} & \\ & & & g\end{bmatrix}, \quad (4.126)$$

where the matrices G_k are defined in (4.120) *and $g = (1-\omega)^{\nu_1+\nu_2}$.*

Proof. Consider the eigenvectors $\boldsymbol{\varphi}_k$ of A and define the matrix

$$Q = \begin{bmatrix}\boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_m & \boldsymbol{\varphi}_2 & \boldsymbol{\varphi}_{m-1} & \cdots & \boldsymbol{\varphi}_{\frac{m+1}{2}-1} & \boldsymbol{\varphi}_{\frac{m+1}{2}+1} & \boldsymbol{\varphi}_{\frac{m+1}{2}}\end{bmatrix}.$$

Now, for $k = 1, \ldots, \frac{m+1}{2} - 1$ with $\widetilde{k} = m + 1 - k$ Lemma 24 implies that

$$G\begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_k & \boldsymbol{\varphi}_{\widetilde{k}}\end{bmatrix} G_k.$$

Moreover, according to Lemma 22 it holds that $R\boldsymbol{\varphi}_{\frac{m+1}{2}} = 0$, which allows us to compute that

$$G\boldsymbol{\varphi}_{\frac{m+1}{2}} = \left(1 - 2\omega s_{\frac{m+1}{2}}^2\right)^{\nu_1+\nu-2} = (1-\omega)^{\nu_1+\nu-2},$$

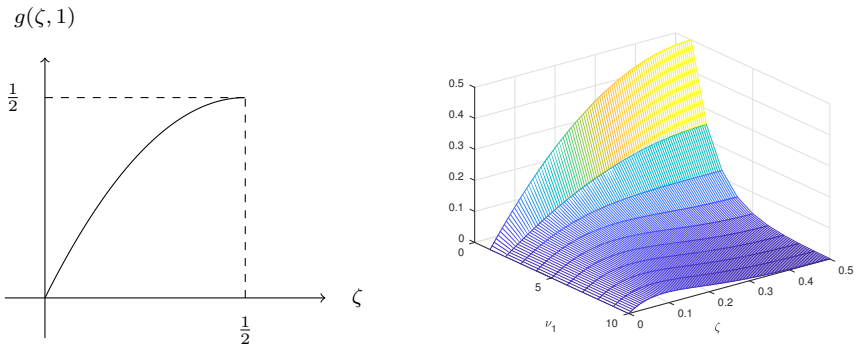


Figure 4.43: Left: Upper bound of the convergence factor $\rho(G_k)$ as a function of ζ for $\nu_1 = 1$. Right: Spectral radius $\rho(G_k)$ as a function of ζ and ν_1.

where that $s^2_{\frac{m+1}{2}} = \frac{1}{2}$ is used. Therefore, we have obtained that $GQ = Q\widetilde{G}$ and the claim follows. □

Lemma 25 implies that the convergence factor of the two-grid method is the maximum over all k of the spectral radius of the 2×2 matrix in (4.119). If we consider for simplicity $\nu_2 = 0$, which means no post-smoothing, and $\omega = \frac{1}{2}$, this 2×2 matrix becomes

$$G_k = \begin{bmatrix} s_k^2 & c_k^2 \\ s_k^2 & c_k^2 \end{bmatrix} \begin{bmatrix} 1 - s_k^2 & 0 \\ 0 & 1 - c_k^2 \end{bmatrix}^{\nu_1} = \begin{bmatrix} s_k^2 & c_k^2 \\ s_k^2 & c_k^2 \end{bmatrix} \begin{bmatrix} c_k^2 & 0 \\ 0 & s_k^2 \end{bmatrix}^{\nu_1} = \begin{bmatrix} s_k^2 c_k^{2\nu_1} & c_k^2 s_k^{2\nu_1} \\ s_k^2 c_k^{2\nu_1} & c_k^2 s_k^{2\nu_1} \end{bmatrix}.$$

Notice that G_k is of the form $\begin{bmatrix} \alpha & \beta \\ \alpha & \beta \end{bmatrix}$, with $\alpha = s_k^2 c_k^{2\nu_1}$ and $\beta = c_k^2 s_k^{2\nu_1}$. Therefore, it is possible to compute its eigenvalues and obtain the spectral radius explicitly,

$$\rho(G_k) = s_k^2(1 - s_k^2)^{\nu_1} + (1 - s_k^2)s_k^{2\nu_1}. \tag{4.127}$$

The convergence factor of the multigrid method is then given by $\max_k \rho(G_k)$. The maximum has to be taken over the discrete values s_k, $k = 1, \ldots, \frac{m+1}{2} - 1$, and s_k^2 varies in the interval $[0, \frac{1}{2}]$ with the value $\frac{1}{2}$ attained at $k = \frac{m+1}{2}$. Hence one can bound the convergence factor by the maximum of the function $g(\zeta, \nu_1) := \zeta^2(1 - \zeta^2)^{\nu_1} + (1 - \zeta^2)\zeta^{2\nu_1}$ for $\zeta \in [0, \frac{1}{2}]$. This bound is shown in Figure 4.43 (left) as a function of ζ (and ν_1 fixed), and in Figure 4.43 (right) as a function of ζ and ν_1. These two figures show clearly that $\rho(G_k) \leq \max_{\zeta \in [0, \frac{1}{2}]} g(\zeta, \nu_1) < 1$.

We thus obtain for the two-grid algorithm the following convergence result:

Theorem 41 (Convergence of the two-grid method)**.** *The two-grid method without post-smoothing and relaxation parameter $\omega = \frac{1}{2}$ described by the iteration matrix G given in* (4.113) *converges independently of h, and we have the estimate*

$$\rho(G) \leq \max_{0 \leq \zeta \leq \frac{1}{2}} \left(\zeta(1 - \zeta)^{\nu_1} + (1 - \zeta)\zeta^{\nu_1} \right) < 1.$$

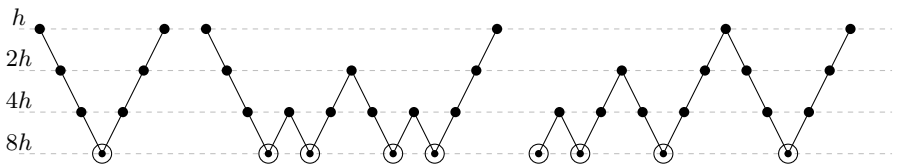


Figure 4.44: V-Cycle (left), W-Cycle (middle) and FMG (right). The symbol • represents a smoothing step, while ⊙ represents a direct solution step.

The theoretical result obtained in Theorem 41 is illustrated numerically in Figure 4.46 (left) below, where the decay of the norm of the residual is compared to the theoretical bound of Theorem 41 (see Problem 49 to produce this plot). This figure shows that the theoretical bound obtained is quite sharp, since the two curves are essentially parallel.

Now with the two-grid method, we did in principle not gain anything yet, since we still have to solve a problem of the same nature, just half the size, in the coarse-correction step. The key idea of multigrid is that in the two grid operator in (4.113), it is possible to apply the two-level method recursively instead of computing A_H^{-1}. One then does this as long as the coarse problem is still too big to be solved directly, which leads to the so-called multigrid V-cycle, see Figure 4.44 (left).

We now give an implementation of the multigrid V-cycle for our two dimensional Laplace model problem in MATLAB:

```
function u=VCycle(A,f,u,l)
% VCYCLE performs a V-cycle of multigrid for the Laplace equation
%   u=VCycle(A,f,u,l) computes for the given linear system Au=f
%   representing a discretized Laplacian on a square domain using
%   the five point finite difference stencil a V-cycle approximation
%   of the solution starting with the guess u using l levels.
%   The resolution size m=sqrt(length(f)) must be such that m+1 can
%   be divided by 2^l. The method uses damped Jacobi, and the number
%   of pre- and post-smoothing steps nu1 and nu2 as well as the damping
%   parameter w can be set in the code.

nu1=1; nu2=1;                          % pre- and post-smoothing steps
w=2/3;                                 % damping parameter for Jacobi
if l==1                                % direct solve on coarsest level
  u=A\f;
else
  for i=1:nu1                          % pre-smoothing with relaxed Jacobi
    u=u+w*(f-A*u)./diag(A);
  end
  r=f-A*u;                             % compute residual
  m=sqrt(length(f));
  R=RMatrix(m);                        % compute restriction matrix R
  M=(m+1)/2-1;
```

```
  H=1/(M+1);
  AH=Laplacian(M,2)/H^2;             % compute coarse matrix
  % AH=4*R*A*R';                     % can use Galerkin instead
  e=VCycle(AH,R*r,zeros(M^2,1),l-1); % compute coarse correction
  u=u+4*R'*e;                        % use full weighting
  for i=1:nu2                        % post-smoothing with relaxed Jacobi
    u=u+w*(f-A*u)./diag(A);
  end
end
```

We see that to get the V-Cycle, one simply calls the program itself when the coarse correction needs to be computed, and the two grid method is obtained by choosing for the level parameter $l = 2$. The program uses our earlier function `Laplacian.m` to compute the coarser matrices needed in the V-Cycle. If the V-Cycle is used several times, as it is usually the case, it is more economic to compute these operators first at the beginning all only once, and to store them for later use. We also left a commented line which gives a different option for the coarse matrix, the so-called Galerkin approach which computes the coarse matrix using the restriction matrix R, which is computed in the V-Cycle routine using the function `RMatrix`:

```
function R=RMatrix(m)
% RMATRIX construct restriction matrix of size m^2/4 x m^2
%   R=RMatrix(m) constructs a restriction matrix of size m^2/4 x m^2
%   to be used in a multigrid code.

e=ones(m^2,1);                      % diagonal entries
e1=repmat([ones(m-1,1);0],m,1);
e2=ones(m^2,1);
B=zeros(m^2,3);
B(1:length(e1),1)=e1/16;
B(1:length(e2),2)=e2/8;
B(1:length(e1),3)=flipud(e1/16);
B(1:length(e1),4)=e1/8;
B(1:length(e),5)=e/4;
B(1:length(e1),6)=flipud(e1/8);     % flipud for spdiags convention
B(1:length(e1),7)=e1/16;
B(1:length(e2),8)=flipud(e2/8);
B(1:length(e1),9)=flipud(e1/16);

R=spdiags(B,[-m-1 -m -m+1 -1 0 1 m-1 m m+1],m^2,m^2);
G=numgrid('S',m+2);                 % extract even gridpoints
id=G(3:2:m,3:2:m);
R=R(id(:),:);
```

This is just the two-dimensional analog of the linear interpolation we have seen. We can now test the V-Cycle multigrid method on our model problem:

```
m=15;                             % resolution must be a power of 2 minus 1
```

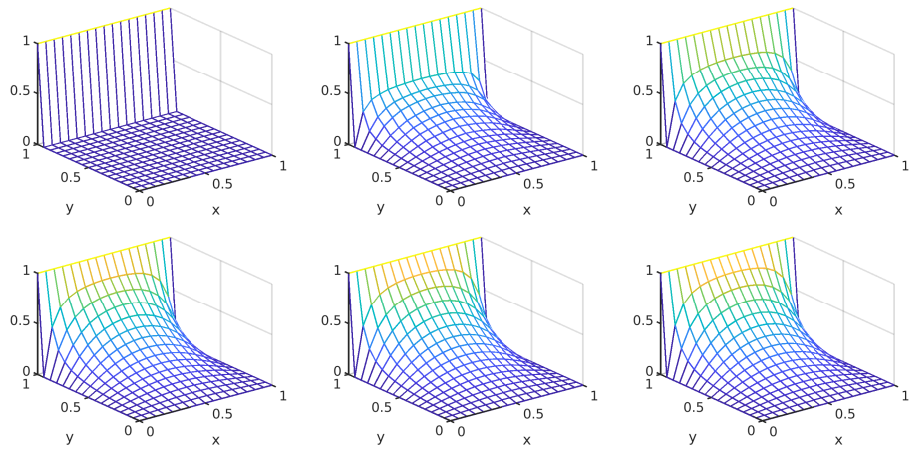


Figure 4.45: Initial guess and first iterations of the multigrid V-cycle applied to our Laplace model problem.

```
l=5;                                % levels, l not bigger than L in m=2^L-1
h=1/(m+1);
A=Laplacian(m,2)/h^2;               % construct the Laplacian on the fine grid
f=zeros(m*m,1);                     % right hand side for our model problem
f(m:m:end)=-1/h^2;
u=A\f;                              % exact solution
umg=zeros(size(f));                 % multigrid initial guess
U=zeros(m+2); U(end,1:m+2)=1;       % for plotting purposes
x=0:1/(m+1):1; y=x;                 % mesh point vectors
for n=0:10
  err(n+1)=max(max(abs(u-umg)));    % error in the infinity norm
  U(2:m+1,2:m+1)=reshape(umg,m,m);
  mesh(x,y,U)                       % plot current approximation
  xlabel('x');ylabel('y');
  pause                             % perform a V-cycle
  umg=umg+VCycle(A,f-A*umg,zeros(size(f)),l);
end
```

This leads to the multigrid iterates shown in Figure 4.45. We used again $m = 15$ interior mesh points, and only one pre- and post-smoothing step, $\nu_1 = \nu_2 = 1$, and for the damping parameter in Jacobi $\omega = \frac{1}{2}$. We also used the maximum number of levels possible, $l = 5$, i.e. the coarsest problem we solve by the direct solver backslash in MATLAB is only of size 1×1. One can see how effective the V-cycle is: we arrive sufficiently close to the solution after only five iterations.

To illustrate that the convergence of the multigrid method is also independent of the mesh parameter h, we show in Figure 4.46 how the error decays in the multigrid method for the mesh we used for Figure 4.45 with $m = 15$, and also on two refined meshes with $m = 31$ and $m = 63$ interior mesh points. We clearly see that convergence is independent of the mesh parameter h, and also

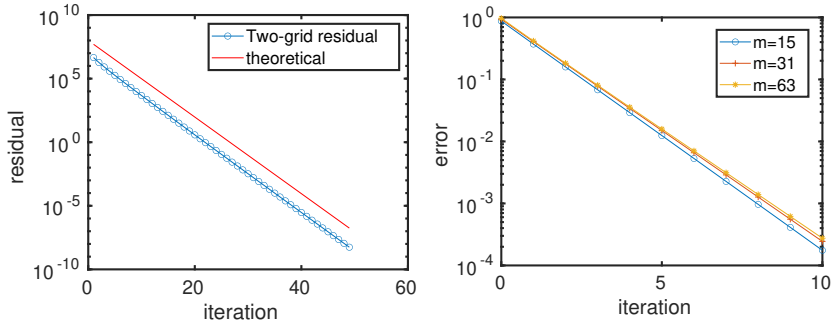


Figure 4.46: Left: decay of the residual (blue curve) of a two-grid iteration (with $\nu_1 = 1$ and $\nu_2 = 0$ for the solution of a one-dimensional Laplace problem and theoretical bound (red curve) corresponding to the developed convergence analysis. Right: Convergence of the multigrid method for different mesh sizes $h = \frac{1}{m+1}$.

quite well predicted by the analysis in Theorem 41, even though we did not include the post-smoothing step in the analysis.

The performance of the multigrid algorithm is impressive, since it basically gives mesh independent convergence at a cost per iteration which is bounded by the cost of $1.65(\nu_1 + \nu_2)$ Jacobi iterations on the fine mesh: the main cost in the algorithm are the Jacobi iterations on the finest mesh, where there are $\nu_1 + \nu_2$ of them, and then on the next coarser mesh the matrix is only a quarter of the size, so the Jacobi iterations on that mesh cost only a quarter of $\nu_1 + \nu_2$ on the finest mesh, and on the next coarser one only a sixteenth, and so on, which leads by summing $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{1}{6}\pi^2 \approx 1.645$ to the estimate given by the factor 1.65.

The V-Cycle is not the only multilevel extension of the two-grid method. Other two common examples are the W-Cycle and the Full multigrid (FMG), whose schematic representations are given in Figure 4.44 (middle and right). The main idea of the W-Cycle is to perform more iterations on the coarse levels which are computationally cheaper and more efficient against low-frequency components of the error. The multigrid setting suggests a natural way of how to get a good initial guess cheaply: one can start the multigrid iterations at the coarser level and move iteratively to the upper levels. This idea leads to the FMG depicted in Figure 4.44 (right).

4.11 Problems

Problem 36. *Implement the preconditioned CG method given in Theorem 33.*

Problem 37. *Consider the two-dimensional discrete Laplace problem (characterized by a matrix A and mesh size h). Use the* MATLAB *function* `ilu` *to obtain*

the incomplete factorization $A = \widetilde{L}\widetilde{R} - R$ corresponding to a certain tolerance ϵ (see Section 4.5). Write a MATLAB *implementation for the stationary method corresponding to the splitting $A = \widetilde{L}\widetilde{R} - R$ and solve the problem for different mesh sizes h. Repeat this experiment changing the ILU parameter ϵ and try to determine how much fill in of the matrices $\widetilde{L}$ and $\widetilde{R}$ is needed in order to obtain a mesh independent stationary method.*

Problem 38. *Consider the two-dimensional discrete Laplace problem (characterized by a matrix A and mesh size h). Modify the* `ILU0` MATLAB *algorithm given in Section 4.5 in order to obtain an incomplete factorization using the sparsity pattern of A^k, for $k \in \mathbb{N}$.*

Problem 39. *Consider the one-dimensional Laplace problem*

$$-\frac{d^2u(x)}{dx^2} = f(x) \text{ in } \Omega = (0,1) \text{ with } u(0) = u(1) = 0.$$

Discretize this problem with finite differences on a uniform mesh of size h to obtain $A\boldsymbol{u} = \boldsymbol{f}$, where A is the discrete Laplacian.

Now, consider a domain decomposition $\Omega = \Omega_1 \cup \Omega_2$ with $\Omega_1 = (0, \alpha)$ and $\Omega_2 = (\beta, 1)$ with $\beta < \alpha$ and $\delta = \alpha - \beta$ being the overlap. The alternating and the parallel Schwarz methods are given by

$$-\frac{d^2u_1^{n+1}(x)}{dx^2} = f(x) \text{ in } (0,\alpha) \text{ with } u_1^{n+1}(0) = 0 \text{ and } u_1^{n+1}(\alpha) = u_2^n(\alpha)$$
$$-\frac{d^2u_2^{n+1}(x)}{dx^2} = f(x) \text{ in } (\beta,1) \text{ with } u_2^{n+1}(\beta) = u_1^n(\beta) \text{ and } u_2^{n+1}(1) = 0$$

and

$$-\frac{d^2u_1^{n+1}(x)}{dx^2} = f(x) \text{ in } (0,\alpha) \text{ with } u_1^{n+1}(0) = 0 \text{ and } u_1^{n+1}(\alpha) = u_2^n(\alpha)$$
$$-\frac{d^2u_2^{n+1}(x)}{dx^2} = f(x) \text{ in } (\beta,1) \text{ with } u_2^{n+1}(\beta) = u_1^{n+1}(\beta) \text{ and } u_2^{n+1}(1) = 0,$$

respectively. Discretize these subproblems by means of finite differences on uniform meshes of size h. Show that, at a discrete level, the alternating and parallel Schwarz methods with minimal overlap, that is $\delta = h$, are equivalent to block-Gauss-Seidel and block-Jacobi methods for the solution to $A\boldsymbol{u} = \boldsymbol{f}$. How fast do these methods converge? (Recall Problem 18 in Chapter 2).

Problem 40. *Consider the one-dimensional Laplace problem and the two-subdomain decomposition as in Problem 39. Prove that (at a continuous level) the alternating and parallel Schwarz methods converge and study the dependence of the corresponding convergence factors on the overlap $\delta = \alpha - \beta$.*

Problem 41. *Consider for simplicity the one-dimensional Laplace problem and the two-subdomain decomposition as in Problem 39. Show that the discrete parallel Schwarz method, the additive Schwarz method and the restricted additive Schwarz (RAS) method are equivalent if $\delta = h$ (minimal overlap) holds.*

Problem 42. *Implement the discrete alternating and parallel Schwarz method for the Laplace problem on the unit square with two rectangular subdomains. Experiment with different mesh sizes, overlap parameters and boundary conditions, and describe your findings.*

Problem 43. *Implement the additive Schwarz method and the restricted additive Schwarz method for the Poisson problem on the unit square with two rectangular subdomains. Experiment with different mesh sizes, overlap parameters and boundary conditions, and describe your findings.*

Problem 44. *Investigate in detail the value of the first 3 iterates of the optimized Schwarz method* (4.52)*, when using the optimal choice of the parameters* p_1 *and* p_2 *in* (4.58)*. Can you also get the result in a finite number of iterations with a different choice? Investigate the same also for the parallel optimized Schwarz variant introduced by Lions.*

Problem 45. *Implement the alternating optimized Schwarz method for the Poisson problem on the unit square with two rectangular subdomains using two different approaches:*

- *By discretizing the continuous formulation and only computing subdomain solutions. Hint: make use of the Robin solver from the Dirichlet-Neumann section 4.7.*
- *By using restriction operators* R_j *in the form of the optimized multiplicative and restricted additive Schwarz method.*

For the second part, you need to show what changes in the subdomain matrices are necessary to obtain an equivalent program to the first one, similar to the proof of Theorem 37. What is the smallest overlap needed so that the second implementation still faithfully represents the discretization of an optimized Schwarz method?

Problem 46. *Implement the parallel optimized Schwarz method for the Poisson problem on the unit square with two rectangular subdomains using two different approaches:*

- *By discretizing the continuous formulation and only computing subdomain solutions.*
- *By using restriction operators* R_j *and* $\widetilde{R}_j$ *in the form of an optimized restricted additive Schwarz method.*

For the second part, you need to show what changes in the subdomain matrices are necessary to obtain an equivalent program to the first one. Is there a restriction on the overlap size? Could this also be done with additive Schwarz?

Problem 47. *Implement RAS for a square decomposed into an arbitrary number* $J \times J$ *of equally sized small square domains. Test the convergence when* J *is increasing. What do you observe? What happens if your domain is a chain of squares, and you add more and more squares to make the domain longer?*

Problem 48. *Show that the damped Jacobi method* (4.109) *corresponds to the splitting* $A = M - N$ *with* $M = \frac{1}{\omega}D$ *and* $N = \frac{1}{\omega}[(1-\omega)D - \omega(L+U)]$.

Problem 49. *Implement a two-grid method (with a damped-Jacobi smoother) for the solution of a discrete one-dimensional Laplace problem. Reproduce the results shown in Figure 4.46 (left).*

Problem 50. *Consider the multigrid method for the solution of the two-dimensional Laplacian. Change the codes provided in Section 4.10 in order to replace the Jacobi smoother with other smoothing operators (e.g., Gauß-Seidel, SOR, RAS, etc.). Compare and discuss the observed convergence.*

Problem 51. *Consider the discrete Laplace problem. Implement and test the preconditioners considered in this chapter. Compare the difference preconditioned GMRES behaviors for different values of mesh size* h.

Chapter 5

Appendix

5.1 Existence, uniqueness and well-posedness of Schwarz iterates

To study the well-posedness of the Schwarz iterates, we first recall some existence and regularity results for the Laplace problem (4.29).

Theorem 42 (Existence, uniqueness, and regularity of the Laplace solution)**.** *Consider a bounded and convex domain $\Omega \subset \mathbb{R}^2$. We have:*

- *For any $f \in L^2(\Omega)$ and $g \in H^{1/2}(\partial\Omega)$ there exists a unique (weak) solution $u \in H^1(\Omega)$ to (4.29).*
- *For any function f bounded and locally Hölder continuous in Ω and $g \in C(\partial\Omega)$ there exists a unique (classical) solution $u \in C(\overline{\Omega}) \cap C^2(\Omega)$ to (4.29).*
- *For any function f bounded and locally Hölder continuous in Ω and any bounded boundary function g there exists a unique (Perron) solution $u \in C^2(\Omega)$ to (4.29). In this case, for any point $\boldsymbol{x}_0$ where g is continuous, the solution u satisfies $\lim_{\boldsymbol{x}\to\boldsymbol{x}_0} u(\boldsymbol{x}) = g(\boldsymbol{x}_0)$.*

Proof. The first statement follows by standard results based, e.g., on the Lax-Milgram theorem [28, Theorem 6.2-1] and on the trace theorem for functions in $H^1(\Omega)$ [80, Corollary 8.16, Theorem 8.17, and Theorem 8.27]; see also [37, Chapter 6]. The second statement follows by classical results based on the Green's function representation [59, Theorem 4.3]; see also [28, pages 342-343]. The last statement follows from the theory of Perron solutions; see [59, Section 2.8]. □

Let us now discuss well-posedness of the Schwarz method (4.30). Consider the Laplace problem (1.5) with a boundary function $g \in H^{1/2}(\partial\Omega)$ and right-hand side function $f \in L^2(\Omega)$. According to Theorem 42, it is uniquely solvable by $u \in H^1(\Omega)$. We have the following result.

Theorem 43 (Well-posedness of the Schwarz method - 1)**.** *Let $\Omega \subset \mathbb{R}^n$ be a bounded Lipschitz domain, and consider an overlapping domain decomposition $\Omega = \Omega_1 \cup \Omega_2$, where Ω_1 and Ω_2 are both Lipschitz domains. Assume that $f \in L^2(\Omega)$ and $g \in H^{1/2}(\partial\Omega)$, and consider the trace operator $\tau : H^1(\Omega) \to H^{1/2}(\partial\Omega)$. Let $u^0 \in H^1(\Omega)$ be an initial guess such that $\tau(u^0) = g$. The Schwarz sequences $\{u_1^n\}_n$ and $\{u_2^n\}_n$ are well-defined in the sense that for $n \geq 1$ the Schwarz subproblems* (4.30) *are uniquely solvable by weak solutions $u_1^n \in H^1(\Omega_1)$ and $u_2^n \in H^1(\Omega_2)$.*

Proof. Consider the trace operators $\tau_j : H^1(\Omega) \to H^{1/2}(\partial\Omega_j)$, for $j = 1, 2$. Since $u^0 \in H^1(\Omega)$, we have that $\tau_1(u^0) \in H^{1/2}(\partial\Omega_1)$. Hence, for $n = 1$, the first-subdomain problem is

$$\begin{aligned} \Delta u_1^1 &= f && \text{in } \Omega_1, \\ u_1^1 &= \tau_1(u^0) && \text{on } \partial\Omega_1, \end{aligned}$$

which is uniquely solved by $u_1^1 \in H^1(\Omega_1)$. Now, we define a function

$$\widetilde{u} := \begin{cases} u_1^1 & \text{in } \Omega_1, \\ u^0 & \text{in } \Omega \setminus \Omega_1. \end{cases}$$

Notice that a direct calculation shows that $\widetilde{u} \in H^1(\Omega)$ (see [23]) and $\tau(\widetilde{u}) = g$. Therefore, the second-subdomain problem is

$$\begin{aligned} \Delta u_2^1 &= f && \text{in } \Omega_1, \\ u_2^1 &= \tau_2(\widetilde{u}) && \text{on } \partial\Omega_2, \end{aligned}$$

which is uniquely solved by $u_2^1 \in H^1(\Omega_2)$. One can now define a function

$$\widehat{u} := \begin{cases} u_2^1 & \text{in } \Omega_2, \\ u_1^1 & \text{in } \Omega \setminus \Omega_2, \end{cases}$$

rewrite the first-subdomain problem, and repeat recursively the above argument to obtain the result. □

Consider the example of Laplace problem (1.5), with boundary function g as described in Figure 1.10, that we always consider in this book as test problem for numerical experiments. According to Theorem 42, it is uniquely solvable by $u \in C^2(\Omega)$. Moreover, since the boundary function g has two singularities (jumps) in the two corners $(x, y) = (0, 0)$ and $(x, y) = (0, 1)$ of the square domain, we have that $\lim_{\boldsymbol{x} \to \boldsymbol{x}_0} u(\boldsymbol{x}) = g(\boldsymbol{x}_0)$ for every $\boldsymbol{x}_0 \in \partial\Omega \setminus \{(0, 0), (0, 1)\}$. For this reason, g is neither in $C(\partial\Omega)$, nor in $H^{1/2}(\partial\Omega)$. Therefore, u is neither a classical solution nor a weak solution. In this case, the Schwarz method is still well posed, as the following theorem shows.

Theorem 44 (Well-posedness of the Schwarz method - 2)**.** *Assume that the domain $\Omega = (0, 1) \times (0, L)$ decomposed as in Figure 4.12. Assume that f is*

bounded and locally Hölder continuous in Ω *and that* g *is bounded in* $\partial\Omega$ *and continuous at the interface points* $(\beta,0)$, $(\alpha,0)$, $(\beta,1)$, *and* $(\alpha,1)$. *If* $u_2^0 \in C(\overline{\Omega})$, *then the Schwarz sequences* $\{u_1^n\}_n$ *and* $\{u_2^n\}_n$ *are well-defined in the sense that for* $n \geq 1$ *the Schwarz subproblems* (4.30) *are uniquely solvable by Perron solutions* $u_1^n \in C^2(\Omega_1)$ *and* $u_2^n \in C^2(\Omega_2)$. *Moreover, it holds that* $u_1^n|_{\overline{\Gamma}_2} \in C(\overline{\Gamma}_2)$ *and* $u_2^n|_{\overline{\Gamma}_1} \in C(\overline{\Gamma}_1)$ *for* $n \geq 1$.

Proof. Since the initial guess u_2^0 is in $C(\overline{\Omega})$, we can trace it along the interface Γ_1. Since f is bounded and locally Hölder continuous in Ω, g is bounded, and the domain is a convex open set in $\mathbb{R}^2$, Theorem 42 guarantees that the Schwarz subproblem (4.30) (left) is uniquely solvable by a (Perron) solution $u_1^1 \in C^2(\Omega_1)$. Notice that u_2^0 is not necessarily equal to g in $(\alpha,0)$ and $(\alpha,1)$ (jumps can occur), and g is not necessarily continuous on $\partial\Omega$. Hence (even in the case that $g \in C(\partial\Omega)$) the solution u_1^1 is in general neither a classical solution nor a weak solution. However, we can trace its value along the interface Γ_2, and since g is continuous at the points $(\beta,0)$ and $(\beta,1)$, it holds that $u_1^1|_{\overline{\Gamma}_2} \in C(\overline{\Gamma}_2)$. With this trace, one can solve (4.30) (right). Its solution is a Perron function $u_2^1 \in C^2(\Omega_2)$. Again, one can trace u_2^1 along the interface Γ_1 and, since g is continuous in $(\alpha,0)$ and $(\alpha,1)$, it holds that $u_2^1|_{\overline{\Gamma}_1} \in C(\overline{\Gamma}_1)$. This can be used to solve again (4.30) (left) to get $u_1^2 \in C^2(\Omega_1)$, which can be traced on Γ_2 to define a boundary condition for (4.30) (right). By continuing with these arguments we obtain a sequence of Perron solutions $u_1^n \in C^2(\Omega_1)$ and $u_2^n \in C^2(\Omega_2)$, for any $n \geq 1$, such that their traces on the interfaces $\overline{\Gamma}_2$ and $\overline{\Gamma}_1$ are continuous functions. □

5.2 Some polynomial identities

In this section, we prove the polynomial identities used in Chapter 3. Some of their proofs are adapted from [71].

Theorem 45. *For any* $z \in \mathbb{C}$, *the following polynomial identities hold:*

$$1 - z^m = \prod_{k=1}^{m} (1 - w^k z) \tag{5.1}$$

and

$$1 = \sum_{k=1}^{m} \frac{1}{m} \prod_{\ell=1,\ell\neq k}^{m} (1 - w^\ell z), \tag{5.2}$$

where $w = \exp(\frac{2\pi i}{m})$, i *is the imaginary unit, and* m *is an arbitrary positive integer.*

Proof. To prove the identity (5.1) we first notice that

$$\prod_{k=1}^{m} w^k = \prod_{k=1}^{m} \exp\Big(\frac{2\pi i k}{m}\Big) = \exp\Big(\frac{2\pi i \sum_{k=1}^{m} k}{m}\Big)$$
$$= \exp\Big(\frac{2\pi i \frac{m(m+1)}{2}}{m}\Big) = \exp(\pi i(m+1)) = (-1)^{m+1},$$

and that w^{-k}, $k = 1, 2, \dots$ are roots of the polynomial $p(z) = z^m - 1$, which implies that $z^m - 1 = \prod_{k=1}^{m}(z - 1/w^k)$. Therefore, we obtain

$$1 - z^m = -(z^m - 1) = -\prod_{k=1}^{m}(z - \frac{1}{w^k}) = -\prod_{k=1}^{m}\frac{w^k z - 1}{w^k}$$
$$= \frac{(-1)^{m+1}\prod_{k=1}^{m}(1 - w^k z)}{\prod_{k=1}^{m} w^k} = \prod_{k=1}^{m}(1 - w^k z),$$

which is exactly (5.1).

Let us now prove the identity (5.2). To do so, we first notice that $w^m = \exp(\frac{2\pi i m}{m}) = 1$. Therefore, for any non-negative integer j, we have that $w^{\ell+j} = w^{m+\widehat{\ell}} = w^{\widehat{\ell}}$ for $\ell+j > m$ and $\widehat{\ell} = \ell+j-m$. Hence the sets $\{w^\ell, w^{\ell+1}, \dots, w^{\ell+m-1}\}$ are equal for any non-negative integer ℓ. This fact implies that the polynomial $p(z) := \sum_{k=0}^{m-1} \frac{1-z^m}{1-\omega^k z}$ satisfies the relation

$$p(w^\ell z) = \sum_{k=0}^{m-1} \frac{1 - w^{\ell m} z^m}{1 - \omega^{k+\ell} z} = \sum_{k=0}^{m-1} \frac{1 - z^m}{1 - \omega^k z} = p(z), \tag{5.3}$$

which means that $p(z)$ is invariant under any transformation $z \mapsto w^\ell z$, $\ell = 1, \dots, m$. Let us now write $p(z)$ as $p(z) = \gamma_0 + \sum_{j=1}^{m-1} \gamma_j z^j$. Using the relation $p(z) = p(w^\ell z)$ for any ℓ, we can compute

$$0 = p(z) - p(w^\ell z) = \sum_{j=1}^{m-1} \gamma_j (1 - w^{j\ell}) z^j \text{ for all } z \in \mathbb{C} \text{ and for } \ell = 1, \dots, m.$$

This implies that $\gamma_j = 0$, $j = 1, \dots, m-1$, which means that the polynomial p is constant: $p(z) = \gamma_0$. The constant coefficient γ_0 can be computed by evaluating p in $z = 0$: $\gamma_0 = p(0) = m$. We have then obtained $m = p(z) = \sum_{k=0}^{m-1} \frac{1-z^m}{1-\omega^k z}$. The identity (5.2) follows by using (5.1) and dividing by m. □

5.3 Sobolev embedding theorems

In this section, we briefly recall the famous *Sobolev embedding theorems* and *Rellich-Kondrachov compact embedding theorems*. We refer to [28].

Definition 14. *Let X and Y be two normed vector spaces. The space X is said continuously embedded in Y, $X \hookrightarrow Y$, if $X \subset Y$ and there exists a constant c such that $\|x\|_Y \leq c\|x\|_X$ for all $x \in X$. Furthermore, X is said compactly embedded in Y, $X \Subset Y$, if $X \hookrightarrow Y$ and every bounded sequence $(x_k)_{k\in\mathbb{N}}$ in X contains a subsequence converging in Y.*

Theorem 46 (Sobolev embedding theorems)**.** *Let Ω be a Lipschitz domain in $\mathbb{R}^N$, let $m \geq 1$ be an integer, and let $1 \leq p < \infty$. Then the following continuous embeddings hold*

(a)	$W^{m,p}(\Omega) \hookrightarrow L^{p^*}(\Omega)$	with $\frac{1}{p^*} = \frac{1}{p} - \frac{m}{N}$ if $m < \frac{N}{p}$,
(b)	$W^{m,p}(\Omega) \hookrightarrow L^{q}(\Omega)$	for all q with $1 \leq q < \infty$ if $m = \frac{N}{p}$,
(c)	$W^{m,p}(\Omega) \hookrightarrow C^{0,m-N/p}(\bar{\Omega})$	if $\frac{N}{p} < m < \frac{N}{p} + 1$,
(d)	$W^{m,p}(\Omega) \hookrightarrow C^{0,\lambda}(\bar{\Omega})$	for all λ with $0 < \lambda < 1$ if $m = \frac{N}{p} + 1$,
(e)	$W^{m,p}(\Omega) \hookrightarrow C^{0,1}(\bar{\Omega})$	if $m > \frac{N}{p} + 1$.

Theorem 47 (Rellich-Kondrachov compact embedding theorems)**.** *Let $\Omega \subset \mathbb{R}^N$ be a bounded domain that satisfies the cone condition, let $m \geq 1$ be an integer, and let $1 \leq p < \infty$. Then the following compact embeddings hold*

(a)	$W^{m,p}(\Omega) \Subset L^{q}(\Omega)$	for all q with $1 \leq q < p^* = \frac{pN}{N-mp}$ if $m < \frac{N}{p}$,
(b)	$W^{m,p}(\Omega) \Subset L^{q}(\Omega)$	for all q with $1 \leq q < \infty$ if $m = \frac{N}{p}$,
(c)	$W^{m,p}(\Omega) \Subset C(\bar{\Omega})$	if $m > \frac{N}{p}$.

5.4 Lax-Milgram Theorem

Theorem 48 (Lax-Milgram theorem)**.** *Let V be a real Hilbert space endowed with the norm $\|\cdot\|$. Let $a : V \times V \to \mathbb{R}$ be a bilinear form and $\ell : V \to \mathbb{R}$ be a bounded linear functional. If*

(a) *a is coercive, that is, there exists a positive constant c such that for all $v \in V$ it holds $a(v, v) \geq c\|v\|^2$, and*

(b) *a is bounded, that is, there exists a positive constant $\widetilde{c}$ such that for all $v, w \in V$ it holds $|a(v, w)| \leq \widetilde{c}\|v\|\|w\|$,*

then, there exists a unique $y \in V$ such that $a(y, v) = \ell(v)$ for all $v \in V$.

5.5 Weak compactness

Definition 15 (Weakly convergent sequence and weakly compact set)**.** *A sequence $(x_n)_{n\in\mathbb{N}}$ is said to converge weakly to a (weak) limit $x \in X$ (denoted by*

$x_n \rightharpoonup x$) if and only if

$$\lim_{n\to\infty} g(x_n) = g(x) \text{ for all } g \in X^*,$$

where X^* is the dual space of X. A set $C \subset X$ is said to be weakly (sequentially) compact if every sequence $(x_n)_{n\in\mathbb{N}} \subset C$ has a weakly convergent subsequence to some (weak) limit $x \in C$.

Definition 16 (Weakly lower semi-continuous functional)**.** *A functional $f : C \to \mathbb{R}$ is called weakly lower semi-continuous at $x \in C$ if for every sequence $(x_n)_{n\in\mathbb{N}}$ with $x_n \rightharpoonup x$ as $n \to \infty$ it holds that*

$$\liminf_{n\to\infty} f(x_n) \geq f(x).$$

If f is weakly lower semi-continuous at any $x \in C$, we say that f is weakly lower semi-continuous.

Theorem 49 (Weakly compact subsets)**.** *Let X be a reflexive Banach space. Any closed, convex and bounded subset of X is weakly sequentially compact.*

Proof. See, e.g., [28, 99]. □

Theorem 50 (Weakly lower semi-continuous functions)**.** *Let C be a convex subset of a normed vector space X. If $f : C \to \mathbb{R}$ is convex and continuous, then it is weakly lower semi-continuous.*

Proof. See, e.g., [28, 99]. □

Bibliography

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM, 1994.

[3] H. Barucq, M. J. Gander, and Y. Xu. On the influence of curvature on transmission conditions. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 323–331. Springer, 2014.

[4] D. Bennequin, M. J. Gander, L. Gouarin, and L. Halpern. Optimized Schwarz waveform relaxation for advection reaction diffusion equations in two dimensions. *Numerische Mathematik*, 134(3):513–567, 2016.

[5] D. Bennequin, M. J. Gander, and L. Halpern. A homographic best approximation problem with application to optimized Schwarz waveform relaxation. *Math. Comp.*, 78(265):185–223, 2009.

[6] M. Benzi. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.*, 182(2):418–477, 2002.

[7] M. Benzi. Splittings of symmetric matrices and a question of Ortega. *Linear Algebra and its Applications*, 429:2340–2343, 2008.

[8] M. Benzi, C. D. Meyer, and M. Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[9] P. E. Bjørstad and O. B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM Journal on Numerical Analysis*, 23(6):1097–1120, 1986.

[10] M. EL Bouajaji, V. Dolean, M. J. Gander, and S. Lanteri. Optimized Schwarz methods for the time-harmonic Maxwell equations with damping. *SIAM Journal on Scientific Computing*, 34(4):A2048–A2071, 2012.

[11] J.-F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu. Variational formulation and algorithm for trace operator in domain decomposition calculations. In Tony Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Domain Decomposition Methods. Second International Symposium on Domain Decomposition Methods*, pages 3–16, Philadelphia, PA, 1989. SIAM. Los Angeles, California, January 14–16, 1988.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[13] A. Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics 1972*, pages 82–89. Springer, 1973.

[14] N. I. Buleev. Numerical method for solving two- and three-dimensiona diffusion equations. *Mat. Sb. (N.S.)*, 51(93)(2):227–238, 1960.

[15] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM journal on scientific computing*, 21(2):792–797, 1999.

[16] F. Chaouqui, G. Ciaramella, M. J. Gander, and T. Vanzan. On the scalability of classical one-level domain-decomposition methods. *Vietnam Journal of Mathematics*, 2018.

[17] F. Chaouqui, M. J. Gander, and K. Santugini-Repiquet. A local coarse space correction leading to a well-posed continuous Neumann-Neumann method in the presence of cross points. In *Proceedings of the 25th International Conference on Domain Decomposition Methods*. Springer, 2020.

[18] P. L. Chebychev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mémoires des Savants étrangers présentés à l'Académie de Saint-Pétersbourg*, 7:539–586, 1854.

[19] P. L. Chebyshev. Theory of the mechanisms known as parallelograms. *Uspekhi Matematicheskikh Nauk*, 1(2):12–37, 1946.

[20] D. Choi. A proof of crouzeix's conjecture for a class of matrices. *Linear Algebra and its Applications*, 438(8):3247 – 3257, 2013.

[21] G. Ciaramella and M. J. Gander. Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part I. *SIAM J. Numer. Anal.*, 55(3):1330–1356, 2017.

[22] G. Ciaramella and M. J. Gander. Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part II. *SIAM J. Numer. Anal.*, 56 (3):1498–1524, 2018.

[23] G. Ciaramella and M. J. Gander. Analysis of the parallel Schwarz method for growing chains of fixed-sized subdomains: Part III. *Electron. Trans. Numer. Anal.*, 49:201–243, 2018.

[24] G. Ciaramella and M. J. Gander. Happy 25th anniversary ddm!... but how fast can the Schwarz method solve your logo? *Domain Decomposition Methods in Science and Engineering XXV, LNCSE, Springer*, 2020.

[25] G. Ciaramella, M. Hassan, and B. Stamm. On the scalability of the Schwarz method. *The SMAI journal of computational mathematics*, 6, 2019.

[26] G. Ciaramella, M. Hassan, and B. Stamm. On the scalability of the parallel Schwarz method in one-dimension. *Domain Decomposition Methods in Science and Engineering XXV, LNCSE, Springer*, 2020.

[27] G. Ciaramella and R. Höfer. Non-geometric convergence of the classical alternating Schwarz method. *Domain Decomposition Methods in Science and Engineering XXV, LNCSE, Springer*, 2020.

[28] P. G. Ciarlet. *Linear and Nonlinear Functional Analysis with Applications*. Applied mathematics. Society for Industrial and Applied Mathematics, 2013.

[29] M. Crouzeix. Bounds for analytical functions of matrices. *Integral Equations and Operator Theory*, 48:461–477, 04 2004.

[30] M. Crouzeix. Numerical range and functional calculus in hilbert space. *Journal of Functional Analysis*, 244(2):668 – 690, 2007.

[31] M. Crouzeix and A. Greenbaum. Spectral sets: Numerical range and beyond. *SIAM Journal on Matrix Analysis and Applications*, 40(3):1087–1101, 2019.

[32] M. Crouzeix and C. Palencia. The numerical range is a $(1+\sqrt{2})$-spectral set. *SIAM Journal on Matrix Analysis and Applications*, 38(2):649–655, 2017.

[33] B. Delyon and F. Delyon. Generalization of von neumann's spectral sets and integral representation of operators. *Bulletin de la Societe Mathematique de France*, 127, 04 1999.

[34] V. Dolean, M. J. Gander, and L. Gerardo-Giorda. Optimized Schwarz methods for Maxwell's equations. *SIAM Journal on Scientific Computing*, 31(3):2193–2213, 2009.

[35] E. Efstathiou and M. J. Gander. Why Restricted Additive Schwarz converges faster than Additive Schwarz. *BIT Numerical Mathematics*, 43(5):945–959, 2003.

[36] M. Embree. How descriptive are gmres convergence bounds? *Technical report, Oxford University Computing Laboratory*, 1999.

[37] L. C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, Providence (R.I.), 2002.

[38] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6):1205–1227, 1991.

[39] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1(5):922–927, 1961.

[40] G. E. Forsythe. Notes. *Mathematical Tables and Other Aids to Computation*, 5(36):255–258, 1951.

[41] G.E. Forsythe, M.R. Hestenes, and J.B. Rosser. Iterative methods for solving linear equations. In *Bulletin of the American Mathematical Society*, volume 57(6), pages 480–480. AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213, 1951.

[42] R. W. Freund and N. M. Nachtigal. Qmr: a quasi-minimal residual method for non-hermitian linear systems. *Numerische mathematik*, 60(1):315–339, 1991.

[43] M. J. Gander. Optimized Schwarz methods. *SIAM Journal on Numerical Analysis*, 44(2):699–731, 2006.

[44] M. J. Gander. Schwarz methods over the course of time. *Electron. Trans. Numer. Anal*, 31(5):228–255, 2008.

[45] M. J. Gander. On the influence of geometry on optimized Schwarz methods. *SeMA Journal*, 53(1):71–78, 2011.

[46] M. J. Gander and L. Halpern. Absorbing Boundary Conditions for the Wave Equation and Parallel Computing. *Math. Comp.*, 74(249):153–176, 2005.

[47] M. J. Gander and L. Halpern. Optimized Schwarz waveform relaxation methods for advection reaction diffusion problems. *SIAM Journal on Numerical Analysis*, 45(2):666–697, 2007.

[48] M. J. Gander, L. Halpern, and F. Magoules. An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation. *International journal for numerical methods in fluids*, 55(2):163–175, 2007.

[49] M. J. Gander, L. Halpern, and F. Nataf. Optimal Schwarz Waveform Relaxation for the One Dimensional Wave Equation. *SIAM J. Numer. Anal.*, 41(5):1643–1681, 2003.

[50] M. J. Gander and F. Kwok. *Numerical analysis of partial differential equations using Maple and Matlab*. SIAM, 2018.

[51] M. J. Gander, F. Kwok, and B. C. Mandal. Dirichlet-Neumann waveform relaxation methods for parabolic and hyperbolic problems in multiple subdomains. *in revision for BIT*, 2020.

[52] M. J. Gander, F. Magoules, and F. Nataf. Optimized Schwarz methods without overlap for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 24(1):38–60, 2002.

[53] M. J. Gander and X. Tu. On the origins of iterative substructuring methods. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 597–605. Springer, 2014.

[54] M. J Gander and T. Vanzan. Heterogeneous optimized schwarz methods for second order elliptic pdes. *SIAM Journal on Scientific Computing*, 41(4):A2329–A2354, 2019.

[55] M. J. Gander and G. Wanner. The origins of the alternating Schwarz method. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 487–495. Springer, 2014.

[56] M. J. Gander and H. Zhang. A class of iterative solvers for the Helmholtz equation: Factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods. *Siam Review*, 61(1):3–76, 2019.

[57] W. Gander, M. J. Gander, and F. Kwok. *Scientific computing-An introduction using Maple and MATLAB*, volume 11. Springer Science & Business, 2014.

[58] C. F. Gauss. Letter to Gerling, December 26, 1823. In *Werke*, volume 9, pages 278–281. Göttingen, 1903.

[59] D. Gilbarg and N. S. Trudinger. *Elliptic partial differential equations of second order*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, Berlin, New York, 1983.

[60] C. Glader, M. Kurula, and M. Lindström. Crouzeix's conjecture holds for tridiagonal 3 x 3 matrices with elliptic numerical range centered at an eigenvalue. *SIAM Journal on Matrix Analysis and Applications*, 39(1):346–364, 2018.

[61] G. H. Golub. *The use of Chebyshev matrix polynomials in the iterative solution of linear equations compared to the method of successive overrelaxation*. PhD thesis, University of Illinois at Urbana-Champaign, 1959.

[62] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1996.

[63] M. Gonzalez. *Classical Complex Analysis*. Chapman & Hall Pure and Applied Mathematics. Taylor & Francis, 1991.

[64] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997.

[65] A. Greenbaum and M. L. Overton. Numerical investigation of crouzeix's conjecture. *Linear Algebra and its Applications*, 542:225 – 245, 2018. Proceedings of the 20th ILAS Conference, Leuven, Belgium 2016.

[66] A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for gmres. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469, 1996.

[67] A. Greenbaum and Z. Strakoš. Matrices that generate the same krylov residual spaces. In Gene Golub, Mitchell Luskin, and Anne Greenbaum, editors, *Recent Advances in Iterative Methods*, pages 95–118, New York, NY, 1994. Springer New York.

[68] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[69] W. Hackbusch. Ein iteratives Verfahren zur schnellen Auflösung elliptischer randwertprobleme, rep. 76-12. Technical Report 76-12, Institute for Applied Mathematics, University of Cologne, West Germany, Cologne, 1976.

[70] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, 2003.

[71] P.R. Halmos. *A Hilbert Space Problem Book*. Graduate Texts in Mathematics. Springer, 1982.

[72] M. R. Hestenes and E. Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS, 1952.

[73] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[74] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[75] A.S. Householder. On the convergence of matrix iterations. Technical Report 1883, Oak Ridge National Laboratory, 1955.

[76] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition, 2008.

[77] C. G. J. Jacobi. Ueber eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden lineären Gleichungen. *Astronomische Nachrichten*, 22(20):297–306, 1845.

[78] F. John. *Advanced Numerical Methods*. Lecture Notes, Department of Mathematics, 1956.

[79] W. Kahan. *Gauss-Seidel Methods of Solving Large Systems of Linear EQUATIONS*. PhD thesis, University of Toronto, 1958.

[80] R. Kress. *Linear Integral Equations*. Applied Mathematical Sciences. Springer New York, 2013.

[81] E. Kreyszig. *Introductory Functional Analysis With Applications*. Wiley Classics Library. John Wiley & Sons, 1978.

[82] A. N. Krylov. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined. *Izvestija AN SSSR (News of Academy of Sciences of the USSR), Otdel. mat. i estest. nauk*, 7(4):491–539, 1931.

[83] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[84] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE transactions on computer-aided design of integrated circuits and systems*, 1(3):131–145, 1982.

[85] J. Liesen and Z. Strakos. *Krylov subspace methods: principles and analysis*. Oxford University Press, 2013.

[86] E. Lindelöf. Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 116(3):454–457, 1894.

[87] P.-L. Lions. On the Schwarz alternating method. I. In *First international symposium on domain decomposition methods for partial differential equations*, pages 1–42. Paris, France, 1988.

[88] P.-L. Lions. On the Schwarz alternating method. II. *Domain Decomposition Methods*, pages 47–70, 1989.

[89] P.-L. Lions. On the Schwarz alternating method. III: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, PA, 1990.

[90] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Mathematics of Computation*, 31(137):148–162, 1977.

[91] F. Nataf and F. Rogier. Factorization of the convection-diffusion operator and the Schwarz algorithm. *Mathematical Models and Methods in Applied Sciences*, 5(01):67–93, 1995.

[92] R. A. Nicolaides. On multiple grid and related techniques for solving discrete elliptic systems. *Journal of Computational Physics*, 19(4):418–431, 1975.

[93] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[94] P. J. Olver. *Introduction to Partial Differential Equations*. Undergraduate Texts in Mathematics. Springer International Publishing, 2013.

[95] A. M. Ostrowski. On the linear iteration procedures for symmetric matrices. *Rend. Mat. Appl.*, 14:140–163, 1954.

[96] C. C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, University of London, England, 1971.

[97] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[98] C. Pearcy. An elementary proof of the power inequality for the numerical radius. *Michigan Math. J.*, 13(3):289–291, 1966.

[99] P. Philip. *Optimal Control of Partial Differential Equations (Lecture Notes)*. HU Berlin, LMU Munich, 2013.

[100] E. Picard. Sur l'application des méthodes d'approximations successives à l'étude de certaines équations différentielles ordinaires. *Journal de Mathématiques Pures et Appliquées*, 9:217–272, 1893.

[101] J. S. Przemieniecki. Matrix structural analysis of substructures. *AIAA Journal*, 1(1):138–147, 1963.

[102] P. J. Psarrakos and M. J. Tsatsomeros. *Numerical Range: (in) a Matrix Nutshell*. Number v. 1 in Mathematics notes from Washington State University. Department of Mathematics, Washington State University, 2002.

[103] V. Rawat and J.-F. Lee. Nonoverlapping domain decomposition with second order transmission condition for the time-harmonic maxwell's equations. *SIAM Journal on Scientific Computing*, 32(6):3584–3603, 2010.

[104] E. Reich. On the convergence of the classical iterative procedures for symmetric matrices. *Ann. Math. Statist*, 20:448–451, 1949.

[105] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210:307–357, 1911.

[106] J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.

[107] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of computation*, 37(155):105–126, 1981.

[108] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, UK, 1992.

[109] Y. Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[110] Y. Saad. ILUT: a dual threshold incomplete LU factorization. *Numerical linear algebra with applications*, 1(4):387–402, 1994.

[111] Y. Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[112] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[113] H. A. Schwarz. Über einen Grenzübergang durch alternierendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, 15:272–286, May 1870.

[114] L. Seidel. Über ein Verfahren, die Gleichungen, auf welche die Methode der kleinsten Quadrate führt, sowie lineäre Gleichungen überhaupt, durch successive Annäherung aufzulösen. In *Abhandlungen der Mathematisch-Physikalischen Klasse der Königlich Bayerischen Akademie der Wissenschaften, Band 11, III. Abtheilung*, pages 81–108. -, 1874.

[115] A. St-Cyr, M. J. Gander, and S. J. Thomas. Optimized multiplicative, additive, and restricted additive Schwarz preconditioning. *SIAM Journal on Scientific Computing*, 29(6):2402–2425, 2007.

[116] E.M. Stein and R. Shakarchi. *Complex Analysis*. Princeton lectures in analysis. Princeton University Press, 2010.

[117] E. Stiefel. Über einige Methoden der Relaxationsrechnung. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 3(1):1–33, 1952.

[118] E. C. Titchmarsh. *The Theory of Functions (Second Edition)*. Oxford science publications. Oxford University Press, 1939.

[119] A. Toselli and O. Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer, 2006.

[120] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[121] N. Trefethen. Approximation theory and numerical linear algebra. *In J. C. Mason and M. G. Cox, eds., Algorithms for Approximation II*, 1990.

[122] H. A. Van der Vorst. Bi-cgstab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.

[123] R. S. Varga. Factorization and normalized iterative methods. *Boundary Problems in Differential Equations*, 1960.

[124] Richard S. Varga. *Matrix Iterative Analysis*. Prentice Hall, first edition, 1962.

[125] S. Warner. *Modern Algebra*. Dover Books on Mathematics. Dover Publications, 1990.

[126] O. Widlund and M. Dryja. An additive variant of the Schwarz alternating method for the case of many subregions. Technical report, Department of Computer Science, Courant Institute, 1987.

[127] S. Yang and M. K. Gobbert. The optimal relaxation parameter for the SOR method applied to the Poisson equation in any space dimensions. *Appl. Math. Lett.*, 22:325–331, 03 2009.

[128] D. M. Young. *Iterative Methods for Solving Partial Difference EQUATIONS of Elliptic Type*. PhD thesis, Harvard University, May 1950.

Index