1

INTRODUCTION

We start this chapter with a short introduction of some of the equations that we will treat in this book in Section 1.1 and with some information on grids and discretization approaches in Section 1.2. In Section 1.3 we will introduce some of our terminology. The 2D Poisson equation with Dirichlet boundary conditions is the prototype of an elliptic boundary value problem. It is introduced and discussed in Section 1.4. In Section 1.5 we will take a first glance at multigrid and obtain an impression of the multigrid idea. Some facts and methods on basic numerics are listed in Section 1.6.

1.1 TYPES OF PDEs

As we will see in this book, elliptic boundary value problems are the type of problem to which multigrid methods can be applied very efficiently. However, multigrid or multigrid-like methods have also been developed for many PDEs with nonelliptic features.

We will start with the usual classification of second-order scalar 2D PDEs. Generalizations of this classification to 3D, higher order equations or systems of PDEs can be found [150]. We consider equations $Lu = f$ in some domain $\Omega \in \mathbb{R}^2$ where

$$Lu = a_{11}u_{xx} + a_{12}u_{xy} + a_{22}u_{yy} + a_1u_x + a_2u_y + a_0u \quad (\Omega), \tag{1.1.1}$$

with coefficients a_{ij}, a_i, a_0 and a right-hand side f which, in general, may depend on x, y, u, u_x, u_y (the quasilinear case). In most parts of this book, $Lu = f$ is assumed to be a linear differential equation, which means that the coefficients and the right-hand side f only depend on (x, y). L is called

- elliptic if $4a_{11}a_{22} > a_{12}^2$,
- hyperbolic if $4a_{11}a_{22} < a_{12}^2$,
- parabolic if $4a_{11}a_{22} = a_{12}^2$.

In general, this classification depends on (x, y) and, in the nonlinear case, also on the solution u. Prototypes of the above equation are

- the Poisson equation $-\Delta u = -u_{xx} - u_{yy} = f$,
- the wave equation $u_{xx} - u_{yy} = 0$,
- the heat equation $u_{xx} - u_y = 0$.

Since multigrid methods work excellently for nicely elliptic problems, most of our presentation in the first chapters is oriented to Poisson's and Poisson-like equations. Other important model equations that we will treat in this book include

- the anisotropic model equation $-\varepsilon u_{xx} - u_{yy} = f$,
- the convection–diffusion equation $-\varepsilon\Delta u + a_1u_x + a_2u_y = f$,
- the equation with mixed derivatives $-\Delta u + \tau u_{xy} = f$.

All these equations will serve as model equations for special features and complications and are thus representative of a larger class of problems with similar features. These model equations depend crucially on a parameter ε or τ. For certain parameter values we have a singular perturbation: the type of the equation changes and the solution behaves qualitatively different (if it exists at all). For instance, the anisotropic equation becomes parabolic for $\varepsilon \to 0$, the equation with mixed derivatives is elliptic for $|\tau| < 2$, parabolic for $|\tau| = 2$ and hyperbolic for $|\tau| > 2$. All the model equations represent classes of problems which are of practical relevance.

In this book, the applicability of multigrid is connected to a quantity, the "h-ellipticity measure E_h", that we will introduce in Section 4.7. This h-ellipticity measure is not applied to the differential operator itself, but to the corresponding discrete operator. It can be used to analyze whether or not the discretization is appropriate for a straightforward multigrid treatment. Nonelliptic problems can also have some h-ellipticity if discretized accordingly.

The above model equations, except the wave and the heat conduction equations, are typically connected with pure *boundary conditions*. The wave and the heat conduction equations are typically characterized by *initial conditions* with respect to one of the variables (y) and by boundary conditions with respect to the other (x).

We will call problems which are characterized by pure boundary conditions *space-type problems*. For problems with initial conditions, we interpret the variable for which the initial condition is stated as the *time variable* t ($= y$), and call these problems *time-type*. Usually these problems exhibit a marching or evolution behavior with respect to t. Space-type equations, on the other hand, usually describe stationary situations. Note that this distinction is different from the standard classification of elliptic, hyperbolic and parabolic. Elliptic problems are usually space-type while hyperbolic and parabolic problems are often time-type. However, the stationary supersonic full potential equation, the convection equation (see Chapter 7) and the stationary Euler equations (see Section 8.9) are examples of hyperbolic equations with space-type behavior. (In certain situations, the same equation can be interpreted as space- or as time-type, and each interpretation may have its specific meaning. An example is the convection equation.)

In this book, we will present and discuss *multigrid methods for space-type problems.*

For time-type problems, there is usually the option to discretize the time variable explicitly, implicitly or in a hybrid manner (i.e. semi-implicitly). The implicit or semi-implicit discretization yields discrete *space-type* problems which have to be solved in each time step. We will briefly consider multigrid approaches for the solution of such space-type problems which arise in each time step in Section 2.8. Typically, these problems have similar but more convenient numerical properties than the corresponding stationary problems with respect to multigrid.

Remark 1.1.1 Some authors have proposed multigrid approaches which include the time direction directly. These approaches consider time to be just another "space-type" direction. Such approaches are discussed in [78, 175, 198, 199, 396] for so-called parabolic multigrid and multigrid-like methods based on "waveform relaxation". ≫

1.2 GRIDS AND DISCRETIZATION APPROACHES

In this book, we assume that the differential equations to be solved are *discretized on a suitable grid* (or, synonymously, mesh). Here, we give a rough survey (with some examples) of those types of grids which are treated systematically in this book and those which are only touched on. We also make some remarks about discretization approaches.

1.2.1 Grids

The general remarks in this section may not be so interesting to multigrid beginners. They may start with Section 1.4 on Poisson's equation and return to this section later.

Most parts of this book refer to: *Cartesian grids*, *boundary-fitted logically rectangular grids* and *block-structured boundary-fitted grids*. Figure 1.1 is an example of a Cartesian grid. For simple domains with simple boundaries, Cartesian grids are numerically convenient. We will use them for Model Problem 1 (see Section 1.3.2) and several other cases.

Figure 1.1 also gives an example of a boundary-fitted grid. Boundary-fitted grids will be used in more advanced examples in this book. A systematic introduction into boundary-fitted grids is given in [391]. We will discuss the *generation* of boundary-fitted grids in Section 10.3.

In the context of boundary-fitted grids, there are two different approaches. In the first, coordinate transformations are used to obtain simple (for example rectangular) domains, and correspondingly simple (rectangular) grids. Here the differential (and/or the discrete) equations are transformed to the new curvilinear coordinates. In the second approach, the computations are performed in the physical domain with the original (nontransformed) equations. In this book we concentrate on the second approach.

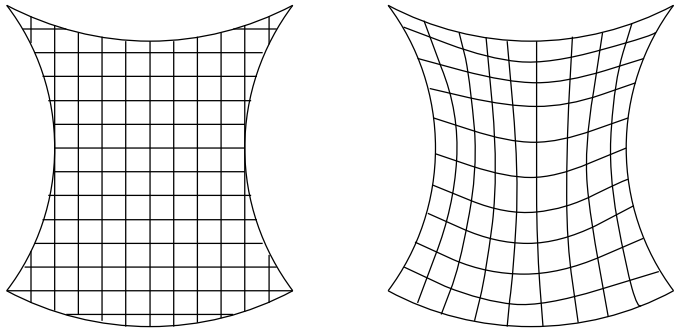


Figure 1.1. A square Cartesian grid (left) in a domain Ω and a boundary-fitted curvilinear (logically rectangular) grid (right).

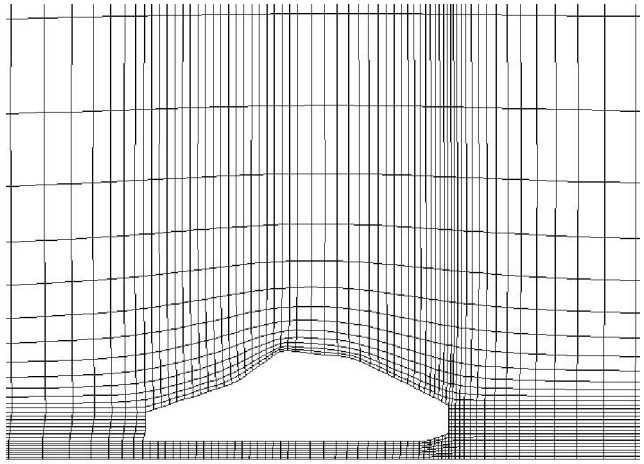


Figure 1.2. Boundary-fitted block-structured grid around a car.

Block-structured boundary-fitted grids are used if the given domain cannot (or cannot reasonably) be mapped to a rectangular domain, but can be decomposed into a finite number of subdomains each of which can be covered with a boundary-fitted grid. Quite complicated domains may be treated with this approach, as can be seen in Fig. 1.2. Block-structured boundary-fitted grids are discussed in Chapters 6 and 8–10.

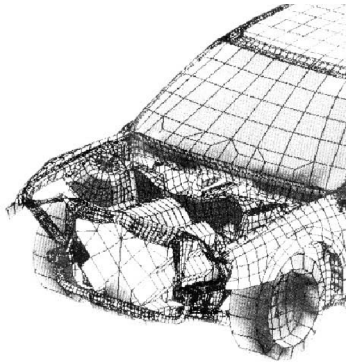


Figure 1.3. Unstructured grid around a car in a crash simulation application.

In many software packages, *unstructured, irregular grids* are used today. These grids have become quite popular because unstructured automatic mesh generation is much easier than the generation of block-structured grids for very complicated 2D and 3D domains.

An example of an unstructured grid is the grid around a car during a crash simulation. A part of this grid is presented in Fig. 1.3.

From the multigrid point of view, unstructured grids are a complication. For a given unstructured grid, it is usually not difficult to define a sequence of *finer* grids, but *it may be difficult to define a sequence of reasonable coarser grids.* (A hierarchy of coarse grids is needed for multigrid.)

The *algebraic multigrid* (AMG) method presented in Appendix A constructs a hierarchy of coarse grids automatically and is thus particularly well-suited for problems on unstructured grids.

Although unstructured grids are widely used, even complicated domains allow other than *purely* unstructured grid approaches. Often a *hybrid* approach, an unstructured grid close to the boundary and a structured grid in the interior part of the domain (or vice versa) is suitable for the treatment of such complicated domains.

More generally, all types of grids mentioned above can be used in the context of *overlapping grids.* A typical situation for the use of overlapping grids is when an overall Cartesian grid is combined with a local boundary-fitted grid (Chimera technique). An example of this approach is shown in Fig. 1.4. Such overlapping grids are also called composite grids.

Finally, we mention *self-adaptive grids.* They are constructed automatically during the solution process according to an error estimator that takes the behavior of the solution into account. Self-adaptive grids are very natural in the multigrid context. We will treat the self-adaptive multigrid approach systematically in Chapter 9. An example is given in Fig. 1.5.

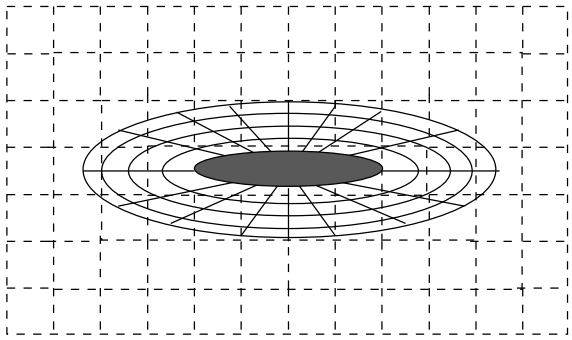


Figure 1.4. An overlapping grid around an object.

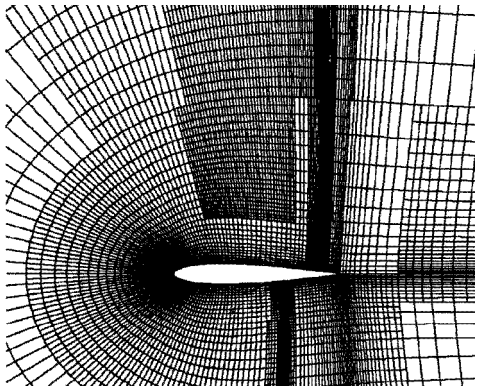


Figure 1.5. Adaptively refined grid around an airfoil.

1.2.2 Discretization Approaches

In principle, any type of grid can be used with any type of *discretization approach.* In practice, however, finite difference and finite volume methods are traditionally used in the context of Cartesian, logically rectangular and block-structured boundary-fitted grids, whereas finite elements and finite volumes are widely used in the context of unstructured grids.

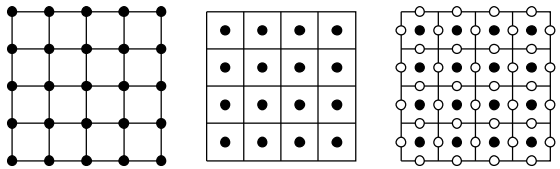


Figure 1.6. Three arrangements of unknowns in a Cartesian grid: (a) a vertex-centered grid; (b) a cell-centered grid; (c) a staggered grid.

In this book, we will focus on finite difference and finite volume discretizations on structured and block-structured grids.

Finally, another important choice to be made in discretization is the *arrangement of the unknowns within a grid*. The unknowns can be defined at the vertices of a grid (*vertex-centered location of unknowns*). Another option is the choice of the unknowns at cell centers (*cell-centered location of unknowns*).

For systems of PDEs it is possible to choose different locations for different types of unknowns. A well-known example is the *staggered grid* for the system of the incompressible Navier–Stokes equations discussed in Chapter 8, in which pressure unknowns are placed at the cell centers and velocity components at cell boundaries. Examples of a vertex-centered, a cell-centered and a staggered Cartesian grid are sketched in Fig. 1.6. Often, the discrete equations are defined at the same locations as the unknowns. It is hard to say which location of unknowns and which location of the discrete equations is best in general. Often these choices depend on the type of boundary conditions and on the application. In the following chapters we mainly present results for vertex-centered locations of unknowns. Multigrid components for cell-centered arrangements of unknowns are presented in Section 2.8.4.

1.3 SOME NOTATION

We start with some general notation. In this book, we use the classical formulation of differential equations with differential (and boundary) operators rather than a weak formulation. For discretization, we use the terminology of *discrete differential operators*. This also means that we use *grid functions* rather than vectors and *grid operators* rather than matrices. In our opinion, the grid-oriented notation emphasizes more clearly the correspondence between the discrete and continuous problems, and between the discrete formulations on different grids of the multigrid structure. In that respect, we regard it as a natural formulation for the multigrid approach. If, for example, the discrete differential operator can be described by a *difference stencil* (see Section 1.3.4) this is clearly a more condensed formulation than a matrix notation. On the other hand, there are situations in which the matrix notation has some advantages and is more general. Then we will not hesitate to use it.

1.3.1 Continuous Boundary Value Problems

We denote scalar linear boundary value problems by

$$\begin{aligned} L^{\Omega}u(\boldsymbol{x}) &= f^{\Omega}(\boldsymbol{x}) \quad (\boldsymbol{x} \in \Omega) \\ L^{\Gamma}u(\boldsymbol{x}) &= f^{\Gamma}(\boldsymbol{x}) \quad (\boldsymbol{x} \in \Gamma := \partial\Omega). \end{aligned} \tag{1.3.1}$$

Here $\boldsymbol{x} = (x_1, \ldots, x_d)^T$ and $\Omega \subset \mathbb{R}^d$ is a given open bounded domain with boundary Γ. L^{Ω} is a linear (elliptic) differential operator on Ω and L^{Γ} represents one or several linear boundary operators. f^{Ω} denotes a given function on Ω and f^{Γ} one or several functions on Γ. We always denote solutions of (1.3.1) by $u = u(\boldsymbol{x})$. For brevity, we also simply write $Lu = f$ instead of (1.3.1). Most concrete considerations refer to the cases $d = 2$ and $d = 3$. (Multigrid is usually not needed for $d = 1$, in which case it degenerates to direct or other simple well-known solvers, see Section 6.4.1.)

In the case $d = 2$ or $d = 3$, we will usually write (x, y) instead of (x_1, x_2) and (x, y, z) instead of (x_1, x_2, x_3).

This and the following chapters essentially refer to scalar equations. Chapter 8 and other parts of the book, however, will refer to systems of PDEs. The above notation is also used in that case. $\boldsymbol{L}^{\Omega}$ then stands for a vector of differential operators and $\boldsymbol{u}$, $\boldsymbol{f}$ etc. are vector-valued functions.

1.3.2 Discrete Boundary Value Problems

The following considerations are formulated for the 2D case. They can, of course, be directly generalized to higher dimensions. The discrete analog of (1.3.1) is denoted by

$$\begin{aligned} L_h^{\Omega} u_h(x, y) &= f_h^{\Omega}(x, y) \quad ((x, y) \in \Omega_h) \\ L_h^{\Gamma} u_h(x, y) &= f_h^{\Gamma}(x, y) \quad ((x, y) \in \Gamma_h). \end{aligned} \tag{1.3.2}$$

Here h is a (formal) discretization parameter. Using the infinite grid

$$\boxed{\mathbf{G}_h := \{(x, y) \colon x = x_i = i\,h_x,\ y = y_j = j\,h_y;\ i, j \in \mathbb{Z}\}} \tag{1.3.3}$$

where $\boldsymbol{h} = (h_x, h_y)$ is a vector of fixed mesh sizes, we define $\Omega_h = \Omega \cap \mathbf{G}_h$ and Γ_h as the set of discrete intersection points of the "grid lines" with Γ. In the special case of square grids, we simply identify $h = h_x = h_y$.

The discrete solution u_h is a function defined on the grid $\Omega_h \cup \Gamma_h$, i.e., a grid function, and f_h^{Ω} and f_h^{Γ} are discrete analogs of f^{Ω} and f^{Γ}, respectively, where f^{Ω}, f^{Γ} are restricted to the grid. Instead of $u_h(x, y) = u_h(x_i, y_j) = u_h(i\,h_x, j\,h_y)$, we sometimes simply write $u_{i,j}$.

L_h^{Ω} and L_h^{Γ} are grid operators, i.e., mappings between spaces of grid functions. (L_h^{Ω} is also called a *discrete* (differential) or *difference operator*, L_h^{Γ} a *discrete boundary operator*.)

Clearly the concrete definitions of Ω_h, Γ_h etc. also depend on the given PDE, the domain Ω, the boundary conditions, the grid approach and the discretization.

For ease of presentation, we will first assume that discrete boundary equations are eliminated from (1.3.2). (In general, a proper multigrid treatment of boundary conditions

needs an explicit consideration of *noneliminated* boundary conditions. This is discussed in more detail in Section 5.6.)

In the case of second-order equations with Dirichlet boundary conditions, which means $u_h = f_h^{\Gamma}, (x, y) \in \Gamma_h$, eliminated boundary conditions can be considered as a natural approach. We then simply write

$$L_h u_h = f_h \quad (\Omega_h). \tag{1.3.4}$$

Here u_h and f_h are grid functions on Ω_h and L_h is a linear operator

$$L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h), \tag{1.3.5}$$

where $\mathcal{G}(\Omega_h)$ denotes the *linear space of grid functions on* Ω_h. Clearly, (1.3.4) can be represented as a system of linear algebraic equations. However, we usually consider it as one *grid equation.*

Even if the boundary conditions are not eliminated, we may use the notation (1.3.4) for the discrete problem. The notation then represents an abstract equation (in a finite dimensional space).

1.3.3 Inner Products and Norms

For convergence considerations and many other purposes, we need to use norms in the finite dimensional space $\mathcal{G}(\Omega_h)$. Most of our considerations (and many of our measurements) will be based on the *Euclidean inner product*

$$\langle u_h, v_h \rangle_2 := \frac{1}{\#\Omega_h} \sum_{\boldsymbol{x} \in \Omega_h} u_h(\boldsymbol{x}) \overline{v_h(\boldsymbol{x})}, \tag{1.3.6}$$

where $\#\Omega_h$ is the number of grid points of Ω_h. The scaling factor $(\#\Omega_h)^{-1}$ allows us to compare grid functions on different grids and also the corresponding continuous functions on Ω. The induced norm is $||u_h||_2 = \sqrt{\langle u_h, u_h \rangle_2}$. The corresponding operator norm for discrete operators L_h on $\mathcal{G}(\Omega_h)$ is the *spectral* norm

$$\|B_h\|_S = \sqrt{\rho(B_h B_h^*)},$$

where B_h denotes any linear operator $B_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h)$, and where ρ is the spectral radius.

For L_h, symmetric and positive definite, we also consider the *energy inner product*

$$\langle u_h, v_h \rangle_E := \langle L_h u_h, v_h \rangle_2 \tag{1.3.7}$$

and the corresponding operator norm $|| \cdot ||_E$, which is given by

$$||B_h||_E = ||{L_h}^{1/2} B_h {L_h}^{-1/2}||_S = \sqrt{\rho(L_h B_h {L_h}^{-1} B_h^*)}. \tag{1.3.8}$$

For practical purposes the *infinity norm*

$$||u_h||_\infty := \max\{|u_h(\boldsymbol{x})|: \boldsymbol{x} \in \Omega_h\} \tag{1.3.9}$$

and the corresponding operator norm $|| \cdot ||_\infty$ are commonly used.

1.3.4 Stencil Notation

For the concrete definition of discrete operators L_h^Ω (on a Cartesian or a logically rectangular grid) the stencil terminology is convenient. We restrict ourselves to the case $d = 2$. The extension to $d = 3$ (and to general d) is straightforward (see Section 2.9). We first introduce the stencil notation for the infinite grid $\mathbf{G}_h$ (defined in Section 1.3.2). On $\mathbf{G}_h$, we consider grid functions

$$\begin{aligned} w_h : \mathbf{G}_h &\longrightarrow \mathbb{R} \quad (\text{or } \mathbb{C}) \\ (x, y) &\longmapsto w_h(x, y). \end{aligned}$$

A general stencil $[s_{\kappa_1\kappa_2}]_h$

$$[s_{\kappa_1\kappa_2}]_h = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & s_{-1,1} & s_{0,1} & s_{1,1} & \cdots \\ \cdots & s_{-1,0} & s_{0,0} & s_{1,0} & \cdots \\ \cdots & s_{-1,-1} & s_{0,-1} & s_{1,-1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}_h \qquad (s_{\kappa_1\kappa_2} \in \mathbb{R})$$

defines an operator on the set of grid functions by

$$\boxed{[s_{\kappa_1\kappa_2}]_h w_h(x, y) = \sum_{(\kappa_1, \kappa_2)} s_{\kappa_1\kappa_2} w_h(x + \kappa_1 h_x, y + \kappa_2 h_y)}. \tag{1.3.10}$$

Here we assume that only a *finite number of coefficients* $s_{\kappa_1\kappa_2}$ *are nonzero.*

Many of the stencils we will consider will be *five-point* or *compact nine-point* stencils

$$\begin{bmatrix} & s_{0,1} & \\ s_{-1,0} & s_{0,0} & s_{1,0} \\ & s_{0,-1} & \end{bmatrix}_h, \qquad \begin{bmatrix} s_{-1,1} & s_{0,1} & s_{1,1} \\ s_{-1,0} & s_{0,0} & s_{1,0} \\ s_{-1,-1} & s_{0,-1} & s_{1,-1} \end{bmatrix}_h. \tag{1.3.11}$$

The discrete operators L_h^Ω are usually given only on a finite grid Ω_h. For the identification of a discrete operator L_h^Ω with "its" stencil $[s_{\kappa_1\kappa_2}]_h$, we usually have to restrict the stencil to Ω_h (instead of $\mathbf{G}_h$). Near boundary points the stencils may have to be modified. In square or rectangular domains Ω, which are the basis for the examples in Chapter 2, this modification of $[s_{\kappa_1\kappa_2}]_h$ to Ω_h is straightforward (see, e.g., Example 1.4.1 below). Let us finally mention that the coefficients $s_{\kappa_1\kappa_2}$ will depend on (x, y):

$$s_{\kappa_1\kappa_2} = s_{\kappa_1\kappa_2}(x, y)$$

if the coefficients of L^Ω and/or L_h^Ω depend on (x, y).

1.4 POISSON'S EQUATION AND MODEL PROBLEM 1

In this section, we introduce Model Problem 1, the classical model for a discrete elliptic boundary value problem. Every numerical solver has been applied to this problem for comparison.

Model Problem 1 *At many places in this book, we will study in detail the discrete Poisson equation with Dirichlet boundary conditions*

$$\begin{aligned} -\Delta_h u_h(x,y) &= f_h^{\Omega}(x,y) \quad ((x,y)\in\Omega_h) \\ u_h(x,y) &= f_h^{\Gamma}(x,y) \quad ((x,y)\in\Gamma_h=\partial\Omega_h) \end{aligned} \tag{1.4.1}$$

in the unit square $\Omega=(0,1)^2\subset\mathbb{R}^2$ *with* $h=1/n$, $n\in\mathbb{N}$. *Here,* $L_h=-\Delta_h$ *is the standard five-point* $O(h^2)$ *approximation (explained below) of the partial differential operator* L,

$$Lu=-\Delta u=-u_{xx}-u_{yy} \tag{1.4.2}$$

on the square grid $\mathbf{G}_h$.

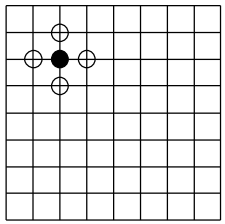
$O(h^2)$ here means that one can derive consistency relations of the form

$$Lu-L_hu=O(h^2) \quad \text{for } h\to 0$$

for sufficiently smooth functions u ($u\in C_4(\bar{\Omega})$, for example).

To illustrate exactly what the elimination of boundary conditions means, we consider the following example.

Example 1.4.1 The discrete Poisson equation with eliminated Dirichlet boundary conditions can formally be written in the form (1.3.4). For $(x,y)\in\Omega_h$ not adjacent to a boundary this means:



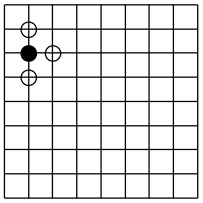
$$\begin{aligned} f_h(x,y) &= f^{\Omega}(x,y) \\ L_hu_h(x,y) &= -\Delta_h u_h(x,y) \\ &= \frac{1}{h^2}[4u_h(x,y)-u_h(x-h,y)-u_h(x+h,y) \\ &\quad -u_h(x,y-h)-u_h(x,y+h)] \\ &= \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h u_h(x,y) \end{aligned}$$

The notation

$$-\Delta_h=\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_h$$

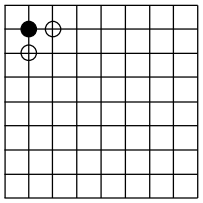
is a first example of the stencil notation (1.3.11).

For $(x,y)\in\Omega_h$ adjacent to a (here: west) boundary, L_h (1.3.4) reads



$$\begin{aligned} f_h(x,y) &= f^{\Omega}(x,y)+\frac{1}{h^2}f^{\Gamma}(x-h,y) \\ L_hu_h(x,y) &= \frac{1}{h^2}[4u_h(x,y)-u_h(x+h,y) \\ &\quad -u_h(x,y-h)-u_h(x,y+h)] \\ &= \frac{1}{h^2}\begin{bmatrix} & -1 & \\ 0 & 4 & -1 \\ & -1 & \end{bmatrix}_h u_h(x,y). \end{aligned}$$

For $(x,y)\in\Omega_h$ in a (here: the north-west) corner we have



$$\begin{aligned} f_h(x,y) &= f^{\Omega}(x,y)+\frac{1}{h^2}[f^{\Gamma}(x-h,y)+f^{\Gamma}(x,y+h)] \\ L_hu_h(x,y) &= \frac{1}{h^2}[4u_h(x,y)-u_h(x+h,y)-u_h(x,y-h)] \\ &= \frac{1}{h^2}\begin{bmatrix} & 0 & \\ 0 & 4 & -1 \\ & -1 & \end{bmatrix}_h u_h(x,y). \end{aligned}$$ △

In this example, elimination of the boundary conditions is simple. Often, elimination of boundary conditions may be complicated and is not preferred. In such cases, a particular treatment of the boundary conditions in the multigrid process may be needed (see Section 5.6).

1.4.1 Matrix Terminology

Discrete operators L_h are often represented by *matrices* A_h. Each matrix row then represents connections of one unknown in the discretization of a PDE to its neighbor unknowns. Which of the matrix entries is different from 0, depends on the ordering of grid points, i.e., the ordering of the components of the vector of unknowns.

As an example we consider Model Problem 1. For a column- or row-wise ordering of grid points (also called *lexicographical* ordering, see Fig. 1.7(a), starting with points at the left lower corner) and eliminated Dirichlet boundary conditions, the resulting matrix is a block tridiagonal matrix with a regular sparsity pattern:

$$A_h=\frac{1}{h^2}\begin{pmatrix} T & -I & & \\ -I & T & -I & \\ & -I & T & -I \\ & & -I & T \end{pmatrix}, \tag{1.4.3}$$

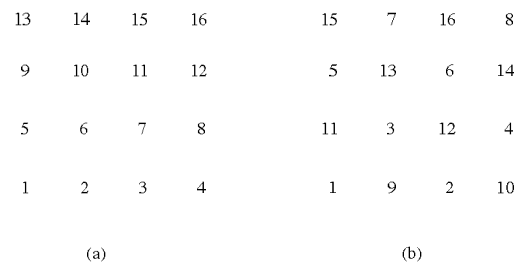


Figure 1.7. (a) Lexicographic; (b) red–black ordering of grid points.

where

$$I = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}.$$

Due to the elimination of Dirichlet boundary points, every matrix row corresponding to a grid point near a boundary has only three or two entries connecting them to neighbor grid points. This can be seen, for example, in the first rows of the matrix, where only two or three -1 entries can be found instead of four for interior points.

The dependence of the matrix structure on the ordering of the grid points can be seen when we write the matrix down for a *red–black ordering* of the grid points (see Fig. 1.7(b)). First all unknowns at odd (red) grid points are considered, then the unknowns at the even (black) points. The corresponding matrix A_h is now a block matrix with blocks A_{rr}, representing the connections of the red grid points to red grid points, A_{rb} the connections of the red points to the black points, A_{br} the connections of black points to red points, and A_{bb} the connections of black points to black points. So:

$$A_h = \begin{bmatrix} A_{rr} & A_{rb} \\ A_{br} & A_{bb} \end{bmatrix}. \tag{1.4.4}$$

For Model Problem 1, the blocks A_{rr} and A_{bb} are diagonal matrices with $4/h^2$ as the diagonal elements. The resulting block matrix A_{rb} $(= A_{br}^T)$ of the above example

in Fig. 1.7(b) is

$$A_{rb} = \frac{1}{h^2} \begin{pmatrix} -1 & 0 & -1 & & & & & \\ -1 & -1 & 0 & -1 & & & & \\ -1 & 0 & -1 & -1 & -1 & & & \\ & -1 & 0 & -1 & 0 & -1 & & \\ & & -1 & 0 & -1 & 0 & -1 & \\ & & & -1 & -1 & -1 & 0 & -1 \\ & & & & -1 & 0 & -1 & -1 \\ & & & & & -1 & 0 & -1 \end{pmatrix}. \tag{1.4.5}$$

Remark 1.4.1 As we will see below (Section 1.5), in a multigrid algorithm it is usually not necessary to build up the matrix A_h coming from the discretization. The multigrid components are based on "local" operations; multiplications and additions are carried out grid point by grid point. The storage that is needed in a multigrid code mainly consists of solution vectors, defects and right-hand sides on all grid levels. ≫

1.4.2 Poisson Solvers

Table 1.1 gives an overview on the complexity of different solution methods (including fast Poisson solvers) applied to Model Problem 1. Here direct and iterative solvers are listed. For the iterative solvers, we assume an accuracy (stopping criterion) in the range of the discretization accuracy. This is reflected by the $\log \varepsilon$ term. The full multigrid (FMG) variant of multigrid which we will introduce in Section 2.6 is a solver up to discretization accuracy.

It is generally expected that the more general a solution method is, the less efficient it is and vice versa. Multigrid is, however, a counter example for this pattern—indeed multigrid

Table 1.1. Complexity of different solvers for the 2D Poisson problem (N denotes the total number of unknowns).

Method	# operations in 2D
Gaussian elimination (band version)	$O(N^2)$
Jacobi iteration	$O(N^2 \log \varepsilon)$
Gauss–Seidel iteration	$O(N^2 \log \varepsilon)$
Successive overrelaxation (SOR) [431]	$O(N^{3/2} \log \varepsilon)$
Conjugate gradient (CG) [194]	$O(N^{3/2} \log \varepsilon)$
Nested dissection (see, for example, [9])	$O(N^{3/2})$
ICCG [264]	$O(N^{5/4} \log \varepsilon)$
ADI (see, for example, [403])	$O(N \log N \log \varepsilon)$
Fast Fourier transform (FFT) [112]	$O(N \log N)$
Buneman [93]	$O(N \log N)$
Total reduction [342]	$O(N)$
Multigrid (iterative)	$O(N \log \varepsilon)$
Multigrid (FMG)	$O(N)$

will turn out to be a general principle. On the other hand, Table 1.1 shows that the most efficient version of multigrid yields an algorithm that is at least as efficient as the highly efficient and highly specialized fast Poisson solvers. Here, the total reduction method and the Buneman algorithm are typical fast Poisson solvers, very efficient but essentially designed and tailored exclusively for elliptic *model* problems.

1.5 A FIRST GLANCE AT MULTIGRID

1.5.1 The Two Ingredients of Multigrid

In this section, we introduce the multigrid idea heuristically for Model Problem 1. We use the grid function oriented notation introduced in Section 1.3.2.

The iteration formula of the classical *lexicographical* Gauss–Seidel method (GS-LEX) for Poisson's equation reads

$$u_h^{m+1}(x_i, y_j) = \tfrac{1}{4}[h^2 f_h(x_i, y_j) + u_h^{m+1}(x_i - h, y_j) + u_h^m(x_i + h, y_j) + u_h^{m+1}(x_i, y_j - h) + u_h^m(x_i, y_j + h)], \tag{1.5.1}$$

where $(x_i, y_j) \in \Omega_h$. Here u_h^m and u_h^{m+1} are the approximations of $u_h(x_i, y_j)$ before and after an iteration, respectively.

If we apply (1.5.1) to Poisson's equation, we recognize the following phenomenon. After a few iteration steps, the *error* of the approximation becomes *smooth*. It doesn't necessarily become small, but it does become smooth. See Fig. 1.8 for an illustration of this error smoothing effect. Looking at the error

$$v_h^m(x_i, y_j) = u_h(x_i, y_j) - u_h^m(x_i, y_j),$$

Formula (1.5.1) means

$$v_h^{m+1}(x_i, y_j) = \tfrac{1}{4}[v_h^{m+1}(x_i - h, y_j) + v_h^m(x_i + h, y_j) + v_h^{m+1}(x_i, y_j - h) + v_h^m(x_i, y_j + h)]. \tag{1.5.2}$$

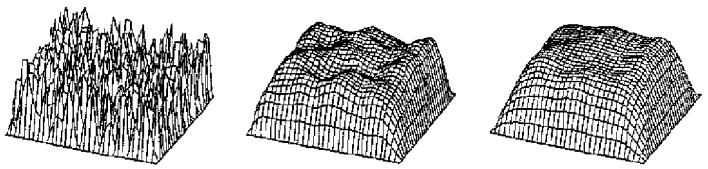


Figure 1.8. Influence of lexicographic Gauss–Seidel iteration (1.5.1) on the error (Model Problem 1).

Obviously, the iteration formula can be interpreted as an error averaging process. Error smoothing is one of the two basic principles of the multigrid approach.

> **Smoothing principle** Many classical iterative methods (Gauss–Seidel etc.) if appropriately applied to discrete elliptic problems have a strong smoothing effect on the error of any approximation.

The other basic principle is the following: a quantity that is smooth on a certain grid can, without any essential loss of information, also be approximated on a coarser grid, say a grid with double the mesh size. In other words: if we are sure that the error of our approximation has become smooth after some iteration steps, we may approximate this error by a suitable procedure on a (much) coarser grid.

Qualitatively, this is the coarse grid approximation principle.

> **Coarse grid principle** A smooth error term is well approximated on a coarse grid. A coarse grid procedure is substantially less expensive (substantially fewer grid points) than a fine grid procedure.

As this principle holds for error or "correction" quantities, we also speak of the *coarse grid correction* (CGC) principle.

Let us illustrate these considerations and explain them heuristically by looking at the Fourier expansion of the error. In our model problem the error $v_h = v_h^m(x, y)$ considered as a function of the discrete variables x and y can be written as

$$v_h(x, y) = \sum_{k,\ell=1}^{n-1} \alpha_{k,\ell} \sin k\pi x \sin \ell\pi y. \tag{1.5.3}$$

For $(x, y) \in \Omega_h$, the functions

$$\varphi_h^{k,\ell}(x, y) = \sin k\pi x \sin \ell\pi y \quad (k, \ell = 1, \ldots, n-1) \tag{1.5.4}$$

are the discrete eigenfunctions of the discrete operator Δ_h. The fact that this error becomes smooth after some iteration steps means that the *high frequency components* in (1.5.3), i.e.

$$\alpha_{k,\ell} \sin k\pi x \sin \ell\pi y \quad \text{with } k \text{ or } l \text{ large} \tag{1.5.5}$$

become small after a few iterations whereas the *low frequency components*, i.e.

$$\alpha_{k,\ell} \sin k\pi x \sin \ell\pi y \quad \text{with } k \text{ and } l \text{ small} \tag{1.5.6}$$

hardly change. The distinction between high and low frequencies is important in the multigrid context. In the following subsection, we will give a first definition of high and low frequencies and show how this concept is related to the *coarse* grid under consideration.

1.5.2 High and Low Frequencies, and Coarse Meshes

We again consider Model Problem 1 on a grid Ω_h with mesh size $h = 1/n$. Additionally, we consider Model Problem 1 on a coarser grid Ω_H with mesh size $H > h$. Assuming that n is an even number, we may, for instance, choose $H = 2h$, which is a very natural choice in the multigrid context. This choice of the coarse grid is, therefore, called *standard coarsening*.

For the definition of the high and low frequencies, we return to the eigenfunctions $\varphi^{k,\ell} = \varphi_h^{k,\ell}$ in (1.5.4). For given (k, ℓ), we consider the (four) eigenfunctions

$$\varphi^{k,\ell}, \quad \varphi^{n-k,n-\ell}, \quad \varphi^{n-k,\ell}, \quad \varphi^{k,n-\ell}$$

and observe that they coincide on Ω_{2h} in the following sense:

$$\varphi^{k,\ell}(x,y) = -\varphi^{n-k,\ell}(x,y) = -\varphi^{k,n-\ell}(x,y) = \varphi^{n-k,n-\ell}(x,y) \quad \text{for } (x,y) \in \Omega_{2h}.$$

This means that these four eigenfunctions cannot be distinguished on Ω_{2h}. (For k or $l = n/2$, the $\varphi^{k,\ell}$ vanish on Ω_{2h}.) This gives rise to the following definition of low and high frequencies:

Definition (in the context of Model Problem 1) For $k, \ell \in \{1, \ldots, n-1\}$, we denote $\varphi^{k,\ell}$ to be an eigenfunction (or a *component*) of

$$\begin{aligned} &\text{low frequency} && \text{if } \max(k,\ell) < n/2, \\ &\text{high frequency} && \text{if } n/2 \le \max(k,\ell) < n. \end{aligned}$$

Obviously, only the low frequencies *are visible* on Ω_{2h} since all high frequencies coincide with a low frequency on Ω_{2h} (or vanish on Ω_{2h}). The fact that high frequencies coincide with low ones is also called *aliasing* of frequencies. For the 1D case with $n = 8$, we illustrate the above definition in Fig. 1.9. We summarize this consideration:

The low frequency components also represent meaningful grid functions on a grid Ω_{2h} with double the mesh size, whereas the high frequency components do not. Their high frequencies are not "visible" on the Ω_{2h} grid.

If we apply this distinction of low and high frequencies to the representation of $v_h(x, y)$ in (1.5.3), we can decompose the sum in (1.5.3) into corresponding partial sums:

$$\sum_{k,\ell=1}^{n-1} \alpha_{k,\ell}\varphi^{k,\ell} = \sum\nolimits^{\text{high}} \alpha_{k,\ell}\varphi^{k,\ell} + \sum\nolimits^{\text{low}} \alpha_{k,\ell}\varphi^{k,\ell}$$

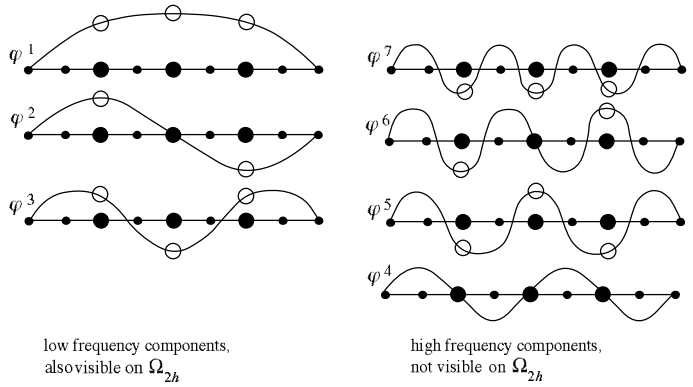


Figure 1.9. Low and high frequency components for a 1D example ($n = 8$).

where

$$\sum\nolimits^{\text{low}} \alpha_{k,\ell}\varphi^{k,\ell} = \sum_{k,\ell=1}^{n/2-1} \alpha_{k,\ell}\varphi^{k,\ell}$$

and

$$\sum\nolimits^{\text{high}} \alpha_{k,\ell}\varphi^{k,\ell} = \sum_{\substack{k,\ell \\ n/2 \le \max(k,\ell)}}^{n-1} \alpha_{k,\ell}\varphi^{k,\ell}.$$

Remark 1.5.1 ($H = 4h$ and other choices of H) From our definition, it immediately becomes clear that the terms "high frequency" and "low frequency" are related to both the fine grid Ω_h *and* the coarse grid Ω_H that we consider. (If we want to emphasize this dependence on the grids Ω_h and Ω_H, we will speak of (h, H)-low and (h, H)-high frequencies.) If, for example, we would choose

$$H = 4h$$

(assuming that n is a multiple of 4), our definition of high and low frequencies would have to be modified in the following way:

$$\begin{aligned} &\varphi^{k,l} \text{ is a } (h,4h)\text{-low frequency component} && \text{if } \max(k,l) < n/4. \\ &\varphi^{k,l} \text{ is a } (h,4h)\text{-high frequency component} && \text{otherwise.} \end{aligned}$$

The choice $H = 4h$ is not very practical in the multigrid context since it usually does not lead to the most efficient algorithms, but it is not out of the question, either.

Other, more practical choices of coarse grids, different from standard coarsening, are introduced in Section 2.3.1. ≫

1.5.3 From Two Grids to Multigrid

In the following, we will continue our heuristic considerations a little further extending them from two-grid levels Ω_h, Ω_{2h} to a sequence of levels. In this setting, we will also be able to explain the *multigrid* (not only the two-grid) idea. Fig. 1.10 shows such a sequence of grids.

We assume additionally that n is a power of 2 meaning that $h = 2^{-p}$. Then we can form the grid sequence

$$\Omega_h, \Omega_{2h}, \Omega_{4h}, \ldots, \Omega_{h_0} \tag{1.5.7}$$

just by doubling the mesh size successively. We assume that this sequence ends with a coarsest grid Ω_{h_0} (which may well be the grid consisting of only one interior grid point $(x, y) = (1/2, 1/2)$, i.e. $h_0 = 1/2$, see Fig. 1.10).

We now consider a decomposition of the error representation (1.5.3) into partial sums which correspond to the grid sequence (1.5.7), having the following idea in mind. In the same way as we have distinguished low and high frequencies with respect to the pair of grids Ω_h and Ω_{2h} in the previous section, we now make an additional distinction between high and low frequencies with respect to the pair Ω_{2h} and Ω_{4h}. We continue with the pair Ω_{4h} and Ω_{8h} and so on.

As discussed above, by using Gauss–Seidel type iterations on the original Ω_h grid, we cause the $(h, 2h)$-high frequency error components to become small rapidly. The remaining low frequency error components are visible and can thus be approximated on Ω_{2h}. (Of course, an equation which determines the low frequency error has to be defined in a suitable way on Ω_{2h}.) Performing Gauss–Seidel-type iteration not only on the original Ω_h grid, but also on Ω_{2h} and correspondingly on Ω_{4h} etc., causes the $(2h, 4h)$-high frequency, the $(4h, 8h)$-high frequency components etc. to decrease rapidly. Only on the coarsest grid might it be necessary to do something special, which is trivial if Ω_{h_0} consists of only one (or few) point(s). Altogether, this leads to a very fast overall reduction of the error.

Summary What we have described and explained so far is the basic idea of multigrid. Our description is not at all an algorithm. It is not even a clear verbal description of an algorithmic principle. For example, we have neither said precisely what a Gauss–Seidel iteration on Ω_{2h}, Ω_{4h} etc. means algorithmically and to which grid functions it is to be applied, nor how to go from one level to the other etc. However, the flavor of multigrid is in it.

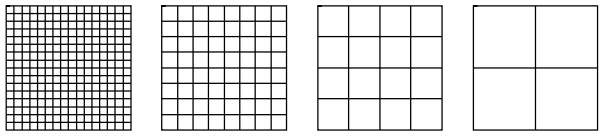


Figure 1.10. A sequence of coarse grids starting with $h = 1/16$.

Flavor of multigrid Gauss–Seidel iteration (or more generally, an appropriate iterative scheme) on different grid levels gives rapid reduction of the corresponding high frequency components and as this process covers all frequencies, a rapid reduction of the overall error can be achieved.

1.5.4 Multigrid Features

The multigrid idea is very fundamental and can also be applied in other contexts. Accordingly, there are also several different ways to view multigrid. In order to clarify our understanding of multigrid, we want to briefly discuss different multigrid aspects and to point out those multigrid features which we regard as the most significant ones.

Multigrid as an iterative linear solver The most basic way to view multigrid is to consider it as an iterative linear solver for a discrete elliptic boundary value problem. Here we assume the problem, the grid and the discretization to be given and fixed. A characteristic feature of the iterative multigrid approach is that the multigrid convergence speed is independent of the discretization mesh size h and that the number of arithmetic operations per iteration step is proportional to the number of grid points. The multigrid principle allows us to construct very efficient linear solvers, and, in that respect, the iterative approach is important and fundamental. This approach is described in detail in Chapter 2 of this book. On the other hand, this view is somewhat restricted and does not exploit the full potential of the multigrid idea. For example, even the direct application of multigrid to *nonlinear problems* (discussed in detail in Section 5.3) is not covered by this approach.

Multigrid as a solver for the differential problem From a more sophisticated point of view, multigrid can be regarded as a solution method for the (continuous) differential problem. In this view it is not appropriate to separate the discretization and the solution of the discrete problem, but rather to regard both processes as interdependent: the solution process can, according to this view, be performed the more efficiently the more the continuous differential background is exploited.

One simple, but very natural way of looking at multigrid as a "differential solver" is represented by the *full multigrid method* (FMG, see Section 2.6). Here, the method is oriented to minimizing the differential error rather than to minimizing the algebraic error (corresponding to the linear system). Self-adaptive grid refinements and related approaches, which are very natural in the multigrid context, also belong to the "differential view" of multigrid (and will be presented in Chapter 9).

Efficiency Multigrid methods are highly efficient. Efficiency here relates to both a theoretical feature and a practical one. The theoretical efficiency is more precisely expressed by the term "optimality" in a complexity theory sense. If interpreted appropriately, (full) multigrid methods are typically optimal in the sense that the number of arithmetic operations needed to solve a (discrete) problem is proportional to the number N of unknowns in the problem considered.

2

BASIC MULTIGRID I

The multigrid idea is based on two principles: error smoothing and coarse grid correction. In this chapter, we will explain how these principles are combined to form a multigrid algorithm. Basic multigrid will be described systematically.

In Section 2.1, we discuss the smoothing properties of classical iterative solvers. Sections 2.2, 2.4 and 2.6 give a systematic introduction to two-grid iteration, multigrid iteration and the full multigrid method, respectively. Some standard multigrid components are described in Section 2.3.

We prefer a presentation of the two-grid cycle in Section 2.2, which starts with the idea of an approximate solution of the defect equation and then brings together smoothing and coarse grid correction. The methods described in Sections 2.2–2.4 and 2.6 are general, although all concrete examples refer to Poisson's equation. Concrete fast multigrid Poisson solvers for the 2D and 3D cases are presented in Sections 2.5 and 2.9, respectively.

Some straightforward generalizations of the 2D method are discussed in Section 2.8. In Section 2.7, we resume the discussion on transfer operators and focus on some practical aspects.

2.1 ERROR SMOOTHING PROCEDURES

We have observed in Section 1.5 for Model Problem 1 that the usual Gauss–Seidel iteration has a remarkable smoothing effect on the error v_h^m of an approximation u_h^m. As this property is fundamental for the multigrid idea, we discuss smoothing procedures in more detail here.

We will, in particular, consider two classical iteration methods: Gauss–Seidel-type and Jacobi-type iterations. We will see that these methods are suitable for error smoothing. The smoothing properties will, however, turn out to be dependent on the right choice of *relaxation parameters* and, in the case of the Gauss–Seidel iteration, also on the *ordering of grid points*.

All iterative methods which we discuss in this chapter, are *pointwise* iterations, line- or block-type iterations are *not* yet considered here. We start our discussion with Jacobi-type iterations since the analysis is particularly easy and illustrative.

In general, however, appropriate Gauss–Seidel-type iterations turn out to be better smoothers than appropriate Jacobi-type iterations.

In the following we will speak of Jacobi- and Gauss–Seidel-type *relaxation* methods rather than *iteration* methods.

Relaxation methods Classical iteration methods such as Gauss–Seidel-type and Jacobi-type iterations are often called *relaxation* methods (or smoothing methods or smoothers) if they are used for the purpose of error smoothing.

2.1.1 Jacobi-type Iteration (Relaxation)

For Model Problem 1, the iteration formula of the Jacobi iteration reads

$$\begin{aligned} z_h^{m+1}(x_i, y_j) &= \tfrac{1}{4}\big[h^2 f_h(x_i, y_j) + u_h^m(x_i - h, y_j) + u_h^m(x_i + h, y_j) \\ &\qquad + u_h^m(x_i, y_j - h) + u_h^m(x_i, y_j + h)\big] \\ u_h^{m+1} &= z_h^{m+1}, \end{aligned} \tag{2.1.1}$$

(with $(x_i, y_j) \in \Omega_h$), where u_h^m denotes the old approximation and u_h^{m+1} the new approximation of the iteration. The Jacobi iteration can be written as

$$u_h^{m+1} = S_h u_h^m + \frac{h^2}{4} f_h$$

with the iteration operator

$$S_h = I_h - \frac{h^2}{4} L_h$$

(where I_h is the identity operator). We can generalize this iteration by introducing a relaxation parameter ω

$$u_h^{m+1} = u_h^m + \omega(z_h^{m+1} - u_h^m), \tag{2.1.2}$$

which is called the ω-(damped) Jacobi relaxation (ω-JAC). Obviously, ω-JAC and Jacobi iteration coincide for $\omega = 1$. The iteration operator for ω-JAC reads

$$S_h(\omega) = I_h - \frac{\omega h^2}{4} L_h = \frac{\omega}{4} \begin{bmatrix} & 1 & \\ 1 & 4(1/\omega - 1) & 1 \\ & 1 & \end{bmatrix}_h . \tag{2.1.3}$$

The *convergence properties* of ω-JAC can be easily analyzed by considering the eigenfunctions of S_h, which are the same as those of L_h, namely

$$\varphi_h^{k,\ell}(x) = \sin k\pi\, x \sin \ell\pi y \quad ((x,y) \in \Omega_h;\ (k,\ell = 1,\dots,n-1)). \tag{2.1.4}$$

The corresponding eigenvalues of S_h are

$$\chi_h^{k,\ell} = \chi_h^{k,\ell}(\omega) = 1 - \frac{\omega}{2}(2 - \cos k\pi h - \cos \ell\pi h). \tag{2.1.5}$$

For the spectral radius $\rho(S_h) = \max\{|\chi_h^{k,\ell}|\colon (k,\ell = 1,\dots,n-1)\}$, we obtain

$$\begin{aligned} &\text{for } 0 < \omega \le 1\colon\ \rho(S_h) = |\chi_h^{1,1}| = |1 - \omega(1 - \cos\pi h)| = 1 - O(\omega h^2) \\ &\qquad \text{else}\colon\ \rho(S_h) \ge 1 \quad (\text{for } h \text{ small enough})\,. \end{aligned} \tag{2.1.6}$$

In particular, when regarding the (unsatisfactory) asymptotic convergence, there is no use in introducing the relaxation parameter: $\omega = 1$ is the best choice.

2.1.2 Smoothing Properties of ω-Jacobi Relaxation

The situation is different with respect to the *smoothing properties* of ω-Jacobi relaxation. In order to achieve reasonable smoothing, we have to introduce a parameter $\omega \neq 1$.

For $0 < \omega \le 1$, we first observe from (2.1.6) that the smoothest eigenfunction $\varphi_h^{1,1}$ is responsible for the slow convergence of Jacobi's method. Highly oscillating eigenfunctions are reduced much faster *if ω is chosen properly*. To see this, we consider the approximations before (w_h) and after ($\bar{w}_h$) one relaxation step and expand the errors before (v_h) and after ($\bar{v}_h$) the relaxation step, namely

$$v_h := u_h - w_h \quad \text{and} \quad \bar{v}_h := u_h - \bar{w}_h,$$

into discrete eigenfunction series

$$v_h = \sum_{k,\ell=1}^{n-1} \alpha_{k,\ell}\varphi_h^{k,\ell}, \qquad \bar{v}_h = \sum_{k,\ell=1}^{n-1} \chi_h^{k,\ell}\alpha_{k,\ell}\varphi_h^{k,\ell}. \tag{2.1.7}$$

As discussed in Section 1.5, in order to analyze the smoothing properties of $S_h(\omega)$ we distinguish between *low* and *high* frequencies (with respect to the coarser grid Ω_{2h} used).

In order to measure the smoothing properties of $S_h(\omega)$ quantitatively, we introduce the *smoothing factor* of $S_h(\omega)$ as follows:

Definition Smoothing factor (of ω-JAC for Model Problem 1)
The *smoothing factor* $\mu(h;\omega)$ of $S_h(\omega)$, representing the worst factor by which *high frequency* error components are reduced per relaxation step, and its supremum μ^* over h, are defined as

$$\begin{aligned} \mu(h;\omega) &:= \max\{|\chi_h^{k,\ell}(\omega)|\colon n/2 \le \max(k,\ell) \le n-1\}, \\ \mu^*(\omega) &:= \sup_{h\in\mathcal{H}} \mu(h;\omega). \end{aligned} \tag{2.1.8}$$

From here on, $\mathcal{H}$ denotes the set of admissible (or reasonable) mesh sizes. Below we will see, for example, that for $\Omega = (0,1)^2$ the coarsest grid on which smoothing is applied is characterized by $h = 1/4$. In this case we would then define $\mathcal{H} = \{h = 1/n\colon n \in \mathbb{N},\ n \ge 4\}$.

Inserting (2.1.5), we obtain from (2.1.8)

$$\begin{aligned} \mu(h;\omega) &= \max\left\{\left|1 - \frac{\omega}{2}(2 - \cos k\pi h - \cos \ell\pi h)\right|\colon n/2 \le \max(k,\ell) \le n-1\right\} \\ \mu^*(\omega) &= \max\{|1 - \omega/2|, |1 - 2\omega|\}. \end{aligned} \tag{2.1.9}$$

This shows that Jacobi's relaxation has no smoothing properties for $\omega \le 0$ or $\omega > 1$

$$\mu(h;\omega) \ge 1 \quad \text{if } \omega \le 0 \quad \text{or } \omega > 1 \ \text{ (and } h \text{ sufficiently small).}$$

For $0 < \omega < 1$, however, the smoothing factor is smaller than 1 and bounded away from 1, independently of h. For $\omega = 1$, we have a smoothing factor of $1 - O(h^2)$ only. In particular, we find from (2.1.9) that

$$\mu(h;\omega) = \begin{cases} \cos\pi h & \text{if } \omega = 1 \\ (2 + \cos\pi h)/4 & \text{if } \omega = 1/2 \\ (1 + 2\cos\pi h)/5 & \text{if } \omega = 4/5 \end{cases} \qquad \mu^*(\omega) = \begin{cases} 1 & \text{if } \omega = 1 \\ 3/4 & \text{if } \omega = 1/2 \\ 3/5 & \text{if } \omega = 4/5. \end{cases} \tag{2.1.10}$$

The choice $\omega = 4/5$ is optimal in the following sense:

$$\inf\{\mu^*(\omega)\colon 0 \le \omega \le 1\} = \mu^*(4/5) = 3/5.$$

With respect to $\mu(h;\omega)$, one obtains

$$\inf\{\mu(h;\omega)\colon 0 \le \omega \le 1\} = \mu\left(h; \frac{4}{4 + \cos\pi h}\right) = \frac{3\cos\pi h}{4 + \cos\pi h} = \frac{3}{5} - |O(h^2)|.$$

This means that one step of ω-JAC with $\omega = 4/5$ reduces all high frequency error components by at least a factor of 3/5 (independent of the grid size h).

The above consideration is a first example of what we call *smoothing analysis*.

2.1.3 Gauss–Seidel-type Iteration (Relaxation)

In Section 1.5.1 we introduced Gauss–Seidel iteration with a lexicographic ordering of the grid points. A different ordering is the so-called red–black ordering (see Fig. 1.7). If we use this red–black ordering for Gauss–Seidel iteration, we obtain the Gauss–Seidel red–black (GS-RB) method.

Remark 2.1.1 The red–black ordering of grid points is also called odd–even ordering. This notation has the advantage that the two types of grid points are more clearly addressed (a grid point (x_i, y_j) is odd/even if $i+j$ is odd/even) than with the term red–black. Since red–black is more often used in the literature, we will stay with red–black. $\gg$

The significance of using a relaxation parameter ω in Gauss–Seidel iteration is well known in the classical Gauss–Seidel *convergence* theory. For Model Problem 1, lexicographic Gauss–Seidel with a relaxation parameter is described by

$$\begin{aligned} z_h^{m+1}(x_i, y_j) &= \tfrac{1}{4}\big[h^2 f_h(x_i, y_j) + u_h^{m+1}(x_i - h, y_j) + u_h^m(x_i + h, y_j) \\ &\quad + u_h^{m+1}(x_i, y_j - h) + u_h^m(x_i, y_j + h)\big] \\ u_h^{m+1} &= u_h^m + \omega(z_h^{m+1} - u_h^m). \end{aligned} \tag{2.1.11}$$

The parameter ω not only enters explicitly in (2.1.11), but also implicitly in the "new values" $u_h^{m+1}(x_i - h, y_j)$ and $u_h^{m+1}(x_i, y_j - h)$. We will call this algorithm ω-GS-LEX in the following. The corresponding algorithm with red–black ordering of the grid points and relaxation parameter ω is called ω-GS-RB in this book.

We recall that, for Model Problem 1, the *convergence* of Gauss–Seidel iteration can be substantially improved by an overrelaxation parameter ω^*. With

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho(\text{JAC})^2}} = \frac{2}{1 + \sin \pi h}$$

we obtain

$$\rho(\omega^*\text{-GS}) = \omega^* - 1 = \frac{1 - \sin \pi h}{1 + \sin \pi h} = 1 - O(h)$$

instead of

$$\rho(\text{GS}) = 1 - O(h^2) \quad (\text{for } \omega = 1).$$

This is the classical result on *successive overrelaxation* (SOR) [431]. Note that this result for Gauss–Seidel iteration is independent of the ordering of grid points.

Gauss–Seidel-type methods represent a particularly important class of smoothers. In the multigrid context the *smoothing properties* of Gauss–Seidel are much more important than the convergence properties. We will, however, not analyze the smoothing properties of Gauss–Seidel-type relaxations, here. Since, different from the Jacobi situation, the eigenfunctions of L_h (2.1.4) are *not* eigenfunctions of the Gauss–Seidel operator, we need different tools for this analysis, which will be discussed in detail in Chapter 4. Here, we summarize the results of the smoothing analysis for Gauss–Seidel-type relaxations. For Model Problem 1, we obtain the smoothing factors

$$\begin{aligned} \mu(\text{GS-LEX}) &= 0.50 \quad (\text{for } \omega = 1), \\ \mu(\text{GS-RB}) &= 0.25 \quad (\text{for } \omega = 1). \end{aligned}$$

(The factor of 0.25 for GS-RB is valid if only one or two smoothing steps are performed.) This result shows that the ordering of the grid points has an essential influence on the smoothing properties in the case of Gauss–Seidel relaxations. On the other hand, for Model Problem 1, the introduction of a relaxation parameter does *not* improve the smoothing properties of GS-LEX relaxation essentially (see Section 4.3).

The situation is somewhat different for GS-RB, for which an overrelaxation parameter can improve the smoothing properties (see Section 2.9 and [427, 428]).

2.1.4 Parallel Properties of Smoothers

Here, we compare the smoothing properties and the parallel features of ω-JAC, GS-LEX and GS-RB. First, ω-JAC is "*fully* Ω_h *parallel*". By this we mean that the ω-Jacobi operator (2.1.3) can be applied to *all* grid points Ω_h simultaneously; the new values do not depend on each other. We also say that the degree of parallelism (par-deg) is

$$\text{par-deg}(\omega\text{-JAC}) = \#\Omega_h.$$

If we use GS-LEX instead, we have dependences since we want to use the most recent values of u_h wherever possible. Grid points lying on a diagonal in Ω_h (see Fig. 2.1) are independent of each other for five-point discretizations and can be treated in parallel. The degree of parallelism here clearly varies from one diagonal to the next and is restricted by

$$\text{par-deg(GS-LEX)} \le (\#\Omega_h)^{1/2}.$$

In case of GS-RB each step of the Gauss–Seidel relaxation consists of two half-steps. In the first half-step, all red grid points (○) are treated simultaneously and independently (see Fig. 2.2). In the second half-step, all black grid points (●) are treated, using the updated values in the red points. The degree of parallelism is

$$\text{par-deg(GS-RB)} = \tfrac{1}{2}\#\Omega_h.$$

Table 2.1 summarizes the properties of these relaxation schemes for Model Problem 1: ω-JAC is fully parallel, but unfortunately not a really good smoother (not even with an

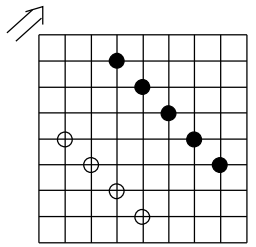


Figure 2.1. Diagonal grid points such as the ● (or the ○) can be treated in parallel in GS-LEX. Going from one diagonal to the next (↗) is sequential.

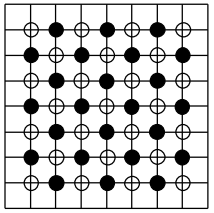


Figure 2.2. Red–black distribution of grid points in Ω_h.

Table 2.1. Smoothing factors for various relaxation schemes. The smoothing factors marked * will be obtained from the analysis in Sections 4.3 and 4.5; the factor marked † is only valid if at most one or two smoothing steps are performed. (Here, N denotes the number of grid points $\#\Omega_h$ corresponding to the unknown grid values of u_h.)

Relaxation	Smoothing factor	Smoothing	Parallel degree	
ω-JAC, $\omega = 1$	1	No	N	Full
ω-JAC, $\omega = 0.5$	0.75	Unsatisfactory	N	Full
ω-JAC, $\omega = 0.8$	0.6	Acceptable	N	Full
GS-LEX, $\omega = 1$	0.5*	Good	$\leq \sqrt{N}$	Square root
GS-RB, $\omega = 1$	0.25*†	Very good	$\frac{1}{2}N$	Half

optimized parameter ω) whereas GS-LEX has reasonable smoothing properties but is not satisfactorily parallel. However, GS-RB is both a very good smoother (much better than ω-JAC) and highly parallel.

2.2 INTRODUCING THE TWO-GRID CYCLE

As stated in Section 1.5, the basic multigrid consists of two ingredients: smoothing and coarse grid correction. We start with the two-grid cycle, the natural basis for any multigrid algorithm. For this purpose, we consider a discrete linear elliptic boundary value problem of the form

$$L_h u_h = f_h \quad (\Omega_h) \tag{2.2.1}$$

and assume that ${L_h}^{-1}$ exists.

As we are not going to give concrete quantitative results in this section, we do not make precise assumptions about the discrete operator L_h, the right-hand side f_h or the grid Ω_h. For a simple but characteristic example, one may always think of Model Problem 1 or some more general Poisson-like equation.

2.2.1 Iteration by Approximate Solution of the Defect Equation

One way of introducing the two-grid idea is to start from a general iteration based on an approximate solution of the defect equation.

For any *approximation* u_h^m of the solution u_h of (2.2.1), we denote the *error* by

$$v_h^m := u_h - u_h^m \tag{2.2.2}$$

and the *defect* (or *residual*) by

$$\boxed{d_h^m := f_h - L_h u_h^m}. \tag{2.2.3}$$

Trivially, the *defect equation*

$$L_h v_h^m = d_h^m \tag{2.2.4}$$

is equivalent to the original equation (2.2.1) since

$$u_h = u_h^m + v_h^m. \tag{2.2.5}$$

We describe these steps by the following procedural formulation:

$$u_h^m \longrightarrow d_h^m = f_h - L_h u_h^m \longrightarrow L_h v_h^m = d_h^m \longrightarrow u_h = u_h^m + v_h^m. \tag{2.2.6}$$

This procedure, however, is not a meaningful numerical process. However, if L_h is approximated here by any "simpler" operator $\hat{L}_h$ such that $\hat{L}_h^{-1}$ exists, the solution $\hat{v}_h^m$ of

$$\hat{L}_h \hat{v}_h^m = d_h^m \tag{2.2.7}$$

gives a new approximation

$$u_h^{m+1} := u_h^m + \hat{v}_h^m. \tag{2.2.8}$$

The procedural formulation then looks like

$$\boxed{u_h^m \longrightarrow d_h^m = f_h - L_h u_h^m \longrightarrow \hat{L}_h \hat{v}_h^m = d_h^m \longrightarrow u_h^{m+1} = u_h^m + \hat{v}_h^m}. \tag{2.2.9}$$

Starting with some u_h^0, the successive application of this process defines an *iterative procedure*. The *iteration operator* of this method is given by

$$M_h = I_h - C_h L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h), \tag{2.2.10}$$

where $C_h := (\hat{L}_h)^{-1}$ and I_h denotes the identity on $\mathcal{G}(\Omega_h)$. We have

$$u_h^{m+1} = M_h u_h^m + s_h \quad \text{with } s_h = (\hat{L}_h)^{-1} f_h \ \ (m = 0, 1, 2, \ldots). \tag{2.2.11}$$

For the errors, it follows that

$$v_h^{m+1} = M_h v_h^m = (I_h - C_h L_h) v_h^m \quad (m = 0, 1, 2, \dots) \tag{2.2.12}$$

and for the defects that

$$d_h^{m+1} = L_h M_h L_h^{-1} d_h^m = (I_h - L_h C_h) d_h^m \quad (m = 0, 1, 2, \dots). \tag{2.2.13}$$

If we start the iteration with $u^0 = 0$, then we can represent u_h^m $(m = 1, 2, \dots)$ as

$$\begin{aligned} u_h^m &= (I_h + M_h + M_h{}^2 + \cdots + M_h^{m-1})(\hat{L}_h)^{-1} f_h \\ &= (I_h - M_h^m)(I_h - M_h)^{-1}(\hat{L}_h)^{-1} f_h \\ &= (I_h - M_h^m) L_h^{-1} f_h. \end{aligned} \tag{2.2.14}$$

For general remarks on iterations such as (2.2.11), we refer to the discussion in Section 1.6.1. The asymptotic convergence properties of the above iterative process are characterized by the *spectral radius* (*asymptotic convergence factor*) of the iteration operator, i.e.

$$\rho(M_h) = \rho(I_h - C_h L_h) = \rho(I_h - L_h C_h). \tag{2.2.15}$$

If some norm $\|\cdot\|$ on $\mathcal{G}(\Omega_h)$ is introduced,

$$\|I_h - C_h L_h\|, \qquad \|I_h - L_h C_h\| \tag{2.2.16}$$

give upper bounds for the *error reduction factor* and the *defect reduction factor*, respectively, for one iteration.

Remark 2.2.1 Classical iterative linear solvers such as Jacobi or Gauss–Seidel iterations if applied to (2.2.1) can be interpreted as (iterated) approximate solvers for the defect equation. For ω-JAC we have, for example,

$$\hat{L}_h = \frac{1}{\omega} D_h$$

where D_h is the "diagonal" part of the matrix corresponding to L_h. Similarly, GS-LEX is obtained by setting $\hat{L}_h$ to be the "lower triangular" part of the matrix corresponding to L_h including its diagonal part. ≫

Remark 2.2.2 More generally, any of the classical iterative linear solvers of the form (2.2.11) can be interpreted as iterated approximate solvers for the defect equation if

$$C_h := (I_h - M_h) {L_h}^{-1}$$

is invertible. For then we can set $\hat{L}_h := {C_h}^{-1}$. ≫

2.2.2 Coarse Grid Correction

One idea to approximately solve the defect equation is to use an appropriate approximation L_H of L_h on a coarser grid Ω_H, for instance the grid with mesh size $H = 2h$. This means that the defect equation (2.2.4) is replaced by

$$L_H \hat{v}_H^m = d_H^m. \tag{2.2.17}$$

Here we assume

$$L_H : \mathcal{G}(\Omega_H) \to \mathcal{G}(\Omega_H), \qquad \dim \mathcal{G}(\Omega_H) < \dim \mathcal{G}(\Omega_h) \tag{2.2.18}$$

and ${L_H}^{-1}$ to exist. As d_H^m and $\hat{v}_H^m$ are grid functions on the coarser grid Ω_H, we assume two (linear) transfer operators

$$I_h^H : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_H), \qquad I_H^h : \mathcal{G}(\Omega_H) \to \mathcal{G}(\Omega_h) \tag{2.2.19}$$

to be given. I_h^H is used to *restrict* d_h^m to Ω_H:

$$\boxed{d_H^m := I_h^H d_h^m} \tag{2.2.20}$$

and I_H^h is used to *interpolate* (or *prolongate*) the correction $\hat{v}_H^m$ to Ω_h:

$$\boxed{\hat{v}_h^m := I_H^h \hat{v}_H^m}. \tag{2.2.21}$$

The simplest example for a restriction operator is the "injection" operator

$$d_H(P) = I_h^H d_h(P) := d_h(P) \quad \text{for } P \in \Omega_H \subset \Omega_h, \tag{2.2.22}$$

which identifies grid functions at coarse grid points with the corresponding grid functions at fine grid points. A fine and a coarse grid with the injection operator are presented in Fig. 2.3.

Altogether, one coarse grid correction step (calculating u_h^{m+1} from u_h^m) proceeds as follows.

Coarse grid correction $u_h^m \to u_h^{m+1}$

– Compute the defect	$d_h^m = f_h - L_h u_h^m$
– Restrict the defect (fine-to-coarse transfer)	$d_H^m = I_h^H d_h^m$
– Solve on Ω_H	$L_H \hat{v}_H^m = d_H^m$
– Interpolate the correction (coarse-to-fine transfer)	$\hat{v}_h^m = I_H^h \hat{v}_H^m$
– Compute a new approximation	$u_h^{m+1} = u_h^m + \hat{v}_h^m$

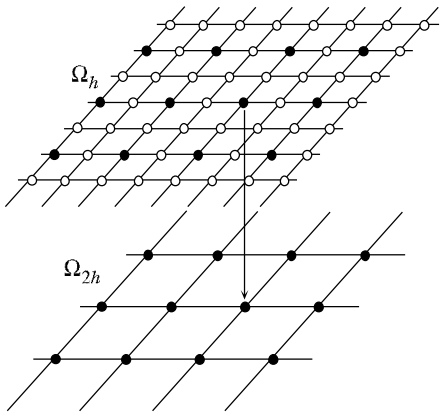


Figure 2.3. A fine and a coarse grid with the injection operator.

The associated iteration operator is given by

$$I_h - C_h L_h \quad \text{with } C_h = I_H^h L_H^{-1} I_h^H. \tag{2.2.23}$$

However:

Remark 2.2.3 Taken on its own, the coarse grid correction process is of no use. It is not convergent! We have

$$\rho\left(I_h - I_H^h\, L_H^{-1} I_h^H L_h\right) \geq 1. \tag{2.2.24}$$

$\gg$

This remark follows directly from the fact that I_h^H maps $\mathcal{G}(\Omega_h)$ into the lower dimensional space $\mathcal{G}(\Omega_H)$ and therefore $C_h = I_H^h L_H^{-1} I_h^H$ is not invertible. This implies that

$$C_h L_h v_h = 0 \quad \text{for certain } v_h \neq 0.$$

Example 2.2.1 It may be illustrative to see what $C_h L_h v_h = 0$ means in practice. For the simple injection operator (2.2.22), for example, any error function $v_h \in \mathcal{G}(\Omega_h)$ with

$$L_h v_h(P) = \begin{cases} 0 & \text{for } P \in \Omega_H \\ \text{arbitrary} & \text{for } P \notin \Omega_H \end{cases}$$

is annihilated by I_h^H and therefore by C_h. Such an error function v_h will thus not be changed by a pure coarse grid correction.

As a more concrete example, consider Model Problem 1 ($L_h = -\Delta_h$), $h = 1/n$, and

$$v_h(x, y) = \sin \frac{n}{2}\pi x \, \sin \frac{n}{2}\pi y. \tag{2.2.25}$$

For standard coarsening, we find

$$v_h(P) = L_h v_h(P) = I_h^{2h} L_h v_h(P) = 0 \quad \text{for all } P \in \Omega_{2h}.$$

(This relation holds for *any* restriction operator I_h^{2h} as long as the stencil of I_h^{2h} is symmetric in x and y.) Clearly, v_h belongs to the high frequency part of the eigenfunctions of L_h.

$\triangle$

2.2.3 Structure of the Two-grid Operator

The above considerations imply that it is necessary to combine the two processes of smoothing and of coarse grid correction.

Each iteration step (cycle) of a two-grid method consists of a *presmoothing*, a *coarse grid correction* and a *postsmoothing* part. One step of such an iterative two-grid method (calculating u_h^{m+1} from u_h^m) proceeds as follows:

Two-grid cycle $u_h^{m+1} = \text{TGCYC}(u_h^m, L_h, f_h, \nu_1, \nu_2)$

(1) Presmoothing
- Compute $\bar{u}_h^m$ by applying ν_1 (≥ 0) steps of a given smoothing procedure to u_h^m:

$$\bar{u}_h^m = \text{SMOOTH}^{\nu_1}(u_h^m, L_h, f_h). \tag{2.2.26}$$

(2) Coarse grid correction (CGC)
- Compute the defect $\bar{d}_h^m = f_h - L_h \bar{u}_h^m$.
- Restrict the defect (fine-to-coarse transfer) $\bar{d}_H^m = I_h^H \bar{d}_h^m$.
- Solve on Ω_H $L_H \hat{v}_H^m = \bar{d}_H^m$. (2.2.27)
- Interpolate the correction (coarse-to-fine transfer) $\hat{v}_h^m = I_H^h \hat{v}_H^m$.
- Compute the corrected approximation $u_h^{m,\text{after CGC}} = \bar{u}_h^m + \hat{v}_h^m$.

(3) Postsmoothing
- Compute u_h^{m+1} by applying ν_2 (≥ 0) steps of the given smoothing procedure to $u_h^{m,\text{after CGC}}$:

$$u_h^{m+1} = \text{SMOOTH}^{\nu_2}(u_h^{m,\text{after CGC}}, L_h, f_h). \tag{2.2.28}$$

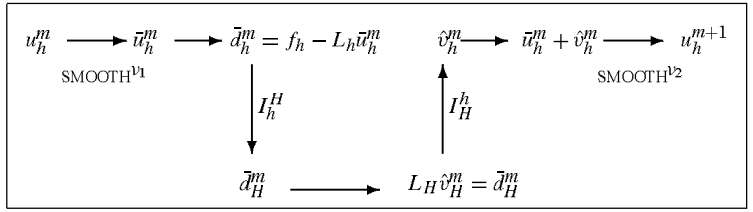


Figure 2.4. Structure of a two-grid cycle.

For the formal description of smoothing procedures we have used the notation

$$\bar{u}_h^m = \text{SMOOTH}^{\nu}(u_h^m, L_h, f_h).$$

With this notation we combine the advantages of a more mathematically oriented operator-like notation and a more computer science oriented formal procedural notation. In particular, the number ν of smoothing steps appears as an upper (power) index. Similar notation is also used for the two-grid and for the multigrid procedure.

The two-grid cycle is illustrated in Fig. 2.4.

From the above description, one immediately obtains the iteration operator M_h^H of the (h, H) two-grid cycle:

$$\boxed{M_h^H = S_h^{\nu_2} K_h^H S_h^{\nu_1} \quad \text{with } K_h^H := I_h - I_H^h L_H^{-1} I_h^H L_h} \,. \tag{2.2.29}$$

From Fig. 2.4, we see that the following individual *components of the two-grid cycle* have to be specified:

- the smoothing procedure SMOOTH (u_h^m, L_h, f_h),
- the numbers ν_1, ν_2 of smoothing steps,
- the coarse grid Ω_H,
- the fine-to-coarse restriction operator I_h^H,
- the coarse grid operator L_H,
- the coarse-to-fine interpolation operator I_H^h.

Experience with multigrid methods (and multigrid theory) shows that the choice of these components may have a strong influence on the efficiency of the resulting algorithm. On the other hand, there are no general simple rules on how to choose the individual components in order to construct optimal algorithms for complicated applications. One can, however, recommend certain choices for certain situations. The main objective of the *elementary multigrid theory* (see Chapter 3) and the *local Fourier analysis* (see Chapter 4) is to analyze the convergence properties of multigrid theoretically and to provide tools for the proper choice of multigrid components.

2.3 MULTIGRID COMPONENTS

In this section, we will introduce and list some important examples of how some of the multigrid components can be specified.

The idea of giving these specifications is to make our presentation more concrete and to introduce corresponding notations. The multigrid components specified here are needed in Section 2.5.1, where a specific multigrid Poisson solver is introduced. Therefore, readers who are more interested in the general structure of multigrid than in specific details may skip this section for the time being.

2.3.1 Choices of Coarse Grids

In this subsection, we will mention some possible and common choices for the grid Ω_H. The simplest and most frequently used choice is *standard coarsening*, doubling the mesh size h in every direction. Most of the results and considerations in this book refer to this choice. In d dimensions, the relation between the number of grid points (neglecting boundary effects) is

$$\#\Omega_H \approx \frac{1}{2^d}\#\Omega_h.$$

We speak of *semicoarsening* in 2D if the mesh size h is doubled in one direction only, i.e. $H = (2h_x, h_y)$ (x-semicoarsening, see Fig. 2.5) or $H = (h_x, 2h_y)$ (y-semicoarsening). This is especially of interest for anisotropic operators (see Section 5.1). Note that in this case

$$\#\Omega_H \approx \tfrac{1}{2}\#\Omega_h. \tag{2.3.1}$$

In 3D, we have additional types of semicoarsening: we can double the mesh size in one or in two directions (see Section 5.2).

We speak of *red–black coarsening* if the coarse grid points are distributed in the fine grid in a red–black (checkerboard) manner (see Fig. 2.5).

Also other coarsenings, like $4h$-coarsening, are sometimes of interest.

We mention that in the context of the AMG approach (see Appendix A), the coarse grid Ω_H is not formed according to such a fixed simple strategy. Using the algebraic relations in the corresponding matrix, Ω_H is determined by the AMG process itself in the course of calculation. We will see that the red–black coarsened grid is a standard choice for Model Problem 1 in AMG.

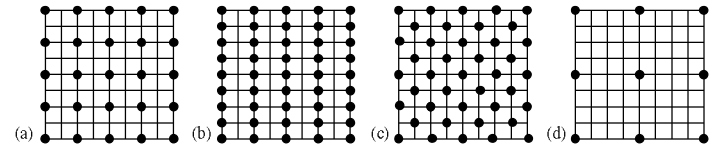


Figure 2.5. Examples of standard; x-semi-; red–black; $(h, 4h)$-coarsening in a square computational domain Ω. The grid points of Ω_H are marked by dots.

2.3.2 Choice of the Coarse Grid Operator

So far, we have not described precisely how the coarse grid operator L_H can be chosen. A natural choice is to use the direct analog of L_h on the grid Ω_H. For Model Problem 1, this means

$$L_H = \frac{1}{H^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_H .$$

The direct coarse grid analog of the fine grid operator L_h will be used in most parts of this book, in particular in all chapters on basic multigrid.

There are, however, applications and multigrid algorithms, which make use of a different approach. *The so-called Galerkin coarse grid operator is defined by*

$$L_H := I_h^H L_h I_H^h, \tag{2.3.2}$$

where I_h^H and I_H^h are appropriate transfer operators. We will return to the Galerkin approach in the context of problems with discontinuous coefficients in Chapter 7 and in Appendix A, where algebraic systems of equations without a grid-oriented background will be treated.

2.3.3 Transfer Operators: Restriction

The choice of restriction and interpolation operators I_h^H and I_H^h, for the intergrid transfer of grid functions, is closely related to the choice of the coarse grid. Here, we introduce transfer operators for standard coarsening (see Fig. 2.5), i.e. the grid transfers between the grid Ω_h and the $2h$-grid Ω_{2h}.

A *restriction operator* I_h^{2h} maps h-grid functions to $2h$-grid functions. One restriction operator already discussed is the *injection* operator (2.2.22). Another frequently used restriction operator is the *full weighting* (FW) operator, which in stencil notation reads

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h} . \tag{2.3.3}$$

Applying this operator to a grid function $d_h(x, y)$ at a coarse grid point $(x, y) \in \Omega_{2h}$ means

$$\begin{aligned} d_{2h}(x, y) &= I_h^{2h} d_h(x, y) \\ &= \tfrac{1}{16}[4d_h(x, y) + 2d_h(x+h, y) + 2d_h(x-h, y) + 2d_h(x, y+h) \\ &\quad + 2d_h(x, y-h) + d_h(x+h, y+h) + d_h(x+h, y-h) \\ &\quad + d_h(x-h, y+h) + d_h(x-h, y-h)]. \end{aligned} \tag{2.3.4}$$

Obviously, a nine-point weighted average of d_h is obtained.

Remark 2.3.1 The FW operator can be derived from a discrete version of the condition

$$\int_{\Omega_{x,y}} w(\tilde{x}, \tilde{y})\, d\Omega = \int_{\Omega_{x,y}} (I_h^{2h} w)(\tilde{x}, \tilde{y})\, d\Omega \tag{2.3.5}$$

for $\Omega_{x,y} = [x-h, x+h] \times [y-h, y+h]$ where the midpoint rule is used to approximate the integral on the right-hand side of the equation and the trapezoidal rule is used to approximate the integral on the left-hand side. ≫

Remark 2.3.2 The FW operator in d dimensions can also be constructed as a tensor product of the one-dimensional FW operators

$$I_h^{2h} = \frac{1}{4}[1 \quad 2 \quad 1] \qquad I_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

These 1D restriction operators are of particular interest in combination with the semicoarsening approach discussed in the previous section. They are also commonly used in case of *noneliminated* boundary conditions (see Section 5.6.2). ≫

A third restriction operator is *half weighting* (HW):

$$\frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}_h^{2h} . \tag{2.3.6}$$

2.3.4 Transfer Operators: Interpolation

The *interpolation (prolongation) operators* map $2h$-grid functions into h-grid functions. A very frequently used interpolation method is *bilinear interpolation* from $\mathbf{G}_{2h}$ to $\mathbf{G}_h$, which is given by:

$$I_{2h}^h \hat{v}_{2h}(x, y) = \begin{cases} \hat{v}_{2h}(x, y) & \text{for } \bullet \\ \frac{1}{2}[\hat{v}_{2h}(x, y+h) + \hat{v}_{2h}(x, y-h)] & \text{for } \square \\ \frac{1}{2}[\hat{v}_{2h}(x+h, y) + \hat{v}_{2h}(x-h, y)] & \text{for } \diamond \\ \frac{1}{4}[\hat{v}_{2h}(x+h, y+h) + \hat{v}_{2h}(x+h, y-h) & \\ \quad + \hat{v}_{2h}(x-h, y+h) + \hat{v}_{2h}(x-h, y-h)] & \text{for } \circ . \end{cases} \tag{2.3.7}$$

Figure 2.6 presents (part of) a fine grid with the symbols for the fine and coarse grid points referred to by (2.3.7).

Remark 2.3.3 Linear interpolation in d dimensions can easily be constructed by a recursive procedure (over the d dimensions) of 1D linear interpolations. Also d-dimensional higher order interpolations can be constructed and programmed very efficiently in this way. ≫

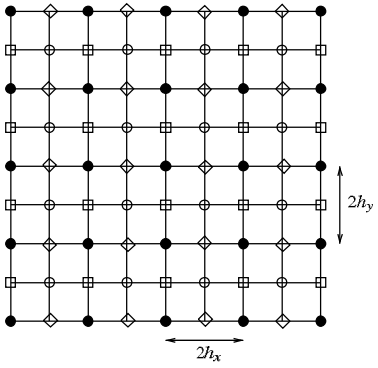


Figure 2.6. A fine grid with symbols indicating the bilinear interpolation (2.3.7) used for the transfer from the coarse grid (•).

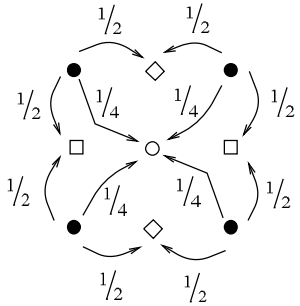


Figure 2.7. The distribution process for the bilinear interpolation operator. •, G_{2h} grid; □, ◇ and ○ as in (2.3.7) and in Fig. 2.6.

In stencil notation we write the bilinear interpolation operator I_{2h}^h (2.3.7) as

$$I_{2h}^h = \frac{1}{4}\left[\begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array}\right]_{2h}^{h} . \tag{2.3.8}$$

In this notation, the stencil entries correspond to weights in a *distribution process* as illustrated in Fig. 2.7. Therefore, the brackets are reversed.

For more general interpolation operators the stencils read

$$I_{2h}^h \triangleq]t_{\kappa_1,\kappa_2}[_{2h}^h := \left[\begin{array}{ccccccc} & & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & t_{-1,1} & t_{0,1} & t_{1,1} & \cdot & \cdot \\ \cdot & \cdot & t_{-1,0} & t_{0,0} & t_{1,0} & \cdot & \cdot \\ \cdot & \cdot & t_{-1,-1} & t_{0,-1} & t_{1,-1} & \cdot & \cdot \\ & & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & & \end{array}\right]_{2h}^{h} . \tag{2.3.9}$$

Again, only a finite number of coefficients t_{κ_1,κ_2} is assumed to be nonzero. The meaning of this stencil terminology is that coarse grid values are distributed to the fine grid and the weights in this distribution process are the factors t_{κ_1,κ_2}.

Remark 2.3.4 The formulas for linear interpolation of corrections can be applied near Dirichlet boundaries even if the boundary conditions have been eliminated. The corrections from a boundary point are assumed to be 0 in this case. ≫

2.4 THE MULTIGRID CYCLE

In Section 2.2 we have described the multigrid principle only in its two-grid version. In the multigrid context, two-grid methods are of little practical significance (due to the still large complexity of the coarse grid problem). However, they serve as the basis for the *multi*grid method.

Remark 2.4.1 Methods involving only two grids (or a fixed, small, number of grids) are of some interest in other frameworks, e.g. in the context of certain domain decomposition and related methods [362]. ≫

From two-grid to multigrid The *multigrid idea* starts from the observation that in a well converged two-grid method it is neither useful nor necessary to solve the coarse grid defect equation (2.2.27) exactly. Instead, without essential loss of convergence speed, one may replace $\hat{v}_H^m$ by a suitable approximation. A natural way to obtain such an approximation is to apply the two-grid idea to (2.2.27) again, now employing an even coarser grid than Ω_H.

This is possible, as obviously the coarse grid equation (2.2.27) is of the same form as the original equation (2.2.1). If the convergence factor of the two-grid method is small enough, it is sufficient to perform only a few, say γ, two-grid iteration steps (see Fig. 2.8) to obtain a good enough approximation to the solution of (2.2.27). This idea can, in a straightforward manner, be applied recursively, using coarser and coarser grids, down to some coarsest grid.

4

LOCAL FOURIER ANALYSIS

The local Fourier analysis (LFA) can be introduced and used in different ways. In this book, we present it as a formal tool. The emphasis is laid on explaining and demonstrating how it is applied in practice.

> In our view, LFA is the most powerful tool for the quantitative analysis and the design of efficient multigrid methods for general problems. We strongly recommend this approach to multigrid practitioners.

LFA (and also the idea of rigorous Fourier analysis) was introduced by Brandt [58] and extended and refined in [63, 69]. Contributions have been made by many others [378, 415]. Brandt prefers the term *local mode analysis* instead of LFA. Both terms denote the same approach.

In Section 4.1 we will discuss the LFA philosophy, in particular its local nature and its objective. LFA terminology will be introduced in Section 4.2. This section may appear somewhat formal at first sight, but the terminology is all elementary, transparent and useful. Some basic examples will be treated in Section 4.3, where the *smoothing factor* μ_{loc} will be defined. The smoothing factor is a very important quantity for the design of (efficient) multigrid methods. Using LFA μ_{loc} can easily be calculated for many smoothers as seen in Section 4.3. For red–black and other "pattern" smoothers, however, an extension of the definition of the smoothing factor is necessary, which we will introduce in Section 4.5.

Section 4.4 will describe the two-grid LFA. Two-grid LFA is needed if one wants more insight into the design and structure of a multigrid algorithm than smoothing analysis can give. Mathematically, the content of this section is closely related to the rigorous two-grid Fourier analysis (in Section 3.3.4).

In Section 4.6, we will list LFA results for basic multigrid algorithms. In Section 4.7, the notation and concept of h-ellipticity will be introduced, based on [63, 66]. The h-ellipticity establishes a qualitative criterion for the existence of local smoothers for a given discrete operator L_h. h-ellipticity is directly connected to the definition of the smoothing factor.

Some practical guidelines on the use of LFA for the design of efficient multigrid algorithms are given in Section 4.6.5. A computer program for performing LFA in general situations is available from `http://www.gmd.de/SCAI/multigrid/book.html`. This program was written by R. Wienands.

4.1 BACKGROUND

We first want to explain the *local* nature of LFA:

> **Local nature of LFA** Under general assumptions, any general discrete operator, nonlinear, with nonconstant coefficients, can be linearized *locally* and can be replaced locally (by freezing the coefficients) by an operator with constant coefficients. Thus, general *linear discrete operators with constant coefficients* are considered in LFA. Formally, they are defined on an infinite grid.

All considerations in the context of LFA are based on grid functions of the form

$$\varphi(\theta, x) = e^{i\theta x/h} \tag{4.1.1}$$

(actually on multidimensional analogs of them). Here, x varies in the given infinite grid $\mathbf{G}_h$ and θ is a continuous parameter that characterizes the frequency of the grid function under consideration.

> In the context of LFA, all operators which we will deal with (including L_h, the smoothing and intergrid transfer operators S_h, I_h^H, I_H^h, the coarse grid operator L_H as well as the coarse grid correction and two-grid operators K_h^H and M_h^H, respectively) will operate on these grid functions $\varphi(\theta, \cdot)$ and are considered on the infinite grids $\mathbf{G}_h$ and $\mathbf{G}_H$, respectively.
> The goal of LFA is to determine smoothing factors for S_h and two-grid convergence factors as well as error reduction factors for M_h^H. We denote these quantities by
>
> $$\mu_{\text{loc}}(S_h), \quad \rho_{\text{loc}}(M_h^H), \quad \sigma_{\text{loc}}(M_h^H).$$

The spectral radius ρ_{loc} gives insight into the *asymptotic convergence* behavior, whereas $\sigma_{\text{loc}}(M_h^H)$ refers to the error reduction *in one iteration step* measured in an appropriate norm.

Since the $\varphi(\theta, \cdot)$ are defined on the infinite grid $\mathbf{G}_h$, the *influence of boundaries and of boundary conditions is not taken into account.* At first sight, this may be regarded as a deficiency of the LFA approach. However, the idea of LFA is compatible with neglecting the effects of boundaries in the following sense.

The objective of LFA is to determine the quantitative convergence behavior and efficiency an appropriate multigrid algorithm *can* attain *if a proper boundary treatment is included.*

A proper boundary treatment typically requires only additional suitable *relaxations near the respective boundaries*, an amount of work which is negligible for $h \to 0$. In the following chapters, we will see examples of such boundary relaxation.

4.2 TERMINOLOGY

In describing the formalism of LFA, we confine ourselves to the 2D case and to standard coarsening in this section, for simplicity and in order to avoid formal complications (like many indices). But we now use a vector terminology that immediately carries over to d dimensions. We write $\boldsymbol{x} = (x_1, x_2)$ instead of (x, y) etc.

In the following,

$$\boldsymbol{h} = (h_1, h_2),$$

is a fixed mesh size (i.e., a vector). With $\boldsymbol{h}$ we associate the infinite grid $\mathbf{G}_h$ (1.3.3), which we now write in the form

$$\mathbf{G}_h = \{\boldsymbol{x} = \boldsymbol{kh} := (k_1 h_1, k_2 h_2),\ \boldsymbol{k} \in \mathbb{Z}^2\}.$$

On $\mathbf{G}_h$, we consider a discrete operator L_h corresponding to a difference stencil

$$L_h \triangleq [s_{\boldsymbol{\kappa}}]_h \quad (\boldsymbol{\kappa} = (\kappa_1, \kappa_2) \in \mathbb{Z}^2) \tag{4.2.1}$$

$$\text{i.e.} \quad L_h w_h(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in V} s_{\boldsymbol{\kappa}} w_h(\boldsymbol{x} + \boldsymbol{\kappa h}) \tag{4.2.2}$$

with *constant coefficients* $s_{\boldsymbol{\kappa}} \in \mathbb{R}$ (or $\mathbb{C}$), which, of course, will usually depend on $\boldsymbol{h}$. Here, $\boldsymbol{V}$ is again a finite index set.

The fundamental quantities in the LFA are the grid functions

$$\boxed{\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{i\boldsymbol{\theta}\boldsymbol{x}/\boldsymbol{h}} := e^{i\theta_1 x_1/h_1} e^{i\theta_2 x_2/h_2} \quad \text{for } \boldsymbol{x} \in \mathbf{G}_h}. \tag{4.2.3}$$

We assume that $\boldsymbol{\theta}$ varies continuously in $\mathbb{R}^2$. One recognizes that

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \mathbf{G}_h$$

if and only if

$$\theta_1 = \theta_1' (\text{mod } 2\pi) \quad \text{and} \quad \theta_2 = \theta_2' (\text{mod } 2\pi)$$

(i.e. if the difference between θ_1 and θ_1' and the difference between θ_2 and θ_2' are both multiples of 2π). Therefore, it is sufficient to consider

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \quad \text{with } \boldsymbol{\theta} \in [-\pi, \pi)^2.$$

For $\boldsymbol{\theta} \in [-\pi, \pi)^2$, we also use the notation $-\pi \le \boldsymbol{\theta} < \pi$, where the inequalities refer to both components θ_1 and θ_2.

The grid functions φ for $-\pi \le \boldsymbol{\theta} < \pi$ are linearly independent on $\mathbf{G}_h$. Since the $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ are defined on $\mathbf{G}_h$ and in that respect depend on $\boldsymbol{h}$, we sometimes write $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$ instead of $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$, for clarity.

Lemma 4.2.1 *For $-\pi \le \boldsymbol{\theta} < \pi$, all grid functions $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ are (formal) eigenfunctions of any discrete operator which can be described by a difference stencil as in (4.2.1). The relation*

$$L_h \varphi(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{L}_h(\boldsymbol{\theta}) \varphi(\boldsymbol{\theta}, \boldsymbol{x}) \quad (\boldsymbol{x} \in \mathbf{G}_h)$$

holds, with

$$\boxed{\tilde{L}_h(\boldsymbol{\theta}) = \sum_{\boldsymbol{\kappa}} s_{\boldsymbol{\kappa}} e^{i\boldsymbol{\theta} \cdot \boldsymbol{\kappa}}}. \tag{4.2.4}$$

We call $\tilde{L}_h(\boldsymbol{\theta})$ the formal eigenvalue *or the* symbol *of L_h.*

The proof of Lemma 4.2.1 is straightforward.

Example 4.2.1 The symbol of the standard discrete Laplace operator $L_h = -\Delta_h$ is

$$\tilde{L}_h(\boldsymbol{\theta}) = \frac{1}{h^2}(4 - (e^{i\theta_1} + e^{i\theta_2} + e^{-i\theta_1} + e^{-i\theta_2})) = \frac{2}{h^2}(2 - (\cos\theta_1 + \cos\theta_2)). \qquad \triangle$$

In addition to $\mathbf{G}_h$, we assume an (infinite) coarse grid

$$\mathbf{G}_H = \{\boldsymbol{x} = \boldsymbol{\kappa H} \colon \boldsymbol{\kappa} \in \mathbb{Z}^2\}$$

is obtained by standard coarsening of $\mathbf{G}_h$ ($\boldsymbol{H} = (2h_1, 2h_2)$). Other coarsenings can be treated similarly by LFA (see Section 4.6.3).

For the smoothing and two-grid analysis, we again have to distinguish *high and low frequency components* on $\mathbf{G}_h$ with respect to $\mathbf{G}_H$. The definition is based on the trivial but fundamental phenomenon that only those frequency components

$$\varphi(\boldsymbol{\theta}, \cdot) \quad \text{with} -\frac{\pi}{2} \le \boldsymbol{\theta} < \frac{\pi}{2}$$

are distinguishable on $\mathbf{G}_H$. For each $\boldsymbol{\theta}' \in [-\pi/2, \pi/2)^2$, three other frequency components $\varphi(\boldsymbol{\theta}, \cdot)$ with $\boldsymbol{\theta} \in [-\pi, \pi)^2$ coincide on $\mathbf{G}_H$ with $\varphi(\boldsymbol{\theta}', \cdot)$ and are not distinguishable (not "visible") on $\mathbf{G}_H$. Actually, we have

$$\boxed{\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \mathbf{G}_H \text{ if and only if } \boldsymbol{\theta} = \boldsymbol{\theta}' (\text{mod } \pi)} \tag{4.2.5}$$

(see also Lemma 4.4.1 below). This leads to the following definition.

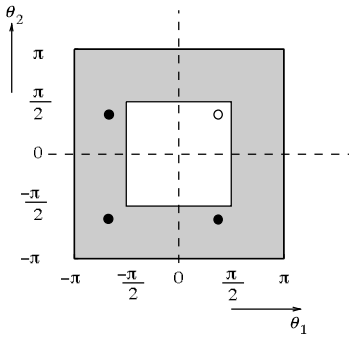


Figure 4.1. Low frequencies (interior white region) and high frequencies (shaded region); for a given low frequency $\boldsymbol{\theta}$(○), the three frequencies $\boldsymbol{\theta}$ for which the corresponding $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ coincide on G_H ($\boldsymbol{H} = (2h_1, 2h_2)$) are marked by •.

Definition 4.2.1 (high and low frequencies for standard coarsening)

$$\varphi \text{ low frequency component} \Longleftrightarrow \boldsymbol{\theta} \in T^{\text{low}} := \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$$

$$\varphi \text{ high frequency component} \Longleftrightarrow \boldsymbol{\theta} \in T^{\text{high}} := [-\pi, \pi)^2 \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$$

(see also Fig. 4.1).

This definition is analogous to (2.1.8) and (3.4.11), respectively. In addition to speaking of high and low frequency components $\varphi(\boldsymbol{\theta}, \cdot)$, we will, for simplicity, sometimes call the corresponding $\boldsymbol{\theta}$s high frequency or low frequency.

4.3 SMOOTHING ANALYSIS I

We will start our discussion and definition of the smoothing factor μ by looking at some very simple but fundamental examples. Considering the discrete Laplace operator $L_h = -\Delta_h$ ($h = h_1 = h_2$), we want to show how LFA is used to analyze the smoothing behavior of GS-LEX and similar relaxation schemes and how the smoothing factor is actually calculated. These examples are fundamental for two reasons. First, GS-LEX is the classical relaxation method which has been used, generalized and analyzed since the beginning of numerical analysis. Secondly, GS-LEX has natural smoothing properties, but the rigorous Fourier analysis presented in Chapter 3 cannot be applied directly to it. LFA, however, is applicable and shows the smoothing properties of GS-LEX very clearly.

Let us consider a discretized PDE, $L_h u_h = f_h$, and assume that a relaxation method can be written *locally* as

$$L_h^+ \bar{w}_h + L_h^- w_h = f_h, \tag{4.3.1}$$

where w_h corresponds to the old approximation of u_h (approximation before the relaxation step) and $\bar{w}_h$ to the new approximation (after the step). In this sense, the relaxation is characterized by a splitting

$$L_h = L_h^+ + L_h^-. \tag{4.3.2}$$

Example 4.3.1 (GS-LEX) For GS-LEX applied to the 2D Laplace operator $L_h = -\Delta_h$, the splitting reads

$$L_h^+ = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ -1 & 4 & 0 \\ & -1 & \end{bmatrix}, \qquad L_h^- = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ 0 & 0 & -1 \\ & 0 & \end{bmatrix}. \tag{4.3.3}$$

△

Example 4.3.2 (GS-LEX in 3D) In 3D, GS-LEX for $L_h = -\Delta_h$ is characterized by

$$L_h^+ = \frac{1}{h^2}\left[\begin{bmatrix}0&0&0\\0&-1&0\\0&0&0\end{bmatrix}_h \begin{bmatrix}0&0&0\\-1&6&0\\0&-1&0\end{bmatrix}_h \begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}_h\right]$$

and

$$L_h^- = \frac{1}{h^2}\left[\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}_h \begin{bmatrix}0&-1&0\\0&0&-1\\0&0&0\end{bmatrix}_h \begin{bmatrix}0&0&0\\0&-1&0\\0&0&0\end{bmatrix}_h\right]. \qquad \triangle$$

Example 4.3.3 (ω-JAC) In 2D, ω-JAC for $L_h = -\Delta_h$ is characterized by

$$L_h^+ = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ 0 & 4/\omega & 0 \\ & 0 & \end{bmatrix}, \qquad L_h^- = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4(1-1/\omega) & -1 \\ & -1 & \end{bmatrix}. \tag{4.3.4}$$

△

Subtracting (4.3.1) from the discrete equation $L_h u_h = f_h$, we obtain the local relation

$$L_h^+ \bar{v}_h + L_h^- v_h = 0$$

or

$$\bar{v}_h = S_h v_h$$

for the errors $\bar{v}_h = u_h - \bar{w}_h$, $v_h = u_h - w_h$, where S_h is the resulting smoothing operator. From this we immediately get the (infinite grid) Fourier representation of L_h^-, L_h^+, S_h.

Applying L_h^- and L_h^+ to the formal eigenfunctions $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$, we obtain

$$L_h^- e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/\boldsymbol{h}} = \tilde{L}_h^-(\boldsymbol{\theta}) e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/\boldsymbol{h}}$$
$$L_h^+ e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/\boldsymbol{h}} = \tilde{L}_h^+(\boldsymbol{\theta}) e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/\boldsymbol{h}},$$

where the $\tilde{L}_h^-$ and $\tilde{L}_h^+$ are the *symbols* (formal eigenvalues) of the operators L_h^- and L_h^+, respectively.

Lemma 4.3.1 *Under the assumptions (4.3.1) and (4.3.2), all $\varphi(\boldsymbol{\theta}, \cdot)$ with $\tilde{L}_h^+(\boldsymbol{\theta}) \neq 0$ are eigenfunctions of S_h:*

$$S_h \varphi(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{S}_h(\boldsymbol{\theta}) \varphi(\boldsymbol{\theta}, \boldsymbol{x}) \quad (-\pi \leq \boldsymbol{\theta} < \pi) \tag{4.3.5}$$

with the amplification factor

$$\tilde{S}_h(\boldsymbol{\theta}) := -\frac{\tilde{L}_h^-\boldsymbol{\theta})}{\tilde{L}_h^+(\boldsymbol{\theta})}. \tag{4.3.6}$$

Example 4.3.4 For GS-LEX, we have, for example,

$$L_h^+ e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/h} = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ -1 & 4 & 0 \\ & -1 & \end{bmatrix} e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/h}$$
$$= \frac{1}{h^2}(4 - e^{-i\theta_1} - e^{-i\theta_2}) e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/h}.$$

The symbols $\tilde{L}_h^-$, $\tilde{L}_h^+$, $\tilde{S}_h$ of the splitting in Example 4.3.1 are thus

$$\tilde{L}_h^+(\boldsymbol{\theta}) = \frac{1}{h^2}(4 - e^{-i\theta_1} - e^{-i\theta_2})$$
$$\tilde{L}_h^-(\boldsymbol{\theta}) = -\frac{1}{h^2}(e^{i\theta_1} + e^{i\theta_2})$$
$$\tilde{S}_h(\boldsymbol{\theta}) = -\frac{\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^+(\boldsymbol{\theta})} = \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}. \qquad \triangle$$

Based on Lemma 4.3.1, we can immediately define the smoothing factor. According to Definition 4.2.1, we distinguish high and low frequency components $\varphi(\boldsymbol{\theta}, \cdot)$ and define the smoothing factor $\mu_{\text{loc}}(S_h)$ by

Definition 4.3.1

$$\mu_{\text{loc}} = \mu_{\text{loc}}(S_h) := \sup\{|\tilde{S}_h(\boldsymbol{\theta})| : \boldsymbol{\theta} \in T^{\text{high}}\}. \tag{4.3.7}$$

For GS-LEX and standard coarsening, for example, this definition results in

$$\mu_{\text{loc}}(S_h) = \sup\left\{\left|\frac{e^{i\theta_1} + e^{i\theta_2}}{e^{-i\theta_1} + e^{-i\theta_2} - 4}\right| : \boldsymbol{\theta} \in T^{\text{high}}\right\}.$$

Since $\tilde{S}_h(\boldsymbol{\theta})$ attains its maximum 0.5 for $(\theta_1, \theta_2) = (\pi/2, \arccos 4/5)$, we obtain

$$\mu_{\text{loc}}(S_h) = 0.5 \quad \text{for GS-LEX}.$$

The above definition of a smoothing factor and its value of 0.5 for GS-LEX were first given by Brandt [57].

Example 4.3.5 (smoothing factor for ω-JAC) For $L_h = -\Delta_h$, the application to ω-JAC is straightforward. According to (4.3.4) and Definition 4.3.1, the symbol $\tilde{S}_h(\boldsymbol{\theta})$ is

$$\tilde{S}_h(\omega, \boldsymbol{\theta}) = 1 - \frac{\omega}{2}(2 - \cos\theta_1 - \cos\theta_2).$$

For the smoothing factor, we obtain

$$\mu_{\text{loc}}(S_h(\omega)) = \max\left\{\left|1 - \frac{\omega}{2}\right|, |1 - 2\omega|\right\}. \qquad \triangle$$

This is the same result that we have obtained in Section 2.1.2 for $\mu^*(\omega)$. So, ω-JAC for symmetric operators can be treated by both rigorous Fourier analysis and LFA. The results are the same, apart from the following slight formal difference: in comparison with (2.1.9), we recognize that the h-dependence of $\mu(S_h(\omega))$ has disappeared here. This is due to the fact that $\boldsymbol{\theta}$ varies continuously in T^{high}.

Remark 4.3.1 If L_h corresponds to a differential operator which contains derivatives of different orders, $\mu_{\text{loc}}(S_h)$ will, in general, depend explicitly on h. $\gg$

Example 4.3.6 (SOR or ω-GS-LEX) We again consider the discrete Laplacian $L_h = -\Delta_h$. If we apply GS-LEX with an additional relaxation parameter ω, we speak of *ω-GS-LEX relaxation* corresponding to the *classical SOR method* [431].

We can apply LFA (smoothing analysis) to investigate the influence of the parameter ω on the smoothing properties of SOR. The splitting of L_h for SOR is given by

$$L_h^+ = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ -1 & 4/\omega & 0 \\ & -1 & \end{bmatrix}, \qquad L_h^- = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ 0 & 4(1 - 1/\omega) & -1 \\ & 0 & \end{bmatrix}. \tag{4.3.8}$$

Figure 4.2 shows the behavior of the smoothing factor $\mu_{\text{loc}}(S_h(\omega))$ with respect to ω. Using a relaxation parameter $\omega \neq 1$ in ω-GS-LEX hardly improves its smoothing property for this problem *in 2D*. The optimal value of ω is not exactly 1 but very close to 1. The gain by using the optimal ω would be very small and does not pay if the additional work (two operations per point per smoothing step) is taken into account. $\triangle$

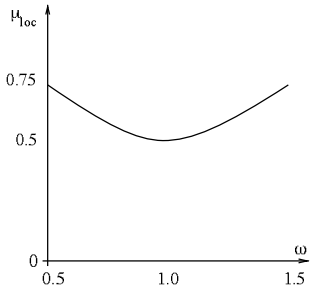


Figure 4.2. μ_{loc} (ω-GS-LEX) as a function of ω.

Many practically important smoothing procedures fulfil the assumptions of Lemma 4.3.1 and the corresponding Definition 4.3.1 of a smoothing factor can be applied to them. As we have seen, pointwise GS-LEX and ω-JAC type relaxations are among them, but also *block versions* of GS and ω-JAC (line and plane relaxation), which we will discuss in Sections 5.1 and 5.2, can be treated this way. Furthermore, ILU-type smoothers, which will be treated in Section 7.5, belong to this class of smoothers.

On the other hand, relaxation methods like *red–black* Gauss–Seidel do *not* fulfil the above assumptions. In particular, they cannot be represented directly by a splitting as in (4.3.1). However, the above assumptions can be generalized to cover red–black-type relaxations (see Section 4.5).

4.4 TWO-GRID ANALYSIS

Let us now apply LFA to the *two-grid operator*

$$M_h^H = S_h^{\nu_2} K_h^H S_h^{\nu_1} \quad \text{with the coarse grid correction operator } K_h^H. \tag{4.4.1}$$

In order to calculate convergence factors and other relevant quantities of M_h^H, we will analyze how the operators L_h, I_h^H, L_H, I_H^h and S_h act on the Fourier components $\varphi(\boldsymbol{\theta}, \cdot)$. The analysis uses the fact that quadruples of the $\varphi(\boldsymbol{\theta}, \cdot)$ coincide on $\mathbf{G}_H$. For any low frequency $\boldsymbol{\theta} = (\theta_1, \theta_2) \in T^{\text{low}} = [-\pi/2, \pi/2)^2$, we consider the frequencies

$$\begin{aligned} \boldsymbol{\theta}^{(0,0)} &:= (\theta_1, \theta_2), & \boldsymbol{\theta}^{(1,1)} &:= (\bar{\theta}_1, \bar{\theta}_2), \\ \boldsymbol{\theta}^{(1,0)} &:= (\bar{\theta}_1, \theta_2), & \boldsymbol{\theta}^{(0,1)} &:= (\theta_1, \bar{\theta}_2), \end{aligned}$$

where

$$\bar{\theta}_i := \begin{cases} \theta_i + \pi & \text{if } \theta_i < 0 \\ \theta_i - \pi & \text{if } \theta_i \geq 0 \end{cases} \tag{4.4.2}$$

(see also Fig. 4.1).

Lemma 4.4.1

(1) For any low frequency $\boldsymbol{\theta}^{(0,0)} \in T^{\text{low}}$, we have

$$\varphi(\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}^{(1,0)}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}^{(0,1)}, \boldsymbol{x}) \quad (\boldsymbol{x} \in \mathbf{G}_{2h}). \tag{4.4.3}$$

(2) Each of these four Fourier components $\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) = \varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ with $\alpha \in \{(0,0), (1,1), (1,0), (0,1)\}$ coincides on $\mathbf{G}_{2h}$ with the respective grid function $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$:

$$\varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \boldsymbol{x}) \equiv \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \boldsymbol{x}) \quad (\boldsymbol{x} \in \mathbf{G}_{2h}). \tag{4.4.4}$$

The proof of this lemma is straightforward.

For $\boldsymbol{\theta}^{(1,1)}$, for example, we find

$$\begin{aligned} \varphi_h(\boldsymbol{\theta}^{(1,1)}, \boldsymbol{x}) &= e^{i\theta_1 x_1/h_1} e^{i\theta_2 x_2/h_2} = e^{i(\theta_1 \pm \pi)x_1/h_1} e^{i(\theta_2 \pm \pi)x_2/h_2} \\ &= e^{i\theta_1 x_1/h_1} e^{\pm i\pi 2k_1} e^{i\theta_2 x_2/h_2} e^{\pm i\pi 2k_2} \quad (x_i = 2k_i h \in \mathbf{G}_{2h},\ k_i \in \mathbb{Z}) \\ &= e^{i\theta_1 2x_1/(2h_1)} e^{i\theta_2 2x_2/(2h_2)} = \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}). \end{aligned}$$

Definition 4.4.1 (harmonics for standard coarsening) The corresponding four $\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ (and sometimes also the corresponding frequencies $\boldsymbol{\theta}^{\boldsymbol{\alpha}}$) are called harmonics (of each other). For a given $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in T^{\text{low}}$, we define its *four-dimensional space of harmonics* by

$$E_h^{\boldsymbol{\theta}} := \text{span}[\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) : \boldsymbol{\alpha} = (\alpha_1, \alpha_2) \in \{(0,0), (1,1), (0,1), (1,0)\}]. \tag{4.4.5}$$

The significance of these spaces $E_h^{\boldsymbol{\theta}}$ is that they turn out to be invariant *under the two-grid operator M_h^H under general assumptions.*

All $\varphi_h(\boldsymbol{\theta}, \cdot)$ are formal eigenfunctions of L_h, and under the assumptions made in the previous section also of S_h. The operator K_h^H intermixes Fourier components with each other. This is a consequence of the fact that the two different grids, $\mathbf{G}_h$ and $\mathbf{G}_H$, are involved. In the following, we will discuss this behavior in detail. For that purpose, we consider an arbitrary $\psi \in E_h^{\boldsymbol{\theta}}$. We can represent ψ in the form

$$\psi = A^{(0,0)}\varphi(\boldsymbol{\theta}^{(0,0)}, \cdot) + A^{(1,1)}\varphi(\boldsymbol{\theta}^{(1,1)}, \cdot) + A^{(1,0)}\varphi(\boldsymbol{\theta}^{(1,0)}, \cdot) + A^{(0,1)}\varphi(\boldsymbol{\theta}^{(0,1)}, \cdot) \tag{4.4.6}$$

with uniquely defined coefficients or amplitudes A^{α}. We will analyze how the coefficients A^{α} are transformed if the multigrid components in (4.4.1) are applied to ψ. We will treat each of the multigrid components separately in the following, but first summarize the results with respect to K_h^H and M_h^H. For clarity, we now write $2h$ instead of H.

For the following theorem, we make the general assumption that L_h, I_h^{2h}, L_{2h} and I_{2h}^h are represented by stencils on $\mathbf{G}_h$ and $\mathbf{G}_{2h}$. Furthermore, in forming K_h^{2h} and M_h^{2h}

according to (4.4.1), we have implicitly assumed that $(L_{2h})^{-1}$ exists. This assumption will be discussed below. All other assumptions are made explicitly in the theorem itself.

Theorem 4.4.1

(1) Under the above assumptions, the coarse grid correction operator K_h^{2h} is represented on E_h^θ by the (4×4)-matrix $\hat{K}_h^{2h}(\boldsymbol{\theta})$

$$\hat{K}_h^{2h}(\boldsymbol{\theta}) = \hat{I}_h - \hat{I}_{2h}^h(\boldsymbol{\theta})\,(\hat{L}_{2h}(2\boldsymbol{\theta}))^{-1}\hat{I}_h^{2h}(\boldsymbol{\theta})\hat{L}_h(\boldsymbol{\theta}) \tag{4.4.7}$$

for each $\boldsymbol{\theta} \in T^{\text{low}}$. Here, $\hat{I}_h$, $\hat{L}_h(\boldsymbol{\theta})$ are (4×4)-matrices, $\hat{I}_h^{2h}(\boldsymbol{\theta})$ is a (4×1)-matrix, $(\hat{L}_{2h}(2\boldsymbol{\theta}))^{-1}$ is a (1×1)-matrix, and $\hat{I}_{2h}^h(\boldsymbol{\theta})$ is a (1×4)-matrix.
In other words: If we apply K_h^{2h} to any $\psi \in E_h^\theta$, the corresponding coefficients A^α in (4.4.6) are transformed according to

$$\begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \Leftarrow \hat{K}_h^{2h}(\boldsymbol{\theta}) \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}. \tag{4.4.8}$$

(2) If the spaces E_h^θ are invariant under the smoothing operator S_h, i.e.

$$S_h : E_h^\theta \to E_h^\theta \quad \text{for all } \boldsymbol{\theta} \in T^{\text{low}}, \tag{4.4.9}$$

we also obtain a representation of M_h^{2h} on E_h^θ by a (4×4)-matrix $\hat{M}_h^{2h}(\boldsymbol{\theta})$ with respect to E_h^θ. Here,

$$\hat{M}_h^{2h}(\boldsymbol{\theta}) = \hat{S}_h(\boldsymbol{\theta})^{\nu_2}\hat{K}_h^{2h}(\boldsymbol{\theta})\,\hat{S}_h(\boldsymbol{\theta})^{\nu_1} \tag{4.4.10}$$

with $\hat{K}_h^{2h}(\boldsymbol{\theta})$ from (4.4.7) and the (4×4)-matrix $\hat{S}_h(\boldsymbol{\theta})$ which represents S_h. This means that $M_h^{2h}\psi$ can be written as

$$\begin{aligned} M_h^{2h}\psi = {} & B^{(0,0)}\varphi(\boldsymbol{\theta}^{(0,0)},\cdot) + B^{(1,1)}\varphi(\boldsymbol{\theta}^{(1,1)},\cdot) \\ & + B^{(1,0)}\varphi(\boldsymbol{\theta}^{(1,0)},\cdot) + B^{(0,1)}\varphi(\boldsymbol{\theta}^{(0,1)},\cdot) \end{aligned} \tag{4.4.11}$$

where

$$\begin{pmatrix} B^{(0,0)} \\ B^{(1,1)} \\ B^{(1,0)} \\ B^{(0,1)} \end{pmatrix} = \hat{M}_h^{2h}(\boldsymbol{\theta}) \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}. \tag{4.4.12}$$

We will prove the above theorem and specify the (4×4)-matrices $\hat{K}_h^{2h}(\boldsymbol{\theta})$ and $\hat{M}_h^{2h}(\boldsymbol{\theta})$ in more detail below. But first, we discuss the assumption that $(L_{2h})^{-1}$ exists. Apart from the existence of $(L_{2h})^{-1}$, the existence of $(L_h)^{-1}$ is also implicitly assumed. If $(L_h)^{-1}$ does not exist, we can, in general, not expect that any iterative method will be convergent. As the following example shows, these assumptions are *not formally fulfilled* even in standard situations.

Example 4.4.1 Let $L_h = -\Delta_h$ and $L_{2h} = -\Delta_{2h}$ be discretizations of the Laplacian on the infinite grid. Then, obviously, for $\boldsymbol{\theta} = 0$ (i.e. $\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv 1$), the corresponding symbols (formal eigenvalues) are zero:

$$\tilde{L}_h(\boldsymbol{\theta}) = \tilde{L}_{2h}(\boldsymbol{\theta}) = 0.$$

In addition, $\tilde{L}_{2h}(\boldsymbol{\theta}) = 0$ for all four harmonics which coincide on $\mathbf{G}_{2h}$ with $\varphi \equiv 1$, i.e. $\boldsymbol{\theta} = (\theta_1, \theta_2) \in \{(0,0), (-\pi, 0), (0, -\pi), (-\pi, -\pi)\}$. We will find the same behavior for those L_h $(= [s_{\kappa,h}])$ and L_{2h} $(= [s_{\kappa,2h}])$, for which

$$\sum_{\kappa\in V} s_{\kappa,h} = 0 \quad \text{and} \quad \sum_{\kappa\in V} s_{\kappa,2h} = 0, \text{ respectively.}$$

△

Remark 4.4.1 These sums are 0 for consistent discretizations of those differential operators L on an infinite grid which contain only derivatives of u. ≫

The complications illustrated in Example 4.4.1 are *only formal ones* (due to the infinite grid assumption). If we have, for example, an application with Dirichlet boundary conditions, we do not have to bother about these symbols and the corresponding eigenfunctions. If we have a singular discrete operator (due to, for example, periodic or pure Neumann boundary conditions), this singular behavior has to be taken into account separately by any iterative solver. In order to make sure that M_h^{2h} can be formed and gives a reasonable iterative operator, we remove all $\boldsymbol{\theta}$ from our analysis for which L_h or L_{2h} have the symbol 0. That is, we will exclude the set

$$\Lambda = \left\{\boldsymbol{\theta} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2 : \tilde{L}_h(\boldsymbol{\theta}) = 0 \text{ or } \tilde{L}_{2h}(\boldsymbol{\theta}) = 0\right\}. \tag{4.4.13}$$

From the above theorem, we can draw conclusions for the convergence behavior of the two-grid method under consideration: *we have reduced the problem to the investigation (of spectral radii and norms) of (4×4)-matrices.*

Supplement Based on the representation of M_h^{2h} by the (4×4)-matrices $\hat{M}_h^{2h}(\boldsymbol{\theta})$, we can calculate the *asymptotic convergence factor*

$$\rho_{\mathrm{loc}}(M_h^{2h}) = \sup\{\rho_{\mathrm{loc}}(\hat{M}_h^{2h}(\boldsymbol{\theta})) \colon \boldsymbol{\theta} \in T^{\mathrm{low}},\ \boldsymbol{\theta} \notin \Lambda\}. \tag{4.4.14}$$

Here, $\rho_{\mathrm{loc}}(\hat{M}_h^{2h}(\boldsymbol{\theta}))$ is the spectral radius of the (4×4)-matrix $\hat{M}_h^{2h}(\boldsymbol{\theta})$.
Similarly, we can introduce *an error reduction factor* $\sigma_{\mathrm{loc}}(M_h^{2h})$ with respect to an appropriate norm. In particular, we consider

$$\sigma_{\mathrm{loc},S}(M_h^{2h}) = \sup\{||\hat{M}_h^{2h}(\boldsymbol{\theta})||\colon \boldsymbol{\theta} \in T^{\mathrm{low}},\ \boldsymbol{\theta} \notin \Lambda\}. \tag{4.4.15}$$

Here $||\cdot||$ denotes the spectral norm associated with the Euclidean vector norm in $\mathbb{C}^4$.

We will now derive the representation of M_h^{2h} by (4×4)-matrices in Theorem 4.4.1 by considering each multigrid component separately.

Operators L_h, I_h Under the general assumption that L_h is characterized by the difference stencil $L_h \triangleq [s_{\boldsymbol{\kappa}}]_h$, Lemma 4.2.1 immediately implies that the $A^{\boldsymbol{\alpha}}$ are transformed by L_h as follows:

$$\begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \Leftarrow \begin{pmatrix} \tilde{L}_h(\boldsymbol{\theta}^{(0,0)}) & & & \\ & \tilde{L}_h(\boldsymbol{\theta}^{(1,1)}) & & \\ & & \tilde{L}_h(\boldsymbol{\theta}^{(1,0)}) & \\ & & & \tilde{L}_h(\boldsymbol{\theta}^{(0,1)}) \end{pmatrix} \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}. \tag{4.4.16}$$

We denote this (4×4)-matrix by $\hat{L}_h(\boldsymbol{\theta})$. Trivially, the identity operator I_h is represented by the (4×4)-identity matrix on $E_h^{\boldsymbol{\theta}}$.

Restriction operator I_h^{2h} We also assume that the transfer operator I_h^{2h} is characterized by a stencil

$$I_h^{2h} \triangleq [\hat{t}_{\boldsymbol{\kappa}}]_h^{2h}, \qquad \text{i.e. } I_h^{2h} w_h(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in V} \hat{t}_{\boldsymbol{\kappa}} w_h(\boldsymbol{x} + \boldsymbol{\kappa}\boldsymbol{h}), \quad (\boldsymbol{x} \in \mathbf{G}_{2h}) \tag{4.4.17}$$

with a finite index set V. Then we obtain

$$I_h^{2h} \varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) = \tilde{I}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot) \tag{4.4.18}$$

(see (4.4.4)) with

$$\tilde{I}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) := \sum_{\boldsymbol{\kappa} \in V} \hat{t}_{\boldsymbol{\kappa}} e^{i\boldsymbol{\theta}^{\boldsymbol{\alpha}} \cdot \boldsymbol{\kappa}}. \tag{4.4.19}$$

For the coefficients A^{α}, this results in the transformation

$$A_{2h} = \hat{I}_h^{2h}(\boldsymbol{\theta}) \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix}, \tag{4.4.20}$$

where A_{2h} is the resulting coefficient of the $2h$ Fourier component $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$ and $\hat{I}_h^{2h}(\boldsymbol{\theta})$ is the (1×4)-matrix

$$[\tilde{I}_h^{2h}(\boldsymbol{\theta}^{(0,0)}), \tilde{I}_h^{2h}(\boldsymbol{\theta}^{(1,1)}), \tilde{I}_h^{2h}(\boldsymbol{\theta}^{(1,0)}), \tilde{I}_h^{2h}(\boldsymbol{\theta}^{(0,1)})].$$

Example 4.4.2 (injection and FW) For injection, the Fourier symbol is 1 for all frequencies:

$$\tilde{I}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = 1$$

For FW, we obtain

$$\tilde{I}_h^{2h}(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \tfrac{1}{4}(1 + \cos\bar{\theta}_1)(1 + \cos\bar{\theta}_2)$$

from (4.4.19) with $\bar{\theta}_i$ as in (4.4.2). △

Solution on $2h$-grid According to the general assumption, L_{2h} is also given by a stencil:

$$L_{2h} \triangleq [s_{\boldsymbol{\kappa},2h}]_{2h}.$$

For $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)}$ and $\boldsymbol{\theta} \in T^{\mathrm{low}}$, we thus have

$$L_{2h} \varphi_{2h}(2\boldsymbol{\theta}, \cdot) = \tilde{L}_{2h}(2\boldsymbol{\theta}) \varphi_{2h}(2\boldsymbol{\theta}, \cdot) \tag{4.4.21}$$

with the symbol

$$\boxed{\tilde{L}_{2h}(2\boldsymbol{\theta}) = \sum_{\boldsymbol{\kappa} \in V} s_{\boldsymbol{\kappa},2h} e^{i2\boldsymbol{\theta}\cdot\boldsymbol{\kappa}}}. \tag{4.4.22}$$

The solution on the $2h$-grid induces (for $2\boldsymbol{\theta} \notin \Lambda$) for the respective coefficient

$$A_{2h} \Leftarrow \frac{1}{(\tilde{L}_{2h}(2\boldsymbol{\theta}))} A_{2h}. \tag{4.4.23}$$

Interpolation operator I_{2h}^h For the interpolation operator, we use the stencil notation from Section 2.3.4. We can prove the following representation for I_{2h}^h if applied to $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$:

$$\boxed{I_{2h}^h \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot) = \sum_{\boldsymbol{\alpha}} \tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) \varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) \quad \text{with } \tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \frac{1}{4} \sum_{\boldsymbol{\kappa} \in V} t_{\boldsymbol{\kappa}} e^{i\boldsymbol{\theta}^{\boldsymbol{\alpha}} \cdot \boldsymbol{\kappa}}}.$$

Interpreting this representation for the coefficients A^{α} and A_{2h} of $\varphi_h(\boldsymbol{\theta}^{\alpha}, \cdot)$ and $\varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$, we obtain

$$\begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \Leftarrow \hat{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)}) A_{2h} \quad \text{with } \hat{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)}) = \begin{pmatrix} \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,1)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,0)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,1)}) \end{pmatrix}. \tag{4.4.24}$$

For the proof of the above representation, we can apply an argument like that already used in Section 3.3.4 (step 3) in the context of rigorous Fourier analysis. Starting with the stencil representation of I_{2h}^h, we can find a representation of $I_{2h}^h \varphi_{2h}(2\boldsymbol{\theta}^{(0,0)}, \cdot)$ in terms of $\varphi_h(\boldsymbol{\theta}^{(0,0)}, \cdot)$ similar to the one in (3.3.14). Here the coefficients $\delta_1, \delta_2, \delta_3, \delta_4$ correspond to the four types of grid points characterized by the indices (i, j) with $x_1 = ih$, $x_2 = jh$: even–even, odd–odd, odd–even, even–odd. We use the following remark, which corresponds directly to Remark 3.3.4.

Remark 4.4.2 Any grid function $c(x_1, x_2) e^{i\boldsymbol{\theta} \cdot \boldsymbol{x}/\boldsymbol{h}}$ with $c(x_1, x_2)$ as in (3.3.15) can be represented on $\mathbf{G}_h$ as a linear combination

$$a^{(0,0)} e^{i\boldsymbol{\theta}^{(0,0)} \cdot \boldsymbol{x}/\boldsymbol{h}} + a^{(1,1)} e^{i\boldsymbol{\theta}^{(1,1)} \cdot \boldsymbol{x}/\boldsymbol{h}} + a^{(1,0)} e^{i\boldsymbol{\theta}^{(1,0)} \cdot \boldsymbol{x}/\boldsymbol{h}} + a^{(0,1)} e^{i\boldsymbol{\theta}^{(0,1)} \cdot \boldsymbol{x}/\boldsymbol{h}},$$

where the coefficients $a^{(0,0)}$, $a^{(1,1)}$, $a^{(1,0)}$, $a^{(0,1)}$ are given by (3.3.16). ≫

Example 4.4.3 For the bilinear interpolation operator (2.3.8), we obtain (with $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)}$)

$$I_{2h}^h \varphi_{2h}(2\boldsymbol{\theta}, \boldsymbol{x}) = \varphi_{2h}(2\boldsymbol{\theta}, \boldsymbol{x}) \cdot \begin{cases} 1 & \text{if } x_1/h_1 \text{ and } x_2/h_2 \text{ even} \\ \cos\theta_1 \cos\theta_2 & \text{if } x_1/h_1 \text{ and } x_2/h_2 \text{ odd} \\ \cos\theta_1 & \text{if } x_1/h_1 \text{ odd}, x_2/h_2 \text{ even} \\ \cos\theta_2 & \text{if } x_1/h_1 \text{ even}, x_2/h_2 \text{ odd}. \end{cases} \tag{4.4.25}$$

The above remark gives

$$\begin{pmatrix} \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,0)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,1)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(1,0)}) \\ \tilde{I}_{2h}^h(\boldsymbol{\theta}^{(0,1)}) \end{pmatrix} = \frac{1}{4} \begin{pmatrix} (1+\cos\theta_1)(1+\cos\theta_2) \\ (1-\cos\theta_1)(1-\cos\theta_2) \\ (1+\cos\theta_1)(1-\cos\theta_2) \\ (1-\cos\theta_1)(1+\cos\theta_2) \end{pmatrix} \tag{4.4.26}$$

or simply

$$\tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \tfrac{1}{4}(1+\cos\bar{\theta}_1)(1+\cos\bar{\theta}_2),$$

which is the same as for the FW operator (up to a constant scaling factor). This also reflects the fact that bilinear interpolation and FW are transpose to each other. △

Remark 4.4.3 It can be shown that the symbol of linear interpolation in d dimensions can be written as the product

$$\tilde{I}_{2h}^h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \prod_{j=1}^{d} \frac{1+\cos\bar{\theta}_j}{2}.$$

If in d dimensions, $2^d I_h^{2h}$ is the transpose of some interpolation I_{2h}^h, we have $\hat{I}_h^{2h}(\boldsymbol{\theta}) = 2^{-d} \hat{I}_{2h}^h(\boldsymbol{\theta})^T$. ≫

From the representations of L_h, I_h^{2h}, I_{2h}^h, I_h and the solution on the coarse grid, we immediately obtain the representation of K_h^{2h}. In order to extend it to M_h^{2h}, we have to include the representations of S_h in the analysis.

Smoothing operator S_h This is particularly simple if the assumptions of the last section are fulfilled. In that case all Fourier components $\varphi(\boldsymbol{\theta}, .)$ are formal eigenfunctions of S_h, and S_h can be represented by the diagonal (4×4)-matrix

$$\begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \Leftarrow \begin{pmatrix} \tilde{S}_h(\boldsymbol{\theta}^{(0,0)}) & & & \\ & \tilde{S}_h(\boldsymbol{\theta}^{(1,1)}) & & \\ & & \tilde{S}_h(\boldsymbol{\theta}^{(1,0)}) & \\ & & & \tilde{S}_h(\boldsymbol{\theta}^{(0,1)}) \end{pmatrix} \begin{pmatrix} A^{(0,0)} \\ A^{(1,1)} \\ A^{(1,0)} \\ A^{(0,1)} \end{pmatrix} \tag{4.4.27}$$

for each $\boldsymbol{\theta} \in T^{\text{low}}$. Under the more general assumption (4.4.9), S_h is represented on E_h^{θ} by a (4×4)-matrix $\hat{S}_h(\boldsymbol{\theta})$.

4.5 SMOOTHING ANALYSIS II

In Section 4.3, we considered smoothing procedures which have the property that all $\varphi(\boldsymbol{\theta}, \cdot)$ are eigenfunctions of the smoothing operators S_h. For such smoothers, the definition of a smoothing factor $\mu_{\text{loc}}(S_h)$ is straightforward. In the two-grid analysis in the last section, we made the more general assumption that only the *invariance property*

$$\boxed{S_h : E_h^{\theta} \to E_h^{\theta} \quad \text{for all } \boldsymbol{\theta} \in T^{\text{low}}} \tag{4.5.1}$$

is valid for the smoothing operator under consideration. This assumption is fulfilled for a larger class of smoothers, including GS-RB relaxation as well as "multicolor" and "zebra"-type relaxations which will be described in the following chapters.

In the following, we will generalize the definition of the smoothing factor for all smoothers which have the invariance property (4.5.1).

We will describe a concrete class of smoothing procedures including red–black type smoothers, based on a straightforward generalization of the splitting formalism in Section 4.3.

If a smoother has the invariance property (4.5.1), high and low frequencies may be intermixed by S_h. In order to be able to measure the smoothing properties of S_h, the idea is to assume *an "ideal" coarse grid operator* Q_h^{2h} (instead of the real coarse grid operator K_h^{2h}), *which annihilates the low frequency error components and leaves the high frequency components unchanged.* More precisely, Q_h^{2h} is a projection operator, defined on E_h^{θ} by

$$Q_h^{2h}\varphi(\boldsymbol{\theta},\cdot) = \begin{cases} 0 & \text{if } \boldsymbol{\theta} = \boldsymbol{\theta}^{(0,0)} \in T^{\text{low}} \\ \varphi(\boldsymbol{\theta},\cdot) & \text{if } \boldsymbol{\theta} \in \{\boldsymbol{\theta}^{(1,0)}, \boldsymbol{\theta}^{(0,1)}, \boldsymbol{\theta}^{(1,1)}\}. \end{cases} \tag{4.5.2}$$

As a consequence, the (4×4)-matrix $\hat{K}_h^{2h}(\boldsymbol{\theta})$ in (4.4.7) is replaced by the (4×4)-projection matrix

$$\hat{Q}_h^{2h}(\boldsymbol{\theta}) = \hat{Q}_h^{2h} = \begin{pmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \quad \text{for} \quad \boldsymbol{\theta} \in T^{\text{low}} \tag{4.5.3}$$

for the smoothing analysis. Furthermore, we replace

$$M_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1} \quad \text{by} \quad S_h^{\nu_2} Q_h^{2h} S_h^{\nu_1}$$

and $\rho_{\text{loc}}(M_h^{2h})$ in (4.4.14) by

$$\rho_{\text{loc}}(S_h^{\nu_2} Q_h^{2h} S_h^{\nu_1}) := \sup\left\{\rho(\hat{S}_h(\boldsymbol{\theta})^{\nu_2}\, \hat{Q}_h^{2h} \hat{S}_h(\boldsymbol{\theta})^{\nu_1}) \colon \boldsymbol{\theta} \in T^{\text{low}}\right\}. \tag{4.5.4}$$

This quantity $\rho_{\text{loc}}(S_h^{\nu_2} Q_h^{2h} S_h^{\nu_1})$ is a good measure of the total smoothing effect resulting from $\nu = \nu_1 + \nu_2$ smoothing steps. In addition, one can expect that this quantity gives a realistic prediction of the convergence factor $\rho_{\text{loc}}(M_h^{2h})$ as long as the replacement of K_h^{2h} by Q_h^{2h} is acceptable. Practically this means that the transfer and the coarse grid operator should be sufficiently good and that not too many smoothing steps are carried out per two-grid cycle. Since

$$\rho_{\text{loc}}(\hat{S}_h(\boldsymbol{\theta})^{\nu_2} \hat{Q}_h^{2h} \hat{S}_h(\boldsymbol{\theta})^{\nu_1}) = \rho_{\text{loc}}(\hat{Q}_h^{2h} \hat{S}_h(\boldsymbol{\theta})^{\nu}) \quad (\text{with } \nu = \nu_1 + \nu_2),$$

we arrive at the following definition of a general LFA smoothing factor.

Definition 4.5.1 Under the assumption that S_h has the invariance property (4.5.1), we define the *smoothing factor* $\mu_{\text{loc}}(S_h, \nu)$ of S_h by

$$\mu_{\text{loc}}(S_h, \nu) := \sup\left\{\sqrt[\nu]{\rho_{\text{loc}}(\hat{Q}_h^{2h} \hat{S}_h(\boldsymbol{\theta})^{\nu})},\ \boldsymbol{\theta} \in T^{\text{low}}\right\}. \tag{4.5.5}$$

Here, μ depends on ν!

Remark 4.5.1 For the practical calculation of $\mu_{\text{loc}}(S_h, \nu)$, note that the first row of the (4×4)-matrix $Q_h^{2h} S_h^{\nu}$ is zero. ≫

Remark 4.5.2 For the special case that all $\varphi(\boldsymbol{\theta},\cdot)$ are eigenfunctions of S_h, the above definition coincides with Definition 4.3.1 given in Section 4.3. In that case $\hat{S}_h(\boldsymbol{\theta})^{\nu}$ is the diagonal matrix in (4.4.27) and $\hat{Q}_h^{2h} \hat{S}_h(\boldsymbol{\theta})^{\nu}$ contains only the high frequency symbols $\tilde{S}_h(\boldsymbol{\theta}^{(1,0)})^{\nu}$, $\tilde{S}_h(\boldsymbol{\theta}^{(0,1)})^{\nu}$, $\tilde{S}_h(\boldsymbol{\theta}^{(1,1)})^{\nu}$. ≫

4.5.1 Local Fourier Analysis for GS-RB

We now apply the above general definition of a smoothing factor. For that purpose, we first generalize the assumption on smoothing procedures in Section 4.3 in such a way that red–black type smoothers and similar "pattern" smoothers, like four-color relaxation, are included.

A corresponding smoothing operator S_h^{partial} is then given by

$$S_h^{\text{partial}} v_h(\boldsymbol{x}) = \begin{cases} -(L_h^+)^{-1} L_h^- v_h(\boldsymbol{x}) & \text{for } \boldsymbol{x} \in \tilde{\mathbf{G}}_h \\ v_h(\boldsymbol{x}) & \text{for } \boldsymbol{x} \in \mathbf{G}_h \backslash \tilde{\mathbf{G}}_h \end{cases} \tag{4.5.6}$$

(assuming that $(L_h^+)^{-1}$ exists). Here, $\tilde{\mathbf{G}}_h$ is a subset of $\mathbf{G}_h$, typically defined by a pattern that is characteristic for the smoothing procedure under consideration. Obviously, only the grid points of $\tilde{\mathbf{G}}_h$ are processed in the above partial relaxation step; the remaining points $(\mathbf{G}_h \backslash \tilde{\mathbf{G}}_h)$ are not treated.

A typical example for such a smoothing operator is the GS-RB smoother for $L_h = -\Delta_h$. Using the above notation, GS-RB consists of two partial steps, each of which is of Jacobi type. In the first partial step, $\tilde{\mathbf{G}}_h$ consists of the "red" and in the second step $\tilde{\mathbf{G}}_h$ consists of the "black" points of $\mathbf{G}_h$. For both steps, we now use (4.3.4) with a relaxation parameter ω.

The partial step smoothing operators, denoted by S_h^{RED} and S_h^{BLACK}, respectively, then both have the form (4.5.6), with $\tilde{\mathbf{G}}_h$ being the red/black points of $\mathbf{G}_h$, respectively. The complete GS-RB smoothing operator is, of course, the product

$$S_h^{\text{RB}} = S_h^{\text{BLACK}} S_h^{\text{RED}}. \tag{4.5.7}$$

In order to calculate the smoothing factor (4.5.5) of S_h^{RB}, we must find the (4×4)-matrix representation $\hat{S}_h^{\text{RB}}(\omega, \boldsymbol{\theta})$ for $S_h^{\text{RB}}(\omega)$. Since we have

$$S_h(\omega) = -(L_h^+)^{-1} L_h^- = I_h - \frac{\omega h^2}{4} L_h$$

for ω-JAC (4.3.4), we find from (4.5.6) that

$$S_h^{\text{partial}} \varphi(\boldsymbol{x}) = \begin{cases} (1 - (\omega h^2/4)\tilde{L}_h(\boldsymbol{\theta}))\varphi(\boldsymbol{\theta}, \boldsymbol{x}) & (\boldsymbol{x} \in \tilde{\mathbf{G}}_h) \\ \varphi(\boldsymbol{\theta}, \boldsymbol{x}) & (\boldsymbol{x} \in \mathbf{G}_h \backslash \tilde{\mathbf{G}}_h). \end{cases} \tag{4.5.8}$$

This is not yet a Fourier representation of S_h^{partial}. Note, however, that any $\psi \in E_h^\theta$

$$\psi(\boldsymbol{\theta}, \boldsymbol{x}) = \begin{cases} \beta_1 \varphi(\boldsymbol{\theta}, \boldsymbol{x}) & (\boldsymbol{x} \in \tilde{\mathbf{G}}_h) \\ \beta_2 \varphi(\boldsymbol{\theta}, \boldsymbol{x}) & (\boldsymbol{x} \in \mathbf{G}_h \backslash \tilde{\mathbf{G}}_h) \end{cases} \tag{4.5.9}$$

(with a fixed Fourier component φ and constants β_1 and β_2) can easily be written as a linear combination of $\varphi(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ with $\boldsymbol{\theta}^{\boldsymbol{\alpha}} = \boldsymbol{\theta}$ and $\varphi(\boldsymbol{\theta}^{\bar{\boldsymbol{\alpha}}}, \cdot)$, where $\bar{\boldsymbol{\alpha}} = (1, 1) - \boldsymbol{\alpha}$. This is actually a special case of Remark 4.4.2. We obtain the following (4×4)-matrix representations for S_h^{RED} and S_h^{BLACK} with respect to $E_h^\theta(-\pi/2 \le \boldsymbol{\theta} < \pi/2)$:

$$\hat{S}_h^{\text{RED}}(\boldsymbol{\theta}, \omega) = \frac{1}{2} \begin{pmatrix} A^{(0,0)} + 1 & A^{(1,1)} - 1 & 0 & 0 \\ A^{(0,0)} - 1 & A^{(1,1)} + 1 & 0 & 0 \\ 0 & 0 & A^{(1,0)} + 1 & A^{(0,1)} - 1 \\ 0 & 0 & A^{(1,0)} - 1 & A^{(0,1)} + 1 \end{pmatrix}$$

$$\hat{S}_h^{\text{BLACK}}(\boldsymbol{\theta}, \omega) = \frac{1}{2} \begin{pmatrix} A^{(0,0)} + 1 & -A^{(1,1)} + 1 & 0 & 0 \\ -A^{(0,0)} + 1 & A^{(1,1)} + 1 & 0 & 0 \\ 0 & 0 & A^{(1,0)} + 1 & -A^{(0,1)} + 1 \\ 0 & 0 & -A^{(1,0)} + 1 & A^{(0,1)} + 1 \end{pmatrix}$$

with

$$A^{\boldsymbol{\alpha}} := 1 - \frac{\omega h^2}{4} \tilde{L}_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}}).$$

From this representation, one can compute the smoothing factor $\mu_{\text{loc}}(S_h^{\text{RB}}(\omega), \nu)$. For the standard value $\omega = 1$, we obtain

$$\mu_{\text{loc}}(\nu) = \max\left\{\tfrac{1}{4}, \chi(\nu)\right\} \tag{4.5.10}$$

(see [378]), where

$$\chi(\nu) = \left(\frac{2\nu - 1}{2\nu}\right)^2 \Big/ \sqrt[\nu]{2(2\nu - 1)}. \tag{4.5.11}$$

Some particular values of $\chi(\nu)$ are $\chi(1) = 0.125$, $\chi(2) = 0.229$ and $\chi(3) = 0.322$. $\chi(\nu) \sim \sqrt[\nu]{1/(4e\nu)}$ for $\nu \to \infty$.

Remark 4.5.3 The smoothing analysis of ω-GS-RB can also be carried out in the context of the rigorous Fourier analysis (see Section 7.5 of [378]). The representation of the (4×4)-block matrices is the same as in the LFA case. ≫

4.6 SOME RESULTS, REMARKS AND EXTENSIONS

In this section we will give some additional LFA results. More results can be found in [58, 378]. We also outline straightforward extensions of LFA with respect to other coarsening strategies. Furthermore, a simplified two-grid analysis is presented, which gives a basic insight into the quality of the coarse grid correction.

4.6.1 Some Local Fourier Analysis Results for Model Problem 1

We consider $L_h = -\Delta_h$ and a multigrid method based on GS-LEX smoothing, FW and bilinear interpolation. LFA two-grid analysis gives the two-grid convergence factors ρ_{loc} as listed in Table 4.1. For comparison, experimentally measured convergence factors for Model Problem 1 are included (W-cycle, $h = 1/128$). The correspondence of theoretical and practical values is excellent and typical for many applications.

We also recognize a good prediction of the two-grid convergence factors by the smoothing analysis. This good agreement can be expected only for small numbers ν of smoothing steps. If too many smoothing steps are performed per multigrid cycle, the coarse grid approximation can no longer cope with the smoothing effect. Therefore, the smoothing factors μ_{loc}^ν are too optimistic for $\nu \ge 4$ in this case.

Table 4.2 compares the influence of the restriction operator on the convergence factors ρ_{loc} for GS-LEX, comparing full weighting (FW) and injection (INJ). The results show that INJ is a satisfactory restriction operator *if combined with GS-LEX* for Model Problem 1. (This is particularly true if the computational effort is taken into account.) INJ is cheaper than FW. Intuitively, it seems to be clear that INJ is the better, the smoother the defects are. This is reflected by the convergence factors for $\nu \ge 3$, where $\rho_{\text{loc}}(\text{INJ})$ is even smaller than $\rho_{\text{loc}}(\text{FW})$.

The application of the INJ operator has, however, the disadvantage that the spectral norm of the corresponding two-grid operators is not bounded, see Table 4.3, where also the influence of the pre- and postsmoothing on the norm $\sigma_{\text{loc},S}$ is presented.

Table 4.1. LFA smoothing factors, LFA two-grid convergence factors and measured W-cycle convergence factors (Model Problem 1).

ν_1, ν_2	$\mu_{\text{loc}}^{\nu_1+\nu_2}$	ρ_{loc}	W-cycle ($h = 1/128$)
1,0	0.500	0.400	0.40
1,1	0.250	0.193	0.19
2,1	0.125	0.119	0.12
2,2	0.063	0.084	0.08

Table 4.2. ρ_{loc} for GS-LEX relaxation (for $L_h = -\Delta_h$).

ν	$\rho_{\text{loc}}(\text{FW})$	$\rho_{\text{loc}}(\text{INJ})$
1	0.400	0.447
2	0.193	0.200
3	0.119	0.089
4	0.084	0.042

Table 4.3. $\sigma_{\text{loc},S}$ corresponding to the methods in Table 4.2.

(ν_1, ν_2)	I_h^{2h} (FW)	I_h^{2h} (INJ)
(1,0)	0.447	∞
(0,1)	1.000	∞
(2,0)	0.208	∞
(1,1)	0.203	∞
(0,2)	1.000	∞
(3,0)	0.128	∞
(2,1)	0.119	∞
(1,2)	0.131	∞
(0,3)	1.000	∞

One thus needs to be careful with this operator, in particular if it is used in FMG or, more generally, if only one or a small number of cycles is used. Remember that injection is a restriction of order 0 (see Section 2.7). Again, FW is more robust and reliable.

All the values given for ρ_{loc} above can be confirmed by numerical measurements. In all cases, the measured convergence factors are close to ρ_{loc} if h is chosen to be small enough (here for instance $h = 1/128$).

In the following chapters, we will use LFA for many different operators and multigrid approaches, 3D cases, singularly perturbed problems, operators with mixed derivatives, systems of PDEs and so on. In most cases, the LFA results will be very helpful in designing multigrid algorithms and understanding difficulties. Of course, LFA also has its limitations. If the local view which is characteristic for LFA is not sufficient for the description of certain global phenomena (e.g. singularly perturbed or hyperbolic situations), other considerations have to be added.

Brandt has proposed and developed extensions and generalizations of LFA. One extension is the so-called half-space analysis [63], which allows the inclusion of boundary effects into the analysis.

4.6.2 Additional Remarks

In this section we add some remarks about LFA approaches, modifications of LFA and about the relationship between LFA and rigorous Fourier analysis.

LFA smoothing analysis is usually the first step in analyzing a given problem and its multigrid solution. Smoothing analysis is typically much easier than two-grid analysis. This is particularly true if the $e^{i\boldsymbol{\theta}\cdot\boldsymbol{x}/h}$ functions are eigenfunctions of S_h as assumed in Section 4.3. But even if only the spaces E_h^{θ} are invariant under S_h, the smoothing analysis is simpler since it uses the "ideal" coarse grid operator given in Section 4.5.

Remark 4.6.1 (coarse grid correction and simplified two-grid analysis) In order to avoid the complete two-grid analysis which may become rather involved, in particular in 3D cases and for systems of PDEs, one can analyze the smoothing procedure and the coarse grid correction separately. Therefore, in addition to smoothing analysis, "coarse grid correction analysis" approaches have been proposed. One idea in this context is the so-called *simplified two-grid analysis*, which is based on the *first differential approximation* (FDA) [426]. The goal here is to obtain some insight into the quality of the approximation of the operator L_h by L_{2h} for very low frequencies. The analysis neglects high frequencies and the coupling of harmonics. Furthermore, for very low frequencies, the transfer operators act nearly like identities. Therefore, the behavior of K_h^{2h} can be approximately described by the quantity

$$\tilde{I}_h - (\tilde{L}_{2h})^{-1}\tilde{L}_h \quad \text{with} \quad \tilde{I}_h = 1$$

(for very low frequencies). The analysis of this term gives some insight into the quality of the coarse grid correction especially for problems with *characteristic directions* (like convection–diffusion equations, see Chapter 7). If we consider a low frequency $\boldsymbol{\theta} = (\theta_1, \theta_2)$ along a characteristic direction with $\theta_2 = c\theta_1$, we can compute

$$\lim_{\theta_1 \to 0} \left(1 - \frac{\tilde{L}_h(\boldsymbol{\theta})}{\tilde{L}_{2h}(\boldsymbol{2\theta})}\right).$$

This estimate of the coarse grid approximation for the lowest frequencies should be a very small number. If it is not, the multigrid performance will be negatively influenced by a bad coarse grid correction due to very low frequencies (see Section 7.2.3 for an example). ≫

Remark 4.6.2 Some authors use the LFA approach without allowing $\boldsymbol{\theta}$ to vary continuously in $-\pi \leq \boldsymbol{\theta} < \pi$. In this case, $\boldsymbol{\theta}$ varies in the finite set I_h as introduced in (3.4.8). Such approaches can be useful if the real convergence factors are substantially dependent on h. ≫

Remark 4.6.3 Table 4.4 illustrates which of the analysis approaches can be used to analyze the three smoothers ω-JAC, GS-LEX and GS-RB. ≫

Remark 4.6.4 As we have pointed out before, GS-LEX is a smoothing procedure which can be analyzed by LFA, not, however, by the rigorous Fourier analysis presented in Section 3.3. The reason is that the GS-LEX smoothing operator S_h for the infinite grid is different from the GS-LEX smoothing operator for periodic boundary conditions on a

Table 4.4. Which analysis can be applied to which smoothing scheme?

	Rigorous Fourier analysis	LFA
$\varphi(\boldsymbol{\theta}, \cdot)$ eigenfunctions of S_h	ω-JAC	ω-JAC GS-LEX
$E_{h,\boldsymbol{\theta}}$ invariant under S_h	ω-JAC	ω-JAC GS-LEX
	GS-RB	GS-RB

finite domain. On an infinite grid there is no natural starting point for GS-LEX, contrary to the finite grid situation. If one tried to mimic the "infinite grid GS-LEX" in a finite domain situation with periodic boundary conditions, an *implicit* smoothing scheme would result. ≫

Remark 4.6.5 (how to use LFA in practice) The two-grid LFA is rather complicated at the first glance. It is usually carried out with *a computer program* (see `http://www.gmd.de/SCAI/multigrid/book.html` for such a program by R. Wienands). An even simpler program can be used for the LFA smoothing analysis. Here, we present some general guidelines for the use of LFA:

If one wants to design (or analyze) a multigrid algorithm for a concrete problem $Lu = f$, we recommend using the following LFA ladder:

(1) Find an appropriate discretization for the problem, e.g. with good "h-ellipticity" $E_h(L_h)$ (to be discussed in Section 4.7).
(2) Find a smoothing scheme with a satisfactory smoothing factor.
(3) Choose appropriate transfer operators and check whether the two-grid LFA convergence factors are close to the smoothing factors.
(4) Check whether the convergence of multigrid cycles in your program approximates the LFA prediction.
(5) Use FMG and check whether you obtain discretization accuracy.

Of course, one could often skip some of these steps, e.g. check directly whether the multigrid program (Step 4) produces the efficiency anticipated by the smoothing factors (Step 2). If not, *then* perform the intermediate steps. ≫

Remark 4.6.6 (debugging) In the debugging phase, it is advisable to use as coarse grids as possible.

If a multigrid code does not show a good convergence for a particular problem, a useful debugging approach is to start with an example, for which the exact discrete solution is known. This is easy if the example can be chosen such that the discretization error is 0, e.g. $u(x, y) = x + 2y$ in the case of the standard Laplacian.

A discrete approximation from the algorithm can then be compared with the exact solution of the discrete problem. One can check, whether the error between the approximation and the reference solution is smooth after the relaxation. Maybe it is not smooth near a boundary (see Section 5.6 for the boundary treatment). Also the defect can give some insight. It might be large in only a few grid points, for example. This debugging can be carried out graphically, e.g. by plotting the defects or the errors with available tools such as "gnuplot".

If a problem without a discretization error is not known, one can solve the problem first (possibly with a single grid solver), store the obtained solution and compare the multigrid approximations with this discrete solution. Additional insight is gained by printing norms of the defect on all grid levels, for example, after presmoothing and after the coarse grid correction and postsmoothing. These defects should also decrease on coarser grids. ≫

4.6.3 Other Coarsening Strategies

In introducing LFA in this chapter, we have assumed standard coarsening, $\boldsymbol{H} = (2h_1, 2h_2)$. LFA can be extended to other coarsening strategies in a straightforward manner. *What has to be changed is the definition of low and high frequencies.* As a consequence, the definitions of spaces of harmonics E_h^φ and of the central quantities $\mu_{\text{loc}}(S_h)$, $\rho_{\text{loc}}(M_h^H)$ and $\sigma_{\text{loc}}(M_h^H)$ have to be adapted.

In Fig. 4.3, we consider the following coarsening strategies:

(1) x-semicoarsening ($\boldsymbol{H} = (2h_1, h_2)$, upper left picture in Fig. 4.3),
(2) y-semicoarsening ($\boldsymbol{H} = (h_1, 2h_2)$, upper right picture in Fig. 4.3),
(3) red–black-coarsening (lower left picture in Fig. 4.3),
(4) $(h, 4h)$-coarsening (lower right picture in Fig. 4.3).

Instead of formally defining the respective low and high frequency terms, we simply show in Fig. 4.3 how Fig. 4.1 is to be changed for these four cases.

Note that the spaces of harmonics are two-dimensional in the first three coarsening strategies listed above, but 16-dimensional for $(h, 4h)$-coarsening. These facts have obvious and straightforward implications for the definition of $\mu_{\text{loc}}(S_h)$, $\rho_{\text{loc}}(M_h^H)$ and $\sigma_{\text{loc}}(M_h^H)$.

Remark 4.6.7 If GS-RB is used in connection with red–black-coarsening, the two-dimensional spaces of harmonics (characterized in Fig. 4.3) are invariant under S_h and K_h^H and thus M_h^H.

If GS-RB is used, however, in connection with x- or y-semicoarsening, the corresponding two-dimensional spaces that are invariant under S_h and under K_h^H are not the same. In this case, the four-dimensional "standard" spaces E_h^φ in (4.4.5) have to be used again for the two-grid LFA. They are again invariant under S_h and K_h^H. ≫

4.7 h-ELLIPTICITY

In Section 1.1, we gave a first impression of the properties a PDE should fulfill for "standard" multigrid to work properly. The *ellipticity* of a PDE was explicitly addressed in this context. In describing basic multigrid algorithms in Chapter 2, we considered discrete elliptic differential operators L_h, or more concretely, Poisson-like equations in 2D or 3D. Finally, all the concrete theoretical results in Chapter 3 were related to the discrete Poisson equation.

Ellipticity of a differential operator L is, however, neither a necessary nor a sufficient condition for the applicability of multigrid. In the following chapters we will also consider differential operators which may lack ellipticity or become nonelliptic depending on some characteristic parameters: anisotropic operators, convection-dominated operators, the full potential equation, Euler equations and Navier–Stokes equations.

In this section, we will use the LFA terminology to (qualitatively) characterize the properties a discrete operator L_h should have to be suitable for multigrid. For that purpose, we will introduce the concept of *h-ellipticity* proposed by Brandt [61]. This property is

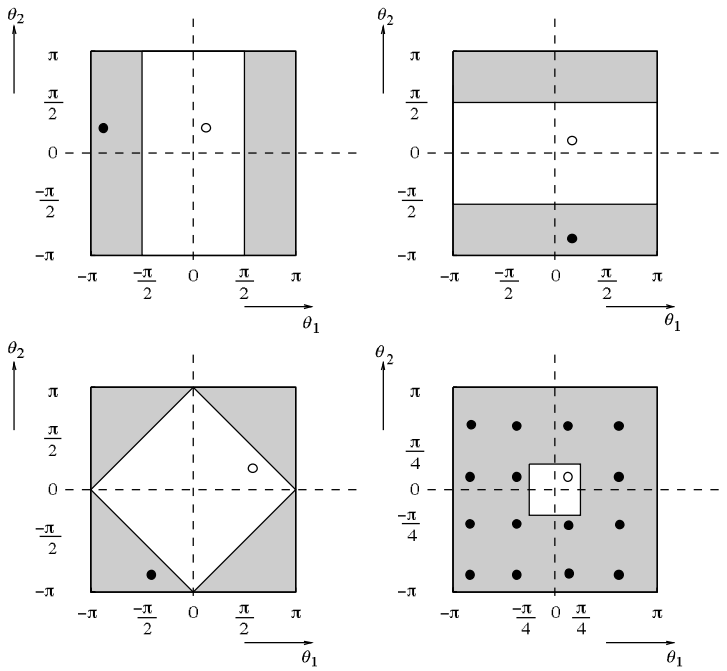


Figure 4.3. Low frequencies (interior white regions) and high frequencies (shaded regions) for various coarsening strategies; for a given low frequency $\boldsymbol{\theta}$(○), the other frequencies $\boldsymbol{\theta}$ for which the corresponding $\varphi(\boldsymbol{\theta}, \boldsymbol{x})$ coincide on $\mathbf{G}_H$ are marked by •.

defined for operators L_h given on an infinite grid $\mathbf{G}_h$. Throughout this section, we make the same assumptions on L_h as for the LFA.

> The central result of this section is that, in a certain sense, the h-ellipticity of L_h is a necessary and sufficient condition for the existence of pointwise smoothing procedures for L_h.

If a discrete operator lacks h-ellipticity, the addition of some "artificial ellipticity" (or "discrete ellipticity") is one way to make L_h sufficiently h-elliptic. The idea of introducing *artificial viscosity* is a classical approach in the case of convection-dominated problems.

5.1 ANISOTROPIC EQUATIONS IN 2D

As a first step away from Poisson-like elliptic equations, we consider anisotropic elliptic problems.

Model Problem 3 *(2D anisotropic model problem)*

$$\begin{aligned} -\varepsilon u_{xx} - u_{yy} &= f^{\Omega}(x, y) \quad (\Omega = (0,1)^2) \\ u &= f^{\Gamma}(x, y) \quad (\partial\Omega) \end{aligned} \tag{5.1.1}$$

with $0 < \varepsilon \ll 1$. *(The case* $\varepsilon \gg 1$ *is similar, with interchanged roles of* x *and* y*.)*

If we discretize Model Problem 3 by the standard five-point difference operator, we obtain the discrete problem

$$\begin{aligned} L_h(\varepsilon) u_h(x, y) &= f_h^{\Omega}(x, y) \quad (\Omega_h) \\ u_h(x, y) &= f_h^{\Gamma}(x, y) \quad (\Gamma_h), \end{aligned} \tag{5.1.2}$$

where $\Omega_h = \mathbf{G}_h \cap \Omega$ is the square grid (1.3.3) with $h = h_x = h_y$, Γ_h is again the set of discrete intersection points of the grid lines with the boundary Γ and

$$L_h(\varepsilon) = \frac{1}{h^2} \begin{bmatrix} & -1 & \\ -\varepsilon & 2(1+\varepsilon) & -\varepsilon \\ & -1 & \end{bmatrix}_h . \tag{5.1.3}$$

Here, we restrict ourselves to the case where the discrete anisotropy is "aligned" with the grid. In 2D such problems are characterized by coefficients in front of the u_{xx} and u_{yy} terms, which may differ by orders of magnitude. Anisotropic problems play an important role in practice. The discretization may also introduce (discrete) anisotropies, in the form of stretched grids.

Remark 5.1.1 (stretched grids) The same discrete operator (5.1.3) is obtained, if we discretize the Laplace operator (in Model Problem 1) by the standard five-point difference operator on a *stretched* grid with mesh sizes $h_x = h_y/\sqrt{\varepsilon}$. ≫

5.1.1 Failure of Pointwise Relaxation and Standard Coarsening

In Example 4.7.7, we have seen that the h-ellipticity measure of an anisotropic operator tends to 0 for $\varepsilon \to 0$:

$$E_h(L_h(\varepsilon)) = O(\varepsilon) \longrightarrow 0 \quad \text{for } \varepsilon \to 0.$$

According to the discussion in Section 4.7.2, it is expected that the smoothing properties of standard pointwise smoothing schemes will deteriorate for $\varepsilon \to 0$.

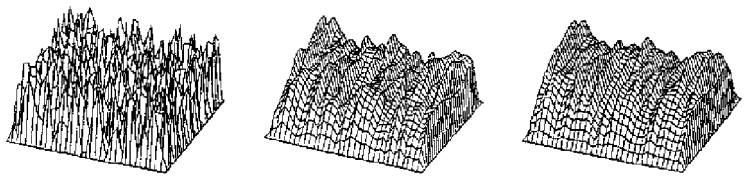


Figure 5.1. Influence of (pointwise) GS-LEX on the error for the 2D anisotropic model problem with $\varepsilon \ll 1$.

In particular, highly anisotropic problems cannot be treated efficiently with standard coarsening and the multigrid components introduced so far. Therefore, they are a first and a very illustrative example of the fact that multigrid, in general, has to be adapted and tuned to the problem at hand. However, the proper ways to treat anisotropic problems by multigrid are fully understood and can easily be explained.

If we apply a standard pointwise relaxation such as ω-JAC, GS-LEX or GS-RB to the above discrete system, we find that the smoothing effect of this relaxation is very poor with respect to the x-direction. The reason is that pointwise relaxation has a smoothing effect only with respect to the "strong coupling" in the operator, i.e. in the y-direction. This effect is illustrated in Fig. 5.1 for $\varepsilon = 10^{-2}$.

If we consider GS-LEX, for example, we find the error relation

$$\bar{v}_h(x_i, y_j) = \frac{1}{2(\varepsilon+1)}[\varepsilon \bar{v}_h(x_{i-1}, y_j) + \varepsilon v_h(x_{i+1}, y_j) + \bar{v}_h(x_i, y_{j-1}) + v_h(x_i, y_{j+1})], \tag{5.1.4}$$

which for $\varepsilon \to 0$ becomes

$$\bar{v}_h(x_i, y_j) = \tfrac{1}{2}[\bar{v}_h(x_i, y_{j-1}) + v_h(x_i, y_{j+1})]. \tag{5.1.5}$$

Obviously, there is no averaging effect with respect to the x-direction and, hence, no smoothing with respect to this direction is achieved. (For $\varepsilon \gg 1$, it is the other way round.) Such nonsmooth errors can no longer be efficiently reduced by means of a coarser grid which is obtained by standard coarsening, i.e. by doubling the mesh size in *both* directions.

This failure can be directly explained by applying LFA smoothing analysis as in Section 4.3 to the GS-LEX pointwise smoother for Model Problem 3. By adapting the

Table 5.1. Two-grid convergence factors ρ_{loc} for the anisotropic model problem using GS-RB (ν = three smoothing steps), FW and bilinear interpolation for different values of ε.

ε	0.001	0.01	0.1	0.5	1	2	10	100	1000
ρ_{loc}	0.99	0.94	0.56	0.088	0.053	0.088	0.56	0.94	0.99

splitting from (4.3.3) to (5.1.3) we obtain

$$L_h^- = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ 0 & 0 & \varepsilon \\ & 0 & \end{bmatrix}, \qquad L_h^+ = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ -\varepsilon & 2+2\varepsilon & 0 \\ & -1 & \end{bmatrix}. \tag{5.1.6}$$

Following the procedure in Section 4.3, the smoothing factor μ_{loc} of GS-LEX for (5.1.3) is found to be

$$\mu_{\text{loc}} = \sup_{\theta \in T^{\text{high}}} |\tilde{S}_h(\theta)| = \sup_{\theta \in T^{\text{high}}} \left| \frac{\varepsilon e^{i\theta_1} + e^{i\theta_2}}{\varepsilon e^{-i\theta_1} + e^{-i\theta_2} - 2 - 2\varepsilon} \right|. \tag{5.1.7}$$

For $\varepsilon \to 0$ (and for $\varepsilon \to \infty$), we obtain

$$\lim_{\varepsilon \to 0} \mu_{\text{loc}} = \lim_{\varepsilon \to 0} \tilde{S}_h(\pi, 0) = 1.$$

Table 5.1 presents corresponding two-grid convergence factors $\rho_{\text{loc}} = \rho_{\text{loc}}(\nu)$ for GS-RB pointwise smoothing, standard coarsening, FW and linear interpolation. For large or small values of ε, standard pointwise smoothers fail to achieve satisfactory two-grid (and thus also multigrid) convergence.

Pointwise relaxation and standard coarsening is not a reasonable combination for highly anisotropic problems. The multigrid convergence factor will increase towards 1 *for* $\varepsilon \to 0$ *or* $\varepsilon \to \infty$.

Remark 5.1.2 It is to some extent possible to keep the GS-RB pointwise smoother for moderate anisotropies and improve the two-grid factors from Table 5.1 *by overrelaxation.* This has been shown in [427], where analytic formulas are presented for optimal relaxation parameters ω. For $\varepsilon = 0.1$, for example, $\omega_{\text{opt}} = 1.41$ leads to $\rho_{\text{loc}} = 0.12$. For $\varepsilon = 0.01$, $\omega_{\text{opt}} = 1.76$ results in $\rho_{\text{loc}} = 0.45$, which is a major improvement compared to the results presented in Table 5.1. For $\varepsilon = 0.001$, $\omega_{\text{opt}} = 1.92$ and $\rho_{\text{loc}} = 0.78$. More robust multigrid remedies to master the difficulty of highly anisotropic problems are presented in the subsequent subsections. ≫

5.1.2 Semicoarsening

The first possibility is to keep pointwise relaxation for smoothing, but to change the grid coarsening according to the one-dimensional smoothness of errors. The coarse grid is

defined by doubling the mesh size only in that direction in which the errors are smooth. Semicoarsening in the y-direction, as introduced in Section 2.3.1, is appropriate if $\varepsilon \ll 1$. (The 1D restriction operator for semicoarsening has been introduced, for example, in Remark 2.3.2.)

In the case of semicoarsening, LFA can also be applied. For example, we can use (5.1.7) for GS-LEX and y-semicoarsening. But the range of high frequencies is changed now to

$$\left\{(\theta_1, \theta_2): \theta_2 \in [-\pi, \pi) \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)\right\}$$

(see Section 4.6.3). Maximizing over this range, we find

$$\mu_{\text{loc}} = \frac{1+\varepsilon}{\sqrt{5+\varepsilon}}.$$

For $\varepsilon \to 0$, we obtain the satisfactory value $\mu_{\text{loc}} \to 1/\sqrt{5} \approx 0.45$. This shows that the quality of a smoother depends on the range of high frequencies and thus on the choice of the coarse grid. (For $\varepsilon \gg 1$, μ_{loc} rapidly increases towards 1. In this case GS-LEX provides good smoothing if *x-semicoarsening* is employed.)

The operator L_H on the coarse grid Ω_H is

$$L_H(\varepsilon) = \frac{1}{H^2}\begin{bmatrix} & -1 & \\ -4\varepsilon & 2(1+4\varepsilon) & -4\varepsilon \\ & -1 & \end{bmatrix}_H \tag{5.1.8}$$

in the case of y-semicoarsening. Compared to the fine grid operator (5.1.4), the anisotropy has decreased. If we continue the y-semicoarsening process, the anisotropy will decrease further.

Remark 5.1.3 For *isotropic* problems such as Model Problem 1 on a standard grid with $h = h_x = h_y$, repeated semicoarsening in the same direction will not give satisfactory convergence results if pointwise smoothers are used. This occurs because a discrete anisotropy that arises from cells that have been stretched too much will be built up by the semicoarsening process (see also Remark 5.1.1). ≫

5.1.3 Line Smoothers

An alternative multigrid approach for highly anisotropic problems is to keep the standard multigrid coarsening, but to change the relaxation procedure from pointwise relaxation to *linewise* relaxation (called line relaxation in short). That is, *all unknowns on a line are updated collectively* (simultaneously). Line relaxations are thus block iterations in which each block of unknowns corresponds to a line. For Model Problem 3, the collective solution of all equations corresponding to a line means the solution of a *tridiagonal system of equations*.

Figure 5.2 shows the order in which the unknowns are relaxed for *lexicographic* x- and y-line Gauss–Seidel relaxation.

Figure 5.2. Order in which unknowns are solved for collectively by lexicographic line Gauss–Seidel relaxation; (a) x-line Gauss–Seidel, (b) y-line Gauss–Seidel relaxation.

Gauss–Seidel-type line relaxations are particularly efficient smoothers for anisotropic problems (if the anisotropy is aligned with the grid). This is due to the general observation that *errors become smooth if strongly connected unknowns are updated collectively.*

The good smoothing properties of (lexicographic) line Gauss–Seidel can also be seen by LFA. Using the same notation as in Section 4.3, the lexicographic y-line Gauss–Seidel for (5.1.3) can be written as

$$\frac{1}{h^2}\begin{bmatrix} -1 \\ 2(1+\varepsilon) \\ -1 \end{bmatrix}_h \bar{w}_h = f_h + \frac{1}{h^2}([\varepsilon \quad 0 \quad 0]_h \bar{w}_h + [0 \quad 0 \quad \varepsilon]_h w_h). \tag{5.1.9}$$

It is thus characterized by the splitting

$$L_h^- = \frac{1}{h^2}\begin{bmatrix} & 0 & \\ 0 & 0 & -\varepsilon \\ & 0 & \end{bmatrix}, \qquad L_h^+ = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -\varepsilon & 2+2\varepsilon & 0 \\ & -1 & \end{bmatrix}, \tag{5.1.10}$$

and the smoothing factor μ_{loc} for (5.1.3) can be calculated as in Section 4.3. We find

$$\mu_{\text{loc}} = \sup_{\boldsymbol{\theta} \in I^{\text{high}}} \left|\tilde{S}_h(\boldsymbol{\theta})\right| = \sup_{\boldsymbol{\theta} \in I^{\text{high}}} \left|\frac{\varepsilon e^{i\theta_1}}{\varepsilon e^{-i\theta_1} + e^{-i\theta_2} - 2 - 2\varepsilon + e^{i\theta_2}}\right|. \tag{5.1.11}$$

μ_{loc} of lexicographic y-line Gauss–Seidel for arbitrary $\varepsilon > 0$ turns out to be

$$\mu_{\text{loc}} = \max\left(\frac{1}{\sqrt{5}}, \frac{\varepsilon}{2+\varepsilon}\right), \tag{5.1.12}$$

the first value being obtained for $(\theta_1, \theta_2) = (\pi/2, 0)$ and the latter one for $(\theta_1, \theta_2) = (0, \pi/2)$. Equation (5.1.12) implies that

$$\mu_{\text{loc}} = 1/\sqrt{5} \quad \text{for } 0 < \varepsilon \leq 1.$$

The influence of the y-line smoother on the error for Model Problem 3 with $\varepsilon = 10^{-2}$ is presented in Fig. 5.3.

For $\varepsilon \gg 1$ the smoothing factor of y-line relaxation tends to 1. In such cases, x-line relaxation is to be used instead: line relaxation parallel to the y-axis (*y-line relaxation*) is suitable if $\varepsilon < 1$, and *x-line relaxation* if $\varepsilon > 1$.

Line smoothers other than lexicographic line Gauss–Seidel are, for example, *line ω-Jacobi* (with underrelaxation), or *zebra line Gauss–Seidel smoothing*. Zebra line Gauss–Seidel relaxation is the line analog to pointwise RB Gauss–Seidel relaxation. Smoothing again consists of two half-steps. First all odd lines are processed, then all even ones. In the second half-step, the updated approximations on the odd lines are used. Figure 5.4 presents the points that are updated collectively in an x- and in a y-line zebra smoothing procedure, by the solution of tridiagonal systems.

Table 5.2 contains two-grid convergence factors ρ_{loc} for zebra line GS obtained by two-grid LFA (see Section 4.4). It can be seen that the line smoother in the direction of the

Figure 5.3. Influence of y-line Gauss–Seidel relaxation on the error for the 2D anisotropic model problem with $\varepsilon = 10^{-2}$.

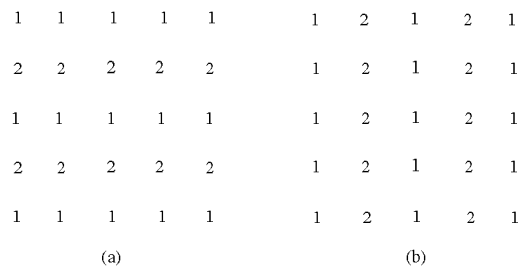


Figure 5.4. Zebra line Gauss–Seidel relaxation: approximations in points marked by 1 are updated in the first, those marked by 2 in the second half-step of the relaxation. (a) x-zebra line Gauss–Seidel (xZGS), (b) y-zebra line Gauss–Seidel relaxation (yZGS).

Table 5.2. Two-grid convergence factors ρ_{loc} for the anisotropic model problem using x- or y-zebra line Gauss–Seidel ($\nu = 2$ smoothing steps), FW and bilinear interpolation for different values of ε.

ε	0.001	0.01	0.1	0.5	1	2	10	100	1000
xZGS:	0.996	0.96	0.68	0.20	0.063	0.028	0.047	0.052	0.053
yZGS:	0.053	0.052	0.047	0.028	0.063	0.20	0.68	0.96	0.996

"weak coupling" fails to achieve satisfactory convergence. Line smoothing in the direction of the strong coupling works perfectly.

The cost of the zebra line smoother is the same as that of lexicographic line Gauss–Seidel.

Remark 5.1.4 One advantage of using zebra line Gauss–Seidel is that its smoothing factors and thus also the multigrid convergence factors are better than those of lexicographic line Gauss–Seidel relaxations (e.g. μ_{loc} ($\nu = 1$) $= 0.125$ for yZGS and $\varepsilon \leq 0.5$ [378] versus $1/\sqrt{5}$ in the lexicographic case). ≫

Remark 5.1.5 The degree of parallelism of zebra line Gauss–Seidel is higher than that of lexicographic line Gauss–Seidel relaxation. For five (or compact nine-point stencils (1.3.11)), the degree of parallelism is

$$\text{par-deg(zebra GS)} \geq \tfrac{1}{2}\sqrt{\#\Omega_h}.$$

Half of the lines can be processed in parallel. It depends on the concrete choice of the tridiagonal solver whether or not there is a further degree of parallelism inherent and how it can be exploited (see Section 6.4.2). Of course, the degree of parallelism is even better with line Jacobi relaxation, but, similarly to the point Jacobi for Model Problem 1, its smoothing factor turns out to be worse than that of zebra line Gauss–Seidel. For example, we find that the smoothing factor μ_{loc} of x-line Jacobi relaxation is $\mu_{\text{loc}} = 1/3$ for $\varepsilon = 1000$ and the optimal $\omega = 2/3$. ≫

Remark 5.1.6 (h-dependence of convergence factors) Generally, line relaxation methods are "less local" than pointwise relaxation; often, long range effects cannot be neglected. This is the reason why the measured multigrid convergence factors can differ considerably for different values of h (and can become much better than predicted by LFA). For example, for Model Problem 3 ρ_{loc} is first observed on a grid with $h = 1/2048$; on coarser grids a much better convergence is observed. This can be confirmed by rigorous Fourier analysis (for a fixed h). ≫

5.1.4 Strong Coupling of Unknowns in Two Directions

Up to now we have discussed the multigrid remedies for anisotropic problems with constant coefficients. In practice, ε is often not constant but varying: $\varepsilon = \varepsilon(x, y)$. Then, ε may be

Table 7.5. Smoothing factors of multistage Jacobi relaxation with optimal parameters (7.4.8).

p	1	2	3	4
μ^*	0.60	0.22	0.074	0.025
ρ^*	0.60	0.22	0.13	0.11

Table 7.6. Three-stage Jacobi smoothing with optimal parameters for standard upwind and Fromm's discretizations of the convection terms with $(a, b) = (1, 1), h = 1/128$.

Discretization	ε	α_1	α_2	α_3	μ_{loc}
$O(h)$ upwind	10^{-2}	0.25	0.85	2.82	0.16
$O(h^2)$ $\kappa = 0$	10^{-2}	0.27	0.85	2.66	0.25
$O(h)$ upwind	10^{-3}	0.28	0.82	2.16	0.31
$O(h^2)$ $\kappa = 0$	10^{-3}	0.22	0.62	1.31	0.58
$O(h)$ upwind	10^{-4}	0.27	0.77	1.93	0.35
$O(h^2)$ $\kappa = 0$	10^{-4}	0.22	0.54	1.03	0.67

Example 7.4.5 For Problem 7.1.1 with fixed coefficients $(a, b) = (1, 1)$, the optimal parameters for a three-stage Jacobi smoother are given in Table 7.6 together with some LFA smoothing factors (for $h = 1/128$). It can be seen that the multistage Jacobi smoother gives reasonable smoothing factors for this convection–diffusion example. They are, however, much worse than the smoothing factors of the KAPPA smoother found in Table 7.4 for the same problem. △

For other values of a and b, for instance for $(a, b) = (1, 0)$ and $\varepsilon = 10^{-4}$ or smaller, the best smoothing factor obtained with a three-stage Jacobi smoother and standard coarsening is $\mu_{\text{loc}} = 0.96$. This indicates that it is not easy to obtain a satisfactory smoothing factor with a three-stage Jacobi method for Fromm's upwind-type discretization if the convection direction is *aligned with the grid*. A remedy is to use semicoarsening in the appropriate direction or to employ multistage linewise smoothing methods.

The coarse grid correction problem, however, remains. It will be further discussed in Section 7.8.1.

7.5 ILU SMOOTHING METHODS

In this section, we leave the convection–diffusion equation behind and discuss a different relaxation approach, the "ILU-type" (incomplete LU matrix decomposition) smoothers. ILU smoothers represent a class of smoothing procedures all of which are based on an incomplete LU-type factorization of the matrix A_h corresponding to L_h. Actually, a variety of such smoothing procedures has been proposed [211, 212, 414, 442] and investigated [415, 420–422, 439, 440]. Certain versions of ILU type smoothing, for example the one discussed in Section 7.5.4, are regarded as particularly robust for anisotropic 2D problems with the anisotropy not necessarily aligned with the grid. For 3D problems, however, robust ILU smoothers for general anisotropic equations are more complicated and expensive.

Remark 7.5.1 We will not give a detailed formal description of an ILU algorithm here (see, for example, [9] for algorithmic details). Furthermore, we assume here that an ILU decomposition exists. Existence is proved, for example, for M-matrices in [264]. ≫

7.5.1 Idea of ILU Smoothing

Since ILU has originally been developed in the framework of *matrices*, we first describe the ILU decomposition in this form. Later, we will also use a corresponding stencil notation.

We consider a problem $A_h u_h = f_h$ with a structured sparse $N \times N$ matrix A_h. For simplicity, we assume here a regular structure consisting of $2+3+2$ diagonals, as sketched in Fig. 7.15. This structure corresponds to a seven-point discretization (in 2D) which is somewhat more general than the five-point stencils considered before. Two examples of seven-point discretizations will be given in Section 7.6.1.

If the usual (complete) LU decomposition is applied to A_h, i.e. $A_h = \mathcal{L}_h \mathcal{U}_h$, then the matrices $\mathcal{L}_h$ and $\mathcal{U}_h$ are also band matrices, but the bands are filled with nonzero elements (see Fig. 7.16). The number of interior "zero diagonals" will increase like h^{-1} if $h \to 0$. After the decomposition, the triangular systems $\mathcal{L}_h v_h = f_h$ and $\mathcal{U}_h u_h = v_h$ are solved successively to compute u_h.

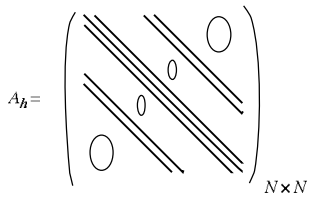


Figure 7.15. A structured sparse matrix.

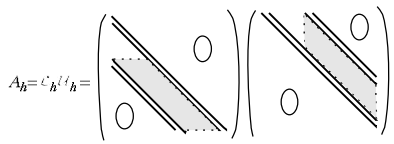


Figure 7.16. An LU-decomposition of a structured sparse matrix with fill-in (indicated in grey).

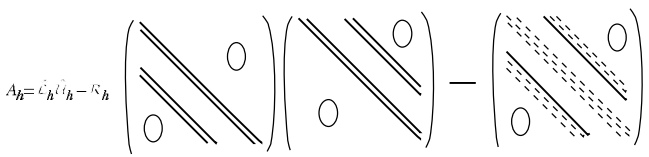


Figure 7.17. An ILU-decomposition of a banded matrix A_h.

In an *incomplete* LU (ILU) decomposition, $A_h = \hat{\mathcal{L}}_h\hat{\mathcal{U}}_h - \mathcal{R}_h$, one forces the matrices $\hat{\mathcal{L}}_h$ and $\hat{\mathcal{U}}_h$ to be sparse also: *a nonzero sparsity pattern is prescribed for the matrices* $\hat{\mathcal{L}}_h$ *and* $\hat{\mathcal{U}}_h$. In the simplest case, $\hat{\mathcal{L}}_h$ and $\hat{\mathcal{U}}_h$ have the same sparsity pattern as A_h restricted to the respective triangular parts.

Roughly speaking, the ILU decomposition (with the same sparsity pattern as the original matrix A_h) is computed as follows: for the calculation of the coefficients $\hat{l}_{ij}$ (of $\hat{\mathcal{L}}_h$) and of the coefficients $\hat{u}_{ij}$ (of $\hat{\mathcal{U}}_h$) proceed as for the complete LU decomposition, but replace in the algorithm (step by step) those $\hat{l}_{\rho\sigma}$ and $\hat{u}_{\rho\sigma}$ by 0 for which (ρ, σ) corresponds to one of the zero diagonals of A_h.

Note that ILU depends on the ordering of the grid points. In particular, the entries in ILU are different for row and column ordering. This allows the definition of *alternating ILU decompositions* used later in this section, where two ILU decompositions, corresponding to different orderings (e.g. row and column) are combined.

Since the ILU decomposition is not exact, an error matrix $\mathcal{R}_h$ remains. In the case of a matrix A_h according to Fig. 7.15, the error matrix $\mathcal{R}_h$ contains two additional diagonals as indicated in Fig. 7.17. (The *dashed* lines in $\mathcal{R}_h$ are zeros and correspond to the nonzero entries in A_h. They are shown here to illustrate the location of the additional diagonals of $\mathcal{R}_h$.)

Remark 7.5.2 ILU was introduced in [264] in its incomplete Cholesky (IC) variant and proposed as a preconditioner for the conjugate gradient method (ICCG). Various modifications have been proposed since then. In the incomplete Cholesky decomposition of a symmetric positive definite matrix A_h, we have

$$A_h = \mathcal{L}_h\mathcal{L}_h^T - \mathcal{R}_h \quad \text{with } \mathcal{R}_h = \mathcal{R}_h^T$$

(see [264] for details). ≫

In principle, the ILU decomposition can be used iteratively (see also (1.6.7)) to solve $A_h u_h = f_h$:

$$u_h^0 = 0, \qquad u_h^{m+1} = u_h^m + \hat{v}_h^m \quad (m = 0, 1, 2, \ldots),$$

where

$$\hat{\mathcal{L}}_h\hat{\mathcal{U}}_h\hat{v}_h^m = d_h^m = f_h - A_h u_h^m. \tag{7.5.1}$$

This iteration can also be written in the form

$$\hat{\mathcal{L}}_h\hat{\mathcal{U}}_h u_h^{m+1} = \mathcal{R}_h u_h^m + f_h, \quad m = 0, 1, \ldots. \tag{7.5.2}$$

≫

However, the ILU iteration has, in general, poor convergence properties. Nevertheless, its *smoothing properties* are good for a wide range of problems. ILU can therefore be used as a smoother in multigrid methods.

7.5.2 Stencil Notation

In order to investigate the smoothing properties of the above ILU iteration by LFA, we now switch to the stencil notation and extend all occurring operators to the infinite grid $\mathbf{G}_h$.

For a given discrete seven-point difference operator L_h,

$$L_h = \begin{bmatrix} \star & \star & \\ \star & \star & \star \\ & \star & \star \end{bmatrix}_h,$$

a *seven-point ILU decomposition* can be represented in stencil notation by

$$L_h = \hat{L}_h\hat{U}_h - R_h, \tag{7.5.3}$$

where the stencils $\hat{L}_h$, $\hat{U}_h$ and R_h correspond to the matrices $\hat{\mathcal{L}}_h$, $\hat{\mathcal{U}}_h$ and $\mathcal{R}_h$, respectively. The multiplication of stencils corresponds directly to the multiplication of operators (for the formulas, see [342]). Depending on the ordering of grid points, in principle, eight different ILU decompositions can be defined. If the ordering of grid points is first in east direction (E), and then in north direction (N), this EN decomposition is of the form

$$\hat{L}_h : \begin{bmatrix} 0 & 0 & \\ \star & \star & 0 \\ & \star & \star \end{bmatrix}_h, \qquad \hat{U}_h : \begin{bmatrix} \star & \star & \\ 0 & \star & \star \\ & 0 & 0 \end{bmatrix}_h, \qquad R_h : \begin{bmatrix} \star & 0 & 0 & \\ & 0 & 0 & 0 \\ & & 0 & 0 & \star \end{bmatrix}_h. \tag{7.5.4}$$

Here, $\star$ stands for possible nonzero elements. This decomposition corresponds to the one in Fig. 7.17. In the following, however, we will analyze the "NE decomposition" with ordering first in the north direction then eastwards. The corresponding stencils read

$$\hat{L}_h : \begin{bmatrix} \star & 0 & \\ \star & \star & 0 \\ & \star & 0 \end{bmatrix}_h, \qquad \hat{U}_h : \begin{bmatrix} 0 & \star & \\ 0 & \star & \star \\ & 0 & \star \end{bmatrix}_h, \qquad R_h : \begin{bmatrix} \star & & \\ 0 & 0 & \\ 0 & 0 & 0 \\ & 0 & 0 \\ & & \star \end{bmatrix}_h. \tag{7.5.5}$$

Remark 7.5.3 Another decomposition is obtained by a north-west ordering (NW),

$$\hat{L}_h : \begin{bmatrix} 0 & 0 & \\ 0 & \star & \star \\ & \star & \star \end{bmatrix}_h, \qquad \hat{U}_h : \begin{bmatrix} \star & \star & \\ \star & \star & 0 \\ & 0 & 0 \end{bmatrix}_h, \qquad R_h : \begin{bmatrix} 0 & 0 & \star \\ 0 & 0 & 0 \\ \star & 0 & 0 \end{bmatrix}_h. \tag{7.5.6}$$

≫

Remark 7.5.4 Other ILU decompositions can be obtained by using larger stencils for $\hat{L}_h$ and $\hat{U}_h$. Such decompositions do no longer reflect the sparsity pattern of L_h (and are somewhat more expensive). For example, *nine-point ILU decompositions* have been used in [415] with good smoothing properties. ≫

In stencil notation, the ILU iteration (7.5.2) reads

$$\hat{L}_h \hat{U}_h u_h^{m+1} = R_h u_h^m + f_h \quad \text{or} \quad (L_h + R_h) u_h^{m+1} = R_h u_h^m + f_h. \tag{7.5.7}$$

We denote the symbols of L_h, $\hat{L}_h$, $\hat{U}_h$ and R_h here by

$$\lambda_h(\boldsymbol{\theta}), \quad \lambda_h^L(\boldsymbol{\theta}), \quad \lambda_h^U(\boldsymbol{\theta}), \quad \lambda_h^R(\boldsymbol{\theta}) \qquad (-\pi \le \boldsymbol{\theta} < \pi),$$

respectively. Correspondingly, the smoothing operator S_h, defined by (7.5.7), is represented by

$$S_h \varphi(\boldsymbol{\theta}, x) = \tilde{S}_h(\boldsymbol{\theta}) \varphi(\boldsymbol{\theta}, x) \quad (-\pi \le \boldsymbol{\theta} < \pi) \tag{7.5.8}$$

with

$$\tilde{S}_h(\boldsymbol{\theta}) := \frac{\lambda_h^L(\boldsymbol{\theta}) \lambda_h^U(\boldsymbol{\theta}) - \lambda_h(\boldsymbol{\theta})}{\lambda_h^L(\boldsymbol{\theta}) \lambda_h^U(\boldsymbol{\theta})} = \frac{\lambda_h^R(\boldsymbol{\theta})}{\lambda_h(\boldsymbol{\theta}) + \lambda_h^R(\boldsymbol{\theta})},$$

(assuming that the denominator $\neq 0$).

LFA smoothing results for Model Problem 1 (Poisson's equation) are given in the next subsection together with results for anisotropic operators (see also [285, 378, 415] for further results and details). Before we discuss smoothing properties, we point out in the following two remarks that the LFA results for ILU smoothing must be interpreted with some care.

Remark 7.5.5 For similar reasons as explained for GS-LEX, ILU smoothing cannot be analyzed by the *rigorous* Fourier analysis. However, LFA can be used.

In this context, note that: for an operator L_h with constant coefficients on an infinite grid, the coefficients of $\hat{L}_h$, $\hat{U}_h$ and R_h are also constant. Due to boundary conditions, for example, the nonzero entries in the matrices corresponding to a finite grid problem are generally *not constant.* ≫

Remark 7.5.6 As also discussed for linewise relaxation in Remark 5.1.6, ILU is "less local" than typical pointwise relaxation methods; often, long range effects can only be neglected asymptotically (for $h \to 0$). We will see in the following section that the ILU decomposition may even closely resemble an LU decomposition for certain problems. The

Table 7.7. LFA results (μ_{loc}, ρ_{loc}) and measured V(1,0) convergence factors $\rho_h (h = 1/128)$ for the anisotropic model operator using *seven-point ILU* (7.5.5) for smoothing.

ε	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	10^2	10^3	10^4
μ_{loc}	0.17	0.17	0.17	0.17	0.13	0.27	0.61	0.84	0.95
ρ_{loc}	0.17	0.17	0.17	0.17	0.13	0.27	0.61	0.84	0.94
ρ_h	0.011	0.10	0.16	0.16	0.12	0.27	0.58	0.65	0.23

error operator R_h then contains very small elements. This is the reason why the measured multigrid convergence factors can differ considerably for different values of h and are often much better than predicted by LFA. ≫

7.5.3 ILU Smoothing for the Anisotropic Diffusion Equation

ILU smoothers can be used for 2D anisotropic problems. We consider (5.1.1) with the five-point stencil $L_h(\varepsilon)$ (5.1.3).

Remark 7.5.7 For five-point stencils, in principle, a five-point ILU decomposition can be used. However, already for the anisotropic diffusion equation $L(\varepsilon)u = -\varepsilon u_{xx} - u_{yy}$, the five-point ILU decomposition turns out to be a bad smoother for (very) small and for (very) large values of ε [415]. The seven-point "north-east" ILU iteration (7.5.5), for example, has much better smoothing properties for small values of ε. The seven-point north-west ILU decomposition (7.5.6), on the other hand, is identical to the five-point decomposition, because of zero entries in $\hat{L}_h$ and $\hat{U}_h$. We therefore consider the seven-point north-east ILU (7.5.5) for five-point stencils in the following. In this case, the sparsity pattern of the decomposition is not identical to that of L_h. ≫

Table 7.7 shows LFA smoothing and two-grid factors μ_{loc} and ρ_{loc} for the 2D anisotropic model problem together with measured numerical V(1,0)-cycle convergence obtained on a 128^2 grid. On the coarse grids, the direct discretization of the PDE is used to define L_H. Furthermore, FW and bilinear interpolation are employed as the transfer operators.

Obviously, the convergence properties of the above method are not symmetric with respect to ε. We have good convergence for $\varepsilon \le 1$ but μ_{loc} and ρ_{loc} tend to 1 for $\varepsilon \to \infty$. This behavior is similar to that observed in connection with y-line ZEBRA relaxation in Table 5.2. For large ε the measured convergence is nevertheless satisfactory for h not very small. The prediction of LFA is too pessimistic for a certain range of ε ($\varepsilon = 10^{\pm 4}$ in Table 7.7). This is due to the fact that, for a fixed mesh size h and $\varepsilon \to \infty$ or $\varepsilon \to 0$, ILU becomes almost a direct solver (see Remark 7.5.6). Consequently, the resulting fast ILU-convergence "supersedes" the multigrid convergence expected (see also [211, 285]). This is not reflected by the LFA because this effect vanishes for fixed ε and $h \to 0$: LFA predicts the asymptotic case $h \to 0$.

7.5.4 A Particularly Robust ILU Smoother

The numerical results discussed above show that multigrid methods with the seven-point ILU smoother (7.5.5) do not always lead to good convergence. The asymmetric behavior of ILU smoothing has to be overcome in order to obtain a robust 2D solution method. This can be achieved by two modifications of the original decomposition.

- **Alternating ILU** As discussed above, the ILU decomposition can be used for smoothing in an alternating manner (like the alternating zebra-line relaxation method in Section 5.1.4), giving a fast method for 2D anisotropic equations, independent of ε. That is, a standard row-ordered ILU is followed by a standard column-ordered ILU or vice versa. In this way, the above nonsymmetric behavior disappears. The smoothing properties are good for both small *and* large ε (see [285]). Here, we combine the north-east (NE) ILU (7.5.5) with the west-south (WS) ILU. The alternating ILU decomposition is, however, not yet an efficient smoother for more general problems which involve, for example, mixed derivatives (see Section 7.6). For that purpose, the following additional modification is also important [210].
- **Modified ILU** In the case of the usual ILU decomposition, the matrix $\mathcal{R}_h$ contains zeros at all positions which correspond to the nonzero structure of $\hat{\mathcal{L}}_h$ and $\hat{\mathcal{U}}_h$, especially along the main diagonal, i.e. $r_{ii} = 0$ (as in Fig. 7.17). With a parameter δ a *modified ILU decomposition*

$$A_h = \hat{\mathcal{L}}_h^\delta \hat{\mathcal{U}}_h^\delta - \mathcal{R}_h^\delta \tag{7.5.9}$$

can be defined, for which the diagonal of $\mathcal{R}_h^\delta$ is no longer zero. Assuming, for example, that the incomplete factorization of A_h is computed row by row, the main diagonal of $\mathcal{R}_h$ and the u_{ii} of $\hat{\mathcal{U}}_h$ can be modified in such a way that

$$r_{ii} \leftarrow \delta \sum_{j \neq i} |r_{ij}|, \qquad u_{ii} \leftarrow u_{ii} + r_{ii}$$

immediately before the coefficients of row number $i+1$ are computed. Reasonable choices for δ are in the range $0 < \delta \leq 1$, where $\delta = 0$ corresponds to the nonmodified ILU decomposition. The modified ILU decomposition can also be analyzed by LFA. Based on the LFA and confirmed by measured convergence factors, it has been shown [285] that the modification of ILU with $\delta = 1$ results in better convergence and in improved robustness. This modification is useful, for instance, in the case that a main diagonal element in the $\hat{\mathcal{U}}_h$-part of the decomposition becomes zero or less than zero numerically. It keeps this main diagonal at a certain positive value and adapts the error matrix $\mathcal{R}_h$ accordingly.

The combination of both modifications presented above leads to *alternating modified ILU* as a smoother, which makes 2D multigrid impressively robust. Here, we show that it handles the anisotropic model problem very well for the whole ε-range. More results will follow in the next section. Table 7.8 gives numerical V(1, 0) multigrid convergence results for the

Table 7.8. Measured V(1, 0) multigrid convergence factors ρ_h for the anisotropic model operator $L_h(\varepsilon)$ using seven-point modified ($\delta = 1$) alternating NE–WS ILU for smoothing.

ε	10^{-4}	10^{-3}	10^{-2}	0.1	1	10	10^2	10^3	10^4
ρ_h	0.001	0.027	0.048	0.048	0.040	0.047	0.049	0.027	0.001

anisotropic Model Problem 3 discretized on a grid with $h = 1/128$. Alternating modified ILU with $\delta = 1$ is the smoother.

Remark 7.5.8 The alternating modified ILU decomposition (and also block ILU decompositions) also have good smoothing properties for the convection–diffusion model problem [211, 415]. ≫

Remark 7.5.9 (ILU and parallelism) It is not at all easy to efficiently parallelize an ILU decomposition and maintain exactly the same method as in the sequential situation. In principle, parallel variants of ILU can be obtained by employing a "parallel ordering" of grid points (like red–black ordering). However, this approach can reduce the smoothing properties of ILU and the corresponding multigrid convergence significantly (for sophisticated modifications leading to satisfactory parallel behavior, see [153]).

≫

Remark 7.5.10 (3D ILU smoothers) For general 3D problems, it is not straightforward to define a robust ILU-based smoother in combination with standard coarsening as is indicated by LFA, for example, in [212]. Only expensive ILU-based smoothers have been proposed so far. An efficient 3D ILU smoother can, however, be developed in the case of one dominant direction or in the case of two dominant directions [212, 213]. ≫

7.6 PROBLEMS WITH MIXED DERIVATIVES

We will now consider a 2D differential operator $L^\tau u$ with a mixed derivative u_{xy}. In particular, we will treat the model equation

$$L^\tau u = -\Delta u - \tau u_{xy} = f \quad (\Omega). \tag{7.6.1}$$

For this equation, any reasonable discretization leads to at least seven- or nine-point stencils with *diagonal entries*. Mixed derivatives are no longer aligned with the x- and the y-axes, but can lead to *diagonal* anisotropies.

Equation (7.6.1) is interesting for two reasons. First, it is a very simple equation, where we can study the multigrid treatment of mixed derivatives systematically. In principle, we have already dealt with mixed derivatives in the context of the full potential equation in Section 5.3.6. From the good convergence results there, we may conclude that their occurrence presents no major problems.