1 Introduction

Readership

The purpose of these notes is to present, at graduate level, an introduction to the application of multigrid methods to elliptic and hyperbolic partial differential equations for engineers, physicists and applied mathematicians. The reader is assumed to be familiar with the basics of the analysis of partial differential equations and of numerical mathematics, but the use of more advanced mathematical tools, such as functional analysis, is avoided. The course is intended to be accessible to a wide audience of users of computational methods. We do not, therefore, delve deeply into the mathematical foundations. This is done in the excellent monograph by Hackbusch [57], which treats many aspects of multigrid, and also contains many practical details. The book [141] is more accessible to non-mathematicians, and pays more attention to applications, especially in computational fluid dynamics.

Other introductory material can be found in the article Brandt [20], the first three chapters of [85] and the short elementary introduction [27]. The notes are based on parts of [141], where further details may be found, and other subjects are discussed, notably applications in computational fluid dynamics.

Significance of multigrid methods for scientific computation

Needless to say, elliptic and hyperbolic partial differential equations are, by and large, at the heart of most mathematical models used in engineering and physics, giving rise to extensive computations. Often the problems that one would like to solve exceed the capacity of even the most powerful computers, or the time required is too great to allow inclusion of advanced mathematical models in the design process of technical apparatus, from microchips to aircraft, making design optimization more difficult. Multigrid methods are a prime source of important advances in algorithmic efficiency, finding a rapidly increasing number of users. Unlike other known methods, multigrid offers the possibility of solving problems with N unknowns with $O(N)$ work and storage, not just for special cases, but for large classes of problems.

Historical development of multigrid methods

Table 1.0.1, based on the multigrid bibliography in [85], illustrates the rapid growth of the multigrid literature, a growth which has continued unabated since 1985.

As shown by Table 1.0.1, multigrid methods have been developed only recently. In what probably was the first 'true' multigrid publication, Fedorenko [43] formulated a multigrid algorithm for the standard five-point finite difference discretization of the Poisson equation on a square, proving that the work required to reach a given precison is $O(N)$. This work was generalized to the central difference discretization of the general linear elliptic partial differential equation (3.2.1) in $\Omega = (0,1)\times(0,1)$ with variable smooth coefficients by Bachvalov [8].

The theoretical work estimates were pessimistic, and the method was not put into practice at the time. The first practical results were reported in a pioneering paper by Brandt [19], who published another paper in 1977 [20], clearly outlining the main principles and the practical utility of multigrid methods, which drew wide attention and marked the beginning of rapid development. The multigrid method was discovered independently by Hackbusch [50], who laid firm mathematical foundations and provided reliable methods ([52], [53], [54]). A report by Frederickson [47] describing an efficient multigrid algorithm for the Poisson equation led the present author to the development of a similar method for the vorticity-stream function formulation of the Navier-Stokes equations, resulting in an efficient method ([135], [143]).

At first there was much debate and scepticism about the true merits of multigrid methods. Only after sufficient initiation satisfactory results could be obtained. This led a number of researchers to the development of stronger and more transparent convergence proofs ([4], [93], [94], [51], [54], [136], [137]) (see [57] for a survey of theoretical developments). Although rate of convergence proofs of multigrid methods are complicated, their structure has now become more or less standartized and trasparenr. Other outhors have tried to spead confidence in multigrid methods by providing efficient and reliable computer programs, as much as possible of 'black-box' type, for uninitiated users. A survey will be given later. The 'multigrid guide' of Brandt ([16], [23]) was provided to give guidelines for researchers writing their own multigrid programs.

Year	64	66	71	72	73	75	76	77	78	79	80	81	82	83	84	85
Number	1	1	1	1	1	1	3	11	10	22	31	70	78	96	94	149

Table 1.0.1: Years number of multigrid publications

Scope of these notes

The following topics will not be treated here: parabolic equations, eigenvalue problems and integral equations. For an introduction to the application of multigrid methods to these subjects, see [56], [57] and [18]. There is relatively little material in these areas, although multigrid can be applied profitably. For important recent advances in the field of integral equations, see [25] and [130]. A recent publication on parabolic multigrid is [91]. Finite element methods will not be discussed, but finite volume and finite difference discretization will be taken as the point of departure. Although most theoretical work has been done in a variational framework, most applications use finite volumes or finite differences. The principles are the same, however, and the reader should have no difficulty in applying the principles outlined in this book in a finite element context.

Multigrid principles are much more widely applicable than just to the numerical solution of differential and integral equations. Applications in such diverse areas as control theory, optimization, pattern recognition, computational tomography and particle physics are beginning to appear. For a survey of the wide ranging applicability of multigrid principles, see [17], [18].

Notation
The notation is explained as it occurs. Latin letter like u denote unknown functions. The bold version $\boldsymbol{u}$ denotes a grid function, with value u_j in grid point x_j, intended as the discrete approximation of $u(x_j)$.

2 The basic principle of multigrid methods for partial differential equations

2.1 Introduction

In this chapter, the basic principle of multigrid for partial differential equations will be explained by studying a one-dimensional model problem. Of course, one-dimensional problems do not require application of multigrid methods, since for the algebraic systems that result from discretization direct solution is efficient, but in one dimension multigrid methods can be analysed by elementary methods, and their essential principle is easily demonstrated.

Introductions to the basic principles of multigrid methods are given by [20], [27], [28] and [141]. More advanced expositions are given by [112], [16] and [57], Chapter 2.

2.2 The basic principle

One-dimensional model problem
The following model problem will be considered

$$-d^2u/dx^2 = f(x) \quad \text{in} \quad \Omega = (0,1), \quad u(0) = du(1)/dx = 0 \tag{2.2.1}$$

A computational grid is defined by

$$G = \{x \in I\!R : x = x_j = jh,\ j = 1, 2, ..., 2n,\ h = 1/2n\} \tag{2.2.2}$$

The points $\{x_j\}$ are called the *vertices* of the grid.
Equation (2.2.1) is discretized with finite differences as

$$h^{-2}(2u_1 - u_2) \quad = \quad f_1$$

$$
\begin{aligned}
h^{-2}(-u_{j-1}+2u_j-u_{j+1}) &= f_j, \quad j=2,3,...,2n-1 \\
h^{-2}(-u_{2n-1}+u_{2n}) &= \frac{1}{2}f_{2n}
\end{aligned}
\tag{2.2.3}
$$

where $f_j = f(x_j)$ and u_j is intended to approximate $u(x_j)$. The solution of Equation (2.2.1) is denoted by u, the solution of Equation (2.2.3) by $\boldsymbol{u}$ and the value of $\boldsymbol{u}$ in x_j by u_j u_j approximates the solution in the vertex x_j; thus Equation (2.2.3) is called a *vertex-centered discretization.* The number of meshes in G is even, to facilitate application of a two-grid method. The system (2.2.3) is denoted by

$$
\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{2.2.4}
$$

Gauss-Seidel iteration

In multidimensional applications of finite difference methods, the matrix $\boldsymbol{A}$ is large and sparse, and the non-zero pattern has a regular structure. These circumstances favour the use of iterative methods for solving (2.2.4). We will present one such method. Indicating the mth iterand by a superscript m, the *Gauss-Seidel iteration method* for solving (2.2.3) is defined by, assuming an initial guess $\boldsymbol{u}^0$ is given,

$$
\begin{aligned}
2u_1^m &= u_2^{m-1} + h^2 f_1 \\
-u_{j-1}^m + 2u_j^m &= u_{j+1}^{m-1} + h^2 f_j, \; j=2,3,...,2n-1 \\
-u_{2n-1}^m + u_{2n}^m &= \frac{1}{2}h^2 f_{2n}
\end{aligned}
\tag{2.2.5}
$$

Fourier analysis of convergence

For ease of analysis, we replace the boundary conditions by *periodic boundary conditions*:

$$
u(1) = u(0) \tag{2.2.6}
$$

Then the error $\boldsymbol{e}^m = \boldsymbol{u}^m - \boldsymbol{u}^\infty$ is periodic and satisfies

$$
-e_{j-1}^m + 2e_j^m = e_{j+1}^{m-1}, \quad e_j^m = e_{j+2n}^m \tag{2.2.7}
$$

As will be discussed in more detail later, such a periodic grid function can be represented by the following Fourier series:

$$
e_j^m = \sum_{\alpha=-n+1}^{n} c_\alpha^m exp(ij\theta_\alpha), \quad \theta_\alpha \equiv \pi\alpha/n \tag{2.2.8}
$$

Because of the orthogonality of $\{e^{ij\theta_\alpha}\}$, it suffies to substitute $e_j^{m-1} = c_\alpha^{m-1}e^{ij\theta_\alpha}$ in (2.2.7). This gives $e_j^m = c_\alpha^m e^{ij\theta_\alpha}$ with

$$
c_\alpha^m = g(\theta_\alpha)c_\alpha^{m-1}, \quad g(\theta_\alpha) = e^{i\theta_\alpha}/(2-e^{-i\theta_\alpha}) \tag{2.2.9}
$$

The function $g(\theta_\alpha)$ is called the *amplification factor*. It measures the growth or decay of a Fourier mode of the error during an iteration. We find

$$|g(\theta_\alpha)| = (5 - 4\cos\theta_\alpha)^{-1/2} \tag{2.2.10}$$

At first it seems that Gauss-Seidel does not converge, because

$$\max\{|g(\theta_\alpha)| : \theta_\alpha = \pi\alpha/n,\ \alpha = -n+1, -n+2, ..., n\} = |g(0)| = 1 \tag{2.2.11}$$

However, with periodic boundary conditions the solution of (2.2.1) is determined up to a constant only, so that there is no need to require that the Fourier mode $\alpha = 0$ decays during iteration. Equation (2.2.11), therefore, is not a correct measure of convergence, but the following quantity is:

$$\begin{aligned}\max\{|g(\theta_\alpha)| : \theta_\alpha &= \pi\alpha/n,\ \alpha = -n+1, -n+2, ..., n, \alpha \neq 0\} = |g(\theta_1)| \\ &= \{1 - 2\theta_1^2 + O(\theta_1^4)\}^{-1/2} = 1 - 4\pi^2 h^2 + O(h^4)\ . \end{aligned} \tag{2.2.12}$$

It follows that the rate of convergence deteriorates as $h \downarrow 0$. Apart from special cases, in the context of elliptic equations this is found to be true of all socalled *basic iterative methods* (more on these later; well known examples are the Jacobi, Gauss-Seidel and successive over-relaxation methods) by which a grid function value is updated using only neighbouring vertices. This deterioration of rate of convergence is found to occur also with other kinds of boundary conditions. The purpose of multigrid is to avoid this deterioration, and to achieve a rate of convergence which is independent of h.

The essential multigrid principle

The rate of convergence of basic iterative methods can be improved with multigrid methods. The basic observation is that (2.2.10) shows that $|g(\theta_\alpha)|$ decreases as α increases. This means that, although long wavellength Fourier modes (α close to 1) decay slowly ($|g(\theta_\alpha)| = 1 - O(h^2)$), short wavelength Fourier modes are reduced rapidly. The essential multigrid principle is to approximate the smooth (long wavelength) part of the error on coarser grids. The non-smooth or rough part is reduced with a small number (independent of h) of iterations with a basic iterative method on the fine grid.

Fourier smoothing analysis

In order to be able to verify whether a basic iterative method gives a good reduction of the rough part of the error, the concept of roughness has to be defined precisely.

Definition 2.2.1 The set of rough wavenumbers Θ_r is defined by

$$\Theta_r = \{\theta_\alpha = \pi\alpha/n,\ |\alpha| \geq cn,\ \alpha = -n+1, -n+2, ..., n\} \tag{2.2.13}$$

where $0 < c < 1$ is fixed constant independent of n.

The performance of a smoothing method is measured by its *smoothing factor* ρ, defined as follows.

Definition 2.2.2 The smoothing factor ρ is defined by

$$\rho = \max\{|g(\theta_\alpha)| : \theta_\alpha \in \Theta_r\} \tag{2.2.14}$$

When for a basic iterative method $\rho < 1$ is bounded away from 1 *uniformly* in h, we say that the method is a *smoother*. Note that ρ depends on the iterative method and on the problem. For Gauss-Seidel and the present model problem ρ is easily determined. Equation (2.2.10) shows that $|g|$ decreases monotonically, so that

$$\rho = (5 - 4\cos c\pi)^{-1/2} \tag{2.2.15}$$

Hence, for the present problem Gauss-Seidel is a smoother.
It is convenient to standardize the choice of c. Only the Fourier modes that cannot be represented on the coarse grid need to be reduced by the basic iterative method; thus it is natural to let these modes constitute Θ_r. We choose the coarse grid by doubling the mesh-size of G. The Fourier modes on this grid have wavenumbers θ_α given by (2.2.8) with n replaced by $n/2$ (assuming for simplicity n to be even). The remaining wavenumbers are defined to be non-smooth, and are given by (2.2.13) with

$$c = 1/2 \tag{2.2.16}$$

Equation (2.2.15) then gives the following smoothing factor for Gauss-Seidel

$$\rho = 5^{-1/2} \tag{2.2.17}$$

This type of Fourier smoothing analysis was originally introduced by Brandt [20]. It is a useful and simple tool. When the boundary conditions are not periodic, its predictions are found to remain qualitatively correct, except in the case of singular perturbation problems, to be discussed later.

With smoothly varying coefficients, experience shows that a smoother which performs well in the 'frozen coefficient' case, will also perform well for variable coefficients. By the 'frozen coefficient' case we mean a set of constant coefficient cases, with coefficient values equal to the values of the variable coefficients under consideration in a sufficiently large sample of points in the domain.

Exercise 2.2.1 Determine the smoothing factor of the damped Jacobi method (defined later) to problem (2.2.5) with boundary conditions (2.2.6). Note that with damping parameter $\omega = 1$ this is not a smoother.

Exercise 2.2.2 Determine the smoothing factor of the Jacobi method applied to problem (2.2.5) with Dirichlet boundary conditions $u(0) = u(1) = 0$, by using the Fourier sine series. Note that the smoothing factor is the same as obtained with the exponential Fourier series.

Exercise 2.2.3 Determine the smoothing factor of the Gauss-Seidel method for central discretization of the convection-diffusion equation $cdu/dx - \varepsilon d^2u/dx^2 = f$. Show that for $|c|h/\varepsilon \gg 1$ and $c < -1$ we have no smoother.

2.3 The two-grid algorithm

Coarse grid approximation
A *coarse grid* $\bar{G}$ is defined by doubling the mesh-size of G:

$$\bar{G} = \{x \in I\!R : x = x_j = j\bar{h},\ j = 1, 2, ..., n,\ \bar{h} = 1/n\} \tag{2.3.1}$$

The vertices of $\bar{G}$ also belong to G; thus this is called *vertex-centered coarsening.* The original grid G is called the *fine grid.* Let

$$U : G \to I\!R, \quad \bar{U} : \bar{G} \to I\!R \tag{2.3.2}$$

be the sets of fine and coarse grid functions, respectively. A *prolongation operator* $\boldsymbol{P} : \bar{U} \to U$ is defined by linear interpolation:

$$\boldsymbol{P}\bar{u}_{2j} = \bar{u}_j, \quad \boldsymbol{P}\bar{u}_{2j+1} = \frac{1}{2}(\bar{u}_j + \bar{u}_{j+1}) \tag{2.3.3}$$

Overbars indicate coarse grid quantities. A *restriction operator* $\boldsymbol{R} : U \to \bar{U}$ is defined by the following weighted average

$$\boldsymbol{R}u_j = \frac{1}{4}u_{2j-1} + \frac{1}{2}u_{2j} + \frac{1}{4}u_{2j+1} \tag{2.3.4}$$

where u_j is defined to be zero outside G. Note that the matrices $\boldsymbol{P}$ and $\boldsymbol{R}$ are related by $\boldsymbol{R} = \frac{1}{2}\boldsymbol{P}^T$, but this property is not essential.
The fine grid equation (2.2.4) must be approximated by a coarse grid equation

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

Like the finite grid matrix $\boldsymbol{A}$, the coarse grid matrix $\bar{\boldsymbol{A}}$ may be obtained by discretizing Equation (2.2.1). This is called *discretization coarse grid approximation.* An alternative is the following. The fine grid problem (2.2.4) is equivalent to

$$(\boldsymbol{A}\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), \quad \boldsymbol{u} \in U,\ \forall \boldsymbol{v} \in U \tag{2.3.5}$$

with $(.,.)$ the standard inner product on U. We want to find an approximated solution $\boldsymbol{P}\bar{\boldsymbol{u}}$ with $\bar{\boldsymbol{u}} \in \bar{U}$. This entails restriction of the test functions $\boldsymbol{v}$ to a subspace with the same dimension as $\bar{U}$, that is, test functions of the type $\tilde{\boldsymbol{P}}\bar{\boldsymbol{v}}$ with $\bar{\boldsymbol{v}} \in \bar{U}$, and $\tilde{\boldsymbol{P}}$ a prolongation operator that may be different from $\boldsymbol{P}$:

$$(\boldsymbol{A}\boldsymbol{P}\bar{\boldsymbol{u}}, \bar{\boldsymbol{P}}\bar{\boldsymbol{v}}) = (\boldsymbol{f}, \bar{\boldsymbol{P}}\bar{\boldsymbol{v}}), \quad \bar{\boldsymbol{u}} \in \bar{U},\ \forall \bar{\boldsymbol{v}} \in \bar{U} \tag{2.3.6}$$

or

$$(\bar{\boldsymbol{P}}^{*}\boldsymbol{A}\boldsymbol{P}\bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}) = (\tilde{\boldsymbol{P}}^{*}\boldsymbol{f}, \bar{\boldsymbol{v}}), \quad \bar{\boldsymbol{u}} \in \bar{U},\ \forall \bar{\boldsymbol{v}} \in \bar{U} \tag{2.3.7}$$

where now of course $(.,.)$ is over $\bar{U}$, and superscript * denotes the adjoint (or transpose in this case). Equation (2.3.7) is equivalent to

$$\bar{\boldsymbol{A}}\bar{\boldsymbol{u}} = \bar{\boldsymbol{f}} \tag{2.3.8}$$

with

$$\bar{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}\boldsymbol{P} \tag{2.3.9}$$

and $\bar{\boldsymbol{f}} = \boldsymbol{R}\boldsymbol{f}$; we have replaced $\bar{\boldsymbol{P}}^{*}$ by $\boldsymbol{R}$. This choice of $\bar{\boldsymbol{A}}$ is called *Galerkin coarse grid approximation.*

With $\boldsymbol{A}$, $\boldsymbol{P}$ and $\boldsymbol{R}$ given by (2.2.3), (2.3.3) and (2.3.4), Equation (2.3.9) results in the following $\bar{\boldsymbol{A}}$

$$\begin{aligned}
\bar{\boldsymbol{A}}\bar{u}_1 &= \bar{h}^{-2}(2\bar{u}_1 - \bar{u}_2) \\
\bar{\boldsymbol{A}}\bar{u}_j &= \bar{h}^{-2}(-\bar{u}_{j-1} + 2\bar{u}_j - \bar{u}_{j+1})\,, \quad j = 2, 3, ..., n-1 \\
\bar{\boldsymbol{A}}\bar{u}_n &= \bar{u}^{-2}(-\bar{u}_{n-1} + \bar{u}_n)
\end{aligned} \tag{2.3.10}$$

which is the coarse grid equivalent of the left-hand side of (2.2.3). Hence, in the present case there is no difference between Galerkin and discretization coarse grid approximation. The derivation of (2.3.10) is discussed in Exercise 2.3.1. The formula (2.3.9) has theoretical advantages, as we shall see.

Coarse grid correction

Let $\hat{\boldsymbol{u}}$ be an approximation to the solution of (2.2.4). The error $\boldsymbol{e} \equiv \hat{\boldsymbol{u}} - \boldsymbol{u}$ is to be approximated on the coarse grid. We have

$$\boldsymbol{A}\boldsymbol{e} = -\boldsymbol{r} \equiv \boldsymbol{A}\hat{\boldsymbol{u}} - \boldsymbol{f} \tag{2.3.11}$$

The coarse grid approximation $\bar{\boldsymbol{u}}$ of $-\boldsymbol{e}$ satisfies

$$\bar{\boldsymbol{A}}\bar{\boldsymbol{u}} = \boldsymbol{R}\boldsymbol{r} \tag{2.3.12}$$

In a two-grid method it is assumed that (2.3.12) is solved exactly. the coarse grid correction to be added to $\hat{\boldsymbol{u}}$ is $\boldsymbol{P}\bar{\boldsymbol{u}}$:

$$\hat{\boldsymbol{u}} := \hat{\boldsymbol{u}} + \boldsymbol{P}\bar{\boldsymbol{u}} \tag{2.3.13}$$

Linear two-grid algorithm

The two-grid algorithm for linear problems consists of smoothing on the fine grid, approximation of the required correction on the coarse grid, prolongation of the coarse grid correction to the fine grid, and again smoothing on the fine grid. The precise definition of the two-grid algorithm is

comment Two-grid algorithm;
Initialize $\boldsymbol{u}^0$;

for $\quad i := 1$ *step* 1 *until ntg do*
$\quad \boldsymbol{u}^{1/3} := S(\boldsymbol{u}^0, \boldsymbol{A}, \boldsymbol{f}, \nu_1);$
$\quad \boldsymbol{r} := \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}^{1/3};$
$\quad \bar{\boldsymbol{u}} := \bar{\boldsymbol{A}}^{-1}\boldsymbol{R}\boldsymbol{r};$
$\quad \boldsymbol{u}^{2/3} := \boldsymbol{u}^{1/3} + \boldsymbol{P}\bar{\boldsymbol{u}};$
$\quad \boldsymbol{u}^1 := S(\boldsymbol{u}^{2/3}, \boldsymbol{A}, \boldsymbol{f}, \nu_2);$
$\quad \boldsymbol{u}^0 := \boldsymbol{u}^1;$
od (2.3.14)

The number of two-grid iterations carried out is ntg. $S(\boldsymbol{u}^0, \boldsymbol{A}, \boldsymbol{f}, \nu_1)$ stands for ν_1 smoothing iterations, for example with the Gauss-Seidel method discussed earlier, applied to $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$, starting with $\boldsymbol{u}^0$. The first application of S is called *pre-smoothing*, the second *post-smoothing.*

Exercise 2.3.1 Derive (2.3.10) (Hint. It is easy to write down $\boldsymbol{R}\boldsymbol{A}u_i$ in the interior and at the boundaries. Next, one replaces u_i by $\boldsymbol{P}\bar{u}_i$.)

2.4 Two-grid analysis

The purpose of two-grid analysis (as of multigrid analysis) is to show that the rate of convergence is independent of the mesh-size h. We will analyse algorithm (2.3.14) for the special case $\nu_1 = 0$ (no-presmoothing).

Coarse grid correction

From (2.3.14) it follows that after coarse grid correction the error $\boldsymbol{e}^{2/3} \equiv \boldsymbol{u}^{2/3} - \boldsymbol{u}$ satisfies

$$\boldsymbol{e}^{2/3} = \boldsymbol{e}^{1/3} + \boldsymbol{P}\bar{\boldsymbol{u}}^{1/3} = \boldsymbol{E}\boldsymbol{e}^{1/3} \tag{2.4.1}$$

with the *iteration matrix* or *error amplification matrix* $\boldsymbol{E}$ defined by

$$\boldsymbol{E} \equiv \boldsymbol{I} - \boldsymbol{P}\bar{\boldsymbol{A}}^{-1}\boldsymbol{R}\boldsymbol{A} \tag{2.4.2}$$

We will express $\boldsymbol{e}^{2/3}$ explicitly in terms of $\boldsymbol{e}^{1/3}$. This is possible only in the present simple one-dimensional case, which is our main motivation for studying this case. Let

$$\boldsymbol{e}^{1/3} = \boldsymbol{d} + \boldsymbol{P}\bar{\boldsymbol{e}}, \quad \text{with } \; \bar{e}_j \equiv e_{2j}^{1/3} \tag{2.4.3}$$

Then it follows that

$$\boldsymbol{e}^{2/3} = \boldsymbol{E}\boldsymbol{e}^{1/3} = \boldsymbol{E}\boldsymbol{d} \tag{2.4.4}$$

We find from (2.4.3) that

$$d_{2j} = 0, \quad d_{2j+1} = -\frac{1}{2}e_j^{1/3} + e_{2j+1}^{1/3} - \frac{1}{2}e_{2j+2}^{1/3} \tag{2.4.5}$$

Furthermore, from (2.4.5) it follows that

$$\boldsymbol{R}\boldsymbol{A}\boldsymbol{d} = 0 \tag{2.4.6}$$

so that

$$\boldsymbol{e}^{2/3} = \boldsymbol{d} \tag{2.4.7}$$

Smoothing

Next, we consider the effect of post-smoothing by one Gauss-Seidel iteration. From (2.2.5) it follows that the error after post-smoothing $\boldsymbol{e}^1 = \boldsymbol{u}^1 - \boldsymbol{u}$ is related to $\boldsymbol{e}^{2/3}$ by

$$\begin{aligned} 2e_1^1 &= e_2^{2/3} \\ -e_{j-1}^1 + 2e_j^1 &= e_{j+1}^{2/3}\,, \quad j = 2,3,...,2n-1 \\ -e_{2n-1}^1 + e_{2n}^1 &= 0 \end{aligned} \tag{2.4.8}$$

Using (2.4.5)(2.4.7) this can be rewritten as

$$\begin{aligned} e_1^1 &= 0 \\ e_{2j}^1 &= \frac{1}{2}d_{2j+1} + \frac{1}{4}e_{2j-2}^1\,, \quad e_{2j+1}^1 = \frac{1}{2}e_{2j}^1\,, \quad j = 1,2,...,n-1 \\ e_{2n}^1 &= e_{2n-1}^1 \end{aligned} \tag{2.4.9}$$

By induction it is easy to see that

$$|e_{2j}^1| \leq \frac{2}{3}\|d\|_\infty\,, \quad \|d\|_\infty = \max\{|d_j| : j = 1,2,...,2n\} \tag{2.4.10}$$

Since $d = e^{2/3}$, we see that Gauss-Seidel reduces the maximum norm of the error by a factor 2/3 or less.

Rate of convergence

Since $e_0^{1/3} = 0$ because of the boundary conditions, it follows from (2.4.5) that

$$|\boldsymbol{d}|_\infty \leq \frac{1}{2}|\boldsymbol{Ae}^0|_\infty \tag{2.4.11}$$

since $\boldsymbol{e}^{1/3} = \boldsymbol{e}^0$ (no pre-smoothing).
From (2.4.9) it follows that

$$\begin{aligned} Ae^1_{2j} &= \tfrac{3}{4}d_{2j+1} - \tfrac{1}{16}d_{2j-1} - \tfrac{1}{64}d_{2j-3} - \dots \\ Ae^1_{2j-1} &= -\tfrac{1}{2}e^1_{2j} \end{aligned} \tag{2.4.12}$$

Hence, using (2.4.10),

$$|Ae^1_{2j}| \leq \frac{5}{6}\|\boldsymbol{d}\|_\infty\ , \quad |A^1e_{2j-1}| \leq \frac{1}{3}\|\boldsymbol{d}\|_\infty \tag{2.4.13}$$

Substitution of (2.4.11) gives

$$\|\boldsymbol{r}^1\|_\infty \leq \frac{5}{12}\|\boldsymbol{r}^0\|_\infty \tag{2.4.14}$$

where $\boldsymbol{r} \equiv \boldsymbol{Ae}$ is the residual. This shows that the maximum norm is reduced by a factor of 5/12 or better, *independent of the mesh-size.*

This type of analysis is restricted to the particular case at hand. More general cases will be treated later by Fourier-analysis. There a drawback is of course the assumption of periodic boundary conditions. The general proofs of rate of convergence referred to in the introduction do not give sharp estimates. Therefore the sharper estimates obtainable by Fourier analysis are more useful for debugging codes. On the sharpness of rate of convergence predictions based on Fourier analysis, see [24].

Again: the essential principle

How is the essential principle of multigrid, discussed in Section 2.2, recognized in the foregoing analysis? Equations (2.4.6) and (2.4.7) show that

$$\boldsymbol{RAe}^{2/3} = 0 \tag{2.4.15}$$

Application of $\boldsymbol{R}$ means taking a local average with positive weights; thus (2.4.15) implies that $\boldsymbol{Ae}^{2/3}$ has many sign changes, and is therefore rough. Since $\boldsymbol{Ae}^{2/3} = \boldsymbol{Au}^{2/2} - \boldsymbol{f}$ is the residual, we see that after coarse grid correction the residual is rough. The smoother is efficient in reducing this non-smooth residual further, which explains the h-independent reduction shown in (2.4.14).

Exercise 2.4.1 In the definition of G (2.2.2) and $\bar{G}$ (2.3.1) we have not included the point $x = 0$, where a Dirichlet condition holds. If Neumann condition is given at $x = 0$, the point $x = 0$ must be included in G and $\bar{G}$. If one wants to write a general multigrid program for both cases, $x = 0$ has to be included. Repeat the foregoing analysis of the two-grid algorithm with $x = 0$ included in G and $\bar{G}$. Note that including $x = 0$ makes $\boldsymbol{A}$ non-symmetric. This difficulty does not occur with cell-centered discretization, to be discussed in the next chapter.

3 Basic Iterative Methods

3.1 Introduction

Smoothing methods in multigrid algorithms are usually taken from the class of *basic iterative methods*, to be defined below. This chapter presents an introduction to these methods. A more detailed account may be found in [141].

Basic iterative methods
Suppose that discretization of the partial differential equation to be solved leads to the following linear algebraic system

$$\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b} \tag{3.1.1}$$

Let the matrix $\boldsymbol{A}$ be split as

$$\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N} \tag{3.1.2}$$

with $\boldsymbol{M}$ non-singular. Then the following iteration method for the solution of (3.1.1) is called a *basic iterative method*:

$$\boldsymbol{M}\boldsymbol{y}^{m+1} = \boldsymbol{N}\boldsymbol{y}^{m} + \boldsymbol{b} \tag{3.1.3}$$

or

$$\boldsymbol{y}^{m+1} = \boldsymbol{S}\boldsymbol{y}^{m} + \boldsymbol{T}\boldsymbol{b} \tag{3.1.4}$$

with

$$\boldsymbol{S} = \boldsymbol{M}^{-1}\boldsymbol{N}\ , \quad \boldsymbol{T} = \boldsymbol{M}^{-1} \tag{3.1.5}$$

so that we have

$$\boldsymbol{y}^{m+1} = \boldsymbol{S}\boldsymbol{y}^{m} + \boldsymbol{M}^{-1}\boldsymbol{b}\ , \quad \boldsymbol{S} = \boldsymbol{M}^{-1}\boldsymbol{N}\ , \quad \boldsymbol{N} = \boldsymbol{M} - \boldsymbol{A} \tag{3.1.6}$$

The matrix $\boldsymbol{S}$ is called the *iteration matrix* of iteration method (3.1.6).
Basic iterative method may be *damped*, by modifying (3.1.6) as follows

$$\begin{aligned} \boldsymbol{y}^{*} &= \boldsymbol{S}\boldsymbol{y}^{m} + \boldsymbol{M}^{-1}\boldsymbol{b} \\ \boldsymbol{y}^{m+1} &= \omega\boldsymbol{y}^{*} + (1-\omega)\boldsymbol{y}^{m} \end{aligned} \tag{3.1.7}$$

By elimination of $\boldsymbol{y}^*$ one obtains

$$\boldsymbol{y}^{m+1} = \boldsymbol{S}^8\boldsymbol{y}^m + \omega\boldsymbol{M}^{-1}\boldsymbol{b} \tag{3.1.8}$$

with

$$\boldsymbol{S}^* = \omega\boldsymbol{S} + (1-\omega)\boldsymbol{I} \tag{3.1.9}$$

The eigenvalues of the undamped iteration matrix $\boldsymbol{S}$ and the damped iteration matrix $\boldsymbol{S}^*$ are related by

$$\lambda(\boldsymbol{S}^*) = \omega\lambda(\boldsymbol{S}) + 1 - \omega \tag{3.1.10}$$

Although the possibility that a divergent method (3.1.6) or (3.1.8) is a good smoother (a concept to be explained later) cannot be excluded, the most likely candidates for good smoothing methods are to be found among convergent methods. In the next section, therefore, some results on convergence of basic iterative methods are presented. For more background, see [129] and [151].

Exercise 3.1.1 Show that (3.1.8) corresponds to the splitting

$$\boldsymbol{M}^* = \boldsymbol{M}/\omega\ , \quad \boldsymbol{N}^* = \boldsymbol{A} - \boldsymbol{M}^* \tag{3.1.11}$$

3.2 Convergence of basic iterative methods

Convergence

In the convergence theory for (3.1.3) the following concepts play an important role. We have $\boldsymbol{My} = \boldsymbol{Ny} + b$, so that the error $\boldsymbol{e}^m = \boldsymbol{y}^m - \boldsymbol{y}$ satisfies

$$\boldsymbol{e}^{m+1} = \boldsymbol{S}\boldsymbol{e}^m \tag{3.2.1}$$

As shown in many textbooks, we have

Theorem 3.2.2 Convergence of (3.1.3) is equivalent to

$$\rho(\boldsymbol{S} < 1 \tag{3.2.2}$$

with $\rho(\boldsymbol{S})$ the spectral radius of $\boldsymbol{S}$.

Regular splittings and M- and K-matrices

Definition 3.2.2 The splitting (3.1.2) is called *regular* if $\boldsymbol{M}^{-1} \geq 0$ and $\boldsymbol{N} \geq 0$ (elementwise). The splitting is *convergent* when (3.1.3) converges.

Definition 3.2.3 ([129], Definition 3.3). The matrix $\boldsymbol{A}$ is called an *M-matrix* if $a_{ij} \leq 0$ for all i, j with $i \neq j$, $\boldsymbol{A}$ is non-singular and $\boldsymbol{A}^{-1} \geq 0$ (elementwise).

Theorem 3.2.3 A regular splitting of am M-matrix is convergent.

Proof. See [129] Theorem 3.13. □

Unfortunately, a regular splitting of an M-matrix does not necessary give a smoothing method. A counterexample is the Jacobi method (to be discussed shortly) applied to Laplace's equation (see later). In practice, however, it is easy to find good smoothing methods if $\boldsymbol{A}$ is an M-matrix. As discussed in [145], a convergent iterative method can always be turned into a smoothing method by introduction of *damping*. We will find later that often the efficacy of smoothing methods to be enhanced significantly by damping. Damped version of the methods to be discussed are obtained easily, using equations (3.1.8), (3.1.9) and (3.1.10).

Hence, it is worthwhile to try to discretize in such way that the resulting matrix $\boldsymbol{A}$ is an M-matrix. In order to make it easy to see if a discretization matrix is an M-matrix we present the following theorem.

Definition 3.2.4 A matrix $\boldsymbol{A}$ is called *irreducible* if from (3.1.1) one cannot extract a subsystem that can be solved independently.

Definition 3.2.5 A matrix $\boldsymbol{A}$ is called a *K-matrix* if

$$a_{ii} > 0, \forall i , \tag{3.2.3}$$

$$a_{ij} \leq 0, \forall i, j \quad \text{with} \quad i \neq j \tag{3.2.4}$$

and

$$\sum_j a_{ij} \geq 0 , \forall i , \tag{3.2.5}$$

with strict inequality for at least one i.

Theorem 3.2.4 An irreducible K-matrix is an M-matrix.

Proof. See [141]. □

Note that inspection of the K-matrix property is easy.
The following theorem is helpful in the construction of regular splittings.

Theorem 3.2.5 Let $\boldsymbol{A}$ be an M-matrix. If $\boldsymbol{M}$ is obtained by replacing certain elements a_{ij} with $i \neq j$ by values b_{ij} satisfying $a_{ij} \leq b_{ij} \leq 0$, then $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ is a regular splitting.

Proof. This theorem is an easy generalization of Theorem 3.14 in [129] suggested by Theorem 2.2 in [87]. □

The basic iterative methods to be considered all result in regular splittings, and lead to numerically stable algorithms, if $\boldsymbol{A}$ is an M-matrix. This is one reason why it is advisable to discretize the partial differential equation to be solved in such a way that the resulting matrix is an M-matrix. This may require upwind differencing for first derivatives. Another reason is the exclusion of numerical wiggles in the computed solution, because a monotonicity principle is associated with the M-matrix property.

3.3 Examples of basic iterative methods: Jacobi and Gauss-Seidel

We present a number of (mostly) common basic iterative methods by defining the corresponding splittings (3.1.2). We assume that $\boldsymbol{A}$ arises from a discretization on a two-dimensional structured grid.

Point Jacobi. $\boldsymbol{M} = diag\,(\boldsymbol{A})$.

Block Jacobi. $\boldsymbol{M}$ is obtained from $\boldsymbol{A}$ by replacing a_{ij} for all i, j with $j \neq i, i \pm 1$ by zero. With the forward ordering of the grid points of Figure 3.3.1 this gives horizontal line Jacobi; with the forward vertical line ordering of Figure 3.3.2 one obtains vertical line Jacobi. One horizontal line Jacobi iteration followed by one vertical line Jacobi iteratin gives alternating Jacobi.

16	17	18	19	20
11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

Forward

5	4	3	2	1
10	9	8	7	6
15	14	13	12	11
20	19	18	17	16

Backward

18	9	19	10	20
6	16	7	17	8
13	4	14	5	15
1	11	2	12	3

White-black

10	14	17	19	20
6	9	13	16	18
3	5	8	12	15
1	2	4	7	11

Diagonal

16	19	17	20	18
11	14	12	15	13
6	9	7	10	8
1	4	2	5	3

Horizontal forward
white-black

17	13	9	5	1
19	15	11	7	3
18	14	10	6	2
20	16	12	8	4

Vertical backward
white-black

Figure 3.3.1: Grid point orderings for point Gauss-Seidel.

Point Gauss-Seidel. $\boldsymbol{M}$ is obtained from $\boldsymbol{A}$ replacing a_{ij} for all i, j with $j > i$ by zero.

4	8	12	16	20
3	7	11	15	19
2	6	10	14	18
1	5	9	13	17

Forward vertical line

16	17	18	19	20
6	7	8	9	10
11	12	13	14	15
1	2	3	4	5

Horizontal zebra

4	16	8	20	12
3	15	7	19	11
2	14	6	18	10
1	13	5	17	9

Vertical zebra

Figure 3.3.2: Grid point orderings for block Gauss-Seidel.

Block Gauss-Seidel. $\boldsymbol{M}$ is obtained from $\boldsymbol{A}$ by replacing a_{ij} for all i, j with $j > i+1$ by zero.

From Theorem 4.2.8 it is immediately clear that, if $\boldsymbol{A}$ is an M-matrix, then the Jacobi and Gauss-seidel methods correspond to regular splittigs.

Gaus-Seidel variants

It turns out that the efficiency of Gauss-Seidel methods depends strongly on the ordering of equations and unknowns in many applications. Also, the possibilities of vectorized and parallel computing depend strongly on this ordering. We now, therefore, discuss some possible orderings. The equations and unknowns are associated in a natural way with points in a computational grid. It suffices, therefore, to discuss orderings of computational grid points. We restrict ourselves to a two-dimensional grid G, which is enough to illustrate the basic ideas. G is defined by

$$G = \{(i, j) : i = 1, 2, ..., I; j = 1, 2, ..., J\} \tag{3.3.1}$$

The points of G represent either vertices or cell centres, depending on the discretization method.

Forward or lexicographic ordering

The grid points are numbered as follows

$$k = i + (j-1)I \tag{3.3.2}$$

Backward ordering

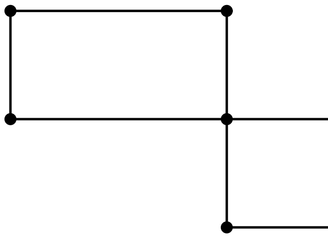
This ordering corresponds to the enumeration

$$k = IJ + 1 - i - (j-1)I \tag{3.3.3}$$

White-black ordering
This ordering corresponds to a chessboard colouring of G, numbering first the black points and then the white points, or vice versa; cf. Figure 3.3.1.

Diagonal ordering
The points are numbered per diagonal, starting in a corner; see Figure 3.3.1. Diferent variants are obtained by starting in different corners. If the matrix $\boldsymbol{A}$ corresponds to a discrete operator with a stencil as in Figure 3.3.3(b), then point Gauss-Seidel with the diagonal ordering of Figure 3.3.1 is mathematically equivalent to forward Gauss-Seidel.

2cm



(a)

Figure 3.3.3: Discretization stencils.

Point Gauss-Seidel-Jacobi
We propose this variant in order to facilitate vectorized and parallel computing; more on this shortly. $\boldsymbol{M}$ is obtained from $\boldsymbol{A}$ by replacing a_{ij} by zero except a_{ii} and $a_{i,i-1}$. We call this point Gauss-Seidel-Jacobi because this is a compromise between the point Gauss-Seidel and Jacobi methods discussed above. Four different methods are obtained with the following four orderings: the forward and backward orderings of Figure 3.3.1, the forward vertical line ordering of Figure 3.3.2, and this last ordering reversed. Applying these methods in succession results in *four-direction point Gauss-Seidel-Jacobi.*

White-black line Gauss-Seidel
This can be seen as a mixture of lexicographic and white-black ordering. The concept is best illustrated with a few examples. With *horizontal forward white-black Gauss-Seidel* the grid points are visited horizontal line by horizontal line in order of increasing j (forward), while per line the grid points are numbered in white-black order, cf. Figure 3.3.1. The lines can also be taken in order of decreasing j, resulting in *horizontal backward white-black Gaus-Seidel.* Doing one after the other gives *horizontal symmetric white-back Gauss-Seidel.* Doing one after the other gives *horizontal symmetric white-black Gauss-Seidel.* The lines can also be

taken vertically; Figure 3.3.1 illustrates *vertical backward white-black Gauss-Seidel*. Combining horizontal and vertical symmetric whte-black Gauss-Seidel gives *alternating white-black Gauss-Seidel*. White-black line Gauss-Seidel ordering has been proposed in [128].

Orderings for block Gauss-Seidel
With block Gauss-Seidel, the unknowns corresponding to lines in the grid are updated simultaneously. *Forward* and *backward horizontal line Gauss-Seidel* correspond to the forward and backward ordering, respectively, in Figure 3.3.1. Figure 3.3.2 gives some more orderings for block Gauss-Seidel.

Symmetric horizontal line Gauss-Seidel is forward horizontal line Gauss-Seidel followed by backward horizontal line Gauss-Seidel, or vice versa.*Alternating zebra Gauss-Seidel* is horizontal zebra followed by vertical zebra Gauss-Seidel, or vice versa. Other combinations come to mind easily.

Vectorized and parallel computing
The basic iterative methods discussed above differ in their suitability for computing with vector or parallel machines. Since the updated quantities are mutually independent, Jacobi parallizes and vectorizes completely, with vector length $I * J$. If the structure of the stencil $[\boldsymbol{A}]$ is as in Figure 3.3.3(c), then with zebra Gauss-Seidel the updated blocks are mutually independent, and can be handled simultaneously on a vector or a parallel machine. The same is true for point Gauss-Seidel if one chooses a suitable four-colour ordering scheme. The vector length for horizontal or vertical zebra Gauss-Seidel is J or I, respectively. The white and black groups in white-black Gauss-Seidel are mutually independent if the structure of $[\boldsymbol{A}]$ is given by Figure 3.3.4. The vector length is $I * J/2$. With diagonal Gauss-Seidel, the points inside a diagonal are mutually independent if the structure of $[\boldsymbol{A}]$ is given by Figure 3.3.3(b), if the diagonals are chosen as in Figure 3.3.1. The same is true when $[\boldsymbol{A}]$ has the structure given in Figure 3.3.3(a), if the diagonals are rotated by 90^0. The average vector length is roughly $I/2$ or $J/2$, depending on the length of largest the diagonal in the grid. With Gauss-Seidel-Jacobi lines in the grid can be handled in parallel; for example, with the forward ordering of Figure 3.3.1 the points on vertical lines Gauss-Seidel points of the same colour can be updated simultaneously, resulting in a vector length of $I/2$ or $J/2$, as the case may be.

Exercise 3.3.1 Let $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$, with $l_{ij} = 0$ for $j \geq i$, $\boldsymbol{D} = diag\ (\boldsymbol{A})$, and $u_{ij} = 0$ for $j \geq i$. Show that the iteration matrix of symmetric point Gauss-Seidel is given by

$$\boldsymbol{S} = (\boldsymbol{U} + \boldsymbol{D})^{-1}\boldsymbol{L}(\boldsymbol{L} + \boldsymbol{D})^{-1}\boldsymbol{U} \tag{3.3.4}$$

Exercise 3.3.2 Prove Theorem 3.3.1.

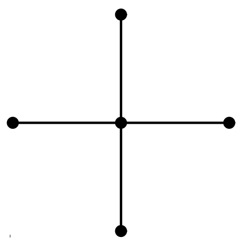


Figure 3.3.4: Five-point stencil.

3.4 Examples of basic iterative methods: incomplete point LU factorization

Complete LU factorization

When solving $Ay = b$ directly, a factorization $\boldsymbol{A} = \boldsymbol{LU}$ is constructed, with $\boldsymbol{L}$ and $\boldsymbol{U}$ a lower and an upper triangular matrix. This we call *complete factorization*. When $\boldsymbol{A}$ represents a discrete operator with stencil structure, for example, as in Figure 3.3.3, then $\boldsymbol{L}$ and $\boldsymbol{U}$ turn out to be much less sparse than $\boldsymbol{A}$, which renders this method inefficient for the class of problems under consideration.

Incomplete point factorization

With *incomplete factorization* or *incomplete* $\boldsymbol{LU}$ *factorization* ($\boldsymbol{ILU}$) one generates a splitting $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ with $\boldsymbol{M}$ having sparse and easy to compute lower and upper triangular factors $\boldsymbol{L}$ and $\boldsymbol{U}$:

$$\boldsymbol{M} = \boldsymbol{LU} \tag{3.4.1}$$

If $\boldsymbol{A}$ is symmetric one chooses a symmetric factorization:

$$\boldsymbol{M} = \boldsymbol{LL}^T \tag{3.4.2}$$

An alternative factorization of $\boldsymbol{M}$ is

$$\boldsymbol{M} = \boldsymbol{LD}^{-1}\boldsymbol{U} \tag{3.4.3}$$

With *incomplete point factorization*, $\boldsymbol{D}$ is chosen to be a diagonal matrix, and *diag* $(\boldsymbol{L}) =$ *diag* $(\boldsymbol{U}) = \boldsymbol{D}$, so that (3.4.3) and (3.4.1) are equivalent. $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ are determined as follows. A *graph* $\mathcal{G}$ of the incomplete decomposition is defined, consisting of two-tuples (i, j) for which the elements l_{ij}, d_{ii} and u_{ij} ae allowed to be non-zero. Then $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ are defined by

$$(\boldsymbol{LD}^{-1}\boldsymbol{U})_{kl} = a_{kl}, \quad \forall (k, l) \in \mathcal{G} \tag{3.4.4}$$

We will discuss a few variants of ILU factorization. These result in a splitting $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ with $\boldsymbol{M} = \boldsymbol{LD}^{-1}\boldsymbol{U}$. *Modified incomplete point factorization* is obtained if $\boldsymbol{D}$ is defined by

(3.4.4) is changed to $\boldsymbol{D} + \sigma\tilde{\boldsymbol{D}}$, with $\sigma \in I\!R$ a parameter, and $\tilde{D}$ a diagonal matrix defined by $\tilde{d}_{kk} = \sum\limits_{l \neq k} |n_{kl}|$. From now on the modified version will be discussed, since the unmodified version follows as a special case. This or similar modifications have been investigated in the context of multigrid methods in [65], [97], [83], [82] and [145], [147]. A srvey is given in [142]. We will discuss a few variants of modified ILU factorization.

Five-point ILU

Let the grid be given by (3.3.1), let the grid points be ordered according to (3.3.2), and let the structure of the stencil be given by Figure 3.3.3. Then the graph of $\boldsymbol{A}$ is

$$\mathcal{G} = \{(k, k-I),\ (k, k-1),\ (k, k),\ (k, k+1),\ (k, k+I)\} \tag{3.4.5}$$

For brevity the following notation is introduced

$$a_k = a_{k,k-I},\ c_k = a_{k,k-1},\ d_k = a_{kk},\ q_k = a_{k,k+1},\ g_k = a_{k,k+I} \tag{3.4.6}$$

Let the graph of the incomplete factorization be given by (3.4.5), and let the non-zero elements of $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ be called $\alpha_k, \gamma_k, \delta_k, \mu_k$ and η_k; the locations of these elements are identical to those of $a_k, ..., g_k$, respectively. Because the graph contains five elements, the resulting method is called *five-point ILU*. Let $\alpha, ..., \eta$ be the $IJ * IJ$ matrices with elements $\alpha_k, ..., \eta_k$, respectively, and similarly for $a, ..., g$. Then one can write

$$\boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} = \alpha + \gamma + \delta + \mu + \eta + \alpha\delta^{-1}\mu + \alpha\delta^{-1}\eta + \gamma\delta^{-1}\mu + \gamma\delta^{-1}\eta \tag{3.4.7}$$

From (3.4.4) it follows

$$\alpha = a, \quad \gamma = c, \quad \mu = q, \quad \eta = g \tag{3.4.8}$$

and, introducing modification as described above,

$$\delta + a\delta^{-1}g + c\delta^{-1}g = d + \sigma\tilde{d} \tag{3.4.9}$$

The rest matrix $\boldsymbol{N}$ is given by

$$\boldsymbol{N} = a\delta^{-1}q + c\delta^{-1}g + \sigma\tilde{d} \tag{3.4.10}$$

The only non-zero entries of $\boldsymbol{N}$ are

$$\begin{aligned} n_{k,k-I+1} &= a_k\delta_{k-I}^{-1}q_{k-I}, \quad n_{k,k+I-1} = c_k\delta_{k-1}^{-1}g_{k-1} \\ n_{kk} &= \sigma(|n_{k,k-I+1}| + |n_{k,k+I-1}|) \end{aligned} \tag{3.4.11}$$

Here and in the following elements in which indices outside the range $[1, IJ]$ occur are to be replaced by zero. From (3.4.9) the following recursion is obtained:

$$\delta_k = d_k - a_k\delta_{k-I}^{-1}g_{k-I} - c_k\delta_{k-1}^{-1}q_{k-1} + n_{kk} \tag{3.4.12}$$

This factorization has been studied in [41].
From (3.4.12) it follows that σ can overwrite d, so that the only additional storage required is for $\boldsymbol{N}$. When required, the residual $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{y}^{m+1}$ can be computed as follows without using $\boldsymbol{A}$:

$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{y}^{m+1} = \boldsymbol{N}(\boldsymbol{y}^{m+1} - \boldsymbol{y}) \tag{3.4.13}$$

which follows easily from (3.1.3). Since $\boldsymbol{N}$ is usually more sparse than $\boldsymbol{A}$, (3.4.13) is a cheap way to compute the residual. For all methods of type (3.1.3) one needs to store only $\boldsymbol{M}$ and $\boldsymbol{N}$, and $\boldsymbol{A}$ can be overwritten.

Seven-point ILU
The terminology *seven-point ILU* indicates that the graph of the incomplete factorization has seven elements. The graph $\mathcal{G}$ is chosen as follows:

$$\mathcal{G} = \{(k, k \pm I),\ (k, k \pm I \mp 1),\ (k, k \pm 1),\ (k, k)\} \tag{3.4.14}$$

For the computation of $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ see [141]. $\boldsymbol{L}, \boldsymbol{D} and \boldsymbol{U}$ can overwrite $\boldsymbol{A}$. The only additional storage required is for $\boldsymbol{N}$. Or, if one prefers, elements of $\boldsymbol{N}$ can be computed when needed.

Nine-point ILU
The principles are the same as for five- and seven-point ILU. Now the graph $\mathcal{G}$ has nine elements, chosen as follows

$$\mathcal{G} = \mathcal{G}_1 \cup \{(k, k \pm I \pm 1)\} \tag{3.4.15}$$

with $\mathcal{G}_1$ given by (3.4.14).
For the computation of $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ see [141].

Alternating ILU
Alternating ILU consists of one ILU iteration of the type just discussed or similar, followed by a second ILU iteration based on a different ordering of the grid points. As an example, let the grid be defined by (3.3.1), and let the grid points be numbered according to

$$k = IJ + 1 - j - (i - 1)J \tag{3.4.16}$$

This ordering is illustrated in Figure 3.4.1, and will be called here the second backward ordering, to distinguish it from the backward ordering defined by (3.3.3). The ordering (3.4.16) will turn out to be preferable in applications to be discussed later. The computation of the corresponding incomplete factorization factors $\bar{\boldsymbol{L}}, \bar{\boldsymbol{D}}$ and $\bar{\boldsymbol{U}}$ is discussed in [141]. If alternating ILU is used, $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ are already stored in the place of $\boldsymbol{A}$, so that additional storage is required for $\bar{\boldsymbol{L}}, \bar{\boldsymbol{D}}$ and $\bar{\boldsymbol{U}}$. $\bar{\boldsymbol{N}}$ can be stored, or is easily computed, as one prefers.

17	13	9	5	1
18	14	10	6	2
19	15	11	7	3
20	16	12	8	4

Figure 3.4.1: Illustration of second backward ordering.

General ILU

Other ILU variant are obtained for other choices of $\mathcal{G}$. See [88] for some possibilities. In general it is advisable to choose $\mathcal{G}$ equal to or slightly larger than the graph of $\boldsymbol{A}$. If $\mathcal{G}$ is smaller that the graph of $\boldsymbol{A}$ then nothing changes in the algorithms just presented, except that the elements of $\boldsymbol{A}$ outside $\mathcal{G}$ are subtracted from $\boldsymbol{N}$.

The following algorithm computes an ILU factorization for general $\mathcal{G}$ by incomplete Gauss elimination. $\boldsymbol{A}$ is an $n \times n$ matrix. We choose $diag\ (\boldsymbol{L}) = diag\ (\boldsymbol{U})$.

Algorithm 1. Incomplete Gauss elimination

$\boldsymbol{A}^0 := \boldsymbol{A}$
for $r := 1$ **step** 1 **until** n **do**
begin $a^r_{rr} := \text{sqrt}\ (a^{r-1}_{rr})$
  for $j > r \wedge (r,j) \in \mathcal{G}$ **do** $a^r_{rj} := a^{r-1}_{rj} / a^r_{rr}$
  for $i > r \wedge (i,r) \in \mathcal{G}$ **do** $a^r_{ir} := a^{r-1}_{ir} / a^r_{rr}$
  for $(i,j) \in \mathcal{G} \wedge i > r \wedge j > r \wedge (i,r) \in \mathcal{G} \wedge (r,j) \in \mathcal{G}$ **do**
  $a^r_{ij} := a^{r-1}_{ij} - a^r_{ir} a^r_{rj}$
  od od od
end of algorithm 1.

$\boldsymbol{A}^n$ contains $\boldsymbol{L}$ and $\boldsymbol{U}$. In [57] one finds an algorithm for the $\boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U}$ version of ILU, for arbitrary $\mathcal{G}$. See [143] and [138] for applications of ILU with a fairly complicated $\mathcal{G}$ (Navier-Stokes equations in the vorticity-stream function formulation).

Final remarks

Existence of ILU factorizations and numerical stability of the associated algorithms has been proved in [87] if $\boldsymbol{A}$ is an M-matrix; it is also shown that the associated splitting is regular, so that ILU converges according to Theorem 4.2.3. For information on efficient implementations of ILU on vector and parallel computers, see [69], [68], [116], [117], [118], [119], [103] and [14].

Exercise 4.4.1 Derive algorithms to compute symmetric ILU factorizations $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{L}^T - \boldsymbol{N}$ and $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{N}$ for $\boldsymbol{A}$ symmetric. See [87].

Exercise 4.4.2 Let $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$, with $\boldsymbol{D} = diag\ (\boldsymbol{A})$, $l_{ij} = 0$, $j > i$ and $u_{ij} = 0$, $j < i$. Show that (3.4.3) results in symmetric point Gauss-Seidel (cf. Exercise 3.3.1). This shows that symmetric point Gauss-Seidel is a special instance of incomplete point factorization.

3.5 Examples of basic iterative methods: incomplete block LU factorization

Complete line LU factorization
The basic idea of *incomplete block LU-factorization* (IBLU) (also called incomplete line LU-factorization (ILLU) in the literature) is presented by means of the following example. Let the stencil of the difference equations to be solved be given by Figure 3.3.3(c). The grid point ordering is given by (3.3.2). Then the matrix $\boldsymbol{A}$ of the system to be solved is as follows:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{U}_1 & & & \\ \boldsymbol{L}_2 & \boldsymbol{B}_2 & \boldsymbol{U}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \boldsymbol{U}_{J-1} \\ & & & \boldsymbol{L}_J & \boldsymbol{B}_J \end{pmatrix} \tag{3.5.1}$$

with $\boldsymbol{L}_j, \boldsymbol{B}_j$ and $\boldsymbol{U}_j$ $I \times I$ tridiagonal matrices.
First, we show that there is a matrix $\boldsymbol{D}$ such that

$$\boldsymbol{A} = (\boldsymbol{L} + \boldsymbol{D})\boldsymbol{D}^{-1}(\boldsymbol{D} + \boldsymbol{U}) \tag{3.5.2}$$

where

$$\begin{aligned} \boldsymbol{L} &= \begin{pmatrix} 0 & & & \\ \boldsymbol{L}_2 & \ddots & & \\ & \ddots & \ddots & \\ & & \boldsymbol{L}_J & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & \boldsymbol{U}_1 & & \\ & 0 & \ddots & \\ & & \ddots & \boldsymbol{U}_{J-1} \\ & & & 0 \end{pmatrix} \\ \boldsymbol{D} &= \begin{pmatrix} \boldsymbol{D}_1 & & & \\ & \boldsymbol{D}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{D}_J \end{pmatrix} \end{aligned} \tag{3.5.3}$$

We call (3.5.2 a *line LU factorization* of $\boldsymbol{A}$, because the blocks in $\boldsymbol{L}, \boldsymbol{D}$ and $\boldsymbol{U}$ correspond to (in our case horizontal) lines in the computational grid. From (3.5.2) it follows that

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U} + \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} \tag{3.5.4}$$

One finds that $\boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U}$ is the following block-diagonal matrix

$$\boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} = \begin{pmatrix} 0 & & & \\ & \boldsymbol{L}_2\boldsymbol{D}_1^{-1}\boldsymbol{U}_1 & & \\ & & \ddots & \\ & & & \boldsymbol{L}_J\boldsymbol{D}_{J-1}^{-1}\boldsymbol{U}_{J-1} \end{pmatrix} \tag{3.5.5}$$

From (3.5.4) and (3.5.5) the following recursion to compute $\boldsymbol{D}$ is obtained

$$\boldsymbol{D}_1 = \boldsymbol{B}_1\ , \quad \boldsymbol{D}_j = \boldsymbol{B}_j - \boldsymbol{L}_j\boldsymbol{D}_{j-1}^{-1}\boldsymbol{U}_j\ , \quad j = 2, 3, ..., J \tag{3.5.6}$$

Provided $\boldsymbol{D}_j^{-1}$ exists, this shows that one can find $\boldsymbol{D}$ such that (3.5.2) holds.

Nine-point IBLU

The matrices $\boldsymbol{D}_j$ are full; therefore incomplete variants of (3.5.2) have been proposed. An incomplete variant is obtained by replacing $\boldsymbol{L}_j\boldsymbol{D}_{j-1}^{-1}\boldsymbol{U}_j$ in (3.5.6) by its tridiagonal part (i.e. replacing all elements with indices i, m with $m \neq i, i \pm 1$ by zero):

$$\tilde{\boldsymbol{D}}_1 = \boldsymbol{B}_1\ , \quad \tilde{\boldsymbol{D}}_j = \boldsymbol{B}_j - tridiag\ (\boldsymbol{L}_j\tilde{\boldsymbol{D}}_{j-1}^{-1}\boldsymbol{U}_j) \tag{3.5.7}$$

The IBLU factorization of $\boldsymbol{A}$ is defined as

$$\boldsymbol{A} = (\boldsymbol{L} + \tilde{\boldsymbol{D}})\tilde{\boldsymbol{D}}^{-1}(\tilde{\boldsymbol{D}} + \boldsymbol{U}) - \boldsymbol{N} \tag{3.5.8}$$

There are three non-zero elements per row in $\boldsymbol{L}$, $\tilde{\boldsymbol{D}}$ and $\boldsymbol{U}$; thus we call this *nine-point IBLU*. For an algorithm to compute $\tilde{\boldsymbol{D}}$ and $\tilde{\boldsymbol{D}}^{-1}$ see [141].

The IBLU iterative method

With IBLU, the basic iterative method (3.1.3) becomes

$$\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{y}^m \tag{3.5.9}$$

$$(\boldsymbol{L} + \tilde{\boldsymbol{D}})\tilde{\boldsymbol{D}}^{-1}(\tilde{\boldsymbol{D}} + \boldsymbol{U})\boldsymbol{y}^{m+1} = \boldsymbol{r} \tag{3.5.10}$$

$$\boldsymbol{y}^{m+1} := \boldsymbol{y}^{m+1} + \boldsymbol{y}^m \tag{3.5.11}$$

Equation (3.5.10) is solved as follows

$$\text{Solve}\ \ (\boldsymbol{L} + \tilde{\boldsymbol{D}})\boldsymbol{y}^{m+1} = \boldsymbol{r} \tag{3.5.12}$$

$$r := \tilde{\boldsymbol{D}}\boldsymbol{y}^{m+1} \tag{3.5.13}$$

$$\text{Solve} \quad (\tilde{\boldsymbol{D}} + \boldsymbol{L})\boldsymbol{y}^{n+1} = r \tag{3.5.14}$$

With the block partioning used before, and with y_j and r_j denoting I-dimensional vectors corresponding to block j, Equation (3.5.12) is solved as follows:

$$\tilde{\boldsymbol{D}}_1 y_1^{n+1} = r_1, \quad \tilde{\boldsymbol{D}}_j y_j^{n+1} = r_j - \boldsymbol{L}_{j-1} y_{j-1}^n, \quad j = 2, 3, ..., J \tag{3.5.15}$$

Equation (3.5.14) is solved in a similar fashion.

Other IBLU variants

Other IBLU variants are obtained by taking other graphs for $\boldsymbol{L}, \tilde{\boldsymbol{D}}$ and $\boldsymbol{U}$. When $\boldsymbol{A}$ corresponds to the five-point stencil of Figure 3.3.3, $\boldsymbol{L}$ and $\boldsymbol{U}$ are diagonal matrices, resulting in the five-point IBLU variants. When $\boldsymbol{A}$ corresponds to the seven-point stencils of Figure 3.3.3(a), (b), $\boldsymbol{L}$ and $\boldsymbol{U}$ are bidiagonal, resulting in seven-point IBLU. There are also other possibilities to approximate $\boldsymbol{L}_j \tilde{\boldsymbol{D}}_{j-1}^{-1} \boldsymbol{U}_j$ by a sparse matrix. See [6], [33], [7], [99], [107] for other versions of IBLU; the first three publications also give existence proofs for $\tilde{\boldsymbol{D}}_j$ if $\boldsymbol{A}$ is an M-matrix; this condition is slightly weakened in [99]. Vectorization and parallelization aspects are discused in [7].

Exercise 3.5.1 Derive an algorithm to compute a symmetric IBLU factorization $\boldsymbol{A} = (\boldsymbol{L} + \tilde{\boldsymbol{D}})\tilde{\boldsymbol{D}}^{-1}(\tilde{\boldsymbol{D}} + \boldsymbol{L}^T) - \boldsymbol{N}$ for $\boldsymbol{A}$ symmetric. See [33].

3.6 Some methods for non-M-matrices

When non-self-adjoint partial differential equations are discretized it my happen that the resulting matrix $\boldsymbol{A}$ is not an M-matrix. This depends on the type of discretization and the values of the coefficients. Examples of other applications leading to non-M-matrix discretizations are the biharmonic equation and the Stokes and Navier-Stokes equations of fluid dynamics.

Defect correction

Defect correction can be used when one has a second-order accurate discretization with a matrix $\boldsymbol{A}$ that is not an M-matrix, and a first-order discretization with a matrix $\boldsymbol{B}$ which is an M-matrix, for example because $\boldsymbol{B}$ is obtained with upwind discretization, or because $\boldsymbol{B}$ contains artificial viscosity. Then one can obtain second-order results as follows.

Algorithm 1. Defect correction

begin Solve $B\bar{y} = b$
  for i := 1 **step** 1 **until** n **do**
    $By = b - A\bar{y} + B\bar{y}$
    $\bar{y} := y$
  od
end of algorithm 1.

It suffices in practice to take $n = 1$ or 2. For simple problems it can be shown that for $n = 1$ already $\boldsymbol{y}$ has second-order accuracy. $\boldsymbol{B}$ is an M-matrix; thus the methods discussed before can be used to solve for $\boldsymbol{y}$.

Distributive iteration

Instead of solving $\boldsymbol{Ay} = \boldsymbol{b}$ one may also solve

$$\boldsymbol{AB\bar{y}} = \boldsymbol{b}, \quad \boldsymbol{y} = \boldsymbol{B\bar{y}} \tag{3.6.1}$$

This may be called *post-conditioning*, in analogy with *preconditioning*, where one solves $\boldsymbol{BAy} = \boldsymbol{Bb}$. $\boldsymbol{B}$ is chosen such that $\boldsymbol{AB}$ is an M-matrix or a small perturbation of an M-matrix, such that the splitting

$$\boldsymbol{AB} = \boldsymbol{M} - \boldsymbol{N} \tag{3.6.2}$$

leads to a convergent iteration method. From (3.6.2) follows the following splitting for the original matrix $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{MB}^{-1} - \boldsymbol{NB}^{-1} \tag{3.6.3}$$

This leads to the following iteration method

$$\boldsymbol{MB}^{-1}\boldsymbol{y}^{m+1} = \boldsymbol{NB}^{-1}\boldsymbol{y}^{m} + \boldsymbol{b} \tag{3.6.4}$$

or

$$\boldsymbol{y}^{m-1} = \boldsymbol{y}^{m} + \boldsymbol{BM}^{-1}(\boldsymbol{b} - \boldsymbol{Ay}^{m}) \tag{3.6.5}$$

The iteration method is based on (3.6.3) rather that on (3.6.2), because if $\boldsymbol{M}$ is modified so that (3.6.2) does not hold, then, obviously, (3.6.5) still converges to the right solution, if it converges. Such modifications of $\boldsymbol{M}$ occur in applications of post-conditioned iteration to the Stokes and Navier-Stokes equations.

Iteration method (3.6.4) is called *distributive iteration*, because the correction $\boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{Ay}^{m})$ is distributed over the elements of $\boldsymbol{y}$ by the matrix $\boldsymbol{B}$. A general treatment of this approach is given in [144], [146], [148], [150], [149], where it is shown that a number of well known iterative methods for the Stokes and Navier-Stokes equations can be interpreted as distributive iteration methods.

Taking $\boldsymbol{B} = \boldsymbol{A}^T$ and choosing (3.6.2) to be the Gauss-Seidel or Jacobi splitting results in the Kaczmarz [78] or Cimmino [32] methods, respectively. These methods converge for every regular A, because Gauss-Seidel and Jacobi converge for symmetric positive definite matrices (a proof of this elementary result may be found in [70]. Convergence is, however, usually slow.

4 Smoothing analysis

4.1 Introduction

The convergence behaviour of a multigrid algorithm depends strongly on he smoother. The efficiency of smoothing methods is problem-dependent. When a smoother is efficient for a large class of problems it is called *robust*. This concept will be made more precise shortly for a certain class of problems. Not every convergent method has the smoothing property, but for symmetric matrices it can be shown that by the introduction of suitable amount of damping every convergent method acquires the smoothing property. This property says little about the actual efficiency. A convenient tool for the study of smoothing efficiency is Fourier analysis, which is also easily applied to the non-symmetric case. Fourier smoothing analysis is the main topic of this chapter.

Many different smoothing methods are employed by users of multigrid methods. Of course, in order to explain the basic principles of smoothing analysis it suffices to discuss only a few methods by way of illustration. To facilitate the making of a good choice of a smoothing method for a particular application it is, however, useful to gather smoothing analysis results which are scattered through the literature in one place, and to complete the information where results for important cases are lacking.

4.2 The smoothing property

The smoothing method is assumed to be a basic iterative method as defined by (3.1.3). We will assume that $\boldsymbol{A}$ is a K-matrix. Often, the smoother is obtained in the way described in Theorem 3.2.5; in practice one rarely encounters anything else.

The *smoothing property* is defined as follows ([57]):

Definition 4.2.1 Smoothing property. $\boldsymbol{S}$ has the smoothing property if there exist a constant C_S and a function $\eta(\nu)$ independent of the mesh-size h such that

$$\|\boldsymbol{A}\boldsymbol{S}^{\nu}\| \leq C_S h^{-2m} \eta(\nu), \quad \eta(\nu) \to 0 \quad \text{for} \quad \nu \to \infty \tag{4.2.1}$$

where $2m$ is the order of the partial differential equation to be solved.

Here $\boldsymbol{S}$ is the iteration matrix defined by (3.1.5). The smoothing property implies converse [141] but, as already remarked, the converse is not true. In [145] it is shown that a convergent method can be turned into a smoother by damping; for a fuller discussion see [141].

Discussion
In [57] the smoothing property is shown for a number of iterative methods. The smoothing property of incomplete factorization methods is studied in [145], [147]. Non-symmetric problems can be handled by perturbation arguments, as indicated by [57]. When the non-symmetric part is dominant, however, as in singular perturbation problems, this does not lead to useful results. Fourier smoothing analysis (which, however, also has its limitations) can handle the non-symmetric case easily, and also provides an easy way to optimize values of damping parameters and to predict smoothing efficiency. The introduction of damping does not necessarily give a robust smoother. The differential equation may contain a parameter, such that when it tends to a certain limit, smoothing efficiency deteriorates. Examples and further discussion of robustness will follow. We will concentrate on Fourier smoothing analysis.

4.3 Elements of Fourier analysis in grid-function space

As preparation we start with the one-dimensional case.

The one-dimensional case

Theorem 4.3.1. Discrete Fourier transform. Let $I = \{0, 1, 2, ..., n-1\}$. Every $\boldsymbol{u} : I \rightarrow I\!R$ can be written as

$$\boldsymbol{u} = \sum_{k=-m}^{m+p} c_k \psi(\theta_k), \quad \psi_j(\theta_k) = exp(ij\theta_k), \quad \theta_k = 2\pi k/n, \quad j \in I \tag{4.3.1}$$

where $p = 0,\ m = (n-1)/2$ for n odd and $p = 1,\ m = n/2 - 1$ for n even, and

$$c_k = n^{-1} \sum_{j=0}^{n-1} u_j \psi_j(-\theta_k) \tag{4.3.2}$$

The functions $\psi(\theta)$ are called Fourier modes or Fourier components. For a proof of this elementary theorem see [141].

The multi-dimensional case
Define

$$\psi_j(\theta) = exp(ij\theta) \tag{4.3.3}$$

with $j \in I$, $\theta \in \Theta$, with

$$I = \{j : j = (j_1, j_2, ..., j_d),\ j_\alpha = 0, 1, 2, ..., n_\alpha - 1,\ \alpha = 1, 2, ..., d\} \tag{4.3.4}$$

$$\Theta = \{\theta : \theta = (\theta_1, \theta_2, ..., \theta_d),\ \theta_\alpha = 2\pi k_\alpha / n_\alpha,$$

$$k_\alpha = -m_\alpha, -m_\alpha + 1, ..., m_\alpha + p_\alpha,\ \alpha = 1, 2, ..., d\} \tag{4.3.5}$$

where $p_\alpha = 0$, $m_\alpha = (n_\alpha - 1)/2$ for n_α odd and $p_\alpha = 1$, $m_\alpha = n_\alpha/2 - 1$ for n_α even. Furthermore,

$$j\theta = \sum_{\alpha=1}^{d} j_\alpha \theta_\alpha \tag{4.3.6}$$

Theorem 4.3.2. Discrete Fourier transform in d dimensions. Every $\boldsymbol{u} : \boldsymbol{I} \to I\!\!R$ can be written as

$$u_j = \sum_{\theta \in \Theta} c_\theta \psi_j(\theta) \tag{4.3.7}$$

with

$$c_\theta = N^{-1} \sum_{j \in I} u_j \psi_j(-\theta),\ N = \prod_{\alpha=1}^{d} n_\alpha \tag{4.3.8}$$

For a proof see [141].

The Fourier series (4.3.7) is appropriate for d-dimensional vertex- or cell-centered grids with periodic boundary conditions. For the use of Fourier sine or cosine series to accommodate Dirichlet or Neumann conditions, see [141].

4.4 The Fourier smoothing factor

Definition of the local mode smoothing factor
Let the problem to be solved on grid G be denoted by

$$\boldsymbol{Au} = \boldsymbol{f} \tag{4.4.1}$$

and let the smoothing method to be used be given by (3.1.6):

$$\boldsymbol{u} := \boldsymbol{Su} + \boldsymbol{M}^{-1}\boldsymbol{f}, \quad \boldsymbol{S} = \boldsymbol{M}^{-1}\boldsymbol{N}, \quad \boldsymbol{M} - \boldsymbol{N} = \boldsymbol{A} \tag{4.4.2}$$

According to (3.2.1) the relation between the error before and after ν smoothing iterations is

$$\boldsymbol{e}^1 = \boldsymbol{S}^\nu \boldsymbol{e}^0 \tag{4.4.3}$$

We now make the following assumption.

Assumption (i). The operator $\boldsymbol{S}$ has a complete set of eigenfunctions or *local modes* denoted by $\psi(\theta)$, $\theta \in \Theta$, with Θ some discrete index set.

Hence

$$\boldsymbol{S}^{\nu}\psi(\theta) = \lambda^{\nu}(\theta)\psi(\theta) \tag{4.4.4}$$

with $\lambda(\theta)$ the eigenvalue belonging to $\psi(\theta)$. we can write

$$\boldsymbol{e}^{\alpha} = \sum_{\theta\in\Theta} c_0^{\alpha}\psi(\theta), \quad \alpha = 0, 1$$

and obtain

$$c_{\theta}^{1} = \lambda^{\nu}(\theta)c_{\theta}^{0} \tag{4.4.5}$$

The eigenvalue $\lambda(\theta)$ is also called the amplification factor of the local mode $\psi(\theta)$.
Next, assume that among the eigenfunctions $\psi(\theta)$ we somehow distinguish between *smooth* eigenfunctions ($\theta \in \Theta_s$) and *rough* eigenfunctions ($\theta \in \Theta_r$):

$$\Theta = \Theta_s \cup \Theta_r, \quad \Theta_s \cap \Theta_r = \emptyset \tag{4.4.6}$$

We now make the following definition.

Definition 4.4.1. Local mode smoothing factor. The local mode smoothing factor ρ of the smoothing method (4.4.2) is defined by

$$\rho = sup\{|\lambda(\theta)| : \theta \in \Theta_r\} \tag{4.4.7}$$

Hence, after ν smoothings the amplitude of the rough components of the error are multiplied by a factor ρ^{ν} or smaller.

Fourier smoothing analysis
In order to obtain from this analysis a useful tool for examining the quality of smoothing methods we must be able to easily determine ρ, and to choose Θ_s such that an error $\boldsymbol{e} = \psi(\theta)$, $\theta \in \Theta_s$ is well reduced by coarse grid correction. This can be done if Assumption (ii) is satisfied.

Assumption (ii). The eigenfunctions $\psi(\theta)$ of $\boldsymbol{S}$ are harmonic functions.

This assumption means that the series preceding (4.4.5) is a Fourier series. When this is so ρ is also called the *Fourier smoothing factor*. In the next section we will give conditions such that Assumption (ii) holds, and show how ρ is easily determined; but first we discuss the choice of Θ_r.

Aliasing
Consider the vertex-centered grid G given by (4.4.8) with n_α even, and the corresponding coarse grid $\bar{G}$ defined by doubling the mesh-size:

$$\begin{aligned} G = \{x \in I\!R^d : x = jh,\ j &= (j_1, j_2, ..., j_d),\ h = (h_1, h_2, ..., h_d), \\ & j_\alpha = 0, 1, 2, ..., n_\alpha,\ h_\alpha = 1/n_\alpha,\ \alpha = 1, 2, ..., d\} \end{aligned} \quad (4.4.8)$$

$$\begin{aligned} \bar{G} = \{x \in I\!R^d : x = j\bar{h},\ j &= (j_1, j_2, ..., j_d),\ \bar{h} = (\bar{h}_1, \bar{h}_2, ..., \bar{h}_d), \\ & j_\alpha = 0, 1, 2, ..., \bar{n}_\alpha,\ \bar{h}_\alpha = 1/\bar{n}_\alpha,\ \alpha = 1, 2, ..., d\} \end{aligned} \quad (4.4.9)$$

with $\bar{n}_\alpha = n_\alpha/2$. Let $d = 1$, and assume that the eigenfunctions of $\boldsymbol{S}$ on the fine grid G are the Fourier modes of Theorem 4.3.1: $\psi_j(\theta) = \exp(ij\theta)$, with

$$\theta \in \Theta = \{\theta : \Theta = 2\pi k/n_1,\ k = -n_1/2 + 1,\ -n_1/2 + 2, ..., n_1/2\} \quad (4.4.10)$$

so that an arbitrary grid function v on G can be represented by the following Fourier series

$$v_j = \sum_{\theta \in \Theta} c_\theta \psi_j(\theta) \quad (4.4.11)$$

An arbitrary grid function $\bar{\boldsymbol{v}}$ on $\bar{G}$ can be represented by

$$\bar{v}_j = \sum_{\bar{\theta} \in \bar{\Theta}} \bar{c}_\theta \bar{\psi}_j(\bar{\theta}) \quad (4.4.12)$$

with $\bar{\psi}(\bar{\theta}) : \bar{G} \to I\!R,\ \bar{\psi}_j(\bar{\theta}) = exp(ij\bar{\theta})$, and

$$\bar{\Theta} = \{\bar{\theta} : \bar{\theta} = 2\pi k/\bar{n}_1,\ k = -\bar{n}_1/2 + 1,\ -\bar{n}_1/2 + 2, ..., \bar{n}_1/2\} \quad (4.4.13)$$

assuming for simplicity that $\bar{n}_1$ is even. The coarse grid point $\bar{x}_j = j\bar{h}$ coincides with the fine grid point $x_{2j} = 2jh$. In these points the coarse grid Fourier mode $\bar{\psi}(\bar{\theta})$ takes on the value

$$\bar{\psi}_j(\bar{\theta}) = exp(ij\bar{\theta}) = exp(i2j\theta) \quad (4.4.14)$$

For $-n_1/4 + 1 \le k \le n_1/4$ the fine grid Fourier mode $\psi(\theta_k)$ takes on in the coarse grid points x_j the values of $\psi_{2j}(\theta_k) = exp(2\pi ijk/\bar{n}_1) = \bar{\psi}_j(2\pi k/\bar{n}_1)$, and we see that it coincides with the coarse grid mode $\bar{\psi}(\theta_k)$ in the coarse grid points. But this is also the case for another fine grid mode. Define k' as follows

$$\begin{aligned} 0 < k \le \bar{n}_1/2 : &\quad k' = -n_1/2 + k \\ -\bar{n}_1/2 < k \le 0 : &\quad k' = n_1/2 + k \end{aligned} \quad (4.4.15)$$

Then the fine grid Fourier mode $\psi(\theta_{k'})$ also coincides with $\bar{\psi}(\theta_k)$ in the coarse grid points. On the coarse grid, $\psi(\theta_{k'})$ cannot be distinguished from $\psi(\theta_k)$. This is called *aliasing*: the rapidly varying function $\psi(\theta_{k'})$ takes on the appearance of the much smoother function $\psi(\theta_k)$ on the coarse grid.

Smooth and rough Fourier modes

Because on the coarse grid $\bar{G}$ the rapidly varying function $\psi(\theta_{k'})$ cannot be approximated, and cannot be distinguished from $\psi(\theta_k)$, where is no hope that the part of the error which consists of Fourier modes $\psi(\theta_{k'})$, k' given by (4.4.15), can be approximated on the coarse grid $\bar{G}$. This part of the error is called *rough* or *non-smooth*. The rough Fourier modes are defined to be $\psi(\theta_{k'})$, with k' given by (4.4.15), that is

$$k' \in \{-n_1/2+1,\ -n_1/2+2, ..., -n_1/4\} \cup \{n_1/4,\ n_1/4+1, ..., n_1/2\} \tag{4.4.16}$$

This gives us the set of rough wavenumbers $\Theta_r = \{\theta : \theta = 2\pi k'/n_1 : k'$ according to (4.4.16)$\}$, or

$$\begin{aligned}\Theta_r = \{\theta : \theta = 2\pi k/n_1,\ k = -n_1/2+1,\ -n_1/2+2, ..., n_1/2 \\ \text{and}\ \ \theta \in [-\pi, -\pi/2] \cup [\pi/2, \pi]\}\end{aligned} \tag{4.4.17}$$

The set of smooth wavenumbers Θ_s is defined as $\Theta_s = \Theta \setminus \Theta_r$, Θ given by (4.4.10) with $d = 1$, or

$$\begin{aligned}\Theta_s = \{\theta : \theta = 2\pi k/n_1,\ k = -n_1/2+1,\ -n_1/2+2, ..., n_1/2 \\ \text{and}\ \ \theta \in (-\pi/2,\ \pi/2)\}\end{aligned} \tag{4.4.18}$$

The smooth and rough parts $\boldsymbol{v}_s$ and $\boldsymbol{v}_r$ of a grid function $\boldsymbol{v} : G \to \mathbb{R}$ can now be defined precisely by

$$\begin{gathered}\boldsymbol{v}_s = \sum_{\theta \in \Theta_s} c_\theta \psi(\theta), \quad \boldsymbol{v}_r = \sum_{\theta \in \Theta_r} c_\theta \psi(\theta) \\ c_\theta = n_1^{-1} \sum_{j=0}^{n-1} v_j \psi_j(-\theta)\end{gathered} \tag{4.4.19}$$

In d dimensions the generalization of (4.4.17) and (4.4.18) (periodic boundary conditions) is

$$\begin{gathered}\Theta = \{\theta : \theta = (\theta_1, \theta_2, ..., \theta_d), \quad \theta_\alpha = 2\pi k_\alpha/n_\alpha, \quad k_\alpha = -n_\alpha/2+1, ..., n_\alpha/2\} \\ \Theta_s = \Theta \cap \prod_{\alpha=1}^{d} (-\pi/2, \pi/2), \quad \Theta_r = \Theta \setminus \Theta_s\end{gathered} \tag{4.4.20}$$

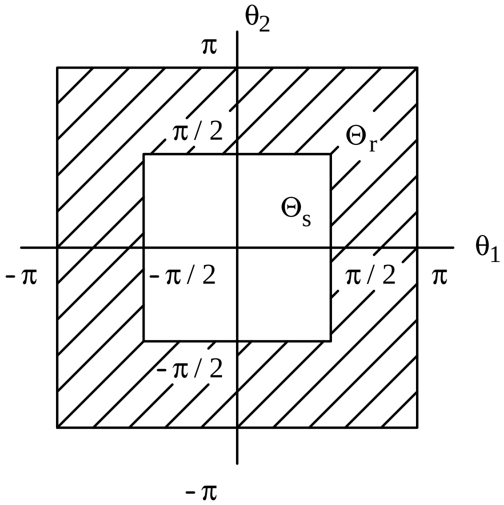


Figure 4.4.1: Smooth (Θ_s) and rough (Θ_r, hatched) wavenumber sets in two dimensions, standard coarsering.

Figure 4.4.1 gives a graphical illustration of the smooth and rough wavenumber sets (4.4.20) for $d = 2$. Θ_r and Θ_s are discrete sets in the two concentric squares. As the mesh-size is decreased (n_α is increased) these discrete sets become more densely distributed.

Semi-coarsening

The above definition of Θ_r and Θ_s in two dimensions is appropriate for *standard coarsening*, i.e. $\bar{G}$ is obtained from G by doubling the mesh-size h_α in all directions $\alpha = 1, 2, ..., d$.
With *semi-coarsering* there is at least one direction in which h_α in $\bar{G}$ is the same as in G. Of course, in this direction no aliasing occurs, and all Fourier modes on G in this direction can be resolved on $\bar{G}$, so hey are not included in Θ_r. To give an example in two dimensions, assume $\bar{h}_1 = h_1$ (semi-coarsering in the x_2-direction). Then (4.4.20) is replaced by

$$\Theta_s = \Theta \cap \{[-\pi, \pi] \times (-\pi/2, \pi/2)\}, \quad \Theta_s = \Theta \setminus \Theta_s \tag{4.4.21}$$

Figure 4.4.2 gives a graphical illustration.

Mesh-size independent definition of smoothing factor

We have a smoothing method on the grid G if uniformly in n_α there exists a ρ^* such that

$$\rho \leq \rho^* < 1, \quad \forall n_\alpha, \quad \alpha = 1, 2, ..., d \tag{4.4.22}$$

However, ρ as defined by (4.4.7) depends on n_α, because Θ_r depends on n_α. In order to obtain a mesh-independent condition which implies (4.4.23) we define a set $\bar{\Theta}_r \supset \Theta_r$ with $\bar{\Theta}_r$

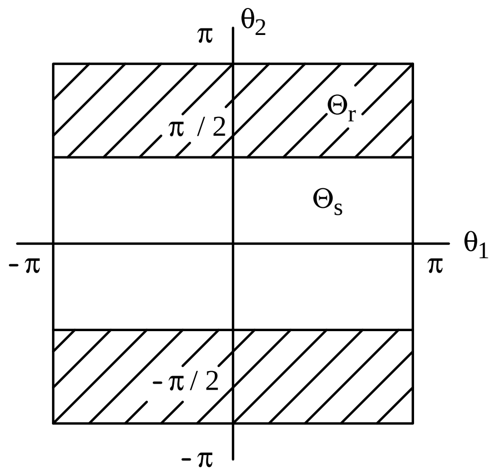


Figure 4.4.2: Smooth (Θ_s) and rough (Θ_r, hatched) wavenumber sets in two dimensions, semi-coarsening in x_2 direction.

independent of n_α and define

$$\bar{\rho} = sup\{|\lambda(\theta)| : \theta \in \bar{\Theta}_r\} \tag{4.4.23}$$

so that

$$\rho \leq \bar{\rho} \tag{4.4.24}$$

and we have a smoothing method if $\bar{\rho} < 1$. For example, if Θ_r is defined by (4.4.20), then we may define $\bar{\Theta}_r$ as follows:

$$\bar{\Theta}_r = \prod_{\alpha=1}^{d}[-\pi, \pi] \setminus \prod_{\alpha=1}^{d}(-\pi/2, \pi/2) \tag{4.4.25}$$

This type of Fourier analysis, and definition (4.4.23) of the smoothing factor, have been introduced by Brandt (1977). It may happen that $\lambda(\theta)$ still depends on the mesh-size, in which case $\bar{\rho}$ is not really independent of the mesh-size, of course.

Modification of smoothing factor for Dirichlet boundary conditions

If $\lambda(\theta)$ is smooth, then $\bar{\rho} - \rho = O(h_\alpha^m)$ for some $m \geq 1$. It may, however, happen that there is a parameter in the differential equation, say ε, such that for example $\bar{\rho} - \rho = O(h_\alpha^2/\varepsilon)$. Then, for $\varepsilon \ll 1$ (singular perturbation problems), for practical values of h_α there may be a large difference between ρ and $\bar{\rho}$. Even if $\bar{\rho} = 1$, one may still have a good smoother. Large discrepancies between predictions based on $\bar{\rho}$ and practical observations may occur for singular perturbation problems when the boundary conditions are not periodic. It turns out that discrepancies due to the fact that the boundary conditions are not of the assumed type arise mainly from the presence or absence of wavenumber components $\theta_\alpha = 0$ (present with periodic boundary conditions, absent with Dirichlet boundary conditions). It has been observed [29], [83], [147] that when using the exponential Fourier series (4.3.7) for smoothing analysis of a practical case with Dirichlet boundary conditions, often better agreement with practical results is obtained by leaving wavenumbers with $\theta_\alpha = 0$ out, changing the definition of Θ_r in (4.4.7) from (4.4.20) to

$$\Theta^D = \{\theta : \theta = (\theta_1, \theta_2, ..., \theta_d), \quad \theta_\alpha = 2\pi k_\alpha / n_\alpha, \quad k_\alpha \neq 0, \quad k_\alpha = -n_\alpha/2 + 1, ..., n_\alpha/2\}$$

$$\Theta_s^D = \Theta^D \cap \prod_{\alpha=1}^{d} (-\pi/2,\ \pi/2), \quad \Theta_r^D = \Theta^D \setminus \Theta_s^D \tag{4.4.26}$$

where the superscript D serves to indicate the case of Dirichlet boundary conditions. The smoothing factor is now defined as

$$\rho_D = sup\{|\lambda(\theta)| : \theta \in \Theta_r^D\} \tag{4.4.27}$$

Figure 4.4.3 gives an illustration of Θ_r^D, which is a discrete set within the hatched region, for $d = 2$. Further support for the usefulness of definitions (4.4.26) and (4.4.27) will be given in the next section.
Notice that we have the following inequality

$$\rho_D \leq \rho \leq \bar{\rho} \tag{4.4.28}$$

If we have a Neumann boundary condition at both $x_\alpha = 0$ and $x_\alpha = 1$, then $\theta_\alpha = 0$ cannot be excluded, but if one has for example Dirichlet at $x_\alpha = 0$ and Neumann at $x_\alpha = 1$ then the error cannot contain a constant mode in the x_α direction, and $\theta_\alpha = 0$ can again be excluded.

Exercise 4.4.1 Suppose $\bar{h}_1 = \mu h_1$ ($\bar{h}_1$: mesh-size of $\bar{G}, h_1$: mesh-size of G, one-dimensional case, μ some integer), and assume periodic boundary conditions. Show that we have aliasing for

$$\theta_k = 2\pi k/n_1, \quad k \in \mathbb{Z} \cap \{(-n_1/2, -n_1/2\mu] \cup [n_1/2\mu, n_1/2]\}$$

and define sets Θ_r, Θ_s.

4.5 Fourier smoothing analysis

Explicit expression for the amplification factor
In order to determine the smoothing factor $\rho, \bar{\rho}$ or ρ_D according to definitions (4.4.7), (4.4.23) and (4.4.27) we have to solve the eigenvalue problem $\boldsymbol{S}\psi(\theta) = \lambda(\theta)\psi(\theta)$ with $\boldsymbol{S}$ given by

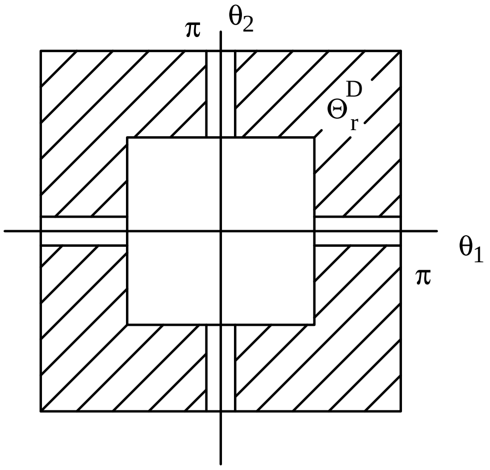


Figure 4.4.3: Rough wavenumber set (Θ_r^D, hatched) in two dimensions, with exclusion of $\theta_\alpha = 0$ modes; standard coarsening.

(4.4.2). Hence, we have to solve $\boldsymbol{N}\psi(\theta) = \lambda(\theta)\boldsymbol{M}\psi(\theta)$. In stencil notation (to be more fully discussed later) this becomes, in d dimensions,

$$\sum_{j\in\mathbb{Z}^d} \boldsymbol{N}(m,j)\psi_{m+j}(\theta) = \lambda(\theta) \sum_{j\in\mathbb{Z}^d} \boldsymbol{M}(m,j)\psi_{m+j}(\theta), \quad m \in \mathbb{Z}^d \tag{4.5.1}$$

with $\mathbb{Z} \equiv \{0, \pm 1, \pm 2, ...\}$.
We now assume the following.

Assumption (i). $\boldsymbol{M}(m,j)$ and $\boldsymbol{N}(m,j)$ do not depend on m.

This assumption is satisfied if the coefficients in the partial differential equation to be solved are constant, the mesh-size of G is uniform and the boundary conditions are periodic. We write $\boldsymbol{M}(j), \boldsymbol{N}(j)$ instead of $\boldsymbol{M}(m,j), \boldsymbol{N}(m,j)$. As a consequence of Assumption (i), Assumption (ii) of Section 4.4 is satisfied: the eigenfunctions of $\boldsymbol{S}$ are given by (4.3.3), since

$$\sum_{j\in\mathbb{Z}^d} \boldsymbol{N}(j) exp\{i(j+m)\theta] = exp(im\theta) \sum_{j\in\mathbb{Z}^d} \boldsymbol{N}(j) exp(ij\theta)$$

so that $\psi_m(\theta) = exp(im\theta)$ satisfies (4.5.1) with

$$\lambda(\theta) = \sum_{j\in\mathbb{Z}^d} \boldsymbol{N}(j) exp(ij\theta) / \sum_{j\in\mathbb{Z}^d} \boldsymbol{M}(j) exp(ij\theta) \tag{4.5.2}$$

Periodicity requieres that $\exp(im_\alpha\theta_\alpha) = \exp[i(m_\alpha + n_\alpha)\theta_\alpha]$, or $exp(in_\alpha\theta_\alpha) = 1$. Hence $\theta \in \Theta$, as defined by (4.3.5), assuming n_α to be even. Hence, the eigenfunctions are the Fourier modes of Theorem 4.3.2.

Variable coefficients, robustness of smoother
In general the coefficients of the partial differential equation to be solved will be variable, of course. Hence Assumption (i) will not be satisfied. The assumption of uniform mesh-size is less demanding, because ofen the computational grid G is a *boundary fitted* grid, obtained by a mapping from the physical space and is constructed such that G is rectangular and has uniform mesh size. This facilitates the implementation of the boundary conditions and of a multigrid code. For the purpose of Fourier smoothing analysis the coefficients $M(m,j)$ and $N(m,j)$ are locally 'frozen'. We may expect to have a good smoother if $\bar{\rho} < 1$ for all values $M(j), N(j)$ that occur. This is supported by theoretical arguments advanced in [57], Section 8.2.2.

A smoother is called *robust* if it works for a large class of problems. Robustness is a quantitative property, which can be defined more precisely once a set of suitable test problems has been defined.

Test problems
In order to investigate and compare efficiency and robustness of smoothing methods the following two special cases in two dimensions are useful

$$-(\varepsilon c^2 + s^2)u_{,11} - 2(\varepsilon - 1)csu_{,12} - (\varepsilon s^2 + c^2)u_{,22} = 0 \tag{4.5.3}$$

$$-\varepsilon(u_{,11} + u_{,22}) + cu_{,1} + su_{,2} = 0 \tag{4.5.4}$$

with $c = \cos\beta,\ s = \sin\beta$. There are two constant parameters to be varied: $\varepsilon > 0$ and β. Equation (4.5.3) is called the *rotated anisotropic diffusion equation*, because it is obtained by a rotation of the coordinate axes over an angle β from the anisotropic diffusion equation:

$$\varepsilon u_{,11} - u_{,22} = s \tag{4.5.5}$$

Equation (4.5.3) models not only anisotropic diffusion, but also variation of mesh aspect ratio, because with $\beta = 0, \varepsilon = 1$ and mesh aspect ration $h_1/h_2 = \delta^{-1/2}$ discretization results in the same stencil as with $\varepsilon = \delta,\ h_1/h_2 = 1$ apart from a scale factor. With $\beta \neq k\pi/2,\ k = 0,1,2,3,$ (4.5.3) also brings in a mixed derivative, which may arise in practice because of the use of non-orthogonal boundary-fitted coordinates. Equation (4.5.4) is the *convection-diffusion* equation. It is not self-adjoint. For $\varepsilon \ll 1$ it is a singular perturbation problem, and is almost hyperbolic. Hyperbolic, almost hyperbolic and convection-dominated problems are common in fluid dynamics.

Equations (4.5.3) and (4.5.4) are not only useful for testing smoothing methods, but also for testing complete multigrid algorithms. General (as opposed to Fourier analysis) multigrid convergence theory is not uniform in the coefficients of the differential equation, and the theoretical rate of convergence is not bounded away from 1 as $\varepsilon \downarrow 0$ or $\varepsilon \to \infty$. In the absence of theoretical justification, one has to resort to numerical experiments to validate a method, and equations (4.5.3) and (4.5.4) constitute a set of discriminating test problems.

Finite difference discretization results in the following stencil for (4.5.3), assuming $h_1 = h_2 = h$ and multiplying by h^2:

$$\begin{aligned}[\boldsymbol{A}] &= (\varepsilon c^2 + s^2)[-1 \;\; 2 \;\; -1] \\ &+ \;\; (\varepsilon - 1)cs \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} + (\varepsilon s^2 + c^2) \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}\end{aligned} \tag{4.5.6}$$

The matrix corresponding to this stencil is not a K-matrix (see Definition 3.2.6) if $\varepsilon - 1)cs > 0$. If that is the case one can replace the stencil for the mixed derivative by

$$\begin{bmatrix} 0 & 1 & -1 \\ 1 & -2 & 1 \\ -1 & 1 & 0 \end{bmatrix} \tag{4.5.7}$$

We will not, however use (4.5.7) in what follows.

A more symmetric stencil for $[\boldsymbol{A}]$ is obtained if the mixed derivative is approximated by the average of the stencil employed in (4.5.6) and (4.5.7), namely

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.5.8}$$

Note that for $[\boldsymbol{A}]$ in (4.5.6) to correspond to a K-matrix it is also necessary that

$$\varepsilon c^2 + s^2 + (\varepsilon - 1)cs \geq 0 \quad \text{and} \quad \varepsilon s^2 + c^2 + (\varepsilon - 1)cs \geq 0 \tag{4.5.9}$$

This condition will be violated if ε differs enough from 1 for certain values of $c = \cos\beta$, $s = \sin\beta$. With (4.5.8) there are always (if $(\varepsilon - 1)cs \neq 0$) positive off-diagonal elements, so that we never have a K-matrix. On the other hand, the 'wrong' elements are a factor 1/2 smaller than with the other two options. Smoothing analysis will show which of these variants lend themselves most for multigrid solution methods.

Finite difference discretization results in the following stencil for (4.5.4), with $h_1 = h_2 = h$ and multiplying by h^2:

$$[\boldsymbol{A}] = \varepsilon \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + c\frac{h}{2}[-1 \;\; 0 \;\; 1] + s\frac{h}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{4.5.10}$$

In (4.5.10) central differences have been used to discretize the convection terms in (4.5.4). With *upwind differences* we obtain

$$\begin{aligned}[\boldsymbol{A}] = \varepsilon & \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + \frac{h}{2}[-c-|c| \;\; 2|c| \;\; c-|c|] \\ + \; & \frac{h}{2}\begin{bmatrix} s-|s| \\ 2|s| \\ -s-|s| \end{bmatrix}\end{aligned} \tag{4.5.11}$$

Stencil (4.5.10) gives a K-matrix only if the well known conditions on the mesh Péclet numbers are fulfilled:

$$|c|h/\varepsilon < 2, \quad |s|h/\varepsilon < 2 \tag{4.5.12}$$

Stencil (4.5.11) always results in a K-matrix, which is the main motivation for using upwind differences. Often, in applications (for example, fluid dynamics) conditions (4.5.12) are violated, and discretization (4.5.10) is hard to handle with multigrid methods; therefore discretization (4.5.11) will mainly be considered.

Definition of robustness

We can now define robustness more precisely: a smoothing methods is called *robust* if, for the above test problems, $\rho \leq \rho^* < 1$ or $\rho_D \leq \rho^* < 1$ with ρ^* independent of ε and h, for some $h_0 \geq h > 0$.

Numerical calculation of Fourier smoothing factor

Using the explicit expression (4.5.2) for $\lambda(\theta)$, it is not difficult to compute $|\lambda(\theta)|$, and to find its largest value on the discrete set Θ_r or Θ_r^D and hence the Fourier smoothing factors ρ or ρ_D. By choosing in the definition of Θ_r (for example (4.4.20) or (4.4.21) various values of n_α one may gather numerical evidence that (4.4.22) is satisfied. Computation of the mesh-independent smoothing factor $\bar{\rho}$ defined in (4.4.23) is more difficult numerically, since this involves finding a maximum on an infinite set. In simple cases $\bar{\rho}$ can be found analytically, as we shall see shortly. Extrema of $|\lambda(\theta)|$ or Θ_r are found where $\partial|\lambda(\theta)|/\partial\theta_\alpha = 0, \alpha = 1, 2, ..., d$ and at the boundary of Θ_r. Of course, for a specific application one can compute ρ for the values of n_α occurring in this application, without worring about the limit $n_\alpha \to \infty$. In the following, we often present results for $n_1 = n_2 = n = 64$. It is found that the smoothing factors ρ, ρ_D do not change much if n is increased beyond 64, except in those cases where ρ and ρ_D differ appreciably. An analysis will be given of what happens in those cases.

All smoothing methods to be discussed in this chapter have been defined in Section 3.3 to 3.5.

Local smoothing

Local freezing of the coefficients is not realistic near points where the coefficients are not smooth. Such points may occur if the computational grid has been obtained as a boundary fitted coordinate mapping of physical domain with non-smooth boundary. Near points on the boundary which are the images of the points where the physical domain boundary is not smooth, and where the mapping is singular, the smoothing performance often deteriorates. This effect may be counterbalanced by performing additional local smoothing in a few grid points in a neighbourhood of these singular points. Because only a few points are involved, the additional cost is usually low, apart from considerations of vector and parallel computing. This procedure is described in [23] and [9] and analysed theoretically in [110] and [24].

4.6 Jacobi smoothing

Anisotropic diffusion equation

Point Jacobi

Point Jacobi with damping corresponds to the following splitting (cf. Exercise 3.1.1), in stencil notation:

$$M(0) = \omega^{-1}A(0), \quad M(j) = 0,\ j \neq 0 \tag{4.6.1}$$

Assuming periodic boundary conditions we obtain, using (4.5.9) and (4.5.2), in the special case $c = 1,\ s = 0$

$$\lambda(\theta) = 1 + \omega(\varepsilon \cos\ \theta_1 - \varepsilon + \cos\ \theta_2 - 1)/(1 + \varepsilon) \tag{4.6.2}$$

Because of symmetry Θ_r can be confined to the hatched region of Figure 4.6.1. Clearly, $\bar{\rho} \geq |\rho(\pi,\pi)| = |1-2\omega| \geq 1$ for $\omega \notin (0,1)$. For $\omega \in (0,1)$ we have for $\theta \in CDEF: \ \lambda(\pi,\pi) \leq$

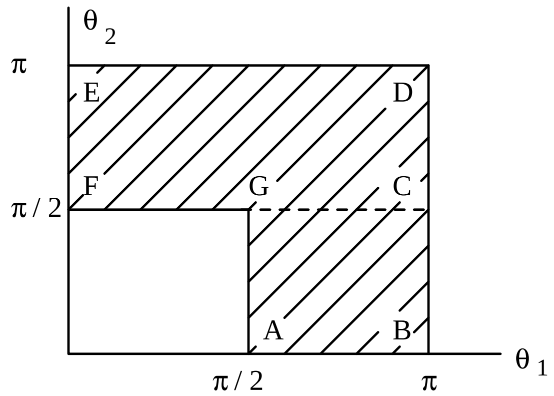


Figure 4.6.1: Rough wavenumbers for damped Jacobi.

$\lambda(0,\pi/2)$, or $1-2\omega \leq \lambda(\theta) \leq 1-\omega/(1+\varepsilon)$. For $\theta \in ABCG$ we have

$$\begin{aligned} &\lambda(\pi,\pi/2) \leq \lambda(\theta) \leq \lambda(\pi/2,0), \\ &\quad \text{or } 1-[(1+2\varepsilon)/(1+\varepsilon)]\omega \leq \lambda(\theta) \leq 1-[\varepsilon/(1+\varepsilon)]\omega \ . \end{aligned}$$

Hence

$$\bar{\rho} = \max\{|1-2\omega|, \ |1-\frac{\omega}{1+\varepsilon}|, \ |1-\frac{1+2\varepsilon}{1+\varepsilon}\omega|, \ |1-\frac{\varepsilon}{1+\varepsilon}\omega|\} \tag{4.6.3}$$

$$\bar{\rho} = (2+\varepsilon)/(2+3\varepsilon) \ , \quad \omega = (2+2\varepsilon)/(2+3\varepsilon) \tag{4.6.4}$$

For $\varepsilon = 1$ (Laplace's equation) we have $\bar{\rho} = 3/5$, $\omega = 4/5$. For $\varepsilon \ll 1$ this is not a good smoother, since $\lim_{\varepsilon \downarrow 0} \bar{\rho} = 1$. The case $\varepsilon > 1$ follows from the case $\varepsilon \leq 1$ by replasing ε by $1/\varepsilon$. Note that $\bar{\rho}$ is attained for $\theta \in \Theta_r$, so that here

$$\rho = \bar{\rho} \tag{4.6.5}$$

For $\omega = 1$ we have $\bar{\rho} = 1$, so that we have an example of a convergent method which is not a smoother.

Dirichlet boundary conditions

In the case of point Jacobi smoothing the Fourier sine series is applicable (see [141]), so that Dirichlet boundary conditions can be handled exactly. It is found that with the sine series $\lambda(\theta)$ is still given by (4.6.2), so all that needs to be done is to replace Θ_r by Θ_r^D in the preceding analysis. This is an example where our heuristic definition of ρ_D leads to the correct result. Assume $n_1 = n_2 = n$. The whole of Θ_r^D is within the hatched region of Figure 4.6.1. Reasoning as before we obtain, for $0 < \varepsilon \leq 1$:

$$\lambda(\pi, \pi) \leq \lambda(\theta) \leq \lambda(2\pi/n, \pi/2), \quad \lambda(\pi, \pi/2) \leq \lambda(\theta) \leq \lambda(\pi/2, 2\pi/n) \qquad (4.6.6)$$

Hence $\rho_D = \max\{|1 - 2\omega|, \ |1 - \varepsilon\omega(1 + 2\pi^2/n^2)/(1 + \varepsilon)|$, so that $\rho_D = \bar{\rho} + O(n^{-2})$, and again we conclude that point Jacobi is not a robust smoother for the anisotropic diffusion equation.

Line Jacobi

We start again with some analytical considerations. Damped vertical line Jacobi iteration applied to the discretized anisotropic diffusion equation (4.5.6) with $c = 1, \ s = 0$ corresponds to the splitting

$$[\boldsymbol{M}] = \omega^{-1} \begin{bmatrix} & -1 & \\ 0 & 2 + 2\varepsilon & 0 \\ & -1 & \end{bmatrix} \qquad (4.6.7)$$

The amplification factor is given by

$$\lambda(\theta) = \omega\varepsilon \cos\ \theta_1/(1 + \varepsilon - \cos\ \theta_2) + 1 - \omega \qquad (4.6.8)$$

both for the exponential and the sine Fourier series. We note immediately that $|\lambda(\pi, 0)| = 1$ if $\omega = 1$, so that for $\omega = 1$ this seems to be a bad smoother. This is surprising, because as $\varepsilon \downarrow 0$ the method becomes an exact solver. This apparent contradiction is resolved by taking boundary conditions into account. In Example 4.6.1 it is shown that

$$\rho_D = |\lambda(\pi, \varphi)| = \varepsilon/(1 + \varepsilon - \cos\ \varphi) \quad \text{for} \quad \omega = 1 \qquad (4.6.9)$$

where $\varphi = 2\pi/n$. As $n \to \infty$ we have

$$\rho_D \simeq (1 + 2\pi^2 h^2/\varepsilon)^{-1} \qquad (4.6.10)$$

so that indeed $\lim\limits_{\varepsilon \downarrow 0} \rho_D = 0$. Better smoothing performance may be obtained by varying ω. In Example 4.6.1 it is shown that $\bar{\rho}$ is minimized by

$$\omega = \frac{2 + 2\varepsilon}{3 + 2\varepsilon} \qquad (4.6.11)$$

Note that for $0 < \varepsilon \leq 1$ we have $2/3 \leq \omega \leq 4/5$, so that the optimum value of ω is only weakly dependent on ε. We also find that for ω in this range the smoothing factor depends only weakly on ω. We will see shortly that fortunately this seems to be true for more general problems also.

With ω according to (4.6.11) we have

$$\bar{\rho} = (1 + 2\varepsilon)/(1 + 3\varepsilon) \qquad (4.6.12)$$

Choosing $\omega = 0.7$ we obtain

$$\bar{\rho} = \max\{1 - 0.7/(1+\varepsilon), 0.6\} \tag{4.6.13}$$

which shows that we have a good smoother for all $0 \le \varepsilon \le 1$, with an ε-independent ω.

Example 4.6.1. Derivation of (4.6.9) and (4.6.11). Note that $\lambda(\theta)$ is real, and that we need to consider only $\theta_\alpha \ge 0$. It is found that $\partial\lambda/\partial\theta_1 = 0$ only for $\theta_1 = 0, \pi$. Starting with ρ_D, we see that $\max\{|\lambda(\theta)| : \theta \in \Theta_r^D\}$ is attained on the boundary of Θ_r^D. Assume $n_1 = n_2 = n$, and define $\varphi = 2\pi/n$. It is easily see that $\max\{|\lambda(\theta)| : \theta \in \Theta_r^D\}$ will be either $|\lambda(\varphi, \pi/2)|$ or $|\lambda(\pi, \varphi)|$. If $\omega = 1$ it is $|\lambda(\pi, \varphi)|$, which gives us (4.6.9). We will determine the optimum value of ω not for ρ_D but for ρ. It is sufficient to look for the maximum of $|\lambda(\theta)|$ on the boundary of Θ_r. It is easily seen that

$$\rho = \max\{|\lambda(0, \pi/2)|,\ |\lambda(\pi, 0)|\} = \max\{1 - \omega/(1+\varepsilon),\ |1 - 2\omega|\}$$

which shows that we must take $0 < \omega < 1$. We find that the optimal ω is given by (4.6.11). Note that in this case we have $\rho = \bar{\rho}$.

Equation (4.5.5), for which the proceeding analysis was done, corresponds to $\beta = 0$ in (4.5.3). For $\beta = \pi/2$ damped vertical line Jacobi does not work, but damped horizontal line Jacobi should be used. The general case may be handled by *alternating Jacobi*: vertical line followed by horizontal line Jacobi. Each step is damped separately with a fixed problem-independent value of ω. After some experimentation $\omega = 0.7$ was found to be suitable; (cf. (4.6.12) and (4.6.13). Table 4.6.1 presents results. Here and in the remainder of this chapter we take $n_1 = n_2 = n$, and β is sampled with intervals of 15^0, unless stated otherwise. The worst case found is included in the tables that follow.

Increasing n, or finer sampling of β around 45^0 or 0^0, does not result in larger values of ρ and ρ_D than those listed in Table 4.6.1. It may be concluded that damped alternating Jacobi with a fixed damping parameter of $\omega = 0.7$ is an efficient and robust smoother for the rotated anisotropic diffusion equation, provided the mixed derivative is discretized according to (4.5.8). Note the good vectorization and parallelization potential of this method.

	(4.5.6)		(4.5.8)	
ε	ρ, ρ_D	β	ρ, ρ_D	β
1	0.28	any	0.28	any
10^{-1}	0.63	45^0	0.38	45^0
10^{-2}	0.95	45^0	0.44	45^0
10^{-3}	1.00	45^0	0.45	45^0
10^{-5}	1.00	45^0	0.45	45^0
10^{-8}	1.00	45^0	0.45	45^0

Table 4.6.1: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation (4.5.3) discretized according to (4.5.6) or (4.5.8); damped alternating Jacobi smoothing; $\omega = 0.7$; $n = 64$.

Convection-diffusion equation
Point Jacobi
For the convection-diffusion equation discretized with stencil (4.5.11) the amplification factor of damped point Jacobi is given by

$$\lambda(\theta) = \omega(2\cos\,\theta_1 + 2\cos\,\theta_2 + P_1 e^{i\theta_1} + P_1 e^{-i\theta_2})/(4 + P_1 + P_2) + 1 - \omega \tag{4.6.14}$$

where $P_1 = ch/\varepsilon,\ P_2 = sh/\varepsilon$. Consider the special case: $P_1 = 0,\ P_2 = 4/\delta$. Then

$$\lambda(\pi, 0) = 1 - \omega + \omega/(1 + \delta) \tag{4.6.15}$$

so that $|\lambda(\pi, 0)| \to 1$ as $\delta \downarrow 0$, for all ω, hence there is no value of ω for which this smoother is robust for the convection-diffusion equation.

Line Jacobi
Let us apply the line Jacobi variant which was found to be robust for the rotated anisotropic diffusion equation, namely damped alternating Jacobi with $\omega = 0.7$, to the convection-diffusion test problem. Results are presented in Table 4.6.2.

Finer sampling of β around $\beta = 0^0$ and increasing n does not result in significant changes. Numerical experiments show $\omega = 0.7$ to be a good value. It may be concluded that damped alternating Jacobi with a fixed damping parameter (for example, $\omega = 0.7$) is a robust and efficient smoother for the convection-diffusion test problem. The same was just found to be true for the rotated anisotropic diffusion test problem. The method vectorizes and parallelizes easily, so that all in all is an attractive smoother.

ε	ρ	β	ρ_D	β
1	0.28	0^0	0.28	0^0
10^{-1}	0.28	0^0	0.29	0^0
10^{-2}	0.29	0^0	0.29	0^0
10^{-3}	0.29	0^0	0.29	0^0
10^{-5}	0.40	0^0	0.30	0^0

Table 4.6.2: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); damped alternating line Jacobi smoothing; $\omega = 0.7$; $n = 64$.

Exercise 4.6.1 Assume semi-coarsening as discussed in Section 4.4: $\bar{h}_1 = h_1$, $\bar{h}_2 = h_2/2$. Show that damped point Jacobi is a good smoother for equation (4.5.5) with $0 < \varepsilon \leq 1$.

Exercise 4.6.2 Show that $\lim\limits_{\varepsilon \downarrow 0} \rho = 1$ for alternating Jacobi with damping parameter $\omega = 1$ applied to the convection-diffusion test problem.

4.7 Gauss-Seidel smoothing

Anisotropic diffusion equation
Point Gauss-Seidel
Forward point Gauss-Seidel iteration applied to (4.5.3) with $c = 1$, $s = 0$ corresponds to the splitting

$$[\boldsymbol{M}] = \begin{bmatrix} & 0 & \\ -\varepsilon & 2\varepsilon + 2 & 0 \\ & -1 & \end{bmatrix}, \quad [\boldsymbol{N}] = \begin{bmatrix} & 1 & \\ 0 & 0 & \varepsilon \\ & 0 & \end{bmatrix} \tag{4.7.1}$$

The amplification factor is given by

$$\lambda(\theta) = (\varepsilon e^{i\theta_2} + e^{i\theta_2})/(-\varepsilon e^{-i\theta_1} + 2\varepsilon + 2 - e^{i\theta_2}) \tag{4.7.2}$$

For $\varepsilon = 1$ (Laplace's equation) one obtains

$$\bar{\rho} = |\lambda(\pi/2,\ \cos^{-1}(4/5))| = 1/2 \tag{4.7.3}$$

To illustrate the technicalities that may be involved in determining $\bar{\rho}$ analytically, we give the details of the derivation of (4.7.3) in the following example.

Example 4.7.1. Smoothing factor of forward point Gauss-Seidel for Laplace equation. We can write

$$|\lambda(\theta)|^2 = (1 + \cos\ \beta)/(9 - 8 \cos\ \frac{\alpha}{2} \cos\ \frac{\beta}{2} + \cos\ \beta) \tag{4.7.4}$$

with $\alpha = \theta_1 + \theta_2$, $\beta = \theta_1 - \theta_2$. Because of symmetry only α, $\beta \geq 0$ has to be considered. We have

$$\partial|\lambda(\theta)|^2/\partial\alpha = 0 \quad \text{for} \quad \sin(\alpha/2)\cos(\beta/2) = 0 \tag{4.7.5}$$

This gives $\alpha = 0$ or $\alpha = 2\pi$ or $\beta = \pi$. For $\beta = \pi$ we have a minimum: $|\lambda|^2 = 0$. With $\alpha = 0$ we have $|\lambda(\theta)|^2 = \cos^2(\beta/2)/(2 - \cos(\beta/2))^2$, which reaches a maximum for $\beta = 2\pi$, i.e. at the boundary of $\bar{\Theta}_r$. With $\alpha = 2\pi$ we are also on the boundary of $\bar{\Theta}_r$. Hence, the maximum of $|\lambda(\theta)|$ is reached on the boundary of $\bar{\Theta}_r$. We have $|\lambda(\pi/2, \theta_2)|^2 = (1+\sin\theta_2)/(9+\sin\theta_2 - 4\cos\theta_2)$, of which the θ_2 derivative equals 0 of $8\ \lambda\cos\ \theta_2 - 4\sin\ \theta_2 - 4 = 0$, hence $\theta_2 = -\pi/2$, which gives a minimum, or $\theta_2 = \pm\cos\ (4/5)$. The largest maximum is obtained for $\theta_2 = \cos^{-1}\ (4/5)$. The extrema of $|\lambda(\pi, \theta_2|$ are studied in similar fashion. Since $\lambda(\theta_1, \theta_2) = \lambda(\theta_2, \theta_1)$ there is not need to study $|\lambda(\theta_1, \pi/2)|$ and $|\lambda(\theta_1, \pi)|$. Equation (4.7.3) follows.

We will not determine $\bar{\rho}$ analytically for $\varepsilon \neq 1$, because this is very cumbersome. To do this numerically is easy, of course. Note that $\lim\limits_{\varepsilon\to 0} \lambda(\pi, 0) = 1$, $\lim\limits_{\varepsilon\to\infty} \lambda(\pi, 0) = -1$, so that forward point Gauss-Seidel is not a robust smoother for the anisotropic diffusion equation, if standard coarsening is used. See also Exercise 4.7.1.

With semi-coarsening in the x_2 direction we obtain in Example 4.7.2: $\bar{\rho} \leq \{(1 + \varepsilon)/(5 + \varepsilon)\}^{1/2}$, which is satisfactory for $\varepsilon \leq 1$. For $\varepsilon \geq 1$ one should use semi-coarsening in the x_1-direction. Since in practice one may ave $\varepsilon \ll 1$ in one part of the domain and $\varepsilon \gg 1$ in another, semi-coarsening gives a robust method with this smoother only if the direction of semi-coarsening is varied in the domain, which results in more complicated code than standard multigrid.

Example 4.7.2. Influence of semi-coarsening. We will show

$$\bar{\rho} \leq [(1 + \varepsilon)/(5 + \varepsilon)]^{1/2} \tag{4.7.6}$$

for the smoother defined by 94.7.1) with semi-coarsening in the x_2 direction. From (4.7.2) it follows that one may write $|\lambda(\theta)|^{-2} = 1 + (2 + 2\varepsilon)\mu(\theta)$ with $\mu(\theta) = (2 + 2\varepsilon - 2\varepsilon\cos\ \theta_1 - 2\cos\ \theta_2)/[1 + \varepsilon^2 + 2\varepsilon\cos\ (\theta_1 - \theta_2]$. In this case, $\bar{\Theta}_r$ is given in Figure 4.4.2. On $\bar{\Theta}_r$ we have

$$\mu(\theta) \geq (2 + 2\varepsilon - 2\varepsilon\cos\ \theta_1 - 2\cos\ \theta_2)/(1 + \varepsilon)^2 \geq 2/(1 + \varepsilon)^2\ .$$

Hence $|\lambda(\theta)| \geq [1 + 4/(1 + \varepsilon)]^{-1/2}$, and (4.7.6) follows.

For backward Gauss-Seidel the amplification factor is $\lambda(-\theta)$, with $\lambda(\theta)$ given by (4.7.2), so that the amplification factor of symmetric Gauss-Seidel is given by $\lambda(-\theta)\lambda(\theta)$. From (4.7.2) it follows that $|\lambda(\theta)| = |\lambda(-\theta)|$, so that the smoothing factor is the square of the smoothing factor for forward point Gauss-Seidel, hence, symmetric Gauss-Seidel is also not robust for the anisotropic diffusion equation. Also, point Gauss-Seidel-Jacobi (Section 3.3) does not work for this test problem.

The general rule is: *points that are strongly coupled must be updated simultaneously.* Here we mean by strongly coupled points: points with large coefficients (absolute) in $[\boldsymbol{A}]$. For example, in the case of Equation (4.5.5) with $\varepsilon \ll 1$ points on the same vertical line are strongly coupled. Updating these points simultaneously leads to the use of line Gauss-Seidel.

Line Gauss-Seidel
Forward vertical line Gauss-Seidel iteration applied to the anisotropic diffusion equation (4.5.5) corresponds to the splitting

$$[\boldsymbol{M}] = \begin{bmatrix} & -1 & \\ -\varepsilon & 2\varepsilon+2 & 0 \\ & -1 & \end{bmatrix}, \quad [\boldsymbol{N}] = \begin{bmatrix} & 0 & \\ 0 & 0 & \varepsilon \\ & 0 & \end{bmatrix} \tag{4.7.7}$$

The amplification factor is given by

$$\lambda(\theta) = \varepsilon e^{i\theta_1}/(2\varepsilon + 2 - 2\cos\ \theta_2 - \varepsilon e^{-i\theta_1}) \tag{4.7.8}$$

and we find Example 4.7.3, which follows shortly:

$$\bar{\rho} = \max\{5^{-1/2},\ (2/\varepsilon + 1)^{-1}\} \tag{4.7.9}$$

Hence, $\lim\limits_{\varepsilon\downarrow 0} \bar{\rho} = 5^{-1/2}$. This is surprising, because for $\varepsilon = 0$ we have, with Dirichlet boundary conditions, uncoupled non-singular tridiagonal systems along vertical lines, so that the smoother is an exact solver, just as in the case of line Jacobi smoothing, discussed before. The behaviour of this smoother in practice is better predicted by taking the influence of Dirichlet boundary conditions into account. We find in Example 4.7.3 below:

$$\begin{aligned} \varepsilon < (1+\sqrt{5})/2: &\quad \rho_D = \varepsilon[\varepsilon^2 + (2\varepsilon + 2 - 2\cos\ \varphi)^2]^{-1/2} \\ \varepsilon \geq (1+\sqrt{5})/2: &\quad \rho_D = \varepsilon[\varepsilon^2 + (2\varepsilon+2)(2\varepsilon + 2 - 2\varepsilon\cos\ \varphi)]^{-1/2} \end{aligned} \tag{4.7.10}$$

with $\varphi = 2\pi h,\ h = 1/n$, assuming for simplicity $n_1 = n_2 = n$. For $\varepsilon < (1+\sqrt{5})/2$ and $h \downarrow 0$ this can be approximated by

$$\rho_D \cong [1 + (2 + \varphi^2/\varepsilon)^2]^{-1/2} \tag{4.7.11}$$

and we see that the behaviour of ρ_D as $\varepsilon \downarrow 0,\ h \downarrow 0$ depends on $\varphi^2/\varepsilon = 4\pi^2 h^2/\varepsilon$. For $h \downarrow 0$ with ε fixed we have $\rho_D \cong \bar{\rho}$ and recover (4.7.9); for $\varepsilon \downarrow 0$ with h fixed we obtain $\rho_D \cong 0$. To give a practical example, with $h = 1/128$ and $\varepsilon = 10^{-6}$ we have $\rho_D \cong 0.0004$.

Example 4.7.3. Derivation of (4.7.9) and (4.7.10). It is convenient to work with $|\lambda(\theta)|^{-2}$. We have

$$|\lambda(\theta)|^{-2} = [(2\varepsilon + 2 - \varepsilon \cos\ \theta_1 - 2\cos\ \theta_2)^2 + \varepsilon^2 \sin^2\ \theta_1]/\varepsilon^2 \ .$$

Min $\{|\lambda(\theta)|^{-2} : \theta \in \Theta_r^D\}$ is determined as follows. We need to consider only $\theta_\alpha \geq 0$. It is found that $\partial|\lambda(\theta)|^{-2}/\partial\theta_2 = 0$ for $\theta_2 = 0$ for $\theta_2 = 0, \pi$ only. Hence the minimum is attained on the boundary of Θ_r^D. Choose for simplicity $n_1 = n_2 = n$, and define $\varphi = 2\pi/n$. It is easily seen that in Θ_r^D we have

$$\begin{aligned}
|\lambda(\theta_1, \varphi)|^{-2} &\geq |\lambda(\pi/2, \varphi)|^{-2}\,, \quad |\lambda(\varphi, \theta_2)|^{-2} \geq |\lambda(\varphi, \pi/2)|^{-2}\,, \\
|\lambda(\pi, \theta_2)|^{-2} &\geq |\lambda(\pi, \varphi)|^{-2}\,, \quad |\lambda(\theta_1, \pi/2)|^{-2} \geq |\lambda(\varphi, \pi/2)|^{2}\,, \\
|\lambda(\pi/2, \theta_2)|^{-2} &\geq |\lambda(\pi/2, \varphi)|^{-2}\,, \quad |\lambda(\theta_1, \pi)|^{-2} \geq |\lambda(\varphi, \pi)|^{-2}
\end{aligned}$$

For $\varepsilon < (1 + \sqrt{5})/$ the minimum is $|\lambda(\pi/2, \varphi)|^{-2}$; for $\varepsilon \geq (1 + \sqrt{5})/2$, the minimum is $|\lambda(\varphi, \pi/2)|^{-2}$. This gives us (4.7.10). We continue with (4.7.9). The behaviour of $|\lambda(\theta)|$ on the boundary of $\bar{\Theta}_r$ is found simply by letting $\varphi \to 0$ in the preceding results. Now there is also the possibility of a minimum in the interior of $\bar{\Theta}_r$, because $\theta_2 = 0$ is allowed, but this leads to the minimum in $(\pi/2, 0)$, which is on the boundary, and (4.7.9) follows.

Equations (4.7.9) and (4.7.10) predict bad smoothing when $\varepsilon \gg 1$. Of course, for $\varepsilon \gg 1$ *horizontal* line Gauss-Seidel should be used. A good smoother for arbitrary ε is *alternating line Gauss-Seidel*. For analytical results, see [141]. Table 4.7.1 presents numerical values of ρ and ρ_D for a number of cases. We take $n_1 = n_2 = n = 64$, $\beta = k\pi/12$, $k = 0, 1, 2, ..., 23$ in (4.5.3), and present results only for a value of β for which the largest ρ or ρ_D is obtained. In the cases listed, $\rho = \rho_D$. Alternating line Gauss-Seidel is found to be a robust smoother for the rotated anisotropic diffusion equation if the mixed derivative is discretized according to (4.5.8), but not if (4.5.6) is used. Using under-relaxation does not change this conclusion.

Convection-diffusion equation

Point Gauss-Seidel

Forward point Gauss-Seidel iteration applied to the central discretization of the convection-diffusion equation (4.5.10) is not a good smoother, see [141].

For the upwind discretization (4.5.11) one obtains, assuming $c > 0,\ s > 0$:

$$\lambda(\theta) = \frac{e^{i\theta_1}[1 + (|P_1| - P_1)/2] + e^{i\theta_2}[1 + (|P_2| - P_2)/2]}{4 + |P_1| + |P_2| - e^{-i\theta_1}[1 + (P_1 + |P_1|)/2] - e^{i\theta_2}[1 + (P_2 + |P_2|)/2]} \tag{4.7.12}$$

with $P_1 = ch/\varepsilon,\ P_2 = sh/\varepsilon$ the mesh-Péclet numbers (for simplicity we assume $n_1 = n_2$). For $P_1 > 0,\ P_2 < 0$ we have $|\lambda(0, \pi)| = |P_2/(4 - P_2)|$, which tends to 1 as $|P_2| \to \infty$. To avoid

	(4.5.6)		(4.5.8)	
ε	ρ, ρ_D	β	ρ, ρ_D	β
1	0.15	any	0.15	any
10^{-1}	0.38	105^0	0.37	15^0
10^{-2}	0.86	45^0	0.54	15^0
10^{-3}	0.98	45^0	0.58	15^0
10^{-5}	1.00	45^0	0.59	15^0

Table 4.7.1: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation (4.5.3) discretized according to (4.5.6) and (4.5.8); alternating line Gauss-Seidel smoothing; $n = 64$.

this the order in which the grid points are visited has to be reversed: backward Gauss-Seidel. Symmetric point Gauss-Seidel (forward followed by backward) therefore is more promising for the convection-diffusion equation. Table 4.7.2 gives some numerical results for ρ, for $n_1 = n_2 = 64$. We give results for a value of β in the set $\{\beta = k\pi/12 : k = 0, 1, 2, ..., 23\}$ for which the largest ρ and ρ_D are obtained.

Although this is not obvious from Table 4.7.2, the type of boundary condition may make a large difference. For instance, for $\beta = 0$ and $\varepsilon \downarrow 0$ one finds numerically for forward point Gauss-Seidel: $\rho = |\lambda(0, \pi/2)| = 1/\sqrt{5}$, whereas $\lim\limits_{\varepsilon \downarrow 0} \rho_D = 0$, which is more realistic, since as $\varepsilon \downarrow 0$ the smoother becomes an exact solver. The difference between ρ and ρ_D is explained by noting that for $\theta_1 = \varphi = 2\pi h$ and $\varphi \ll 1$ we have $|\lambda(\varphi, \pi/2)|^2 = 1/(5 + y + \frac{1}{2}y^2)$ with $y = 2\pi h^2 \varepsilon$.
For $\varepsilon \ll 1$ and $\beta = 105^0$ Table 4.7.2 shows rather large smoothing factors. In fact, symmetric point Gauss-Seidel smoothing is not robust for this test problem. This can be seen as follows. If $P_1 < 0,\ P_2 > 0$ we find

$$\lambda(\frac{\pi}{2}, 0) = \frac{1 - P_1 + i}{3 - P_1 + i} \cdot \frac{1 + P_2 - i}{3 - P_1 + P_2 - i(1 - P_1)} \tag{4.7.13}$$

Choosing $P_1 = -\alpha P_2$ one obtains, assuming $P_2 \gg 1,\ \alpha P_2 \gg 1$:

$$|\lambda(\frac{\pi}{2}, 0) \cong (1 + \alpha)^{-2} \tag{4.7.14}$$

so that ρ may get close to 1 if α is small. The remedy is to include more sweep directions. Four-direction point Gauss-Seidel (consisting of four successive sweeps with four orderings:

ε	ρ	ρ_D	β
1	0.25	0.25	0
10^{-1}	0.27	0.25	0
10^{-2}	0.45	0.28	105^0
10^{-3}	0.71	0.50	105^0
10^{-5}	0.77	0.71	105^0

Table 4.7.2: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); symmetric point Gauss-Seidel smoothing.

the forward and backward orderings of Figure 3.3.1, the forward vertical line ordering of Figure 3.3.1, and this last ordering reversed) is robust for this test problem, as illustrated by Table 4.7.3.

As before, we have taken $\beta = k\pi/12$, $k = 0, 1, 2, ..., 23$; Table 4.7.3 gives results only for a value of β for which the largest ρ and ρ_D are obtained. Clearly, four-direction point Gauss-Seidel is an excellent smoother for the convection-diffusion equation. It is found that ρ and ρ_D change little when n is increased further.
Another useful smoother for this test problem is four-direction point Gauss-Seidel-Jacobi,

ε	ρ	ρ_D	β
1	0.040	0.040	0^0
10^{-1}	0.043	0.042	0^0
10^{-2}	0.069	0.068	0^0
10^{-3}	0.16	0.12	0^0
10^{-5}	0.20	0.0015	15^0

Table 4.7.3: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); four-direction point Gauss-Seidel smoothing; $n = 64$.

defined in Section 3.3. As an example, we give for discretization (4.5.11) the splitting for the forward step:

$$[\boldsymbol{M}] = \varepsilon \begin{bmatrix} & 0 & \\ -1 & 4 & 0 \\ & 0 & \end{bmatrix} + \tfrac{h}{2}[-c - |c| \;\; 2|c| \;\; 0] \qquad (4.7.15)$$
$$[\boldsymbol{N}] = [\boldsymbol{M}] - [\boldsymbol{A}]$$

The amplification factor is easily derived. Table 4.7.4 gives results, sampling β as before. The results are satisfactory, but there seems to be a degradation of smoothing performance in the vicinity of $\beta = 0^0$ (and similarly near $\beta = k\pi/2,\ k = 1,2,3$). Finer sampling with intervals of 1^0 gives the results of Table 4.7.5.

This smoother is clearly usable, but it is found that damping improves performance still further. Numerical experiments show that $\omega = 0.8$ is a good value; each step is damped separately. Results are given in Table 4.7.6. Clearly, this is an efficient and robust smoother for the convection-diffusion equation, with ω fixed at $\omega = 0.8$. Choosing $\omega = 1$ gives a little improvement for $\varepsilon/h \geq 0.1$, but in practice a fixed value of ω is to be preferred, of course.

ε	ρ	ρ_D	β
1	0.130	0.130	0^0
10^{-1}	0.130	0.130	45^0
10^{-2}	0.127	0.127	45^0
10^{-3}	0.247	0.242	15^0
10^{-5}	0.509	0.494	15^0
10^{-8}	0.514	0.499	15^0

Table 4.7.4: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); four-direction point Gauss-Seidel-Jacobi smoothing; $n = 64$.

ε	n	ρ	β	ρ_D	β
10^{-8}	64	0.947	1^0	0.562	8^0
10^{-8}	128	0.949	1^0	0.680	5^0

Table 4.7.5: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); four-direction point Gauss-Seidel-Jacobi smoothing.

Line Gauss-Seidel
For forward vertical line Gauss-Seidel we have

$$\begin{aligned}\lambda(\theta) = e^{i\theta_1}[1 - P_1 - |P_1|)/2]/\{4 + |P_1| + |P_2| - e^{i\theta_1}[1 + (P_1 + |P_1|)/2] \\ -e^{i\theta_2}[1 + (|P_2| - P_2)/2] - e^{i\theta_2}[1 + (P_2 + |P_2|)/2]\} \qquad (4.7.16)\end{aligned}$$

ε	ρ, ρ_D	β
1.0	0.214	0^0
10^{-1}	0.214	0^0
10^{-2}	0.214	45^0
10^{-3}	0.217	45^0
10^{-5}	0.218	45^0
10^{-8}	0.218	45^0

Table 4.7.6: Fourier smoothing factors, ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); four-direction point Gauss-Seidel-Jacobi smoothing with damping parameter $\omega = 0.8$; $n = 64$.

For $P_1 < 0$, $P_2 > 0$ this gives $|\lambda(\pi, 0)| = (1 - P_1)/(3 - P_1)$, which tends to 1 as $|P_1| \to \infty$, so that this smoother is not robust. Alternating line Gauss-Seidel is also not robust for this test problem. If $P_2 < 0$, $P_1 = \alpha P_2$, $\alpha > 0$ and $|P_2| \gg 1$, $|\alpha P_2| \gg 1$ then

$$\lambda(0, \pi/2) \cong i\alpha/(1 + \alpha - i) \tag{4.7.17}$$

so that $|\lambda(0, \pi/2)| \cong \alpha/[(1+\alpha)^2 + 1]^{1/2}$, which tends to 1 if $\alpha \gg 1$. Symmetric (forward followed by backward) horizontal and vertical line Gauss-Seidel are robust for this test problem. Table 4.7.7 presents some results. Again, $n = 64$ and $\beta = k\pi/2$, $k = 0, 1, 2, ..., 23$; Table 4.7.7

ε	ρ	β	ρ_D	β
1	0.20	90^0	0.20	90^0
10^{-1}	0.20	90^0	0.20	90^0
10^{-2}	0.20	90^0	0.20	90^0
10^{-3}	0.30	0^0	0.26	0^0
10^{-5}	0.33	0^0	0.0019	75^0

Table 4.7.7: Fourier smoothing factors, ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); symmetric vertical line Gauss-Seidel smoothing; $n = 64$.

gives results only for the worst case in β.
We will not analyse these results further. Numerically we find that for $\beta = 0$ and $\varepsilon \ll 1$ that $\rho = (\lambda(0, \pi/2) = (1 + P_1)/(9 + 3P_1) \cong 1/3$. As $\varepsilon \downarrow 0$, ρ_D depends on the value of $n\varepsilon$. It is clear that we have a robust smoother.

We may conclude that alternating symmetric line Gauss-Seidel is robust for both test problems, provided the mixed derivative is discretized according to (4.5.8). A disadvantage of this smoother is that it does not lend itself to vectorized or parallel computing.
The Jacobi-type methods discussed earlier and Gauss-Seidel with pattern orderings (white-black, zebra) are more favourable in this respect. Fourier smoothing analysis of Gauss-Seidel with pattern orderings is more involved, and is postponed to a later section.

Exercise 4.7.1 Show that damped point Gauss-Seidel is not robust for the rotated anisotropic diffusion equation with $c = 1,\ s = 0$, with standard coarsening.

Exercise 4.7.2 An Exercise 4.7.1, but for Gauss-Seidel-Jacobi method.

4.8 Incomplete point LU smoothing

For Fourier analysis it is necessary that $[\boldsymbol{M}]$ and $[\boldsymbol{N}]$ are constant, i.e. do not depend on the location in the grid. For the methods just discussed this is the case if $[\boldsymbol{A}]$ is constant. For incomplete factorization smoothing methods this is not, however, sufficient. Near the boundaries of the domain $[\boldsymbol{M}]$ (and hence $[\boldsymbol{N}] = [\boldsymbol{M}] - [\boldsymbol{A}]$) varies, usually tending rapidly to a constant stencil away from the boundaries. Nevertheless, useful predictions about the smoothing performance of incomplete factorization smoothing can be made by means of Fourier analysis. How this can be done is best illustrated by means of an example.

Five-point ILU
This incomplete factorization has been defined in Section 4.4, in standard matrix notation. In Section 4.4 $\boldsymbol{A}$ was assumed to have a five-point stencil. With application to test problem (4.5.6) in mind, $\boldsymbol{A}$ is assumed to have the seven-point given below. In stencil notation we have

$$\begin{aligned}
[\boldsymbol{A}] &= \begin{bmatrix} f & g & \\ c & d & q \\ & a & b \end{bmatrix}, \quad [\boldsymbol{L}]_i = \begin{bmatrix} & 0 & \\ c & \delta_i & 0 \\ & a & \end{bmatrix} \\
[\boldsymbol{D}]_i &= \begin{bmatrix} & 0 & \\ 0 & \delta_i & 0 \\ & 0 & \end{bmatrix}, \quad [\boldsymbol{U}]_i = \begin{bmatrix} & g & \\ 0 & \delta_i & q \\ & 0 & \end{bmatrix}
\end{aligned} \tag{4.8.1}$$

where $i = (i_1, i_2)$. We will study the unmodified version. For δ_i we have the recursion (3.4.12) with $\sigma = 0$:

$$\delta_i = d - ag/\delta_{i-e_2} - cq/\delta_{i-e_1} \tag{4.8.2}$$

where $e_1 = (1, 0),\ e_2 = (0, 1)$. Terms involving negative values of $i_\alpha,\ \alpha = 1$ or 2, are to be repaced by zero. We will show the following Lemma.

Lemma 4.8.1. If

$$a + c + d + q + g \geq 0, \quad a, c, q, g \leq 0, \quad d > 0 \tag{4.8.3}$$

then

$$\lim_{i_1, i_2 \to \infty} \delta_i = \delta \equiv d/2 + [d^2/4 - (ag + cq)]^{1/2} \tag{4.8.4}$$

The proof is given in [141]. Note that (4.8.3) is satisfied if $b = f = 0$ and $\boldsymbol{A}$ is a K-matrix (Section 3.2). Obviously, δ is real, and $\delta \geq d$. The rate at which the limit is reached in (4.8.4) is studied in [141]. A sufficient number of mesh points away from the boundaries of the grid G we have approximately $\delta_i = \delta$, and replacing δ_i by δ we obtain for $[\boldsymbol{M}] = [\boldsymbol{L}][\boldsymbol{D}^{-1}][\boldsymbol{U}]$:

$$[\boldsymbol{M}] = \begin{bmatrix} cg/\delta & g & \\ c & d & q \\ & a & aq/\delta \end{bmatrix} \tag{4.8.5}$$

and standard Fourier smoothing analysis can be applied. Equation (4.8.5) is derived easily by nothing that in stencil notation $(\boldsymbol{AB}u)_i = \Sigma_j \Sigma_k \boldsymbol{A}(i,j)\boldsymbol{B}(i+j,k)u_{i+j+k}$, so that $\boldsymbol{A}(i,j)\boldsymbol{B}(i+j,k)$ gives a contribution to $\boldsymbol{C}(i,j+k)$, where $\boldsymbol{C} = \boldsymbol{AB}$; by summing all contributions one obtains $\boldsymbol{C}(i,l)$. An explicit expression for $\boldsymbol{C}(i,l)$ is $\boldsymbol{C}(i,l) = \Sigma_j \boldsymbol{A}(i,j)\boldsymbol{B}(i+j,l-j)$, since one can write $(\boldsymbol{C}u)_i = \Sigma_l \Sigma_j \boldsymbol{A}(i,j)\boldsymbol{B}(i+j,l-j)u_{i+l}$.

Smoothing factor of five-point ILU

The modified version of incomplete factorization will be studied. As remarked in [145] modification is better than damping, because if the error matrix $\boldsymbol{N}$ is small with $\sigma = 0$ it will also be small with $\sigma \neq 0$. The optimum σ depends on the problem. A fixed σ for all problems is to be preferred. From the analysis and experiments in [145] and [147] and our own experiments it follows that $\sigma = 0.5$ is a good choice for all point-factorizations considered here and all problems. Results will be presented with $\sigma = 0$ and $\sigma = 0.5$. The modified version of the recursion (3.4.12) for δ_k is

$$\delta_k = d - ag/\delta_{k-I} - cq/\delta_{k-1} + \sigma\{|aq/\delta_{k-I} - b| + |cg/\delta_{i-1} - f|\} \tag{4.8.6}$$

The limiting value δ in the interior of the domain, far from the boundaries, satisfies (4.8.6) with the subscripts omitted, and is easily determined numerically by the following recursion

$$\delta_{k+1} = d - (aq + cq)/\delta_k + \sigma\{|aq/\delta_k - b| + |cg/\delta_k - f|\} \tag{4.8.7}$$

The amplification factor is given by

$$\begin{aligned} \lambda(\theta) \;=\; & \{(aq/\delta - b)exp[i(\theta_1 - \theta_2)] + (cg/\delta - f)exp[i(\theta_2 - \theta_1)] + \sigma p\}/ \\ & \{aexp(-i\theta_2) + aqexp[i(\theta_1 - \theta_2)]/\delta + cexp(-i\theta_1) + d + \sigma p \\ & + qexp(i\theta_i) + cgexp[i(\theta_2 - \theta_1)]/\delta + gexp(i\theta_2)\} \end{aligned} \tag{4.8.8}$$

where $p = |aq/\delta - b| + |cg/\delta - f|$.

Anisotropic diffusion equation

For the (non-rotated $\beta = 0^0$) anisotropic diffusion equation with discretization (4.5.6) we have $g = a = -1,\ c = q = -\varepsilon,\ d = 2 + 2\varepsilon,\ b = f = 0$, and we obtain: $\delta = 1 + \varepsilon + [2\varepsilon(1+\sigma)]^{1/2}$, and

$$\begin{aligned}\lambda(\theta) = &[\varepsilon\cos(\theta_1 - \theta_2)/\delta + \sigma\varepsilon/\delta]/\\ &[1 + \varepsilon + \sigma\varepsilon/\delta - \varepsilon\cos\theta_1 - \cos\theta_2 + \varepsilon\cos(\theta_1 - \theta_2)/\delta]\end{aligned} \tag{4.8.9}$$

We will study a few special cases. For $\varepsilon = 1$ and $\sigma = 0$ we find in [141]:

$$\bar{\rho} = |\lambda(\pi/2, -\pi/3)| = (2\sqrt{3} + \sqrt{6} - 1)^{-1} \simeq 0.2035 \tag{4.8.10}$$

The case $\varepsilon = 1,\ \sigma \neq 0$ is analytically less tractable. For $\varepsilon \ll 1$ we find in [141]:

$$\begin{aligned}0 \le \sigma < 1/2: &\quad \bar{\rho} \cong |\lambda(\pi, 0) = (1-\sigma)/(2\delta - 1 + \sigma)\\ 1/2 \le \sigma \le 1: &\quad \bar{\rho} \cong |\lambda(\pi/2) = \sigma/(\sigma + \delta)\end{aligned} \tag{4.8.11}$$

$$\begin{aligned}0 \le \sigma < 1/2: &\quad \rho_D \cong |\lambda(\pi, \tau)| = (1-\sigma)/(2\delta - 1 + \sigma + \delta\tau^2/2\varepsilon)\\ 1/2 \le \sigma \le 1: &\quad \rho_D \cong |\lambda(\pi/2, \tau)| = (\sigma + \tau)(\sigma + \delta + \delta\tau^2/2\varepsilon)\end{aligned} \tag{4.8.12}$$

where $\tau = 2\pi/n_2$. These analytical results are confirmed by Table 4.8.1. For example, for $\varepsilon = 10^{-3}$, $n_2 = 64$ and $\sigma = 1/2$ equation (4.8.12) gives $\rho_D \cong 0.090$, $\bar{\rho} \cong 1/3$. Table 4.8.1 includes the worst case for β in the set $\{\beta = k\pi/12,\ k = 0, 1, 2, ..., 23)$. Here we have another example showing that the influence of the type of the boundary conditions on smoothing analysis may be important. For the non-rotated anisotropic diffusion equation ($\beta = 0^0$ or $\beta = 90^0$) we have a robust smoother both for $\sigma = 0$ and $\sigma = 1/2$, provided the boundary conditions are of Dirichlet type at those parts of the boundary that are perpendicular to the direction of strong coupling. When β is arbitrary, five-point ILU is not a robust smoother with $\sigma = 0$ or $\sigma = 1/2$. We have not experimented with other values of σ, because, as it will turn out, there are other smoothers that are robust, with a fixed choice of σ, that does not depend on the problem.

ε	σ	ρ $\beta = 0^0, 90^0$	ρ $\beta = 15^0$	ρ_D $\beta = 0^0, 90^0$	ρ_D $\beta = 15^0$
1	0	0.20	0.20	0.20	0.20
10^{-1}	0	0.48	1.48	0.46	1.44
10^{-2}	0	0.77	7.84	0.58	6.90
10^{-3}	0	0.92	13.0	0.16	10.8
10^{-5}	0	0.99	13.9	0.002	11.5
1	0.5	0.20	0.20	0.20	0.20
10^{-1}	0.5	0.26	0.78*	0.26	0.78*
10^{-2}	0.5	0.30	1.06	0.025	1.01
10^{-3}	0.5	0.32	1.25	0.089	1.18
10^{-5}	0.5	0.33	1.27	0.001	1.20

Table 4.8.1: Fourier smoothing factors, ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to (4.5.6); five-point ILU smoothing; $n = 64$. In the cases marked with *, $\beta = 45^0$

Convection-diffusion equation

Let us take $P_1 = -\alpha P_2,\ \alpha > 0,\ P_2 > 0$, where $P_1 = ch/\varepsilon,\ P_2 = sh/\varepsilon$. Then we have for the convection-diffusion equation discretized according to (4.5.11): $a = -1 - P_2,\ b = f = 0,\ c = -1,\ d = 4 + (1+\alpha)P_2,\ q = -1 - \alpha P_2,\ g = -1$. After some manipulation one finds that if $\alpha \ll 1,\ P_2 \gg 1,\ \alpha P_2 \gg 1$, then $\lambda(\pi/2, 0) \to i$ as $P_2 \to \infty$. This is accordance with Table 4.8.2. The worst case obtained when β is varied according to $\beta = k\pi/12,\ k = 0, 1, 2, ..., 23$ is listed. Clearly, five-point ILU is not robust for the convection-diffusion equation, at least for $\sigma = 0$ and $\sigma = 0.5$

Seven-point ILU

Seven-point ILU tends to be more efficient and robust than five-point ILU. Assume

$$[\boldsymbol{A}] = \begin{bmatrix} f & g & \\ c & d & q \\ & a & b \end{bmatrix} \tag{4.8.13}$$

The seven-point incomplete factorization $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} - \boldsymbol{N}$ discussed in Section 4.4 is defined in stencil notation as follows:

$$[\boldsymbol{L}]_i = \begin{bmatrix} 0 & 0 & \\ \gamma_i & \delta_i & 0 \\ & \alpha_i & \beta_i \end{bmatrix},\quad [\boldsymbol{D}_i = \begin{bmatrix} 0 & 0 & \\ 0 & \delta_i & 0 \\ & 0 & 0 \end{bmatrix},\quad [\boldsymbol{U}]_i = \begin{bmatrix} \zeta_i & \eta & \\ 0 & \delta_i & \mu_i \\ & 0 & 0 \end{bmatrix} \tag{4.8.14}$$

ε	ρ	ρ_D	β	ρ	ρ_D
	$\sigma = 0$			$\sigma = 0.5$	
1	0.20	0.20	0^0	0.20	0.20
10^{-1}	0.21	0.21	0^0	0.20	0.20
10^{-2}	0.24	0.24	120^0	0.24	0.24
10^{-3}	0.60	0.60	105^0	0.48	0.48
10^{-5}	0.77	0.71	105^0	0.59	0.58

Table 4.8.2: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); five-point ILU smoothing; $n = 64$

We have, taking the limit $i \to \infty$, assuming the limit exists and writing $\lim_{i\to\infty} \alpha_i = \alpha$ etc.,

$$\begin{aligned} \alpha &= a,\ \beta = b - a_\mu/\delta,\ \gamma = c - a\zeta/\delta\ , \\ \mu &= q - \beta g/\delta,\ \zeta = f - \gamma g/\delta,\ \eta = g \end{aligned} \tag{4.8.15}$$

with δ the appropriate root of

$$\delta = d - (ag + \beta\zeta + \gamma\mu)\delta + \sigma(|\beta\mu/\delta| + |\gamma\zeta/\delta|) \tag{4.8.16}$$

Numerical evidence indicates that the limiting δ resulting as $i \to \infty$ is the same as that for the following recursion,

$$\begin{gathered} \beta_0 = b,\ \gamma_0 = c,\ \delta_0 = d,\ \mu_0 = q,\ \zeta_0 = f \\ \beta_{j+1} = b - a\mu_j/\delta_j,\ \gamma_{j+1} = c - a\zeta_j/\delta_j \\ \delta_{j+1} = d - (ag + \beta_{j+1}\zeta_j + \gamma_{j+1}\mu_j)/\delta_j + \sigma(|\beta_{j+1}\mu_j/\delta_j| + |\gamma_{j+1}\zeta_j/\delta_j|) \\ \mu_{j+1} = q - \beta_{j+1}g/\delta_j,\ \zeta_{j+1} = f - \gamma_{j+1}g/\delta_j \end{gathered} \tag{4.8.17}$$

For $\boldsymbol{M}$ we find $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} = \boldsymbol{A} + \boldsymbol{N}$, with

$$[\boldsymbol{N}] = \begin{bmatrix} p_2 & 0 & 0 & & \\ & 0 & p_3 & & \\ & 0 & 0 & 0 & p_1 \end{bmatrix}, \quad \begin{aligned} p_1 &= \beta\mu/\delta,\ p_2 = \gamma\zeta/\delta, \\ p_3 &= \sigma(|p_1| + |p_2|_1) \end{aligned} \tag{4.8.18}$$

The amplification factor is given by

$$\begin{aligned} \lambda(\theta) = {} & \{p_3 + p_1\exp[i(2\theta_1 - \theta_2)] + p_2\exp[-i(2\theta_1 - \theta_2)]\}/ \\ & \{a\exp(-i\theta_2) + b\exp[i(\theta_2 - \theta_1)] + p_1\exp[i(2\theta_1 - \theta_2)] + c\exp(-i\theta_1) \\ & + d + p_3 + q\exp(i\theta_1) + p_2\exp[-i(2\theta_1 - \theta_2)] \\ & + f\exp[-i(\theta_1 - \theta_2)] + g\exp(i\theta_2)\} \end{aligned} \tag{4.8.19}$$

Anisotropic diffusion equation

For the anisotropic diffusion problem discretized according to (4.5.9) we have symmetry: $\mu = \gamma,\ \zeta = \beta,\ g = a,\ f = b,\ q = c$, so that (4.8.19) becomes

$$\lambda(\theta) = [\sigma p + p\cos(2\theta_1 - \theta_2]/ \\ [a\cos\theta_2 + b\cos(\theta_1 - \theta_2) + c\cos\theta_1 + d/2 + \sigma p + p\cos(2\theta_1 - \theta_2)] \qquad (4.8.20)$$

with $p = \beta\mu/\delta$.

With rotation angle $\beta = 90^0$ and $\varepsilon \ll 1$ we find in [141]:

$$\begin{array}{ll} 0 \le \sigma < 1/2: & \bar{\rho} \simeq |\lambda(0,\pi)| \simeq \frac{(1-\sigma)p}{2\varepsilon+\sigma p - p} \\ 1/2 \ge \sigma \ge 1: & \bar{\rho} \simeq |\lambda(0,\pi/2)| \simeq \frac{\sigma p}{\varepsilon + \sigma p} \end{array} \qquad (4.8.21)$$

$$\begin{array}{ll} 0 \le \sigma < 1/2: & \rho_D \simeq |\lambda(\varphi,\pi)| \simeq |(\sigma - 1 + 2\varphi^2)/[\delta^2(2+\varphi^2/2\varepsilon) + \sigma - 1]| \\ 1/2 \le \sigma \le 1: & \rho_D \simeq |\lambda(\varphi,\pi/2)| \simeq |(\sigma + 2\varphi)/[\delta^2(1+\varphi^2/2\varepsilon) + \sigma - 2\varphi]| \end{array} \qquad (4.8.22)$$

with $\varphi = 2\pi/n_1$. These results agree approximately with Table 4.8.3. For example, for $\varepsilon = 10^{-3}$, $n_1 = 64$ Equation (4.8.22) gives $\rho_D \simeq 0.152$ for $\sigma = 0$, and $\rho_D \simeq 0.103$ for $\sigma = 0.5$. Table 4.8.3 includes the worst case for β in the set $\{\beta = k\pi/12,\ k = 0, 1, 2, ..., 23\}$. Equations (4.8.21) and (4.8.22) and Table 4.8.3 show that the boundary conditions may have an important influence. For rotation angle $\beta = 0$ or $\beta = 90^0$, seven-point ILU is a good smoother for

ε	σ	ρ $\beta = 0^0$	ρ $\beta = 90^0$	ρ, β	ρ_D $\beta = 0^0$	ρ_D $\beta = 90^0$	ρ_D, β
1	0	0.13	0.13	0.13, any	0.12	0.12	0.12, any
10^{-1}	0	0.17	0.27	0.45, 75^0	0.16	0.27	0.44, 75^0
10^{-2}	0	0.17	0.61	1.35, 75^0	0.11	0.45	1.26, 75^0
10^{-3}	0	0.17	0.84	1.69, 75^0	0.02	0.16	1.55, 75^0
10^{-5}	0	0.17	0.98	1.74, 75^0	10^{-4}	0.002	1.59, 75^0
1	0.5	0.11	0.11	1.11, any	0.11	0.11	0.11, any
10^{-1}	0.5	0.089	0.23	0.50, 60^0	0.087	0.23	0.50, 60^0
10^{-2}	0.5	0.091	0.27	0.77, 60^0	0.075	0.25	0.77, 60^0
10^{-3}	0.5	0.091	0.31	0.82, 60^0	0.029	0.097	0.82, 60^0
10^{-5}	0.5	0.086	0.33	0.83, 60^0	4×10^{-4}	10^{-3}	0.82, 60^0

Table 4.8.3: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according (4.5.6); seven-point ILU smoothing; $n = 64$

the anisotropic diffusion equation. With $\sigma = 0.5$ we have a robust smoother; finer sampling of β and increasing n gives results indicating that ρ and ρ_D are bounded away from 1. For some values of β this smoother is not, however, very effective. One might try other values of σ to diminish ρ_D. A more efficient and robust ILU type smoother will be introduced shortly. In [141] it is shown that for $\beta = 45^0$ and $\varepsilon \ll 1$

$$\rho \simeq \max\left\{\left|\frac{\sigma-1}{\sigma+1}\right|, \left|\frac{\sigma}{\sigma+1}\right|\right\} \tag{4.8.23}$$

Hence, the optimal value of σ for this case is $\sigma = 0.5$, for which $\rho \simeq 1/3$.

Convection-diffusion equation

Table 4.8.4 gives some results for the convection-diffusion equation. The worst case for β is the set $\{\beta = k\pi/12 : \ k = 0, 1, 2, ..., 23\}$ is listed. It is found numerically that $\rho \ll 1$ and $\rho_D \ll 1$ when $\varepsilon \ll 1$, except for β close to 0^0 or 180^0, where ρ and ρ_D are found to be larger than for other values of β, which may spell trouble. We, therefore, do some analysis. Numerically it is found that for $\varepsilon \ll 1$ and $|s| \ll 1$ we have $\rho \simeq |\lambda(0, \pi/2)|$, both for $\sigma = 0$ and $\sigma = 1/2$. We proceed to determine $\lambda(0, \pi/2)$. Assume $c < 0$, $s > 0$; then (4.5.11) gives $a = -\varepsilon - hs$, $b = 0$, $c = -\varepsilon$, $d = 4\varepsilon - ch + sh$, $q = -\varepsilon + hc$, $f = 0$, $g = -\varepsilon$. Equations (4.8.15) and (4.8.16) give, assuming $\varepsilon \ll 1$, $|s| \ll 1$ and keeping only leading terms in ε and s, $\beta \simeq (\varepsilon + sh)ch/\delta$, $\gamma \simeq -\varepsilon$, $\mu \simeq ch$, $\zeta \simeq 0$, $\delta \simeq (s-c)h$, $p_1 \simeq (\varepsilon + sh)c^2/(s-c)^2$, $p_2 = 0$. Substitution in (4.8.19) and neglect of a few higher order terms results in

$$\lambda(0, \pi/2) \simeq \frac{(\sigma - i)(\tau + 1)}{(\tau + 2)(1 - 2\tan\beta) + \sigma(1 + \tau) + i(1 - 2\tau\tan\ \beta)} \tag{4.8.24}$$

where $\tau = sh/\varepsilon$, so that

	$\sigma = 0$			$\sigma = 0.5$		
ε	ρ	ρ_D	β	ρ	ρ_D	β
1	0.13	0.12	90^0	0.11	0.11	0^0
10^{-1}	0.13	0.13	90^0	0.12	0.12	0^0
10^{-2}	0.16	0.16	0^0	0.17	0.17	165^0
10^{-3}	0.44	0.43	165^0	0.37	0.37	165^0
10^{-5}	0.58	0.54	165^0	0.47	0.47	165^0

Table 4.8.4: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); seven-point ILU smoothing; $n = 64$

$$\rho^2 \simeq (\tau+1)^2(\sigma^2+1)/\{[(\tau+2)(1-2\tan\ \beta) + \sigma(1+\tau)]^2 + (1-2\tau\tan\ \beta)^2\} \tag{4.8.25}$$

hence,

$$\rho^2 \leq (\sigma^2 + 1)/(\sigma + 1)^2 \tag{4.8.26}$$

Choosing $\sigma = 1/2$, (4.8.26) gives $\rho \leq \frac{1}{3}\sqrt{5} \simeq 0.75$, so that the smoother is robust. With $\sigma = 0$, inequality (4.8.26) does not keep ρ away from 1. Equation (4.8.25) gives, for $\sigma = 0$:

$$\lim_{\tau \to 0} \rho = 1/\sqrt{5}, \quad \lim_{\tau \to \infty} \rho = (1 - 4 \tan\ \beta + 8 \tan^2\ \beta)^{-1/2} \tag{4.8.27}$$

This is confirmed by numerical experiments. With $\sigma = 1/2$ we have a robust smoother for the convection-diffusion equation. Alternating ILU, to be discussed shortly, may, however, be more efficient. With $\sigma = 0$, $\rho \ll 1$ except in a small neighbourhood of $\beta = 0^0$ and $\beta = 180^0$. Since in practice τ remains finite, some smoothing effect remains. For example, for $s = 0.1$ ($\beta \simeq 174.3$), $h = 1/64$ and $\varepsilon = 10^{-5}$ we have $\tau \simeq 156$ and (4.8.27) gives $\rho \simeq 0.82$. This explains why in practice seven-point ILU with $\sigma = 0$ is a satisfactory smoother for the convection-diffusion equation but $\sigma = 1/2$ gives a better smoother.

Nine-point ILU

Assume

$$[\boldsymbol{A}] = \begin{bmatrix} f & g & p \\ c & d & q \\ z & a & b \end{bmatrix} \tag{4.8.28}$$

Reasoning as before, we have

$$[\boldsymbol{L}] = \begin{bmatrix} 0 & 0 & 0 \\ \gamma & \delta & 0 \\ \omega & \alpha & \beta \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} \zeta & \eta & \tau \\ 0 & \delta & \mu \\ 0 & 0 & 0 \end{bmatrix} \tag{4.8.29}$$

For $\omega, \alpha, ..., \tau$ we have equations (4.4.22) in [141], here interpreted as equations for scalar unknowns. The relevant solution of three equations may be obtained as the limit of the following recursion

$$\begin{gathered} \alpha_0 = a, \quad \beta_0 = b, \quad \gamma_0 = c, \quad \delta_0 = d, \quad \mu_0 = q, \quad \zeta_0 = f, \quad \eta_0 = g \\ \alpha_{j+1} = a - z\mu_j/\delta_j, \quad \beta_{j+1} = b - a_{j+1}\mu_j/\delta_j \\ \gamma_{j+1} = c - (z\eta_j + \alpha_{j+1}\zeta_j)/\delta_j \\ n_{j+1} = \{|\beta_{j+1}\mu_j| + |z\zeta_j| + |\beta_{j+1}p| + |\gamma_{j+1}\zeta_j|\}/\delta_j \\ \delta_{j+1} = d - (zp + \alpha_{j+1}\eta_j + \beta_{j+1}\zeta_j + \gamma_{j+1}\mu_j)/\delta_j + \sigma n_{j+1} \\ \mu_{j+1} = q - (\alpha_{j+1}p + \beta_{j+1}\eta_j)/\sigma_{j+1} \\ \zeta_{j+1} = f - \gamma_{j+1}\eta_j/\delta_j, \quad \eta_{j+1} = g - \gamma_{j+1}p/\delta_{j+1} \end{gathered} \tag{4.8.30}$$

For $\boldsymbol{M}$ we find $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U} = \boldsymbol{A} + \boldsymbol{N}$, with

$$\boldsymbol{N} = \frac{1}{8} \begin{bmatrix} \gamma\zeta & 0 & 0 & 0 & \\ z\zeta & 0 & \sigma n & 0 & \beta p \\ & 0 & 0 & 0 & \beta\mu \end{bmatrix} \tag{4.8.31}$$

with $n = |\gamma\zeta| + |z\zeta| + |\beta p| + |\beta\mu|$. The amplification factor is given by

$$\lambda(\theta) = B(\theta)/\{(B(\theta) + A(\theta)\} \tag{4.8.32}$$

where

$$\begin{aligned} B(\theta) = \{&\gamma\zeta \exp\,[i(\theta_2 - 2\theta_1)] + z\zeta \exp\,(-2i\theta_1) \\ &+\beta p \exp\,(2i\theta_1) + \beta\mu \exp\,[i(2\theta_1 - \theta_2] + \sigma n\}/\delta \end{aligned}$$

and

$$\begin{aligned} A(\theta) = &\, z \exp\,[-i(\theta_1 + \theta_2)] + a \exp\,(-i\theta_2) + b \exp[i(\theta_1 - \theta_2)] + c \exp\,(-i\theta_1) \\ &+ d + q \exp\,(i\theta_1) + f \exp\,[i(\theta_2 - \theta_1)] + g \exp\,(i\theta_2) + p \exp\,[i(\theta_1 + \theta_2)] \end{aligned}$$

Anisotropic diffusion equation

For the anisotropic diffusion equation discretized according to (4.5.6) the nine-point ILU factorization is identical to the seven-point ILU factorization. Table 4.8.5 gives results for the case that the mixed derivative is discretized according to (4.5.8). In this case seven-point ILU performs poorly. When the mixed derivative is absent ($\beta = 0^0$ or $\beta = 90^0$) nine-point ILU is identical to seven-point ILU. Therefore Table 4.8.5 gives only the worst case for β in the set $\{\beta = k/2\pi,\ k = 0, 1, 2, ..., 23\}$. Clearly, the smoother is not robust for $\sigma = 0$. But also for $\sigma = 1/2$ there are values of β for which this smoother is not very effective. For example, with finer sampling of β around 75^0 one finds a local maximum of approximately $\rho_D = 0.73$ for $\beta = 85^0$.

	$\sigma = 0$				$\sigma = \frac{1}{2}$			
ε	ρ	β	ρ_D	β	ρ	β	ρ_D	β
1	0.13	any	0.12	any	0.11	any	0.11	any
10^{-1}	0.52	75^0	0.50	75^0	0.42	75^0	0.42	60^0
10^{-2}	1.51	75^0	1.34	75^0	0.63	75^0	0.63	75^0
10^{-3}	1.87	75^0	1.62	75^0	0.68	75^0	0.68	75^0
10^{-5}	1.92	75^0	1.66	75^0	0.68	75^0	0.68	75^0

Table 4.8.5: Fourier smoothing factors ρ, ρ_D for the rotated anisoptropic diffusion equation discretized according to (4.5.6), but the mixed derivative discretized according to (4.5.8); nine-point ILU smoothing; $n = 64$

Alternating seven-point ILU

The amplification factor of the second part (corresponding to the second backward grid point ordering defined by (3.4.16)) of alternating seven-point ILU smoothing, with factors denoted by $\bar{\boldsymbol{L}}, \bar{\boldsymbol{D}}, \bar{\boldsymbol{U}}$, may be determined as follows. Let $[\boldsymbol{A}]$ be given by (4.8.13). The stencil representation of the incomplete factorization discussed in Section 3.4 is

$$[\bar{\boldsymbol{L}}] = \begin{bmatrix} 0 & \bar{\gamma} & \\ 0 & \bar{\delta} & \bar{\alpha} \\ & 0 & \bar{\beta} \end{bmatrix}, \quad [\bar{\boldsymbol{D}}] = \begin{bmatrix} 0 & 0 & \\ 0 & \bar{\delta} & 0 \\ & 0 & 0 \end{bmatrix}, \quad [\bar{\boldsymbol{U}}] = \begin{bmatrix} \bar{\zeta} & 0 & \\ \bar{\eta} & \bar{\delta} & 0 \\ & \bar{\mu} & 0 \end{bmatrix} \tag{4.8.33}$$

In [141] it is shown that $\bar{\alpha}, \bar{\beta}, ..., \bar{\eta}$ are given by (4.8.15) and (4.8.16), provided the following substitutions are made:

$$a \to q, \quad b \to b, \quad c \to g, \quad d \to d, \quad q \to a, \quad f \to f, \quad g \to c \tag{4.8.34}$$

The iteration matrix is $\bar{\boldsymbol{M}} = \bar{\boldsymbol{L}}\bar{\boldsymbol{D}}^{-1}\bar{\boldsymbol{U}} = \boldsymbol{A} + \bar{\boldsymbol{N}}$. According to [141],

$$[\bar{\boldsymbol{N}}] = \begin{bmatrix} \bar{p}_2 & & \\ 0 & 0 & 0 \\ 0 & \bar{p}_3 & 0 \\ 0 & 0 & 0 \\ & & \bar{p}_1 \end{bmatrix} \tag{4.8.35}$$

with $\bar{p}_1 = \bar{\beta}\bar{\mu}/\bar{\delta}$, $\bar{p}_2 = \bar{\gamma}\bar{\zeta}/\bar{\delta}$, $\bar{p}_3 = \sigma(|\bar{p}_1| + |\bar{p}_2|)$. It follows that the amplification factor $\bar{\lambda}(\theta)$ of the second step of alternating seven-point ILU smoothing is given by

$$\begin{aligned} \bar{\lambda}(\theta) \quad = \quad & \{\bar{p}_3 + \bar{p}_1 \exp\,[i(\theta_1 - 2\theta_2)] + \bar{p}_2 \exp\,[i(2\theta_2 - \theta_1)]\}/ \\ & \{a \exp\,(-i\theta_2) + b \exp\,[i(\theta_1 - \theta_2)] + c \exp\,(i\theta_1) + d + \bar{P}_3 + q \exp\,(i\theta) \\ & + f \exp\,[-i(\theta_1 - \theta_2)] + g \exp\,(i\theta_2) + \bar{p}_1 \exp\,[i(\theta_1 - 2\theta_2)] \\ & + \bar{p}_2 \exp\,[i(2\theta_2 - \theta_1)\} \end{aligned} \tag{4.8.36}$$

The amplification factor of alternating seven-point ILU is given by $\lambda(\theta)\bar{\lambda}(\theta)$, with $\lambda(\theta)$ given by (4.8.19).

Anisotropic diffusion equation

Table 4.8.6 gives some results for the rotated anisotropic diffusion equation. The worst case for β in the set $\{\beta = k\pi/12,\ k = 0, 1, 2, ..., 23\}$ is included. We see that with $\sigma = 0.5$ we have a robust smoother for this test case. Similar results (not given here) are obtained when the mixed derivative is approximated by (4.5.8) with alternating nine-point ILU.

ε	σ	ρ $\beta = 0^0, 90^0$	ρ_D $\beta = 0^0, 90^0$	ρ, ρ_D	β
1	0	9×10^{-3}	9×10^{-3}	9×10^{-3}	any
10^{-1}	0	0.021	0.021	0.061	30^0
10^{-2}	0	0.041	0.024	0.25	45^0
10^{-3}	0	0.057	3×10^{-3}	0.61	45^0
10^{-5}	0	0.064	10^{-6}	0.94	45^0
1	0.5	4×10^{-3}	4×10^{-3}	4×10^{-3}	any
10^{-1}	0.5	0.014	0.014	0.028	15^0
10^{-2}	0.5	0.20	0.012	0.058	45^0
10^{-3}	0.5	0.026	2×10^{-3}	0.090	45^0
10^{-5}	0.5	0.028	0	0.11	45^0

Table 4.8.6: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to (4.5.6); alternating seven-point ILU smoothing; $n = 64$

Convection-diffusion equation

Symmetry considerations imply that the second step of alternating seven-point ILU smoothing has, for $\varepsilon \ll 1, \rho \simeq 1$ for β around 90^0 and 270^0. Here, however, the first step has $\rho \ll 1$. Hence, we expect the alternating smoother to be robust for the convection-diffusion equation. This is confirmed by the results of Table 4.8.7. The worst case for β in the set $\{\beta = k\pi/12 : k = 0, 1, 2, ..., 23\}$ is listed.

To sum up, alternating modified point ILU is robust and very efficient in all cases. The use of alternating ILU has been proposed in [97].

	$\sigma = 0$		$\sigma = 0.5$	
ε	ρ, ρ_D	β	ρ, ρ_D	β
1.0	9×10^{-3}	0^0	4×10^{-3}	0^0
10^{-1}	9×10^{-3}	0^0	4×10^{-3}	0^0
10^{-2}	0.019	105^0	7×10^{-3}	0^0
10^{-3}	0.063	105^0	0.027	120^0
10^{-5}	0.086	105^0	0.036	105^0

Table 4.8.7: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); alternating seven-point ILU smoothing; $n = 64$

Modification has been analyzed and tested in [65], [97], [83], [82], [145] and [147].

4.9 Incomplete block factorization smoothing

For the smoothing analysis of incomplete block factorization we refer to [141]. We present some results.

Anisotropic diffusion equation
Tables 4.9.1 and 4.9.2 give results for the two discretizations (4.5.6) and (4.5.8) of the rotated anisotropic diffusion equation. The worst cases for β in the set $\{\beta = k\pi/12,\ k = 0, 1, ..., 23\}$ are included. In cases where the elements of $\bar{\boldsymbol{D}}$ do not settle down quickly to values independent of location as one moves away from the grid boundaries, so that in these cases Fourier smoothing analysis is not realistic.

ε	ρ $\beta = 0^0$	ρ $\beta = 90^0$	ρ, β	ρ_D $\beta = 0^0$	ρ_D $\beta = 90^0$	ρ_D, β
1	0.058	0.058	0.058, any	0.056	0.056	0.056, any
10^{-1}	0.108	0.133	$0.133, 90^0$	0.102	0.116	$0.116, 90^0$
10^{-2}	0.149	0.176	$0.131, 45^0$	0.195	0.078	$0.131, 45^0$
10^{-3}	0.164^*	0.194	$0.157^*, 45^0$	0.025^*	0.005	$0.157^*, 45^0$
10^{-5}	0.141	0.120	$0.166^*, 45^0$	0^0	0	$0.166^*, 45^0$

Table 4.9.1: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to (4.5.6); IBLU smoothing; $n = 64$. The symbol * indicates that the coefficients do not become constant rapidly away from the boundaries; therefore the corresponding value is not realistic.

Convection-diffusion equation
Table 4.9.3 gives results for the convection-diffusion equation, sampling β as before.
If is clear that IBLU is an efficient smoother for all cases. This is confirmed by the multigrid results presented in [107].

4.10 Fourier analysis of white-black and zebra Gauss-Seidel smoothing

The Fourier analysis of white-black and zebra Gauss-Seidel smoothing requires special treatment, because the Fourier modes $\psi(\theta)$ as defined in Section 4.3 are not invariant under these iteration methods. The Fourier analysis of these methods is discussed in detail in [112]. They use sinusoidal Fourier modes. The resulting analysis is applicable only to special cases of the

ε	ρ $\beta = 0^0$	ρ $\beta = 90^0$	ρ_D $\beta = 0^0$	ρ_D $\beta = 90^0$
1	0.058	0.058	0.056	0.056
10^{-1}	0.108	0.133	0.102	0.116
10^{-2}	0.49	0.176	0.096	0.078
10^{-3}	0.164*	0.194	0.025*	5×10^{-3}
10^{-5}	0.141*	0.200	0.000*	0.000

Table 4.9.2: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to (4.5.6) but with mixed derivative according to (4.5.8); IBLU smoothing; $n = 64$. The symbol * has the same meaning as in the preceding table.

ε	ρ	β	ρ_D	β
1.0	0.058	0^0	0.056	0^0
10^{-1}	0.061	0^0	0.058	0^0
10^{-2}	0.092	0^0	0.090	0^0
10^{-3}	0.173	0^0	0.121	0^0
10^{-5}	0.200	0^0	10^{-3}	15^0

Table 4.9.3: Fourier smoothing factors ρ, ρ_D for the convection-diffusion equation discretized according to (4.5.11); IBLU smoothing; $n = 64$.

set of test problems defined in Section 4.5. Therefore we will continue to use exponential Fourier modes.

The amplification matrix
Specializing to two dimensions and assuming n_1 and n_2 to be even, we have

$$\psi_j(\theta) = \exp\,(ij\theta) \tag{4.10.1}$$

with

$$j = (j_1, j_2), \quad j_\alpha = 0, 1, 2, ..., n_\alpha - 1 \tag{4.10.2}$$

and

$$\theta \in \Theta = \{(\theta_1, \theta_2),\ \theta_\alpha = 2\pi k_\alpha / n_\alpha,\ k_\alpha = -m_\alpha, -m_\alpha + 1, ..., m_\alpha + 1\} \tag{4.10.3}$$

where $m_\alpha = n_\alpha/2 - 1$. Define

$$\begin{array}{c} \theta^1 \in \Theta_{\bar{s}} \equiv \Theta \cap [-\pi/2, \pi/2)^2 \,, \quad \theta^2 = \theta^1 - \begin{pmatrix} sign(\theta_1^1)\pi \\ sign(\theta_2^1)\pi \end{pmatrix} \\ \theta^3 = \theta^1 - \begin{pmatrix} 0 \\ sign(\theta_2^1)\pi \end{pmatrix}, \quad \theta^4 = \theta^1 - \begin{pmatrix} sign(\theta_1^1)\pi \\ 0 \end{pmatrix} \end{array} \tag{4.10.4}$$

where $sign(t) = -1,\ t \leq 0$; $sign(t) = 1,\ t > 0$. Note that $\Theta_{\bar{s}}$ almost coincides with the set of smooth wavenumbers Θ_s defined by (4.4.20). As we will see, $Span\ \{\psi(\theta^1),\ \psi(\theta^2),\ \psi(\theta^3),\ \psi(\theta^4)\}$ is left invariant by the smoothing methods considered in this section.
Let $\boldsymbol{\psi}(\theta) = (\psi(\theta^1),\ \psi(\theta^2),\ \psi(\theta^3),\ \psi(\theta^4))^T$. the Fourier representation of an arbitrary periodic grid function (4.3.7) can be written as

$$u_j = \sum_{\theta \in \Theta_{\bar{s}}} c_\theta^T \boldsymbol{\psi}_j(\theta) \tag{4.10.5}$$

with c_θ a vector of dimension 4.

If the error before smoothing is $c_\theta^T \boldsymbol{\psi}(\theta)$, then after smoothing it is given by $(\boldsymbol{\Lambda}(\theta) c_\theta)^T \boldsymbol{\psi}(\theta)$, with $\boldsymbol{\Lambda}(\theta)$ a 4×4 matrix, called the amplification matrix.

The smoothing factor
The set of smooth wavenumbers Θ_s has been defined by (4.4.20). Comparison with $\Theta_{\bar{s}}$ as defined by (4.10.4) shows that $\psi(\theta^k),\ k = 2, 3, 4$ are rough Fourier modes, whereas $\psi(\theta^1)$ is smooth, except when $\theta_1^1 = -\pi/2$ or $\theta_2^1 = -\pi/2$. The projection operator on the space spanned by the rough Fourier modes is, therefore, given by the following diagonal matrix

$$\boldsymbol{Q}(\theta) = \begin{pmatrix} \delta(\theta) & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \tag{4.10.6}$$

with $\delta(\theta) = 1$ if $\theta_1 = -\pi/2$ and $\theta_2 = -\pi/2$, and $\delta(\theta) = 0$ otherwise. Hence, a suitable definition of the Fourier smoothing factor is

$$\rho = \max\{\chi(\boldsymbol{Q}(\theta)\boldsymbol{\Lambda}(\theta)) : \theta \in \Theta_{\bar{s}}\} \tag{4.10.7}$$

with χ the spectral radius.
The influence of Dirichlet boundary conditions can be taken into account heuristically in a similar way as before. Wavenumbers of the type $(0, \theta_2^s)$ and $(\theta_1^s, 0)$, $s = 1, 3, 4$, are to be

disregated (note that $\theta_\alpha^2 = 0$ cannot occur), that is, the corresponding elements of c_θ are to be replaced by zero. This can be implemented by replacing $\boldsymbol{Q\Lambda}$ by $\boldsymbol{PQ\Lambda}$ with

$$\boldsymbol{P}(\theta) = \begin{pmatrix} p_1(\theta) & & & \\ & 1 & & 0 \\ 0 & & p_3(\theta) & \\ & & & p_4(\theta) \end{pmatrix} \tag{4.10.8}$$

where $p_1(\theta) = 0$ if $\theta_1 = 0$ and/or $\theta_2 = 0$, and $p_1(\theta) = 1$ otherwise; $p_3(\theta) = 0$ if $\theta_1 = 0$ (hence $\theta_1^3 = 0$), and $p_3(\theta) = 1$ otherwise; similarly, $p_4(\theta) = 0$ if $\theta_2 = 0$ (hence $\theta_2^4 = 0$), and $p_4(\theta) = 1$ otherwise. The definition of the smoothing factor in the case of Dirichlet boundary conditions can now be given as

$$\rho_D = \max\left\{\chi(\boldsymbol{P}(\theta)\boldsymbol{Q}(\theta)\boldsymbol{\Lambda}(\theta)) : \theta \in \Theta_{\bar{s}}\right\} \tag{4.10.9}$$

Analogous to (4.4.23) a mesh-size independent smoothing factor $\bar{\rho}$ is defined as

$$\bar{\rho} = \sup\{\chi(\boldsymbol{Q}(\theta)\boldsymbol{\Lambda}(\theta)) : \theta \in \bar{\Theta}_s\} \tag{4.10.10}$$

with $\bar{\Theta}_s = (-\pi/2, \pi/2)^2$.

White-black Gauss-Seidel

Let $\boldsymbol{A}$ have the five-point stencil given by (4.8.1) with $b = f = 0$. The use of white-black Gauss-Seidel makes no sense for the seven-point stencil (4.8.1) or the nine-point stencil (4.8.28), since the unknowns in points of the same colour cannot be updated independently. For these stencil multi-coloured Gass-Seidel can be used, but we will not go into this.

Define grid points (j_1, j_2) with $j_1 + j_2$ even to be white and the remainder black. We will study white-black Gauss-Seide with damping. Let $\boldsymbol{\varepsilon}^0$ be the initial error , $\boldsymbol{\varepsilon}^{1/3}$ the error after the white step, $\boldsymbol{\varepsilon}^{2/3}$ the error after the black step, and $\boldsymbol{\varepsilon}^1$ the error after damping with parameter ω. Then we have

$$\begin{array}{rcll} \varepsilon_j^{1/3} & = & -(a\varepsilon_{j-e_2}^0 + c\varepsilon_{j-e_1}^0 + q\varepsilon_{j+e_1}^0 + g\varepsilon_{j+e_2}^0)/d, & j_1 + j_2 \quad \text{even} \\ \varepsilon_j^{1/3} & = & \varepsilon_j^0, & j_1 + j_2 \quad \text{odd.} \end{array} \tag{4.10.11}$$

The relation between $\boldsymbol{\varepsilon}^{2/3}$ and $\boldsymbol{\varepsilon}^{1/3}$ is obtained from (4.10.11) by interchanging even and odd. The final error $\boldsymbol{\varepsilon}^1$ is given by

$$\varepsilon_j^1 = \omega\varepsilon_j^{2/3} + (1-\omega)\varepsilon_j^0 \tag{4.10.12}$$

Let the Fourier representation of $\boldsymbol{\varepsilon}^\alpha$, $\alpha = 0, 1/3, 2/3, 1$ be given by

$$\varepsilon_j^\alpha = \sum_{\theta \in \Theta_{\bar{s}}} c_\theta^{\alpha T} \boldsymbol{\psi}_j(\theta) .$$

If $\varepsilon_j^0 = \psi_j(\theta^s),\ s = 1,2,3$ or 4, then

$$\begin{aligned} \varepsilon_j^{1/3} &= \mu(\theta^s)\psi_j(\theta^s), \quad j_1 + j_2 \quad \text{even} \\ \varepsilon_j^{1/3} &= \psi_j(\theta^s), \qquad\qquad j_1 + j_2 \quad \text{odd} \end{aligned} \tag{4.10.13}$$

with $\mu(\theta) = -[a\exp(-i\theta_2) + c\exp(-i\theta_1) + q\exp(i\theta_1) + g\exp(i\theta_2)/d$. Hence

$$\begin{aligned} \varepsilon_j^{1/3} = &\frac{1}{2}(\mu(\theta^s)+1)\exp(ij\theta^s) + \frac{1}{2}(\mu(\theta^s)-1) \\ &\times \exp[ij_1(\theta_1^s - \pi)]\exp[ij_2(\theta_2^s - \pi)] \end{aligned} \tag{4.10.14}$$

so that

$$c_\theta^{1/3} = \frac{1}{2}\begin{pmatrix} 1+\mu_1 & -1-\mu_1 & 0 & 0 \\ \mu_1 - 1 & 1-\mu_1 & 0 & 0 \\ 0 & 0 & 1+\mu_2 & -1-\mu_2 \\ 0 & 0 & \mu_2 - 1 & 1-\mu_2 \end{pmatrix} c_\theta^0\,, \quad \theta \in \Theta_{\bar{s}} \tag{4.10.15}$$

where $\mu_1 = \mu(\theta),\ \mu_2 = (a\exp(-i\theta_2) - c\exp(-i\theta_1) - q\exp(i\theta_1) + g\exp(i\theta_2))/d$. If the black step is treated in a similar way one finds, combining the two steps and incorporating the damping step,

$$c_\theta^1 = \{\omega \boldsymbol{\Lambda}(\theta) + (1-\omega)\boldsymbol{I}\}c_\theta^0 \tag{4.10.16}$$

with

$$\boldsymbol{\Lambda}(\theta) = \frac{1}{2}\begin{pmatrix} \mu_1(1+\mu_1) & -\mu_1(1+\mu_1) & 0 & 0 \\ \mu_1(1-\mu_1) & \mu_1(\mu_1-1) & 0 & 0 \\ 0 & 0 & \mu_2(1+\mu_2) & -\mu_2(1+\mu_2) \\ 0 & 0 & \mu_2(1-\mu_2) & \mu_2(\mu_2-1) \end{pmatrix} \tag{4.10.17}$$

Hence

$$\begin{aligned} &\boldsymbol{P}(\theta)\boldsymbol{Q}(\theta)\boldsymbol{\Lambda}(\Theta) \\ &= \frac{1}{2}\begin{pmatrix} p_1\delta\mu_1(1+\mu_1) & -p_1\delta\mu_1(1+\mu_1) & 0 & 0 \\ \mu_1(1-\mu_1) & \mu_1(\mu_1-1) & 0 & 0 \\ 0 & 0 & p_3\mu_2(1+\mu_2) & -p_3\mu_2(1+\mu_2) \\ 0 & 0 & p_4\mu_2(1-\mu_2) & p_4\mu_2(\mu_2-1) \end{pmatrix} \end{aligned} \tag{4.10.18}$$

The eigenvalues of $\boldsymbol{PQ\Lambda}$ are

$$\begin{aligned} &\lambda_1(\theta) = 0, \quad \lambda_2(\theta) = \frac{1}{2}\mu_1\{\mu_1 - 1 + p_1\delta(1+\mu_1)\}, \\ &\lambda_3(\theta) = 0, \quad \lambda_4(\theta) = \frac{1}{2}\mu_2[p_3 - p_4 + \mu_2(p_3+p_4)\} \end{aligned} \tag{4.10.19}$$

and the two types of Fourier smoothing factor are found to be

$$\rho, \rho_D = \max \{|\omega\lambda_2(\theta) + 1 - \omega|,\ |\omega\lambda_4(\theta) + 1 - \omega| : \theta \in \Theta_{\bar{s}}\} \tag{4.10.20}$$

where $p_1 = p_3 = p_4 = 1$ in (4.10.19) gives ρ, and choosing $p_1,\ p_3,\ p_4$ as defined after equation (4.10.8) gives ρ_D.

With $\omega = 1$ we have $\bar{\rho} = \bar{\rho}_D = 1/4$ for Laplace's equation [112]. This is better than lexicographic Gauss-Seidel, for which $\bar{\rho} = 1/2$ (Section 4.7). Furthermore, obviously, white-black Gauss-Seidel lends itself very well for vectorized and parallel computing. This fact, combined with the good smoothing properties for the Laplace equation, has led to some of the fastest Poisson solvers in existence, based on multigrid with white-black smoothing [13], [113].

Convection-diffusion equation

With $\beta = 0$ equation (4.5.11) gives $a = -\varepsilon,\ c = -\varepsilon - h,\ d = 4\varepsilon + h,\ q = -\varepsilon,\ g = -\varepsilon$, so that $\mu_{1,2}(0, -\pi/2) = (2+P)/(4+P)$, with $P = h/\varepsilon$ the mesh Péclet number. Hence, with $p_1 = p_3 = p_4 = 1$ we have $\lambda_{2,4}(o, -\pi/2) = (2+P)^2/(4+P)^2$, so that $\rho \to 1$ as $P \to \infty$ for all ω, and the same is true for ρ_D. Hence white-black Gauss-seidel is not a good smoother for this test problem.

Smoothing factor of zebra Gauss-Seidel

Let $\boldsymbol{A}$ have the following nine-point stencil:

$$[\boldsymbol{A}] = \begin{bmatrix} f & g & p \\ c & d & q \\ z & a & b \end{bmatrix} \tag{4.10.21}$$

Let us consider horizontal zebra smoothing with damping. Define grid points (j_1, j_2) with j_2 even to be white and the remainder to be black. Let $\boldsymbol{\varepsilon}^0$ be the initial error, $\boldsymbol{\varepsilon}^{1/3}$ the error after the 'white' step, $\boldsymbol{\varepsilon}^{2/3}$ the error after the 'black' step, and $\boldsymbol{\varepsilon}^1$ the error after damping with parameter ω. Then we have

$$\begin{aligned} & c\varepsilon^{1/3}_{j-e_1} + d\varepsilon^{1/3}_{j} + q\varepsilon^{1/3}_{j+e_1} \\ &= -(z\varepsilon^0_{j-e_1-e_2} + a\varepsilon^0_{j-e_2} + b\varepsilon^0_{j+e_1-e_2} + f\varepsilon^0_{j-e_1+e_2} + g\varepsilon^0_{j+e_2} + p\varepsilon^0_{j+e_1+e_2}), \\ & \qquad\qquad\qquad\qquad\qquad\qquad j_2 \ \text{even} \\ \varepsilon^{1/3}_j &= \varepsilon^0_j\,, \qquad\qquad\qquad\qquad\qquad j_2 \ \text{odd} \end{aligned} \tag{4.10.22}$$

$$\tag{4.10.23}$$

where $e_1 = (1, 0)$ and $e_2 = (0, 1)$.

The relation between $\varepsilon^{2/3}$ and $\varepsilon^{1/3}$ is obtained from (4.10.23) by interchanging even and odd, and the final error ε^1 is given by (4.10.12).
It turns out that zebra iteration leaves certain two-dimensional subspaces invariant in Fourier space. In order to facilitate the analysis of alternating zebra, for which the invariant subspaces are the same as for white-black, we continue the use of the four-dimensional subspaces $\boldsymbol{\psi}(\theta)$ introduced earlier.

In [141] it is shown that the eigenvalues of $\boldsymbol{P}(\theta)\boldsymbol{Q}(\theta)\boldsymbol{\Lambda}(\theta)$ are

$$\begin{array}{c}\lambda_1(\theta)=0,\quad \lambda_2(\theta)=\frac{1}{2}p_1\delta\mu_1(1+\mu_1)-\frac{1}{2}p_3\mu_1(1-\mu_1),\quad \lambda_3(\theta)=0\\ \lambda_4(\theta)=\frac{1}{2}\mu_2(1+\mu_2)+\frac{1}{2}p_4\mu_2(\mu_2-1)\end{array}\tag{4.10.24}$$

with

$$\begin{aligned}\mu_1(\theta)=&-\quad\{z\ \exp(-i(\theta_1+\theta_2)]+a\ \exp(-i\theta_2)+b\ \exp[i(\theta_1-\theta_2)]\\ &+\quad f\ \exp[i(\theta_2-\theta_1)]+g\ \exp(\theta_2)+p\ \exp[i(\theta_1+\theta_2)\}/\\ &\qquad[c\ \exp(-i\theta_1)+d+q\ \exp(i\theta_1)]\end{aligned}$$

and $\mu_2=\mu_1(\theta_1-\pi,\ \theta_2-\pi)$.

The two types of Fourier smoothing factor are given by (4.10.20), taking λ_2,λ_4 from (4.10.24).

Anisotropic diffusion equation
For $\varepsilon=1$ (Laplace's equation), $\omega=1$ (no damping) and $p_1=p_3=p_4=1$ (periodic boundary conditions) we have $\mu_1(\theta)=\cos\ \theta_2/(2-\cos\ \theta_1)$ and $\mu_2(\theta)=-\cos\ \theta_2/(2+\cos\ \theta_1)$. One finds $\max\ \{|\lambda_2(\theta)|:\theta\in\Theta_{\bar{s}}\}=|\lambda_2(\pi/2,0)|=\frac{1}{4}$ and $\max\ \{|\lambda_4(\theta)|:\theta\in\Theta_{\bar{s}}\}=|\lambda_4(\pi/2,\pi/2)|=\frac{1}{4}$,so that the smoothing factor is $\bar{\rho}=\rho=\frac{1}{4}$.
For $\varepsilon\ll 1$ and the rotation angle $\beta=0$ we have strong coupling in the vertical direction, so that horizontal zebra smoothing is not expected to work. We have $\mu_2(\theta)=-\cos\ \theta_2/(1+\varepsilon+\varepsilon\cos\ \theta_1)$, so that $|\lambda_4(\pi/2,0)|=(1+\varepsilon)^{-2}$, hence $\lim\limits_{\varepsilon\downarrow 0}\rho_D\geq 1$. Furthermore, with $\varphi=2\pi/n$, we have $|\lambda_4(\pi/2),\varphi)|=\cos^2\ \varphi/(1+\varepsilon)^2$, so that $\lim\limits_{\varepsilon\downarrow 0}\rho_D\geq 1-O(h^2)$. Damping does not help here. We conclude that horizontal zebra is not robust for the anisotropic diffusion equation, and the same is true for vertical zebra, of course.

Convection-diffusion equation
With convection angle $\beta=\pi/2$ in (4.5.11) we have

$$\mu_2(\theta)=[(1+P)\exp\ (-i\theta_2)+\exp\ (i\theta_2)]/(4+P+2\cos\ \theta_1)\ ,$$

where $P = h/\varepsilon$ is the mesh Péclet number. With $p_4 = 1$ (periodic boundary conditions) we have $\lambda_4 = \mu_2^2$, so that $\lambda_4(\pi/2, 0) = (2+P)^2/(4+P)^2$, and we see that $\omega\lambda_4(\pi/2, 0) + 1 - \omega \approx 1$ for $P \gg 1$, so that $\rho \gtrsim 1$ for $P \gg 1$ for all ω. Hence, zebra smoothing is not suitable for the convection-diffusion equation at large mesh Péclet number.

Smoothing factor of alternating zebra Gauss-Seidel

As we saw, horizontal zebra smoothing does not work when there is strong coupling (large diffusion coefficient or strong convection) in the vertical direction. This suggests the use of *alternating zebra*: horizontal and vertical zebra combined. Following the suggestion in [112], we will arrange alternating zebra in the following 'symmetric' way: in vertical zebra we do first the 'black' step and then the 'white' step, because this gives slightly better smoothing factors, and leads to identical results for $\beta = 0^0$ and $\beta = 90^0$. The 4×4 amplification matrix of vertical zebra is found to be

$$\boldsymbol{\Lambda}_v(\theta) = \frac{1}{2}\begin{pmatrix} \nu_1(\nu_1+1) & 0 & 0 & \nu_1(\nu_1+1) \\ 0 & \nu_2(\nu_2+1) & \nu_2(\nu_2+1) & 0 \\ 0 & \nu_2(\nu_2-1) & \nu_2(\nu_2-1) & 0 \\ \nu_1(\nu_1-1) & 0 & 0 & \nu_1(\nu_1-1) \end{pmatrix} \tag{4.10.25}$$

where

$$\begin{aligned} \nu_1(\theta) = -\{& z\exp\,[-i(\theta_1+\theta_2)] + b\exp\,[i(\theta_1-\theta_2)] + c\exp\,(-i\theta_1) \\ &+ q\exp\,(i\theta_1) + f\exp\,[i(\theta_2-\theta_1)] + p\exp\,[i(\theta_1+\theta_2)]\}/ \\ &\qquad\qquad [a\exp\,(-i\theta_2) + d + g\exp\,(i\theta_2)] \end{aligned}$$

and $\nu_2(\theta) = \nu_1(\theta_1 - \pi, \theta_2 - \pi)$. We will consider two types of damping: damping the horizontal and vertical steps separately (to be referred to as double damping) and damping only after the two steps have been completed. Double damping results in an amplification matrix given by

$$\boldsymbol{\Lambda} = \boldsymbol{PQ}[(1-\omega_d)\boldsymbol{I} + \omega_d\boldsymbol{\Lambda}_v][(1-\omega_d)\boldsymbol{I} + \omega_d\boldsymbol{\Lambda}_h] \tag{4.10.26}$$

where $\boldsymbol{\Lambda}_h$ is given in [141]. In the case of single damping, put $\omega_d = 1$ in (4.10.26) and replace $\boldsymbol{\Lambda}$ by

$$\boldsymbol{\Lambda} : (1-\omega_s)\boldsymbol{I} + \omega_s\boldsymbol{\Lambda} \tag{4.10.27}$$

The eigenvalues of the 4×4 matrix $\boldsymbol{\Lambda}$ are easily determined numerically.

Anisotropic diffusion equation

Tables 4.10.1 and 4.10.2 give results for the smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation. The worst cases for the rotation angle β in the set $\{\beta = k\pi/12,\ k = 0, 1, 2, ..., 23\}$ are included. For the results of Table 4.10.1 no damping was used. Introduction of damping ($\omega_d \neq 1$ or $\omega_s \neq 1$) gives no improvement. However, as shown by Table 4.10.2, if the mixed derivative is discretized according to (4.5.8) good results are obtained. For cases with $\varepsilon = 1$ or $\beta = 0^0$ or $\beta = 90^0$ the two discretizations are identical of course, so for these cases without damping Table 4.10.1 applies. For Table 4.10.2 β has been sampled with an

ε	ρ $\beta = 0^0, 90^0$	ρ_D $\beta = 0^0, 90^0$	ρ, ρ_D	β
1	0.048	0.048	0.048	any
10^{-1}	0.102	0.100	0.480	45^0
10^{-2}	0.122	0.121	0.924	45^0
10^{-3}	0.124	0.070	0.992	45^0
10^{-5}	0.125	0.001	1.000	45^0

Table 4.10.1: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to (4.5.6); alternating zebra smoothing; $n = 64$

	$\omega_s = 1$		$\omega_s = 0.7$		
ε	ρ, ρ_D	β	ρ, ρ_D $\beta = 0^0, 90^0$	ρ, ρ_D	β
1	0.048	any	0.317	0.317	any
10^{-1}	0.229	30^0	0.302	0.460	34^0
10^{-2}	0.426	14^0	0.300	0.598	14^0
10^{-3}	0.503	8^0	0.300	0.653	8^0
10^{-5}	0.537	4^0	0.300	0.668	8^0
10^{-8}	0.538	4^0	0.300	0.668	8^0

Table 4.10.2: Fourier smoothing factors ρ, ρ_D for the rotated anisotropic diffusion equation discretized according to(4.5.6) but with the mixed approximated by (4.5.8); alternating zebra smoothing with single damping; $n = 64$

interval of 2^0. Symmetry means that only $\beta \in [0^0, 45^0]$ needs to be considered. Results with single damping ($\omega_s = 0.7$) are included. Clearly, damping is not needed in this case and even somewhat disadvantageous. As will be seen shortly, this method, however, works for the convection diffusion test problem only if damping is applied. Numerical experiments show that a fixed value of $\omega_s = 0.7$ is suitable, and that there is not much difference between single damping and double damping. We present results only for single damping.

Convection-diffusion equation
For Table 4.10.3, β has been sampled with intervals of 2^0; the worst cases are presented. The results of Table 4.10.3 show that alternating zebra without damping is a reasonable smoother

	$\omega_s = 1$				$\omega_s = 0.7$	
ε	ρ	β	ρ_D	β	ρ, ρ_D	β
1	0.048	0^0	0.048	0^0	0.317	0^0
10^{-1}	0.049	0^0	0.049	0^0	0.318	20^0
10^{-2}	0.080	28^0	0.079	26^0	0.324	42^0
10^{-3}	0.413	24^0	0.369	28^0	0.375	44^0
10^{-5}	0.948	4^0	0.584	22^0	0.443	4^0
10^{-8}	0.995	2^0	0.587	22^0	0.448	4^0

Table 4.10.3: Fourier smoothing factors ρ for the convection-diffusion equation discretized according to (4.5.11); alternating zebra smoothing with single damping; $n = 64$

for the convection-diffusion equation. If the mesh Péclet numbers $h \cos\ \beta/\varepsilon$ or $h \sin\ \beta/\varepsilon$ becomes large (> 100, say), ρ approaches 1, but ρ_D remains reasonable.
A fixed damping parameter $\omega_s = 0.7$ gives good results also for ρ. The value $\omega_s = 0.7$ was chosen after some experimentation.

We see that with $\omega_s = 0.7$ alternating zebra is robust and reasonably efficient for both the convection-diffusion and the rotated anisotropic diffusion equation, provided the mixed derivative is discretized according to (4.5.8).

4.11 Multistage smoothing methods

As we will see, multistage smoothing methods are also of the basic iterative method type (3.1.3) (of the semi-iterative kind, as will be explained), but in the multigrid literature they are usually looked upon as techniques to solve systems of ordinary differential equations, arising from the spatial discretization of systems of hyperbolic or almost hyperbolic partial differential equations.

The convection-diffusion test problem (4.5.4) is of this type, but (4.5.3) is not. We will, therefore, consider the application of multistage smoothing to (4.5.4) only. Multistage methods have been introduced in [74] for the solution of the Euler equations of gas dynamics, and as smoothing methods in a multigrid approach in [71]. For the simple scalar test problem

(4.5.4) multistage smoothing is less efficient than the better ones of the smoothing methods discussed before. The simple test problem (4.5.4), however, lends itself well for explaining the basic principles of multistage smoothing, which is the purpose of this section.

Artificial time-derivative
The basic idea of multistage smoothing is to add a time-derivative to the equation to be solved, and to use a time-stepping method to damp the short wavelength components of the error. The time-stepping method is of multistage (Runge-Kutta) type. Damping of short waves occurs only if the discretization is dissipative, which implies that for hyperbolic or almost hyperbolic problems some form of upwind discretization must be used, or an artificial dissipation term must be added. Such measures are required anyway to obtain good solutions. The test problem (4.5.4) is replaced by

$$\frac{\partial u}{\partial t} - \varepsilon(u_{,11} + u_{,22}) + cu_{,1} + su_{,2} = f \tag{4.11.1}$$

Spatial discretization according to (4.5.10) or (4.5.11) gives a system of ordinary differential equations denoted by

$$\frac{d\boldsymbol{u}}{dt} = -h^{-2}\boldsymbol{A}\boldsymbol{u} + \boldsymbol{f} \tag{4.11.2}$$

where $\boldsymbol{A}$ is the operator defined in (4.5.10) or (4.5.11); $\boldsymbol{u}$ is the vector of grid function values.

Multistage method
The time-derivative in (4.11.2) is an artefact; the purpose is to solve $\boldsymbol{A}\boldsymbol{u} = h^2\boldsymbol{f}$. Hence, the temporal accuracy of the discretization is irrelevant. Denoting the time-level by a superscript n and stage number k by a superscript (k), a p-stage (Runge-Kutta) discretization of (4.11.2) is given by

$$\begin{aligned}
\boldsymbol{u}^{(0)} &= \boldsymbol{u}^n \\
\boldsymbol{u}^{(k)} &= \boldsymbol{u}^{(0)} - c_k\nu h^{-1}\boldsymbol{A}\boldsymbol{u}^{(k-1)} + c_k\Delta t\boldsymbol{f}, \quad k = 1, 2, ..., p \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^{(p)}
\end{aligned} \tag{4.11.3}$$

with $c_p = 1$. Here $\nu \equiv \Delta t/h$ is the so-called Courant-Frederichs-Lewy (CFL) number. Eliminating $\boldsymbol{u}^{(k)}$, this can be rewritten as

$$\boldsymbol{u}^{n+1} = P_p(-\nu h^{-1}\boldsymbol{A})\boldsymbol{u}^n + Q_{p-1}(-\nu h^{-1}\boldsymbol{A})\boldsymbol{f} \tag{4.11.4}$$

with the *amplification polynomial* P_p a polynomial of degree p defined by

$$P_p(z) = 1 + z(1 + c_{p-1}z(1 + c_{p-2}z(...(1 + c_1 z)...) \tag{4.11.5}$$

and Q_{p-1} is polynomial of degree $p-1$ which plays no role in further discussion.

Semi-iterative methods
Obviously, equation (4.11.6) can be interpreted as an iterative method for solving $h^{-2}\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ of the type introduced in Section 4.1 with iteration matrix

$$\boldsymbol{S} = P_p(-\nu h^{-1}\boldsymbol{A}) \tag{4.11.6}$$

Such methods, for which the iteration matrix is a polynomial in the matrix of the system to be solved, are called *semi-iterative methods*. See [129] for the theory of such methods. For $p = 1$ (one-stage method) we have

$$\boldsymbol{S} = \boldsymbol{I} - \nu h^{-1}\boldsymbol{A} \tag{4.11.7}$$

which is in fact the damped Jacobi method (Section 4.3) with diagonal scaling ($diag\ (\boldsymbol{A}) = \boldsymbol{I}$), also known as the one-stage Richardson method. As a solution method for differential equations this is known as the *forward Euler method*. Following the trend in the multigrid literature, we will analyse method (4.11.3) as a multistage method for differential equations, but the analysis could be couched in the language of linear algebra just as well.

The amplification factor
The time step Δt is restricted by stability. In order to assess this stability restriction and the smoothing behaviour of (4.11.4), the Fourier series (4.3.7) is substituted for $\boldsymbol{u}$. It suffices to consider only one component $\boldsymbol{u} = \boldsymbol{\psi}(\theta)$, $\theta \in \Theta$. We have $\nu h^{-1}\boldsymbol{A}\boldsymbol{\psi}(\theta) = \nu h^{-1}\mu(\theta)\boldsymbol{\psi}(\theta)$. With $\boldsymbol{A}$ defined by (4.5.11) one finds

$$\begin{aligned}\mu(\theta) \quad = \quad & 4\varepsilon + h(|c| + |s|) - (2\varepsilon + h|c|)\cos\ \theta_1 \\ & -(2\varepsilon + h|s|)\cos\ \theta_2 + ihc\sin\ \theta_1 + ihs\sin\ \theta_2\end{aligned} \tag{4.11.8}$$

and

$$\boldsymbol{u}^{n+1} = g(\theta)\boldsymbol{u}^n \tag{4.11.9}$$

with the amplification factor $g(\theta)$ given by

$$g(\theta) = P_p(-\nu\mu(\theta)/h) \tag{4.11.10}$$

The smoothing factor
The smoothing factor is defined as before:

$$\rho = \max\ \{|g(\theta)| : \theta \in \Theta_r\} \tag{4.11.11}$$

in the case of periodic boundary conditions, and

$$\rho_D = \max\{|g(\theta)| : \theta \in \Theta_r^D\} \tag{4.11.12}$$

for Dirichlet boundary conditions.

Stability condition
Stability requires that

$$|g(\theta)| \leq 1, \quad \forall \theta \in \Theta \tag{4.11.13}$$

The stability domain D of the multistage method is defined as

$$D = \{z \in C : |P_p(z) \leq 1\} \tag{4.11.14}$$

Stability requires that ν is chosen such that $z = -\nu\mu(\theta)/h \in D$, $\forall \theta \in \Theta$. If $\rho < 1$ but (4.11.13) is not satisfied, rough modes are damped but smooth modes are amplified, so that the multistage method is unsuitable.

Local time-stepping
When the coefficients c and s in the convection-diffusion equation (4.11.1) are replaced by general variable coefficients v_1 and v_2 (in fluid mechanics applications v_1, v_2 are fluid velocity components), an appropriate definition of the CFL number is

$$\nu = v\Delta t/h, \quad v = |v_1| + |v_2| \tag{4.11.15}$$

Hence, if Δt is the same in every spatial grid point, as would be required for temporal accuracy, ν will be variable if v is not constant. For smoothing purposes it is better to fix ν at some favourable value, so that Δt will be different in different grid points and on different grids in multigrid applications. This is called *local time-stepping*.

Optimization of the coefficients
The stability restriction on the CFL number ν and the smoothing factor ρ depend on the coefficients c_k. In the classical Runge-Kutta methods for solving ordinary differential equations these are chosen to optimize stability and accuracy. For analyses see for example [115], [106]. For smoothing c_k is chosen not to enhance accuracy but smoothing; smoothing is also influenced by ν. The optimum values of ν and c_k are problem dependent. Some analysis of the optimization problem involved may be found in [127]. In general, this optimization problem can only be solved numerically.

We proceed with a few examples.

A four-stage method
Based upon an analysis of Catalano and Deconinck (prive-communication), in which optimal coefficients c_k and CFL number ν are sought for the upwind discretization (4.5.11) of (4.11.1) with $\varepsilon = 0$, we choose

$$c_1 = 0.07, \quad c_2 = 0.19, \quad c_3 = 0.42, \quad \nu = 2.0 \tag{4.11.16}$$

ε	$\beta = 0^0$	$\beta = 15^0$	$\beta = 30^0$	$\beta = 45^0$
0	1.00	0.593	0.477	0.581
10^{-5}	0.997	0.591	0.482	0.587

Table 4.11.1: Smoothing factor ρ for (4.11.1) discretized according to (4.5.11); four-stage method; $n = 64$

Table 4.11.1 gives some results.

It is found that ρ_D differs very little from ρ. It is not necessary to choose β outside $[0^0, 45^0]$, since the results are symmetric in β. For $\varepsilon \gtrsim 10^{-3}$ the method becomes unstable for certain values of β. Hence, for problems in which the mesh Péclet number varies widely in the domain it would seem necessary to adopt c_k and ν to the local stencil. With $\varepsilon = 0$ all multistage smoothers have $\rho = 1$ for grid-aligned flow ($\beta = 0^0$ or 90^0) : waves perpendicular to the flow are not damped.

A five-stage method

The following method has been proposed in [73] for a central discretization of the Euler equations of gas dynamics:

$$c_1 = 1/4, \quad c_2 = 1/6, \quad c_3 = 3/8, \quad c_4 = 1/2 \tag{4.11.17}$$

The method has also been applied to the compressible Navier-Stokes equations in [75]. We will apply this method to test problem (4.11.1) with the central discretization (4.5.10). Since $\mu(\theta) = ih(c \sin\ \theta_1 + s \sin\ \theta_2)$ we have $\mu(0, \pi) = 0$, hence $|g(0, \pi)| = 1$, so that we have no smoother. An artificial dissipation term is therefore added to (4.11.2), which becomes

$$\frac{d\boldsymbol{u}}{dt} = -h^{-2}\boldsymbol{A}\boldsymbol{u} - h^{-1}\boldsymbol{B}\boldsymbol{u} + \boldsymbol{f} \tag{4.11.18}$$

with

$$[\boldsymbol{B}] = \chi \begin{bmatrix} & & 1 & & \\ & & -4 & & \\ 1 & -4 & 12 & -4 & 1 \\ & & -4 & & \\ & & 1 & & \end{bmatrix} \tag{4.11.19}$$

where χ is a parameter.

We have $\boldsymbol{B}\boldsymbol{\psi}(\theta) = \eta(\theta)\boldsymbol{\psi}(\theta)$ with

β	0^0	15^0	30^0	45^0
ρ	0.70	0.77	0.82	0.82

Table 4.11.2: Smoothing factor ρ for (4.11.1) discretized according to (4.5.10); five-stage method; $n = 64$

$$\eta(\theta) = 4\chi[(1 - \cos\ \theta_1)^2 + (1 - \cos\ \theta_2)^2] \tag{4.11.20}$$

For reasons of efficiency the artificial dissipation term is updated in [73] only in the first two stages. This gives the following five-stage method:

$$\begin{array}{rcll} \boldsymbol{u}^{(k)} & = & \boldsymbol{u}^{(0)} - c_k\nu(h^{-1}\boldsymbol{A} + \boldsymbol{B})\boldsymbol{u}^{(k-1)}, & k = 1,2 \\ \boldsymbol{u}^{(k)} & = & \boldsymbol{u}^{(0)} - c_k\nu(h^{-1}\boldsymbol{A}\boldsymbol{u}^{(k-1)} + \boldsymbol{B}\boldsymbol{u}^{(1)}, & k = 3,4,5 \end{array} \tag{4.11.21}$$

The amplification polynomial now depends on two arguments z_1, z_2 defined by $z_1 = \nu h^{-1}\mu(\theta)$, $z_2 = \nu\eta(\theta)$, and is given by the following algorithm:

$$\begin{array}{c} P_1 = 1 - c_1(z_1 + z_2), \quad P_2 = 1 - c_2(z_1 + z_2)P_1 \\ P_3 = 1 - c_3z_1P_2 - c_3z_1P_2 - c_3z_2P_1, \quad P_4 = 1 - c_4z_1P_3 - c_4z_2P_1 \\ P_5(z_1, z_2) = 1 - z_1P_4 - z_2P_1 \end{array} \tag{4.11.22}$$

In one dimension Jameson and Baker [73] advocate $\nu = 3$ and $\chi = 0.04$; for stability ν should not be much larger than 3. In two dimensions max $\{\nu h^{-1}|\mu(\theta)|\} = \nu(c+s) \leq \nu\sqrt{2}$. Choosing $\nu\sqrt{2} = 3$ gives $\nu \simeq 2.1$. With $\nu = 2.1$ and $\chi = 0.04$ we obtain the results of Table 4.11.2, for both $\varepsilon = 0$ and $\varepsilon = 10^{-5}$. Again, $\rho_D \simeq \rho$. This method allows only $\varepsilon \ll 1$; for example, for $\varepsilon = 10^{-3}$ and $\beta = 45^0$ we find $\rho = 0.96$.

Final remarks

Advantages of multistage smoothing are excellent vectorization and parallelization potential, and easy generalization to systems of differential equations. Multistage methods are in widespread use for hyperbolic and almost hyperbolic systems in computational fluid dynamics. They are not, however, robust, because, like all point-wise smoothing methods, they do not work when the unknowns are strongly coupled in one direction due to high mesh aspect ratios. Also their smoothing factors are not small. Various strategems have been proposed in the literature to improve multistage smoothing, such as residual averaging, including implicit stages, and local adaptation of c_k, but we will not discuss this here; see [73], [75] and [127].

4.12 Concluding remarks

In this chapter Fourier smoothing analysis has been explained, and efficiency and robustness of a great number of smoothing methods has been investigated by determining the smoothing factors ρ and ρ_D for the two-dimensional test problems (4.5.3) and (4.5.4). The following methods work for both problems, assuming the mixed derivative in (4.5.3) is suitably discretized, either with (4.5.6) or (4.5.8):

(i) Damped alternating Jacobi;

(ii) Alternating symmetric line Gauss-Seidel;

(iii) Alternating modified incomplete point factorization;

(iv) Incomplete block factorization;

(v) Alternating damped zebra Gauss-Seidel.

Where damping is needed the damping parameter can be fixed, independent of the problem. It is important to take the type of boundary condition into account. The heuristic way in which this has been done within the framework of Fourier smoothing analysis correlates well with multigrid convergence results obtained in practice.

Generalization of incomplete factorization to systems of differential equations and to nonlinear equations is less straightforward than for the other methods. Application to the incompressible Navier-Stokes equations has, however, been worked out in [144], [146], [148], [150] and [149], and is discussed in [141].

Of course, in three dimensions robust and efficient smoothers are more elusive than in two dimensions. Incomplete block factorization, the most powerful smoother in two dimensions, is not robust in three dimensions [81]. Robust three-dimensional smoothers can be found among methods that solve accurately in planes (plane Gauss-Seidel) [114]. For a successful multigrid approach to a complicated three-dimensional problem using ILU type smoothing, see [124], [122], [125], [123].

5 Prolongation, restriction and coarse grid approximation

5.1 Introduction

In this chapter the transfer operations between fine and coarse grids are discussed.

Fine grids
The domain Ω in which the partial differential equation is to be solved is assumed to be the d-dimensional unit cube. In the case of vertex-centered discretization, the computational grid is defined by

$$\begin{aligned} G = \{x \in I\!R^d : x = jh,\ j = (j_1, j_2, ..., j_d),\ h = (h_1, h_2, ..., h_d), \\ j_\alpha = 0, 1, 2, ..., n_\alpha,\ h_\alpha = 1/n_\alpha,\ \alpha = 1, 2, ..., d\} \end{aligned} \tag{5.1.1}$$

In the case of cell-centered discretization, G is defined by

$$\begin{aligned} G = \{x \in I\!R^d : x = (j - s)h,\ j = (j_1, j_2, ..., j_d),\ s = (1, 1, ..., 1)/2, \\ h = (h_1, h_2, ..., h_d),\ j_\alpha = 1, 2, ..., n_\alpha,\ h_\alpha = 1/n_\alpha,\ \alpha = 1, 2, ..., d\} \end{aligned} \tag{5.1.2}$$

These grids, on which the given problem is to be solved, are called fine grids. Without danger of confusion, we will also consider G to be the set of d-tuples j occuring in (5.1.1) or (5.1.2).

Coarse grids
In this chapter it suffices to consider only one coarse grid. From the vertex-centered grid (5.1.1) a coarse grid is derived by *vertex-centered coarsening*, and from the cell-centered grid (5.1.2) a coarse grid is derived by *cell-centered coarsening.* Coarse grid quantities will be identified by an overbar. Vertex-centered coarsening consists of deleting every other vertex in each direction. Cell-centered coarsening consists of taking unions of fine grid cells to obtain coarse grid cells. Figures 5.1.1 and 5.1.2 give an illustration. It is assumed that n_α in (5.1.1) and (5.1.2) is even.
Denote spaces of grid function by U:

$$U = \{\boldsymbol{u} : G \to I\!R\}, \quad \bar{U} = \{\bar{\boldsymbol{u}} : \bar{G} \to I\!R\} \tag{5.1.3}$$

The transfer operators are denoted by $\boldsymbol{P}$ and $\boldsymbol{R}$:

$$\boldsymbol{P} : \bar{U} \to U, \quad \boldsymbol{R} : U \to \bar{U} \tag{5.1.4}$$

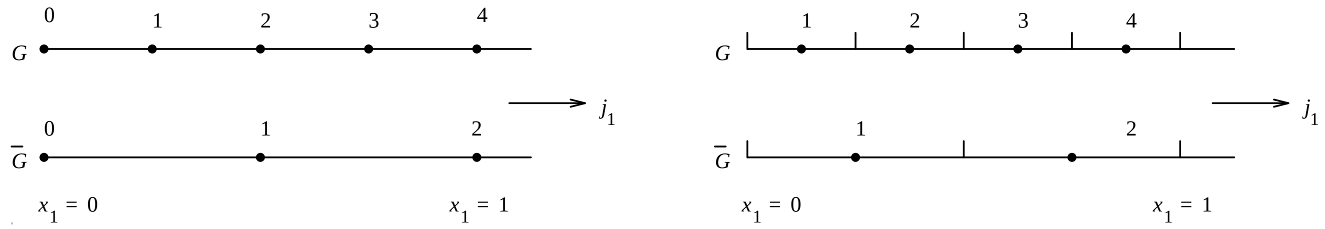


Figure 5.1.1: Vertex-centered and cell-centered coarsening in one dimension. (• grid points)

$\boldsymbol{P}$ is called *prolongation*, and $\boldsymbol{R}$ *restriction*.

Here only vertex-centered coarsening will be discussed. For cell-centered coarsening, see [141].

5.2 Stencil notation

In order to obtain a concise description of the transfer operators, *stencil notation* will be used.

Stencil notation for operators of type $U \to U$
Let $\boldsymbol{A} : U \to U$ be a linear operator. Then, using stencil notation, $\boldsymbol{Au}$ can be denoted by

$$(\boldsymbol{Au})_i = \sum_{j \in \mathbb{Z}^d} \boldsymbol{A}(i,j) u_{i+j}, \quad i \in G \tag{5.2.1}$$

with $\mathbb{Z} = \{0, \pm 1, \pm 2, ...\}$. The subscript $i = (i_1, i_2, ..., i_d)$ identifies a point in the computational grid in the usual way; cf. Figure 5.1.2 for the case $d = 2$.
The set S_A defined by

$$S_{\boldsymbol{A}} = \{ j \in \mathbb{Z}^d : \exists i \in G \quad \text{with} \quad A(i,j) \neq 0 \} \tag{5.2.2}$$

is called the *structure* of $\boldsymbol{A}$. The set of values $\boldsymbol{A}(i,j)$ with $j \in S_{\boldsymbol{A}}$ is called the *stencil* of $\boldsymbol{A}$ at grid point i. Often the word 'stencil' refers more specifically to an array of values denoted by $[\boldsymbol{A}]_i$ in which the values of $\boldsymbol{A}(i,j)$ are given; for example, in two dimensions,

$$[\boldsymbol{A}]_i = \begin{bmatrix} \boldsymbol{A}(i, -e_1 + e_2) & \boldsymbol{A}(i, e_2) & \\ \boldsymbol{A}(i, -e_1) & \boldsymbol{A}(i, 0) & \boldsymbol{A}(i, e_1) \\ & \boldsymbol{A}(i, -e_2) & \boldsymbol{A}(i, e_1 - e_2) \end{bmatrix} \tag{5.2.3}$$

where $e_1 = (1, 0)$ and $e_2 = (0, 1)$. For the representation of three-dimensional stencils, see [141].

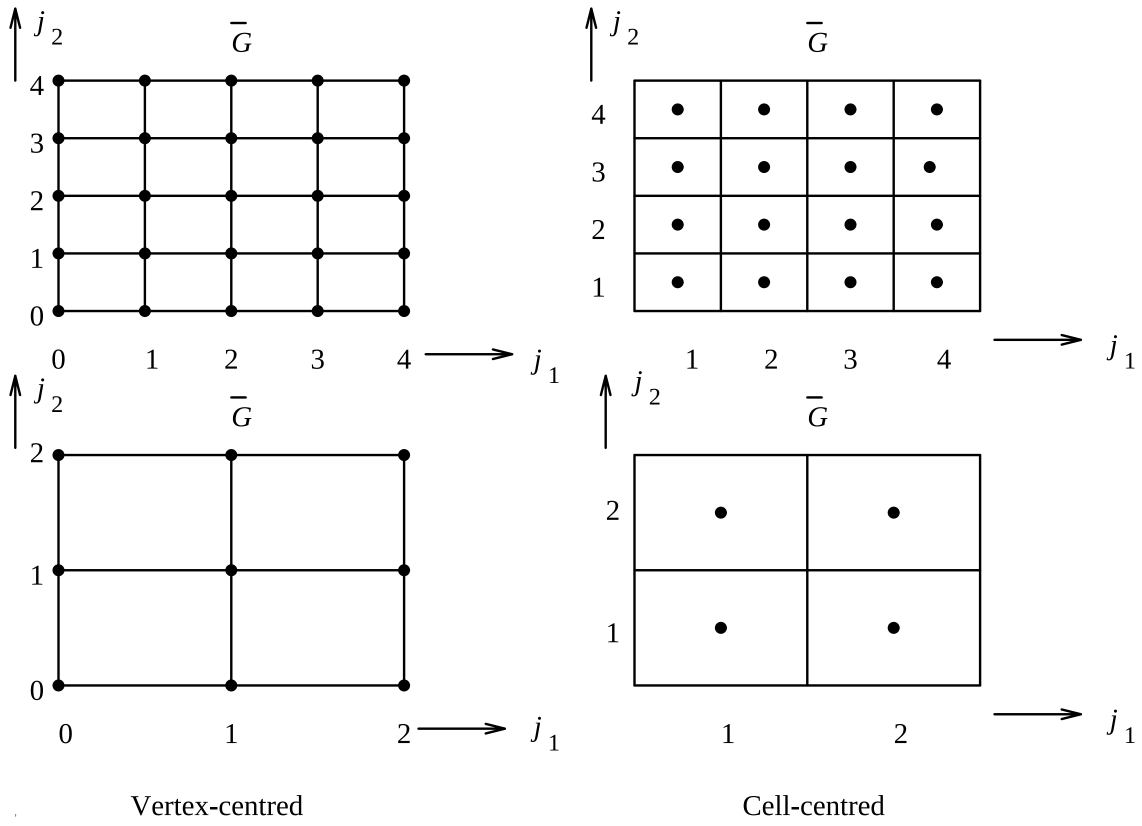


Figure 5.1.2: Vertex-centered and cell-centered coarsening in two dimensions. (• grid points.)

Example 5.2.1 One-dimensional discrete Laplacian:

$$[\boldsymbol{A}]_i = h^{-2}[-1 \quad 2 \quad -1] \tag{5.2.4}$$

Stencil notation for restriction operators
Let $\boldsymbol{R} : U \rightarrow \bar{U}$ be a restriction operator. Then, using stencil notation, $\boldsymbol{Ru}$ can be represented by

$$(\boldsymbol{Ru})_i = \sum_{j \in \mathbb{Z}^d} \boldsymbol{R}(i, j) u_{2i+j}, \quad i \in \bar{G} \tag{5.2.5}$$

Example 5.2.2 Consider vertex-centered grids $G, \bar{G}$ for $d = 1$ as defined by (5.1.1) and as depicted in Figure 5.1.1. Let $\boldsymbol{R}$ be defined by

$$\boldsymbol{R}u_i = w_i u_{2i-1} + \frac{1}{2} u_{2i} + e_i u_{2i+1}, \quad i = 0, 1, ..., n/2 \tag{5.2.6}$$

with $w_0 = 0$; $w_i = 1/4$, $i \neq 0$; $e_i = 1/4$, $i \neq n/2$; $e_{n/2} = 0$. Then we have (cf. (5.2.5)):

$$\boldsymbol{R}(i,-1) = w_i, \quad \boldsymbol{R}(i,0) = 1/2, \quad \boldsymbol{R}(i,1) = e_i \tag{5.2.7}$$

or

$$[\boldsymbol{R}]_i = [w_i \quad 1/2 \quad e_i] \tag{5.2.8}$$

We can also write $[\boldsymbol{R}] = [1 \quad 2 \quad 1]/4$ and stipulate that stencil elements that refer to values of u at points outside G are to be replaced by 0.

The relation between the stencil of an operator and that of its adjoint

For prolongation operators, a nice definition of stencil notation is less obvious than for restriction operators. As a preparation for the introduction of a suitable definition we first discuss the relation between the stencil of an operator and its adjoint. Define the *inner product* on U in the usual way:

$$(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i \in \mathbb{Z}^d} u_i v_i \tag{5.2.9}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are defined to be zero outside G. Define the *transpose* $\boldsymbol{A}^*$ of $\boldsymbol{A} : u \to U$ in the usual way by

$$(\boldsymbol{Au}, \boldsymbol{v}) = (\boldsymbol{u}, \boldsymbol{A}^*\boldsymbol{v}), \quad \forall \boldsymbol{u}, \boldsymbol{v} \in U \tag{5.2.10}$$

Defining $\boldsymbol{A}(i,j) = 0$ for $i \neq G$ or $j \neq S_A$ we can write

$$\begin{aligned}(\boldsymbol{Au}, \boldsymbol{v}) &= \sum_{i,j \in \mathbb{Z}^d} \boldsymbol{A}(i,j) u_{i+j} v_i = \sum_{i,k \in \mathbb{Z}^d} \boldsymbol{A}(i, k-i) u_k v_i \\ &= \sum_{k \in \mathbb{Z}^d} u_k \sum_{i \in \mathbb{Z}^d} \boldsymbol{A}(i, k-i) v_i = (\boldsymbol{u}, \boldsymbol{A}^*\boldsymbol{v})\end{aligned} \tag{5.2.11}$$

with

$$(\boldsymbol{A}^*\boldsymbol{v})_k = \sum_{i \in \mathbb{Z}^d} \boldsymbol{A}(i, k-i) v_i = \sum_{i \in \mathbb{Z}^d} \boldsymbol{A}(i+k, -i) v_{k+i} = \sum_{i \in \mathbb{Z}^d} \boldsymbol{A}^*(k,i) v_{k+i} \tag{5.2.12}$$

Hence, we obtain the following relation between the stencils of $\boldsymbol{A}$ and $\boldsymbol{A}^*$:

$$\boldsymbol{A}^*(k,i) = \boldsymbol{A}(k+i, -i) \tag{5.2.13}$$

Stencil notation for prolongation operators

If $\boldsymbol{R} : U \to \bar{U}$, then $\boldsymbol{R}^* : \bar{U} \to U$ is a prolongation. The stencil of $\boldsymbol{R}^*$ is obtained in similar fashion as that of $\boldsymbol{A}^*$. Defining $\boldsymbol{R}(i,j) = 0$ for $i \notin \bar{G}$ or $j \notin S_R$, we have

$$\begin{aligned}(\boldsymbol{Ru}, \bar{\boldsymbol{v}}) &= \sum_{i,j \in \mathbb{Z}^d} \boldsymbol{R}(i,j) u_{2i+j} \bar{v}_i = \sum_{i,k \in \mathbb{Z}^d} \boldsymbol{R}(i, k-2i) u_k \bar{v}_i \\ &= \sum_{k \in \mathbb{Z}^d} u_k \sum_{i \in \mathbb{Z}^d} \boldsymbol{R}(i, k-2i) \bar{v}_i = (\boldsymbol{u}, \boldsymbol{R}^*\boldsymbol{v})\end{aligned} \tag{5.2.14}$$

with $\boldsymbol{R}^* : \bar{U} \to U$ defined by

$$(\boldsymbol{R}^*\bar{\boldsymbol{v}})_k = \sum_{i \in \mathbb{Z}^d} \boldsymbol{R}(i, k-2i)\bar{v}_i \tag{5.2.15}$$

Equation (5.2.15) shows how to define the stencil of a prolongation operator $\boldsymbol{P} : \bar{U} \to U$:

$$(\boldsymbol{P}\bar{\boldsymbol{u}})_i = \sum_{j \in \mathbb{Z}^d} \boldsymbol{P}^*(j, i-2j)\bar{u}_j \tag{5.2.16}$$

Hence, a convenient way to define $\boldsymbol{P}$ is by specifying $\boldsymbol{P}^*$. Equation (5.2.16) is the desired stencil notation for prolongation operators.
Suppose a rule has been specified to determine $\boldsymbol{P}\bar{\boldsymbol{u}}$ for given $\bar{\boldsymbol{u}}$, then $\boldsymbol{P}^*(k, m)$ can be obtained as follows. Choose $\bar{\boldsymbol{u}} = \bar{\delta}^k$ as follows

$$\bar{\delta}_k^i = 1, \quad \bar{\delta}_j^k = 0, \quad j \neq k \tag{5.2.17}$$

Then (5.2.16) gives $\boldsymbol{P}^*(k, i-2k) = (\boldsymbol{P}\bar{\boldsymbol{\delta}})_i$, or

$$\boldsymbol{P}^*(k, j) = (\boldsymbol{P}\bar{\boldsymbol{\delta}}^k)_{2k+j}, \quad k \in \bar{G},\ j \in G. \tag{5.2.18}$$

In other words, $[\boldsymbol{P}^*]_k$ is precisely the image of $\bar{\boldsymbol{\delta}}^k$ under $\boldsymbol{P}$.
The usefulness of stencil notation will become increasingly clear in what follows.

Exercise 5.2.1 Verify that (5.2.13) and (5.2.15) imply that, if $\boldsymbol{A}$ and $\boldsymbol{R}$ are represented by matrices, $\boldsymbol{A}^*$ and $\boldsymbol{R}^*$ follow from $\boldsymbol{A}$ and $\boldsymbol{R}$ by interchanging rows and columns. (Remark: for $d = 1$ this is easy; for $d > 1$ this exercise is a bit technical in the case of $\boldsymbol{R}$).

Exercise 5.2.2 Show that if the matrix representation of $\boldsymbol{A} : U \to U$ is symmetric, then its stencil has the property $\boldsymbol{A}(k, i) = \boldsymbol{A}(k+i, -i)$.

5.3 Interpolating transfer operators

We begin by giving a number of examples of prolongation operators, based on interpolation.

Let $d = 1$, and let G and $\bar{G}$ be vertex-centred (cf. Figure 5.1.1). Defining $\boldsymbol{P} : \bar{U} \to U$ by linear interpolation, we have

$$(\boldsymbol{P}\bar{\boldsymbol{u}})_{2i} = \bar{u}_i, \quad (\boldsymbol{P}\bar{\boldsymbol{u}})_{2i+1} = \frac{1}{2}(\bar{u}_i + \bar{u}_{i+1}) \tag{5.3.1}$$

Using (5.3.1) we find that the stencil of $\boldsymbol{P}^*$ is given by

$$[\boldsymbol{P}^*] = \frac{1}{2}[1\ 2\ 1] \tag{5.3.2}$$

In two dimensions, *linear interpolation* is exact for functions $f(x_1, x_2) = 1, x_1, x_2$, and takes place in triangles, cf. Figure 5.3.1. Choosing triangles ABD and ACD for interpolation, one obtains $u_A = \bar{u}_A$, $u_a = \frac{1}{2}(\bar{u}_A + \bar{u}_B)$, $u_e = \frac{1}{2}(\bar{u}_A + \bar{u}_D)$ etc. Alternatively, one may choose

$$\begin{array}{ccc} C & d & D \\ b & e & c \\ A & a & B \end{array}$$

Figure 5.3.1: Interpolation in two dimensions, vertex-centered grids. (Coarse grid point: capital letters; fine grid points: capital and lower case letters.)

triangles ABC and BDC, which makes no essential difference. Bilinear interpolation is exact for functions $f(x_1, x_2) = 1, x_1, x_2, x_1x_2$, and takes place in the rectangle ABCD. The only difference with linear interpolation is that now $u_e = \frac{1}{4}(u_A + u_B + u_C + u_D)$. In other words: $u_{2i+e_1+e_2} = \frac{1}{4}(\bar{u}_i + \bar{u}_{i+e_1} + \bar{u}_{i+e_2} + \bar{u}_{i+e_1+e_2})$, with $e_1 = (1, 0)$ and $e_2 = (0, 1)$.
The stencil for bilinear interpolation is

$$[P^*] = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.3.3}$$

For the three-dimensional case, see [141].

Restrictions
We can be brief about restrictions. One may simply take

$$\boldsymbol{R} = \sigma \boldsymbol{P}^* \tag{5.3.4}$$

with σ a suitable scaling factor. The scaling of $\boldsymbol{R}$, i.e. the value of $\sum_j \boldsymbol{R}(i, j)$, is important. If $\boldsymbol{Ru}$ is to be a coarse grid approximation of $\boldsymbol{u}$ (this situation occurs in non-linear multigrid methods, which will be discussed later, then one should obviously have $\sum_j \boldsymbol{R}(i, j) = 1$. If however, $\boldsymbol{R}$ is used to transfer the residual $\boldsymbol{r}$ to the coarse grid, then the correct value of $\sum_j \boldsymbol{R}(i, j)$ depends on the scaling of the coarse and fine grid problems. The rule is that the coarse grid problem should be consistent with the differential problem in the same way as the fine grid problem. This means the following. Let the differential equation to be solved be denoted as

$$Lu = s \tag{5.3.5}$$

and the discrete approximation on the fine grid by

$$\boldsymbol{Au} = \boldsymbol{b} \tag{5.3.6}$$

Suppose that (5.3.6) is scaled such that it is consistent with $h^\alpha Lu = h^\alpha s$ with h a measure of the mesh-size of G. Finite volume discretization leads naturally to $\alpha = d$ with d the number of dimensions; often (5.3.6) is scaled in order to get rid of divisions by h. Let the discrete approximation of (5.3.5) on the coarse grid $\bar{G}$ be denoted by

$$\bar{\boldsymbol{A}}\bar{\boldsymbol{u}} = \boldsymbol{R}\boldsymbol{b} \tag{5.3.7}$$

and let $\bar{\boldsymbol{A}}$ approximate $\bar{h}^\alpha L$. Then $\boldsymbol{R}\boldsymbol{b}$ should approximate $\bar{h}^\alpha s$. Since $\boldsymbol{b}$ approximates $h^\alpha s$, we find a scaling rule, as follows.

Rule scaling of $\boldsymbol{R}$:

$$\sum_j R(i,j) = (\bar{h}/h)^\alpha \tag{5.3.8}$$

We emphasize that this rule applies only if $\boldsymbol{R}$ is to be applied to right-hand sides and/or residuals. Depending on the way the boundary conditions are implemented, at the boundaries α may be different from the interior. Hence the scaling of $\boldsymbol{R}$ should be different at the boundary. Another reason why $\sum_j \boldsymbol{R}(i,j)$ may come out different at the boundary is that use is made of the fact that due to the boundary conditions the residual to be restricted is known to be zero in certain points.
A restriction that cannot be obtained by (5.3.4) with interpolating prolongation is *injection*:

$$(\boldsymbol{R}\boldsymbol{u})_i = \sigma u_{2i} \tag{5.3.9}$$

Accuracy condition for transfer operators
The proofs of mesh-size independent rate of convergence of MG assume that $\boldsymbol{P}$ and $\boldsymbol{R}$ satisfy certain conditions [21], [57]. The last reference (p. 149) gives the following simple condition:

$$m_P + m_R > 2m \tag{5.3.10}$$

A necessary condition (not discussed here) is given in [66]. Here *orders* m_P, m_R of $\boldsymbol{P}$ and $\boldsymbol{R}$ are defined as the highest degree plus one of the polynomials that are interpolated exactly by $\boldsymbol{P}$ or $\boldsymbol{s}\boldsymbol{R}^*$, respectively, with s a scaling factor that can be chosen freely, and $2m$ is the order of the partial differential equation to be solved. For example, (5.3.9) has $m_R = 0$, (5.3.3) has $m_P = 2$. Practical experience (see e.g. [139]) confirms that (5.3.10) is necessary.

Operator-dependent transfer operators

If the coefficients in the differential equations are discontinuous across certain interfaces between subdomains of different physical properties, then $u \notin C^1(\Omega)$, and linear interpolation across discontinuities in $u_{,\alpha}$ is inaccurate. (See [141] for more details). Instead of interpolation, *operator-dependent prolongation* has to be used. Such prolongations aim to approximate the correct jump condition by using information from the discrete operator. They are required only in vertex-centered multigrid, but not in cell-centered multigrid, as shown in [141], where a full discussion of operator-dependent transfer operators may be found.

5.4 Coarse grid Galerkin approximation

The problem to be solved on the fine grid is denoted by

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{5.4.1}$$

The two-grid algorithm (2.3.14) requires an approximation $\bar{A}$ of $\boldsymbol{A}$ on the coarse grid. There are basically two ways to chose $\bar{\boldsymbol{A}}$, as already discussed in Chapter 2.

(i) *Discretization coarse grid approximation(DCA)*: like $\boldsymbol{A}$, $\bar{\boldsymbol{A}}$ is obtained by discretization of the partial differential equation.

(ii) *Galerkin coarse grid approximation (GCA)*:

$$\bar{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}\boldsymbol{P} \tag{5.4.2}$$

A discussion of (5.4.2) has been given in Chapter 2.

The construction of $\bar{\boldsymbol{A}}$ with DCA does not need to be discussed further. We will use stencil notation to obtain simple formulae to compute $\bar{\boldsymbol{A}}$ with GCA. The two methods will be compared, and some theoretical back-ground will be given.

Explicit formula for coarse grid operator

The matrices $\boldsymbol{R}$ and $\boldsymbol{P}$ are very sparse and have a rather irregular sparsity pattern. Stencil notation provides a very simple and convenient storage scheme. Storage rather than repeated evaluation is to be recommended if $\boldsymbol{R}$ and $\boldsymbol{P}$ are operator-dependent. We will derive formulae for $\bar{\boldsymbol{A}}$ using stencil notation. We have (cf. (5.2.16))

$$(\boldsymbol{P}\bar{\boldsymbol{u}}) = \sum_j \boldsymbol{P}^*(j, i-2j)\bar{u}_j \tag{5.4.3}$$

Unless indicated otherwise, summation takes place over $\mathbb{Z}^d$. Equation (5.2.1) gives

$$(\boldsymbol{A}\boldsymbol{P}\bar{\boldsymbol{u}})_i = \sum_k \boldsymbol{A}(i,k)(\boldsymbol{P}\bar{\boldsymbol{u}})_{i+k} = \sum_k \sum_j \boldsymbol{A}(i,k)\boldsymbol{P}^*(j, i+k-2j)\bar{u}_j \tag{5.4.4}$$

Finally, equation (5.2.5) gives

$$\begin{aligned}(\boldsymbol{RAP}\bar{\boldsymbol{u}})_i &= \sum_m \boldsymbol{R}(i,m)(\boldsymbol{AP}\bar{\boldsymbol{u}})_{2i+m} \\ &= \sum_m \sum_k \sum_j \boldsymbol{R}(i,m)\boldsymbol{A}(2i+m,k)\boldsymbol{P}^*(j,2i+m+k-2j)\bar{u}_j\end{aligned} \tag{5.4.5}$$

With the change of variables $j = i + n$ this becomes

$$(\bar{\boldsymbol{A}}\bar{\boldsymbol{u}})_i = \sum_m \sum_k \sum_n \boldsymbol{R}(i,m)\boldsymbol{A}(2i+m,k)\boldsymbol{P}^*(i+n,m+k-2n)\bar{u}_{i+n} \tag{5.4.6}$$

from which it follows that

$$\bar{\boldsymbol{A}}(i,n) = \sum_m \sum_k \boldsymbol{R}(i,m)\boldsymbol{A}(2i+m,k)\boldsymbol{P}^*(i+n,m+k-2n) \tag{5.4.7}$$

For calculation of $\bar{\boldsymbol{A}}$ by computer the ranges of m and k have to be finite. $S_{\boldsymbol{A}}$ is the *structure* of $\boldsymbol{A}$ as defined in (5.2.2), and $S_{\boldsymbol{R}}$ is the structure $\boldsymbol{R}$, i.e.

$$S_{\boldsymbol{R}} = \{j \in \mathbb{Z}^d : \exists i \in \bar{G} \ \text{ with } \quad \boldsymbol{R}(i,j) \neq 0\} \tag{5.4.8}$$

Equation (5.4.7) is equivalent to

$$\bar{\boldsymbol{A}}(i,n) = \sum_{m\in S_{\boldsymbol{R}}} \sum_{k\in S_{\boldsymbol{A}}} \boldsymbol{R}(i,m)\boldsymbol{A}(2i+m,k)\boldsymbol{P}^*(i+n,m+k-2n) \tag{5.4.9}$$

With this formula, computation of $\bar{\boldsymbol{A}}$ is straightforward, as we will now show.

Calculation of coarse grid operator by computer

For efficient computation of $\bar{\boldsymbol{A}}$ it is useful to first determine $\boldsymbol{S}_{\bar{\boldsymbol{A}}}$. This can be done with the following algorithm

Algorithm STRURAP

comment Calculation of $S_{\bar{\boldsymbol{A}}}$
begin $S_{\bar{\boldsymbol{A}}} = \emptyset$
    for $q \in S_{\boldsymbol{P}^*}$ ***do***
        for $m \in S_{\boldsymbol{R}}$ ***do***
            for $k \in S_{\boldsymbol{A}}$ ***do***
                begin $n = (m+k-q)/2$
                    if $(n \in \mathbb{Z}^d)$ ***then*** $S_{\bar{\boldsymbol{A}}} = S_{\bar{\boldsymbol{A}}} \cup n$
                end
    od od od
end STRURAP

Having determined $S_{\bar{\boldsymbol{A}}}$ it is a simple matter to compute $\bar{\boldsymbol{A}}$. This can be done with the following algorithm.

Algorithm CALRAP

```
comment Calculation of Ā
begin Ā = 0
    for n ∈ S_Ā do
        for m ∈ S_R do
            for k ∈ S_A do
                q = m + k − 2n
    if q ∈ S_P* then
                    Ḡ1 = {i ∈ Ḡ : 2i + m ∈ G} ∩ {i ∈ Ḡ : i + n ∈ Ḡ}
                for i ∈ Ḡ1 do
                    Ā(i, n) = Ā(i, n) + R(i, m)A(2i + m, k)P*(i + n, q)
    od od od
end CALRAP
```

Keeping computation on vector and parallel machines in mind, the algorithm has been designed such that the innermost loop is the longest.

To illustrate how $\bar{G}_1$ is obtained we given an example in two dimensions. Let G and $\bar{G}$ be given by

$$\begin{aligned} G &= \{i \in \mathbb{Z}^2 : 0 \leq i_1 \leq 2n_1,\ 0 \leq i_2 \leq 2n_2\} \\ \bar{G} &= \{i \in \mathbb{Z}^2 : 0 \leq i_1 \leq n_1,\ 0 \leq i_2 \leq n_2\} \end{aligned}$$

Then $i \in \bar{G}_1$ is equivalent so

$$\max(-j_\alpha, -m_\alpha/2, 0) \leq i_\alpha \leq \min(n_\alpha - m_\alpha/2, n_\alpha - j_\alpha, n_\alpha) \quad \alpha = 1, 2$$

It is easy to see that the inner loop vectorizes along grid lines.

Comparision of discretization and Galerkin coarse grid approximation

Although DCA seems more straightford, GCA has some advantages. The coarsest grids employed in multigrid methods may be very coarse. On such very coarse grids DCA may be unreliable if the coefficients are variable, because these coefficients are sampled in very few points. An example where multigrid fails because of this effect is given in [137]. The situation can be remedied by not sampling the coefficients pointwise on the coarse grids, but taking suitable averages. This is, however, precisely that GCA does accurately and automaticaly.

For the same reason GCA is to be used for interface problems (discontinuous coefficients), in which case the danger of pointwise sampling of coefficients is most obvious. Another advantage of GCA is that it is purely algebraic in nature; no use in made of the underlying differential equation. This opens the possibility of developing autonomous or 'black box' multigrid subroutines, requiring as input only a matrix and right-hand side. On the other hand, for non-linear problems and for systems of differential equations there is no general way to implement GCA. Both DCA and GCA are in widespread use.

Structure of coarse grid operator stencil
Galerkin coarse grid approximation will be useful only if $S_{\bar{\boldsymbol{A}}}$ is not (much) larger than $\boldsymbol{S}_A$, otherwise the important property of MG, that computing work is proportional to the number of unknowns, may get lost. For examples and further discussion of CGA, including the possible loss of the K-matrix property on coarse grids, see [141].

6 Multigrid algorithms

6.1 Introduction

The order in which the grids are visited is called the *multigrid schedule*. Several schedules will be discussed. All multigrid algorithms are variants of what may be called the *basic multigrid algorithm*. This basic algorithm is nonlinear, and contains linear multigrid as a special case. The most elegant description of the basic multigrid algorithm is by means of a recursive formulation. FORTRAN does not allow recursion, thus we also present a non-recursive formulation. This can be done in many ways, and various flow diagrams have been presented in the literature. If, however, one constructs a structure diagram not many possibilities remain, and a well structured non-recursive algorithm containing only one goto statement results. The decision whether to go a finer or to a coarser grid is taken in one place only.

6.2 The basic two-grid algorithm

Preliminaries
Let a sequence $\{G^k : k = 1, 2, ..., K\}$ of increasingly finer grids be given. Let U^k be the set of grid functions $G^k \to \mathbb{R}$ on G^k; a grid function $\boldsymbol{U}^k \in U^k$ stands for m functions in the case where we want to solve a set of equations for m unknowns. Let there be given transfer operators $\boldsymbol{P}^k : U^{k-1} \to U^k$ (prolongation) and $\boldsymbol{R}^k : U^k \to U^{k-1}$ (restriction). Let the problem to be solved on G^k be denoted by

$$\boldsymbol{L}^k(\boldsymbol{u}^k) = \boldsymbol{b}^k \tag{6.2.1}$$

The operator $\boldsymbol{L}^k$ may be linear or non-linear. Let on every grid a smoothing algorithm be defined, denoted by $S(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{f}, \nu, k)$. S changes an initial guess $\boldsymbol{u}^k$ into an improved approximation $\boldsymbol{v}^k$ with right-hand side $\boldsymbol{f}^k$ by ν_k iterations with a suitable smoothing method. The use of the same symbol $\boldsymbol{u}^k$ for the solution of (6.2.1) and for approximations of this solution will not cause confusion; the meaning of $\boldsymbol{u}^k$ will be clear from the context. On the coarse grid G^1 we sometimes wish to solve (6.2.1) exactly; in general we do not wish to be specific about this, and we write $S(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{f}, \bullet, 1)$ for smoothing or solving on G^1.

The nonlinear two-grid algorithm

Let us first assume that we have only two grids G^k and G^{k-1}. The following algorithm is a generalization of the linear two-grid algorithm discussed in Section 2.3. Let some approximation $\tilde{\boldsymbol{u}}^k$ of the solution on G^k be given. How $\tilde{\boldsymbol{u}}^k$ may be obtained will be discussed later. The non-linear two-grid algorithm is defined as follows. Let $\boldsymbol{f}^k = \boldsymbol{b}^k$.

Subroutine TG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$
comment nonlinear two-grid algorithm
begin
(1) $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \nu, k)$
(2) $\boldsymbol{r}^k = \boldsymbol{f}^k - \boldsymbol{L}^k(\boldsymbol{u}^k)$
(3) Choose $\tilde{\boldsymbol{u}}^{k-1}, s_{k-1}$
(4) $\boldsymbol{f}^{k-1} = \boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1}) + s_{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^k$
(5) $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k-1)$
(6) $\boldsymbol{u}^k = \boldsymbol{u}^k + (1/s_{k-1})\boldsymbol{P}^k(\boldsymbol{u}^{k-1} - \tilde{\boldsymbol{u}}^{k-1})$
(7) $S(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{f}, \mu, k)$
end of TG

A call of TG gives us one two-grid iteration. The following program performs ntg two-grid iterations:

Choose $\tilde{\boldsymbol{u}}^k$
$\boldsymbol{f}^k = \boldsymbol{b}^k$
for $i = 1$ ***step*** 1 ***until*** ntg ***do***
    TG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$
    $\tilde{\boldsymbol{u}} = \boldsymbol{u}$
od

Discussion

Subroutine TG is a straightforward implementation of the basic multigrid principles discussed in Chapter 2, but there are a few subtleties involved.

We proceed with a discussion of subroutine TG. Statement (1) represents ν_k smoothing iterations (pre-smoothing), starting from an initial guess $\tilde{\boldsymbol{u}}^k$. In (2) the residual $\boldsymbol{r}^k$ is computed; $\boldsymbol{r}^k$ is going to steer the coarse grid correction. Because 'short wavelength accuracy' already achieved in $\boldsymbol{u}^k$ must not get lost, $\boldsymbol{u}^k$ is to be kept, and a correction $\delta\boldsymbol{u}^k$ (containing 'long wavelength information') is to be added to $\boldsymbol{u}^k$. In the non-linear case, $\boldsymbol{r}^k$ cannot be taken for the right-hand side of the problem for $\delta\boldsymbol{u}^k$; $\boldsymbol{L}(\delta\boldsymbol{u}^k) = \boldsymbol{r}^k$ might not even have a solution. For the same reason, $\boldsymbol{R}^{k-1}\boldsymbol{r}^k$ cannot be the right-hand side for the coarse grid problem on G^{k-1}. Instead, it is added in (4) to $\boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})$, with $\tilde{\boldsymbol{u}}^{k-1}$ an approximation to the solution of (6.2.1) in some sense (e.g. $\boldsymbol{P}\tilde{\boldsymbol{u}}^{k-1} \simeq$ solution of equation (6.2.1). Obviously, $\boldsymbol{L}^{k-1}(\boldsymbol{u}^{k-1}) = \boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})$ has a solution, and if $\boldsymbol{R}^{k-1}\boldsymbol{r}^k$ is not too large, then $\boldsymbol{L}^{k-1}(\boldsymbol{u}^{k-1}) = \boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1}) + \boldsymbol{R}^{k-1}\boldsymbol{r}^k$ can also be solved, which is done in statement (5) (exactly or approximately).

$\boldsymbol{R}^{k-1}\boldsymbol{r}^k$ will be small when $\tilde{\boldsymbol{u}}^k$ is close to the solution of equation (6.2.1), i.e. when the algorithm is close to convergence. In order to cope with situations where $\boldsymbol{R}^{k-1}\boldsymbol{r}^k$ is not smal enough, the parameter s_{k-1} is introduced. By choosing s_{k-1} small enough one can bring $\boldsymbol{f}^{k-1}$ arbitrarily close to $\boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})$. Hence, solvability of $\boldsymbol{L}^{k-1}(\boldsymbol{u}^{k-1}) = \boldsymbol{f}^{k-1}$ can be ensured. Furthermore, in bifurcation problems, $\boldsymbol{u}^{k-1}$ can be kept on the same branch as $\tilde{\boldsymbol{u}}^{k-1}$ by means of s_{k-1}. In (6) the coarse grid correction is added to $\boldsymbol{u}^k$. Omission of the factor $1/s_{k-1}$ would mean that only part of the coarse grid correction is added to $\boldsymbol{u}^k$, which amounts to damping of the coarse grid correction; this would slow down convergence. Finally, statement (7) represents μ_k smoothing iterations (post-smoothing).

The linear two-grid algorithm

It is instructive to see what happens when $\boldsymbol{L}^k$ is linear. It is reasonable to assume that then $\boldsymbol{L}^{k-1}$ is also linear. Furthermore, let us assume that the smoothing method is linear, that is to say, statement (5) is equivalent to

$$\boldsymbol{u}^{k-1} = \tilde{\boldsymbol{u}}^{k-1} + \boldsymbol{B}^{k-1}(\boldsymbol{f}^{k-1} - \boldsymbol{L}^{k-1}\tilde{\boldsymbol{u}}^{k-1}) \tag{6.2.2}$$

with $\boldsymbol{B}^{k-1}$ some linear operator. With $\boldsymbol{f}^{k-1}$ from statement (4) this gives

$$\boldsymbol{u}^{k-1} = \tilde{\boldsymbol{u}}^{k-1} + s_{k-1}\boldsymbol{B}^{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^k \tag{6.2.3}$$

Statement (6) gives

$$\boldsymbol{u}^k = \boldsymbol{u}^k + \boldsymbol{P}^k\boldsymbol{B}^{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^k \tag{6.2.4}$$

and we see that the coarse grid correction $\boldsymbol{P}^k\boldsymbol{B}^{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^k$ is independent of the choice of s_{k-1} and $\tilde{\boldsymbol{u}}^{k-1}$ in the linear cas. Hence, we may as well choose $s_{k-1} = 1$ and $\tilde{\boldsymbol{u}}^{k-1} = 0$ in the linear case. This gives us the following linear two-grid algorithm.

Subroutine LTG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$
comment linear two-grid algorithm
begin

$S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \nu, k)$
$\boldsymbol{r}^k = \boldsymbol{f}^k - \boldsymbol{L}^k \boldsymbol{u}^k$
$\boldsymbol{f}^{k-1} = \boldsymbol{R}^{k-1} \boldsymbol{r}^k$
$\tilde{\boldsymbol{u}}^{k-1} = 0$
$S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k-1)$
$\boldsymbol{u}^k = \boldsymbol{u}^k + \boldsymbol{P}^k \boldsymbol{u}^{k-1}$
$S(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{f}, \mu, k)$

end of LTG

Choice of $\tilde{\boldsymbol{u}}^{k-1}$ and s_{k-1}

There are several possibilities for the choice of $\tilde{\boldsymbol{u}}^{k-1}$. One possibility is

$$\tilde{\boldsymbol{u}}^{k-1} = \tilde{\boldsymbol{R}}^{k-1} \boldsymbol{u}^k \tag{6.2.5}$$

where $\tilde{\boldsymbol{R}}^{k-1}$ is a restriction operator which may or may not be the same as $\boldsymbol{R}^{k-1}$.

With the choice $s_{k-1} = 1$ this gives us the first non-linear multigrid algorithm that has appeared, the FAS (full approximation storage) algorithm proposed by Brandt [20]. The more general algorithm embodied in subroutine TG, containing the parameter s_{k-1} and leaving the choice of $\tilde{\boldsymbol{u}}_{k-1}$ open, has been proposed by Hackbusch [54], [49], [57]. In principle it is possible to keep $\tilde{\boldsymbol{u}}_{k-1}$ fixed, provided it is sufficiently close to the solution of $\boldsymbol{L}^{k-1}(\boldsymbol{u}^{k-1}) = \boldsymbol{b}^{k-1}$. This decreases the cost per iteration, since $\boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})$ needs to be evaluated only once, but the rate of convergence may be slower than with $\tilde{\boldsymbol{u}}^{k-1}$ defined by (5). We will not discuss this variant. Another choice of $\tilde{\boldsymbol{u}}^{h-1}$ is provided by nested iteration, which will be discussed later. Hackbusch [54], [49], [57] gives the following guidelines for the choice of $\tilde{\boldsymbol{u}}^{k-1}$ and the parameter s_{k-1}. Let the non-linear equation $\boldsymbol{L}^{k-1}(\boldsymbol{u}^{k-1}) = \boldsymbol{f}^{k-1}$ be solvable for $\|\boldsymbol{f}^{k-1}\| < \rho_{k-1}$. Let $\|\boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})\| < \rho_{k-1}/2$. Choose s_{k-1} such that $\|s_{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^h\| < \rho_{k-1}/2$, for example:

$$s_{k-1} = \frac{1}{2}\rho_{k-1}/\|\boldsymbol{R}^{k-1}\boldsymbol{r}^k\| \,. \tag{6.2.6}$$

Then $\|\boldsymbol{f}^{k-1}\| < \rho_{k-1}$, so that the coarse grid problems has a solution.

6.3 The basic multigrid algorithm

The recursive non-linear multigrid algorithm

The basic multigrid algorithm follows from the two-grid algorithm by replacing the coarse grid solution statement (statement (5) in subroutine TG) by γ_k multigrid iterations. This leads to

Subroutine MG1 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k, \gamma)$
comment recursive non-linear multigrid algorithm
begin
  if $(k$ ***eq*** $1)$ ***then***
(1) $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k)$
  else
(2) $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \nu, k)$
(3) $\boldsymbol{r}^k = \boldsymbol{f}^k - \boldsymbol{L}^k(\boldsymbol{u}^k)$
(4)   Choose $\tilde{\boldsymbol{u}}^{k-1}, s_{k-1}$
(5) $\boldsymbol{f}^{k-1} = \boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1}) + s_{k-1}\boldsymbol{R}^{k-1}\boldsymbol{r}^k$
    for $i = 1$ ***step*** 1***until*** γ_k ***do***
(6)   MG1 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k-1, \gamma)$
    od
(7) $\boldsymbol{u}^k = \boldsymbol{u}^k + (1/s_{k-1})\boldsymbol{P}^k(\boldsymbol{u}^{k-1} - \tilde{\boldsymbol{u}}^{k-1})$
(8) $S(\boldsymbol{u}, \boldsymbol{u}, f, \mu, k)$
  endif
end of MG1

After our discussion of the two-grid algorithm, this algorithm is self explanatory.
The following program carries out nmg multigrid iterations, starting on the finest grid G^K:

Program 1:
Choose $\tilde{\boldsymbol{u}}^K$
$\boldsymbol{f}^K = \boldsymbol{b}^K$
for $i = 1$ ***step*** 1 ***until*** nmg ***do***
  MG1 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, K, \gamma)$
od

The recursive linear multigrid algorithm
The linear multigrid algorithm follows easily from the linear two-grid algorithm LTG:

Subroutine LMG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$
comment recursive linear multigrid algorithm
begin
  if $(k = 1)$ ***then***
    $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k)$
  else
    $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \nu, k)$
    $\boldsymbol{r}^k = \boldsymbol{f}^k - \boldsymbol{L}^k \boldsymbol{u}^k$
    $\boldsymbol{f}^{k-1} = \boldsymbol{R}^{k-1} \boldsymbol{r}^k$
    $\tilde{\boldsymbol{u}}^{k-1} = 0$
    for $i = 1$ ***step*** 1 ***until*** γ_k ***do***
      LMG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k-1)$
      $\tilde{\boldsymbol{u}}^{k-1} = \boldsymbol{u}^{k-1}$
    od
    $\boldsymbol{u}^k = \boldsymbol{u}^k + \boldsymbol{P}^k \boldsymbol{u}^{k-1}$
    $S(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{f}, \mu, k)$
  endif
end LMG

Here $\tilde{\boldsymbol{u}}$ plays the role of an initial guess.

Multigrid schedules
The order in which the grids are visited is called the *multigrid schedule* or *multigrid cycle*. If the parameters $\gamma_k,\ k = 1, 2, ..., K-1$ are fixed in advance we have a *fixed schedule*; if γ_k depends on intermediate computational results we have an *adaptive schedule*. Figure 6.3.1 shows the order in which the grids are visited with $\gamma_k = 1$ and $\gamma_k = 2,\ k = 1, 2, ..., K-1$, in the case $K = 4$. A dot represents a smoothing operation. Because of the shape of these diagrams, these schedules are called the V-, W- and sawtooth cycles, respectively. The sawtooth cycle is a special case of the V-cycle, in which smoothing before coarse grid correction (pre-smoothing) is deleted. A schedule intermediate between these two cycles is the F-cycle. In this cycle coarse grid correction takes place by means of one F-cycle followed by one V-cycle. Figure 6.3.2 gives a diagram for the F-cycle, with $K = 5$.

Recursive algorithm for V-, F- and W-cycle
A version of subroutine MG1 for the V-, W- and F-cycles is as follows. The parameter γ is now an integer instead of an integer array.

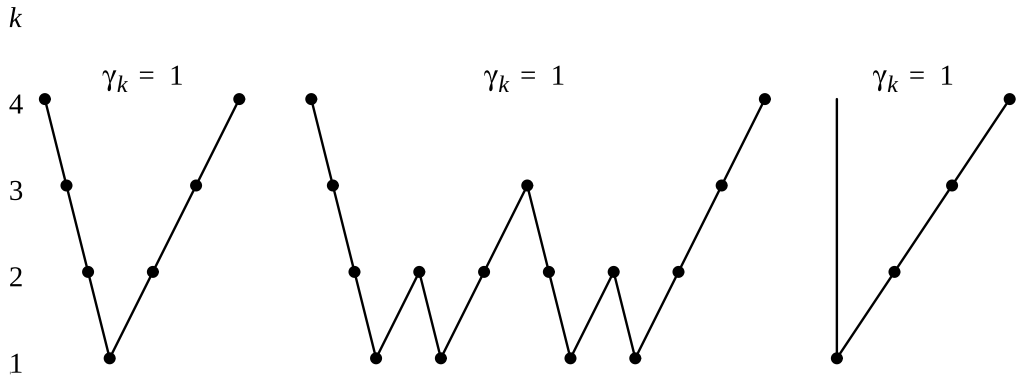


Figure 6.3.1: V-, W- and sawtooth-cycle diagrams.

Subroutine MG2 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k, \gamma)$
comment nonlinear multigrid algorithm V-, W-, or F-cycle
begin
    if $(k$ ***eq*** $1)$ ***then***
        $S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k)$
        if (cycle ***eq*** F) ***then*** $\gamma = 1$ ***endif***
    else
        A
        for $i = 1$ ***step*** 1 ***until*** γ ***do***
            MG2 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k-1, \gamma)$
        od
        B
        if $(k$ ***eq*** K ***and*** cycle ***eq*** $F)$ ***then*** $\gamma = 2$ ***endif***
    endif
end MG2

Here A and B represent statements (2) to (5) and (7) and (8) in subroutine MG1. The following program carries out nmg V-, W-, or F-cycles.

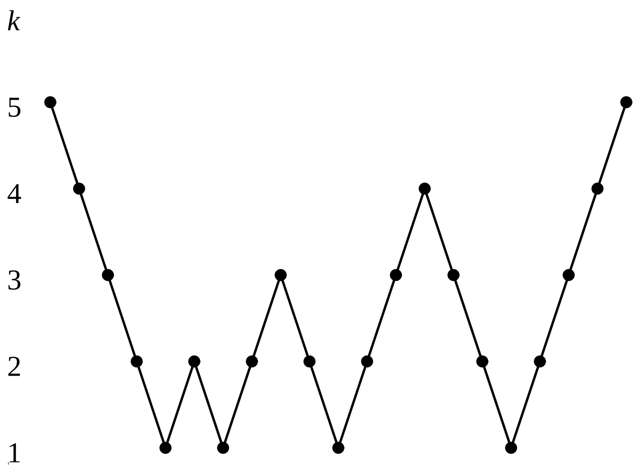


Figure 6.3.2: F-cycle diagram.

Program 2:
Choose $\tilde{\boldsymbol{u}}^K$
$\boldsymbol{f}^K = \boldsymbol{b}^K$
if (cycle ***eq*** W ***or*** cycle ***eq*** F) ***then*** $\gamma = 2$ ***else*** $\gamma = 1$
for $i = 1$ ***step*** 1 ***until*** nmg ***do***
    MG2 $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, K, \gamma)$
od

Adaptive schedule
An example of an adaptive strategy is the following. Suppose we do not carry out a fixed number of multigrid iterations on level G^k, but wish to continue to carry out multigrid interactions, until the problem on G^k is solved to within a specified accuracy. Let the accuracy requirement be

$$||\boldsymbol{L}^k(\boldsymbol{u}^k) - \boldsymbol{f}^k|| \leq \varepsilon_k = \delta_k||\boldsymbol{L}^{k+1}(\boldsymbol{u}^{k+1}) - \boldsymbol{f}^{k+1}|| \tag{6.3.1}$$

with $\delta \in (0,1)$ a parameter.

At first sight, a more natural definition of ε^k would seem to be $\varepsilon^k = \delta||\boldsymbol{f}^k||$. Since $\boldsymbol{f}^k$ does not, however, go to zero on convergence, this would lead to skipping of coarse grid correction when $\boldsymbol{u}^{k+1}$ approaches convergence. Analysis of the linear case leads naturally to condition (6.3.1). An adaptive multigrid schedule with criterion (6.3.1) is implemented in the following algorithm. In order to make the algorithm finite, the maximum number of multigrid iterations allowed is γ.

$$
\begin{array}{|l}
\textbf{\textit{Subroutine}}\ \ \mathrm{MG3}\ \ (\bar{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k) \\
\textbf{\textit{comment}}\ \ \text{recursive nonlinear multigrid algorithm with adaptive} \\
\text{schedule} \\
\textbf{\textit{begin}} \\
\qquad \textbf{\textit{if}}\ (k\ \textbf{\textit{eq}}\ 1)\ \textbf{\textit{then}} \\
\qquad\qquad\qquad S(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, \bullet, k) \\
\qquad \textbf{\textit{else}} \\
\qquad\qquad\qquad A \\
(1)\quad t_{k-1} = \|\boldsymbol{r}^k\| - \varepsilon_k \\
\qquad\qquad\qquad \varepsilon_{k-1} = \delta s_{k-1} \|\boldsymbol{r}^k\| \\
\qquad\qquad\qquad n_{k-1} = \gamma \\
\qquad \textbf{\textit{while}}\ (t_{k-1} > 0\ \textbf{\textit{and}}\ n_{k-1} \geq 0) \\
\qquad\qquad\qquad \mathrm{MG3}\ \ (\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k-1) \\
\qquad\qquad\qquad n_{k-1} = n_{k-1} - 1 \\
\qquad\qquad\qquad t_{k-1} = \|\boldsymbol{L}^{k-1}(\tilde{\boldsymbol{u}}^{k-1})_{-} f^{k-1}\| - \varepsilon_{k-1} \\
\qquad \textbf{\textit{od}} \\
\qquad B \\
\textbf{\textit{endif}} \\
\textbf{\textit{end}}\quad \mathrm{MG3}
\end{array}
$$

Here A and B stand for the same groups of statements as in subroutine MG2. The purpose of statement (1) is to allow the possibility that the required accuracy is already reached by pre-smoothing on G^k, so that coarse grid correction can be skipped. The following program solves the problem on G^K within a specified tolerance, using the adaptive subroutine MG3:

$$
\begin{array}{|l}
\mathit{Program}\ 3: \\
\mathit{Choose}\ \tilde{\boldsymbol{u}}^K \\
\boldsymbol{f}^K = \boldsymbol{b}^K;\ \varepsilon_K = \mathrm{tol}\ *\ \|\boldsymbol{b}^K\|\ ;\ t_K = \|L^K(\tilde{\boldsymbol{u}}^K) - \boldsymbol{b}^K\| - \varepsilon_K \\
n = \mathrm{nmg} \\
\textbf{\textit{while}}\ (t_K > 0\ \textbf{\textit{and}}\ n \geq 0)\ \textbf{\textit{do}} \\
\qquad \mathrm{MG3}\ \ (\tilde{\boldsymbol{u}}, \boldsymbol{u}, f, K) \\
\qquad n = n - 1 \\
\qquad t_K = \|L^K(\tilde{\boldsymbol{u}}^K) - \boldsymbol{b}^K\| - \varepsilon_K \\
\textbf{\textit{od}}
\end{array}
$$

The number of iterations is limited by mng.

Storage requirements

Let the finest grid G^K be either of the vertex-centered type given by (5.1.1) or of the cell-centered type given by (5.1.2). Let in both cases $n_\alpha = n_\alpha^{(K)} = m_\alpha \cdot 2^K$. Let the coarse grids G^k, $k = K-1, K-2, ..., 1$ be constructed by successive doubling of the mesh-sizes h_α (*standard coarsening*). Hence, the number of grid-points N_k of G^k is

$$N_k = \prod_{\alpha=1}^{d}(1 + m_\alpha \cdot 2^k) \simeq M2^{kd} \tag{6.3.2}$$

in the vertex-centered case, with

$$M = \prod_{\alpha=1}^{d} m_\alpha,$$

and

$$N_k = M2^{kd} \tag{6.3.3}$$

in the cell-centered case. In order to be able to solve efficiency on the coarsest grid G^1 it is desirable that m_α is small. Henceforth, we will not distinguish between the vertex-centered and cell-centered case, and assume that N_k is given by (6.3.3).

It is to be expected that the amount of storage reqired for the computations that take place on G^k is given by $c_1 N_k$, with c_1 some constant independent of k. Then the total amount of storage required is given by

$$c_1 \sum_{k=1}^{K} N_k \simeq \frac{2^d}{2^d - 1} c_1 N_K \tag{6.3.4}$$

Hence, as compared to single grid solutions on method selected, the use of multigrid increases the storage required by a factor of $2^d/(2^d-1)$, which is $4/3$ in two and $8/7$ in three dimensions, so that the additional storage requirement posed by multigrid seems modest.
Next, suppose that *semi-coarsening* (cf. Section 7.3) is used for the construction of the coarse grids $G^k, k < K$. Assume that in one coordinate direction the mesh-size is the same on all grids. Then

$$N_k = M2^{K+k(d-1)} \tag{6.3.5}$$

and the total amount of storage required is given by

$$c_1 \sum_{k=1}^{K} N_k \simeq \frac{2^{d-1}}{2^{d-1} - 1} c_1 N_K \tag{6.3.6}$$

Now the total amount of storage required by multigrid compared with single grid solution on G^K increases by a factor 2 in two and $4/3$ in three dimensions. Hence, in two dimensions the storage cost associated with semi-coarsening multigrid is not negligible.

Computational work

We will estimate the computational work of one iteration with the fixed schedule algorithm MG2. A close approximation of the computational work w_k to be performed on G_k will be $w_k = c_2 N_k$, assuming the number of pre- and post-smoothings ν_k and μ_k are independent of k, and that the operators $\boldsymbol{L}^k$ are of similar complexity (for example, in the linear case, $\boldsymbol{L}^k$ are matrices of equal sparsity). More precisely, let us define w_k to be all computing work involved in MG2 $(\bar{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$, except the recursive call of MG2. Let W_k be all work involved in MG2 $(\bar{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$. Let $\gamma_k = \gamma,\ k = 2, 3, ..., K-1$, in subroutine MG2 (e.g., the V- or W-cycles). Assume *standard coarsering.* Then

$$W_k = c_2 M 2^{kd} + \gamma W_{k-1} \tag{6.3.7}$$

In [141] it is shown that if

$$\tilde{\gamma} \equiv \gamma / 2^d < 1 \tag{6.3.8}$$

then

$$\tilde{W}_K < \tilde{W} = 1/(1-\tilde{\gamma}) \tag{6.3.9}$$

with $\tilde{W}_K \equiv W_K/(c_2 N_K)$.

The following conclusions may be drawn from (6.3.10). $\tilde{W}_K$ is the ration of multigrid work and work on the finest grid. The bulk of the work on the finest grid usually consists of smoothing. Hence, $\tilde{W}_K - 1$ is a measure of the additional work required to accelerate smoothing on the finest grid G^K by means of multigrid.

If $\tilde{\gamma} \geq 1$ the work W_K is superlinear in the number of unknowns N_K, see [141].

If $\tilde{\gamma} < 1$ equation (6.3.9) gives

$$W_K < c_2 N_K/(1-\tilde{\gamma}) \tag{6.3.10}$$

so that W_K is linear in N_K. It is furthermore significant that the constant of proportionality $c_2/(1-\tilde{\gamma})$ is small. This because c_2 is just a little greater than the work per grid point of the smoothing method, which is supposed to be a simple iterative method (if not, multigrid

is not applied in an appropriate way). Since an (perhaps the main) attractive feaure of multigrid is the possibility to realize linear computational complexity with small constant of proportionality, one chooses $\tilde{\gamma} < 1$, or $\gamma < 2^d$. In practice it is usually found that $\gamma > 2$ does not result in significantly faster convergence. The rapid growth of W_K with γ means that it is advantageous to choose $\gamma \leq 2$, which is why the V- and W-cycles are widely used.

The computational cost of the F-cycle may be estimated as follows. In Figure 6.3.3 the diagram of the F-cycle has been redrawn, distinguishing between the work that is done on

G^k preceding coarse grid correction (pre-work, statements A in subroutine MG2) and after coarse grid correction (post-work, statements B in subroutine MG2). The amount of pre- and post-work together is $c_2 M 2^{kd}$, as before. It follows from the diagram, that on G^k the cost of pre- and post-work is incurred j_k times, with $j_k = K - k + 1,\ k = 2, 3, ..., K$, and $j_1 = K - 1$. For convenience we redefine $j_1 = K$, bearing our earlier remarks on the inaccuracy and unimportance of the estimate of the work in G^1 in mind. One obtains

$$W_K = c_2 M \sum_{k=1}^{K} (K - k + 1) 2^{kd} \tag{6.3.11}$$

We have

$$\sum_{k=1}^{K} k 2^{kd} = \frac{2^{(K+1)d}}{(2^d - 1)^2}[K(2^d - 1) - 1] + \frac{2^d}{(2^d - 1)^2} \tag{6.3.12}$$

as is checked easily. It follows that

$$W_K = c_2 M (2^{d(K+2)} + K + 1 - K 2^d)/(2^d - 1)^2$$

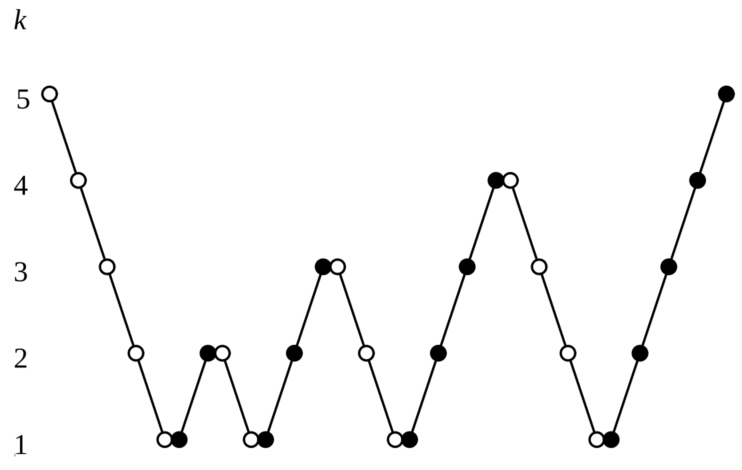


Figure 6.3.3: F-cycle ($\circ$ pre-work, $\bullet$ post-work).

so that

$$\tilde{W}_K < \tilde{W} = 1/(1 - 2^{-d})^2 \tag{6.3.13}$$

Table 6.3.1 gives $\tilde{W}$ as given by (6.3.9) and (6.3.13) for a number of cases. The ratio of multigrid over single grid work is seen to be not large, especially in three dimensions. The

d	2	3
V-cycle	4/3	8/7
F-cycle	16/9	64/49
W-cycle	2	4/3
$\gamma = 3$	4	8/5

Table 6.3.1: Values of $\tilde{W}$, standard coarsening

F-cycle is not much cheaper than the W-cycle. In three dimensions the cost of the V-, F- and W-cycles is almost the same.
Suppose next that *semi-coarsening* is used. Assume that in one coordinate direction the mesh-size is the same on all grids. The number of grid-points N_k of G^k is given by (6.3.5). With $\gamma_k = \gamma,\ k = 2, 3, ..., K-1$ we obtain

$$W_k = c_2 M 2^{K+k(d-1)} + \gamma W_{k-1} \tag{6.3.14}$$

Hence W_k is given by (6.3.8) and $\tilde{W}$ by (6.3.9) with $\tilde{\gamma} = \gamma/2^{d-1}$. For the F-cycle we obtain

$$W_K = c_2 M 2^K \sum_{k=1}^{K} (K - k + 1) 2^{k(d-1)} \tag{6.3.15}$$

Hence

$$W_K < \tilde{W} = 1/(1 - 2^{1-d})^2$$

d	2	3
V-cycle	2	4/3
F-cycle	4	16/9
W-cycle	–	2
γ=3	–	4

Table 6.3.2: Values of $\tilde{W}$, semi-coarsening

Table 6.3.2 gives $\tilde{W}$ for a number of cases. In two dimensions $\gamma = 2$ or 3 is not useful, because $\tilde{\gamma} \geq 1$. It may happen that the rate of convergence of the V-cycle is not independent of the mesh-size, for example if a singular perturbation problem is being solved (e.g. convection-diffusion problem with $\varepsilon \ll 1$), or when the solution contains singularities. With the W-cycle we have $\tilde{\gamma} = 1$ with semi-coarsening, hence $\tilde{W}_k = K$. In practice, K is usually not greater than 6 or 7, so that the W-cycle is still affordable. The F-cycle may be more efficient.

Work units

The ideal computing method to approximate the behaviour of a given physical problem involves an amount of computing work that is proportional to the number and size of the physical changes that are modeled. This has been put forward as the 'golden rule of computation' by Brandt [16]. As has been emphasized by Brand in a number of publications, e.g. [20], [21], [22], [16], this involves not only the choice of methods to solve (6.2.1), but also the choice of the mathematical model and its discretization. The discretization and solution processes should be interwined, leading to *adaptive disretization*. We shall not discuss adaptive methods here, but regard (6.2.1) as given. A practical measure of the minimum computing work to solve (6.2.1) is as follows. Let us define one *work unit* (WU) as the amount of computing work required to evaluate the residual $\boldsymbol{L}^K(\boldsymbol{u}^K) - \boldsymbol{b}^K$ of Equation 6.2.1) on the finest grid G^K. Then it is to be expected that (6.2.1) cannot be solved at a cost less than few WU, and one should be content if this is realized. Many publications show that this goal can indeed be achieved with multigrid for significant physical problems, for example in computational fluid dynamics. In practice the work involved in smoothing is by far the dominant part of the total work. One may, therefore, also define one work unit, following [20], as the work involved in one smoothing iteration on the finest grid G^K. This agrees more or less with the first definition only if the smoothing algorithm is simple and cheap. As was already mentioned, if this is not the case multigrid is not applied in an appropriate way. One smoothing iteration on G^k then adds $2^{d(k-K)}$ WU to the total work. It is a good habit, followed by many authors, to publish convergence histories in terms of work units. This facilitaties comparisons between methods, and helps in developing and improving multigrid codes.

6.4 Nested iteration

The algorithm

Nested iteration, also called *full multigrid* (FMG, [22], [16]) is based on the following idea. When no *a priori* information about the solution is available to assist in the choice of the initial guess $\boldsymbol{u}^K$ on the finest grid G^K, it is obviously wasteful to start the computation on the finest grid, as is done by subroutines MGi, $i = 1, 2, 3$ of the preceding section. With an unfortunate choice of the initial $\boldsymbol{u}^K$, the algorithm might even diverge for a nonlinear problem. Computing on the coarse grids is so much cheaper, thus it is better to use the coarse grids to provide an informed guess for $\boldsymbol{u}^K$. At the same time, this gives us a choice for $\tilde{\boldsymbol{u}}^k$, $k < K$. Nested iteration is defined by the following algorithm.

$$
\begin{array}{r|l}
 & \textit{Program } 1 \\
 & \textbf{\textit{comment}} \text{ nested iteration algorithm} \\
 & \text{Choose } \tilde{\boldsymbol{u}}^1 \\
(1) & S(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{u}}, \boldsymbol{f}, \cdot, 1) \\
 & \textbf{\textit{for}}\ k = 2\ \textbf{\textit{step}}\ 1\ \textbf{\textit{until}}\ K\ \textbf{\textit{do}} \\
(2) & \qquad \boldsymbol{u}^k = \tilde{\boldsymbol{u}}^k = \tilde{\boldsymbol{P}}^k \tilde{\boldsymbol{u}}^{k-1} \\
 & \qquad \textbf{\textit{for}}\ i = 1\ \textbf{\textit{step}}\ 1\ \textbf{\textit{until}}\ \tilde{\gamma}_k\ \textbf{\textit{do}} \\
(3) & \qquad\qquad \text{MG } (\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k) \\
 & \qquad \textbf{\textit{od}} \\
 & \textbf{\textit{od}}
\end{array}
$$

Of course, the value of γ_k inside MG may be different from $\tilde{\gamma}_k$.

Choice of prolongation operator

The prolongation operator $\tilde{\boldsymbol{P}}^k$ does not need to be identical to $\boldsymbol{P}^k$. In fact, there may be good reason to choose it differently. As discussed in [57] (for a simplified analysis see [141]), it is often advisable to choose $\tilde{\boldsymbol{P}}^k$ such that

$$m_{\tilde{P}} > m_c \tag{6.4.1}$$

where $m_{\tilde{P}}$ is the order of the prolongation operator as defined in Section 5.3, and m_c is the order of consistency of the discretizations $\boldsymbol{L}^k$, here assumed to be the same on all grids. Often $m_c = 2$ (second-order schemes). Then (6.4.1) implies that $\tilde{\boldsymbol{P}}^k$ is exact for second-order polynomials.

Note that nested iteration provides $\tilde{\boldsymbol{u}}^k$; this is an alternative to (6.2.5).

As discussed in [57] and [141], if MG converges well then the nested iteration algorithm results in a $\boldsymbol{u}^K$ which differs from the solution of (6.2.1) by an amount of the order of the truncation error. If one desires, the accuracy of $\boldsymbol{u}^K$ may be improved further by following the nested iteration algorithm with a few more multigrid iterations.

Computational cost of nested iteration

Let $\tilde{\gamma}_k = \hat{\gamma},\ k = 2, 3, ..., K$, in the nested iteration algorithm, let W_k be the work involved in MG $(\tilde{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{f}, k)$, and assume for simplicity that the (negligible) work on G^1 equals W_1. Then the computational work W_{ni} of the nested iteration algorithm, neglecting the cost of $\tilde{\boldsymbol{P}}^k$, is given by

$$W_{ni} = \sum_{k=1}^{K} \hat{\gamma} W_k \tag{6.4.2}$$

Assume inside Mg $\gamma_k = \gamma$, $k = 2, 3, ..., K$ and let $\tilde{\gamma} = \gamma/2^d < 1$. Note that γ and $\hat{\gamma}$ may be different. Then it follows from (6.3.9) that

$$W_{ni} < \hat{\gamma}\frac{c_2}{1-\tilde{\gamma}}\sum_{k=1}^{K} N_k \simeq \frac{c_2\hat{\gamma}}{(1-\tilde{\gamma})(1-2^{-d})}N_K \tag{6.4.3}$$

Defining a work unit as 1 WU $= c_2 N_K$, i.e. approximately the work of $(\nu + \mu)$ smoothing iterations on the finest grid, the cost of a nested iteration is

$$W_{ni} = \hat{\gamma}/[(1-\hat{\gamma})(1-2^{-d})]\mathrm{WU} \tag{6.4.4}$$

Table 6.4.1 gives the number of work units required for nested iteration for a number of cases. The cost of nested iteration is seen to be just a few work units. Hence the fundamental property, which makes multigrid methods so attractive: *multigrid methods can solve many problems to within truncation error at a cost of cN arithmetic operations.* Here N is the number of unknowns, and c is a constant which depends on the problem and on the multigrid method (choice of smoothing method and of the parameters ν_k, μ_k, γ_k). If the cost of the residual $\boldsymbol{b}^K - \boldsymbol{L}^K(\boldsymbol{u}^K)$ is bN, then c need not be larger than a small multiple of b. Other numerical methods for elliptic equations require $O(N^\alpha)$ operations with $\alpha > 1$, achieving $O(N \ln N)$ only in special cases (e.g. separable equations). A class of methods which is competitive with multigrid for linear problems in practice are *preconditioned conjugate gradient* methods. Practice and theory (for special cases) indicate that these require $O(N^\alpha)$ operations, with $\alpha = 5/4$ in two and $\alpha = 9/8$ in three dimensions. Comparisons will be given later.

	d	
γ	2	3
1	16/9	64/49
2	8/3	48/21

Table 6.4.1: Computational cost of nested iteration in work units; $\hat{\gamma} = 1$

6.5 Non-recursive formulation of the basic multigrid algorithm

Structure diagram for fixed multigrid schedule

In FORTRAN, resursion is not allowed: a subroutine cannot call itself. The subroutines MG1, 2, 3 of Section 6.3 cannot, therefore, be implemented directly in FORTRAN. A non-recursive version will, therefore, be presented. At the same time, we will allow grater flexibility in the decision whether to go to a finer or to a coarses grid.

Various *flow diagrams* describing non-recursive multigrid algorithms have been published, for example in [20] and [57]. In order to arrive at a well structured program, we begin by presenting a *structure diagram*. A structure diagram allows much less freedom in the design of the control structure of an algorithm than a flow diagram. We found basically only one way to represent the multigrid algorithm in a structure diagram ([134], [140]). This structure diagram might, therefore, be called the *canonical form* of the basic multigrid algorithm. The structure diagram is given in Figure 6.5.1. This diagram is equivalent to Program 2 calling MG2 to do nmg multigrid iterations with finest grid G^K in Section 6.3. The schedule is fixed and includes the V-, W- and F-cycles. Parts A and B are specified after subroutine MG2 in Section 6.3. Care has been taken that the program also works as a single grid method for $K = 1$.

FORTRAN implementation of *while* clause

Apart from the ***while*** clause, the structure diagram of Figure 6.5.1 can be expressed directly in FORTRAN. A FORTRAN implementation of a ***while*** clause is as follows. Suppose we have the following program

while $(n(K) > 0)$ ***do***
    Statement 1
    $n(K) = ...$
    Statement 2
od

A FORTRAN version of this program is

10 ***if*** $(n(K) > 0)$ ***then***
        Statement 1
        $n(K) = ...$
        Statement 2
        goto 10
    endif

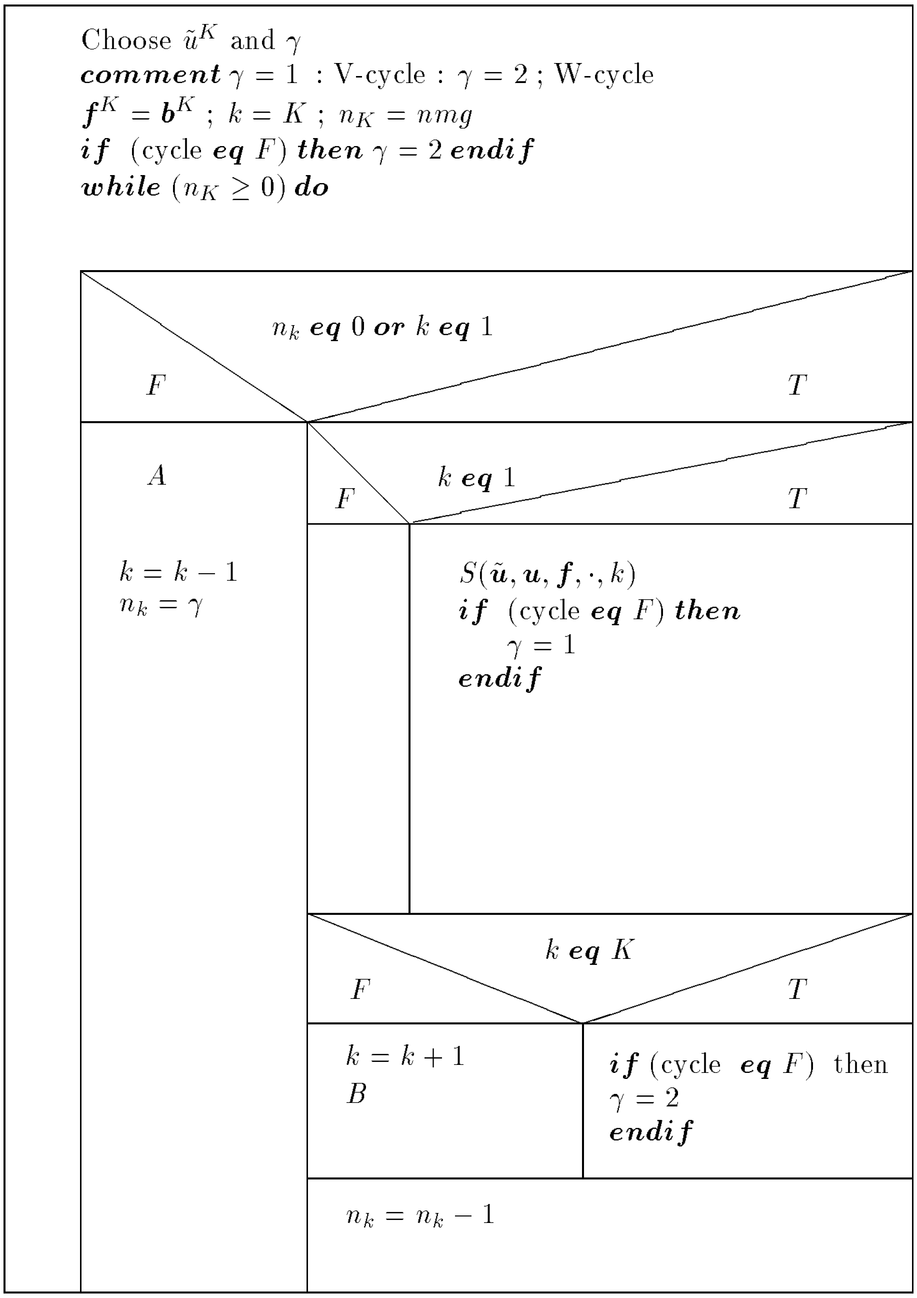


Figure 6.5.1: Structure diagram of non-recursive multigrid algorithm with fixed schedule, including V-, W- and F-cycles.

The ***goto*** statement required for the FORTRAN version of the ***while*** clause is the only ***goto*** needed in the FORTRAN implementation of the structure diagram of Figure 6.5.1. This FORTRAN implementation is quite obvious, and will not be given.

Testing of multigrid software
A simple way to test whether a multigrid algorithm is functioning properly is to measure the residual before and after each smoothing operation, and before and after each visit coarser grids. If a significant reduction of the size of the residual is not found, then the relevant part of the algorithm (smoothing or coarse grid correction) is not functioning propoly. For simple test problems predictions by Fourier smoothing analysis and the contraction number of the multigrid method should be correlated. If the coarse grid problem is solved exactly (a situation usually approximately realized with the W-cycle) the multigrid contraction number should usually be approximately equal to the smoothing factor.

Local smoothing
It may, however, happen, happen that for a well designed multigrid algorithm the contraction number is significantly worse than predicted by the smoothing factor. This may be caused by the fact that Fourier smoothing analysis is locally not applicable. The cause may be a local singularity in the solution. This occurs for example when the physical domain has a reentrant corner The coordinate mapping from the physical domain onto the computational rectangle is singular at that point. It may well be that the the smoothing method does not reduce the residual sufficiently in the neighbourhood of this singularity, a fact that does not remain undetected if the testing procedures recommended above are applied. The remedy is to apply additional local smoothing in a small number of points in the neighbourhood of the singularity. This procedure is recommended in [16], [17], [18], [9], and justified theoretically in [110] and [24]. This local smoothing is applied only to a small number of points, thus the computing work involved is negligible.

6.6 Remarks on software

Multigrid software development can be approached in various ways, two of which will be examined here.

The first approach is to develop general building blocks and diagnostic tools, which helps users to develop their own software for particular applications without having to start from scratch. users will, therefore, need a basic knowledge of multigrid methods. Such software tool are described in [26].

The second approach is to develop *autonomous (black box)* programs, for which the user has to specify only the problem on the finest grid. A program or subroutine may be called

autonomous if it does not require any additional input from the user apart from problem specification, consisting of the linear discrete system of equations to be solved and the right-hand side. The user does not need to know anything about multigrid methods. The subroutine is perceived by the user as if it were just another linear algebra solution method. This approach is adopted by the MGD codes [133], [67], [69], [68], [107], [108], which are available in the NAG library, and by the MGCS code [36].
Of course, it is possible to steer a middle course between the two approaches just outlined, allowing or requiring the user to specify details about the multigrid method to be used, such as offering a selection of smoothing methods, for example. Programs developed in this vein are BOXMG [37], [38], [39], the MG00 series of codes [45], [44], [113] which is available in ELLPACK [100], MUDPACK [3], [2], and the PLTMG code [11], [10], [12]. Exept for PLTMG and MGD, the user specifies the linear differential equation to be solved and the program generates a finite difference discretization. PLTMG generates adaptive finite element disretizations of non-linear equations, and therefore has a much wider scope then the other packages. As a consequence, it is not (meant to be) a solver as fast as the other methods.

By sacrificing generality for efficiency very fast multigrid methods can be obtained for special problems, such as the Poisson or the Helmholtz equation. In MG00 this can be done by setting certain parameters. A very fast multigrid code for the Poisson equation has been developed in [13]. This is probably the fastest two-dimensional Poisson solver in existence.
If one wants to emulate a linear algebraic systems solver, with only the fine grid matrix and right-hand side suplied by the user, then the se of coarse grid Galerkin appoximation (Chapter 5) is mandatory. Coarse grid Galerkin approximation is also required if the coefficients in the differential equations are discontinuous. Coarse grid Galerkin approximation is used in MGD, MGCS and BOXMG; the last two codes use operator-dependent transfer operators and are applicable to problems with discontinuous coefficients.

In an autonomous subroutine the method cannot be adapted to the problem, so that user expertise is not required. The method must, therefore, be very robust. If one of the smoothers that were fund to be robust in Chapter 4 is used, the required degree of robustness is indeed obtained for linear problems.

Non-linear problems may be solved with multigrid codes for linear problems in various ways. The problem may be linearized and solved iteratively, for example by Newton method. This works well as long as the Jacobian of the non-linear discrete problem is non-singular. It may well happen, however, that the given continuous problem has no Frechet derivative. In this case the condition of the Jacobian deteriorates as the grid is refined, and the Newton method does not converge rapidly or not at all. An example of this situation is given [96], [95]. The non-linear multigrid method can be used safely and efficiently, because the global

system is not linearized. A systematic way of applying numerical software outside the class of problems to which the software is directly applicable is the defect correction approach. In [5] and [15] it is pointed out how this ties in with multigrid methods.
Much software is made available on MGNet.

6.7 Comparison with conjugate gradient methods

Although the scope and applicability of multigrid principles are much broader, multigrid methods can be regarded as very efficient ways to solve linear systems arising from discretization of partial differential equations. As such multigrid can be viewed as a technique to accelerate the convergence of basic iterative methods (called smoothers in the multigrid context). Another powerful technique to accelerate basic iterative methods for linear problems that also has come to fruition relatively recently is provided by *conjugate gradient* and related methods. For an introduction to conjugate gradient acceleration of iterative methods, see [63], [48] or (for a very brief synopsis) [141].

It is surprising that, although the algorithm is much simpler, the rate of convergence of conjugate gradient methods is harder to estimate theoretically than for multigrid methods. In two dimensions, $O(N^{5/4})$ computational complexity, and probably $O(N^{9/8})$ in three dimensions seems to hold approximately quite generally for conjugate gradient methods preconditioned by approximate factorization, which comes close to the $O(N)$ of multigrid methods.

Conjugate gradient acceleration of multigrid
The conjugate gradient method can be used to accelerate any iterative method, including multigrid methods. If the multigrid algorithm is well designed and fits the problem it will converge fast, making conjugate gradient acceleration superfluous or even wasteful. If multigrid does not converge fast one may try to remedy this by improving the algorithm (for example, introducing additional local smoothing near singularities, or adapting the smoother to the problem), but if this is impossible because an autonomous (black box) multigrid code is used, or difficult because one cannot identify the cause of the trouble, then conjugate gradient acceleration is an easy and often very efficient way out.

The non-symmetric case
A severe limitation of conjugate gradient methods is their restriction to linear systems with symmetric positive definite matrices. A number of conjugate gradient type methods have been proposed that are applicable to the non-symmetric case. Although no theoretical estimates are available, their rate of convergence is often satisfactory in practice. Two such methods are CGS (conjugate gradient squared), described in [107], [108], [105] and [141], and GMRES, described in [102], [121], [120], [131], [132]. Good convergence is expected if the eigen eigenvalues of $\boldsymbol{A}$ have positive real part, cf. the remarks on convergence in [105].

Comparison of conjugate gradient and multigrid methods
Realistic estimates of the performance in practice of conjugate gradient and multigrid methods by purely theoretical means are possible only for very simple problems. Therefore numerical experiments are necessary to obtain insight and confidence in the efficiency and robustness of a particular method. Numerical experiments can be used only to rule out methods that fail, not to guarantee good performance of a method for problems that have not yet been attempted. Nevertheless, one strives to build up confidence by carefully choosing tests problems, trying to make them representative for large classes of problems, taking into account the nature of the mathematical models that occur in the field of application that one has in mind. For the development of conjugate gradient and multigrid methods, in particular the subject areas of computational fluid dynamics, petroleum reservoir engineering and neutron diffusion are pace-setting.

Important constant coefficient test problems are (4.5.3) and (4.5.4). Problems with constant coefficients are thought to be representative of problems with smoothly varying coefficients. Of course, in the code to be tested the fact that the coefficients are constant should not be exploited. As pointed out in [35], one should keep in mind that for constant coefficient problems the spectrum of the matrix resulting from discretization can have very special properties, that are not present when the coefficients are variable. Therefore one should also carry out tests with variable coefficients, especially with conjugate gradient methods, for which the properties of the spectrum are very important. For multigrid methods, constant coefficient test problems are often more demanding than variable coefficient problems, because it may happen that the smoothing process is not effective for certain combinations of ε and β. This fact goes easily unnoticed with variable coefficients, where the unfavourable values of ε and β perhaps occur only in a small part of the domain.

In petroleum reservoir engineering and neutron diffusion problems quite often equations with strongly discontinuous coefficients appear. For these problems equations (4.5.3) and (4.5.4) are not representative. Suitable test problems with strongly discontinuous coefficients have been proposed in [111] and [79]; a definition of these test problems may also be found in [80]. In Kershaw's problem the domain is non-rectangular, but is a rectangular polygon. The matrix for both problems is symmetric positive definite. With vertex-centered multigrid, operator-dependent transfer operators have to be used, of course.

The four test problems just mentioned, i.e. (4.5.3), (4.5.4) and the problems of Stone and Kershaw, are gaining acceptance among conjugate gradient and multigrid practitioners as standard test problems. Given these test problems, the dilemma of robustness versus efficiency presents itself. Should one try to devise a single code to handle all problems (robustness), or develop codes that handle only a subset, but do so more efficiently than a robust code? This dilemma is not novel, and just as in other parts of numerical mathematics, we expect that both approaches will be fruitful, and no single 'best' code will emerge.

Numerical experiments for the test problems of Stone and Kershaw and equations (4.5.3) and (4.5.4), comparing CGS and multigrid, are described in [107], using ILU and IBLU preconditioning and smoothing. As expected, the rate of convergence of multigrid is unaffected when the mesh size is decreased, whereas CGS slow down. On a 65×65 grid there is not great difference in efficiency. Another comparison of conjugate gradients and multigrid is presented in [40]. Robustness and efficiency of conjugate gradient and multigrid methods are determined to a large extent by the preconditioning and the smoothing method respectively. The smoothing methods that were found to be robust on the basis of Fourier smoothing analysis in Chapter 4 suffice, also as preconditioners. It may be concluded that for medium-sized linear problems conjugate gradient methods are about equally efficient as multigrid in accelerating basic iterative methods. As such they are limited to linear problems, unlike multigrid. On the other hand, conjugate gradient methods are much easier to program, especially when the computational grid is non-rectangular.

7 Finite volume discretization

7.1 Introduction

In this chapter some essentials of finite volume discretization of partial differential equations are summarised. For a more complete elementary introduction, see for example [46] or [89]. We will pay special attention to the handling of discontinuous coefficients, because there seem to be no texts giving a comprehensive account of discretization methods for this situation. Discontinuous coefficients arise in important application areas, such as porous media flows (reservoir engineering), and require special treatment in the multigrid context. Furthermore, hyperbolic systems will be briefly discussed.

7.2 An elliptic equation

Cartesian tensor notation is used with convectional summation over repeated Greek subscripts (not over Latin subscripts). Greek subscripts stand for dimension indices and have range 1, 2, ..., d with d the number of space dimensions. The subscript $_{,\alpha}$ denotes the partial derivative with respect to x_α.
The general single second-order elliptic equation can be written as

$$Lu \equiv -(a_{\alpha\beta}u_{,\alpha})_{,\beta} + (b_\alpha u)_{,\alpha} + cu = s \quad \text{in} \quad \Omega \subset \mathbb{R}^d \tag{7.2.1}$$

The diffusion tensor $a_{\alpha\beta}$ is assumed to be symmetric: $a_{\alpha\beta} = a_{\beta\alpha}$. The boundary conditions will be discussed later. Uniform ellipticity is assumed: there exists a constant $C > 0$ such that

$$a_{\alpha\beta} v_\alpha v_\beta \geq C v_\alpha v_\alpha, \quad \forall \boldsymbol{v} \in I\!R^d \tag{7.2.2}$$

For $d = 2$ this is equivalent to Equation (7.2.9).

The domain Ω

The domain Ω is taken to be the d-dimensional unit cube. This greatly simplifies the construction of the various grids and the transfer operators between them, used in multigrid. In practice, multigrid for finite difference and finite volume discretization can in principe be applied to more general domains, but the description of the method becomes complicated, and general domains will not be discussed here. This is not a serious limitation, because the current main trend in grid generation consists of decomposition of the physical domain in subdomains, each of which is mapped onto a cubic computational domain. In general, such mappings change the coefficients in (7.2.1). As a result, special properties, such as separability or the coefficients being constant, may be lost, but this does not seriously hamper the application of multigrid, because this approach is applicable to (7.2.1) in its general form. This is one of the strengths of multigrid as compared with older methods.

The weak formulation

Assume that a is discontinuous along some manifold $\Gamma \subset \Omega$, which we will call an *interface*; then Equation (7.2.1) now has to be interpreted in the *weak sense*, as follows. From (7.2.1) it follows that

$$(Lu, v) = (s, v) \quad \forall v \in H, \quad (u, v) \equiv \int_\Omega uv d\Omega \tag{7.2.3}$$

where H is a suitable Sobolev space. Define

$$\begin{aligned} a(u, v) &\equiv \int_\Omega a_{\alpha\beta} u_{,\alpha} v_{,\beta} d\Omega - \int_{\partial\Omega} a_{\alpha\beta} u_{,\alpha} n_\beta v d\Gamma \\ b(u, v) &\equiv \int_\Omega (b_\alpha u)_{,\alpha} v d\Omega \end{aligned} \tag{7.2.4}$$

with n_β the x_β component of the outward unit normal on the boundary $\partial\Omega$ of Ω. Application of the Gauss divergence theorem gives

$$(Lu, v) = a(u, v) + b(u, v) + (cu, v) \tag{7.2.5}$$

The weak formulation of (7.2.1) is

$$\text{Find} \quad u \in H \quad \text{such that} \quad a(u, v) + b(u, v) + (cu, v) = (s, v), \quad \forall v \in \tilde{H} \tag{7.2.6}$$

For suitable choices of $H, \tilde{H}$ and boundary conditions, existence and uniqueness of the solution of (7.2.6) has been established. For more details on the weak formulation (not needed here), see for example [31] or [58].

The jump condition

Consider the case with one interface Γ, which divides Ω in two parts Ω_1 and Ω_2, in each of which $a_{\alpha\beta}$ is continuous. At Γ, $a_{\alpha\beta}(x)$ is discontinuous. Let indices 1 and 2 denote quantities on Γ at the side of Ω_1 and Ω_2, respectively. Application of the Gauss divergence theorem to (7.2.5) gives, if u is smooth enough in Ω^1 and Ω^2,

$$a(u,v) = -\int\limits_{\Omega\backslash\Gamma} (a_{\alpha\beta}u_{,\alpha})_{,\beta} v d\Omega + \int\limits_{\Gamma} (a^1_{\alpha\beta}u^1_{,\alpha} - a^2_{\alpha\beta}u^2_{,\alpha})n^1_\beta v d\Gamma \tag{7.2.7}$$

Hence, the solution of (7.2.6), if it is smooth enough in Ω_1 and Ω_2, satisfies (7.2.1) in $\Omega \setminus \Gamma$, together with the following *jump condition* on the interface Γ

$$a^1_{\alpha\beta}u^1_{,\alpha}n^1_\beta = a^2_{\alpha\beta}u^2_{,\alpha}n^1_\beta \quad \text{on} \quad \Gamma \tag{7.2.8}$$

This means that where $a_{\alpha\beta}$ is discontinuous, so is $u_{,\alpha}$. This has to be taken into account in constructing discrete approximations.

Exercise 3.2.1. Show that in two dimensions Equation (7.2.2) is equivalent to

$$a_{11}a_{22} - a^2_{12} > 0 \tag{7.2.9}$$

7.3 A one-dimensional example

The basic ideas of finite difference and finite volume discretization taking discontinuities in $a_{\alpha\beta}$ into account will be explained for the following example

$$-(au_{,1})_{,1} = s, \quad x \in \Omega \equiv (0,1) \tag{7.3.1}$$

Boundary conditions will be given later.

Finite difference discretization

A computational grid $G \subset \bar{\Omega}$ is defined by

$$G = \{x \in I\!R : x = x_j = jh, \ j = 0,1,2,...,n, \ h = 1/n\} \tag{7.3.2}$$

Forward and backward difference operators are defined by

$$\Delta u_j \equiv (u_{j+1} - u_j)/h, \quad \nabla u_j = (u_j - u_{j-1})/h \tag{7.3.3}$$

A finite difference approximation of (7.3.1) is obtained by replacing d/dx by Δ or ∇. A nice symmetric formula is

$$-\frac{1}{2}\{\nabla(a\Delta)+\Delta(a\nabla)\}u_j = s_j, \quad j=1,2,...,n-1 \tag{7.3.4}$$

where $s_j = s(x_j)$ and u_j is the numerical approximation of $u(x_j)$. Written out in full, Equation (7.3.4) gives

$$\begin{aligned}\{-(a_{j-1}+a_j)u_{j-1}+(a_{j-1}+2a_j+a_{j+1})u_j-(a_j+a_{j+1})u_{j+1}\}/2h^2 = s_j,\\ j=1,2,...,n-1\end{aligned} \tag{7.3.5}$$

If the boundary condition at $x=0$ is $u(0)=f$ (Dirichlet), we elimine u_0 from (7.3.5) with $u_0=f$. If the boundary condition is $a(0)u_{,1}(0)=f$ (Neumann), we write down (7.3.5) for $j=0$ and replace the quantity $-(a_{-1}+a_0)u_{-1}+(a_{-1}+a_0)u_0$ by $2f$. If the boundary condition is $c_1u_1(0)+c_2u(0)=f$ (Robin), we again write down (7.3.5) for $j=0$, and replace the quantity just mentioned by $2(f-c_2u_0)a(0)/c_1$. The boundary condition at $x=1$ is handled in a similar way.

An interface problem

In order to show that (7.3.4) can be inaccurate for interface problems, we consider the following example

$$a(x)=\varepsilon, \quad 0<x\le x^*, \quad a(x)=1, \quad x^*<x<1 \tag{7.3.6}$$

The boundary conditions are: $u(0)=0,\ u(1)=1$. The jump condition (7.2.8) becomes

$$\varepsilon \lim_{x\uparrow x^*} u_{,1} = \lim_{x\downarrow x^*} u_{,1} \tag{7.3.7}$$

By postulating a piecewise linear solution the solution of (7.3.1) and (7.3.7) is found to be

$$\begin{aligned} u &= \alpha x,\ 0\le x<x^*, \quad u=\varepsilon\alpha x+1-\varepsilon\alpha, \quad x^*\le x\le 1,\\ \alpha &= 1/(x^*-\varepsilon x^*+\varepsilon)\end{aligned} \tag{7.3.8}$$

Assume $x_k < x^* \le x_{k+1}$. By postulation a piecewise linear solution

$$u_j=\alpha j, \quad 0\le j\le k, \quad u_j=\beta j-\beta n+1, \quad k+1\le j\le n \tag{7.3.9}$$

one finds that the solution of (7.3.5), with the boundary conditions given above, is given by (7.3.9) with

$$\beta=\varepsilon\alpha, \quad \alpha=\left(\varepsilon\frac{1-\varepsilon}{1+\varepsilon}+\varepsilon(n-k)+k\right)^{-1} \tag{7.3.10}$$

Hence

$$u_k=\frac{x_k}{\varepsilon h(1-\varepsilon)/(1+\varepsilon)+(1-\varepsilon)x_k+\varepsilon} \tag{7.3.11}$$

Let $x^* = x_{k+1}$. The exact solution in x_k is

$$u(x_k) = \frac{x_k}{(1-\varepsilon)x_{k+1} + \varepsilon} \tag{7.3.12}$$

Hence, the error satisfies

$$u_k - u(x_k) = O\left(\varepsilon\frac{1-\varepsilon}{1+\varepsilon}h\right) \tag{7.3.13}$$

As another example, let $x^* = x_k + h/2$. The numerical solution in x_k is still given by (7.3.11). The exact solution in x_k is

$$u(x_k) = \frac{x_k}{(1-\varepsilon)x_k + \varepsilon + h(1-\varepsilon)/2} \tag{7.3.14}$$

The error in x_k satisfies

$$u_k - u(x_k) = O\left(\frac{(1-\varepsilon)^2}{\varepsilon(1+\varepsilon)}h\right) \tag{7.3.15}$$

When $a(x)$ is continuous ($\varepsilon = 1$) the error is zero. For general continuous $a(x)$ the error is $O(h^2)$. When $a(x)$ is discontinuous, the error of (7.3.4) increases to $O(h)$.

Finite volume discretization

By starting from the weak formulation (7.2.6) and using *finite volume discretization* one may obtain $O(h^2)$ accuracy for discontinuous $a(x)$. The domain Ω is (almost) covered by cells or finite volumes Ω_j,

$$\Omega_j = (x_j - h/2, x_j + h/2), \quad j = 1, 2, ..., n-1 \tag{7.3.16}$$

Let $v(x)$ be the characteristic function of Ω_j

$$v(x) = 0, \quad x \notin \Omega_j; \quad v(x) = 1, \quad x \in \Omega_j \tag{7.3.17}$$

A convenient unified treatment of both cases: $a(x)$ continuous and $a(x)$ discontinuous, is as follows. We approximate $a(x)$ by a piecewise constant function that has a constant value a_j in each Ω_j. Of course, this works best if discontinuities of $a(x)$ lie at boundaries of finite volumes Ω_j. One may take $a_j = a(x_j)$, or

$$a_j = h^{-1}\int\limits_{\Omega_j} a d\Omega \, .$$

With this approximation of $a(x)$ and v according to (7.3.17) one obtains from (7.2.7)

$$\begin{aligned} a(u,v) &= -\int\limits_{\Omega_j} (au_{,1})_{,1} d\Omega \\ &= -au_{,1}\,\Big|_{x_j-h/2}^{x_j+h/2} \quad \text{if} \quad 1 \le j \le n-1 \end{aligned} \tag{7.3.18}$$

By taking successively $j = 1, 2, .., n-1$, Equation (7.2.6) leads to $n-1$ equations for the $n-1$ unknowns $u_j (u_0 = 0$ and $u_n = 1$ are given), after making further approximations in (7.3.18).

In order to approximate $au_{,1}(x_j + h/2)$ we proceed as follows. Because $au_{,1}$ is smooth, $u_{,1}(x_j + h/2)$ does not exist if $a(x)$ jumps at $x = x_j + h/2$. Hence, it is a bad idea to discretize $u_{,1}(x_j + h/2)$. Instead, we write

$$u\,|_j^{j+1} = \int\limits_{x_j}^{x_{j+1}} u_{,1} dx = \int\limits_{x_j}^{x_{j+1}} \frac{1}{a} a u_{,1} dx \cong (au_{,1})_{j+1/2} \int\limits_{x_j}^{x_{j+1}} \frac{1}{a} dx \tag{7.3.19}$$

where we have exploited the smoothness of $au_{,1}$. We have

$$\int\limits_{x_j}^{x_{j+1}} \frac{1}{a} dx = h/w_j \tag{7.3.20}$$

with w_j the *harmonic average* of a_j and a_{j+1}:

$$w_j \equiv 2a_j a_{j+1}/(a_j + a_{j+1}) \tag{7.3.21}$$

and we obtain the following approximation:

$$(au_{,1})_{j+1/2} \cong w_j(u_{j+1} - u_j)/h \tag{7.3.22}$$

With equations (7.3.18) and (7.3.22), the weak formulation (7.2.6) leads to the following discretization:

$$w_{j-1}(u_j - u_{j-1})/h - w_j(u_{j+1} - u_j)/h = hs_j, \quad j = 1, 2, ..., n-1 \tag{7.3.23}$$

with

$$s_j \equiv h^{-1} \int\limits_{\Omega_j} s dx \ .$$

When $a(x)$ is smooth, $w_j \approx (a_j + a_{j+1})/2$, and we recover the finite difference approximation (7.3.5).
Equation (7.3.23) can be solved in a similar way as (7.3.5) for the interface problem under consideration. Assume $x^* = x_k + h/2$. Hence

$$w_j = \varepsilon, \quad 1 \leq j < k; \quad w_k = 2\varepsilon/(1+\varepsilon); \quad w_j = 1, \quad k < j \leq n-1 \ . \tag{7.3.24}$$

Again postulating a solution as in (7.3.9) one finds

$$\beta = \alpha\varepsilon, \quad \alpha = w/[\varepsilon - w\varepsilon(k+1-n) + wk] \tag{7.3.25}$$

or

$$\alpha = [(1-\varepsilon)/2 + \varepsilon(n-k) + k]^{-1} = h/[(x_k + h/2)(1-\varepsilon) + \varepsilon] \tag{7.3.26}$$

Comparison with (7.3.8) shows that $u_j = u(x_j)$: the numerical error is zero. In more general circumstances the error will be $O(h^2)$. Hence, finite volume discretization is more accurate than finite difference discretization for interface problems.

Exercise 3.3.1 The discrete maximum and l_2 norms are defined by, respectively,

$$|u|_\infty = \max\{|u_j| : 0 \leq j \leq n\}, \quad |u|_0 = h\left\{\sum_{j=0}^{n} u_j^2\right\}^{1/2} \tag{7.3.27}$$

Estimate the error in the numerical solution given by (7.3.9) in these norms.

7.4 Cell-centered discretization in two dimensions

Cell-centered grid

The domain Ω is divided in cells as before, but now the grid points are the centers of the cells, see Figure 7.4.1. The computational grid G is defined by

$$\begin{aligned} G = \{x \in \Omega : x = x_j = (j-s)h,\ j = (j_1, j_2),\ s = (\frac{1}{2}, \frac{1}{2}), \\ h = (h_1, h_2),\ j_\alpha = 1, 2, ..., n_\alpha,\ h_\alpha = 1/n_\alpha\} \end{aligned} \tag{7.4.1}$$

The cell with centre x_j is called Ω_j. Note that in a cell-centered grid there are no grid points on the boundary $\partial\Omega$.

Finite volume discretization in two dimensions

Integration of (7.2.1) over a finite volume Ω_j gives, with $c \equiv s \equiv 0$ for brevity,

$$-\int_{\Gamma_j} a_{\alpha\beta} u_{,\alpha} n_\beta d\Gamma + \int_{\Gamma_j} b_\alpha u n_\alpha d\Gamma = 0 \tag{7.4.2}$$

with Γ_j the boundary of Ω_j and $\boldsymbol{n}$ the outward unit normal. Let us (denote) the "east" part of Γ_j, at $x_1 = (j_1 + \frac{1}{2})h_1$, by Γ_e. On $\Gamma_e, \boldsymbol{n} = (1, 0)$.

The convection term

We write

$$\int_{\Gamma_e} b_1 u d\Gamma \cong h_2 (b_1 u)_{j_1+1/2, j_2} \tag{7.4.3}$$

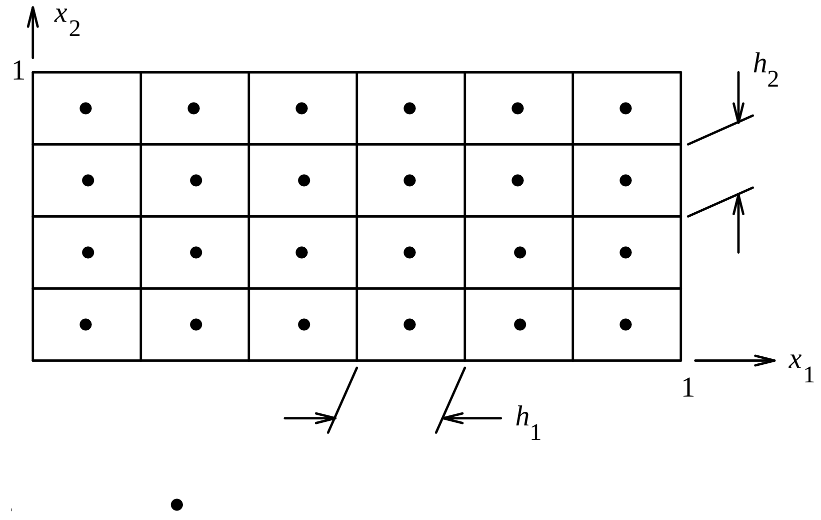


Figure 7.4.1: Cell-centered grid. (• grid points; ————- finite volume boundaries.)

Central discretization gives

$$(b_1u)_{j_1+1/2,j_2} \cong \frac{1}{2}\{(b_1u)_j + (b_1u)_{j_1+1,j_2}\} \tag{7.4.4}$$

Upwind discretization gives

$$(b_1u)_{j_1+1/2,j_2} \cong \frac{1}{2}\{(b_1 + |b_1|u\}_j + \frac{1}{2}\{(b_1 - |b_1|)u\}_{j_1+1,j_2} \tag{7.4.5}$$

If $a_{12} = 0$ then (7.4.4) results in a K-matrix (definition 3.2.5) only if with w defined below

$$\frac{h|b_{j_1+1,j_2}|}{w_{j_1+1/2,j_2}} \leq 2 \ , \quad \frac{h|b_{j_1-1,j_2}|}{w_{j_1-1/2,j_2}} \leq 2 \tag{7.4.6}$$

whereas, if $a_{12} = 0$, (7.4.5) always results in a K-matrix. The advantages of having a K-matrix are

- Monotonicity: absence of numerical wiggles.
- Good behaviour of iterative solution methods, including multigrid, as discussed in Chapter 4.

The diffusion term
We write

$$\int\limits_{\Gamma_e} a_{\alpha 1}u_{,\alpha}d\Gamma \cong h_2(a_{\alpha 1}u_{,\alpha})_{j_1+1/2,j_2} \tag{7.4.7}$$

and approximate the *flux* $F_{j_1+1/2,j_2} = (a_{\alpha 1}u_{,\alpha})_{j_1+1/2,j_2}$ in a similar way as in the one-dimensional case, taking into account the fact that $(u_{,1})_{j+1/2,j_2}$ does not exist, but F and $u_{,2}$ are smooth. We have

$$
\begin{aligned}
u|_{j_1,j_2}^{j_1+1,j_2} &= \int\limits_{x_{j_1,j_2}}^{x_{j_1+1,j_2}} u_{,1}dx_1 = \int\limits_{x_{j_1,j_2}}^{x_{j_1+1,j_2}} \tfrac{1}{a_{11}}(a_{11}u_{,1})dx_1 \\
\int\limits_{x_j}^{x_{j_1+1,j_2}} \tfrac{1}{a_{11}}dx_1 = h_1/w_j \\
&= \int\limits_{x_{j_1,j_2}}^{x_{j_1+1,1/2}} \tfrac{1}{a_{11}}\{(a_{1\alpha}u_{,\alpha}) - a_{12}u_{,2}\}dx_1 \\
&\cong F_{j_1+1/2,j_2} \int\limits_{x_j}^{x_{j_1+1/2,j_2}} \tfrac{1}{a_{11}}dx_1 - (u_{,2})_{j_1+1/2,j_2} \int\limits_{x_j}^{x_{j_1+1,1/2}} a_{12}/a_{11}dx_1
\end{aligned}
\tag{7.4.8}
$$

We now assume that $a_{12} = 0$, or else that $u_{,2}$ may be approximated by

$$
(u_{,2})_{j_1+1/2,j_2} \cong \frac{1}{4h_2}\{u|_{j_1,j_2-1}^{j_1,j_2+1} + u|_{j_1+1,j_2-1}^{j_1+1,j_2+1}\} \tag{7.4.9}
$$

This will be accurate only of $a_{\alpha\beta}$ is smooth at the north and south edges, the general case seems not to have been investigated.

We obtain, with

$$
w_{j_1+1/2,j_2} \equiv h/\int\limits_{x_{j_1,j_2}}^{x_{j_1+1,j_2}} \frac{1}{a_{11}}dx_1 \tag{7.4.10}
$$

$$
F_{j_1+1/2,j_2} \cong w_{j_1+1/2,j_2}u|_{j_1,j_2}^{j_1+1,j_2}/h + (u_{,2})_{j_1+1/2,j_2} \int\limits_{x_{j_1,j_2}}^{x_{j_1+1,j_2}} a_{12}/a_{11}dx_1
$$

With (7.4.9) the K-matrix property is lost. The off-diagonal elements with the wrong side are much smaller than those generated by central discretization of the convection term at high Péclet number, and usually results obtained and performance of iterative methods are still satisfactory. See [141] for more details, including a discretization that gives a K-matrix for $a_{12} \neq 0$.

7.5 A hyperbolic system

Hyperbolic system of conservation laws

In this section we consider the following *hyperbolic system of conservation laws*:

$$
\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} + \frac{\partial \boldsymbol{g}(\boldsymbol{u})}{\partial y} = s, \quad (x,y) \in \Omega, \quad t \in (0,T] \tag{7.5.1}
$$

where

$$\boldsymbol{u} : [0,T] \times \Omega \to S_a \subset I\!R^p, \quad s : [0,T] \times \Omega \to I\!R^p, \quad \boldsymbol{f}, \boldsymbol{g} : S_a \to I\!R^p \tag{7.5.2}$$

Here S_a is the set of admissible states. For example, if one of the p unknowns, u_i say, is the fluid density or the speed of sound in a fluid mechanics application, then $u_i < 0$ is not admissible. Equation (7.5.1) is a system of p equations with p unknowns. Here we abandon Cartesian tensor notation for the more convenient notation above. Equation (7.5.1) is assumed to be *hyperbolic*.

Definition 7.5.1 Equation (7.5.1) is called hyperbolic with respect to t if there exist for all $\varphi \in [0, 2\pi)$ and admissible $\boldsymbol{u}$ a real diagonal matrix $\boldsymbol{D}(\boldsymbol{u}, \varphi)$ and non-singular matrix $\boldsymbol{R}(\boldsymbol{u}, \varphi)$ such that

$$\boldsymbol{A}(\boldsymbol{u}, \varphi)\boldsymbol{R}(\boldsymbol{u}, \varphi) = \boldsymbol{R}(\boldsymbol{u}, \varphi)\boldsymbol{D}(\boldsymbol{u}, \varphi) \tag{7.5.3}$$

where

$$\boldsymbol{A}(\boldsymbol{u}, \varphi) = \cos\ \varphi \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial \boldsymbol{u}} + \sin\ \varphi \frac{\partial \boldsymbol{g}(\boldsymbol{u})}{\partial \boldsymbol{u}} \tag{7.5.4}$$

The main example to date of systems of type (7.5.1) to which multigrid methods have been applied successfully are the Euler equations of gas dynamics. See [34] for more details on the mathematical properties of these equations and of hyperbolic systems in general. For numerical aspects of hyperbolic systems, see [101] or [104].

For the discretization of (7.5.1), schemes of Law-Wendroff type (see [101]) have long been popular and still are widely used. These schemes are explicit and, for time-dependent problems, there is no need for multigrid: stability and accuracy restrictions on the time step Δt are about equaly severe. If the time-dependent formulation is used solely as a means to compute a steady state, then one would like to be unrestricted in the choice of Δt ands/or use artificial means to get rid of the transients quickly.

In [92] a method has been proposed to do this using multiple grids. This method has been developed further in [76], [30] and [77]. The method is restricted to Lax-Wendroff type formulations.

Finite volume discretization

Following the main trend in contemporary computational fluid dynamics, we discuss only the cell-centered case. The grid is given in Figure 7.4.1. Integration of (7.5.1) over Ω_j gives, using the Gauss divergence theorem,

$$\frac{d}{dt}\int\limits_{\Omega_j} \boldsymbol{u} d\Omega + \int\limits_{\Gamma_j} (\boldsymbol{f}(\boldsymbol{u})n_x + \boldsymbol{g}(\boldsymbol{u})n_y) d\Gamma = \int\limits_{\Omega_j} S d\Omega \tag{7.5.5}$$

where Γ_j is the boundary of Ω_j. With the approximations

$$\int\limits_{\Omega_j} \boldsymbol{u} d\Omega \simeq |\Omega_j| \boldsymbol{u}_j, \quad \int\limits_{\Omega_j} S d\Omega \simeq |\Omega_j| \boldsymbol{s}_j \tag{7.5.6}$$

where $|\Omega_j|$ is the area of Ω_j, Equation (7.5.5) becomes

$$|\Omega_j| d\boldsymbol{u}_j/dt + \int\limits_{\Gamma_j} (\boldsymbol{f}(\boldsymbol{u}) n_x + \boldsymbol{g}(\boldsymbol{u}) n_y) d\Gamma = |\Omega_j| \boldsymbol{s}_j \tag{7.5.7}$$

The time discretization will not be discussed. The space discretization takes place by approximating the integral over Γ_j.
Let $A = x_j + (h_1/2, -h_2/2)$, $B = x_j + (h_1/2, h_2/2)$, so that AB is part of Γ_j. On AB, $n_x = 1$ and $n_y = 0$. We write

$$\int\limits_A^B \boldsymbol{f}(\boldsymbol{u}) dx_2 \cong h_2 \boldsymbol{f}(\boldsymbol{u})_C \tag{7.5.8}$$

with C the midpoint of AB. *Central space discretization* is obtained with

$$\boldsymbol{f}(\boldsymbol{u})_C \cong \frac{1}{2} \boldsymbol{f}(\boldsymbol{u}_j) + \frac{1}{2} \boldsymbol{f}(\boldsymbol{u}_{j+e_1}) \tag{7.5.9}$$

In the presence of shocks, this does not lead to the correct weak solution, unless thermodynamic irreversibility is enforced. This may be done by introducing artificial viscosity, an approach followed in [72]. Another approach is to use *upwind space discretization*, obtained by *flux splitting*:

$$\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}^+(\boldsymbol{u}) + \boldsymbol{f}^-(\boldsymbol{u}) \tag{7.5.10}$$

with $\boldsymbol{f}^\pm(\boldsymbol{u})$ choosen such that the eigenvalues of the Jacobians of $\boldsymbol{f}^\pm(\boldsymbol{u})$ satisfy

$$\lambda(\partial \boldsymbol{f}^+/\partial \boldsymbol{u}) \geq 0, \quad \lambda(\partial \boldsymbol{f}^-/\partial \boldsymbol{u}) \leq 0 \tag{7.5.11}$$

There are many splittings satisfying (7.5.11). For a survey of flux splitting, see [64] and [126]. With upwind discretization, $\boldsymbol{f}(\boldsymbol{u})_C$ is approximated by

$$\boldsymbol{f}(\boldsymbol{u})_C \cong \boldsymbol{f}^+(\boldsymbol{u}_j + \boldsymbol{f}^-(\boldsymbol{u}_{j+e_1})) \tag{7.5.12}$$

The implementation of boundary conditions for hyperbolic systems is not simple, and will not be discussed here; the reader is referred to the literature mentioned above.

Exercise 7.5.1 Show that the flux splitting (7.4.5) satisfies (7.5.11).

8 Conclusion

An introduction has been presented to the application of multigrid methods to the numerical solution of elliptic and hyperbolic partial differential equations.

Because robustness is stongly influenced by the smoothing method used, much attention has been given to smoothing analysis, and many possible smoothing methods have been presented.

An attempt has been made to review much of the literature, to help the reader to find his way quickly to material relevant to his interests. For more information, see [141].

In this book application of multigrid to the eqations of fluid dynamics is reviewed, a topic not covered here. There the full potential equation, the Euler equations, the compressible Navier-Stokes equations and the incompressible Navier-Stokes and Boussinesq equations are treated.

The principles discussed in these notes hold quite generally, making solution possible at a cost of a few work units, as discussed in chapter 6, for problems more difficult than considered here.

References

1. Braunschweig/Wiesbaden, 1985. Vieweg. Proceedings, Oberwolfach 1984. Notes on Numerical Fluid Mechanics 11.

2. J.C. Adams. FMG results with the multigrid software package MUDPACK. In J. Mandel, S.F. Mccormick, J.E. Dendy, Jr., C. Farhat, G. Lonsdale, S.V. Parter, J.W. Ruge, and K. Stüben, editors, *Proc. of the Fourth Copper Mountain Conference on Multigrid Methods*, pages 1–12, Philadelphia, 1989. SIAM.

3. J.C. Adams. MUDPACK: Multigrid portable FORTRAN software for the efficient solution of linear elliptic partial differential equations. *Appl. Math. and Comp.*, 34:113–146, 1989.

4. G.P. Astrakhantsev. An interative method of solving elliptic net problems. *U.S.S.R. Comp. Math. and Math. Phys.*, 11 no.2:171–182, 1971.

5. W. Auzinger and H.J. Stetter. Defect correction and multigrid iterations. In *Hackbusch and Trottenberg (1982)*, pages 327–351.

6. O. Axelsson, S. Brinkkemper, and V.P. Il'in. On some versions of incomplete block matrix factorization iterative methods. *Lin. Algebra Appl. 59*, pages 3–15, 1984.

7. O. Axelsson and B. Polman. On approximate factorization methods for block matrices suitable for vector and parallel processors. *Lin. Alg. Appl. 77*, pages 3–26, 1986.

8. N.S. Bachvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Comp. Math. and Math. Phys 6*, pages 101–135, 1966.

9. D. Bai and A. Brandt. Local mesh refinement multilevel techniques. *SIAM J. Sci Stat. Comp. 8*, pages 109–134, 1987.

10. R.E. Bank. A comparison of two multi-level iterative methods for nonsymmetric and indefinite elliptic finite element equations. *SIAM J. Numer. Anal.*, 18:724–743, 1981.

11. R.E. Bank. A multi-level iterative method for nonlinear elliptic equations. In M.H. Schultz, editor, *Elliptic problem solvers*, pages 1–16, New York, 1981. Academic Press.

12. R.E. Bank and A.H. Sherman. An adaptive multi-level method for elliptic boundary value problems. *Computing*, 26:91–105, 1981.

13. D. Barkai and A. Brandt. Vectorized multigrid poisson solver for the cdc cyber 205. *Appl. Math. Comput. 13*, pages 215–227, 1983.

14. P. Bastian and G. Horton. Parallization of robust multi-grid methods: ILU factorization and frequency decomposition method. In *Hackbusch and Rannacher (1990)*, pages 24–36.

15. K. Böhmer, P. Hemker, and H. Stetter. The defect correction approach. *Computing, Suppl.*, 5:1–32, 1984.

16. A. Brandt. Guide to multigrid development. In *Hackbusch and Trottenberg (1982)*, pages 220–312.

17. A. Brandt. Multilevel computations: Review and recent developments. In *McCormick (1988)*, pages 35–62.

18. A. Brandt. The Weizmann Institute research in multilevel computation: 1988 report. In *J. Mandel et al. (1989)*, pages 13–53.

19. A. Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In H. Cabannes and R. Temam, editors, *Proc. Third Int. Conf. on Numerical Methods in Fluid Mechanics, Vol. 1*, pages 82–89, Berlin, 1973. Springer-Verlag. Lecture Notes in Physics 18.

20. A. Brandt. Multi-level adaptive solutions to boundary value problems. *Math. Comp.*, 31:333–390, 1977.

21. A. Brandt. Multi-level adaptive techniques MLAT for partial differential equations: Ideas and software. In J. Rice, editor, *Proceedings of Symposium on Mathematical Software*, pages 277–318, New York, 1977. Academic Press.

22. A. Brandt. Multilevel adaptive computations in fluid dynamics. *AIAA J.*, 18:1165–1172, 1980.

23. A. Brandt. Multigrid techniques: 1984 guide, with applications to fluid dynamics. GMD Studien no. 85, P.O. Box 1240, D-2505, Sankt Augustin, Germany, 1984. Gesellschaft für Mathematik und Datenverarbeitung.

24. A. Brandt. Rigorous quantitative analysis of multigrid. *SIAM J. Numer. Anal.*, 1994. To appear.

25. A. Brandt and A.A. Lubrecht. Multilevel matrix multiplication and fast solution of integral equations. *J. Comp. Phys.*, 90:348–370, 1990.

26. A. Brandt and D. Ophir. Gridpack: Toward unification of general grid programming. In *Engquist and Smedsaas (1983)*, pages 269–290.

27. W.L. Briggs. *A multigrid tutorial.* SIAM, Philadelphia, 1987.

28. W.L. Briggs and S.F. McCormick. Introduction. chapter 1. In *McCormick (1987)*, 1987.

29. T.F. Chan and H.C. Elman. Fourier analysis of iterative methods for elliptic boundary value problems. *SIAM Rev. 31*, pages 20–49, 1989.

30. R.V. Chima and G.M. Johnson. Efficient solution of the Euler and Navier-Stokes equations with a vectorized multiple-grid algorithm. *AIAA J.*, 23:23–32, 1985.

31. Ph.G. Ciarlet. *The Finite Element Method for elliptic problems.* North-Holland, Amsterdam, 1978.

32. G. Cimmino. La ricerca scientifica ser. ii 1. In *Pubblicazioni dell'Instituto per le Applicazioni del Calculo 34*, pages 326–333, 1938.

33. P. Concus, G.H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comp. 6*, pages 220–252, 1985.

34. R. Courant and K.O. Friedrichs. *Supersonic Flow and Shock Waves.* Springer-Verlag, New York, 1949.

35. A.R. Curtiss. On a property of some test equations for finite difference or finite element methods. *IMA J. Numer. Anal.*, 1:369–375, 1981.

36. P.M. de Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J. of Comp. and Appl. Math.*, 3:1–27, 1990.

37. J.E. Dendy Jr. Black box multigrid. *J. Comp. Phys.*, 48:366–386, 1982.

38. J.E. Dendy Jr. Black box multigrid for nonsymmetric problems. *Appl. Math. Comp.*, 13:261–284, 1983.

39. J.E. Dendy Jr. Black box multigrid for systems. *Appl. Math. Comp.*, 19:57–74, 1986.

40. J.E. Dendy, Jr. and J.M. Hyman. Multi-grid and ICCG for problems with interfaces. In M.H. Schultz, editor, *Elliptic problem solvers*, pages 247–253, New York, 1981. Academic Press.

41. T. Dupont, R.P. Kendall, and H.H. Jr. Rachford. An approximate factorization procedure for solving self-adjoint difference equations. *SIAM J. Numer. Anal. 5*, pages 559–573, 1968.

42. B. Enquist and T. Smedsaas, editors. *PDE software: Modules interfaces and systems*, Amsterdam, 1984. North-Holland. Procs. IFIP WG 2.5 Working Conference.

43. R.P. Fedorenko. The speed of convergence of one iterative process. *USSR comp. Math. and Math. Phys.*, 4 no. 3:227–235, 1964.

44. H. Foerster and K. Witsch. Multigrid software for the solution of elliptic problems on rectangular domains: MG00 (Release 1). In *Hackbusch and Trottenberg (1982)*, pages 427–460.

45. H. Foerster and K. Witsch. On efficient multigrid software for elliptic problems on rectangular domains. *Math. Comp. Simulation*, 23:293–298, 1981.

46. G.E. Forsythe and W.R. Wasow. *Finite Difference Methods for Partial Diffetrential Equations.* Wiley, New York etc., 1960.

47. P.O. Frederickson. Fast approximate inversion of large elliptic systems. Report 7–75, Lakehead University, Thunderbay, Canada, 1975.

48. G.H. Golub and C.F. Van Loan. *Matrix Computations (second edition).* The Johns Hopkins University Press, Baltimore, 1989.

49. W. Hackbusch. Multigrid convergence theory. In *Hackbusch and Trottenberg (1982)*, pages 177–219.

50. W. Hackbusch. Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme. Report 76–12, Universität Köln, 1976.

51. W. Hackbusch. On the convergence of a multi-grid iteration applied to finite element equations. Technical Report 77-8, Universität zu Köln, 1977.

52. W. Hackbusch. On the multi-grid method applied to difference equations. *Computing*, 20:291–306, 1978.

53. W. Hackbusch. Survey of convergence proofs for multi-grid iterations. In J. Frehse, D. Pallaschke, and U. Trottenberg, editors, *Special topics of applied mathematics*, pages 151–164, Amsterdam, 1980. North-Holland. Proceedings, Bonn, Oct. 1979.

54. W. Hackbusch. On the convergence of multi-grid iterations. *Beiträge Numer. Math. 9*, pages 231–329, 1981.

55. W. Hackbusch, editor. *Efficient solutions of elliptic systems*, Braunschweig/Wiesbaden, 1984. Vieweg. Notes on Numerical Fluid Mechanics 10.

56. W. Hackbusch. Parabolic multi-grid methods. In R. Glowinski and J.L. Lions, editors, *Computing methods in applied sciences and engineering VI*, pages 189–197, Amsterdam, 1984. North-Holland. Proc. of the Sixth Int. Symposium, Versailles, Dec. 1983.

57. W. Hackbusch. *Multi-grid methods and applications*. Springer-Verlag, Berlin, 1985.

58. W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen*. Teubner, Stuttgart, 1986.

59. W. Hackbusch, editor. *Robust Multi-Grid Methods*, Braunschweig, 1989. Vieweg. Proc. 4th GAMM-Seminar, Kiel, 1988. Notes on Numerical Fluid Mechanics 23.

60. W. Hackbusch and R. Rannacher, editors. *Numerical treatment of the Navier-Stokes equations*, Braunschweig, 1990. Vieweg. Notes on Numerical Fluid Mechanics 30.

61. W. Hackbusch and U. Trottenberg, editors. *Multigrid methods*, Berlin, 1982. Springer-Verlag. Lecture Notes in Mathematics (960).

62. W. Hackbusch and U. Trottenberg, editors. *Multigrid Methods II*, Berlin, 1986. Springer-Verlag. Lecture Notes in Mathematics 1228.

63. L.A. Hageman and D.M. Young. *Applied iterative methods*. Academic Press, New York, 1981.

64. A. Harten, P.D. Lax, and B. Van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25:35–61, 1983.

65. P.W. Hemker. The incomplete LU-decomposition as a relaxation method in multi-grid algorithms. In J.H. Miller, editor, *Boundary and Interior Layers - Computational and Asymptotic Methods*, pages 306–311, Dublin, 1980. Boole Press.

66. P.W. Hemker. On the order of prolongations and restrictions in multigrid procedures. *J. of Comp. and Appl. Math.*, 32:423–429, 1990.

67. P.W. Hemker, R. Kettler, P. Wesseling, and P.M. De Zeeuw. Multigrid methods: Development of fast solvers. *Appl. Math. Comp.*, 13:311–326, 1983.

68. P.W. Hemker and P.M. de Zeeuw. Some implementations of multigrid linear systems solvers. In *Paddon and Holstein (1985)*, pages 85–116.

69. P.W. Hemker, P. Wesseling, and P.M. de Zeeuw. A portable vector-code for autonomous multigrid modules. In *Enquist and Smedsaas (1984)*, pages 29–40.

70. E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*. John Wiley and Sons, New York, 1966.

71. A. Jameson. Solution of the euler equations for two-dimensional flow by a multigrid method. *Appl. Math. and Comp. 13*, pages 327–355, 1983.

72. A. Jameson. Computational transonics. *Comm. Pure Appl. Math.*, 41:507–549, 1988.

73. A. Jameson and T.J. Baker. Multigrid solution of the euler equations for aircraft configurations. AIAA-Paper 84-0093, 1984.

74. A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the euler equations by finite volume methods using Runge-Kutta time stepping schemes. AIAA Paper 81-1259, 1981.

75. M. Jayaram and A. Jameson. Multigrid solution of the navier-stokes equations for flow over wings. AIAA-Paper 88-0705, 1988.

76. G.M. Johnson. Multiple-grid convergence acceleration of viscous and inviscid flow computations. *Appl. Math. Comp.*, 13:375–398, 1983.

77. G.M. Johnson and J.M. Swisshelm. Multiple-grid solution of the three-dimensional Euler and Navier-Stokes equations. In *Soubbaramayer and Boujot(1985)*, pages 286–290.

78. S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin de l'Academie Polonaise des Sciences et Lettres A*, 35:355–357, 1937.

79. D. S. Kershaw. The incomplete Choleski-conjugate gradient method for the iterative solution of systems of linear equations. *J. Comp. Phys.*, 26:43–65, 1978.

80. R. Kettler. Analysis and comparison of relaxation schemes in robust multigrid and conjugate gradient methods. In *Hackbusch and Trottenberg (1982)*, pages 502–534.

81. R. Kettler and P. Wesseling. Aspects of multigrid methods for problems in three dimensions. *Appl. Math. Comp.*, 19:159–168, 1986.

82. M. Khalil. Local mode smoothing analysis of various incomplete factorization iterative methods. In *Hackbusch (1989)*, pages 155–164.

83. M. Khalil. *Analysis of Linear Multigrid Methods for Elliptic Differential Equations with Discontinuous and Anisotropic Coefficients*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 1989.

84. J. Mandel, S.F. McCormick, J.E. Dendy, Jr., C. Farhat, G. Lonsdale, S.V. Parter, J.W. Ruge, and K. Stüben, editors. *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods*, Philadelphia, 1989. SIAM.

85. S.F. McCormick, editor. *Multigrid methods*, Philadelphia, 1987. SIAM. Frontiers in Applied Mathematics 3.

86. S.F. McCormick, editor. *Multigrid methods*, New. York, 1988. Marcel Dekker Inc. Lecture Notes in Pure and Applied Mathematics 110.

87. 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. *Math. Comp.*, 31:148–162, 1977.

88. J.A. Meijerink and H.A. Van der Vorst. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *J. Comput. Phys.*, 44:134–155, 1981.

89. A.R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations*. Wiley, Chichester, 1980.

90. K.W. Morton and M.J. Baines, editors. *Numerical Methods for Fluid Dynamics II*, Oxford, 1986. Clarendon Press.

91. S. Murata, N. Satofuka, and T. Kushiyama. Parabolic multi-grid method for incompressible viscous flows using a group explicit relaxation scheme. *Comp. & Fluids*, 19:33–41, 1991.

92. R.H. Ni. A multiple grid scheme for solving Euler equations. *AIAA J.*, 20:1565–1571, 1982.

93. R.A. Nicolaides. On multiple grid and related techniques for solving discrete elliptic systems. *J. Comp. Phys.*, 19:418–431, 1975.

94. R.A. Nicolaides. On the l^2 convergence of an algorithm for solving finite element equations. *Math. Comp.*, 31:892–906, 1977.

95. Z. Nowak. Calculations of transonic flows around single and multi-element airfoils on a small computer. In *Braess and Hackbusch (1985)*, pages 84–101.

96. Z. Nowak and P. Wesseling. Multigrid acceleration of an iterative method with application to compressible flow. In R. Glowinski and J.-L. Lions, editors, *Computing Methods in Applied Sciences and Engineering VI*, pages 199–217, Amsterdam, 1984. North-Holland.

97. K.-D. Oertel and K. Stüben. Multigrid with ILU-smoothing: Systematic tests and improvements. In *Hackbusch(1989)*, pages 188–199.

98. D.J. Paddon and H. Holstein, editors. *Multigrid Methods for Integral and Differential Equations*, Oxford, 1985. Clarendon Press.

99. B. Polman. Incomplete blockwise factorizations of (block) H-matrices. *Lin. Alg. Appl.*, 90:119–132, 1987.

100. R. Rice and R.F. Boisvert. *Solving Elliptic Systems Using ELLPACK*, volume 2 of *Springer Series in Comp. Math.* Springer-Verlag, Berlin, 1985.

101. R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial Value Problems.* John Wiley, New York, 1967.

102. Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–869, 1986.

103. J.J.F.M. Schlichting and H.A. Van der Vorst. Solving 3D block bidiagonal linear systems on vector computers. *J. of Comp. and Appl. Math.*, 27:323–330, 1989.

104. G.A. Sod. *Numerical Methods in Fluid Dynamics: Initial and Initial Boundary-Value Problems.* Cambridge University Press, Cambridge, 1985.

105. P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

106. P. Sonneveld and B. Van Leer. A minimax problem along the imaginary axis. *Nieuw Archief voor Wiskunde*, 3:19–22, 1985.

107. P. Sonneveld, P. Wesseling, and P.M. de Zeeuw. Multigrid and conjugate gradient methods as convergence acceleration techniques. In *Paddon and Holstein (1985)*, pages 117–168, 1985.

108. P. Sonneveld, P. Wesseling, and P.M. de Zeeuw. Multigrid and conjugate gradient acceleration of basic iterative methods. In *Morton and Baines (1986)*, pages 347–368, 1986.

109. Soubbaramayer and J.P. Boujot, editors. *Ninth International Conference on Numerical Methods in Fluid Dynamics*, Berlin, 1985. Springer-Verlag.

110. R.P. Stevenson. *On the Validity of local mode analysis of multi-grid methods.* PhD thesis, University of Utrecht, The Netherlands, 1990.

111. H.L. Stone. Iterative solution of implicit approximations of multi-dimensional partial difference equations. *SIAM J. Num. Anal.*, 5, 1968.

112. K. Stüben and U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In *Hackbusch and Trottenberg (1982)*, pages 1–176, 1982.

113. K. Stüben, U. Trottenberg, and K. Witsch. Software development based on multigrid techniques. In *Engquist and Smedsaas (1984)*, pages 241–267.

114. C.-A. Thole and U. Trottenberg. Basic smoothing procedures for the multigrid treatment of elliptic 3D-operators. *Appl. Math. Comp.*, 19:333–345, 1986.

115. P.J. Van der Houwen. *Construction of integration formulas for initial-value problems.* North-Holland, Amsterdam, 1977.

116. H.A. Van der Vorst. A vectorizable variant of some ICCG methods. *SIAM J. Sci. Stat. Comp.*, 3:350–356, 1982.

117. H.A. Van der Vorst. The performance of FORTRAN implementations for preconditioned conjugate gradients on vector computers. *Parallel Computing*, 3:49–58, 1986.

118. H.A. Van der Vorst. High performance preconditioning. *SIAM J. Sci. Stat. Comp.*, 10:1174–1185, 1989.

119. H.A. Van der Vorst. ICCG and related methods for 3D problems on vector computers. *Computer Physics Comm.*, 53:223–235, 1989.

120. H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Numer. L.A.A.*, 1993. to appear.

121. H.A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, 48:327–341, 1993.

122. A.J. Van der Wees. FAS multigrid employing ILU/SIP smoothing: a robust fast solver for 3D transonic potential flow. In *Hackbusch and Trottenberg (1986)*, pages 315–331.

123. A.J. Van der Wees. Impact of multigrid smoothing analysis on three-dimensional potential flow calculations. In *Mandel et al. (1989)*, pages 399–416.

124. A.J. Van der Wees. Robust calculation of 3D potential flow based on the nonlinear FAS multi-grid method and a mixed ILU/SIP algorithm. In J.G. Verwer, editor, *Colloquium Topics in Applied Numerical Analysis*, pages 419–459, Amsterdam, 1984. Centre for Mathematics and Computer Science. CWI Syllabus.

125. A.J. Van der Wees. *A nonlinear multigrid method for three-dimensional transonic potential flow*. PhD thesis, Delft University of Technology, 1988.

126. B. Van Leer. On the relation between the upwind-differencing schemes of Godunov, Enquist-Osher and Roe. *SIAM J. Sci. Stat. Comp.*, 5:1–20, 1984.

127. B. Van Leer, C.-H. Tai, and K.G. Powell. Design of optimally smoothing multistage schemes for the euler equations. AIAA Paper 89-1933-CP, 1989.

128. S.P. Vanka and K. Misegades. Vectorized multigrid fluid flow calculations on a cray x-mp48. AIAA Paper 86-0059, 1986.

129. R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

130. C.H. Venner. *Multilevel solution of the EHL line and point contact problems*. PhD thesis, Twente University, Enschede, The Netherlands, 1991.

131. C. Vuik. Further experiences with GMRESR. *Supercomputer*, 55:13–27, 1993.

132. C. Vuik. Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *Int. J. for Num. Meth. Fluids*, 16:507–523, 1993.

133. P. Wesseling. A robust and efficient multigrid method. In *Hackbusch and Trottenberg (1982)*, pages 614–630.

134. P. Wesseling. Two remarks on multigrid methods. In *Hackbusch (1989)*, pages 209–216.

135. P. Wesseling. Numerical solution of the stationary navier-stokes equations by means of a multiple grid method and newton iteration. Technical Report Report NA-18, Dept. of Technical Math. and Inf., Delft University of Technology, 1977.

136. P. Wesseling. The rate of convergence of a multiple grid method. In G.A. Watson, editor, *Numerical Analysis. Proceedings, Dundee 1979*, pages 164–184, Berlin, 1980. Springer-Verlag. Lecture Notes in Math. 773.

137. P. Wesseling. Theoretical and practical aspects of a multigrid method. *SIAM J. Sci. Stat. Comp. 3*, pages 387–407, 1982.

138. P. Wesseling. Multigrid solution of the Navier-Stokes equations in the vorticity-streamfunction formulation. In *Hackbusch (1984)*, pages 145–154, 1984.

139. P. Wesseling. Linear multigrid methods. In *McCormick (1987)*, pages 31–56, 1987.

140. P. Wesseling. Multigrid methods in computational fluid dynamics. *Z. angew. Math. Mech.*, 70:T337–T348, 1990.

141. P. Wesseling. *An introduction to multigrid methods.* John Wiley & Sons, Chichester, 1992.

142. P. Wesseling. The role of incomplete LU-factorization in multigrid methods. In W. Hackbusch and C. Wittum, editors, *Incomplete decompositions (ILU) - Algorithms, theory and applications*, pages 202–214, Vieweg, 1993. Braunschweig.

143. P. Wesseling and P. Sonneveld. Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In R. Rautmann, editor, *Approximation methods for Navier-Stokes problems*, pages 543–562, Berlin, 1980. Springer-Verlag. Lecture Notes in Math. 771.

144. G. Wittum. *Distributive iterationen für indefinite Systeme als Glätter in Mehrgitterverfahren am Beispiel der Stokes- und Navier-Stokes-Gleichungen mit Schwerpunkt auf unvollständingen Zerlegungen.* PhD thesis, Christan-Albrechts Universität, Kiel, 1986.

145. G. Wittum. Linear iterations as smoothers in multigrid methods: Theory with applications to incomplete decompositions. *Impact of Comp. Science Engng.*, 1:180–215, 1989.

146. G. Wittum. Multi-grid methods for Stokes and Navier-Stokes equations with transforming smoothers: Algorithms and numerical results. *Numer. Math.*, 54:543–563, 1989.

147. G. Wittum. On the robustness of ILU smoothing. *SIAM J. Sci. Stat. Comp.*, 10:699–717, 1989.

148. G. Wittum. On the convergence of multi-grid methods with transforming smoothers. *Num. Math.*, 57:15–38, 1990.

149. G. Wittum. R-transforming smoothers for the incompressible Navier- Stokes equations. In W. Hackbusch and R. Rannacher, editors, *Numerical treatment of the Navier-Stokes equations*, pages 153–162, Braunschweig, 1990. Vieweg. Notes on Numerical Fluid Mechanics 30.

150. G. Wittum. The use of fast solvers in computational fluid dynamics. In P. Wesseling, editor, *Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 574–581, Braunschweig/Wiesbaden, 1990. Vieweg-Verlag. Notes on Numerical Fluid Mechanics 29.

151. D.M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.