Hannah Rittich

Extending and Automating Fourier Analysis for Multigrid Methods

A dissertation submitted to the Faculty of Mathematics and Natural Sciences of the University of Wuppertal in partial fulfillment of the requirements for the degree of Doctor of Sciences

June 2017

Supervised by Prof. Dr. Matthias Bolten

Die Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20171220-120820-1
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20171220-120820-1]

It is not possible to be a scientist unless you believe that it is good to learn. It is not good to be a scientist, and it is not possible, unless you think that it is of the highest value to share your knowledge, to share it with anyone who is interested. It is not possible to be a scientist unless you believe that the knowledge of the world, and the power which this gives, is a thing which is of intrinsic value to humanity, and that you are using it to help in the spread of knowledge, and are willing to take the consequences.

—Robert Oppenheimer, November 2, 1945

Contents

Preface

This thesis gives a general framework for *local Fourier analysis* of *multigrid methods* that is versatile and well suited for computer implementation. Multigrid methods are often a crucial component for the efficient numerical solution of partial differential equations. Local Fourier analysis is an idealized, quantitative analysis for these methods, i.e., it gives an estimation of the behavior of a multigrid method.

Multigrid methods [12, 15, 32, 63, 68] consist of two components. A *smoother* and a *coarse grid correction*. There exist many different smoothing methods and many different ways to construct a coarse grid correction. These two components have to complement each other to be efficient. Furthermore, they have to be chosen in such a way that matches the problem that is to be solved. As the implementation of a multigrid method can be tedious, one wants to know how well different smoothers and coarse grid corrections work for a specific problem in advance. Local Fourier analysis is an efficient tool that provides the desired information.

Local Fourier analysis is performed by analyzing a *substitute* problem that is similar to the one we are interested in, but simpler to analyze. Therefore, local Fourier analysis is an *idealized* analysis. As we expect that similar problems behave similarly, by knowing the behavior of the substitute problem, we can predict the behavior of the original problem. To apply local Fourier analysis to a problem, however, we need to find a suitable substitute problem.

Local Fourier analysis has been successfully applied to many problems [9, 13, 26, 27, 36–38, 45, 48, 49, 63, 68, 74–77] and there already exists a software [74] that performs local Fourier analysis. Consequently, why do we need a new framework and a new analysis tool?

We need a new framework, as there are still applications that cannot be analyzed with local Fourier analysis. The new framework, which we introduce in this thesis, allows the analysis of new applications, for example the analysis of jumping coefficient problems and block smoothers (Chapter 4). On the other hand, we can analyze many of the problems that have been successfully treated by local Fourier analysis within the new framework as well. As a consequence it allows the analysis of combinations of multigrid components that have been analyzed with different local Fourier approaches, as these approaches can now be formulated within the same framework.

We need a new software tool, because the program described in [74] has some limitations. This software tool is essentially a collection of templates, i.e., the software allows to fill in some blank spots, but it is not possible to reuse or recombine components. This limits the situations where the software can applied and extending the software is complicated. In contrast, the new analysis software that we propose (Chapter 5) builds on a limited set of

primitive components[1]. These components can then be combined in various ways to analyze arbitrary complex methods. The local Fourier analysis framework that we shall introduce will provide us with these primitive components and ways to combine them.

The primitive components that we shall use will be *matrix symbols*. They can be used to represent *periodic stencil operators*, *restriction operators*, and *interpolation operators* (Chapter 3). Matrix symbols, can be combined in various ways, and the combined symbols then represent corresponding combined operators. Combining a few primitive operators is often sufficient to describe a multigrid method. Hence, this approach gives a great flexibility. Furthermore, as we shall see, these symbols allow us to determine many properties of the method they describe, like the *convergence rate* of the method.

We start with a basic introduction to multigrid methods, formal eigenfunction analysis (Chapter 1), and a rigorous treatment of local Fourier analysis for *constant stencil operators* (Chapter 2).

Acknowledgments

If you work hard to reach a goal it is easy to think that you reached it on your own, and it is easy to forget that you depend on many things to accomplish your work.

For example, I depend on something to eat every day, as trivial as that might sound. Everybody who knows me well will confirm that I do not function well when I am hungry—besides, what would life be without French fries with mayonnaise? My position at the University of Wuppertal gave me a salary and much time, allowing me to fill my fridge, to learn, to be curious, and to work on my thesis. Without this position this thesis would never have been written, and Andreas, Bruno, and Matthias were very successful in convincing the Deutsche Forschungsgemeinschaft (DFG) to fund[2] my position.

I would not have gotten this position if I had not gone to university, and I would never have gone to university if my parents had not supported me and had not encouraged me to pursue my curiosity. Going to university, however, is not enough to write a thesis; I also needed a topic.

To find a topic, I was reading many articles. There are, however, so many articles to read and so many things have already been discovered that it was difficult for me to find a topic that was worth to work on, and it was Matthias who pointed me in the right direction. Furthermore, discussing with Matthias was always helpful. When I got stuck, when I had too many options to pursue, or when I had no ideas, the discussions with him gave me the momentum I needed and a direction to move into.

When I was writing my thesis I needed feedback. There was a time when I was writing all day, and at some point I became insecure, worrying whether I was writing anything comprehensible. Kathryn offered to read my thesis and encouraged me to continue and also improved my writing.

[1] In [3] these are called *primitive expressions*.

[2] This work is partly supported by the German Research Foundation (DFG) through the Priority Programme 1648 “Software for Exascale Computing” (SPPEXA) and the Transregional Collaborative Research Centre 55 (SFB/TRR55) “Hadron Physics from Lattice QCD”.

Writing this thesis was sometimes hard. It caused me sleepless nights when I could not work out a proof or when I was not understanding the article I was reading. It made me angry when my computer was crashing or when it was producing the wrong numbers. I am glad that my family and friends distracted me in those moments. I am glad that they made me forget my problems and let me have a good time. I am especially happy to have Isabel by my side.

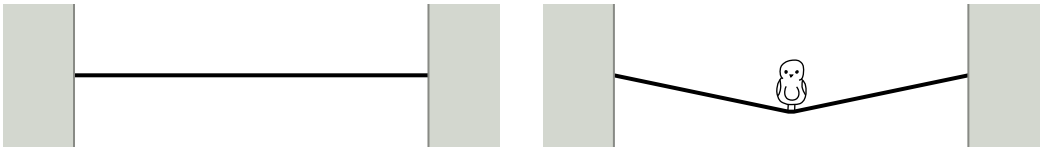
I like to thank all of these people. Without them, I would not have written this thesis.

1. Multigrid Essentials

Multigrid methods form a class of methods rather than a single method. They have to adapted be to the problem they are supposed to solve, and until now nobody has found a general recipe for the construction of multigrid methods that works for arbitrary problems. Therefore, for every new problem that is supposed to be solved by a multigrid method, some clever thinking is required, to choose the right *multigrid components*. Hence, multigrid methods are best explained by constructing a multigrid method for a specific and simple example.

1.1. Wire under Tension

Imagine a wire that is strained between two walls. The tension of the wire is so large that the wire is a straight line between the endpoints. Then a load is attached to the wire which deforms it. We want to determine the shape of the wire.



(a) Wire without load. (b) Wire with load.

Figure 1.1.: Shape of a wire.

The wire always assumes the shape that has the smallest *energy* of all possible shapes. The energy is the sum of the *tension energy* and the *potential energy* of the load. When the load is attached the wire expands. The load moves down and therefore its potential energy decreases. On the other hand, the strain energy increases due to the expansion. Thus, by minimizing the total energy, the system balances between the expansion of the wire and height of the load.

To describe the shape of the wire, we first pick a coordinate system. Let the endpoints of the wire be $(0, 0)$, and $(1, 0)$, and let the vector $(0, 1)$ be pointing in the upward direction, as depicted in Figure 1.2. Now we can describe the shape of the wire by a function $y : [0, 1] \to \mathbb{R}$. Before the load is attached, the wire has the shape $y_0(x) = 0$. Let $L(y)$ be the length of the wire that has the shape y. The length is given by

$$L(y) = \int_0^1 \sqrt{1 + y'(x)^2}\, \mathrm{d}x\,.$$

We start by computing the tension energy of the wire. Let τ be the tension of the wire. Assume that the wire is only stretched by a small amount. That means that τ is nearly

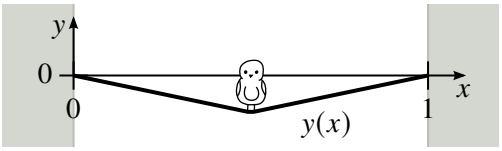


Figure 1.2.: The function $y(x)$ describes the shape of the wire.

constant, i.e., the force required while expanding the wire from its initial length $L(y_0)$ to its final length $L(y)$ equals τ. As a result, the work [29, 67, 78]—and therefore the tension energy—is proportional to the extension of the wire, i.e.,

$$T(y) = \int_{L(y_0)}^{L(y)} \tau \, \mathrm{d}s = \tau \cdot (L(y) - L(y_0)) = \tau \cdot \int_0^1 \sqrt{1 + y'(x)^2} - 1 \, \mathrm{d}x \, .$$

We can simplify this expression if we assume that y' is small. In this case, the Taylor expansion yields $\sqrt{1+t} = 1 + t/2 + O(t^2)$; leading to

$$T(y) \approx \tau \cdot \int_0^1 \tfrac{1}{2} y'(x)^2 \, \mathrm{d}x \, , \tag{1.1}$$

which is our desired formula for the tension energy.

We continue by computing the potential energy. Assume that the mass of the wire is so small in comparison to the load that we can neglect it. Therefore, the only force in vertical direction is caused by the weight of the load. Let $-f(x)$ be the force that acts on the wire at x in the downward direction.

The potential energy is only determined up to a constant. Hence, we can define the potential energy to be zero when the wire has the shape y_0; if the wire has the shape y then the potential energy is

$$V(y) = -\int_0^1 f(x) y(x) \, \mathrm{d}x \, . \tag{1.2}$$

The tension energy (1.1) and the potential energy (1.2) give the *total energy* of the wire-load system:

$$T(y) + V(y) = \int_0^1 \tfrac{\tau}{2} \cdot y'(x)^2 - f(x) y(x) \, \mathrm{d}x = \int_0^1 \mathcal{L}\left(x, y(x), y'(x)\right) \, \mathrm{d}x \, ,$$

where $\mathcal{L}(x, y, y') := \frac{\tau}{2} \cdot y'^2 - f(x) y$. Recall that the wire assumes the shape from all possible shapes that has the smallest total energy. Which is why we can find the shape of the wire under load by finding the function y that minimizes

$$\int_0^1 \mathcal{L}\left(x, y(x), y'(x)\right) \, \mathrm{d}x \, .$$

If the minimum y is in $C^2[0,1]$ then it fulfills the Euler-Lagrange equation [28, 29, 67, 71], i.e.,

$$\tfrac{\partial \mathcal{L}}{\partial y}(x, y(x), y'(x)) - \tfrac{\mathrm{d}}{\mathrm{d}x} \left(\tfrac{\partial \mathcal{L}}{\partial y'}(x, y(x), y'(x)) \right) = 0 \, .$$

Thus,

$$0 = -f(x) - \frac{\mathrm{d}}{\mathrm{d}x}\left(\tau y'(x)\right) = -f(x) - \tau y''(x)\,.$$

If y is the shape of the wire, then it is necessary that y is the solution of the *boundary value problem*

$$-\tfrac{\partial^2 y}{\partial x^2}(x) = f(x) \quad \text{for} \quad x \in (0,1) \tag{1.3a}$$

$$y(0) = y(1) = 0\,. \tag{1.3b}$$

1.1.1. Solving the Boundary Value Problem

Let us solve a slightly more general version of the boundary value problem (1.3). In the original boundary value problem, we required that the function y is zero at the boundary. We shall now require that the function y takes the values $g(0)$ and $g(1)$ at the boundary points 0 and 1, where g is some given function. In the light of the previous section, this situation would correspond to the wire to be attached to the wall at different heights, on the left hand side at height $g(0)$ and on the right hand side at height $g(1)$.

Furthermore, let us introduce some notation. The operator $\frac{\partial^2}{\partial x^2}$ is known as the *one-dimensional Laplace operator*, which is usually denoted by Δ. Hence, we can write the generalized boundary value problem as

$$-\Delta u(x) = f(x) \quad \text{for} \quad x \in (0,1) \tag{1.4a}$$

$$u(x) = g(x) \quad \text{for} \quad x \in \{0,1\}\,. \tag{1.4b}$$

To solve the boundary value problem, we integrate both sides of (1.4a) twice; giving

$$-u(x) + c_1 x + c_2 = \iint f(x)\,\mathrm{d}x\,\mathrm{d}x =: F(x)\,,$$

where $c_1 \in \mathbb{R}$ and $c_2 \in \mathbb{R}$ are constants of integration. Thus, by rearranging the terms

$$u(x) = c_1 x + c_2 - F(x)\,. \tag{1.5}$$

We substitute zero for x into (1.5) to get $c_2 = F(0) + g(0)$. This fixes c_2. Substituting one for x into (1.5) yields $c_1 = F(1) + g(0) - c_2$ which also fixes c_1. Consequently, (1.5) is the solution of the boundary value problem (1.4).

Note that the solution is unique; the antiderivative is unique up to a constant. Both constants of integration, however, are completely determined by the boundary values (1.4b).

1.2. Numerical Approximation

The boundary value problem (1.4) can be solved analytically if we can find a second antiderivative of f. Finding the antiderivative is, however, not always possible [54, 55]. In this case a numerical approximation of the solution is desired.

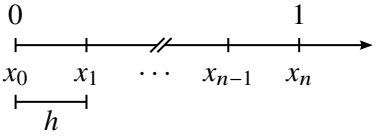


Figure 1.3.: The points of the grid $\overline{\Omega}_h$.

1.2.1. Discretization

To approximate the solution numerically we need to simplify the problem, i.e., we just compute approximations to the function values of u just at a finite set of $n \in \mathbb{N}$ points $x_0, \ldots, x_n$ ($n \in \mathbb{N}$). We choose these points $x_i \in [0, 1]$ to be equidistant, i.e.,

$$x_i := i \cdot h \quad \text{where} \quad h = \tfrac{1}{n}\,.$$

We denote this set of points by $\overline{\Omega}_h$. It is called *a (one-dimensional) grid* and illustrated in Figure 1.3. Furthermore, we define the *interior* and *boundary* of the grid by

$$\Omega_h := \{x_i : i \in \mathbb{N}, 1 \le i \le n-1\} \quad \text{and} \quad \partial\Omega_h := \{x_0, x_n\}\,,$$

respectively. Consequently, the whole grid $\overline{\Omega}_h = \Omega_h \cup \partial\Omega_h$. Let us denote the approximation to u by u_h.

We want to compute $u_h(x_0), \ldots u_h(x_n)$ such that the boundary value problem (1.4) is approximately solved, i.e.,

$$-\Delta u_h(x) = f(x) \quad \text{for } x \in \Omega_h, \tag{1.6a}$$

$$u_h(x) = g(x) \quad \text{for } x \in \partial\Omega_h. \tag{1.6b}$$

We can write this equation more explicitly as

$$-\Delta u_h(x_i) = f(x_i) \quad \text{for } i = 1, \ldots, n-1, \quad u_h(x_0) = g(0), \quad u_h(x_n) = g(1).$$

These are $(n+1)$ equations for $(n+1)$ unknowns. However, we cannot directly use these equations to compute the approximation u_h, as they involve the derivative of the function u_h. We do not have access to the derivative of u_h, because we compute u_h only at a finite set of points. Therefore, we need to approximate Δu using only the values $u_h(x_0), \ldots, u_h(x_n)$.

We construct this approximation using polynomial interpolation. For $x \in \mathbb{R}$ we can find a polynomial p with $\deg p \le 2$ such that

$$p(-h) = u(x-h), \quad p(0) = u(x), \quad \text{and} \quad p(h) = u(x+h)\,.$$

From the Lagrange interpolation formula [34, 60] it follows that this formula is given by

$$p(z) = u(x-h)\frac{z^2 - zh}{2h^2} - u(x)\frac{z^2 - h^2}{h^2} + u(x+h)\frac{z^2 + zh}{2h^2}\,.$$

Applying the Laplace operator, i.e., differentiating twice, yields

$$\Delta p(z) = u(x-h)\frac{1}{h^2} - u(x)\frac{2}{h^2} + u(x+h)\frac{1}{h^2}\,.$$

The value $p(z)$ is an approximation to $u(x+z)$ on the interval for $z \in [-h, h]$. Thus, $\Delta p(0)$ should be an approximation to $\Delta u(x)$. Let $\Delta_h u(x) := \Delta p(0)$, i.e.,

$$\Delta_h u(x) := \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}, \tag{1.7}$$

then $\Delta u(x) \approx \Delta_h u(x)$, which is the so called *finite difference discretization* of the one-dimensional Laplace operator Δ.

We can plug the approximation (1.7) into the reduced boundary value problem (1.6) to find

$$\begin{aligned} \Delta_h u_h(x) &= f(x) \quad \text{for } x \in \Omega_h \\ u_h(x) &= g(x) \quad \text{for } x \in \partial\Omega_h. \end{aligned}$$

More explicitly

$$\begin{aligned} \frac{u_h(x_{i-1}) - 2u_h(x_i) + u_h(x_{i+1})}{h^2} &= f(x_i) \quad \text{for } i = 1, \dots, n-1, \\ u_h(x_0) = g(0), \quad u_h(x_n) &= g(1). \end{aligned}$$

This is a linear system in the variables $u_h(x_i)$, which we can write as

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_h(x_1) \\ u_h(x_2) \\ \vdots \\ u_h(x_{n-2}) \\ u_h(x_{n-1}) \end{pmatrix} = \begin{pmatrix} f(x_1) + g(0)/h^2 \\ f(x_2) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) + g(1)/h^2 \end{pmatrix}. \tag{1.8}$$

Thus, we have a linear system with $n-1$ equations and $n-1$ variables that we can (try to) solve. If we can find its solution, then its solution $u_h(x_i)$ for $i = 0, \dots, n$ is then an approximation to the solution u of the boundary value problem (1.4).

1.2.2. Consistency

We constructed the discretization

$$\Delta_h u(x) := \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} \tag{1.7 revisited}$$

of the Laplace operator Δu. It is, however, not clear that it yields useful results, i.e., that the approximation u_h is close to the true solution. We want that the approximation u_h converges to the true solution u, if the step-size h converges to zero, which can be shown by proving that the discretization is *consistent* and *stable*. In this section we show that Δ_h is consistent; in section 1.2.3 we show that it is stable and that consistency and stability imply the desired convergence.

Definition 1.1. The discretization Δ_h is *consistent* if there exists a function $c : [0, \infty) \to \mathbb{R}$ with $c(h) \to 0$ for $h \to 0$ and

$$\|\Delta_h u - \Delta u\|_h \leq c(h), \tag{1.9}$$

where

$$\|u\|_h := \max_{x \in \Omega_h} |u(x)|.$$

In other words a consistent discretization converges to the continuum operator. Let us now prove that the discretization Δ_h is consistent. For this purpose we need the following result.

Lemma 1.2. *Let $u(x)$ be two times continuously differentiable on $[x_0 - h, x_0 + h]$. Then there exists $\xi \in (x_0 - h, x_0 + h)$ such that*

$$\Delta_h u(x_0) = \Delta u(\xi) \,.$$

Proof. Special case of [60, Satz 3.4]. □

Let $\omega(v; \cdot) : [0, \infty) \to [0, \infty)$ be the *modulus of continuity* of v on $[a, b]$ (see e.g. [56]), i.e.,

$$\omega(v; \delta) := \sup\{|v(x_1) - v(x_2)| : x_1, x_2 \in [a, b], |x_1 - x_2| \leq \delta\} \,.$$

Then by using Lemma 1.2 and the definition of the modulus of continuity we have that

$$\begin{aligned} |\Delta_h u(x_0) - \Delta u(x_0)| &= |\Delta_h u(x_0) - \Delta u(\xi) + \Delta u(\xi) - \Delta u(x_0)| \\ &\leq |\Delta_h u(x_0) - \Delta u(\xi)| + |\Delta u(\xi) - \Delta u(x_0)| \\ &\leq 0 + \omega(\Delta u; h) \,. \end{aligned}$$

It is known that $\lim_{\delta \to 0} \omega(v; \delta) = 0$ if and only if $u(x)$ is uniformly continuous on $[a, b]$ [56]. Thus, if we restrict ourselves to functions u where Δu is uniformly continuous, then Δ_h is consistent. To make a qualitative statement about how fast $\Delta_h u$ approaches Δu, we have to make an additional assumption.

Lemma 1.3. *If $u \in C^4[0, 1]$ then for $x \in [h, 1 - h]$*

$$|\Delta_h u(x) - \Delta u(x)| \leq \frac{h^2}{12} \|u^{(4)}\|_\infty \,.$$

Proof. Using the Taylor expansion yields that there exists $\xi_- \in [x - h, x]$ and $\xi_+ \in [x, x + h]$ such that

$$u(x + h) = f(x) + \frac{u'(x)}{1!} h + \frac{u''(x)}{2!} h^2 + \frac{u^{(3)}(x)}{3!} h^3 + \frac{u^{(4)}(\xi_+)}{4!} h^4 \quad \text{and} \tag{1.10a}$$

$$u(x - h) = u(x) - \frac{u'(x)}{1!} h + \frac{u''(x)}{2!} h^2 - \frac{u^{(3)}(x)}{3!} h^3 + \frac{u^{(4)}(\xi_-)}{4!} h^4 \,. \tag{1.10b}$$

Substituting the Taylor expansions (1.10) into the discretized Laplace operator (1.7) gives

$$\Delta_h u(x) = u''(x) + \tfrac{1}{24} \left(u^{(4)}(\xi_-) + u^{(4)}(\xi_+) \right) h^2 \,.$$

Rearranging the terms and taking the absolute modulus then gives that

$$|\Delta_h u(x) - u''(x)| = \tfrac{1}{24} \left| u^{(4)}(\xi_-) + u^{(4)}(\xi_+) \right| h^2 \leq \tfrac{h^2}{12} \|u^{(4)}\|_\infty \,.$$ □

If we restrict ourselves to functions $u \in C^4[0, 1]$, then Lemma 1.3 shows that the difference between Δ_h and Δ can be bounded by a function proportional to h^2. We say that the discretization Δ_h is *consistent of order* 2.

1.2.3. Stability and Convergence

Consistency alone is not sufficient to show that $h \to 0$ implies that $u_h \to u$. We only have that if $u_h(x_i) = u(x_i)$ then $\Delta_h u_h \approx \Delta u$. What we need, however, is the reverse statement, because the solution u of the PDE and the solution u_h of the discretized PDE fulfill

$$\Delta u(x_i) = f(x_i) \quad \text{and} \quad \Delta_h u_h(x_i) = f(x_i)\,.$$

Thus, we have $\Delta u(x_i) = \Delta_h u_h(x_i)$ and want to show that $u(x_i) \approx u_h(x_i)$.

Let us assume that we have a consistent discretization, i.e.,

$$\|\Delta_h u - \Delta u\|_h \leq c(h)\,. \tag{1.9 revisited}$$

Then, since $\Delta u(x_i) = f(x_i) = \Delta_h u_h(x_i)$,

$$\|\Delta_h u - \Delta_h u_h\|_h \leq c(h)\,.$$

Since Δ_h is a linear operator,

$$\|\Delta_h (u - u_h)\|_h \leq c(h)\,.$$

By defining the error $e_h(x_i) := u(x_i) - u_h(x_i)$ we get

$$\|\Delta_h e_h\|_h \leq c(h)\,. \tag{1.11}$$

We have that $\Delta_h e_h \to 0$ if $h \to 0$. Thus, if we can show that $e_h \to 0$ if $\Delta_h e_h \to 0$ then we have our desired result. A *stable* discretization fulfills this property.

Definition 1.4. If there exists a constant $\epsilon > 0$ independent of h such that

$$\epsilon \|e_h\|_h \leq \|\Delta_h e_h\|_h \quad \text{for all } e_h,$$

then the discretization Δ_h is called *stable*.

The bound (1.11) implies for a stable discretization

$$\|e_h\|_h \leq \tfrac{1}{\epsilon} c(h)\,. \tag{1.12}$$

Thus, if $c(h) \to 0$ then the norm of the error goes to zero as well and therefore $u_h(x_i) \to u(x_i)$. In other words, consistency and stability imply that the approximation converges to the solution. Note that it is important that the constant ϵ is independent of h. If it decreases with h then $\frac{1}{\epsilon} c(h)$ might go to infinity for $h \to 0$ even if $c(h) \to 0$.

We show that the discretization (1.7) is stable, i.e. the bound (1.12) holds.

Lemma 1.5. *Let Δ_h be defined in* (1.7) *and $e_h : \overline{\Omega}_h \to \mathbb{R}$. If $e_h(x_0) = e_h(x_n) = 0$ then*

$$\|e_h\|_h \leq C \|\Delta_h e_h\|_h\,,$$

where $C > 0$ is independent of h.

Proof. We prove the assertion in three steps.

1. We show the *discrete maximum/minimum principle*, i.e., for a function $u_h : \overline{\Omega}_h \to \mathbb{R}$ with $u_h(x_0) = u_h(x_n) = 0$ the following statements hold.
   a) If $[\Delta_h u_h](x_i) \leq 0$ for $i = 1, \ldots, n-1$, then
   $$\max_i u_h(x_i) \leq 0.$$
   b) If $[\Delta_h u_h](x_i) \geq 0$ for $i = 1, \ldots, n-1$, then
   $$\min_i u_h(x_i) \geq 0.$$
2. We show using the discrete maximum principle that if we can find a function $w_h : \overline{\Omega}_h \to \mathbb{R}$ with $w_h(x_0) = w_h(x_n) = 0$ such that
   $$[\Delta_h e_h - \Delta_h w_h](x_i) \leq 0 \quad \text{for} \quad i = 1, \ldots, n-1 \quad \text{and} \tag{1.13a}$$
   $$[\Delta_h e_h + \Delta_h w_h](x_i) \geq 0 \quad \text{for} \quad i = 1, \ldots, n-1\,, \tag{1.13b}$$
   then
   $$-w_h(x_i) \leq e_h(x_i) \leq w_h(x_i) \quad \text{for} \quad i = 1, \ldots, n-1. \tag{1.14}$$
   This inequality implies that
   $$\|e_h\|_h \leq \|w_h\|_h\,. \tag{1.15}$$
3. We show that there exists a function $w_h : \overline{\Omega}_h \to \mathbb{R}$ with $w_h(x_0) = w_h(x_n) = 0$ that fulfills (1.13) and
   $$\|w_h\|_h \leq C\|\Delta e_h\|_h\,,$$
   where the constant $C > 0$ is independent of the step-size h. Combining the last equation with (1.15) proves the assertion.

(1) We prove Statement 1a). Let j be the index such that $u_h(x_j)$ is the maximum value of u_h. If $j = 0$ or $j = n$, i.e., the maximum value is located at the boundary, we have that the maximum value of u_h is zero, because the u_h is zero at the boundary by definition. Hence, in this case the statement is true.

Let us consider the other case; assume that j, the index of the maximum value of $u(x_j)$, is an index of an interior point. From the definition of Δ_h (1.7) and the assumption that $[\Delta_h u_h](x_j) \leq 0$, we have that

$$[\Delta_h u_h](x_j) = \frac{-u_h(x_{j-1}) + 2u_h(x_j) - u_h(x_{j+1})}{h^2} \leq 0\,.$$

Rewriting the last equation yields that

$$u_h(x_j) \leq \frac{u_h(x_{j-1}) + u_h(x_{j+1})}{2}\,.$$

In other words, $u_h(x_j)$ is smaller than the average of the function value at x_{j-1} and x_{j+1}. In combination with the fact that $u_h(x_j)$ is the maximum value of u_h we conclude that $u_h(x_{j-1}) = u_h(x_j) = u_h(x_{j+1})$.

Now $j-1$ and $j+1$ are also indices which correspond to maximum values of u_h. Repeating the above argument, it follows that u_h must be constant, and as the boundary values are zero, the whole function must be equal to zero. Thus, we have proven Statement 1a).

Statement 1b) follows from applying Statement 1a) to the function $-u_h$.

(2) Assume we have a function w_h that fulfills (1.13a). Using the linearity of Δ_h we obtain that

$$[\Delta_h(e_h - w_h)](x_i) \leq 0 \quad \text{for} \quad i = 1, \ldots, n-1 \,.$$

Hence, from the discrete maximum principle, it follows that

$$e_h(x_i) - w_h(x_i) \leq 0$$

and

$$e_h(x_i) \leq w_h(x_i) \,.$$

In a similar way, it follows from (1.13b) and the discrete minimum principle that

$$-w_h(x_i) \leq e_h(x_i)$$

and combining the previous two equations then gives (1.14).

(3) To find a function w_h that fulfills (1.13), we just have to find a function w_h whose second derivative is large enough and which is zero at the boundary. We choose

$$w_h(x) := \tfrac{\|\Delta_h e_h\|_h}{2} \cdot (x - x^2) = \tfrac{\|\Delta_h e_h\|_h}{2} \cdot (\tfrac{1}{4} - (x - \tfrac{1}{2})^2),$$

The function w_h is a polynomial of degree two, and the operator Δ_h computes the exact derivative for all polynomials of degree less or equal than two, because it was constructed using an interpolation polynomial of degree two. Hence[1],

$$[\Delta_h w_h](x) = [\Delta w_h](x) = \|\Delta_h e_h\|_h \,.$$

This equation implies that w_h fulfills (1.13).

It remains to bound the values $w_h(x_i)$ for $i = 1, \ldots, n-1$. From the definition of w_h it is easy to see that w_h has its largest value at $x = \frac{1}{2}$. Hence,

$$w_h(x_i) \leq w_h(\tfrac{1}{2}) = \tfrac{\|\Delta_h e_h\|_h}{8} \,,$$

and setting $C = \frac{1}{8}$ proves the assertion. □

The stability estimate from Lemma 1.5 applies only to the Poisson equation, however, only small changes to the proof are needed, to prove stability estimates for other equations [46]. Combining the results from this section gives the following theorem.

Theorem 1.6. *Let $u \in C^4$ be the solution of the Poisson equation and u_h the approximation given by* (1.8)*. Then there exists a constant $\widetilde{C}$ that is independent of h such that the error $e_h = u - u_h$ fulfills*

$$\|e_h\|_h \leq \widetilde{C} h^2 \,.$$

Proof. The assertion follows from Lemma 1.3, Lemma 1.5, and the bound (1.12). □

Finally, note that Lemma 1.5 gives a lower bound for the modulus of the smallest eigenvalue of the matrix representing Δ_h. Thus, we know that Δ_h is represented by a regular matrix and therefore the linear system (1.8) has a unique solution.

The finite difference has been known for a long time and it has many applications. For a more in-depth introduction and further applications see, e.g., [46, 51, 60].

[1] This formula can also be shown by a straightforward, but lengthy calculation.

1.3. Multigrid Methods

The numerical approximation of the boundary value problem (1.4) requires the solution of the linear system (1.8). There are two classes of methods for solving linear systems—*direct* and *iterative*.

Direct methods compute the exact solution (up to machine precision); *iterative methods* compute an approximation [30, 59]. A direct method computes the solution in one monolithic step; an iterative method starts with an initial guess u_0 of the solution x and successively computes the *iterates* $u_1, u_2, u_3, \ldots$. In every step the current iterate is improved such that u_k converges to the solution u for $k \to \infty$. Therefore, the runtime of an iterative method depends on the approximation accuracy, as the iteration can be stopped when the approximation is sufficient. If machine precision accuracy is not needed then in many applications iterative methods are faster than direct methods.

This is, for example, the case when solving the discrete Poisson equation (1.8). The solution of the linear system is an approximation to the exact solution; the difference between the exact solution and the approximation is called the *discretization error*. The difference of the exact solution of the *linear system* and an approximation is called *algebraic error*. If the algebraic error is small in comparison to the discretization error, the overall error is nearly unaffected. Thus, an iterative method can speed up the solution process.

It can even be reasonable to make the algebraic error of the same order of magnitude as the discretization error, and reduce the discretization error a little bit, to get the overall best performance.

As mentioned at the beginning of this chapter, multigrid methods are best explained by constructing a multigrid method for a simple problem. For this reason we shall derive a multigrid method for the discrete Poisson equation (1.8).

1.3.1. A Relaxation Scheme

A vital component of multigrid methods are simple iterative methods like the weighted Jacobi method. To introduce this method, we consider the linear system $Au = f$ or equivalently $f - Au = 0$, where $A \in \mathbb{R}^{n\times n}$ and $u \in \mathbb{R}^n$ are given and $u \in \mathbb{R}^n$ is the desired solution. For an arbitrary vector $w \in \mathbb{R}^n$ the *residual* r_w fulfills

$$r_w := f - Aw\,. \tag{1.16}$$

Thus, w equals the solution u if and only if its residual is zero, and w is close to u if the residual is small. The main idea of the Jacobi method is to successively reduce the residual.

The method starts with an arbitrary vector $u_0 \in \mathbb{R}^n$. In every step the method takes the current vector u_k and produces a new vector u_{k+1}.

The n entries of the solution u are determined by the n equations

$$\sum_{j=1}^{n} a_{ij} \cdot u_j = f_i \qquad (i = 1, \ldots, n).$$

When $n \gg 1$ finding n variables that satisfy n equations is difficult; finding one variable that satisfies one equation is easier. In every step the Jacobi method constructs corrections

$v_i \in \mathbb{R}^n$ $(i = 1, \ldots, n)$. The new iterate is

$$u_{k+1} = u_k + \omega \cdot (v_1 + \cdots + v_n), \tag{1.17}$$

where $\omega \in (0, 2)$ is the *weight*. The corrections have the form $v_i = \xi_i e_i$ where e_i is the ith unit vector and $\xi_i \in \mathbb{R}$. The value ξ_i is chosen such that the ith component of the residual of $u_k + v_i$ is zero, i.e.,

$$r_{(u_k+v_i),i} = f_i - \left(\sum_{i=1}^{n} a_{ij} \cdot u_j \right) - a_{ii}\xi_i = 0 \,.$$

Thus, the size of the ith correction ξ_i depends only on *one* equation; its solution is $\xi_i = \frac{1}{a_{ii}} r_{u_k,i}$. A step of the Jacobi method then reads

$$u_{k+1,i} = u_{k,i} + \frac{\omega}{a_{ii}} r_{u_k,i} \,.$$

We give the whole method in the Jacobi-Iteration procedure, where we use the pseudocode notation from [16].

Jacobi-Iteration()

```
1  while u is a bad approximation
2      for i ← 1 to n
3          u_{k+1,i} ← u_{k,i} + (ω/a_{ii}) r_{u_k,i}
```

Let $D := \operatorname{diag}(a_{11}, \ldots, a_{nn})$ then we can formulate the method more compactly by using the following:

$$u_{k+1} = u_k + \omega D^{-1} \cdot r_{u_k} \,. \tag{1.18}$$

The weight ω is used to fine-tune the corrections v_i. For example for the discrete Poisson equation (1.8), the corrections are somewhat too large. Thus, choosing $\omega = 0.8$ improves the convergence rate.

It remains to specify when the iterate u_k is a bad approximation. The iteration u_k is a bad approximation if the difference of the iterate u_k and the solution u is large, where we measure the site of the difference in some norm, e.g., the Euclidean 2-norm. However, as we do not have access to the solution u, we generylly cannot measure the difference of u_k and u.

There are different ways to estimate the difference between the iterate and the solution. One way is to measure the size of the residual r_{u_k}. When the residual is small, we expect u_k to be close to u; remember, $u_k = u$ only if $\|r_{u_k}\| = 0$.

Finally note that the convergence of the sequence $u_0, u_1, u_2, \ldots$ to the solution u depends on the properties of the matrix A and the initial value u_0 [59].

1.3.2. The Error Iteration

The first step in introducing multigrid methods is to describe iterative methods more abstractly. Assume u_k, the kth iterate, is an approximation to the solution u of the linear system $Au = f$. The (algebraic) error of the kth iterate is defined as

$$e_k := u - u_k \,. \tag{1.19}$$

Thus, a small value for $\|e_k\|$ (for an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^n$) implies that u_k is close to the solution u. Therefore, iterative methods aim to reduce the norm of the error. We can rearrange (1.19) to get that

$$u = u_k + e_k\,,$$

which means that we can compute the solution if we know the error. Although in general the error is not easily available, an approximation to the error $\tilde{e}_k \approx e_k$ might be available. If the approximation $\tilde{e}_k$ is sufficiently close to e_k then

$$u_{k+1} := u_k + \tilde{e}_{k+1} \tag{1.20}$$

is a better approximation than u_k, and the error e_{k+1} is smaller than e_k. We can repeat this idea to successively obtain better approximations of the solution. The procedure Error-Iteration describes this process.

Error-Iteration()

```
1  choose u
2  while u is a bad approximation
3      ẽ ← approximation of the error
4      u ← u + ẽ
5  return u
```

To use the Error-Iteration procedure we need a way to compute approximations of the error. Consider

$$Ae_k = A(u - u_k) = f - Au_k\,.$$

We see that the right hand side of this equation is the residual (1.16) w.r.t. u_k. Thus

$$Ae_k = r_{u_k}\,. \tag{1.21}$$

We can multiply this equation by A^{-1} from the left to obtain

$$e_k = A^{-1} r_{u_k}\,. \tag{1.22}$$

The residual r_{u_k} is easily computed. Nevertheless solving (1.22) for e_k is not practical, as it is as expensive as the original problem. Though if we have a matrix B^{-1} which is an approximation to A^{-1} and $B^{-1} r_{u_0}$ is easy to compute, then we can replace A^{-1} by B^{-1} in (1.22) to get the error approximation

$$\tilde{e}_k = B^{-1} r_{u_k}\,. \tag{1.23}$$

By using this equation in the *error correction step* (1.20),

$$u_{k+1} = u_k + \tilde{e}_k = u_k + B^{-1} r_{u_k} = u_k + B^{-1}(f - Au_k). \tag{1.24}$$

Note that if the error correction step has this shape, i.e., the approximation of the error is obtained by multiplying the residual with a matrix, we call the iterative method *stationary*.

For example, the Jacobi method is a stationary method, as we can see from comparing (1.18) and (1.24). These two equations give that

$$B^{-1} = \omega D^{-1} \tag{1.25}$$

in the case of the Jacobi method.

It is important to know, how the error changes from one step of the iteration to the next. The new error after one step of the iteration is

$$e_{k+1} = u - u_{k+1} = u - (u_k + \tilde{e}_k) = e_k - \tilde{e}_k \,. \tag{1.26}$$

In other words, the error approximation $\tilde{e}_k$ is removed from the error.

1.3.3. The Jacobi Method and the Poisson Equation

The second step in introducing multigrid methods is the following crucial observation. Consider the boundary value problem (1.4) with the functions

$$f(x) = (12\pi)^2 \cdot \sin(12\pi \cdot x), \quad u(0) = 0, \quad u(1) = 10 \,.$$

In this case the solution of the continuous equation is

$$u(x) = \sin(12\pi \cdot x) + 10 \cdot x.$$

We approximate the solution u by applying the Jacobi method to the linear system (1.8) with initial iterate $u_0 = 0$. In Figure 1.4a we see the initial iterate u_0 and in Figure 1.4b the initial (algebraic) error e_0. As $u_0 = 0$ the initial error equals the solution of the linear system and is therefore a good approximation of the solution u.

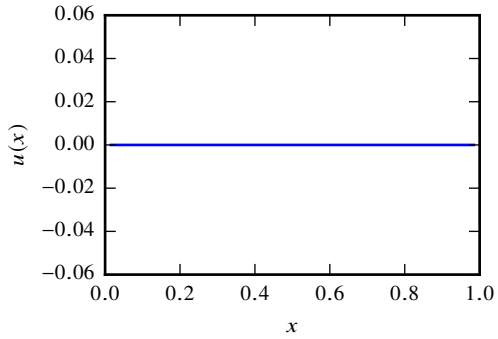
After 20 iterations of the Jacobi method we inspect the iterate u_{20} (Figure 1.4c) and the error e_{20} (Figure 1.4d). The ripples of the solution are already visible in the iterate u_{20}. The smooth increase from left to right, however, is not captured well, which is also visible in the error e_{20}; it is slowly varying. Thus, the Jacobi method is reducing oscillatory part of the error well and the slowly varying part of the error remains. We call an iterative method with such a behavior a *smoother* and one iteration of the method a *smoothing step*.

Now take a look at the 2-norm of the residuals and errors during the iteration (Figure 1.5). In the beginning the residual norm and error norm are strongly reduced; in the end only slightly. Thus, the Jacobi method is loosing efficiency when the error is slowly varying.

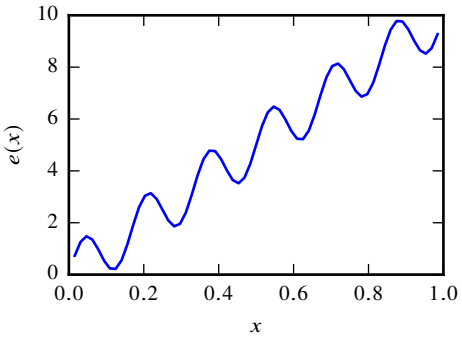
1.3.4. Coarse Approximations

The third step in introducing multigrid methods is to describe *coarse approximations*. For now assume we have a continuous operator $\mathcal{A}$, e.g., the Laplace operator Δ that maps functions from $e : \Omega \to \mathbb{R}$ to functions $r : \Omega \to \mathbb{R}$, and we want to find an approximation for the solution e of the equation

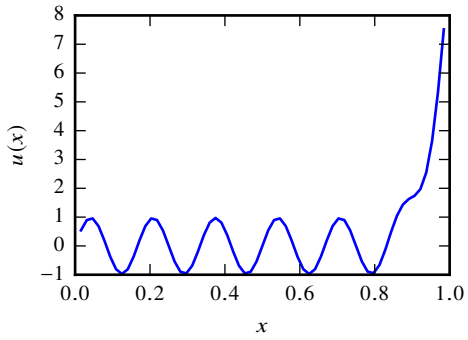
$$\mathcal{A}e = r \,. \tag{1.27}$$



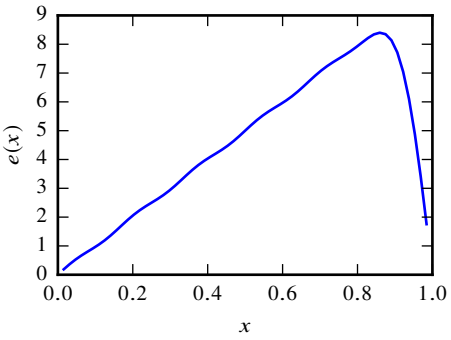
(a) The initial iterate u_0. It is equal to zero.



(b) The initial algebraic error e_0. It is close to the continuous solution.

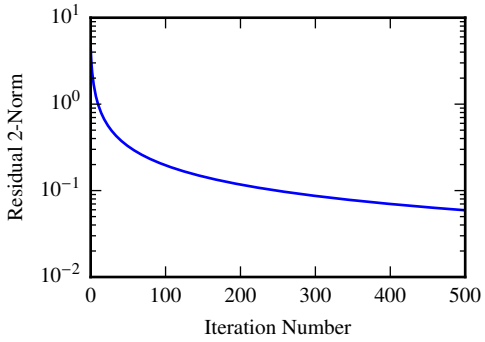


(c) After 20 steps the iterate u_{20} captures the oscillatory components of the solution well.

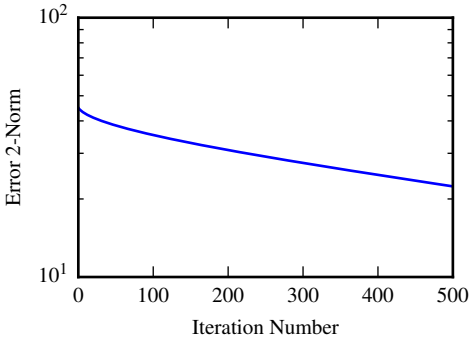


(d) The algebraic error after 20 iterations e_{20} is slowly varying.

Figure 1.4.: Jacobi method applied to the Poisson equation.



(a) Residual.



(b) Error.

Figure 1.5.: Convergence of the Jacobi method for the Poisson equation.

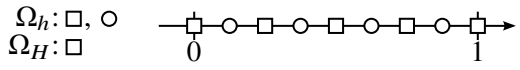


Figure 1.6.: Coarse and fine grid: The squares are in $\overline{\Omega}_h$ and $\overline{\Omega}_H$, while the circles are only in $\overline{\Omega}_h$.

This approximation should be the solution $e_h : \overline{\Omega}_h \to \mathbb{R}$ of

$$A_h e_h = r_h \tag{1.28}$$

where A_h is a discretization of $\mathcal{A}$ on $\overline{\Omega}_h$ and $r_h(x) = r(x)$.

If e_h is slowly varying, we can compute a good approximation $\tilde{e}_h$ to e_h, by solving a linear system, having less degrees of freedom than (1.28). To show this let $H := 2h$ and consider the *coarse grid* $\overline{\Omega}_H$. The step-size of $\overline{\Omega}_H$ is twice as large as that of $\overline{\Omega}_h$, thus $\overline{\Omega}_H$ contains every second point from $\overline{\Omega}_h$; see Figure 1.6. Furthermore, consider the approximation e_H on $\overline{\Omega}_H$ of e, given by

$$A_H e_H = r_H, \tag{1.29}$$

where A_H is a discretization of $\mathcal{A}$ and $r_H(x) = r(x)$.

For $x \in \overline{\Omega}_H$ we have that $e_H(x) \approx e_h(x)$, but for $x \in \overline{\Omega}_h \setminus \overline{\Omega}_H$ the value $e_H(x)$ is not defined. As e_h is slowly varying, i.e., if x is close to x' then $e_h(x)$ is close to $e_h(x')$, we can, however, approximate $e_h(x)$ by linear interpolation between $e_H(x - h)$ and $e_H(x + h)$.

To sum it up, let P be the interpolation operator given by

$$[Pe_H](x) = \begin{cases} e_H(x) & \text{if } x \in \overline{\Omega}_H, \\ \frac{1}{2}e_H(x-h) + \frac{1}{2}e_H(x+h) & \text{if } x \notin \overline{\Omega}_H \end{cases} \quad \text{for } x \in \overline{\Omega}_h \,.$$

Then $\tilde{e}_h := Pe_H$ is an approximation of e_h.

Note that $r_H(x) = r(x) = r_h(x)$ for $x \in \overline{\Omega}_H$. Thus, if we define

$$[Rr_h](x) = r_h(x) \quad \text{for} \quad \overline{\Omega}_H \tag{1.30}$$

then

$$\tilde{e}_h = PA_H^{-1}Rr_h \tag{1.31}$$

is a *coarse approximation* to e_h. As the size of A_H is only half the size of A_h it is cheaper to compute than e_h.

The restriction operator R defined in (1.30) is called *injection*. There are other types of restriction that are sometimes useful. For example, the full weighting restriction operator, which will be introduced in Section 1.3.7.

Note that in the computation of the coarse approximation (1.31) neither the continuous functions e and r nor the continuous operator $\mathcal{A}$ are directly involved. They were only necessary to establish the connection between the fine (1.28) and the coarse linear system (1.29). Therefore, they are not needed for practical computation.

1.3.5. Two Grids

The fourth step is to combine the ideas and observations from the previous three sections to construct the *two-grid method*. The Jacobi method solves the discrete Poisson equation inefficiently; it reduces only the oscillatory parts of the error well while the slowly varying part remains. Therefore, after applying a smoothing method like Jacobi, we can compute a coarse approximation (1.31) of the error, which we use to remove the slowly varying part from the error by performing an error correction step (1.20). The procedure of performing an error correction with a coarse approximation is called *coarse grid correction*.

The two-grid method combines two components—the smoother and the coarse grid correction. Those two are applied alternately. The method performs ν_1 smoothing steps, e.g., ν_1 steps of the Jacobi method. Then it performs η coarse grid corrections and further ν_2 smoothing steps. The whole procedure is repeated until the approximation is sufficient. The method is given by the Two-Grid-Iteration procedure. Typical values are $\nu_1 = 2$, $\nu_2 = 2$ and $\eta = 1$.

```
Two-Grid-Iteration()
1  choose u
2  while u is a bad approximation
3      perform ν1 smoothing steps on u
4      for k ← 1 to η
5          ẽ ← coarse approximation of the error
6          u ← u + ẽ
7      perform ν2 smoothing steps on u
```

1.3.6. Multiple Grids

We can now introduce the multigrid method by modifying the two-grid method. The two-grid method iteratively computes a solution on the fine grid. This computation involves the solution of a linear system on the coarse grid in every iteration. The coarse grid system is smaller than the fine grid system, but still expensive to solve. However, as the coarse grid system has the same structure as the fine grid system, we can use the two-grid method to solve the coarse grid system, as well. By that we obtain a three-grid method. We can repeat this idea until the coarsest grid is so small that the coarse grid approximation can be computed easily.

As the two-grid method solves the coarse grid system exactly, it would be reasonable to assume that we need many iterations of the coarse-grid solver to get a good coarse grid error approximation. We need, however, only an approximation to the coarse grid error. Thus, it is sufficient to make only a few iterations of the coarse grid solver.

Let h_1 be the step-size of the finest grid. Assume we have $L \in \mathbb{N}$ grids, with the step-size $h_{\ell+1} := h_\ell$, for $\ell = 1, \ldots, L-1$. For every grid we have the discretization $A_{h_\ell} : \overline{\Omega}_{h_\ell} \to \overline{\Omega}_{h_\ell}$ and for all except the coarsest grids we have interpolations $P_\ell : \overline{\Omega}_{h_{\ell+1}} \to \overline{\Omega}_{h_\ell}$ and restrictions $R_\ell : \overline{\Omega}_{h_\ell} \to \overline{\Omega}_{h_{\ell+1}}$. The multigrid method is given by two parts. The Multigrid-Iteration

procedure repeatedly calls the MULTIGRID-CYCLE procedure. The MULTIGRID-CYCLE then recursively calls itself to correct the error on the different grid levels.

```
MULTIGRID-ITERATION()
1  choose u
2  while u is a bad approximation
3      u ← MULTIGRID-CYCLE(f, u, 1)
```

```
MULTIGRID-CYCLE(f, u, ℓ)
 1  if ℓ = L
 2      return A_ℓ^{-1} f
 3  else perform ν_1 smoothing steps on u
 4      e_{ℓ+1} ← R(f − A_ℓ u)
 5      r_{ℓ+1} ← 0
 6      for i ← 1 to η
 7          e_{ℓ+1} ← MULTIGRID-CYCLE(r_{ℓ+1}, e_{ℓ+1}, ℓ + 1)
 8      u ← u + P e_{ℓ+1}
 9      perform ν_2 smoothing steps on u
10      return u
```

Depending on the number of coarse grid corrections that are performed in every iteration this method has further names. In case of $\eta = 1$ it is called a V-cycle. In case of $\eta = 2$ it is called a W-cycle.

1.3.7. Weighted Restriction

Using the injection operator for the restriction sometimes leads to a method that converges badly. This is for example the case when the function that is restricted is very oscillatory, as we shall see.

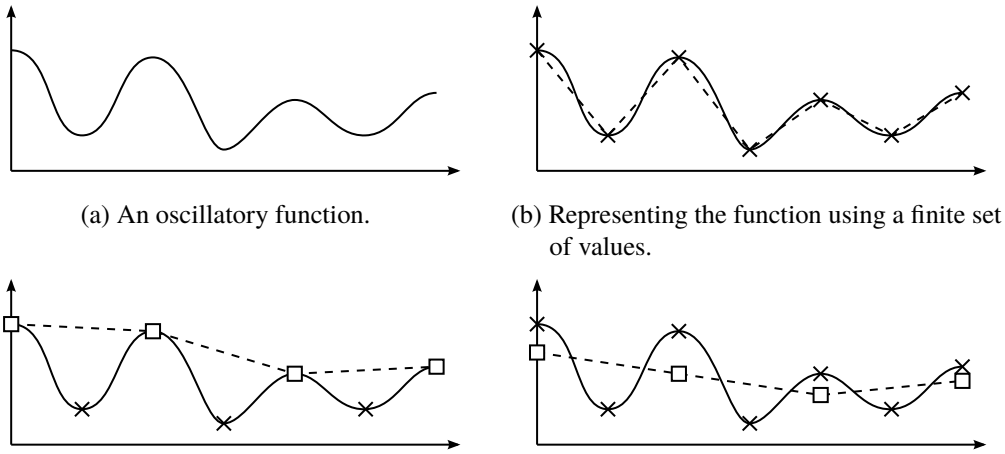
Restriction operators were introduced in the section about the coarse approximations, Section 1.3.4. In that section we used the fact that we want to solve a linear system that stems from a continuous equation

$$\mathcal{A}e = r\,. \tag{1.27 revisited}$$

The continuous residual r is approximated on two different grids: on the fine grid by r_h and on the coarse grid by r_H. The restriction operator R describes the relation between the two residuals, i.e., $Rr_h = r_H$. We now want to determine the quality of a restriction operator.

A suitable restriction operator maps a good approximation r_h of r to a good approximation r_H. Consequently, to determine the quality of a restriction operator, we need to be able to determine the quality of the approximation r_H. The vector r_H contains values that correspond to function values at certain points. If we interpolate those points we obtain a function which can be compared to the continuous residual r. Let us consider an example.

Assume that the residual r is the function shown in Figure 1.7a. We approximate the function r by a finite set of points, which is shown in Figure 1.7b. Using linear interpolation



(a) An oscillatory function.

(b) Representing the function using a finite set of values.

(c) Restrict the function to a coarse set of values using injection.

(d) Restrict the function to a coarse set of values using full weighting.

Figure 1.7.: Restriction of an oscillatory function.

we see that the fine grid approximation is a good representation of the continuous residual function. Figure 1.7c shows the coarse grid approximation obtained by the injection operator; demonstrating that this approximation does not represent the continuous function well. In this case, the *full weighting restriction* will be a better choice.

The idea of the full weighting restriction is to compute the restricted value at some point by a weighted sum of function values at neighboring grid points. The weights can be derived from the linear interpolation in the following way. As the interpolation is a linear operator, it can be written in matrix form, i.e., we have

$$u_{h,i} = [Pu_H]_i = \sum_{j=1}^{n} p_{ij} \cdot u_{H,j}\,,$$

where p_{ij} are the entries of the matrix representing the interpolation P. An interpretation of this formula is that the entry p_{ij} describes the influence of the jth entry of u_H on the i-th entry of u_h. It is reasonable to assume that we can reverse this relation when constructing a restriction operator. Thus, for the restriction we assume that the entries $p_{1,j}, \ldots, p_{n,j}$ describe the relative importance of $r_{h,1}, \ldots, r_{h,n}$ when computing $r_{H,j}$. More precisely we set

$$r_{H,j} = w_j \sum_{i=1}^{n} p_{ij} \cdot r_{h,i}\,,$$

where w_j is a suitable scaling factor. This equation can be written as

$$r_H = WP^T r_h\,,$$

where $W = \mathrm{diag}(w_1, \ldots, w_n)$ and P^T is the transpose of P. It remains to determine the scaling factors.

We want to choose w_j such that the constant vector $(1, \dots, 1)^T$ is restricted to the constant vector $(1, \dots, 1)^T$. This constraint implies

$$1 = w_j \sum_{i=1}^{n} p_{ij} \cdot 1 \,,$$

and consequently

$$w_j = \left(\sum_{i=1}^{n} p_{ij} \right)^{-1} .$$

Summing it up, we make the following definition.

Definition 1.7. Let $P \in \mathbb{R}^{n \times m}$ be an interpolation operator. The *restriction operator* $R \in \mathbb{R}^{m \times n}$ *induced by* P is given by

$$R = WP^T \,,$$

where W is the diagonal matrix with diagonal entries

$$w_j = \left(\sum_{i=1}^{n} p_{ij} \right)^{-1} .$$

We can now apply this construction to the linear interpolation. The 1D linear interpolation in matrix form is

$$P = \begin{pmatrix} \ddots & & & & & \\ & 1 & & & & \\ & \frac{1}{2} & \frac{1}{2} & & & \\ & & 1 & & & \\ & & \frac{1}{2} & \frac{1}{2} & & \\ & & & 1 & & \\ & & & & \ddots & \end{pmatrix} .$$

From the matrix form of P we obtain

$$R = \begin{pmatrix} \ddots & & & & & \\ & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & & \\ & & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \\ & & & & & \ddots \end{pmatrix} ,$$

which is the *full weighting restriction* in 1D. Figure 1.7d shows the result of applying the full weighting restriction to our example problem. We can see that the function is much better approximated than when the function is restricted using injection. Note, however, that there are applications where injection is a better restriction than the full weighting restriction.

1.3.8. The General Multigrid Method

Until now we have only discussed a multigrid method for the 1D Poisson equation. Multigrid methods, however, are a class of methods, and to apply a multigrid method to a specific problem, the method needs to be tailored towards this problem. So how can we apply multigrid methods to other problems?

This can be done by starting with the Multigrid-Iteration procedure. This procedure uses a smoother and the operators A_ℓ, R_ℓ, and P_ℓ. Choosing these components in a way that matches the problem then gives a multigrid methods.

This type of multigrid method is called *multigrid iteration*. There are other types of multigrid methods. For example the full multigrid method [32, 68] or K-cycles [53]. However, we will not address them here.

1.3.9. The Error Propagator

Multigrid methods are best analyzed by considering the behavior of the error during the iteration. In many iterative methods, the error e_k before the $(k+1)$th iteration and the error e_{k+1} after the iteration can be related by

$$e_{k+1} = E e_k \,,$$

where E is a matrix that does not depend on the iteration number or the iterates. In this case, the whole method can be analyzed in terms of the operator E, which is called the *error propagator* of the method. Furthermore, we can use the error propagator to compute the error of any iterate given the initial error e_0, by computing

$$e_k = E e_{k-1} = E^2 e_{k-2} = \cdots = E^k e_0 \,.$$

We shall now show the existence of error propagators for different methods.

Stationary Methods

First, we show that stationary iterative methods have an error propagator. From (1.26) we know that after one iteration, the error e_k becomes

$$e_{k+1} = e_k - \tilde{e}_k \,.$$

Recall that the error approximation $\tilde{e}_0$ in a stationary method is given by

$$\tilde{e}_k = B^{-1} r_{u_k}, \qquad \text{(1.23 revisited)}$$

where B^{-1} is a matrix that approximates A^{-1}. Furthermore

$$A e_k = r_{u_k} \,. \qquad \text{(1.21 revisited)}$$

Combining the last three equations yields

$$e_{k+1} = e_k - B^{-1} A e_k = (I - B^{-1} A) e_k \,.$$

It follows that the matrix

$$E = I - B^{-1}A \tag{1.32}$$

is the error propagator of the stationary iterative method given by the matrix B^{-1}.

Recall that for the Jacobi method

$$B^{-1} = \omega D^{-1}. \tag{1.25 revisited}$$

Thus, the error propagator of the weighted Jacobi method is

$$E = I - \omega D^{-1}A. \tag{1.33}$$

The Two-Grid Method

Second, we show that the entire two-grid method has an error propagator.

Proposition 1.8. *Let S be the error propagator of the smoother, ν_1 the number of pre-smoothing steps, and ν_2 the number of post-smoothing steps. Furthermore, let P be an interpolation operator and R a restriction operator. Then the error propagator of the coarse grid correction is*

$$E_{\text{CGC}} = I - PA_H^{-1}RA_h, \tag{1.34}$$

and the error propagator of the two-grid method is

$$E_{\text{TG}} = S^{\nu_2}(I - PA_H^{-1}RA_h)S^{\nu_1}. \tag{1.35}$$

Proof. The coarse grid error approximation in the two-grid method is

$$\tilde{e}_h = PA_H^{-1}Rr_h. \tag{1.31 revisited}$$

The residual and the error are related by $A_h e_0 = r_h$ (cf. (1.21)). Thus

$$\tilde{e}_h = PA_H^{-1}RA_h e_0\,.$$

As the error approximation $\tilde{e}_h$ is used in an error correction step, the error after the coarse grid correction is, by (1.26), given by

$$e_1 = e_0 - \tilde{e}_h = e_0 - PA_H^{-1}RA_h e_0 = (I - PA_H^{-1}RA_h)e_0\,.$$

Thus, $(I - PA_H^{-1}RA_h)$ is the error propagator of the coarse grid correction. The two-grid method applies ν_1 smoothing steps before the coarse grid correction and ν_2 smoothing steps afterwards, which yields (1.35). □

The Multigrid Method

Third, we show that even the whole multigrid method has an error propagator. To prove this assertion, we need to generalize the result of Proposition 1.8. As the multigrid method solves the coarse grid recursively, we need to allow for the coarse grid to be approximated by a stationary method, instead of solving it exactly.

Lemma 1.9. *Assume that the two-grid method solves the coarse grid system inexactly with a stationary method, whose error propagator is E_S. Furthermore, assume that the iteration is started with zero as initial guess. Then the two-grid method is stationary and has the error propagator*

$$E_{TG} = S^{\nu_2}(I - P(I - E_S)A_H^{-1}RA_h)S^{\nu_1}\,. \tag{1.36}$$

Proof. In the original two-grid method, we compute the coarse approximation, by computing $A_H^{-1}f$, for a certain right hand side f. Here, we replace this computation, by the application of one iteration of an iterative method.

Recall from (1.24) that the error correction step of a stationary method is given by

$$u_1 = u_0 + B^{-1}(f - Au_0)\,.$$

This method has the error propagator

$$E = I - B^{-1}A\,. \tag{1.32 revisited}$$

When we are given the error propagator, then we can obtain B^{-1} by

$$B^{-1} = (I - E)A^{-1}\,,$$

and thus the error correction reads

$$u_1 = u_0 + (I - E)A^{-1}(f - Au_0)\,.$$

Replacing E by the stationary coarse grid error propagator E_S and A by the coarse system matrix A_H, and using that $u_0 = 0$ by assumption, we get

$$u_1 = (I - E_S)A_H^{-1}f\,.$$

Hence, in this case, the coarse grid inverse A_H^{-1} is approximated by $(I - E_S)A_H^{-1}$, and we obtain the result by replacing A_H^{-1} by $(I - E_S)A_H^{-1}$ in (1.35). □

This lemma allows us to derive the error propagator for a multigrid iteration with L levels.

Theorem 1.10. *The error propagator E_1 of the multigrid method is given by the recursive formula*

$$\begin{aligned} E_i &= S_i^{\nu_2}(I - P(I - E_{i+1}^{\eta})A_{i+1}^{-1}RA_i)S_i^{\nu_1} && \text{for } i < L,\\ E_i &= 0 && \text{for } i = L. \end{aligned} \tag{1.37}$$

Proof. The assertion is shown by proving by induction that E_i is the error propagator of the multigrid method for the levels $i, \dots, L$. We start with $i = L$. Then E_i is the error propagator on the coarsest level. On this level the system is solved exactly; implying that no error is made and therefore $E_i = 0$, when $i = L$.

Now assume that $i < L$, thus by the induction hypothesis E_{i+1} is the error propagator of the multigrid method for the levels $i + 1, \dots, L$, and E_{i+1}^{η} represents η iterations of the coarse grid iteration. The multigrid method for the levels $i, \dots, L$ is a two-grid method, where the coarse grid is solved by the multigrid method for the levels $i + 1, \dots, L$. Thus, E_i is given by (1.36), where the error propagator of the coarse grid solver $E_S = E_{i+1}$. □

1.3.10. Galerkin Coarse Approximation

There are applications where it is not feasible to obtain the coarse approximation using a discretization of the continuous problem on a coarse grid. For instance, this is the case if the discretization requires a certain minimal resolution, the construction of the discretization on a coarse grid is very complicated, or the discretization is very expensive. The *Galerkin coarse approximation* avoids these problems as it requires only a restriction and an interpolation operator for its construction. Furthermore, the Galerkin coarse approximation appears naturally in multigrid methods for variational problems, e.g., the finite element method [10, 41], and it is optimal in the sense that it minimizes the error in the energy norm [66, Corollary A.2.1].

To approximate the action of the fine grid operator on a coarse grid function it is reasonable to just interpolate the coarse grid function, then apply the fine grid operator, and then restrict the result back to the coarse grid. The operator that performs this procedure is called the Galerkin coarse approximation.

Definition 1.11. Let $A_h \in \mathbb{R}^{n_1 \times n_1}$ be a fine grid operator, $R \in \mathbb{R}^{n_2 \times n_1}$ a restriction, and $P \in \mathbb{R}^{n_1 \times n_2}$ an interpolation operator. The *Galerkin coarse approximation* A_H is given by

$$A_H = R A_h P \, .$$

Another useful property of the Galerkin coarse approximation is that it simplifies the computation of the induced restriction operator. To see this, assume we have two multigrid methods. The only difference between the two is the first uses the restriction that is induced by the interpolation operator (Definition 1.7), while the second uses the transpose of the interpolation as restriction. If both methods use the Galerkin coarse approximation then the two methods are identical, as we shall show.

Theorem 1.12. *Let $A_1 \in \mathbb{R}^{n_1 \times n_1}$ be given. The matrix A_1 is the matrix of a linear system we want to solve. Furthermore assume we have a hierarchy of interpolation operators $P_1, \dots, P_{L-1}$ with $P_i \in \mathbb{R}^{n_i \times n_{i+1}}$ and invertible matrices $W_1, \dots, W_{L-1}$ with $W_i \in \mathbb{R}^{n_{i+1} \times n_{i+1}}$. Let E_1 and $\widetilde{E}_1$ be the error propagators of two multigrid methods.*

1. *The method described by E_1 has the restriction operators $R_i := W_i P_i^T$ and the Galerkin coarse grid approximations $A_{i+1} := R_i A_i P_i$.*
2. *The method described by $\widetilde{E}_1$ has the restriction operators $\widetilde{R}_i := P_i^T$ and the Galerkin coarse grid approximations $\widetilde{A}_{i+1} := \widetilde{R}_i A_i P_i$.*

Then

$$E_1 = \tilde{E}_1 \,.$$

Proof. According to Theorem 1.10 the error propagators E_1 and $\widetilde{E}_1$ can be computed recursively. We prove the assertion by induction over this recursion.

(a) Let $i = L$. Then $E_i = E_L = 0$ and $\widetilde{E}_i = \widetilde{E}_L = 0$. Thus $E_i = \widetilde{E}_i$.

(b) Let $i < L$. We have

$$E_i = S_i^{\nu_2}(I - P_i(I - E_{i+1}^{\eta})A_{i+1}^{-1}R_iA_i)S_i^{\nu_1} \,.$$

By the induction hypothesis we have that $E_{i+1} = \widetilde{E}_{i+1}$. Thus, the previous equation reads

$$E_i = S_i^{\nu_2}(I - P_i(I - \widetilde{E}_{i+1}^{\eta})A_{i+1}^{-1}R_iA_i)S_i^{\nu_1} \,. \tag{1.38}$$

Let us have a closer look at the term $A_{i+1}^{-1}R_i$; we have

$$\begin{aligned} A_{i+1}^{-1}R_i &= (R_iA_iP_i)^{-1}R_i = (W_iP_i^TA_iP_i)^{-1}W_iP^T = (P_i^TA_iP_i)^{-1}W_i^{-1}W_iP^T \\ &= (P_i^TA_iP_i)^{-1}P^T = \widetilde{A}_{i+1}^{-1}\widetilde{R}_i \,. \end{aligned}$$

Plugging this relation into (1.38) we get

$$E_i = S_i^{\nu_2}(I - P_i(I - \widetilde{E}_{i+1}^{\eta})\widetilde{A}_{i+1}^{-1}\widetilde{R}_iA_i)S_i^{\nu_1} = \widetilde{E}_i \,. \qquad \square$$

In other words, when using the Galerkin coarse approximation, it is not necassary to normalize the transpose of the interpolation operator to obtain a suitable restriction.

1.4. Formal Eigenfunction Analysis

In the experiment conducted in Section 1.3.3, we saw that after some iterations of the Jacobi method the error was slowly varying. Without further justifications we concluded that the Jacobi method is a smoother, i.e., that for *any* initial error the error is slowly varying after a few iterations of the method. We shall provide some justification for this conclusion by using formal eigenfunctions. This analysis will then be turned into a rigorous statement about operators on infinite grids in Chapter 2.

Formal eigenfunction analysis is a way to gain insight into the behavior of operators, by probing them with wave functions of different frequencies. Under the right conditions these wave functions are the eigenfunctions of the operator. Hence, as we shall see, we gain insight, by inspecting the relation between the wave function frequency and the eigenvalue. To begin with the introduction of formal eigenfunction analysis, we first need some notation.

1.4.1. Stencil Notation

Many grid operators are local in the sense that the value of the result at a specific point x can be computed using only the values of the argument that are close to x. *Stencil* notation makes use of this locality to represent these operators in a compact way. Conversely, if we write an operator in stencil form, the structure of the locality of the operator is revealed.

We call a family $\{s_x\}_{x\in\overline{\Omega}_h}$ with $s_x : h\mathbb{Z} \to \mathbb{C}$ a *variable stencil*. The corresponding stencil operator A is

$$[Au](x) := \sum_{\substack{y\in h\mathbb{Z}\\ x+y\in\overline{\Omega}_h}} s_x(y)\cdot u(x+y)\,.$$

The *stencil* at x, s_x, can be written in matrix form by

$$\begin{bmatrix}\cdots & s_x(-h) & \underline{s_x(0)} & s_x(h) & s_x(2h) & \cdots\end{bmatrix}_h\,,$$

where we underlined the central element.

As an example consider the error propagator of the weighted Jacobi method applied to the discrete Poisson equation (1.8):

$$\begin{aligned} E &= I - \omega D^{-1}A \\ &= \begin{pmatrix} 1-\omega & \omega/2 & & & \\ \omega/2 & 1-\omega & \omega/2 & & \\ & \ddots & \ddots & \ddots & \\ & & \omega/2 & 1-\omega & \omega/2 \\ & & & \omega/2 & 1-\omega \end{pmatrix}. \end{aligned}$$

Recall that the variables of this matrix correspond to function values of grid functions. If we define $e := (e_h(x_1), e_h(x_2), \ldots, e_h(x_{n-1}))^T$ and $[E_h e_h](x_i) := [Ee]_i$ for $i = 1, \ldots, n-1$, then

$$\begin{aligned} [E_h e_h](x_1) &= (1-\omega)\cdot e_h(x_1) + \tfrac{\omega}{2}\cdot e_h(x_2) \\ [E_h e_h](x_i) &= \tfrac{\omega}{2}\cdot e_h(x_{i-1}) + (1-\omega)\cdot e_h(x_i) + \tfrac{\omega}{2}\cdot e_h(x_{i+1}) \quad \text{for } 1 < i < n-1 \\ [E_h e_h](x_{n-1}) &= \tfrac{\omega}{2}\cdot e_h(x_{n-2}) + (1-\omega)\cdot e_h(x_{n-1})\,. \end{aligned}$$

From these three equations we see that the operator E_h can be written in stencil form by

$$s_{x_1} = \begin{bmatrix}\underline{1-\omega} & \omega/2\end{bmatrix}_h\,, \tag{1.39a}$$

$$s_{x_i} = \begin{bmatrix}\omega/2 & \underline{1-\omega} & \omega/2\end{bmatrix}_h \quad \text{for } 1 < i < n-1, \tag{1.39b}$$

$$s_{x_{n-1}} = \begin{bmatrix}\omega/2 & \underline{1-\omega}\end{bmatrix}_h\,. \tag{1.39c}$$

We observe that almost all equations of the linear system are given by the same stencil (1.39b). Only the boundary points require different stencils.

1.4.2. Formal Eigenfunctions

Formal eigenfunction analysis is an idealized analysis. Instead of analyzing the operator we are interested in, we analyze an operator that is similar but defined on an *infinite grid* and given by a *constant stencil*. The infinite grid with step-size h is G_h and defined by

$$G_h = \{hz : z \in \mathbb{Z}\}\,.$$

An operator A is a constant stencil operator, if it can be written in the form

$$[Au](x) := \sum_{y \in G_h} s(y) \cdot u(x+y), \tag{1.40}$$

where $s : G_h \to \mathbb{C}$ is called the *constant stencil* of the operator.

Constant stencil operators can be analyzed by the help of the *wave functions* $\phi_\theta : G_h \to \mathbb{C}$ with $\theta \in [-\frac{\pi}{h}, \frac{\pi}{h})$ and

$$\phi_\theta(x) = e^{\mathrm{i}\cdot\theta x}\,. \tag{1.41}$$

When a constant stencil operator is applied to a wave function the result is a scaled version of this wave function, as the next proposition shows.

Definition & Lemma 1.13. *Let A be the constant stencil operator given by* (1.40) *and ϕ_θ a wave function* (1.41)*. Then*

$$A\phi_\theta = \hat{a}(\theta) \cdot \phi_\theta\,,$$

where $\hat{a}(\theta)$ is the scalar function given by

$$\hat{a}(\theta) := \sum_y s_y e^{\mathrm{i}\cdot\theta y}\,. \tag{1.42}$$

We call $\hat{a}(\theta)$ the Fourier symbol*, $\phi_\theta(x)$ a* formal eigenfunction*, and $A(\theta)$ the corresponding formal eigenvalue of A.*

Proof. We have

$$(A\phi_\theta)(x) = \sum_y s_y e^{\mathrm{i}\cdot\theta(x+y)} = \sum_y s_y e^{\mathrm{i}\cdot\theta x} \cdot e^{\mathrm{i}\cdot\theta y} = \left(\sum_y s_y e^{\mathrm{i}\cdot\theta y}\right) \cdot \phi_\theta(x)\,. \qquad \square$$

To study the Jacobi method using formal eigenfunctions, we have to find an error propagator which is defined on an infinite grid, given by a constant stencil, and behaves similarly to the error propagator of the Jacobi method on the finite grid. Considering the stencil of the latter operator (1.39b), we see that for large n almost all stencil entries are equal to

$$\begin{bmatrix} \omega/2 & 1-\omega & \omega/2 \end{bmatrix}\,.$$

Hence, we define the constant stencil operator on the infinite grid to have all entries equal to this stencil. Computing the symbol $\hat{e}$ of this operator gives

$$\begin{aligned}
\hat{e}(\theta) &= \tfrac{\omega}{2} e^{\mathrm{i}\cdot\theta(-h)} + (1-\omega) e^{\mathrm{i}\cdot\theta 0} + \tfrac{\omega}{2} e^{\mathrm{i}\cdot\theta h} \\
&= \tfrac{\omega}{2} \overline{e^{\mathrm{i}\cdot\theta h}} + (1-\omega) + \tfrac{\omega}{2} e^{\mathrm{i}\cdot\theta h} \\
&= (1-\omega) + \mathrm{Re}\left(\omega e^{\mathrm{i}\cdot\theta h}\right) \\
&= (1-\omega) + \omega\cos(\theta h)\,.
\end{aligned}$$

This symbol is shown in Figure 1.8 for different values of ω.

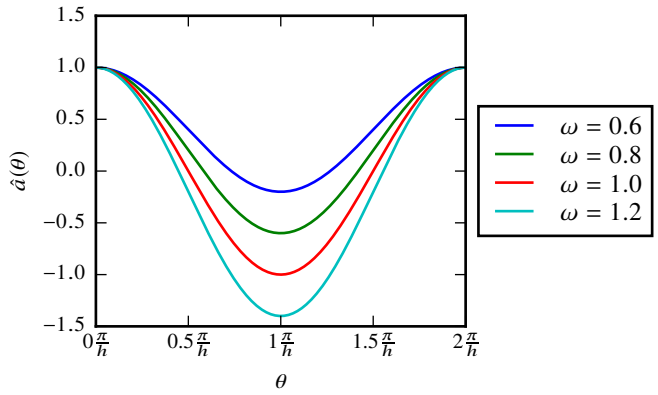


Figure 1.8.: Fourier symbol of the error propagator of the weighted Jacobi method applied to the Poisson equation for different weights ω.

1.4.3. Wave Functions

Before we can interpret the Fourier symbol of an operator, we need to discuss a few properties of the wave functions (1.41). If the domain of the wave function is equal to the whole line of real numbers, then every choice of θ leads to a different wave function ϕ_θ. If, however, the domain is restricted to the grid G_h then there are $\theta, \theta' \in \mathbb{R}$ such that $\phi_\theta = \phi_{\theta'}$, i.e., $\phi_\theta(x) = \phi_{\theta'}(x)$ for all $x \in G_h$. To prove this assertion we will need the following lemma.

Lemma 1.14. *Let* $0 < h \in \mathbb{R}$. *Then*

$$\left(e^{\mathrm{i}\theta x} = 1 \quad \textit{for all} \quad x \in G_h\right) \quad \textit{if and only if} \quad \theta \in \tfrac{2\pi}{h}\mathbb{Z}\,. \tag{1.43}$$

Proof. Recall that for a real number $a \in \mathbb{R}$ we have

$$e^{\mathrm{i}a} = 1 \quad \text{if and only if} \quad a \in 2\pi\mathbb{Z}\,.$$

Assume that $\theta \in \frac{2\pi}{h}\mathbb{Z}$ and $x \in h\mathbb{Z}$. Then $\theta x \in 2\pi\mathbb{Z}$ and therefore $e^{\mathrm{i}\theta x} = 1$.

Conversely, assume that $e^{\mathrm{i}\theta x} = 1$ for all $x \in h\mathbb{Z}$. Thus, for $x = h$ we have $1 = e^{\mathrm{i}\theta x} = e^{\mathrm{i}\theta h}$, implying that $\theta h \in 2\pi\mathbb{Z}$, and therefore $\theta \in \frac{2\pi}{h}\mathbb{Z}$. □

Theorem 1.15. *Let* $0 < h \in \mathbb{R}$ *and* $\theta, \theta' \in \mathbb{R}$. *Then*

$$\left(e^{\mathrm{i}\theta x} = e^{\mathrm{i}\theta' x} \quad \textit{for all} \quad x \in G_h\right) \quad \textit{if and only if} \quad (\theta - \theta') \in \tfrac{2\pi\mathbb{Z}}{h}\,.$$

Proof. Consider

$$\begin{aligned} e^{\mathrm{i}\theta x} = e^{\mathrm{i}\theta' x} &\iff e^{\mathrm{i}(\theta-\theta'+\theta')x} = e^{\mathrm{i}\theta' x} \\ &\iff e^{\mathrm{i}(\theta-\theta')x} \cdot e^{\mathrm{i}\theta' x} = e^{\mathrm{i}\theta' x}\,. \end{aligned}$$

As $e^{\mathrm{i}\theta' x} \neq 0$, we can divide both sides by $e^{\mathrm{i}\theta' x}$ and obtain that

$$e^{\mathrm{i}\theta x} = e^{\mathrm{i}\theta' x} \iff e^{\mathrm{i}(\theta-\theta')x} = 1\,,$$

and the assertion follows from Lemma 1.14. □

Let $\Theta_h := \left[0, \frac{2\pi}{h}\right)$, which we call the set of *visible frequencies* on G_h. Theorem 1.15 implies that every wave function on G_h is uniquely written as ϕ_θ with $\theta \in \Theta_h$.

There are other sets with this property. We choose, however, Θ_h to represent the wave functions on G_h as this choice simplifies the discussions in the later chapters.

Low and High Frequencies

Let us now address the questions: When are wave functions slowly varying; when are they oscillatory?

If we consider the continuous variable x, the absolute modulus of the derivative with respect to x of the wave function answers these questions. The absolute modulus of the derivative is

$$\left|\frac{d\phi_\theta}{dx}(x)\right| = \left|\theta \mathrm{i} \cdot e^{\mathrm{i}\theta x}\right| = \left|\theta\right| \cdot \left|\mathrm{i}\right| \cdot \left|e^{\mathrm{i}\theta x}\right| = \left|\theta\right|.$$

Thus, for continuous x the wave function ϕ_θ becomes more oscillatory the larger $|\theta|$ becomes.

The behavior of the wave function is different when we restrict x to the grid G_h. In this case we can measure the amount of oscillation by the difference of the wave function at neighboring grid points, i.e., $\left|\phi_\theta(x) - \phi_\theta(x+h)\right|$. This difference can be expressed by a simple formula.

Proposition 1.16. *Let $x, \theta, h \in \mathbb{R}$. Then*

$$\left|\phi_\theta(x) - \phi_\theta(x+h)\right| = \sqrt{2 - 2\cos(\theta h)}. \tag{1.44}$$

Proof. We have

$$\begin{aligned}\left|\phi_\theta(x) - \phi_\theta(x+h)\right| &= \left|e^{\mathrm{i}\theta x} - e^{\mathrm{i}\theta(x+h)}\right| = \left|e^{\mathrm{i}\theta x}(1 - e^{\mathrm{i}\theta h})\right| \\ &= \left|e^{\mathrm{i}\theta x}\right| \cdot \left|(1 - e^{\mathrm{i}\theta h})\right| = \left|1 - e^{\mathrm{i}\theta h}\right|.\end{aligned}$$

This equation can be further simplified by using the definition of the absolute value of complex numbers;

$$\begin{aligned}\left|1 - e^{\mathrm{i}\theta h}\right|^2 &= (1 - e^{\mathrm{i}\theta h}) \cdot \overline{(1 - e^{\mathrm{i}\theta h})} = (1 - e^{\mathrm{i}\theta h}) \cdot (1 - e^{-\mathrm{i}\theta h}) \\ &= 1 - e^{-\mathrm{i}\theta h} - e^{\mathrm{i}\theta h} + e^{\mathrm{i}\theta h} e^{-\mathrm{i}\theta h} = 1 - (e^{-\mathrm{i}\theta h} + e^{\mathrm{i}\theta h}) + e^0 \\ &= 2 - 2\cos(\theta h). \end{aligned} \tag{1.45}$$

Taking the square root on both sides proves the assertion. □

The amount of oscillation in dependence on the frequency (1.44) is shown in Figure 1.9. The amount is periodic in θ with period $\frac{2\pi}{h}$, which is in agreement with Theorem 1.15. The amount of oscillation at $\theta = 0$ and $\theta = \frac{2\pi}{h}$ is zero and has its maximum value of two at $\frac{\pi}{h}$. The amount increases on the interval $\left[0, \frac{\pi}{h}\right]$ and decreases on the interval $\left[\frac{\pi}{h}, \frac{2\pi}{h}\right]$. Thus, the amount of oscillation is small if θ is close to a multiple of $\frac{2\pi}{h}$; otherwise it is large.

Definition 1.17. Let $h > 0$ be given. We say

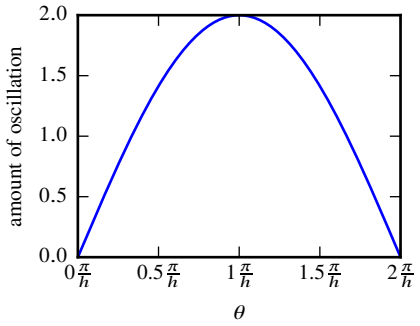


Figure 1.9.: The amount of oscillation of a wave function on a grid with step-size h. Small values yield a slowly varying function.

1. $\theta \in \Theta_h$ is a low frequency if θ is close to a multiple of $\frac{2\pi}{h}$, and
2. $\theta \in \Theta_h$ is a high frequency if θ is away from every multiple of $\frac{2\pi}{h}$.

This definition is of course vague, as we have not stated what "close" and "away" mean. We will be more precise when discussing *harmonic frequencies*. There we will see that the context determines when to consider a value as close enough to $\frac{2\pi}{h}$ to call it "close". All that matters now is that if θ is closer to a multiple of $\frac{2\pi}{h}$ than θ' implies that ϕ_θ is varying more slowly than $\phi_{\theta'}$. Or, in other words, if a frequency θ is *lower* than θ' then ϕ_θ is varying more slowly than $\phi_{\theta'}$.

Corollary 1.18. *Let $h > 0$ be given. A wave function ϕ_θ*
1. *is slowly varying if and only if θ is a low frequency, and*
2. *is oscillatory if and only if θ is a high frequency.*

Interpretation of Fourier Symbols

We want to give an interpretation of Fourier symbols. For this purpose, recall that a wave function ϕ_θ is a formal eigenfunction of any constant stencil operator A. The corresponding eigenvalue is $\hat{a}(\theta)$, i.e.,

$$\hat{a}(\theta) := \sum_y s_y e^{\mathrm{i}\cdot\theta y}\,, \qquad \text{(1.42 revisited)}$$

where $\hat{a}$ is the Fourier symbol of A. From this equation we see that if the operator A is applied to the function ϕ_θ, the function

$$\phi_\theta \text{ is } \begin{cases} \text{damped} & \text{if } |\hat{a}(\theta)| < 1, \\ \text{amplified} & \text{if } |\hat{a}(\theta)| > 1, \\ \text{unchanged} & \text{if } |\hat{a}(\theta)| = 1. \end{cases}$$

Let us go a step further and consider a linear combination of wave functions;

$$\phi := \sum_{k=1}^{m} \alpha_k \phi_{\theta_k} \quad \text{for} \quad \theta_k \in \Theta_h \text{ and } \alpha_k \in \mathbb{C}\,.$$

Since the operator A is linear

$$A\phi = \sum_{k=1}^{m} \alpha_k (A\phi_{\theta_k}) = \sum_{k=1}^{m} \alpha_k \hat{a}(\theta_k) \cdot \phi_{\theta_k} \ .$$

Therefore, if we know the eigenvalues corresponding to the formal eigenfunctions of A, we already know how A acts on a linear combination of eigenfunctions.

This observation allows us to study the smoothing behavior of a method. Assume, e.g., that the operator A damps oscillatory wave functions strongly. If we apply the operator to a linear combination of oscillatory and slowly varying functions, the slowly varying functions will dominate the result, as the other functions have been damped. In this case, the operator A smooths wave functions.

Let us now consider the error propagator of the Jacobi method applied to the discrete Poisson equation. The Fourier symbol of this error propagator is shown in Figure 1.8 on page 27.

For $\omega = 0.8$ the error propagator damps the wave functions significantly if θ is a high frequency, i.e., θ is not too close to 0 or $\frac{2\pi}{h}$. Consequently, the Jacobi method for $\omega = 0.8$ damps oscillatory error functions strongly, while slowly varying error functions are only weakly damped or remain unchanged. This is the ideal behavior of a smoother.

The behavior of the Jacobi method for other values of ω is less optimal. For example for $\omega = 1.2$, highly oscillatory error functions are amplified.

Harmonic Frequencies

We introduce an idealized analysis of the two-grid method in this section. We start by considering the restriction of wave functions to a coarse grid.

Assume that the fine grid has the step-size h and the step-size of the coarse grid is an integer multiple of h, i.e., the coarse grid step-size is $n \cdot h$ for $n \in \mathbb{N}$. We call n the *coarsening range*. Let R be the injection restriction operator from the grid G_h to a grid $G_{n \cdot h}$. The operator is given by

$$[Ru](x) = u(x) \quad \text{for } x \in G_{n \cdot h}.$$

According to Theorem 1.15 there are frequencies $\theta, \theta' \in \Theta_h$ such that the wave functions ϕ_θ and $\phi_{\theta'}$ differ on G_h, but are the same when restricted to $G_{n \cdot h}$, i.e.,

$$\phi_\theta \neq \phi_{\theta'} \quad \text{but} \quad R\phi_\theta = R\phi_{\theta'}.$$

More precisely, the theorem states that on the grid $G_{n \cdot h}$ the parameters θ and θ' describe the same grid functions if

$$(\theta - \theta') \in \tfrac{2\pi}{nh}\mathbb{Z}.$$

In this case we say that $\theta' \in \Theta_h$ is an *n,h-harmonic* of $\theta \in \Theta_h$. It is easy to see that n is the number of n,h-harmonics that every frequency has. Furthermore, the predicate "is an n,h-harmonic of" is an equivalence relation. Thus, the frequencies of the wave functions that map to the same wave function on the coarse grid is the equivalence class

$$[\theta] = \{\theta' \in \Theta_h : \theta' \text{ is an } n,h\text{-harmonic of } \theta\} \ .$$

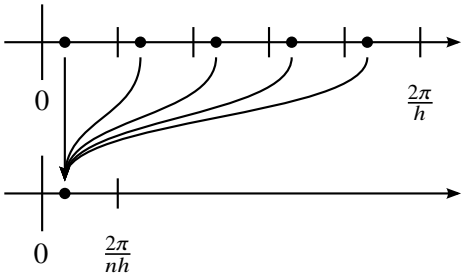


Figure 1.10.: The relation of the frequencies (upper points) and their base frequency (lower point) for $n = 5$.

Recall that every wave function on $G_{n \cdot h}$ is uniquely described by a frequency in $\Theta_{n \cdot h}$. We call this the *base frequency*. Using the base frequency, we can write the set of harmonics as

$$[\theta] = \left\{ \theta_b + j \cdot \tfrac{2\pi}{n \cdot h} : 0 \leq j < n, j \in \mathbb{Z} \right\} .$$

In other words, we can say that the restriction maps all harmonic frequencies to the *same* frequency—the base frequency. Figure 1.10 illustrates the relation of the frequencies and their base frequency.

Now we construct an idealized interpolation for wave functions. This interpolation should be close to the interpolation we use—the linear interpolation—and it should map a grid function on the coarse grid to a grid function on the fine grid. We cannot use the linear interpolation, as it does not return a wave function in general.

As a natural requirement, we want that interpolating and then restricting a wave function should return the same wave function, i.e.,

$$RP\phi_\theta = \phi_\theta .$$

This relation implies that $P\phi_\theta = \phi_{\theta'}$ for some n,h-harmonic θ' of θ, as the wave functions of all n,h-harmonics are restricted to the same wave function. Hence, to uniquely determine P we need to determine which harmonic θ' to choose.

We want that P is close to the linear interpolation. As the linear interpolation returns slowly varying functions, we choose θ' to be the lowest frequency from the n,h-harmonics $[\theta]$. In other words, we pick θ' to be the frequency from $[\theta]$ that is closest[2] to a multiple of $\frac{2\pi}{h}$. This frequency is called the *low n,h-harmonic of* θ and denoted by θ_ℓ. Furthermore we say that θ is *a low n,h-harmonic* if there exists a $\tilde{\theta}$ such that $\theta = \tilde{\theta}_\ell$. Using this definition we can define the idealized interpolation operator P as

$$P\phi_\theta = \phi_{\theta_\ell} .$$

Note that this makes Definition 1.17 precise under the assumption that we have a coarsening range n that we consider.

[2] In case of ambiguity we choose the larger one.

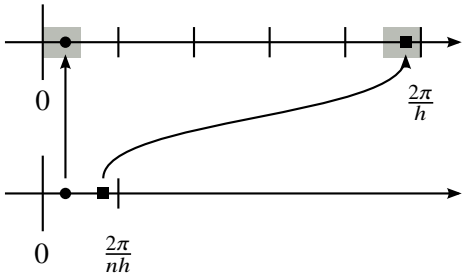


Figure 1.11.: Computation of the low harmonic of a frequency.

The set of low n,h-harmonics can be characterized as follows. The n,h-harmonics of a frequency θ are evenly spaces with a distance $\frac{2\pi}{nh}$. Hence, a frequency θ in this set is closest to a multiple of $\frac{2\pi}{h}$ if and only if

$$0 \le \theta < \tfrac{\pi}{nh} \quad \text{or} \quad \tfrac{(2n-1)\pi}{nh} \le \theta < \tfrac{2\pi}{h}\,.$$

Thus, a frequency θ is a low n,h-harmonic if and only if the above inequalities hold. Using this relation, we can give a formula for the frequency of the wave function $\phi_{\theta_\ell} = P\phi_\theta$. In this case $\theta \in \Theta_{n\cdot h}$, and the corresponding low harmonic is

$$\theta_\ell = \begin{cases} \theta & \text{if } \theta < \frac{\pi}{nh}, \\ \frac{2\pi(n-1)}{nh} + \theta & \text{if } \theta \ge \frac{\pi}{nh}. \end{cases} \tag{1.46}$$

This relation, which is shown in Figure 1.11, completes the description of the idealized interpolation operator P.

We idealize the two-grid method by assuming that the coarse-grid error is exactly computed on G_{nh}. In other words, we assume that the error propagator of the coarse-grid correction is

$$Q_n := I - PR,$$

where R and P are the idealized restriction and interpolation operators introduced in this section. Therefore, the error propagator of the idealized two-grid method is

$$E = S^{\nu_2} Q_n S^{\nu_1}\,, \tag{1.47}$$

where S is the error propagator of the smoothing method. The error propagator of the idealized coarse grid correction Q_n is given by

$$Q_n\phi_\theta = \begin{cases} 0 & \text{if } \theta \text{ is a low } n,h\text{-harmonic}, \\ \phi_\theta & \text{otherwise.} \end{cases}$$

Thus, ϕ_θ is a formal eigenfunction of the idealized coarse grid correction, and its Fourier symbol is given by

$$\hat{q}_n(\theta) = \begin{cases} 0 & \text{if } \theta \text{ is a low } n,h\text{-harmonic}, \\ 1 & \text{otherwise.} \end{cases}$$

Using this symbol, we can write down the Fourier symbol of the idealized two-grid method (1.47);

$$\hat{e}(\theta) = \hat{s}(\theta)^{\nu_2} \cdot \hat{q}_n(\theta) \cdot \hat{s}(\theta)^{\nu_1} .$$

The number $|\hat{e}(\theta)|$ is the ratio of the size of the wave function ϕ_θ before one iteration of the two-grid method and after. We define the *smoothing factor* (w.r.t. the coarsening range n) as

$$\mathrm{smf}(S, n) := \max_{\theta \in \Theta_h} |\hat{e}(\theta)| = \max_{\theta \in \Theta_h} |\hat{s}(\theta)^{\nu_2} \cdot \hat{q}_n(\theta) \cdot \hat{s}(\theta)^{\nu_1}| .$$

As the coarse grid correction is idealized, the smoothing factor only gives information about the quality of the smoother when using the coarsening range n—hence the name smoothing factor.

1.5. Higher Dimensions

To this point we considered only multigrid methods for one-dimensional problems. In the one-dimensional case, notation is easier and concepts of multigrid methods can be well explained. Many real world problem, however, have higher dimensionality. Furthermore, in 1D there are often better methods; multigrid methods are often the best choice in 2D and 3D.

1.5.1. Multi-Index Notation

To deal with higher dimensional problems, we need additional notation. We shall use this notation frequently in the following sections and chapters.

We denote the dimensionality by the letter d. We represent the vectors in $\mathbb{R}^d$ and the elements from $\mathbb{Z}^d$ and $\mathbb{N}^d$ by bold letters. Most of the time they will be used as (multi-)indices. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$. Whenever we multiply or divide two vectors the operation is meant to be carried out *component wise*, i.e.,

$$\mathbf{a} \cdot \mathbf{b} := (a_\ell b_\ell)_{\ell=1}^d \quad \text{and} \quad \mathbf{a}/\mathbf{b} := (a_\ell / b_\ell)_{\ell=1}^d .$$

In unambiguous situations we shall also write $\mathbf{ab}$ for $\mathbf{a} \cdot \mathbf{b}$ and $\frac{\mathbf{a}}{\mathbf{b}}$ for $\mathbf{a}/\mathbf{b}$. The *scalar product* of two vectors is denoted by angle brackets;

$$\langle \mathbf{a}, \mathbf{b} \rangle := \sum_{i=1}^{d} \overline{b_i} \cdot a_i .$$

Whenever we compare vectors it is to be understood component wise as well, i.e.,

$$\mathbf{a} \le \mathbf{b} \iff (a_\ell \le b_\ell \quad \text{for all} \quad \ell = 1, \dots, d) .$$

Furthermore, whenever we write "for $\mathbf{i} = \mathbf{a}, \dots, \mathbf{b}$" we mean "for $i_1 = a_1, \dots, b_1$, $i_2 = a_2, \dots, b_2, \dots, i_d = a_d, \dots, b_d$". We define

$$\sum_{\mathbf{j}=\mathbf{a}}^{\mathbf{b}} u_{\mathbf{j}} := \sum_{j_1=a_1}^{b_1} \sum_{j_2=a_2}^{b_2} \cdots \sum_{j_d=a_d}^{b_d} u_{(j_1, j_2, \dots, j_d)^T}$$

and

$$\left(u_j\right)_{\mathbf{j}=\mathbf{a}}^{\mathbf{b}} := \left(\cdots\left(\left(u_{(j_1,j_2,\ldots,j_d)^T}\right)_{j_1=a_1}^{b_1}\right)_{j_2=a_2}^{b_2}\cdots\right)_{j_d=a_d}^{b_d}.$$

Furthermore, let $\mathbf{0} := (0)_{\ell=1}^{d}$ and $\mathbf{1} := (1)_{\ell=1}^{d}$ the vector whose entries are all zero and all one, and let $\mathrm{vol}_{\mathbf{a}} := \prod_{j=1}^{d} a_j$ be the *volume of the hypercube* with side lengths $a_1, \ldots, a_d$.

When we write $|\mathbf{a}|$ we mean the vector that contains the element-wise absolute value of the elements of $\mathbf{a}$, i.e.,

$$|\mathbf{a}| = \left(|a_\ell|\right)_{\ell=1}^{d}.$$

The euclidean norm of a vector $\mathbf{a}$ is denoted as $\|\mathbf{a}\|$;

$$\|\mathbf{a}\| := \left(\sum_{i=1}^{d} |a_i|^2\right)^{1/2}.$$

1.5.2. The Poisson Equation in 2D

The multigrid method can be applied to higher dimensional problems, e.g., 2D and 3D problems. It requires, however, some adjustments.

Let us consider a 2D example: the shape of a square membrane deformed by a load. Assume that the membrane, when seen from above, forms a square located at $\Omega := (0,1) \times (0,1)$. The height of the membrane is described by a function $u : \overline{\Omega} \to \mathbb{R}$. At the boundary, the membrane is fixed at height $g(x)$. The load on the membrane is given by the function $f : \Omega \to \mathbb{R}$. The vertical position u of the membrane is then described by the 2D Poisson equation [6]

$$-\Delta u(\mathbf{x}) = f(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \Omega \tag{1.48a}$$

$$u(\mathbf{x}) = g(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \partial\Omega, \tag{1.48b}$$

where $\Delta u := \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}$.

Discretization

We want to compute a numerical approximation to the solution of the Poisson equation in 2D (1.48). For this purpose let $\mathbf{n} \in \mathbb{N}^2$, $\mathbf{h} := 1/\mathbf{n}$ and

$$G_{\mathbf{h}} = \{\mathbf{z} \cdot \mathbf{h} : \mathbf{z} \in \mathbb{Z}^2\}$$

We define the grid, grid boundary and closure of the grid by

$$\Omega_{\mathbf{h}} := \Omega \cap G_{\mathbf{h}}, \quad \partial\Omega_{\mathbf{h}} := (\partial\Omega) \cap G_{\mathbf{h}}, \quad \text{and} \quad \overline{\Omega}_{\mathbf{h}} := \Omega_{\mathbf{h}} \cup (\partial\Omega_{\mathbf{h}}),$$

respectively. We seek for an approximation of the solution u at the points $\overline{\Omega}_h$.

Like in the 1D case, which is described in Section 1.2, we need an approximation $\Delta_{\mathbf{h}}$ to the operator Δ that needs only the values of u at the discrete points $\overline{\Omega}_h$ for its computation. Using the idea of polynomial interpolation in a similar way as in Section 1.2, it can be shown that

$$\frac{\partial^2 u}{\partial x_1}(x_1, x_2) \approx \frac{u(x_1 - h_1, x_2) - 2u(x_1, x_2) + u(x_1 + h_1, x_2)}{h_1^2}$$

and

$$\frac{\partial^2 u}{\partial x_2}(x_1, x_2) \approx \frac{u(x_1, x_2 - h_2) - 2u(x_1, x_2) + u(x_1, x_2 + h_2)}{h_2^2} .$$

If we plug these approximations into the definition of the operator Δ, we obtain a reasonable approximation:

$$\Delta_{\mathbf{h}} u := \frac{u(x_1 - h, x_2) - 2u(x_1, x_2) + u(x_1 + h, x_2)}{h_1^2} + \frac{u(x_1, x_2 - h) - 2u(x_1, x_2) + u(x_1, x_2 + h)}{h_2^2} .$$

Using this approximation results in the linear system of equations

$$\Delta_{\mathbf{h}} u_{\mathbf{h}}(\mathbf{x}) = f(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \Omega_{\mathbf{h}} \tag{1.49a}$$

$$u_{\mathbf{h}}(\mathbf{x}) = g(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \partial\Omega_{\mathbf{h}} . \tag{1.49b}$$

Multigrid

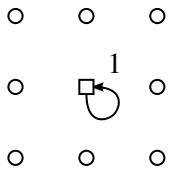
To construct a multigrid method for the 2D Poisson equation, we need to specify a smoother, a restriction, an interpolation, and a coarse grid operator. We can choose the Jacobi method as a smoother, as it can directly be applied to the linear system and it is known that it is a suitable smoother for this problem. For the coarse grid operator we can just use the operator $\Delta_{2\mathbf{h}}$. The restriction and the interpolation operator, however, have to be adapted to the new dimension.

The formula for the injection restriction in 2D is very similar to the 1D case (1.30). The 2D formula is:

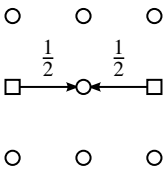
$$[Ru](\mathbf{x}) = u(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in G_{2\mathbf{h}} .$$

The description of the bilinear interpolation—the interpolation we will chose for the 2D case—is a little more complicated. Let us denote the neighbors of $\mathbf{x}$ in the grid $G_{\mathbf{h}}$ in the following way:

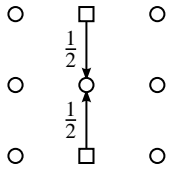
$$\begin{array}{ccc} \mathbf{x}_{\mathrm{nw}} & \mathbf{x}_{\mathrm{n}} & \mathbf{x}_{\mathrm{ne}} \\ \mathbf{x}_{\mathrm{w}} & \mathbf{x} & \mathbf{x}_{\mathrm{e}} \\ \mathbf{x}_{\mathrm{sw}} & \mathbf{x}_{\mathrm{s}} & \mathbf{x}_{\mathrm{se}} \end{array}$$



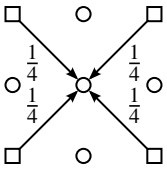
(a) Interpolating the value at a point that is also contained in $G_{2\mathbf{h}}$.



(b) Interpolating a value at a point, whose left and right neighbor are in $G_{2\mathbf{h}}$.



(c) Interpolating the value at a point, whose top and right neighbor are in $G_{2\mathbf{h}}$.



(d) Interpolating the value at a point, whose top-left, top-right, bottom-left, and bottom-right neighbor are in $G_{2\mathbf{h}}$.

Figure 1.12.: Rules for the bilinear interpolation.

Using this notation, the bilinear interpolation in 2D is defined as

$$[Pu](\mathbf{x}) := \begin{cases} u(\mathbf{x}) & \text{if } \mathbf{x} \in \overline{\Omega}_{2\mathbf{h}}, \\ \frac{u(\mathbf{x}_\mathrm{w})+u(\mathbf{x}_\mathrm{e})}{2} & \text{if } \mathbf{x}_\mathrm{w}, \mathbf{x}_\mathrm{e} \in \overline{\Omega}_{2\mathbf{h}}, \\ \frac{u(\mathbf{x}_\mathrm{s})+u(\mathbf{x}_\mathrm{n})}{2} & \text{if } \mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{n} \in \overline{\Omega}_{2\mathbf{h}}, \\ \frac{u(\mathbf{x}_\mathrm{sw})+u(\mathbf{x}_\mathrm{se})+u(\mathbf{x}_\mathrm{nw})+u(\mathbf{s}_\mathrm{ne})}{4} & \text{if } \mathbf{x}_\mathrm{sw}, \mathbf{x}_\mathrm{se}, \mathbf{x}_\mathrm{nw}, \mathbf{s}_\mathrm{ne} \in \overline{\Omega}_{2\mathbf{h}}, \end{cases}$$

Figure 1.12 gives a visual description of this formula.

The equation can be written more compactly using the definition of the neighborhood of a point $\mathbf{x}$ in the grid $G_{\mathbf{h}}$. The neighborhood of $\mathbf{x}$ is defined by

$$\mathcal{N}(\mathbf{x}) := \{\mathbf{y} \in \overline{\Omega}_H : |\mathbf{y} - \mathbf{x}| \leq \mathbf{h}\} .$$

The interpolation P can be written as

$$[Pu](\mathbf{x}) = \frac{1}{\#\mathcal{N}(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} u(\mathbf{y}) \quad \text{for } \mathbf{x} \in \Omega_h. \tag{1.50}$$

1.5.3. Formal Eigenfunctions

In the 1D case we used formal eigenfunctions to show that every constant stencil operator has a Fourier symbol[3], and the Fourier symbol provided insight into the behavior of the corresponding operator. We shall see that the same is true for higher dimensions.

[3] See Section 1.4.2.

In dD the infinite grid $G_{\mathbf{h}}$ with step-size $\mathbf{h}$ is

$$G_{\mathbf{h}} := \{\mathbf{h} \cdot \mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\} = \{(h_1 z_1, \dots, h_d z_d)^T : \mathbf{z} \in \mathbb{Z}^d\}\,. \tag{1.51}$$

On this grid a constant stencil operator A is given by

$$[Au](\mathbf{x}) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot u(\mathbf{x} + \mathbf{y})\,. \tag{1.52}$$

Furthermore, let $\theta \in \mathbb{R}^d$ and $\mathbf{x} \in G_{\mathbf{h}}$; the wave functions in dD are then

$$\phi_\theta(\mathbf{x}) = e^{\mathrm{i}\langle \theta, \mathbf{x}\rangle}, \tag{1.53}$$

where $\langle \mathbf{a}, \mathbf{b}\rangle := \sum_{i=1}^d \overline{b}_i a_i$ is the Euclidean scalar product. Note that θ and $\mathbf{x}$ are vectors. Thus, in comparison to the 1D case (1.41), we replace the multiplication of the two scalars by the scalar product of two vectors. The function ϕ_θ is a *plane wave*. The wave propagates into the direction of the vector θ and the length of θ is the frequency.

In 1D, the wave functions are the formal eigenfunctions of the constant stencil operator (cf. Lemma 1.13). The same is true in dD:

Lemma 1.19. *Let A be the constant stencil operator given by* (1.52) *and ϕ_θ a wave function* (1.53)*. Then*

$$A\phi_\theta = \hat{a}(\theta) \cdot \phi_\theta, \tag{1.54}$$

where $\hat{a}(\theta)$ is the scalar function given by

$$\hat{a}(\theta) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{y}\rangle}\,. \tag{1.55}$$

Proof. We have

$$\begin{aligned}
[A\phi_\theta](\mathbf{x}) &= \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{x}+\mathbf{y}\rangle} = \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{x}\rangle + \mathrm{i}\langle \theta, \mathbf{y}\rangle} \\
&= \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{x}\rangle} \cdot e^{\mathrm{i}\langle \theta, \mathbf{y}\rangle} = \left(\sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{y}\rangle} \right) \cdot e^{\mathrm{i}\langle \theta, \mathbf{x}\rangle} = \hat{a}(\theta) \cdot \phi_\theta(\mathbf{x})\,. \qquad \square
\end{aligned}$$

1.5.4. Wave Functions

We discussed in Section 1.4.3 that in 1D the restriction of different wave functions to a coarse grid can give the same wave function. This statement can be generalized to the dD case. In the following we generalize Lemma 1.14 and Theorem 1.15.

Lemma 1.20. *Let $0 < \mathbf{h} \in \mathbb{R}^d$.*

$$\left(e^{\mathrm{i}\langle \theta, \mathbf{x}\rangle} = 1 \quad \textit{for all} \quad \mathbf{x} \in G_{\mathbf{h}}\right) \quad \textit{if and only if} \quad \theta \in 2\pi\mathbb{Z}^d/\mathbf{h}\,.$$

Proof. Lemma 1.14 shows the assertion for $d = 1$, i.e.,

$$\left(e^{\mathrm{i}\theta x} = 1 \quad \text{for all} \quad x \in G_h\right) \quad \text{if and only if} \quad \theta \in \tfrac{2\pi}{h}\mathbb{Z}\,. \tag{1.43 revisited}$$

Assume that $\theta \in \mathbb{Z}^d/\mathbf{h}$ and $\mathbf{x} \in G_{\mathbf{h}}$. Then

$$e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}\theta_1 x_1} \cdot e^{\mathrm{i}\theta_2 x_2} \cdots e^{\mathrm{i}\theta_d x_d} = 1 \cdot 1 \cdots 1 = 1\,.$$

Conversely, assume that $e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = 1$ for all $\mathbf{x} \in G_{\mathbf{h}}$. Let $\tilde{x} \in h_j\mathbb{Z}$, $j = 1, \ldots, d$, and $x_k = \tilde{x}\delta_{kj}$, where δ_{kj} is the *Kronecker delta*, i.e.,

$$\delta_{kj} = \begin{cases} 1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$1 = e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}\theta_j\tilde{x}}\,.$$

Thus, the one-dimensional (1.43) case implies $\theta_j \in 2\pi\mathbb{Z}/h_j$. □

Theorem 1.21. *Let* $0 < \mathbf{h} \in \mathbb{R}$ *and* $\theta, \theta' \in \mathbb{R}^d$. *Then*

$$\left(e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} \quad \textit{for all} \quad x \in G_{\mathbf{h}}\right) \quad \textit{if and only if} \quad (\theta - \theta') \in 2\pi\mathbb{Z}^d/\mathbf{h}\,.$$

Proof. Let $\mathbf{x} \in \mathbb{R}^d$. Consider

$$\begin{aligned} e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} &\iff e^{\mathrm{i}\langle\theta-\theta'+\theta',\mathbf{x}\rangle} = e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} \\ &\iff e^{\mathrm{i}\langle\theta-\theta',\mathbf{x}\rangle} \cdot e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} = e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle}\,. \end{aligned}$$

Since $e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} \neq 0$, we can divide by $e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle}$ to obtain that

$$e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}\langle\theta',\mathbf{x}\rangle} \iff e^{\mathrm{i}\langle\theta-\theta',\mathbf{x}\rangle} = 1$$

Thus, the assertion follows from Lemma 1.20. □

We define the set of *visible frequencies* on $G_{\mathbf{h}}$ as

$$\Theta_{\mathbf{h}} := \left[0, \tfrac{2\pi}{h_1}\right) \times \cdots \times \left[0, \tfrac{2\pi}{h_d}\right)\,. \tag{1.56}$$

A consequence of Theorem 1.21 is that for every wave function ϕ_θ, there exists a unique $\theta' \in \Theta_{\mathbf{h}}$ such that

$$\phi_\theta(\mathbf{x}) = \phi_{\theta'}(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{h}}\,.$$

1.5.5. Low and High Frequencies

Depending on the frequency θ, a wave function can be oscillatory or slowly varying. This statement is also true in dD.

The dD wave function with frequency θ can be written as

$$\phi_\theta(\mathbf{x}) = e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} = e^{\mathrm{i}(\theta_1 x_1 + \cdots + \theta_d x_d)} = e^{\mathrm{i}\theta_1 x_1} \cdots e^{\mathrm{i}\theta_d x_d} = \phi_{\theta_1}(x_1) \cdots \phi_{\theta_d}(x_d)\,. \tag{1.57}$$

Thus, the dD wave function is the product of d 1D wave functions. If one of the wave functions is oscillatory, the whole function will be oscillatory, which motivates the following definition.

Definition 1.22. Let $\mathbf{h} > 0$ be given. The frequency θ

1. is a low frequency if all θ_k $(k = 1, \ldots, d)$ are low frequencies.
2. is a high frequency if one θ_k $(k = 1, \ldots, d)$ is a high frequency.

The amount of oscillations of a wave function on the grid $G_\mathbf{h}$ is the difference of the function value at neighboring grid points. In contrast to the 1D case, which is characterized by Proposition 1.16, there is more than one direction in which a wave function can vary. Therefore, we can measure the amount of oscillation in different directions.

Let $\mathbf{x} \in G_\mathbf{h}$ and $\mathbf{x} + \Delta\mathbf{x} \in G_\mathbf{h}$ be two neighboring points. Then the difference

$$\left|\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + \Delta\mathbf{x})\right|$$

is the amount of oscillation of ϕ_θ at $\mathbf{x}$ in direction $\Delta\mathbf{x}$. It is computed in:

Proposition 1.23. *Let $\theta, \Delta\mathbf{x} \in \mathbb{R}^d$. Then for all $\mathbf{x} \in \mathbb{R}^d$*

$$\left|\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + \Delta\mathbf{x})\right| = \sqrt{2 - 2\cos\langle\theta, \Delta\mathbf{x}\rangle}\,.$$

Proof. The wave function

$$\begin{aligned}\left|e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} - e^{\mathrm{i}\langle\theta,\mathbf{x}+\Delta\mathbf{x}\rangle}\right| &= \left|e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} - e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle+\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right| = \left|e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} - e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} \cdot e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right| \\ &= \left|e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle}\right| \cdot \left|1 - e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right| = \left|1 - e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right|.\end{aligned}$$

Combining this equation with the simplification (1.45) yields

$$\left|1 - e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right| = \sqrt{2 - 2\cos\langle\theta, \Delta\mathbf{x}\rangle}\,. \qquad \square$$

Using this result we can determine the amount of oscillation in the direction of a coordinate axis. Let $\mathbf{e}_k$ be the kth unit vector; if $\Delta\mathbf{x} = \mathbf{h} \cdot \mathbf{e}_k$ then $\langle\theta, \Delta\mathbf{x}\rangle = \theta_k h_k$.

Corollary 1.24. *Let $\theta, \mathbf{h} \in \mathbb{R}^d$. Then for all $\mathbf{x} \in \mathbb{R}^d$*

$$\left|\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + h_k\mathbf{e}_k)\right| = \sqrt{2 - 2\cos(\theta_k h_k)}\,.$$

Comparing the last equation to the amount of oscillation of a 1D function, given in (1.44), we see that the amount of oscillation in the x_k-direction is the amount of oscillation of the factor $e^{\mathrm{i}\theta_k x_k}$ in the product (1.57). In other words, the wave function ϕ_θ is oscillatory in the x_k-direction, if the wave function $e^{\mathrm{i}\theta_k x_k}$ is oscillatory.

We can also show the converse statement. If the wave functions $e^{\mathrm{i}\theta_k x_k}$ for $k = 1, \ldots, d$ have a small amount of oscillation, i.e., they are slowly varying, then the product of the wave functions also has a small amount of oscillation. This fact is proven in the next theorem, by bounding the amount of oscillation in any direction in terms of the amount of oscillation of the function in the direction of the coordinate axes, which is the sum of the amount of oscillation of the factors $e^{\mathrm{i}\theta_k x_k}$.

Theorem 1.25. *Let $\theta, \mathbf{h} \in \mathbb{R}^d$ and $\mathbf{z} \in \{-1, 0, 1\}^d$. Then*

$$\left|\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + \mathbf{h} \cdot \mathbf{z})\right| \leq \sum_{k=1}^{d} \sqrt{2 - 2\cos(\theta_k h_k)}.$$

Proof. We define the sequence

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{0}, \\ \mathbf{p}_k &= \mathbf{p}_{k-1} + h_k z_k \mathbf{e}_k \quad \text{for } k > 0. \end{aligned}$$

As $\mathbf{h} \cdot \mathbf{z} = h_1 z_1 \mathbf{e}_1 + \cdots + h_d z_d \mathbf{e}_d$, we have that $\mathbf{p}_d = \mathbf{h} \cdot \mathbf{z}$. Combining this with the fact that $\mathbf{p}_0 = 0$ yields

$$\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + \mathbf{h} \cdot \mathbf{z}) = \sum_{k=1}^{d} \phi_\theta(\mathbf{x} + \mathbf{p}_{k-1}) - \phi_\theta(\mathbf{x} + \mathbf{p}_k),$$

It follows that

$$\left|\phi_\theta(\mathbf{x}) - \phi_\theta(\mathbf{x} + \mathbf{h} \cdot \mathbf{z})\right| \leq \sum_{k=1}^{d} \left|\phi_\theta(\mathbf{x} + \mathbf{p}_{k-1}) - \phi_\theta(\mathbf{x} + \mathbf{p}_k)\right|. \tag{1.58}$$

Now note that

$$\left|\phi_\theta(\mathbf{x} + \mathbf{p}_{k-1}) - \phi_\theta(\mathbf{x} + \mathbf{p}_k)\right| = \left|\phi_\theta(\mathbf{x} + \mathbf{p}_{k-1}) - \phi_\theta((\mathbf{x} + \mathbf{p}_{k-1}) + h_k z_k \mathbf{e}_k)\right|.$$

Thus, by using Corollary 1.24 we see that

$$\left|\phi_\theta(\mathbf{x} + \mathbf{p}_{k-1}) - \phi_\theta(\mathbf{x} + \mathbf{p}_k)\right| = \begin{cases} 0 & \text{if } z_k = 0, \\ \sqrt{2 - 2\cos(\theta_k h_k)} & \text{if } z_k = \pm 1. \end{cases} \tag{1.59}$$

Combining the sum (1.58) with the upper bound (1.59) completes the proof. □

1.5.6. Harmonic Frequencies

In higher dimensions the coarsening range $\mathbf{n} \in \mathbb{N}^d$ is a vector instead of a scalar. If $G_\mathbf{h}$ is the fine grid and $\mathbf{n}$ is the coarsening range then $G_{\mathbf{n}\cdot\mathbf{h}}$ is the corresponding coarse grid. The meaning of the kth entry of $\mathbf{n}$ is as follows. We obtain the coarse grid from the fine grid by selecting every n_k-th point in the x_k-direction. With this definition of the coarsening range, we can introduce the harmonic frequencies in d dimensions.

A frequency $\theta \in \Theta_\mathbf{h}$ is a $\mathbf{n}$,$\mathbf{h}$-harmonic of $\theta' \in \Theta_\mathbf{h}$ if the wave functions ϕ_θ and $\phi_{\theta'}$ coincide on $G_{\mathbf{n}\cdot\mathbf{h}}$, i.e.,

$$\phi_\theta(\mathbf{x}) = \phi_{\theta'}(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}.$$

Theorem 1.21 then states that θ is a $\mathbf{n}$,$\mathbf{h}$-harmonic of $\theta' \in \Theta_\mathbf{h}$ if

$$(\theta - \theta') \in 2\pi\mathbb{Z}^d/(\mathbf{n}\cdot\mathbf{h})\,.$$

Besides, in dD we have that "is $\mathbf{n}$,$\mathbf{h}$-harmonic of" is an equivalence relation; the equivalence class

$$[\theta] := \{\theta' : \theta \text{ is a } \mathbf{n},\mathbf{h}\text{-harmonic of } \theta'\}$$

is the set of all *harmonics of* θ. To get a unique representation of the harmonics we define the *base frequency* θ_b to be the unique frequency in $G_{\mathbf{n}\cdot\mathbf{h}}$ that is in $[\theta]$. With this definition, we can write the set of $\mathbf{n}$,$\mathbf{h}$-harmonics as

$$[\theta] = \{\theta_b + \mathbf{j}\cdot(2\pi)/(\mathbf{n}\cdot\mathbf{h}) : 0 \le \mathbf{j} < \mathbf{n}, \mathbf{j} \in \mathbb{Z}^d\}. \tag{1.60}$$

In Section 1.5.5 we realized that a dD wave function is slowly varying if and only if it is slowly varying in the direction of *all* coordinate axes. Hence, the low $\mathbf{n}$,$\mathbf{h}$-harmonic of θ is defined coordinate-wise; θ_ℓ is the low $\mathbf{n}$,$\mathbf{h}$-harmonic of θ if $\theta_{\ell,k}$ is the n_k,h_k-harmonic of θ_k for all k. In other words,

$$\theta_{\ell,k} := \theta_{k,\ell}\,.$$

We define a frequency θ to be a low $\mathbf{n}$,$\mathbf{h}$-harmonic if there exists a frequency θ' such that θ is the $\mathbf{n}$,$\mathbf{h}$-harmonic of θ'. From (1.46) we obtain that a frequency $\theta \in \Theta_\mathbf{h}$ is a $\mathbf{n}$, $\mathbf{h}$-low harmonic if and only if

$$0 \le \theta_k < \tfrac{\pi}{n_k h_k} \quad \text{or} \quad \tfrac{(2n_k-1)\pi}{n_k h_k} \le \theta_k < \tfrac{2\pi}{h_k} \quad \text{for all} \quad k = 1,\dots,d\,. \tag{1.61}$$

To define the smoothing factor we define the Fourier symbol of the ideal coarse grid correction (cf. Section 1.4.3) as

$$\hat{q}_\mathbf{n}(\theta) := \begin{cases} 0 & \text{if } \theta \text{ is a low } \mathbf{n},\mathbf{h}\text{-harmonic,} \\ 1 & \text{otherwise.} \end{cases} \tag{1.62}$$

Then the *smoothing factor* is

$$\mathrm{smf}(S,\mathbf{s}) := \max_{\theta\in\Theta_\mathbf{h}} |\hat{s}(\theta)^{\nu_2}\cdot\hat{q}_\mathbf{n}(\theta)\cdot\hat{s}(\theta)^{\nu_1}|, \tag{1.63}$$

which generalizes the smoothing analysis to d dimensions.

1.5.7. A 2D Fourier Symbol

As an example, let us consider the Fourier symbol of the Jacobi method when applied to the discrete 2D Poisson equation (1.49). To analyze the error propagator of this method, we need to specify its stencil. Accordingly, it is useful to denote a constant stencil s in 2D by

$$s = \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & s(-h,+h) & s(0,+h) & s(+h,+h) & \cdots \\ \cdots & s(-h,\ 0) & \underline{s(0,\ 0)} & s(+h,\ 0) & \cdots \\ \cdots & s(-h,-h) & s(0,-h) & s(+h,-h) & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{\mathbf{h}},$$

where we have underlined to central element $s(0,0)$.

Let us assume that $h_1 = h_2$, i.e., the grid spacing in each direction is the same. For the interior points the finite difference discretization of the 2D Poisson equation (1.49) is

$$\tfrac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}_{\mathbf{h}}.$$

Let us apply the weighted Jacobi method to the discrete Poisson equation. Then the stencil of the error propagator for the interior points is

$$\begin{bmatrix} & \omega/4 & \\ \omega/4 & (1-\omega) & \omega/4 \\ & \omega/4 & \end{bmatrix}_{\mathbf{h}}.$$

The Fourier symbol of the error propagator of the Jacobi method applied to the 2D Poisson equation is

$$\begin{aligned} \hat{e}(\theta) &= \tfrac{\omega}{4}e^{\mathrm{i}\theta_1(-h_1)} + \tfrac{\omega}{4}e^{\mathrm{i}\theta_2(-h_2)} + (1-\omega)e^{\mathrm{i}\cdot 0} + \tfrac{\omega}{4}e^{\mathrm{i}\theta_2 h_2} + \tfrac{\omega}{4}e^{\mathrm{i}\theta_1 h_1} \\ &= (1-\omega) + 2\,\mathrm{Re}\left(\tfrac{\omega}{4}e^{\mathrm{i}\theta_1 h_1}\right) + 2\,\mathrm{Re}\left(\tfrac{\omega}{4}e^{\mathrm{i}\theta_2 h_2}\right) \\ &= (1-\omega) + \tfrac{\omega}{2}\big(\cos(\theta_1 h_1) + \cos(\theta_2 h_2)\big). \end{aligned}$$

This symbol is shown in Figure 1.13 for $\omega = 0.8$. We see that the values of $\hat{e}(\theta)$ are only large if θ_k is close to 0 or $\frac{2\pi}{h}$ for any $k = 1, 2$. Thus, the weighted Jacobi method reduces the oscillatory wave functions and is therefore a smoother.

1.6. Operator Norm and Spectral Radius

Despite the formal eigenfunction analysis, there are further ways to gain insight into an operator and, consequently, insight into stationary iterative methods. For example, let V be a Banach space—a normed, closed vector space. Furthermore, let $E : V \to V$ be the error propagator of an iterative method, and let $e \in V$ be the error. We are interested in the

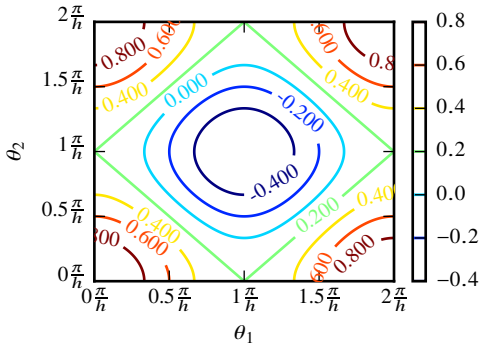


Figure 1.13.: Fourier symbol of the error propagator of the weighted Jacobi method applied to the Poisson equation in 2D with $\omega = 0.8$.

difference of the size of the error e before the application of E and afterwards. Consequently, we want to know the ratio

$$c_1 := \frac{\|Ee\|}{\|e\|}$$

of the new and the old error. If the ratio $c_1 < 1$ then E reduces the error e; if the ratio $c_1 > 1$ then E amplifies the error. The *operator norm* gives an upper bound for the ratio c_1. It is defined by

$$\|E\| := \sup\left\{\frac{\|Ef\|}{\|f\|} : f \in V, f \neq 0\right\}.$$

This definition implies

$$\|Ef\| \leq \|E\| \cdot \|f\|, \tag{1.64}$$

which gives an upper bound for c_1.

In an iterative method the error propagator is applied multiple times. We are, thus, also interested in the ratio

$$c_n := \frac{\|E^n e\|}{\|e\|}.$$

If $c_n \to 0$ for $n \to \infty$ then the error converges to zero during the iteration, and therefore the iterative method finds the solution.

Equation (1.64) implies

$$\|E^n\| \leq \|E\| \cdot \|E^{n-1}\| \leq \cdots \leq \|E\|^n.$$

That means $\|E\|^n$ is an upper bound for c_n. For many operators, however, $\|E^n\| < \|E\|^n$. To improve the estimate, we use the geometric mean to define an average of norms for n applications of E by $\|E^n\|^{1/n}$. Similarly, the *spectral radius* is defined as

$$r(E) := \lim_{n\to\infty} \|E^n\|^{1/n}. \tag{1.65}$$

Thus, in the above sense it is the average of norms of infinitely many applications of E. The definition implies that for large n there is a small $\epsilon > 0$ such that

$$\|E^n\| \leq (r(E) + \epsilon)^n,$$

which yields that

$$\|E^n f\| \leq (r(E) + \epsilon)^n \cdot \|f\| .$$

The spectral radius also provides a criterion for $E^n \to 0$ for $n \to \infty$:

Proposition 1.26. *If $r(E) < 1$ then $\|E^n\| \to 0$ for $n \to \infty$, and as $\|\cdot\|$ is continuous $E^n \to 0$ for $n \to \infty$.*

Proof. As $r(E) < 1$, we can pick a number $q \in \mathbb{R}$ such that $r(E) < q < 1$. By definition of the spectral radius (1.65), there exists a number $n_0 \in \mathbb{N}$ such that

$$\|E^n\|^{1/n} \leq q \quad \text{for all} \quad n \geq n_0 .$$

Thus,

$$\|E^n\| \leq q^n .$$

The number $q < 1$. Hence, $q^n \to 0$ for $n \to \infty$ and so does $\|E^n\|$. □

We shall see in the next chapter that the Fourier symbol of an operator can be used to compute the operators norm and spectral radius.

1.7. Further Applications

The multigrid method can be used to efficiently solve discretizations of other partial differential equations (PDEs) than the Poisson equation. For instance, consider the Poisson-like diffusion problem

$$-\tfrac{\partial}{\partial x}(a\tfrac{\partial u}{\partial x}) - \tfrac{\partial}{\partial y}(a\tfrac{\partial u}{\partial y}) = f \quad \text{on } \Omega, \tag{1.66a}$$

$$u = g \quad \text{on } \partial\Omega, \tag{1.66b}$$

where $a : \Omega \to \mathbb{R}$ is a smooth, non-negative function. A finite difference discretization can be constructed [33, Section 5.1.4] leading to a linear system similar to the discrete Poisson equation (1.49). This linear system can be efficiently solved with the multigrid method described in Section 1.5.2.

The multigrid method, however, sometimes needs to be modified before it can be applied to a certain PDE. The method consists of different components—the smoother, the interpolation, the restriction, and the coarse grid approximation. These need to be chosen so that the interplay between them yields an efficient method. Occasionally, the Jacobi method is a bad choice or the linear interpolation does not work well. For example, consider the *anisotropic* problem

$$-\epsilon\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f \quad \text{on } \Omega$$

$$u = g \quad \text{on } \partial\Omega,$$

with $0 < \epsilon \ll 1$. This equation can be discretized using finite differences as well [68]. The Jacobi method is not a good smoother for this problem. It smooths only in the *direction of strong connection*, i.e., in y-direction. In the other direction, the error remains oscillatory.

There are two ways to construct an efficient multigrid method for this problem. Either one chooses a different smoother or one constructs a coarse grid approximation that is capable of approximating an error that is only slowly varying in one direction.

Furthermore, consider the PDE (1.66), but allow for a to be discontinuous. More precisely, assume that the domain Ω is split into two parts, i.e., $\Omega = \Omega_1 \cup \Omega_2$ and a jumps several orders of magnitude at the boundary between Ω_1 and Ω_2. In this situation the usual finite difference discretization is not feasible anymore. A finite volume discretization can instead be applied [68], for which the coarse grid correction needs adjustments.

The construction of multigrid methods for many different problems can be found in monographs [12, 15, 32, 63]. Especially [63] treats many problems from a practical point of view. Furthermore, people have been wondering, whether there is an automatic way of constructing multigrid methods. This resulted in the development of *algebraic multigrid methods* [15, 58, 59, 62, 66], which use heuristics to construct the components of a multigrid method.

1.8. Literature and Contributions

This chapter starts with the famous example of the deflection of a wire under load. This example is, e.g., discussed in [40]. We derived the equations describing the shape of the wire, using calculus of variations. The derivation is based on techniques found in, e.g., [6, 71]. For a more elementary derivation of similar problems see, e.g., [22, 67].

Finite Differences are a well established technique to approximate the solution of partial differential equations, and the convergence of the method is usually analyzed by proving consistency and stability separately. In this thesis, we followed the lines of [46]. Similar results can be found, e.g., in [33].

An introduction to multigrid methods followed the derivation of the finite difference discretization. Introductions to multigrid methods are given in [12, 15, 32, 59, 63, 68]. The presentation of the Jacobi method in this thesis was especially inspired by [59]. Analyzing multigrid methods in terms of spectral radii and norms of their error propagation matrix is a well known technique, which can be found, e.g., in [32, 63, 74]. In the literature the error propagation matrix is also known as *iteration matrix*.

After introducing multigrid methods, we discussed the formal eigenfunction analysis of them. This analysis can be found in the monographs [68, 74]. Formal eigenfunction analysis is an idealized analysis, however, for some cases it can be turned into a rigorous one, see [64].

2. Local Fourier Analysis

The Fourier symbol $\hat{a}$ of a constant stencil operator A tells us that the operator acts on a wave function ϕ_θ, by

$$A\phi_\theta = \hat{a}(\theta) \cdot \phi_\theta. \qquad \text{(1.54 revisited)}$$

Especially, it tells us whether a wave function with a certain frequency is damped or amplified, and if we have a linear combination of wave functions, we use $\hat{a}$ can determine how A acts on this combination; each part of the combination is treated separately and damped or amplified according to $\hat{a}(\theta)$.

This relation allowed us to determine if an operator is a smoother, i.e., if repeated application of the operator returns a slowly varying function. This is the case when $\hat{a}$ is small on the high frequencies and large on the low frequencies. The theory from the previous chapter allows us to make these predictions for linear combinations of wave functions. There are, however functions that cannot be written as a finite sum of wave functions. The question is: is it still true for such a function that repeated application of an operator returns a slowly varying function if $\hat{a}$ is small on the high frequencies and large on the low ones?

An example of a function that cannot be written as a finite sum of wave functions is the function $\delta : G_{\mathbf{h}} \to \mathbb{C}$ with

$$\delta(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{0}, \\ 0 & \text{otherwise.} \end{cases}$$

This function is important, e.g., as it can be used to obtain discrete analogs of Green's functions [18–20]. Hence, we would like to know how an operator acts on this function. Furthermore, any function with a bounded support cannot be written as a finite sum of wave functions. It is important to know whether an operator is a smoother for these functions as well. We shall show in this chapter that the Fourier symbol gives the desired information also in this case.

Another limitation of the theory from the previous chapter is that it does not allow us to determine the properties of an operator like the norm and the spectral radius. Assume we would like to determine the norm of a constant stencil operator $A : V \to V$ by using its Fourier symbol, where V is a suitable space. To compute the norm of A we would have to determine

$$\sup \left\{ \frac{\|Au\|}{\|u\|} : u \in V, u \neq 0 \right\}.$$

If $u \in V$, however, cannot be written as a finite sum of wave functions, the theory of the previous chapter gives no way to compute Au by using the Fourier symbol. Thus, to be able to use Fourier symbols we need to improve the theory we have so far. In this chapter we show that for a suitable space V there are simple formulas that allow to compute the operator norm and the spectral radius by using the Fourier symbol. Let us start with the required definitions.

Recall that the infinite grid with step-size $\mathbf{h}$ is

$$G_{\mathbf{h}} := \{\mathbf{h} \cdot \mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\} = \{(h_1 z_1, \ldots, h_d z_d)^T : \mathbf{z} \in \mathbb{Z}^d\}. \qquad \text{(1.51 revisited)}$$

A *grid function* assigns a complex number to every point of an infinite grid $G_{\mathbf{h}}$. Let $f, g : G_{\mathbf{h}} \to \mathbb{C}$ be two grid functions. We define their scalar product by

$$\langle f, g \rangle := \sum_{\mathbf{x} \in G_{\mathbf{h}}} \overline{g(\mathbf{x})} \cdot f(\mathbf{x}),$$

which induces the norm

$$\|f\| := \langle f, f \rangle^{1/2}.$$

A grid function f is *bounded* if $\|f\| < \infty$ and we denote the set of bounded grid functions as

$$\ell_2(G_{\mathbf{h}}) := \{f : G_{\mathbf{h}} \to \mathbb{C} : \|f\| < \infty\}.$$

It is well known that the set $\ell_2(G_{\mathbf{h}})$ is a Hilbert space [42]. This is the set of grid functions that we shall consider in the following.

It might seem arbitrary that we restrict ourselves to bounded grid functions. We are interested, however, in the norm of the operator, and therefore need to be able to compare the norm of a function before and after applying the operator, meaning that these norms need to be finite. There are of course other norms, however, the ℓ_2-norm is the most useful one for our purposes.

The main idea of this chapter will be to use the *discrete time Fourier transform* to transform functions from *position space* to *Fourier space* and perform the analysis in Fourier space. This idea first appeared in the local Fourier analysis literature in [11]. In this chapter we expand the idea from [11] in various ways. For some treatment of the discrete time Fourier transform of grid functions see also [35].

2.1. Operators on Infinite Grids

Before dealing with constant stencil operators consider the more general case of linear bounded operators that map from $\ell_2(G_{\mathbf{h}})$ to $\ell_2(G_{\mathbf{h}})$. A linear operator $A : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{h}})$ is bounded if its operator norm is finite. The *operator norm* is given by

$$\|A\| := \sup \left\{ \frac{\|Af\|}{\|f\|} : f \in \ell_2(G_{\mathbf{h}}), f \neq 0 \right\}.$$

We denote the set of all linear bounded operators by

$$L(\ell_2(G_{\mathbf{h}})) := \{A : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{h}}) : A \text{ is linear}, \|A\| < \infty\}.$$

In general, we will denote $L(X)$ to be

$$L(X; Y) := \{A : X \to Y : A \text{ is linear}, \|A\| < \infty\}$$

for any Banach space X. Furthermore, we write $L(X) := L(X; X)$.

2.1.1. Matrix Representation and Matrix Norms

Matrices are a simple representation for linear operators. They allow us to describe an operator just by an array of numbers, instead of a formula. This property makes matrices very useful. Like in the finite dimensional case we can define a matrix representation of operators in $L(\ell_2(G_\mathbf{h}))$ (cf. [42]). We shall use them later to show that any operator in $L(\ell_2(G_\mathbf{h}))$ has a stencil representation.

Theorem 2.1. *Let $A \in L(\ell_2(G_\mathbf{h}))$ and for $\mathbf{x} \in G_\mathbf{h}$ define the grid function $e_\mathbf{x}$ by $e_\mathbf{x}(\mathbf{z}) = \delta_{\mathbf{xz}}$. Then with $a_{\mathbf{xy}} := \langle Ae_\mathbf{y}, e_\mathbf{x}\rangle$*

$$[Au](\mathbf{x}) = \sum_{\mathbf{y}\in G_\mathbf{h}} a_{\mathbf{xy}} u(\mathbf{y})\,. \tag{2.1}$$

Proof. Let $f = Au$. The orthonormal basis $\{e_\mathbf{x} : \mathbf{x} \in G_\mathbf{h}\}$ of $\ell_2(G_\mathbf{h})$ permits us to write

$$f = \sum_{\mathbf{x}\in G_\mathbf{h}} f(\mathbf{x})e_\mathbf{x} \quad \text{and} \quad u = \sum_{\mathbf{y}\in G_\mathbf{h}} f(\mathbf{y})e_\mathbf{y}\,.$$

It follows that

$$f(\mathbf{x}) = \langle f, e_\mathbf{x}\rangle = \langle Au, e_\mathbf{x}\rangle = \langle A\sum_{\mathbf{y}\in G_\mathbf{h}} u(\mathbf{y})e_\mathbf{y}, e_\mathbf{x}\rangle = \sum_{\mathbf{y}\in G_\mathbf{h}} u(\mathbf{y})\cdot\langle Ae_\mathbf{y}, e_\mathbf{x}\rangle = \sum_{\mathbf{y}\in G_\mathbf{h}} u(\mathbf{y})\cdot a_{\mathbf{xy}}\,. \quad \square$$

Theorem 2.1 shows that every operator $A \in \ell_2(G_\mathbf{h})$ has a matrix representation. The converse, however, is not true. There are infinite matrices that do not define operators from $L(\ell_2(G_\mathbf{h}))$. For example if

$$a_{\mathbf{xy}} = 1 \quad \text{for all} \quad \mathbf{x}, \mathbf{y} \in G_\mathbf{h}\,.$$

the sum (2.1) gives an unbounded grid function if $u \neq 0$. Furthermore, it is easy to see that there are matrices $a_{\mathbf{xy}}$ and functions u such that the sum (2.1) diverges.

Let $a \in \mathbb{C}^{G_\mathbf{h}\times G_\mathbf{h}}$ be an infinite matrix. For a to define a bounded operator A two conditions have to be fulfilled. First, the sum (2.1) has to converge for every $u \in \ell_2(G_\mathbf{h})$. If this condition is fulfilled, Au is defined for any $u \in \ell_2(G_\mathbf{h})$. Second, the operator A has to be bounded, i.e., $\|A\| < \infty$. Let us give a sufficient condition for the sum (2.1) to converge.

Proposition 2.2. *Let $u \in \ell_2(G_\mathbf{h})$ and*

$$\sum_{\mathbf{y}\in G_\mathbf{h}} |a_{\mathbf{xy}}|^2 < \infty \quad \textit{for all} \quad \mathbf{x} \in G_\mathbf{h}. \tag{2.2}$$

Then (2.1) *converges for all $\mathbf{x} \in G_\mathbf{h}$.*

Proof. By the Cauchy-Schwarz inequality

$$\Big(\sum_{y\in G_\mathbf{h}} |a_{\mathbf{xy}}|\cdot|u(\mathbf{y})|\Big)^2 \le \Big(\sum_{y\in G_\mathbf{h}} |a_{\mathbf{xy}}|^2\Big)\cdot\Big(\sum_{y\in G_\mathbf{h}} |u(\mathbf{y})|^2\Big) < \infty.$$

Therefore, the sum (2.1) converges absolutely. $\square$

The following norms provide us with a sufficient condition for an operator that is given by a matrix to be well defined and bounded.

Definition 2.3. Let $A \in L(\ell_2(G_\mathbf{h}))$. The *Hilbert-Schmidt norm* of A is

$$\|A\|_{\mathrm{HS}} := \left(\sum_{\mathbf{x} \in G_\mathbf{h}} \sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}|^2 \right)^{1/2},$$

the *column sum norm* is

$$\|A\|_{\mathrm{CS}} := \sup_{\mathbf{y} \in G_\mathbf{h}} \sum_{\mathbf{x} \in G_\mathbf{h}} |a_{\mathbf{xy}}|,$$

and the *row sum norm* is

$$\|A\|_{\mathrm{RS}} := \sup_{\mathbf{x} \in G_\mathbf{h}} \sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}|.$$

The functions $\|\cdot\|_{\mathrm{HS}}$, $\|\cdot\|_{\mathrm{CS}}$, and $\|\cdot\|_{\mathrm{RS}}$ are actually norms for the spaces

$$\{A : \ell_2(G_\mathbf{h}) \to \ell_2(G_\mathbf{h}) : A \text{ is given by an infinite matrix, } \|A\|_{\mathrm{HS}} < \infty\},$$
$$\{A : \ell_2(G_\mathbf{h}) \to \ell_2(G_\mathbf{h}) : A \text{ is given by an infinite matrix, } \|A\|_{\mathrm{CS}} < \infty\}, \text{ and}$$
$$\{A : \ell_2(G_\mathbf{h}) \to \ell_2(G_\mathbf{h}) : A \text{ is given by an infinite matrix, } \|A\|_{\mathrm{CS}} < \infty\},$$

respectively. Furthermore, these functions can be used to show that a matrix defines an operator. To prove this statement, we need to first prove an auxiliary result.

Lemma 2.4. *Let* $1 \le p < q < \infty$ *and* $\{a_k\}_{k=1}^{\infty}$ *be a sequence such that*

$$\sum_{k=1}^{\infty} |a_k|^p < \infty. \tag{2.3}$$

Then

$$\sum_{k=1}^{\infty} |a_k|^q < \infty.$$

Proof. The sum (2.3) of the elements of the series is a finite value thus $a_k \to 0$ for $k \to \infty$. Therefore, only finitely many $|a_k|^p$ are larger than one, meaning that we can find $m \in \mathbb{N}$ such that

$$|a_k|^p \le 1 \quad \text{for all} \quad k \ge m.$$

Thus, for $k \ge m$

$$|a_k|^q = \left(|a_k|^p\right)^{q/p} \le |a_k|^p.$$

Consider

$$\sum_{k=1}^{\infty} |a_k|^q = \sum_{k=1}^{m-1} |a_k|^q + \sum_{k=m}^{\infty} |a_k|^q.$$

The first sum on the right hand side of this equation is finite. Furthermore, the second sum on the right hand side of the equation can be bounded by

$$\sum_{k=m}^{\infty} |a_k|^q \leq \sum_{k=m}^{\infty} |a_k|^p$$

and is therefore also finite. □

Proposition 2.5. *Let $a \in \mathbb{C}^{G_\mathbf{h} \times G_\mathbf{h}}$ be an infinite matrix. If $\|A\|_{\mathrm{HS}} < \infty$ or $\|A\|_{\mathrm{RS}} < \infty$ then* (2.1) *converges for all $\mathbf{x} \in G_\mathbf{h}$.*

Proof. It is easy to see that from $\|A\|_{\mathrm{HS}} < \infty$ condition (2.2) follows. Using Lemma 2.4 we find that from $\|A\|_{\mathrm{RS}} < \infty$ the condition (2.2) follows. □

The operator norm can be bounded in terms of the Hilbert-Schmidt or the column-sum and the row-sum norm [42].

Proposition 2.6. *Let A be given by the infinite matrix $a \in \mathbb{C}^{G_\mathbf{h} \times G_\mathbf{h}}$; then*

$$\|A\| \leq \|A\|_{\mathrm{HS}} \quad \textit{and} \tag{2.4}$$

$$\|A\| \leq \sqrt{\|A\|_{\mathrm{CS}} \cdot \|A\|_{\mathrm{RS}}}\,. \tag{2.5}$$

Proof. Let $u \in \ell_2(G_\mathbf{h})$. We start with the definition of the norm

$$\|Au\|^2 := \sum_{\mathbf{x} \in G_\mathbf{h}} \left| \sum_{\mathbf{y} \in G_\mathbf{h}} a_{\mathbf{xy}} u(\mathbf{y}) \right|^2 \leq \sum_{\mathbf{x} \in G_\mathbf{h}} \left(\sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}| \cdot |u(\mathbf{y})| \right)^2$$

The term in brackets is a scalar product; the Cauchy-Schwarz inequality gives

$$\begin{aligned} \|Au\|^2 &\leq \sum_{\mathbf{x} \in G_\mathbf{h}} \left(\left(\sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}|^2 \right)^{1/2} \left(\sum_{\mathbf{y} \in G_\mathbf{h}} |u(\mathbf{y})|^2 \right)^{1/2} \right)^2 \\ &= \left(\sum_{\mathbf{x} \in G_\mathbf{h}} \sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}|^2 \right) \|u\|^2 \\ &= \|A\|_{\mathrm{HS}}^2 \cdot \|u\|^2, \end{aligned}$$

which proves the Hilbert-Schmidt norm inequality (2.4).

To prove the other inequality consider that

$$\|Au\|^2 \leq \sum_{\mathbf{x} \in G_\mathbf{h}} \left(\sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}| \cdot |u(\mathbf{y})| \right)^2 = \sum_{\mathbf{x} \in G_\mathbf{h}} \left(\sum_{\mathbf{y} \in G_\mathbf{h}} |a_{\mathbf{xy}}|^{1/2} \cdot \left(|a_{\mathbf{xy}}|^{1/2} \cdot |u(\mathbf{y})| \right) \right)^2 .$$

By the Cauchy-Schwarz inequality

$$\begin{aligned}\|Au\|^2 &\le \sum_{\mathbf{x}\in G_\mathbf{h}} \left(\sum_{\mathbf{y}\in G_\mathbf{h}} |a_{\mathbf{xy}}|\right)\cdot\left(\sum_{\mathbf{y}\in G_\mathbf{h}} |a_{\mathbf{xy}}|\cdot|u(\mathbf{y})|^2\right) \le \sum_{\mathbf{x}\in G_\mathbf{h}} \|A\|_{\mathrm{RS}}\cdot\left(\sum_{\mathbf{y}\in G_\mathbf{h}} |a_{\mathbf{xy}}|\cdot|u(\mathbf{y})|^2\right)\\ &= \|A\|_{\mathrm{RS}} \sum_{\mathbf{y}\in G_\mathbf{h}}\sum_{\mathbf{x}\in G_\mathbf{h}} |a_{\mathbf{xy}}|\cdot|u(\mathbf{y})|^2 \le \|A\|_{\mathrm{RS}} \sum_{\mathbf{y}\in G_\mathbf{h}} \left(\sum_{\mathbf{x}\in G_\mathbf{h}} |a_{\mathbf{xy}}|\right)|u(\mathbf{y})|^2\\ &\le \|A\|_{\mathrm{RS}} \sum_{\mathbf{y}\in G_\mathbf{h}} \|A\|_{\mathrm{CS}}\cdot|u(\mathbf{y})|^2 = \|A\|_{\mathrm{RS}}\cdot\|A\|_{\mathrm{CS}}\cdot\|u(\mathbf{y})\|^2 .\end{aligned}$$

Hence, the row/column-sum estimate (2.5) is established. □

From Proposition 2.5 and Proposition 2.6 we know that if either $\|A\|_{\mathrm{HS}}$ or $\sqrt{\|A\|_{\mathrm{CS}}\cdot\|A\|_{\mathrm{RS}}}$ are bounded then $A \in L(\ell_2(G_\mathbf{h}))$.

2.1.2. Stencil Representation

In many grid based applications stencils are a convenient description for operators (cf. Section 1.4.1 and 1.5.2). We shall see that every operator in $L(\ell_2(G_\mathbf{h}))$ can be represented by a variable stencil. A *variable stencil* on an infinite, dD grid is a family $\{s_\mathbf{x}\}_{\mathbf{x}\in G_\mathbf{h}}$ with $s_\mathbf{x} : G_\mathbf{h} \to \mathbb{C}$; the corresponding stencil operator A is defined by

$$[Au](\mathbf{x}) := \sum_{\mathbf{y}\in G_\mathbf{h}} s_\mathbf{x}(\mathbf{y})\cdot u(\mathbf{x}+\mathbf{y}) .$$

Theorem 2.7. *Let $A \in L(\ell_2(G_\mathbf{h}))$. Then A has a variable stencil representation.*

Proof. According to Theorem 2.1, the operator A has a matrix representation $a \in \mathbb{C}^{G_\mathbf{h}\times G_\mathbf{h}}$. Let

$$s_\mathbf{x}(\mathbf{y}) := a_{\mathbf{x},(\mathbf{x}+\mathbf{y})} .$$

This definition implies that the corresponding stencil operator $\widetilde{A}$ fulfills that

$$[\widetilde{A}u](\mathbf{x}) = \sum_{\mathbf{y}\in G_\mathbf{h}} s_\mathbf{x}(\mathbf{y})\cdot u(\mathbf{x}+\mathbf{y}) = \sum_{\mathbf{y}\in G_\mathbf{h}} a_{\mathbf{x},\mathbf{x}+\mathbf{y}}\cdot u(\mathbf{x}+\mathbf{y}) .$$

Substituting $\mathbf{y}-\mathbf{x}$ for $\mathbf{y}$ gives that

$$[\widetilde{A}u](\mathbf{x}) = \sum_{(\mathbf{y}-\mathbf{x})\in G_\mathbf{h}} a_{\mathbf{x},\mathbf{y}}\cdot u(\mathbf{y}) = \sum_{\mathbf{y}\in G_\mathbf{h}} a_{\mathbf{x},\mathbf{y}}\cdot u(\mathbf{y}) .$$

Comparing the last term to the matrix representation of A,

$$[Au](\mathbf{x}) = \sum_{\mathbf{y}\in G_\mathbf{h}} a_{\mathbf{xy}}u(\mathbf{y}), \qquad \text{(2.1 revisited)}$$

we conclude that $\widetilde{A} = A$, i.e., the stencil s represents the operator A. □

Let us turn to *constant stencil* operators. A stencil $\tilde{s}_{\mathbf{x}}$ is called *constant* if there exists $s : G_{\mathbf{h}} \to \mathbb{C}$ such that

$$\tilde{s}_{\mathbf{x}} = s \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{h}} .$$

Thus, a *constant stencil operator* is given by

$$[Au](\mathbf{x}) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot u(\mathbf{x} + \mathbf{y}). \tag{1.52 revisited}$$

Like in the case of infinite matrices, there are stencils that do not give a well-defined and bounded operator. For this reason, it is useful to have a sufficient condition that guarantees that a stencil describes a well-defined and bounded operator.

Theorem 2.8. *Let A be an operator given by the constant stencil $s : G_{\mathbf{h}} \to \mathbb{C}$; then*

$$\|A\| \leq \sum_{x \in G_{\mathbf{h}}} |s(x)| .$$

Proof. By the definition of the column sum norm

$$\|A\|_{\mathrm{CS}} := \sup_{\mathbf{y} \in G_{\mathbf{h}}} \sum_{\mathbf{x} \in G_{\mathbf{h}}} |a_{\mathbf{xy}}| = \sup_{\mathbf{y} \in G_{\mathbf{h}}} \sum_{\mathbf{x} \in G_{\mathbf{h}}} |s(\mathbf{y} - \mathbf{x})| = \sup_{\mathbf{y} \in G_{\mathbf{h}}} \sum_{\mathbf{x} \in G_{\mathbf{h}}} |s(\mathbf{x})| = \sum_{\mathbf{x} \in G_{\mathbf{h}}} |s(\mathbf{x})| .$$

Similarly,

$$\|A\|_{\mathrm{RS}} := \sup_{\mathbf{x} \in G_{\mathbf{h}}} \sum_{\mathbf{y} \in G_{\mathbf{h}}} |a_{\mathbf{xy}}| = \sup_{\mathbf{x} \in G_{\mathbf{h}}} \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(\mathbf{y} - \mathbf{x})| = \sup_{\mathbf{x} \in G_{\mathbf{h}}} \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(\mathbf{y})| = \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(\mathbf{y})| .$$

By Proposition 2.6,

$$\|A\| \leq \sqrt{\|A\|_{\mathrm{CS}} \cdot \|A\|_{\mathrm{RS}}} = \sum_{\mathbf{x} \in G_{\mathbf{h}}} |s(\mathbf{x})| . \qquad \square$$

In particular, Theorem 2.8 implies that a stencil operator with a finite number of non-zero entries represents a bounded operator from $L(\ell_2(G_{\mathbf{h}}))$.

2.1.3. Shift Invariance Characterization

Sometimes it is helpful to know whether the stencil of an operator in $L(\ell_2(G_{\mathbf{h}}))$ is constant or not without actually computing the stencil. In what follows, we introduces *shift invariance* and its relationship to constant stencils.

Let $\mathbf{z} \in G_{\mathbf{h}}$ and let the *shift operator* $S_{\mathbf{z}} : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{h}})$ be defined by

$$[S_{\mathbf{z}} u](\mathbf{x}) = u(\mathbf{x} - \mathbf{z}) . \tag{2.6}$$

An operator $A \in L(\ell_2(G_{\mathbf{h}}))$ is *shift invariant* if

$$AS_{\mathbf{z}} = S_{\mathbf{z}} A \quad \text{for all} \quad \mathbf{z} \in G_{\mathbf{h}} .$$

Theorem 2.9. *An operator $A \in L(\ell_2(G_{\mathbf{h}}))$ is shift invariant if and only if it is represented by a constant stencil.*

Proof. Assume that A is represented by the constant stencil t; then

$$(AS_{\mathbf{z}}u)(\mathbf{x}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} t(\mathbf{y}) \cdot (S_{\mathbf{z}}u)(\mathbf{x}+\mathbf{y}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} t(\mathbf{y}) \cdot u(\mathbf{x}+\mathbf{y}-\mathbf{z})$$

and

$$(S_{\mathbf{z}}Au)(\mathbf{x}) = (Au)(\mathbf{x}-\mathbf{z}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} t(\mathbf{y}) \cdot u(\mathbf{x}-\mathbf{z}+\mathbf{y})\,.$$

Thus, A is shift invariant.

Conversely, assume that A is shift invariant and that s is the stencil representing A. If $e_{\mathbf{z}}$ is the canonical basis vector given by $e_{\mathbf{z}}(\mathbf{x}) = \delta_{\mathbf{xz}}$ then

$$[Ae_{\mathbf{z}}](\mathbf{x}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s_{\mathbf{x}}(\mathbf{y}) \cdot e_{\mathbf{z}}(\mathbf{x}+\mathbf{y}) = s_{\mathbf{x}}(\mathbf{z}-\mathbf{x})\,. \tag{2.7}$$

It follows that for $\mathbf{v}, \mathbf{w} \in G_{\mathbf{h}}$

$$(AS_{-\mathbf{v}}e_{\mathbf{v}+\mathbf{w}})(\mathbf{0}) = (Ae_{\mathbf{w}})(\mathbf{0}) = s_{\mathbf{0}}(\mathbf{w})\,.$$

If we use (2.7) again then

$$(S_{-\mathbf{v}}Ae_{\mathbf{v}+\mathbf{w}})(\mathbf{0}) = (Ae_{\mathbf{v}+\mathbf{w}})(\mathbf{v}) = s_{\mathbf{v}}(\mathbf{v}+\mathbf{w}-\mathbf{v}) = s_{\mathbf{v}}(\mathbf{w})\,.$$

Combining the last two equations and using the shift invariance of A yields

$$s_{\mathbf{0}}(\mathbf{w}) = s_{\mathbf{v}}(\mathbf{w}) \quad \text{for all} \quad \mathbf{v}, \mathbf{w} \in G_{\mathbf{h}}\,. \qquad \square$$

2.2. Fourier Representation

We shall introduce the *Fourier representation* of operators in $L(\ell_2(G_{\mathbf{h}}); \ell_2(G_{\mathbf{h}'}))$, where $\mathbf{h} \in \mathbb{R}^d$ and $\mathbf{h}' \in \mathbb{R}^d$ are (potentially) different step-sizes of two grids. Every operator has a Fourier representation and vice versa. The Fourier representation of a constant stencil operator is especially simple and directly related to its Fourier symbol.

2.2.1. Discrete Time Fourier Transform

To introduce the Fourier representation of an operator we first have to define the discrete time Fourier transform and derive some of its properties. For this purpose, we define the wave functions

$$\psi(\theta, \mathbf{x}) := \frac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} \quad \text{for} \quad \theta \in \mathbb{R}^d, \mathbf{x} \in G_{\mathbf{h}}, \tag{2.8}$$

where $\mathrm{vol}_{\mathbf{h}} := h_1 h_2 \cdots h_d$. Despite the factor $\frac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}}$, these are just the wave functions (1.53).

Furthermore, we need to introduce the space $L_2(\Theta_{\mathbf{h}})$. This space consists of all functions[1] $\hat{f} : \Theta_{\mathbf{h}} \to \mathbb{C}$ that are bounded by the norm induced by the scalar product

$$\langle \hat{f}, \hat{g} \rangle := \int_{\Theta_{\mathbf{h}}} \overline{\hat{g}(\theta)} \cdot \hat{f}(\theta)\, \mathrm{d}\theta\,.$$

[1] With "function" we actually mean the equivalence class of functions that are equal almost everywhere.

In other words, $\|f\| := \langle \hat{f}, \hat{f} \rangle^{1/2}$, and

$$L_2(\Theta_{\mathbf{h}}) = \{\hat{f} : \Theta_{\mathbf{h}} \to \mathbb{C} : \|\hat{f}\| < \infty\}\,.$$

We can now use the wave functions (2.8) to define the *discrete time Fourier transform* (DTFT). The transform maps a function $f \in \ell_2(G_{\mathbf{h}})$ to a function $\hat{f} \in L_2(\Theta_{\mathbf{h}})$, which describes how f can be represented by a combination of the wave functions $\psi(\theta, \mathbf{x})$.

Definition 2.10. Let $f \in \ell_2(G_{\mathbf{h}})$. The discrete time Fourier transform $\mathcal{F}_{\mathbf{h}} : \ell_2(G_{\mathbf{h}}) \to L_2(\Theta_{\mathbf{h}})$ is

$$(\mathcal{F}_{\mathbf{h}} f)(\theta) = \sum_{\mathbf{x} \in G_{\mathbf{h}}} f(\mathbf{x}) \cdot \overline{\psi(\theta, \mathbf{x})}\,. \tag{2.9}$$

In the following theorem we state the most important facts about the DTFT.

Theorem 2.11. *The discrete time Fourier transform has the following properties.*

1. *It is well defined.*
2. *It is an isomorphism—linear and bijective.*
3. *Its inverse is*
$$(\mathcal{F}_{\mathbf{h}}^{-1} \hat{f})(\mathbf{x}) = \int_{\Theta_{\mathbf{h}}} \hat{f}(\theta) \cdot \psi(\theta, \mathbf{x})\, \mathrm{d}\theta\,. \tag{2.10}$$
4. *It preserves the scalar product, i.e.,*
$$\langle \mathcal{F}_{\mathbf{h}} f, \mathcal{F}_{\mathbf{h}} g \rangle = \langle f, g \rangle\,. \tag{2.11}$$
5. *It is an isometry, i.e.,*
$$\|\mathcal{F}_{\mathbf{h}} f\| = \|f\|\,. \tag{2.12}$$

Proof. We proceed by proving the individual statements.

1. The set of wave functions $\left\{\overline{\psi(\cdot, \mathbf{x})} : \mathbf{x} \in G_{\mathbf{h}}\right\}$ is—as shown in Appendix A—an orthonormal basis of the space $L_2(\Theta_{\mathbf{h}})$. Thus, for every $f \in \ell_2(G_{\mathbf{h}})$
$$\sum_{\mathbf{x} \in G_{\mathbf{h}}} f(x) \cdot \overline{\psi(\cdot, \mathbf{x})} \in L_2(\Theta_{\mathbf{h}}),$$
meaning that $\mathcal{F}_{\mathbf{h}}$ is well defined.
2. First, it is easily seen from the definition of the DTFT (2.9) that the DTFT is a linear map.
Second, the fact that the wave functions form an orthonormal basis implies that every $\hat{f} \in L_2(\Theta_{\mathbf{h}})$ can be written in the form of
$$\hat{f} = \sum_{\mathbf{x} \in G_{\mathbf{h}}} f(\mathbf{x}) \cdot \overline{\psi(\cdot, \mathbf{x})}\,.$$
for some $f \in \ell_2(G_{\mathbf{h}})$. Therefore, $\mathcal{F}_{\mathbf{h}}$ is surjective.
Third, to show that $\mathcal{F}_{\mathbf{h}}$ is also injective let $f \in \ell_2(G_{\mathbf{h}})$ and
$$\hat{f} := \mathcal{F}_{\mathbf{h}} f = \sum_{\mathbf{x} \in G_{\mathbf{h}}} f(\mathbf{x}) \cdot \psi(\cdot, \mathbf{x})\,.$$

We want to show that the operator $\mathcal{F}_{\mathbf{h}}^{-1}$ defined by (2.10) fulfills $\mathcal{F}_{\mathbf{h}}^{-1}\mathcal{F}_{\mathbf{h}} f = f$. Using the orthogonality of the wave functions we have that

$$\begin{aligned}(\mathcal{F}_{\mathbf{h}}^{-1}\hat{f})(\mathbf{x}') &= \int_{\Theta_{\mathbf{h}}} \hat{f}(\theta)\cdot\psi(\theta,\mathbf{x})\,\mathrm{d}\theta = \langle\overline{\psi(\cdot,\mathbf{x}')},\hat{f}\rangle \\ &= \langle\overline{\psi(\cdot,\mathbf{x}')},\sum_{\mathbf{x}\in G_{\mathbf{h}}} f(\mathbf{x})\cdot\overline{\psi(\cdot,\mathbf{x})}\rangle \\ &= \sum_{\mathbf{x}\in G_{\mathbf{h}}} f(\mathbf{x})\cdot\langle\overline{\psi(\cdot,\mathbf{x})},\overline{\psi(\cdot,\mathbf{x}')}\rangle = f(\mathbf{x}')\,.\end{aligned}$$

Thus, $\mathcal{F}_{\mathbf{h}}^{-1}\mathcal{F}_{\mathbf{h}} f = f$, and therefore $\mathcal{F}_{\mathbf{h}}$ is injective.

3. As we already know that $\mathcal{F}_{\mathbf{h}}$ is surjective, the fact that $\mathcal{F}_{\mathbf{h}}^{-1}\mathcal{F}_{\mathbf{h}} f = f$ also implies that $\mathcal{F}_{\mathbf{h}}^{-1}$ is the unique inverse of $\mathcal{F}_{\mathbf{h}}$.
4. Let $\hat{f},\hat{g}\in L_2(\Theta_{\mathbf{h}})$. Then

$$\langle\hat{f},\hat{g}\rangle = \sum_{\mathbf{x}\in G_{\mathbf{h}}}\langle\hat{f},\psi(\cdot,\mathbf{x})\rangle\langle\psi(\cdot,\mathbf{x}),\hat{g}\rangle = \langle\mathcal{F}_{\mathbf{h}}^{-1}\hat{f},\mathcal{F}_{\mathbf{h}}^{-1}\hat{g}\rangle\,.$$

Thus, by substituting $\hat{f}$ and $\hat{g}$ by $\mathcal{F}_{\mathbf{h}} f$ and $\mathcal{F}_{\mathbf{h}} g$ we obtain

$$\langle\mathcal{F}_{\mathbf{h}} f,\mathcal{F}_{\mathbf{h}} g\rangle = \langle f,g\rangle\,.$$

5. The last equation also implies that

$$\|\mathcal{F}_{\mathbf{h}} f\| = \|f\|. \qquad \square$$

As $\mathcal{F}_{\mathbf{h}}$ is an isomorphism, we can say that every grid function has two representations: a representation f in *position space* and a representation $\hat{f}$ in *Fourier space*. These representations are related by $\hat{f} = \mathcal{F}_{\mathbf{h}} f$. Theorem 2.11 shows that the DTFT preserves many important properties of grid functions, especially the scalar product. Consequently, when we are interested in any of these properties of a certain function, we can choose to look at the position space or Fourier space representation of this function, and use the more convenient representation. This representation will often be the Fourier space representation. Furthermore, Theorem 2.11 gives the formula for the inverse of the DTFT (2.10), which shows that the grid function is represented by a combination of wave functions.

If we shift a function, then the DTFT of the shifted function can be computed from the DTFT of the original one, which is another property of the DTFT that we will use a few times.

Lemma 2.12. *Let $f\in\ell_2(G_{\mathbf{h}})$ and $S_{\mathbf{z}}$ be the shift operator* (2.6)*. We have*

$$[\mathcal{F}_{\mathbf{h}} S_{\mathbf{z}} f](\theta) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle}\cdot[\mathcal{F}_{\mathbf{h}} f](\theta). \tag{2.13}$$

Proof. The DTFT of $S_{\mathbf{z}} f$ fulfills

$$[\mathcal{F}_{\mathbf{h}}[S_{\mathbf{z}} f]](\theta) = \sum_{\mathbf{x}\in G_{\mathbf{h}}}\big[S_{\mathbf{z}} f\big](\mathbf{x})\cdot\overline{\psi(\theta,\mathbf{x})} = \sum_{\mathbf{x}\in G_{\mathbf{h}}} f(\mathbf{x}-\mathbf{z})\cdot\frac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}}e^{-\mathrm{i}\langle\theta,\mathbf{x}\rangle}\,.$$

Substituting $(\mathbf{x}+\mathbf{z})$ for $\mathbf{x}$ yields that

$$\begin{aligned}[\mathcal{F}_{\mathbf{h}}[S_{\mathbf{z}}f]](\theta) &= \sum_{\mathbf{x}\in G_{\mathbf{h}}} f(x)\cdot \tfrac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} e^{-\mathrm{i}\langle\theta,\mathbf{x}+\mathbf{z}\rangle} = \sum_{\mathbf{x}\in G_{\mathbf{h}}} f(x)\cdot \tfrac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} e^{-\mathrm{i}\langle\theta,\mathbf{x}\rangle}\cdot e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \\ &= e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle}\cdot\left(\sum_{\mathbf{x}\in G_{\mathbf{h}}} f(x)\cdot \tfrac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} e^{-\mathrm{i}\langle\theta,\mathbf{x}\rangle}\right) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle}\cdot\left(\sum_{\mathbf{x}\in G_{\mathbf{h}}} f(x)\cdot\overline{\psi(\theta,\mathbf{x})}\right) \\ &= e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle}\cdot\left[\mathcal{F}_{\mathbf{h}}f\right](\theta)\,. \end{aligned}$$

□

We shall now consider, how to determine the amount of oscillation of a general grid function $f\in\ell_2(G_{\mathbf{h}})$. We measure the amount of oscillation by comparing the size of the function value at neighboring grid points. Thus, if $\Delta\mathbf{x}=h_k\mathbf{e}_k$, the amount of oscillation in direction of x_k is

$$\frac{\|f-S_{-\Delta\mathbf{x}}f\|}{\|f\|}.$$

We show that the numerator can be computed using the DTFT.

Theorem 2.13. *Let $\Delta\mathbf{x}\in G_{\mathbf{h}}$, $f\in\ell_2(G_{\mathbf{h}})$ and $S_{\mathbf{z}}$ be the shift operator* (2.6)*. Then*

$$\|f-S_{-\Delta\mathbf{x}}f\|^2=\int_{\Theta_{\mathbf{h}}}\left|\hat{f}(\theta)\right|^2\cdot\left(2-2\cos\langle\theta,\Delta\mathbf{x}\rangle\right)\mathrm{d}\theta \tag{2.14}$$

Proof. From Lemma 2.12 we have

$$[\mathcal{F}_{\mathbf{h}}S_{-\Delta\mathbf{x}}f](\theta)=\hat{f}(\theta)\cdot e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}.$$

Thus, by the linearity of the DTFT

$$[\mathcal{F}_{\mathbf{h}}(f-S_{-\Delta\mathbf{x}}f)](\theta)=\hat{f}(\theta)-\hat{f}(\theta)\cdot e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}=\hat{f}(\theta)\left(1-e^{\mathrm{i}\langle\theta,\mathbf{h}\rangle}\right).$$

The DTFT preserves the norm; thus

$$\begin{aligned}\|f-S_{-\Delta\mathbf{x}}f\|^2=\|\mathcal{F}_{\mathbf{h}}(f-S_{-\Delta\mathbf{x}}f)\|^2&=\int_{\Theta_{\mathbf{h}}}\left|\hat{f}(\theta)\left(1-e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right)\right|^2\mathrm{d}\theta\\&=\int_{\Theta_{\mathbf{h}}}\left|\hat{f}(\theta)\right|^2\cdot\left|1-e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right|^2\mathrm{d}\theta\end{aligned}$$

Recall the simplification (1.45), i.e.,

$$\left|1-e^{\mathrm{i}\langle\theta,\Delta\mathbf{x}\rangle}\right|^2=2-2\cos\langle\theta,\Delta\mathbf{x}\rangle.$$

Thus, by combining the last two equations the assertion is proved. □

Let us interpret the formula (2.14). First, assume the large function values of f are concentrated around function arguments θ where $(2-2\cos\langle\theta,\Delta\mathbf{x}\rangle)$ is *large*. Then the integral (2.14) is large in comparison to $\|f\|$. Thus, the function values differ strongly in the direction of $\Delta\mathbf{x}$, and therefore the function is oscillatory in the direction $\Delta\mathbf{x}$.

$$\begin{array}{ccc} \ell_2(G_{\mathbf{h}}) & \xrightarrow{\mathcal{F}_{\mathbf{h}}} & L_2(\Theta_{\mathbf{h}}) \\ A \downarrow & & \downarrow \widehat{A} \\ \ell_2(G_{\mathbf{h}'}) & \xrightarrow[\mathcal{F}_{\mathbf{h}'}]{} & L_2(\Theta_{\mathbf{h}}) \end{array}$$

Figure 2.1.: Relation between a bounded operator A and its Fourier representation $\widehat{A}$.

Second, assume that the large function values of f are concentrated around function arguments θ where $(2 - 2\cos\langle\theta, \Delta\mathbf{x}\rangle)$ is *small*. Then the integral is small in comparison to $\|f\|$, and thus the function is slowly varying in the direction of $\Delta\mathbf{x}$.

Note that we already encountered the function $(2-\cos\langle\theta, \Delta\mathbf{x}\rangle)$ when discussing the amount of oscillation of wave functions in Section 1.5.5. The function value is large when θ is a high frequency and small when θ is a low frequency. In other words, a function is slowly varying if the large function values of its Fourier transform are concentrated at low frequency function arguments and oscillatory otherwise.

In Chapter 1 we used low and high frequencies to determine the amount of oscillation of *wave functions*. Theorem 2.13 now allows us to determine the amount of oscillation of *general functions* from $\ell_2(G_{\mathbf{h}})$.

2.2.2. Fourier Representation

The DTFT $\hat{f}$ of a grid function f gives us information about the amount of oscillation of a function. When analyzing multigrid methods we are interested in this information, e.g., to determine whether an operator is a suitable smoother for a specific problem or not. Hence, if an operator maps the function $u \mapsto f$ in position space, we want to know the corresponding operator in Fourier space that maps $\hat{u} \mapsto \hat{f}$. This action on $\hat{f}$ is the Fourier representation of A, which is defined as follows.

Definition 2.14. Let $A \in L(\ell_2(G_{\mathbf{h}}); \ell_2(G_{\mathbf{h}'}))$. The *Fourier representation of the operator A* is

$$\widehat{A} := \mathcal{F}_{\mathbf{h}'} A \mathcal{F}_{\mathbf{h}}^{-1} .$$

The relation between A and its Fourier representation $\widehat{A}$ is depicted by the diagram in Figure 2.1. As the diagram commutes, the Fourier representation $\widehat{A}$ contains of course all the information of A, since we can reconstruct A from $\widehat{A}$ by

$$A = \mathcal{F}_{\mathbf{h}'}^{-1} \widehat{A} \mathcal{F}_{\mathbf{h}} .$$

Furthermore, we can show that many operations on the operator A directly carry over to $\widehat{A}$,

Theorem 2.15. *Let $A, B \in L(\ell_2(G_{\mathbf{h}}); \ell_2(G_{\mathbf{h}'}))$, $C \in L(\ell_2(G_{\mathbf{h}'}); \ell_2(G_{\mathbf{h}''}))$, and the operators $\widehat{A}, \widehat{B} \in L(L_2(\Theta_{\mathbf{h}}); L_2(\Theta_{\mathbf{h}'}))$, $\widehat{C} \in L(L_2(\Theta_{\mathbf{h}'}); L_2(\Theta_{\mathbf{h}''}))$ their Fourier representations. Furthermore, let $\lambda \in \mathbb{C}$. Then the Fourier representation of*

1. $\lambda A + B$ *is* $\lambda\widehat{A} + \widehat{B}$,
2. CB *is* $\widehat{C}\widehat{B}$,
3. A^{-1} *is* $\widehat{A}^{-1}$, *and*
4. A^* *is* $\widehat{A}^*$.

Proof. Let us prove the statements individually.

1. Using the linearity of the DTFT, proven in Theorem 2.11, we have for all $\hat{u} \in L_2(\Theta_{\mathbf{h}})$
$$\mathcal{F}_{\mathbf{h}'}(\lambda A + B)\mathcal{F}_{\mathbf{h}}^{-1}\hat{u} = \lambda\mathcal{F}_{\mathbf{h}'}A\mathcal{F}_{\mathbf{h}}^{-1}\hat{u} + \mathcal{F}_{\mathbf{h}'}B\mathcal{F}_{\mathbf{h}}^{-1}\hat{u} = (\lambda\widehat{A} + \widehat{B})\hat{u}\,.$$
2. Using that $\mathcal{F}_{\mathbf{h}}^{-1}\mathcal{F}_{\mathbf{h}}$ is the identity, we obtain
$$\mathcal{F}_{\mathbf{h}''}CB\mathcal{F}_{\mathbf{h}'}^{-1} = \mathcal{F}_{\mathbf{h}''}C\mathcal{F}_{\mathbf{h}'}^{-1}\mathcal{F}_{\mathbf{h}'}B\mathcal{F}_{\mathbf{h}}^{-1} = \widehat{C}\widehat{B}\,.$$
3. By properties of the inverse of the composition of operators,
$$\mathcal{F}_{\mathbf{h}}A^{-1}\mathcal{F}_{\mathbf{h}}^{-1} = (\mathcal{F}_{\mathbf{h}}A\mathcal{F}_{\mathbf{h}}^{-1})^{-1} = \widehat{A}^{-1}\,.$$
Note that as A is invertible, we have that $\mathbf{h} = \mathbf{h}'$.
4. We start with
$$\langle Af, g\rangle = \langle f, A^*g\rangle\,.$$
The scalar product is invariant under the DTFT (2.11), and thus,
$$\langle\mathcal{F}_{\mathbf{h}'}Af, \mathcal{F}_{\mathbf{h}'}g\rangle = \langle\mathcal{F}_{\mathbf{h}}f, \mathcal{F}_{\mathbf{h}}A^*g\rangle$$
On the other hand
$$\langle\mathcal{F}_{\mathbf{h}'}Af, \mathcal{F}_{\mathbf{h}'}g\rangle = \langle\widehat{A}\mathcal{F}_{\mathbf{h}}f, \mathcal{F}_{\mathbf{h}'}g\rangle = \langle\mathcal{F}_{\mathbf{h}}f, \widehat{A}^*\mathcal{F}_{\mathbf{h}'}g\rangle\,.$$
Combining the last two equations yields that
$$\langle\mathcal{F}_{\mathbf{h}}f, \mathcal{F}_{\mathbf{h}}A^*g\rangle = \langle\mathcal{F}_{\mathbf{h}}f, \widehat{A}^*\mathcal{F}_{\mathbf{h}'}g\rangle\,.$$
As this equation has to hold for arbitrary f and g, we obtain that
$$\mathcal{F}_{\mathbf{h}}A^* = \widehat{A}^*\mathcal{F}_{\mathbf{h}}\,.$$
□

The DTFT preserves norms as shown in Theorem 2.11; it is no surprise that the Fourier representation of an operator has the same norm as the operator itself.

Proposition 2.16. *Let* $\widehat{A}$ *be the Fourier representation of* A. *Then* $\|\widehat{A}\| = \|A\|$.

Proof. By definition of the Fourier representation
$$\|\widehat{A}\| = \max_{\|\hat{f}\|=1}\|\widehat{A}\hat{f}\| = \max_{\|\hat{f}\|=1}\|\mathcal{F}_{\mathbf{h}'}A\mathcal{F}_{\mathbf{h}}^{-1}\hat{f}\|\,.$$
We substitute $\hat{f}$ by $\mathcal{F}_{\mathbf{h}}f$ and obtain that
$$\|\widehat{A}\| = \max_{\|\mathcal{F}_{\mathbf{h}}f\|=1}\|\mathcal{F}_{\mathbf{h}'}Af\|\,.$$
Since the DTFT does not change the norm (2.12),
$$\|\widehat{A}\| = \max_{\|f\|=1}\|Af\| = \|A\|\,.$$
□

Another important property of an operator is its spectrum. Recall that the *resolvent* set ρ and *spectrum* σ of an operator are

$$\rho(A) := \{\lambda \in \mathbb{C} : (A - \lambda I) \text{ is bijective}\} \quad \text{and} \quad \sigma(A) := \mathbb{C} \setminus \rho(A)\,. \tag{2.15}$$

Let us show that the spectrum of the Fourier representation and the original operator are identical.

Proposition 2.17. *Let $\widehat{A}$ be the Fourier representation of A. Then $\sigma(\widehat{A}) = \sigma(A)$.*

Proof. Since $\mathcal{F}_{\mathbf{h}}$ is bijective,

$$(A - \lambda I \quad \text{is bijective}) \quad \text{if and only if} \quad (\mathcal{F}_{\mathbf{h}}(A - \lambda I)\mathcal{F}_{\mathbf{h}}^{-1} \quad \text{is bijective})\,.$$

Furthermore, by Theorem 2.15,

$$\mathcal{F}_{\mathbf{h}}(A - \lambda I)\mathcal{F}_{\mathbf{h}}^{-1} = \widehat{A} - \lambda I\,.$$

Thus, by definition (2.15), $\rho(\widehat{A}) = \rho(A)$ and therefore $\sigma(\widehat{A}) = \sigma(A)$. □

2.2.3. Constant Stencils

If A is a constant stencil operator, then the wave functions introduced in Section 1.5.3 are the formal eigenfunctions of the operator. The function values of the Fourier symbol $\hat{a}$ of A are the corresponding eigenvalues. Hence, if we know the Fourier symbol $\hat{a}$, we know how the operator A maps any linear combination of eigenfunctions. We shall now show that we know how A maps *any grid function from* $\ell_2(G_{\mathbf{h}})$, if we know $\hat{a}$. For this purpose we have to define the space $L_\infty(\Theta_{\mathbf{h}})$.

The space $L_\infty(\Theta_{\mathbf{h}})$ is the set of functions[2] $\hat{a} : \Theta_{\mathbf{h}} \to \mathbb{C}$ that are bounded in the infinity-norm $\|\cdot\|_\infty$, where

$$\|\hat{a}\|_\infty := \inf\{C \geq 0 : \mu(|\hat{a}| > C) = 0\} \quad \text{and}$$

μ the Lebesgue measure, i.e.,

$$L_\infty(\Theta_{\mathbf{h}}) = \{\hat{a} : \Theta_{\mathbf{h}} \to \mathbb{C} : \|\hat{a}\|_\infty < \infty\}\,.$$

Theorem 2.18. *Let $A \in L(\ell_2(G_{\mathbf{h}}))$ be given by the constant stencil s and let $\hat{a} \in L_\infty(\Theta_{\mathbf{h}})$ be given by*

$$\hat{a}(\theta) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{y}\rangle}\,. \tag{1.55 revisited}$$

Then the Fourier representation is the multiplication with the function $\hat{a}$, i.e.,

$$\widehat{A}\hat{f} = \hat{a} \cdot \hat{f}\,. \tag{2.16}$$

[2] With "function" we actually mean the equivalence class of functions that are equal almost everywhere.

The function $\hat{a}$ in (1.55) is called the *Fourier symbol* of A as already defined in Section 1.5.3. The action of the Fourier representation $\widehat{A}$ is given by the pointwise multiplication of the function $\hat{a}$ with the operator argument. An operator of the form of (2.16) is called *multiplication operator*. Theorem 2.18 is a remarkable result for the following reasons.

First, it means that $\widehat{A}$ is completely described by the Fourier symbol $\hat{a}$, and as A can be constructed from $\widehat{A}$ so is A. Second, in general we would expect that the function value $[\widehat{A}\hat{f}](\theta)$ depends on all function values $\hat{f}(\theta')$ where $\theta \in \Theta_{\mathbf{h}}$. In the case of a constant stencil operator, however, the resulting function value $[\widehat{A}\hat{f}](\theta)$ depends only on the function value $\hat{f}(\theta)$, which has an important consequence.

Assume that $a(\theta)$ is substantially smaller than 1 at high frequencies θ. That means that repeated application of A makes $\hat{f}(\theta)$ small for high frequencies θ, meaning that, due to Theorem 2.13, the function f becomes slowly varying. Hence, in this case A is a smoothing operator. We can therefore use the symbol $\hat{a}$ to analyze the smoothing properties of constant stencil operator not only for wave functions but for general functions $f \in \ell_2(G_{\mathbf{h}})$.

To prove Theorem 2.18 we first consider the Fourier representation of the shift operator $S_{\mathbf{z}}$ (2.6).

Lemma 2.19. *Let $S_{\mathbf{z}}$ be the shift operator* (2.6)*. It has a Fourier representation;*

$$\widehat{S_{\mathbf{z}}}\hat{f} = \left\{\theta \mapsto e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \cdot \hat{f}(\theta)\right\}. \tag{2.17}$$

Thus, the Fourier symbol $\hat{s}$ of the shift operator is

$$\hat{s}(\theta) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle}.$$

Proof. Recall from Lemma 2.12 that

$$[\mathcal{F}_{\mathbf{h}} S_{\mathbf{z}} f](\theta) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \cdot [\mathcal{F}_{\mathbf{h}} f](\theta). \tag{2.13 revisited}$$

Substitute f by $\mathcal{F}_{\mathbf{h}}^{-1}\hat{f}$ to get

$$[\mathcal{F}_{\mathbf{h}} S_{\mathbf{z}} \mathcal{F}_{\mathbf{h}}^{-1}\hat{f}](\theta) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \cdot [\mathcal{F}_{\mathbf{h}}\mathcal{F}_{\mathbf{h}}^{-1}\hat{f}](\theta) = e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \cdot \hat{f}(\theta). \qquad \square$$

Proof of Theorem 2.18. Recall the definition of a constant stencil operator,

$$[Au](\mathbf{x}) := \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(\mathbf{y}) \cdot u(\mathbf{x}+\mathbf{y}). \tag{1.52 revisited}$$

Using the shift operator we can write this equation as

$$A = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(\mathbf{y}) \cdot S_{-\mathbf{y}}.$$

The mapping that maps an operator to its Fourier representation is linear. Thus, it follows

$$\widehat{A} = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(\mathbf{y}) \cdot \widehat{S_{-\mathbf{y}}}.$$

We know the Fourier representation of $S_{-\mathbf{y}}$ from Lemma 2.19, implying that

$$\left[\widehat{A\hat{f}}\,\right](\theta) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(\mathbf{y}) \cdot \left[\widehat{S_{-\mathbf{y}}\hat{f}}\right](\theta) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} \cdot \hat{f}(\theta) = \hat{a}(\theta) \cdot \hat{f}(\theta)\,.$$

It remains to show that $\|a\|_\infty < \infty$, which we prove by contradiction. Assume that A is a bounded operator and $\|a\|_\infty$ is unbounded, implying that for every $C > 0$ there exists a measurable set $M \subseteq \mathbb{R}^d$ such that $a(x) \geq C$ for all $x \in M$. Let $\hat{f} := \chi_M / \|\chi_M\|_2$, where χ_M is the *characteristic function* for the set M, i.e.,

$$\chi_M(x) = \begin{cases} 1 & \text{if } x \in M, \\ 0 & \text{if } x \notin M. \end{cases}$$

With this definition

$$\|\hat{A}\hat{f}\|_2 = \|a \cdot \hat{f}\|_2 = \left(\int_{\Theta_{\mathbf{h}}} |a \cdot \hat{f}|^2 \,\mathrm{d}\theta\right)^{1/2} \geq C \left(\int_{\Theta_{\mathbf{h}}} |\hat{f}|^2 \,\mathrm{d}\theta\right)^{1/2} = C\,.$$

Thus, for an arbitrarily large $C > 0$ we find a function $\hat{f}$ with norm one such that $\|\widehat{A}\hat{f}\| \geq C$, implying that $\|\widehat{A}\| = \infty$ and therefore $\|A\| = \infty$, which contradicts the assumption. □

2.3. Multiplication Operators

The Fourier representation of a constant stencil operator is a multiplication operator—as we saw in Section 2.2.2. Let us consider multiplication operators in general. In the following we shall write L_2 for $L_2(\Theta_{\mathbf{h}})$ and L_∞ for $L_\infty(\Theta_{\mathbf{h}})$ in unambiguous situations.

Definition 2.20. Let $a \in L_\infty$ and $f \in L_2$. The operator $A \in L(L_2)$ given by

$$Af = a \cdot f$$

is called a *multiplication operator*. The function a is called the *symbol* of A.

Proposition 2.21. *The multiplication operator is well defined.*

Proof. We need to show that $A \in L(L_2)$. For the pointwise product $a \cdot f$ we have

$$\|a \cdot f\|_2 \leq \|a\|_\infty \cdot \|f\|_2\,.$$ □

2.3.1. Properties of Multiplication Operators

We note a few properties of multiplication operators. First of all, there is a one-to-one correspondence between multiplication operators and their symbols.

Proposition 2.22. *If A is a multiplication operator, then its symbol is unique.*

Proof. Assume that $Af = a \cdot f$ and $Af = a' \cdot f$; implying $a \cdot f = a' \cdot f$. With $f \equiv 1$ we have $a = a'$ (a.e.). □

Furthermore, many operations on symbols directly translate into operations of the multiplication operators and vice versa.

Theorem 2.23. *Let A, B be two multiplication operators with symbols $a, b \in L_\infty$, and let $\lambda \in \mathbb{C}$. Then*

1. *1 is the symbol of the identity I,*
2. *$\lambda \cdot a + b$ is the symbol of $\lambda A + B$,*
3. *$a \cdot b$ is the symbol of AB, and*
4. *$\overline{a}$ is the symbol of A^*.*

Proof. Let us prove the statements individually.

1. The assertion is trivial.
2. Consider the following:
$$(\lambda A + B)u = \lambda Au + Bu = \lambda \cdot a \cdot u + b \cdot u = (\lambda \cdot a + b) \cdot u \,.$$
3. We have
$$(AB)u = A(Bu) = A(bu) = a(bu) = (ab)u \,.$$
4. The adjoint is the unique operator A^* that fulfills the equation
$$\langle Af, g\rangle = \langle f, A^* g\rangle \quad \text{for all} \quad f, g \in L_2 \,.$$
Thus
$$\langle a \cdot f, g\rangle = \int \overline{g} \cdot (a \cdot f)\,\mathrm{d}\mathbf{x} = \int (\overline{\overline{a} \cdot g}) \cdot f\,\mathrm{d}\mathbf{x} = \langle f, \overline{a} \cdot g\rangle \,,$$
proving the assertion.

□

An interesting consequence of Theorem 2.23 is that all multiplication operators commute.

Proposition 2.24. *Let A, B be two multiplication operators then*
$$AB = BA \,.$$

Proof. The symbol of AB is $(a \cdot b)$; the symbol of BA is $(b \cdot a)$. Furthermore $a \cdot b = b \cdot a$. □

Corollary 2.25. *Every multiplication operator A is normal, i.e., $A^*A = AA^*$.*

That two multiplication operators commute has another consequence.

Proposition 2.26. *Let $A \in L(\ell_2(G_\mathbf{h}))$ have a Fourier symbol $\widehat{A}$. Then A is shift invariant.*

Proof. To show that A is shift invariant, we need to show that
$$S_\mathbf{z} A = A S_\mathbf{z}$$
for arbitrary $\mathbf{z} G_\mathbf{h}$. Since the DTFT is bijective, we can equivalently show that the Fourier representation of the left equals the Fourier representation of the right hand side, i.e.,
$$\mathcal{F}_\mathbf{h} S_\mathbf{z} A \mathcal{F}_\mathbf{h}^{-1} = \mathcal{F}_\mathbf{h} A S_\mathbf{z} \mathcal{F}_\mathbf{h}^{-1} \,.$$
By using Theorem 2.15 this equation is equivalent to
$$\widehat{S_\mathbf{z}}\,\widehat{A} = \widehat{A}\,\widehat{S_\mathbf{z}} \,.$$
The assertion then follows from Proposition 2.24. □

2.3.2. The Spectrum of Multiplication Operators

We now turn to the computation of the spectra and spectral radii of multiplication operators, which will allows us to obtain the spectral radius of Fourier symbols. The spectrum of an operator is equal to the spectrum of its Fourier symbol. Hence, we can use this result to obtain the spectrum of constant Stencil operators.

The spectrum of an operator is closely linked to the question whether an operator is invertible or not. Recall,

$$\rho(A) := \{\lambda \in \mathbb{C} : (A - \lambda I) \text{ is bijective}\} \quad \text{and} \quad \sigma(A) := \mathbb{C} \setminus \rho(A)\,. \qquad \text{(2.15 revisited)}$$

We shall define the inverse of a symbol and show that a symbol is invertible if and only if the corresponding multiplication operator is invertible. As the symbol of $(A - \lambda I)$ is $(a - \lambda)$, if $(a - \lambda)$ is not invertible then λ is in the spectrum of A.

We start with the definition of the inverse of a symbol a.

Definition 2.27. Let $a \in L_\infty$. The symbol a is invertible if and only if

1. the pointwise inverse a^{-1} of a exists almost everywhere, and
2. the pointwise inverse a^{-1} is bounded, i.e., $a^{-1} \in L_\infty$.

The following lemma provides a simple way to check if a symbol is invertible.

Lemma 2.28. *The symbol $a \in L_\infty$ is invertible if and only if*

$$\textit{there exists } \epsilon > 0 \quad \textit{such that} \quad \mu(|a| < \epsilon) = 0.$$

Proof. The proof consists of two parts.

1. Assume that there exists $\epsilon > 0$ such that $\mu(|a| < \epsilon) = 0$, which implies that $a \neq 0$ almost everywhere. Thus, a^{-1} exists. Furthermore $\mu(|a^{-1}| > \epsilon^{-1}) = 0$. Hence, by the definition of the infinity norm $\|a^{-1}\|_\infty \leq \epsilon^{-1}$.
2. Conversely, assume that for all $\epsilon > 0$ we have $\mu(|a| < \epsilon) \neq 0$. We have to consider two cases.
   a) If $\mu(a = 0) \neq 0$ then a^{-1} does not exist, and therefore a is not invertible.
   b) If $a \neq 0$ almost everywhere, the pointwise inverse a^{-1} exists. For $\epsilon > 0$ we have $0 \neq \mu(|a| < \epsilon) = \mu(|a^{-1}| > \epsilon^{-1})$. Thus, by definition $\|a^{-1}\|_\infty \geq \epsilon^{-1}$. As the inequality holds for a arbitrary $\epsilon > 0$ we have $\|a^{-1}\|_\infty = \infty$ and thus a is not invertible. □

The following Lemma is a first step in proving the relation between the inverse of a multiplication operator and its symbol.

Lemma 2.29. *Let A be an invertible multiplication operator with symbol a. Then $a \neq 0$ almost everywhere, i.e., a^{-1} exists.*

Proof. We prove the assertion by contradiction. Assume that $\mu(a = 0) \neq 0$. Define set $M := \{x : a(x) = 0\}$ and the function $u = \chi_M$. Then $\chi_M \neq 0$. However,

$$A\chi_M = a \cdot \chi_M = 0,$$

which implies that the kernel of A contains more functions than just the zero function. Therefore, A is not injective, which is a contradiction to the assumption that A is invertible. □

We prove the relation between the inverse of the multiplication operator and its symbol.

Theorem 2.30. *Let A be a multiplication operator with symbol a. Then A is invertible if and only if its symbol a is invertible. If the inverse of A exists, it is a multiplication operator with symbol a^{-1}.*

Proof. We prove the two implications individually.

1. Assume that A is invertible. Let $u \in L_2$ be arbitrary. Then

   $$a \cdot u = [Au]$$

   By Lemma 2.29, the inverse of a exists, and we have that

   $$u = a^{-1} \cdot [Au] \,.$$

   Consequently, the inverse of A can be written as

   $$A^{-1}u = a^{-1} \cdot u$$

   meaning that A^{-1} is a multiplication operator and a^{-1} is its symbol.
   It remains to show that $a^{-1} \in L_\infty$. We assume that this is not the case, i.e., for all $c > 0$ we have that $\mu(|a^{-1}| > c) \neq 0$. Let $M(c) := \{x : |a^{-1}(x)| > c\}$. Then

   $$\|A^{-1}\chi_{M(c)}\|^2 = \int_{M(c)} |a^{-1}(x)|^2 \,\mathrm{d}x \geq \int_{M(c)} |c|^2 \,\mathrm{d}x = c \cdot \|\chi_{M(c)}\|.$$

   By the definition of $\chi_{M(c)}$ we have $\|\chi_{M(c)}\|_2 > 0$ and thus the above inequality implies

   $$\frac{\|A^{-1}\chi_{M(c)}\|^2}{\|\chi_{M(c)}\|} \geq c \,.$$

   Therefore, by the definition of the operator norm, we have for all $c > 0$ that $\|A^{-1}\| > c$, i.e., $\|A^{-1}\| = \infty$, which is a contradiction to the assumption that A is invertible, because in this case it is well known that the inverse of A is also bounded. Thus, $a^{-1} \in L_\infty$.
2. Assume that a is invertible, i.e., $a^{-1}a = 1$ and $\|a^{-1}\|_\infty < \infty$. We define

   $$A^{-1}u := a^{-1}u \quad \text{for} \quad u \in L_2 \,.$$

   Thus,

   $$A^{-1}Au = a^{-1} \cdot a \cdot u = u \quad \text{for} \quad u \in L_2 \,.$$

   It remains to show that $A^{-1}u \in L_2$, which follows from the fact that $\|A^{-1}u\|_2 = \|a^{-1}u\|_2 \leq \|a^{-1}\|_\infty \cdot \|u\|_2$ and that a^{-1} is bounded. □

Combining Lemma 2.28 and Theorem 2.30 yields a characterization of invertible multiplication operators.

Corollary 2.31. *The multiplication operator A with symbol a is bijective if and only if there exists $\epsilon > 0$ such that*

$$\mu(|a| < \epsilon) = 0 \,.$$

With this knowledge we can compute the spectrum of multiplication operators. If A is a multiplication operator then $A - \lambda I$ is a multiplication operator with symbol $(a - \lambda)$. Thus, Corollary 2.31 yields that

$$\rho(A) = \{\lambda \in \mathbb{C} : \mu(|a - \lambda| < \epsilon) = 0 \text{ for any } \epsilon > 0\} \quad \text{and}$$
$$\sigma(A) = \{\lambda \in \mathbb{C} : \mu(|a - \lambda| < \epsilon) \neq 0 \text{ for all } \epsilon > 0\}\,.$$

The spectrum $\sigma(A)$ is a special set; it is the *essential range* of a. We denote it by

$$\text{ess-im}(a) := \{\lambda \in \mathbb{C} : \mu(|a - \lambda| < \epsilon) \neq 0 \text{ for all } \epsilon > 0\}.$$

Thus, for a multiplication operator A the spectrum coincides with the essential range of its symbol a, i.e.,

$$\sigma(A) = \text{ess-im}(a)\,. \tag{2.18}$$

The following lemma motivates the name.

Lemma 2.32. *Let $a \in L_\infty$. Then*

$$\text{ess-im}(a) = \bigcap_{\tilde{a} \in [a]} \overline{\text{im}(\tilde{a})},$$

where $[a]$ is the equivalence class of all functions that are equal almost everywhere.

Proof. We show the assertion in two steps. First, we show that the left hand side is a subset of the right hand side. Then we show the converse statement.

1. Let $\lambda \notin \text{ess-im}(a)$. Then there exists $\epsilon > 0$ such that $\mu(|a - \lambda| < \epsilon) = 0$. We can alter a at a set of zero measure to obtain $\tilde{a} \in [a]$ with $\{x : |\tilde{a}(x) - \lambda| < \epsilon\} = \emptyset$. Thus, $\lambda \notin \overline{\text{im}(a)}$ and therefore
$$\lambda \notin \bigcap_{\tilde{a} \in [a]} \overline{\text{im}(\tilde{a})}\,. \qquad \square$$

2. Let $U_\epsilon(\lambda) := \{\tilde{\lambda} \in \mathbb{C} : |\tilde{\lambda} - \lambda| < \epsilon\}$, and let $\lambda \notin \bigcap_{\tilde{a} \in [a]} \overline{\text{im}(\tilde{a})}$. Then there exists $\tilde{a}$ such that $\lambda \notin \overline{\text{im}(\tilde{a})}$, and thus as $\overline{\text{im}(\tilde{a})}$ is closed, there exists $\epsilon > 0$ such that for all $\tilde{\lambda} \in U_\epsilon(\lambda)$, we have that $\tilde{\lambda} \notin \overline{\text{im}(\tilde{a})}$. Hence, $\{x : |\tilde{a} - \lambda| < \epsilon\} = \emptyset$, and
$$0 = \mu(|\tilde{a} - \lambda| < \epsilon) = \mu(|a - \lambda| < \epsilon),$$
which implies that $\lambda \notin \text{ess-im}(a)$.

We can relate the essential range of a symbol a to its L_∞-norm.

Proposition 2.33. *We have that*

$$\|a\|_\infty = \sup(\text{ess-im}(a))\,.$$

We shall prove this proposition below. Let us first note, however, a few consequences of this proposition. First, it gives us an easy way to compute the spectral radius of the multiplication operator A.

Corollary 2.34. *The spectral radius $r(A)$ of A fulfills $r(A) = \|a\|_\infty$.*

As stated in Corollary 2.25, every multiplication operator is normal. For a normal operator A, we have $r(A) = \|A\|$. Thus, we also get the operator norm of A by using the infinity norm.

Corollary 2.35. *Let A be a multiplication operator with symbol a. Then the norm of A fulfills $\|A\| = \|a\|_\infty$.*

We now turn to the proof of Proposition 2.33. First, we need the following two auxiliary results.

Lemma 2.36. *If $M \subseteq \mathbb{C}$ is a compact, measurable set, $f : \mathbb{R}^d \to \mathbb{C}$ a measurable function and $\mu(f^{-1}(M)) \neq 0$. Then there exists for every $\epsilon > 0$ a complex number $z \in M$ such that $\mu(f^{-1}(U_\epsilon(z))) \neq 0$.*

Proof. Let $\epsilon > 0$ be given. As the set M is compact, there exist $z_1, \ldots, z_n \in M$ such that $M \subseteq \bigcup_{i=1}^n U_\epsilon(z_i)$. Thus, $f^{-1}(M) \subseteq \bigcup_{i=1}^n f^{-1}(U_\epsilon(z_i))$, and we have

$$0 < \mu(f^{-1}(M)) \leq \sum_{i=1}^{n} \mu(f^{-1}(U_\epsilon(z_i)) .$$

Therefore, for at least one index i, we have $0 < \mu(f^{-1}(U_\epsilon(z_i))$, implying that $z := z_i \in M$ is the sought-for number. □

Lemma 2.37. *If $M \subseteq \mathbb{C}$ is a compact, measurable set, $f : \mathbb{R}^d \to \mathbb{C}$ a measurable function and $\mu(f^{-1}(M)) \neq 0$. Then there exists a number $z \in M$ such that for every $\epsilon > 0$ we have $\mu(f^{-1}(U_\epsilon(z))) \neq 0$.*

Proof. According to Lemma 2.36, there exists $z_1 \in M$ such that $\mu(f^{-1}(U_1(z_1))) > 0$. We construct a sequence recursively as follows. Let $n > 1$. According to Lemma 2.36 there exists $z_{n+1} \in \overline{U_{2^{-n}}(z_n)}$ such that $\mu(f^{-1}(U_{2^{-(n+1)}}(z_{n+1}))) > 0$.

For $n \geq m$ we have $z_n - z_m = \sum_{i=n}^{m-1} z_i - z_{i+1}$. Thus

$$|z_n - z_m| \leq \sum_{i=n}^{m-1} |z_i - z_{i+1}| \leq \sum_{i=n}^{m-1} 2^{-i} \leq \sum_{i=n}^{\infty} 2^{-i} = 2^{1-n} .$$

Therefore, z_n is a Cauchy sequence, and thus converges to a limit z, which has the desired properties. □

Proof of Proposition 2.33. Let $c := \|a\|_\infty := \inf\{b \in \mathbb{R} : \mu(|a| > b) = 0\}$. If $\lambda > c$ then for $\epsilon := |\lambda - c|/2$ the equation

$$\{x : |a(x) - \lambda| < \epsilon\} \subseteq \{x : |a(x)| > c\}$$

holds. By the definition of c, we have that $\mu(|a| > c) = 0$ and $\mu(|a - \lambda| < \epsilon) = 0$. Thus, $\lambda \notin \text{ess-im}(a)$ and therefore $\sup(\text{ess-im}(a)) \leq c$.

On the other hand, if $0 \leq \bar{c} < c$ then $\mu(|a| > \bar{c}) \neq 0$. Hence, the fact that $\mu(|a| > c) = 0$ implies that $\mu(\bar{c} \leq |a| \leq c) \neq 0$. Thus, due to Lemma 2.37, there exists $\lambda \in \mathbb{C}$ with $\bar{c} \leq |\lambda| \leq c$ such that for all $\epsilon > 0$ we have $\mu(|a - \lambda| < \epsilon) \neq 0$ implying $\lambda \in \text{ess-im}(a)$. Hence, for every $0 < \bar{c}$ with $\bar{c} < c$, we find λ with $\bar{c} \leq |\lambda| \leq c$, which implies $\sup(\text{ess-im}(a)) \geq c$. □

2.4. Operators with Fourier Symbols

An operator A has a *Fourier symbol*, if its Fourier representation $\widehat{A}$ can be written in the form

$$\widehat{A}\hat{f} = \hat{a} \cdot \hat{f}\,, \tag{2.16 revisited}$$

where $\hat{a}$ is the Fourier symbol of A. Such an operator can be characterized as follows.

Theorem 2.38. *Let $A \in L(\ell_2(G_\mathbf{h}))$. The following statements are equivalent.*

1. *The operator A is represented by a constant stencil.*
2. *The operator A is shift invariant.*
3. *The operator A has a Fourier symbol.*

Proof. Proposition 2.26 shows that assertion 3 implies 2. Theorem 2.9 shows that assertion 2 implies 1. Theorem 2.18 shows that assertion 1 implies 3. □

The operator norm and spectral radius of the operator A can be easily obtained from the Fourier symbol of A.

Theorem 2.39. *Let A be a constant stencil operator with Fourier symbol $\hat{a}$. Then*

$$r(A) = \|A\| = \|\hat{a}\|_\infty. \tag{2.19}$$

Proof. Proposition 2.17 states that $\sigma(A) = \sigma(\widehat{A})$, where $\widehat{A}$ is the Fourier representation of A. Thus, to complete the proof it is sufficient to show that $r(\widehat{A}) = \|\widehat{A}\| = \|\hat{a}\|_\infty$. As A is a constant stencil operator, $\widehat{A}$ is a multiplication operator with symbol $\hat{a}$. Hence, we have by Corollary 2.34 that $r(\widehat{A}) = \|\hat{a}\|_\infty$ and by Corollary 2.35 that $\|\widehat{A}\| = \|\hat{a}\|_\infty$. □

2.5. Smoothing Factor

In Section 1.5.6 we defined the smoothing factor by

$$\mathrm{smf}(S, \mathbf{s}) := \max_{\theta \in \Theta_\mathbf{h}} |\hat{s}(\theta)^{\nu_2} \cdot \hat{q}_\mathbf{n}(\theta) \cdot \hat{s}(\theta)^{\nu_1}|\,, \tag{1.63 revisited}$$

where $\hat{s}$ is the Fourier symbol of the error propagator of the smoother S, and $\hat{q}_\mathbf{n}$ is the Fourier symbol of the idealized coarse grid correction $Q_\mathbf{n}$. Using Theorem 2.39 we can justify this definition.

Assume we are interested in either the norm or spectral radius of the error propagator

$$E = S^{\nu_2} Q_\mathbf{n} S^{\nu_1}\,.$$

of the idealized two-grid method, as they give the worst case and the asymptotic convergence rate of the idealized method. Then

$$r(E) = \|E\| = \|\hat{e}\|_\infty = \|\hat{s}^{\nu_2} \hat{q}_\mathbf{n} \hat{s}^{\nu_1}\|_\infty\,.$$

If the symbol $\hat{e}$ is continuous, then the infinity norm equals the maximum function value of $\hat{e}$ and hence it is equal to the smoothing factor, as defined above.

As we cannot guarantee that the symbol $\hat{e}$ is continuous, from now on, we define the smoothing factor, as

$$\mathrm{smf}(S, \mathbf{n}) := r(S^{\nu_2} Q_\mathbf{n} S^{\nu_1}) = r(\widehat{S}^{\nu_2} \widehat{Q}_\mathbf{n} \widehat{S}^{\nu_1}) = \|\hat{s}^{\nu_2} \cdot \hat{q}_\mathbf{n} \cdot \hat{s}^{\nu_1}\|_\infty\,. \tag{2.20}$$

2.6. Literature and Contributions

This chapter is principally a summary of known results from different mathematical subjects. The foundation of this chapter can be found in [11], which connects the analysis of operators in terms of wave functions to the DTFT, to compute norms of stencil operators.

The Fourier representation of constant stencil operators leads to multiplication operators, which are well understood. The fact that the essential range of a multiplication operator coincides with its spectrum can be found, e.g., in [57].

The connection between multiplication operators and shift invariant operators via Fourier transforms is a result from linear systems theory [5]. Furthermore, the representation of linear operators on sequence spaces is, e.g., discussed in [42].

Theorem 2.13, which measures the amount of oscillation of a function in terms of its DTFT, is closely related to the definition of norms in discrete Sobolev spaces in [31, 64].

3. Matrix Symbols

Fourier symbols represent constant stencil operators, but they have their limitations. For example, we cannot describe the two-grid method by using Fourier symbols, which is why we introduced the smoothing analysis in Section 2.5. Smoothing analysis is idealized, as it uses an idealized coarse grid correction. Using *matrix symbols*, however, it is possible to represent the coarse grid correction exactly.

While symbols are functions that map a frequency to a scalar, matrix symbols are matrices of symbols. Hence, they give a greater freedom for representing operators. We shall use this freedom to represent restriction and interpolation operators. Furthermore, we show that *periodic stencil operators* can be represented by matrix symbols.

Periodic stencil operators have many applications. In this chapter we shall use them to analyze multicoloring smoothers. In Chapter 4 we give more applications of local Fourier analysis using matrix symbols and periodic stencils.

We start by deriving the Fourier representation of restriction and interpolation operators, which will allows us to show how they can be represented using matrix symbols.

3.1. The Two-Grid Method

The smoothing factor is an analysis of an idealized two-grid method. It is idealized even if we assume that the problem was posed on an infinite grid. We shall introduce a two-grid analysis that is exact for problems posed on infinite grids.

3.1.1. Restriction

We consider the restriction of a grid function defined on the grid $G_{\mathbf{h}}$ to a grid $G_{\mathbf{n}\cdot\mathbf{h}}$, for some $1 \leq \mathbf{n} \in \mathbb{N}^d$. In Section 1.5.6 we considered a similar situation—the restriction of *wave functions*. When wave functions are restricted, there are different wave functions that are different on the fine grid, but identical on the coarse grid. We shall see that a grid function from $\ell_2(G_{\mathbf{h}})$ behaves in a similar way.

To do so, we consider the injection restriction operator $R_{\text{inj}} : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{n}\cdot\mathbf{h}})$ given by

$$[R_{\text{inj}}\, u](x) = u(x)\,. \tag{3.1}$$

Furthermore, recall that the set of harmonic frequencies can be written as

$$[\theta] = \{\theta_b + \mathbf{j}\cdot(2\pi)/(\mathbf{n}\cdot\mathbf{h}) : 0 \leq \mathbf{j} < \mathbf{n}, \mathbf{j} \in \mathbb{Z}^d\}, \tag{1.60 revisited}$$

where $\theta_b \in \Theta_{\mathbf{n}\cdot\mathbf{h}}$ is the base frequency of θ. Thus, if we define the $\mathbf{n},\mathbf{h}$-*frequency shifts*,

$$\mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}} := 2\pi\frac{\mathbf{j}}{\mathbf{h}\cdot\mathbf{n}} \quad \text{for} \quad \mathbf{j} = \mathbf{0},\ldots,\mathbf{n}-\mathbf{1}\,, \tag{3.2}$$

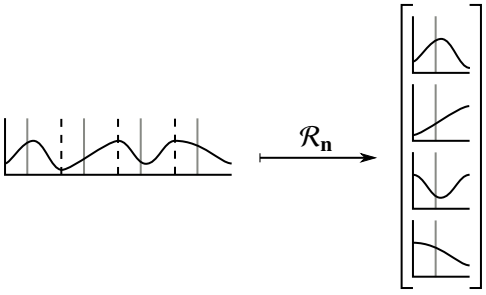


Figure 3.1.: The action of the frequency splitting operator $\mathcal{R}_{\mathbf{n}}$ for a 1D function and $\mathbf{n} = (4)$. The gray vertical bars mark one set of $\mathbf{n}$,$\mathbf{h}$-harmonics.

we can write

$$[\theta] = \{\theta_b + \mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}} : \mathbf{j} = \mathbf{0}, \dots, \mathbf{n} - \mathbf{1}\}.$$

Writing $[\theta]$ in this way has the advantage that we can write every frequency uniquely as the sum of its base frequency and a frequency shift. In the following we will often write in unambiguous situations $\mathbf{s}_{\mathbf{j}}$ for $\mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}}$.

To denote the Fourier representation of the injection restriction operator, we need the *frequency splitting operator*. This operator groups the function values of frequencies with respect to their frequency shift.

Definition 3.1. Let $\mathcal{R}_{\mathbf{n}} : L_2(\Theta_{\mathbf{h}}) \to L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})^{\mathbf{n}}$ the operator given by

$$[\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{j}}(\theta) := \hat{u}(\theta + \mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}}), \tag{3.3}$$

where $L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})^{\mathbf{n}}$ is the $\mathbf{n}$-dimensional space of vectors with entries in $L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})$. The operator $\mathcal{R}_{\mathbf{n}}$ is called the *frequency splitting operator*.

The frequency splitting operator returns a vector of functions, whose $\mathbf{j}$th entry corresponds to the function values of the frequencies of the frequency shift $\mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}}$. Note that if we evaluate all entries of the vector $\mathcal{R}_{\mathbf{n}}\hat{u}$ at the same frequency $\theta \in \Theta_{\mathbf{n}\cdot\mathbf{h}}$, we obtain the function values of $\hat{u}$ at all $\mathbf{n}$,$\mathbf{h}$-harmonics of θ. The action of $\mathcal{R}_{\mathbf{n}}$ is illustrated in Figure 3.1. Its definition is motivated by the following theorem.

Theorem 3.2. *The Fourier representation of the injection restriction R_{inj} is given by*

$$\widehat{R}_{\mathrm{inj}}\hat{u} = \frac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} [\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{j}} \,. \tag{3.4}$$

Proof. Let $\hat{u} := \mathcal{F}_{\mathbf{h}}u$. For $\mathbf{x} \in G_{\mathbf{h}}$ we have by the inverse of the DTFT (2.10) that

$$u(\mathbf{x}) = \frac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_h} \hat{u}(\vartheta)\, e^{\mathrm{i}\langle\vartheta,\mathbf{x}\rangle}\, \mathrm{d}\vartheta \,.$$

Thus, by definition of R_{inj} for $\mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}$,

$$[R_{\text{inj}}u](\mathbf{x}) = \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_h} \hat{u}(\vartheta)\, e^{\mathrm{i}\langle\vartheta,\mathbf{x}\rangle}\, \mathrm{d}\vartheta\,.$$

Let $\Phi_{\mathbf{j}} := \mathbf{s}_{\mathbf{j}} + \Theta_{\mathbf{h}\cdot\mathbf{n}}$ for $\mathbf{j} = \mathbf{0}, \ldots, \mathbf{n}-\mathbf{1}$. This family of sets is a measurable partition of $\Theta_{\mathbf{h}}$, i.e., $\Theta_{\mathbf{h}} = \bigcup_{\mathbf{j}=\mathbf{1}}^{\mathbf{n}} \Phi_{\mathbf{j}}$ and $\Phi_{\mathbf{j}}$ is measurable. It follows that

$$[R_{\text{inj}}u](\mathbf{x}) = \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Phi_j} \hat{u}(\vartheta)\, e^{\mathrm{i}\langle\vartheta,\mathbf{x}\rangle}\, \mathrm{d}\vartheta$$

and a variable substitution leads to

$$[R_{\text{inj}}u](\mathbf{x}) = \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{h}\cdot\mathbf{n}}} \hat{u}(\vartheta + \mathbf{s}_{\mathbf{j}})\, e^{\mathrm{i}\langle\vartheta+\mathbf{s}_{\mathbf{j}},\mathbf{x}\rangle}\, \mathrm{d}\vartheta\,.$$

Note that the frequencies $\theta + \mathbf{s}_{\mathbf{j}}$ are harmonic frequencies of each other. Thus, for $\mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}$ we have by Theorem 1.21 that $e^{\mathrm{i}\langle\vartheta+\mathbf{s}_{\mathbf{j}},\mathbf{x}\rangle} = e^{\mathrm{i}\langle\vartheta+\mathbf{s}_{\mathbf{0}},\mathbf{x}\rangle} = e^{\mathrm{i}\langle\vartheta+\mathbf{0},\mathbf{x}\rangle}$ for $\mathbf{j} = \mathbf{1}, \ldots, \mathbf{n}$. Therefore,

$$\begin{aligned}
[R_{\text{inj}}u](\mathbf{x}) &= \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{h}\cdot\mathbf{n}}} \hat{u}(\vartheta + \mathbf{s}_{\mathbf{j}})\, e^{\mathrm{i}\langle\vartheta,x\rangle}\, \mathrm{d}\vartheta \\
&= \frac{\text{vol}_{\mathbf{h}\cdot\mathbf{n}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_{\mathbf{h}\cdot\mathbf{n}}} \left(\frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} [\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{j}}(\vartheta) \right) \cdot e^{\mathrm{i}\langle\vartheta,x\rangle}\, \mathrm{d}\vartheta
\end{aligned}$$

The term in parenthesis is the operator $\widehat{R}_{\text{inj}}$ as defined by (3.4). Using the formula of the inverse DTFT (2.10), we see that

$$\begin{aligned}
[R_{\text{inj}}u](\mathbf{x}) &= \big(\mathcal{F}_{\mathbf{h}\cdot\mathbf{n}}^{-1} \widehat{R}_{\text{inj}} \hat{u}\big)(\mathbf{x}) \\
&= \big(\mathcal{F}_{\mathbf{h}\cdot\mathbf{n}}^{-1} \widehat{R}_{\text{inj}} \mathcal{F}_{\mathbf{h}} u\big)(\mathbf{x})\,.
\end{aligned}$$

Hence, the operator $\widehat{R}_{\text{inj}}$ defined by (3.4) is indeed the Fourier representation of R_{inj}. □

Theorem 3.2 is a generalization of the statement that wave functions corresponding to harmonic frequencies are mapped to the same wave function on the coarse grid. We can realize this fact by considering a linear combination of *wave functions*. The frequencies of these wave functions are chosen to be the $\mathbf{n}$,$\mathbf{h}$-harmonics of a base frequency θ_b. We write this linear combination as

$$f(\mathbf{x}) := \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{j}} \cdot \phi_{\theta_b+\mathbf{s}_{\mathbf{j}}}(\mathbf{x}) \quad \text{where} \quad a_{\mathbf{j}} \in \mathbb{C}\,.$$

Then if $f(\mathbf{x})$ is restricted to the coarse grid $G_{\mathbf{n}\cdot\mathbf{h}}$,

$$f(\mathbf{x}) = \Big(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{j}}\Big) \cdot \phi_{\theta_b}(\mathbf{x}) \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}\,.$$

Thus, the coefficients of the wave functions corresponding to harmonic frequencies merge into one, by being summed. When a grid function $u \in \ell_2(G_{\mathbf{h}})$ is restricted to a coarse grid then the function values at the harmonic frequencies similarly merge to yield the coarse grid Fourier representation of the function, i.e., the DTFT $\hat{u}_{\mathrm{c}}$ of $R_{\mathrm{inj}}u$ is given by

$$\hat{u}_{\mathrm{c}}(\theta_b) = \tfrac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \hat{u}(\theta_b + \mathbf{s}_{\mathbf{j}})\,.$$

Let us summarize a few other properties of the frequency splitting operator. First of all, it is invertible:

$$[\mathcal{R}_{\mathbf{n}}^{-1}\hat{w}](\theta) = \hat{w}_{\mathbf{j}}(\theta_b) \quad \text{where } \mathbf{j} \text{ is determined by } \theta = \theta_b + \mathbf{s}_{\mathbf{j}}.$$

Then if we define the scalar product on $L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})^{\mathbf{n}}$ by

$$\langle f, g\rangle = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{k}-\mathbf{1}} \langle f_{\mathbf{j}}, g_{\mathbf{j}}\rangle\,,$$

the frequency splitting operator is an isometry, as the next proposition shows.

Proposition 3.3. *Let $f, g \in L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})^{\mathbf{n}}$ and $\mathcal{R}$ the frequency splitting operator. Then*

$$\langle \mathcal{R}f, \mathcal{R}g\rangle = \langle f, g\rangle\,.$$

Proof. We use that $\big\{\Theta_{\mathbf{n}\cdot\mathbf{h}} + \mathbf{s}_{\mathbf{j}}\big\}_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}}$ is a partition of $\Theta_{\mathbf{h}}$ to show that

$$\begin{aligned}\int_{\Theta_{\mathbf{h}}} \overline{g}(\theta)\cdot f(\theta)\,\mathrm{d}\theta &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}+\mathbf{s}_{\mathbf{j}}} \overline{g}(\theta)\cdot f(\theta)\,\mathrm{d}\theta = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \overline{g}(\theta+\mathbf{s}_{\mathbf{j}})\cdot f(\theta+\mathbf{s}_{\mathbf{j}})\,\mathrm{d}\theta \\ &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \overline{[\mathcal{R}g]_{\mathbf{j}}}\cdot[\mathcal{R}f]_{\mathbf{j}}\,\mathrm{d}\theta = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \langle \mathcal{R}f, \mathcal{R}g\rangle\,.\end{aligned}$$

□

To this point we only considered the injection restriction operator. Many other restriction operators, however, can be written in the form $R_{\mathrm{inj}}R_{\mathrm{st}}$ where R_{st} is a constant stencil operator. For example, it is easy to see that the weighted restriction operator can be written in this way. In this case the Fourier representation of the restriction operator is $\widehat{R}_{\mathrm{inj}}\widehat{R}_{\mathrm{st}}$.

3.1.2. Interpolation

To define the smoothing factor in Section 2.5 we used an idealized interpolation operator. We shall derive the Fourier representation of interpolation operators, which are actually used in multigrid methods, e.g., the linear interpolation. Using these interpolation operators in our analysis we expect better predictions.

We start by defining the *injection interpolation operator* P_{inj}. Given the injection restriction operator R_{inj} we want to define P_{inj} in such a way that $R_{\text{inj}}P_{\text{inj}} = I$, which is equivalent to the condition that

$$\widehat{R}_{\text{inj}}\widehat{P}_{\text{inj}} = I. \tag{3.5}$$

For the injection restriction we have that

$$\widehat{R}_{\text{inj}}\hat{u} = \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} [\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{j}}, \tag{3.4 revisited}$$

meaning that all harmonic frequencies of the fine grid merge into one frequency on the coarse grid. Thus, if we have a function value at a frequency on the coarse grid, we cannot say to which harmonic frequency on the fine grid it belongs. The idea of the injection interpolation is, to distribute this function value evenly to all harmonic frequencies on the fine grid; we define $P_{\text{inj}}u$ via

$$[\mathcal{R}_{\mathbf{n}}\widehat{P}_{\text{inj}}\hat{u}]_{\mathbf{j}} := \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}}\hat{u}\,. \tag{3.6}$$

Then

$$\widehat{R}_{\text{inj}}\widehat{P}_{\text{inj}}\hat{u} = \tfrac{1}{\text{vol}_n^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} [\mathcal{R}_{\mathbf{n}}\widehat{P}_{\text{inj}}\hat{u}]_{\mathbf{j}} = \tfrac{1}{\text{vol}_n^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \tfrac{1}{\text{vol}_n^{1/2}}\hat{u} = \tfrac{1}{\text{vol}_n}\hat{u} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} 1 = \hat{u}$$

meaning that, indeed, the requirement (3.5) is fulfilled.

The position space representation of the injection interpolation operator will be useful for the definition of general interpolation operators.

Theorem 3.4. *Let P_{inj} be the injection interpolation operator. Then*

$$[P_{\text{inj}}u](\mathbf{x}) = \begin{cases} u(\mathbf{x}) & \text{if } \mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}, \\ 0 & \text{otherwise}. \end{cases}$$

To prove Theorem 3.4 we first need a few auxiliary results.

Lemma 3.5. *If ξ is a n-th root of unity, i.e., $\xi^n = 1$, then*

$$\sum_{i=0}^{n-1} \xi^i = \begin{cases} n & \text{if } \xi = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The formula for the geometric series yields for $\xi \neq 1$ that

$$\sum_{i=0}^{n-1} \xi^i = \frac{1-\xi^n}{1-\xi} = \frac{1-1}{1-\xi} = 0.$$

For $\xi = 1$ we have

$$\sum_{i=0}^{n-1} \xi^i = \sum_{i=0}^{n-1} 1 = n. \qquad \square$$

Lemma 3.6. *Let $k, n \in \mathbb{N}$. We have*

$$\sum_{j=0}^{n-1} e^{2\pi \mathrm{i} \frac{kj}{n}} = \begin{cases} n & \text{if } k \text{ is a multiple of } n, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. We can write

$$\sum_{j=0}^{n-1} e^{2\pi \mathrm{i} \frac{kj}{n}} = \sum_{j=0}^{n-1} \left(e^{2\pi \mathrm{i} \frac{k}{n}} \right)^j .$$

As

$$\left(e^{2\pi \mathrm{i} \frac{k}{n}} \right)^n = e^{2\pi \mathrm{i} k} = 1$$

the term $e^{2\pi \mathrm{i}(k/n)}$ is an nth root of unity. Thus, the assertion follows from Lemma 3.5. $\square$

Lemma 3.7. *Let $\mathbf{k}, \mathbf{n} \in \mathbb{N}^d$. Then*

$$\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{2\pi \mathrm{i} \langle \mathbf{k}/\mathbf{n}, \mathbf{j} \rangle} = \begin{cases} \mathrm{vol}_{\mathbf{n}} & \text{if } \mathbf{k} \text{ is a (pointwise) multiple of } \mathbf{n}, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Assume we have d sequences $a_1, \ldots, a_d$ where $a_\ell \in \mathbb{C}^{n_\ell}$. Then it holds in general that

$$\sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} a_{1 j_1} \cdot a_{2 j_2} \cdots a_{d j_d} = \left(\sum_{j_1=0}^{n_1-1} a_{1 j_1} \right) \cdot \left(\sum_{j_2=0}^{n_2-1} a_{2 j_2} \right) \cdots \left(\sum_{j_d=0}^{n_d-1} a_{d j_d} \right).$$

Consequently,

$$\begin{aligned} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{2\pi \mathrm{i} \langle \mathbf{k}/\mathbf{n}, \mathbf{j} \rangle} &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{2\pi \mathrm{i} \frac{k_1 j_1}{n_1}} \cdot e^{2\pi \mathrm{i} \frac{k_2 j_2}{n_2}} \cdots e^{2\pi \mathrm{i} \frac{k_d j_d}{n_d}} \\ &= \left(\sum_{j_1=0}^{n_1-1} e^{2\pi \mathrm{i} \frac{k_1 j_1}{n_1}} \right) \cdot \left(\sum_{j_2=0}^{n_2-1} e^{2\pi \mathrm{i} \frac{k_2 j_2}{n_2}} \right) \cdots \left(\sum_{j_d=0}^{n_d-1} e^{2\pi \mathrm{i} \frac{k_d j_d}{n_d}} \right), \end{aligned}$$

and the assertion is, therefore, a consequence of Lemma 3.6. $\square$

Proof of Theorem 3.4. The formula for the inverse DTFT (2.10) yields that

$$[P_{\mathrm{inj}} u](\mathbf{x}) = \tfrac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_{\mathbf{h}}} [\widehat{P}_{\mathrm{inj}} \hat{u}](\theta) \cdot e^{\mathrm{i} \langle \theta, \mathbf{x} \rangle} \, \mathrm{d}\mathbf{x} \, .$$

Using the partition $\left\{\Theta_{\mathbf{n}\cdot\mathbf{h}} + \mathbf{s_j}\right\}_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}}$ of $\Theta_{\mathbf{h}}$ and the definition of the injection interpolation operator (3.6) yields that

$$\begin{aligned}
[P_{\text{inj}}u](\mathbf{x}) &= \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \hat{u}(\theta) \cdot e^{\mathrm{i}\langle\theta+s_{\mathbf{j}},\mathbf{x}\rangle}\,\mathrm{d}\theta \\
&= \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \cdot \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \hat{u}(\theta) \cdot e^{\mathrm{i}\langle\theta+s_{\mathbf{j}},\mathbf{x}\rangle}\,\mathrm{d}\theta \\
&= \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \cdot \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \hat{u}(\theta) \cdot e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} \cdot \left(\sum_{\mathbf{j}=\mathbf{1}}^{\mathbf{n}} e^{\mathrm{i}\langle\mathbf{s_j},\mathbf{x}\rangle}\right) \mathrm{d}\theta\,.
\end{aligned}$$

Let us inspect the term in parenthesis. As $\mathbf{x} \in G_{\mathbf{h}}$ we can write $\mathbf{x} = \mathbf{h}\cdot\mathbf{z}$ for some $\mathbf{z} \in \mathbb{Z}^d$. Thus,

$$\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{\mathrm{i}\langle\mathbf{s_j},\mathbf{x}\rangle} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{\mathrm{i}\langle 2\pi\mathbf{j}/(\mathbf{n}\cdot\mathbf{h}),\mathbf{h}\cdot\mathbf{z}\rangle} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} e^{2\pi\mathrm{i}\langle\mathbf{j}/\mathbf{n},\mathbf{z}\rangle}\,.$$

Therefore, from Lemma 3.7, we have that if $\mathbf{x} \in G_{\mathbf{n}\cdot\mathbf{h}}$, then

$$[P_{\text{inj}}u](\mathbf{x}) = \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \cdot \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \hat{u}(\theta) \cdot e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} \cdot \text{vol}_{\mathbf{n}}\,\mathrm{d}\theta = u(\mathbf{x})$$

and if $\mathbf{x} \notin G_{\mathbf{n}\cdot\mathbf{h}}$, then

$$[P_{\text{inj}}u](\mathbf{x}) = \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \cdot \frac{\text{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} \int_{\Theta_{\mathbf{n}\cdot\mathbf{h}}} \hat{u}(\theta) \cdot e^{\mathrm{i}\langle\theta,\mathbf{x}\rangle} \cdot 0\,\mathrm{d}\theta = 0\,. \qquad \square$$

Let us now apply Theorem 3.4 to derive the Fourier representation of the linear interpolation operator for the 1D case. We saw in Section 1.3.4 that the linear interpolation is given by

$$[P_{\text{lin}}u](x) = \begin{cases} u(x) & \text{if } x \in G_{2h} \\ \frac{1}{2}u(x-h) + \frac{1}{2}u(x+h) & \text{if } x \notin G_{2h} \end{cases}. \tag{3.7}$$

Let the stencil s be defined by

$$s := \begin{bmatrix} \frac{1}{2} & \underline{1} & \frac{1}{2} \end{bmatrix}_h$$

and associate the stencil operator P_{st} with s. Then it is easy to see that we can write the linear interpolation as

$$P_{\text{lin}} = P_{\text{st}}P_{\text{inj}}.$$

In other words, applying the linear interpolation operator is equivalent to first applying the injection interpolation and then applying a constant stencil operator.

Many interpolation operators can be written as the product of a constant stencil operator and the injection interpolation. We shall now give a general result, which describes how a certain kind of interpolation operator can be written in this form. For this purpose, we introduce the *space shifts*

$$\mathbf{t_k} := \mathbf{k}\cdot\mathbf{h} \quad \text{for} \quad \mathbf{k} = \mathbf{0},\ldots,\mathbf{n}-\mathbf{1}. \tag{3.8}$$

Every point in $\mathbf{x} \in G_{\mathbf{h}}$ can be written as the sum of a base point $\mathbf{x}_b \in G_{\mathbf{n}\cdot\mathbf{h}}$ and a corresponding space shift, i.e.,

$$\mathbf{x} = \mathbf{x}_b + \mathbf{t}_{\mathbf{k}} \quad \text{for some} \quad \mathbf{k} \in \{\mathbf{0}, \dots, \mathbf{n} - \mathbf{1}\}\,.$$

Many interpolation operators compute the interpolated value at some point by computing a weighted sum of the neighboring coarse grid points. The weights depend on the index $\mathbf{k}$ of the corresponding space shift. The following lemma states that every such interpolation can be written as the product of a constant stencil operator and the injection interpolation.

Theorem 3.8. *Let the interpolation P_{w} (*w *for weighted) be given by*

$$[P_{\mathrm{w}}u](\mathbf{x}_b + \mathbf{t}_{\mathbf{k}}) = \sum_{\mathbf{y} \in G_{\mathbf{n}\cdot\mathbf{h}}} s_{\mathbf{k}}(\mathbf{y}) \cdot u(\mathbf{x}_b + \mathbf{y}), \tag{3.9}$$

where $s_{\mathbf{k}} : G_{\mathbf{n}\cdot\mathbf{h}} \to \mathbb{C}$ for $\mathbf{k} = \mathbf{0}, \dots, \mathbf{n} - \mathbf{1}$. Furthermore, let P_{st} be the stencil operator corresponding to the stencil $s'(\mathbf{x}_b - \mathbf{t}_{\mathbf{k}}) = s_{\mathbf{k}}(\mathbf{x}_b)$. Then

$$P_{\mathrm{w}} = P_{\mathrm{st}} P_{\mathrm{inj}}\,.$$

Proof. We consider the operator $P_{\mathrm{st}}P_{\mathrm{inj}}$. We have that

$$\begin{aligned}[P_{\mathrm{st}}P_{\mathrm{inj}}](\mathbf{x}) &= \sum_{\mathbf{z} \in G_{\mathbf{h}}} s'(\mathbf{z}) \cdot [P_{\mathrm{inj}}u](\mathbf{x} + \mathbf{z}) \\ &= \sum_{\mathbf{z} \in G_{\mathbf{h}}} s'(\mathbf{z}) \cdot [P_{\mathrm{inj}}u](\mathbf{x}_b + \mathbf{t}_{\mathbf{k}} + \mathbf{z})\,.\end{aligned}$$

Substituting $\mathbf{z}$ by $\mathbf{y} - \mathbf{t}_{\mathbf{k}}$ yields the equation

$$[P_{\mathrm{st}}P_{\mathrm{inj}}](\mathbf{x}) = \sum_{\mathbf{y} \in G_{\mathbf{h}}} s'(\mathbf{y} - \mathbf{t}_{\mathbf{k}}) \cdot [P_{\mathrm{inj}}u](\mathbf{x}_b + \mathbf{y})\,.$$

Using the formula for P_{inj} given in Theorem 3.4, we can write the last equation as

$$\begin{aligned}[P_{\mathrm{st}}P_{\mathrm{inj}}](\mathbf{x}) &= \sum_{\mathbf{y} \in G_{\mathbf{n}\cdot\mathbf{h}}} s'(\mathbf{y} - \mathbf{t}_{\mathbf{k}}) \cdot u(\mathbf{x}_b + \mathbf{y}) \\ &= \sum_{\mathbf{y} \in G_{\mathbf{n}\cdot\mathbf{h}}} s_{\mathbf{k}}(\mathbf{y}) \cdot u(\mathbf{x}_b + \mathbf{y})\,.\end{aligned}$$

□

Let us apply Theorem 3.8 to the linear interpolation in 1D. To apply the theorem we need to define the stencils $s_{\mathbf{0}}$ and $s_{\mathbf{1}}$, where $s_{\mathbf{0}}$ is the stencil describing the interpolation for the points $\mathbf{x} \in G_{2\mathbf{h}}$ and $s_{\mathbf{1}}$ is the stencil describing the interpolation for the points $\mathbf{x} \in G_{2\mathbf{h}} + \mathbf{t}_{\mathbf{1}}$. If we define

$$s_{\mathbf{0}} = \left[\underline{1}\right]_{2\mathbf{h}} \quad \text{and} \quad s_{\mathbf{1}} = \left[\tfrac{1}{2} \quad \underline{\tfrac{1}{2}}\right]_{2\mathbf{h}}\,,$$

some simple calculation shows that (3.9) is the linear interpolation (3.7). In this case the stencil s' defined in Theorem 3.8 is given by

$$s' = \left[\tfrac{1}{2} \quad \underline{1} \quad \tfrac{1}{2}\right]_{\mathbf{h}}\,.$$

We already know that this stencil gives an operator P_{st} such that $P_{\mathrm{lin}} = P_{\mathrm{st}}P_{\mathrm{inj}}$.

3.2. Matrix Symbols

We saw that Fourier representations of constant stencil operators are multiplication operators. In this section we introduce matrix multiplication operators. They will be used in later sections to represent the Fourier representation of operators like the injection restriction or injection interpolation operators. We start with the definition of *matrix symbols*.

Definition 3.9. Let $0 < \mathbf{h} \in \mathbb{R}^d$. An $\mathbf{m} \times \mathbf{n}$-matrix symbol with domain $\Theta_{\mathbf{h}}$ is a is a family of functions in $L_\infty(\Theta_{\mathbf{h}})$. More precisely,

$$a_{\mathbf{ij}} \in L_\infty(\Theta_{\mathbf{h}}) \quad \text{for} \quad \mathbf{i} = \mathbf{0}, \ldots, \mathbf{n} - \mathbf{1},\ \mathbf{j} = \mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\,.$$

We denote the set of all $\mathbf{m} \times \mathbf{n}$-matrix symbols with domain $\Theta_{\mathbf{h}}$ by $L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_{\mathbf{h}})$.

Note that the indices $\mathbf{i}$ and $\mathbf{j}$ are multiindices. Furthermore, we start counting by zero, as it will simplify some of the proofs later. Let us define the most common operations on matrix symbols.

Definition 3.10. Let $a \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_{\mathbf{h}})$.

1. Let $b \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_{\mathbf{h}})$. Then the matrix addition is given by
$$(a + b)_{\mathbf{ij}} = a_{\mathbf{ij}} + b_{\mathbf{ij}}\,. \tag{3.10}$$
2. Let $\lambda \in \mathbb{C}$. Then the scalar multiplication is given by
$$(\lambda a)_{\mathbf{ij}} = \lambda \cdot a_{\mathbf{ij}}\,.$$
3. Let $u \in L_2(\Theta_{\mathbf{h}})^{\mathbf{n}}$. Then the matrix-vector-product $a \cdot u$ gives a vector $f \in L_2(\Theta_{\mathbf{h}})^{\mathbf{m}}$ such that
$$f_{\mathbf{i}} = (a \cdot u)_{\mathbf{i}} := \sum_{\mathbf{j}=0}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{ij}} \cdot u_{\mathbf{j}}\,. \tag{3.11}$$
4. Let $b \in L_\infty^{\mathbf{n}\times\mathbf{p}}$. Then the matrix-matrix-product $a \cdot b$ gives a matrix $c \in L_\infty^{\mathbf{m}\times\mathbf{p}}$ such that
$$c_{\mathbf{ij}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{ik}} \cdot b_{\mathbf{kj}}\,. \tag{3.12}$$
5. For $\mathbf{n} = \mathbf{m}$ the symbol a is called *invertible* if there exists a matrix symbol $a^{-1} \in L_\infty^{\mathbf{n}\times\mathbf{n}}$ such that
$$a^{-1} \cdot a = a \cdot a^{-1} = I\,,$$
where I is the identity matrix. The matrix symbol a^{-1} is called the *inverse* of a.
6. The *adjoint* $a^* \in L_\infty^{\mathbf{n}\times\mathbf{m}}$ of the matrix symbol a is given by
$$a^*_{\mathbf{ij}} = \overline{a_{\mathbf{ji}}}\,.$$

Assume that we multiply a matrix symbol to a vector from $L_2(\Theta_{\mathbf{h}})^{\mathbf{n}}$ and multiply another matrix symbol to the result of this operation. The result of these two multiplications can be obtained by multiplying by just one matrix symbol, as the following proposition shows.

Proposition 3.11. *Let $a \in L_\infty^{\mathbf{n}\times\mathbf{m}}(\Theta_\mathbf{h})$, $b \in L_\infty^{\mathbf{m}\times\mathbf{p}}$, and $u \in L_2(\Theta_\mathbf{h})^\mathbf{p}$. Then*

$$a \cdot (b \cdot u) = (a \cdot b) \cdot u \,.$$

Proof. It holds that

$$\begin{aligned}\big(a \cdot (b \cdot f)\big)_\mathbf{i} &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_\mathbf{ij} \cdot (b \cdot f)_\mathbf{j} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_\mathbf{ij} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{p}-\mathbf{1}} b_\mathbf{jk} f_\mathbf{k} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{p}-\mathbf{1}} \left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_\mathbf{ij} b_\mathbf{jk} \right) f_\mathbf{k} \\ &= \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{p}-\mathbf{1}} (a \cdot b)_\mathbf{ik} f_\mathbf{k} = \big((a \cdot b) \cdot f\big)_\mathbf{i} \,.\end{aligned}$$

□

In a similar way as we defined ordinary multiplication operators, we can define *matrix* multiplication operators.

Definition 3.12. Let $a \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_\mathbf{h})$. Then a is the *matrix symbol* of the *(matrix) multiplication operator* $A : L_2(\Theta_\mathbf{h})^\mathbf{n} \to L_2(\Theta_\mathbf{h})^\mathbf{m}$ given by

$$Af := a \cdot f \,.$$

In the remainder of this section let us sum up a few properties of matrix multiplication operators.

Proposition 3.13. *If A is a matrix multiplication operator, then its matrix symbol is unique.*

Proof. The statement can be proven in a similar way as Proposition 2.22. □

Many actions on multiplication operators directly translate into actions on their symbols (Section 2.3.1). The same is true for matrix multiplication operators.

Theorem 3.14. *Let A, B, C be three multiplication operators with symbols $a, b \in L_\infty^{\mathbf{m}\times\mathbf{n}}$, $c \in L_\infty^{\mathbf{n}\times\mathbf{p}}$, and let $\lambda \in \mathbb{C}$. Then*

1. *1 is the symbol of the identity I,*
2. *$\lambda \cdot a + b$ is the symbol of $\lambda A + B$,*
3. *$a \cdot c$ is the symbol of AC, and*
4. *a^* is the symbol of A^*.*

Proof. The first two assertions can be proven in a similar way as the corresponding statements in Theorem 2.23. For the third assertion consider the following:

$$A(Cf) = a \cdot (c \cdot f) = (a \cdot c) \cdot f = (AC)f \,.$$

The last assertion follows from the fact that

$$\begin{aligned}\langle Af, g\rangle &= \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \langle \big(Af\big)_\mathbf{i}, g_\mathbf{i}\rangle = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \langle \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_\mathbf{ij} f_\mathbf{j}, g_\mathbf{i}\rangle = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_\mathbf{ij} \cdot \langle f_\mathbf{j}, g_\mathbf{i}\rangle \\ &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \langle f_\mathbf{j}, \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \overline{a_\mathbf{ij}} g_\mathbf{i}\rangle = \langle f, A^* g\rangle \,.\end{aligned}$$

□

Proposition 3.15. *If a is invertible then*

$$a^{-1}(\theta) = (a(\theta))^{-1} \quad a.e. \tag{3.13}$$

Proof. As a is invertible

$$a^{-1} \cdot a = I$$

which implies

$$a^{-1}(\theta) \cdot a(\theta) = I \quad \text{a.e.}$$

The function value $a(\theta)$ is a matrix, and the last formula implies that it is invertible for almost all θ. Thus, we can multiply by $(a(\theta))^{-1}$ from the left to get

$$a^{-1}(\theta) = (a(\theta))^{-1} \quad \text{a.e.}$$ □

It is sometimes useful to have a way of determining whether an operator has a matrix symbol or not. The next theorem gives a necessary and sufficient condition for this property.

Theorem 3.16. *The operator $A \in L(L_2^{\mathbf{n}}(\Theta_{\mathbf{h}}); L_2^{\mathbf{m}}(\Theta_{\mathbf{h}}))$ is a matrix multiplication operator if and only if*

$$A(\alpha \cdot u) = \alpha \cdot [Au] \quad \textit{for all} \quad \alpha \in L_\infty, u \in L_\infty^{\mathbf{n}}\,. \tag{3.14}$$

Proof. Assume that A is a matrix multiplication operator. Then

$$[A(\alpha \cdot u)]_{\mathbf{i}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} a_{\mathbf{ij}} \cdot (\alpha \cdot u) = \alpha \cdot \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} a_{\mathbf{ij}} \cdot u = \alpha \cdot [Au],$$

which shows the first implication.

Conversely, assume that condition (3.14) holds. Any $u \in L_2^{\mathbf{n}}$ can be written as

$$u = e_{\mathbf{0}} u_{\mathbf{0}} + \cdots + e_{\mathbf{n-1}} u_{\mathbf{n-1}}\,,$$

where $e_{\mathbf{j}} \in L_\infty^2(\Theta_{\mathbf{h}})$ with $e_{\mathbf{j},\mathbf{k}} = \delta_{\mathbf{jk}}$. By linearity of A we have

$$Au = [Ae_{\mathbf{0}} u_{\mathbf{0}}] + \cdots + [Ae_{\mathbf{n-1}} u_{\mathbf{n-1}}],$$

and using the condition (3.14),

$$Au = u_{\mathbf{0}}[Ae_{\mathbf{0}}] + \cdots + u_{\mathbf{n-1}}[Ae_{\mathbf{n-1}}]\,.$$

Restricting this equation to the $\mathbf{i}$-th component we have

$$[Au]_{\mathbf{i}} = \sum_{\mathbf{j}} [Ae_{\mathbf{j}}]_{\mathbf{i}} \cdot u_{\mathbf{j}}\,,$$

which is a matrix-vector multiplication where the matrix entries are given by $a_{\mathbf{ij}} = [Ae_{\mathbf{j}}]_{\mathbf{i}}$.

It remains to show that the entries $a_{\mathbf{ij}}$ are bounded, which can be proven by contradiction. For this purpose, assume that there exist $\mathbf{i}, \mathbf{j}$ such that $a_{\mathbf{ij}}$ is unbounded in the infinity norm. In this case, there exists for any $C > 0$ a set M of nonzero measure such that $a_{\mathbf{ij}}(\theta) > C$ for all $\theta \in M$. Using the characteristic function χ_M of the set M, we obtain that

$$\|[A(e_{\mathbf{j}} \chi_M)]_{\mathbf{i}}\| = \|a_{\mathbf{ij}} \chi_M\| \geq C \|\chi_M\|\,.$$

Hence, A would be unbounded, which is a contradiction. Therefore, $a_{\mathbf{ij}}$ must be bounded. □

A consequence of the previous theorem is the next proposition.

Proposition 3.17. *Let A be a matrix multiplication operator. If A is invertible then the inverse A^{-1} is also a matrix multiplication operator.*

Proof. As A is a matrix multiplication operator, we have for all $\alpha \in L_\infty$ and $u \in L_\infty^{\mathbf{n}}$ that

$$A(\alpha \cdot u) = \alpha \cdot [Au]\,.$$

If we multiply the last equation by A^{-1} from the left and substitute $A^{-1}w$ for u, we get

$$\alpha \cdot [A^{-1}w] = A^{-1}(\alpha \cdot [A[A^{-1}w]]) = A^{-1}(\alpha \cdot w)\,. \qquad \square$$

Corollary 3.18. *A matrix multiplication operator is invertible if and only if its symbol is invertible.*

3.3. Fourier Matrix Symbols

Fourier symbols, as introduced in Section 2.2, can only represent constant stencil operators, which limits their applicability. We shall introduce *Fourier matrix symbols*, which are a generalization of ordinary Fourier symbols and are not limited to representing constant stencil operators. For example, they can represent the injection restriction and injection interpolation operators.

Definition 3.19 (Fourier Matrix Symbol)**.** Let $0 < \mathbf{h}_{\text{in}}, \mathbf{h}_{\text{out}} \in \mathbb{R}^d$, $0 < \mathbf{m}, \mathbf{n} \in \mathbb{N}^d$ such that

$$\mathbf{n} \cdot \mathbf{h}_{\text{in}} = \mathbf{m} \cdot \mathbf{h}_{\text{out}}\,. \tag{3.15}$$

Furthermore let $A \in L(\ell_2(G_{\mathbf{h}_{\text{in}}}); \ell_2(G_{\mathbf{h}_{\text{out}}}))$. If the Fourier representation $\widehat{A}$ of A can be written in the form

$$\widehat{A}\hat{u} = \mathcal{R}_{\mathbf{m}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}}\hat{u}\,, \tag{3.16}$$

where $\hat{a} \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})$, then $\hat{a}$ is called the *Fourier matrix symbol* of A.

The relation between A, its Fourier representation $\widehat{A}$, and its Fourier matrix symbol $\hat{a}$ is shown in Figure 3.2. When an operator has an ordinary Fourier symbol, the action of the Fourier representation is just the pointwise multiplication with the corresponding symbol. In contrast to that, if an operator A has a Fourier *matrix* symbol $\hat{a}$, then to give the action of the Fourier matrix, we first have to use $\mathcal{R}_{\mathbf{n}}$ to split the argument into a vector of functions. Then we can apply the matrix product with $\hat{a}$, and after that we need to combine the elements of the resulting vector using $\mathcal{R}_{\mathbf{m}}^{-1}$ into a single function.

Remark 3.20. Equation (3.16) is the important part of Definition 3.19. The relation between the step-sizes $\mathbf{h}_{\text{in}}$ and $\mathbf{h}_{\text{out}}$ is just needed to make (3.16) consistent, which can be seen as follows. If $\hat{u} \in L_2(\Theta_{\mathbf{h}_{\text{in}}})$ then $\mathcal{R}_{\mathbf{n}}\hat{u} \in L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})^{\mathbf{n}}$, implying that the entries of $\hat{a}$ have to be in $L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})$ in order to multiply $\hat{a}$ and $\mathcal{R}_{\mathbf{n}}\hat{u}$. It follows that $\hat{a}\mathcal{R}_{\mathbf{n}}\hat{u} \in L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})^{\mathbf{m}}$ and from this relation that $\mathcal{R}_{\mathbf{m}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}}\hat{u} \in L_2(\Theta_{(\mathbf{n}/\mathbf{m})\cdot\mathbf{h}_{\text{in}}})$, which implies $\mathbf{h}' = (\mathbf{n}/\mathbf{m}) \cdot \mathbf{h}_{\text{in}}$. Now, the last equation is equivalent to (3.15). In short, we can sum up the argument by

$$L_2(\Theta_{\mathbf{h}_{\text{in}}}) \xrightarrow{\mathcal{R}_{\mathbf{n}}} L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})^{\mathbf{n}} \xrightarrow{\hat{a}} L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})^{\mathbf{m}} \xrightarrow{\mathcal{R}_{\mathbf{m}}^{-1}} L_2(\Theta_{(\mathbf{n}/\mathbf{m})\cdot\mathbf{h}_{\text{in}}})\,.$$

$$\begin{array}{ccccc}
\ell_2(G_{\mathbf{h}_{\text{in}}}) & \xrightarrow{\mathcal{F}_{\mathbf{h}}} & L_2(\Theta_{\mathbf{h}_{\text{in}}}) & \xrightarrow{\mathcal{R}_{\mathbf{n}}} & L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})^{\mathbf{n}} \\
\Big\downarrow A & & \Big\downarrow \widehat{A} & & \Big\downarrow \hat{u} \mapsto \hat{a}\cdot\hat{u} \\
\ell_2(G_{\mathbf{h}_{\text{out}}}) & \xrightarrow[\mathcal{F}_{\mathbf{h}}]{} & L_2(\Theta_{\mathbf{h}_{\text{out}}}) & \xrightarrow[\mathcal{R}_{\mathbf{m}}]{} & L_2(\Theta_{\mathbf{m}\cdot\mathbf{h}_{\text{out}}})^{\mathbf{m}}
\end{array}$$

Figure 3.2.: Relation between an operator $\widehat{A}$, its Fourier representation $\widehat{A}$, and its Fourier matrix symbol $\hat{a}$.

We shall now prove that the injection restriction and interpolation can be written in the form of (3.16). In later sections of this chapter we shall see how we can use the Fourier matrix symbol of an operator to obtain its operator norm and spectral radius.

Proposition 3.21. *The injection restriction operator* $R_{\text{inj}} : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{n}\cdot\mathbf{h}})$ *from* (3.1) *has a Fourier matrix symbol* $\hat{r} \in L_\infty^{\mathbf{1}\times\mathbf{n}}(\Theta_{\mathbf{h}\cdot\mathbf{n}})$*. Its entries are all equal to the same constant function. More precisely,*

$$\hat{r}_{\mathbf{0j}}(\theta) = \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \quad \textit{for} \quad \theta \in \Theta_{\mathbf{h}\cdot\mathbf{n}}\,. \tag{3.17}$$

Proof. To derive the Fourier matrix symbol of the injection restriction let us rewrite (3.16)

$$\mathcal{R}_{\mathbf{m}}\widehat{R}_{\text{inj}}u = \hat{a}\cdot[\mathcal{R}_{\mathbf{n}}u].$$

This equation is equivalent to

$$[\mathcal{R}_{\mathbf{m}}\widehat{R}_{\text{inj}}u]_{\mathbf{k}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \hat{a}_{\mathbf{kj}}\cdot[\mathcal{R}_{\mathbf{n}}u]_{\mathbf{j}} \quad \text{for} \quad \mathbf{k} = \mathbf{0},\dots,\mathbf{m}-\mathbf{1}\,. \tag{3.18}$$

Comparing this equation to the Fourier representation of the injection restriction,

$$\widehat{R}_{\text{inj}}\hat{u} = \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} [\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{j}}, \tag{3.4 revisited}$$

shows that if we choose $\mathbf{m} = \mathbf{1}$ and

$$\hat{r}_{\mathbf{0j}} = \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}}, \tag{3.19}$$

then $\hat{r}$ is the Fourier matrix symbol of the injection restriction. □

In a similar way, comparing (3.18) to the Fourier representation of the injection interpolation (3.6) gives that $\hat{p} \in L_\infty^{\mathbf{n}\times\mathbf{1}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$ with

$$\hat{p}_{\mathbf{k0}} = \tfrac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \tag{3.20}$$

is the Fourier matrix symbol of the injection interpolation.

Fourier matrix symbols are a generalization of Fourier symbols; if we interpret the Fourier symbol $\hat{a}$ as a $\mathbf{1}\times\mathbf{1}$-matrix symbol, it is easy to see that we can write the Fourier representation $\widehat{A}$ of the corresponding operator as

$$\widehat{A} = \mathcal{R}_{\mathbf{1}}^{-1}\hat{a}\mathcal{R}_{\mathbf{1}}\,.$$

Hence, in principle every Fourier symbol is also a $\mathbf{1}\times\mathbf{1}$-Fourier matrix symbols.

The next theorem shows that every operator which can be represented by an ordinary Fourier symbol can also be represented by an $\mathbf{n}\times\mathbf{n}$-Fourier matrix symbol for any $1 \leq \mathbf{n} \in \mathbb{R}^d$. To formulate this theorem, we need to define the frequency splitting of a symbol $\hat{a} \in L_\infty(\Theta_\mathbf{h})$. In this case the frequency splitting operator is defined like in (3.3), i.e.,

$$[\mathcal{R}_\mathbf{n}\hat{a}]_\mathbf{j}(\theta) := \hat{a}(\theta + \mathbf{s}_\mathbf{j}^{\mathbf{h},\mathbf{n}}). \tag{3.21}$$

Theorem 3.22. *If A has a Fourier (scalar) symbol $\hat{a}$, then for any factor $0 < \mathbf{n} \in \mathbb{N}^d$ the operator has a $\mathbf{n}\times\mathbf{n}$-Fourier matrix symbol $\acute{a}$ given by*

$$\acute{a}_{\mathbf{kj}}(\theta) = [\mathcal{R}_\mathbf{n}\hat{a}]_\mathbf{k}\cdot\delta_{\mathbf{kj}} = \begin{cases}[\mathcal{R}_\mathbf{n}\hat{a}]_\mathbf{k} & \text{if } \mathbf{k} = \mathbf{j},\\ 0 & \text{otherwise.}\end{cases}$$

We call $\acute{a}$ the expansion *of $\hat{a}$ and denote it by* $\operatorname{expand}_\mathbf{n}(\hat{a}) := \acute{a}$.

Proof. We have

$$\widehat{A}\hat{u} = \hat{a}\cdot\hat{u} = \mathcal{R}_\mathbf{n}^{-1}\mathcal{R}_\mathbf{n}\hat{a}\cdot\hat{u} = \mathcal{R}_\mathbf{n}^{-1}\acute{a}\mathcal{R}_\mathbf{n}\hat{u}\,. \qquad \square$$

Many interpolation operators can be written as the composition of a constant stencil and injection restriction operator. We derive the Fourier matrix symbol of the composed operator.

Proposition 3.23. *Let P_st be a (constant) stencil operator and $\hat{p}_\text{inj}$ its Fourier symbol. Furthermore, let P_inj be the injection interpolation and $\hat{p}_\text{inj}$ its Fourier matrix symbol. Then the Fourier matrix symbol $\hat{p} \in L_\infty^{\mathbf{m}\times\mathbf{1}}(\Theta_\mathbf{h})$ of $P := P_\text{st}P_\text{inj}$ is*

$$\hat{p}_{\mathbf{k0}} = \tfrac{1}{\operatorname{vol}_\mathbf{m}^{1/2}}[\mathcal{R}_\mathbf{m}\hat{p}_\text{st}]_\mathbf{k}\,. \tag{3.22}$$

Proof. Let $\hat{p}'_\text{st}$ be the expansion of $\hat{p}_\text{st}$. The Fourier matrix symbol of P is then given by the matrix-matrix-product of $\hat{p}'_\text{st}$ and $\hat{p}_\text{inj}$. We have

$$\begin{aligned}\hat{p}_{\mathbf{k0}} &= \sum_\mathbf{s}\hat{p}'_{\text{st},\mathbf{ks}}\cdot\hat{p}_{\text{inj},\mathbf{s0}}\\ &= \sum_\mathbf{s}[\mathcal{R}_\mathbf{m}\hat{p}]_\mathbf{k}\delta_\mathbf{ks}\cdot\tfrac{1}{\operatorname{vol}_\mathbf{m}^{1/2}}\\ &= \tfrac{1}{\operatorname{vol}_\mathbf{m}^{1/2}}\cdot[\mathcal{R}_\mathbf{m}\hat{p}]_\mathbf{k}\,.\end{aligned} \qquad \square$$

The next proposition is a similar result for restriction operators. We omit the prove as it is very similar to the proof of the last proposition.

Proposition 3.24. *Let R_st be a (constant) stencil operator and $\hat{r}_\text{inj}$ its Fourier symbol. Furthermore, let R_inj be the injection restriction and $\hat{r}_\text{inj}$ is Fourier matrix symbol. Then the Fourier matrix symbol $\hat{r} \in L_\infty^{\mathbf{1}\times\mathbf{n}}(\Theta_\mathbf{h})$ of $R := R_\text{inj}R_\text{st}$ is*

$$\hat{r}_{\mathbf{0j}} = \tfrac{1}{\operatorname{vol}_\mathbf{n}^{1/2}}[\mathcal{R}_\mathbf{m}\hat{r}_\text{st}]_\mathbf{j}\,. \tag{3.23}$$

3.3.1. The Two-Grid Method

The error propagator of the two-grid method is

$$E_{\mathrm{TG}} = S^{\nu_2}(I - PA_H^{-1}RA_h)S^{\nu_1}\,, \qquad \text{(1.35 revisited)}$$

where S is the error propagator of the smoother, P the interpolation, A_H the coarse grid matrix, R the restriction, and A_h the fine grid matrix. If we consider the Fourier matrix symbol of each of these operators, we can derive the Fourier matrix symbol $\hat{e}_{\mathrm{TG}}$ of the two-grid method.

Assume that S, A_h, and A_H have ordinary Fourier symbols: $\hat{s}$, $\hat{a}_h$, and $\hat{a}_H$. We have discussed that we can interpret these symbols as $\mathbf{1}\times\mathbf{1}$-matrix symbols, implying that $\hat{s}, \hat{a}_h \in L_\infty^{\mathbf{1}\times\mathbf{1}}(\Theta_{\mathbf{h}})$ and $\hat{a}_H \in L_\infty^{\mathbf{1}\times\mathbf{1}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$. Furthermore, we assume that $\hat{p} \in L_\infty^{\mathbf{n}\times\mathbf{1}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$ and $\hat{r} \in L_\infty^{\mathbf{1}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$ are the Fourier matrix symbols of the interpolation P and the restriction R.

The problem is that we want to multiply $\hat{r}$ and $\hat{a}_h$, but their dimensions do not match. We can, however, use Theorem 3.22 to expand $\hat{a}_h$ to a $\mathbf{n}\times\mathbf{n}$-matrix symbol $\acute{a}_h \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$. Then we can compute the product $\hat{r}\acute{a}_h$. For the same reason we expand the symbol $\hat{s}$ to a $\mathbf{n}\times\mathbf{n}$-matrix symbol $\acute{s} \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{n}\times\mathbf{h}})$. Using these expanded matrix symbols, we obtain the Fourier matrix symbols of the error propagator of the two-grid method by

$$\hat{e}_{\mathrm{TG}} = \acute{s}^{\nu_2}(1 - \hat{p}\hat{a}_H^{-1}\hat{r}\acute{a}_h)\acute{s}^{\nu_1}\,.$$

3.4. Interpretation and Visualization of Matrix Symbols

We are able to represent the error propagator of many two-grid methods (cf. Section 3.3.1) by a Fourier matrix symbol. As we shall see, the Fourier matrix symbol gives us some information about the behavior of the error propagator.

Let us first assume that A has a (scalar) Fourier symbol $\hat{a}$; then $\hat{a}(\theta)$ is a scalar for some $\theta \in \Theta_{\mathbf{h}}$. This scalar provides a *direct relation* between *one* input frequency and *one* output frequency. The input frequency is multiplied by the scalar to obtain the corresponding output frequency, i.e.,

$$[\widehat{A}\hat{f}](\theta) = \hat{a}(\theta)\cdot\hat{f}(\theta)\,.$$

This relation becomes handy if we analyze, e.g., a smoothing error propagator E. If $\hat{e}(\theta)$, the Fourier matrix symbol of E, is small for high frequencies θ, then the norm of highly varying inputs will be substantially reduced, *and* the output will consist mainly of low frequencies. Thus, we have answered two questions at once: First, how does a dominating frequency of the input influence the norm of the output? Second, which frequencies dominate the output? In the case where $\hat{a}$ is a matrix symbol, the relation is not as obvious. We only have that $\hat{a}(\theta)$ for $\theta \in \Theta_{\mathbf{h}}$ is a matrix that relates the $\mathbf{m}$-harmonics of θ of the input to the $\mathbf{n}$-harmonics of the output by the matrix-vector product (3.11).

For $d \leq 3$ the scalar symbols $\hat{a} \in L_\infty(\Theta_{\mathbf{h}})$ can be visualized, e.g., by a contour or an isosurface plot. If we have a matrix symbol in $L_\infty^{\mathbf{n}\times\mathbf{m}}(\Theta_{\mathbf{h}})$, we have $\mathrm{vol}_{\mathbf{n}}\cdot\mathrm{vol}_{\mathbf{m}}$ functions from $L^\infty(\Theta_{\mathbf{h}})$. As this number grows quickly, plotting every single function is not feasible. In this section we discuss how to reduce a matrix symbol to a scalar function, to obtain the

same pieces of information as in the scalar case. Unlike in the scalar case, two different transformations are needed. The choice of the transformation depends on whether we are interested in the influence of certain input frequencies or in the dominating frequencies in the output.

3.4.1. Frequency Damping

Let us start by considering how strong the influence of the function value $\hat{u}(\theta)$ for a function $\hat{u}$ and a frequency θ is on the size of the function resulting from the computation of $\widehat{A}\hat{u}$. For this purpose, consider the function

$$u_{\theta_0}(\theta) := \begin{cases} 1 & \text{if } \theta = \theta_0, \\ 0 & \text{otherwise,} \end{cases}$$

and compute

$$\widehat{A}u_{\theta_0} = \mathcal{R}_{\mathbf{m}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}}u_{\theta_0}\,.$$

We start by computing $\mathcal{R}_{\mathbf{n}}u_{\theta_0}$. Decomposing the frequency θ_0 into a base frequency $\theta_b \in \Theta_{\mathbf{n}\cdot\mathbf{h}}$ and a frequency shift $\mathbf{s_k}$, i.e., $\theta_0 = \theta_b + \mathbf{s_k}$, gives that

$$\begin{aligned}
[\mathcal{R}_{\mathbf{n}}u_{\theta_0}]_{\mathbf{j}}(\theta) &= u_{\theta_0}(\theta + \mathbf{s_j}) \\
&= \begin{cases} 1 & \text{if } \theta_b + \mathbf{s_k} = \theta + \mathbf{s_j}, \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} 1 & \text{if } \theta_b = \theta \text{ and } \mathbf{s_k} = \mathbf{s_j}, \\ 0 & \text{otherwise} \end{cases} \\
&= \delta_{\theta_b\theta_0} \cdot \delta_{\mathbf{jk}}\,.
\end{aligned}$$

If we apply the matrix symbol $\hat{a}$ we obtain that

$$\begin{aligned}
[\hat{a}\mathcal{R}_{\mathbf{n}}u_{\theta_0}]_{\mathbf{i}}(\theta) &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \hat{a}_{\mathbf{ij}}(\theta) \cdot \delta_{\theta_b\theta_0} \cdot \delta_{\mathbf{jk}} \\
&= \hat{a}_{\mathbf{ik}}(\theta) \cdot \delta_{\theta\theta_0} \\
&= \begin{cases} \hat{a}_{\mathbf{ik}}(\theta) & \text{if } \theta = \theta_0, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Thus, $\hat{a}\mathcal{R}_{\mathbf{n}}u_{\theta_0}$ is a vector of functions, where each entry is a function that is zero everywhere, except at θ_b. If we join the frequencies back together, i.e., by applying $\mathcal{R}_{\mathbf{m}}^{-1}$, the result is a function that is non-zero only at the $\mathbf{n}$,$\mathbf{h}$-harmonics of θ and takes the function values $a_{\mathbf{ik}}(\theta_b)$ for $\mathbf{i} = \mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}$. Hence, to measure the impact of the frequency θ_0 on the size of the output, we compute the Euclidean norm of the vector $\left(\hat{a}_{\mathbf{ik}}(\theta_b)\right)_{\mathbf{i}=\mathbf{0}}^{\mathbf{m}-\mathbf{1}}$. Let us define the

function $w_{\text{in}} : \Theta_h \to \mathbb{R}$, which assigns every *input frequency* this measure for the impact that the frequency has on the size of the output, i.e.,

$$w_{\text{in}}(\theta_b + \mathbf{s_k}) = \left(\sum_{\mathbf{i=0}}^{\mathbf{n-1}} |\hat{a}_{\mathbf{ik}}(\theta_b)|^2 \right)^{1/2}.$$

Applying the frequency splitting operator yields that

$$[\mathcal{R}_{\mathbf{n}} w_{\text{in}}]_{\mathbf{k}}(\theta_b) = \left(\sum_{\mathbf{i=0}}^{\mathbf{n-1}} |\hat{a}_{\mathbf{ik}}(\theta_b)|^2 \right)^{1/2}.$$

If we define $\text{cn}(\hat{a})$ to be the vector

$$[\text{cn}(\hat{a})]_{\mathbf{k}} = \left(\sum_{\mathbf{i=0}}^{\mathbf{n-1}} |\hat{a}_{\mathbf{ik}}|^2 \right)^{1/2},$$

i.e., $\text{cn}(\hat{a})$ is the vector of *column norms* of $\hat{a}$, then $\mathcal{R}_{\mathbf{n}} w_{\text{in}} = \text{cn}(\hat{a})$. Hence, if we are interested in the damping of the input of $\widehat{A}$ then

$$\mathcal{R}_{\mathbf{n}}^{-1} \text{cn}(\hat{a})$$

describes how the input of the Fourier representation influence the result. We refer to this function as the *frequency damping* of the operator.

3.4.2. Frequency Emission

We shall now derive a way to measure, how strong a frequency θ will be in the output of $\widehat{A}$, in general. For this purpose, we compute a bound for

$$\left| [\widehat{A}u](\theta) \right| = \left| [\mathcal{R}_{\mathbf{m}}^{-1} \hat{a} \mathcal{R}_{\mathbf{n}} u](\theta) \right|$$

for some *normalized* functions u. How to properly normalize u, will become clear during the following computation. To bound $\left| [\widehat{A}u](\theta) \right|$ let us split the frequency θ into the base frequency $\theta_b \in \Theta_{\mathbf{n}\cdot\mathbf{h}}$ and a frequency shift $\mathbf{s_k}$, i.e., $\theta = \theta_b + \mathbf{s_k}$. With this splitting, we obtain that

$$\begin{aligned} \left| [\mathcal{R}_{\mathbf{m}}^{-1} \hat{a} \mathcal{R}_{\mathbf{n}} u](\theta_b + \mathbf{s_k}) \right| &= \left| [\hat{a} \mathcal{R}_{\mathbf{n}} u]_{\mathbf{k}}(\theta_b) \right| \\ &= \Big| \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \hat{a}_{\mathbf{kj}}(\theta_b) \cdot [\mathcal{R}_{\mathbf{n}} u]_{\mathbf{k}}(\theta_b) \Big|. \end{aligned}$$

Using the Cauchy-Schwarz inequality yields that

$$\left| [\mathcal{R}_{\mathbf{m}}^{-1} \hat{a} \mathcal{R}_{\mathbf{n}} u](\theta_b + \mathbf{s_k}) \right| \leq \left(\sum_{\mathbf{j=0}}^{\mathbf{n-1}} |\hat{a}_{\mathbf{kj}}(\theta_b)|^2 \right)^{1/2} \cdot \left(\sum_{\mathbf{j=0}}^{\mathbf{n-1}} |[\mathcal{R}_{\mathbf{n}} u]_{\mathbf{k}}(\theta_b)|^2 \right)^{1/2}.$$

We now assume that u is normalized in such a way that

$$\left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} |[\mathcal{R}_{\mathbf{n}} u]_{\mathbf{k}}(\theta_b)|^2 \right)^{1/2} = 1 \, .$$

In other words, the Euclidean norm of the vector $[\mathcal{R}_{\mathbf{n}} u](\theta_b)$ should be equal to one. We then have that

$$\left| [\mathcal{R}_{\mathbf{m}}^{-1} \hat{a} \mathcal{R}_{\mathbf{n}} u](\theta_b + \mathbf{s_k}) \right| \leq \left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} |\hat{a}_{\mathbf{kj}}(\theta_b)|^2 \right)^{1/2} \, .$$

Let us define the function $w_{\text{out}} : \Theta_{\mathbf{h}} \to \mathbb{R}$, where $w_{\text{out}}(\theta)$ equals the upper bound for the *output* Au at the frequency θ in the case that u is normalized from the last equation, i.e.,

$$w_{\text{out}}(\theta_b + \mathbf{s_k}) := \left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} |\hat{a}_{\mathbf{kj}}(\theta_b)|^2 \right)^{1/2} \, .$$

Using the frequency splitting operator $\mathcal{R}_{\mathbf{m}}$, we obtain that

$$[\mathcal{R}_{\mathbf{m}} w_{\text{out}}]_{\mathbf{k}}(\theta_b) = \left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} |\hat{a}_{\mathbf{kj}}(\theta_b)|^2 \right)^{1/2} \, .$$

If we define $\text{rn}(\hat{a})$ to be the vector

$$[\text{rn}(\hat{a})]_{\mathbf{k}} := \left(\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} |\hat{a}_{\mathbf{kj}}|^2 \right)^{1/2} \, ,$$

i.e., $\text{rn}(\hat{a})$ is the vector of *row norms* of $\hat{a}$, then $\mathcal{R}_{\mathbf{m}} w_{\text{out}} = \text{rn}(\hat{a})$. Hence, by construction,

$$\mathcal{R}_{\mathbf{m}}^{-1} \text{rn}(\hat{a})$$

describes an upper bound for the output of $\widehat{A}$ for a normalized input. We call this function the *frequency emission* of the operator.

3.4.3. The Red-Black Jacobi Method

To illustrate the difference of the frequency damping and the frequency emission, let us consider the *red-black Jacobi method*. We will discuss the derivation of the method and the derivation of its Fourier matrix symbol later in section 3.6.1. For now, we just look at the frequency damping and emission of the error propagator of the method when applied to the discrete Poisson equation in 2D (1.49). The two quantities are shown in Figure 3.3.

When looking at the frequency damping of the method, we see that the damping has some large values in the high part of the spectrum, i.e., in the center of the plot. Thus, we might conclude that the red-black Jacobi method is not a good smoother for the Poisson equation.

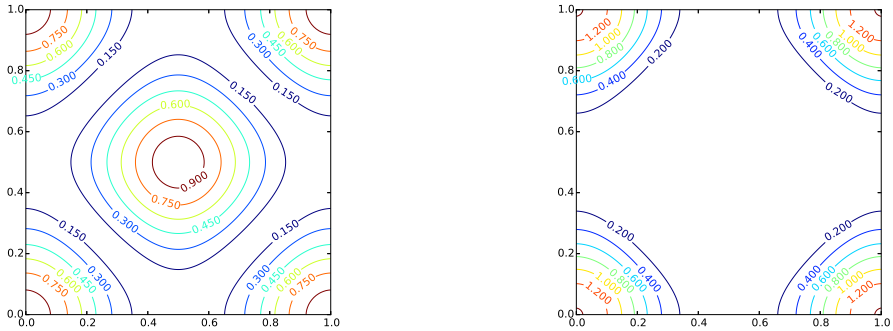


Figure 3.3.: Frequency damping (left) and emission (right) of the red-black Jacobi method.

This conclusion, however, is wrong, as we can see from looking at the frequency emission. There, the large values are only located in the corners, where the low modes are located.

The difference between the frequency damping and emission can be explained in the following way. The frequency damping describes the influence of a certain input frequency on the size of the output. Thus, the frequency damping plot of the red-black Jacobi method shows that there are some oscillatory functions that result in a large output. On the other hand, the frequency emissions shows that the output will consists of mainly slowly varying functions.

This behavior is due to the fact that the red-black Jacobi method actually maps some oscillatory functions to slowly varying functions without reducing their norm. Hence, the red-black Jacobi method is indeed a smoother for the Poisson equation, because the output will usually be slowly varying, even though some oscillatory functions lead to large outputs.

This example illustrates the difference between the frequency damping and emission, and it shows that when we are interested in the smoothing effect of a certain method, we should look at the frequency emission.

In Chapter 4, we will look at further contour plots of the frequency damping and the frequency emission functions of various operators.

3.5. Spectral Properties of Matrix Multiplication Operators

The norm and the spectral radius are interesting properties of operators, especially of error propagators (see Section 1.6). If we consider an operator that has a Fourier symbol, we can obtain its norm and spectral radius from its symbol in a simple way (Corollary 2.34 and 2.35). We shall see that the Fourier *matrix* symbol gives the norm and spectral radius of the corresponding operator in a similar way. Let us start by giving the norm and spectral radius of a matrix multiplication operator in terms of its symbol.

Theorem 3.25. *Let $A \in L(L_2(\Theta_{\mathbf{h}}); L_2(\Theta_{\mathbf{h}}))$ be a matrix multiplication operator with symbol a. Then*

$$r(A) = \text{ess-sup}_{\theta \in \Theta_{\mathbf{h}\cdot\mathbf{n}}} r(a(\theta)) \qquad \textit{and} \qquad \|A\| = \text{ess-sup}_{\theta \in \Theta_{\mathbf{h}\cdot\mathbf{n}}} \|a(\theta)\|, \tag{3.24}$$

where

$$\operatorname{ess-sup}_{\theta\in\Theta_{\mathbf{h}\cdot\mathbf{n}}} f(\theta) := \inf\{\alpha > 0 : \mu(f > \alpha) = 0\}$$

is the essential supremum *of the function* $f \in L_\infty(\Theta_{\mathbf{h}\cdot\mathbf{n}})$.

We shall give the proof of Theorem 3.25 later; let us first consider some of its consequences. The theorem states that analogously to the scalar case the norm and spectral radius are given by the essential supremum of some function. In the case of the last theorem, the function is not the pointwise absolute value of the symbol but the pointwise norm and the pointwise spectral radius of the matrix symbol. This result carries over to operators with Fourier matrix symbols.

Theorem 3.26. *Let* $A \in L(\ell_2(G_{\mathbf{h}}); \ell_2(G_{\mathbf{h}}))$ *with Fourier matrix symbol* $\hat{a}$. *Then*

$$r(A) = \operatorname{ess-sup}_{\theta\in\Theta_{\mathbf{n}\cdot\mathbf{h}}} r(\hat{a}(\theta)) \qquad \text{and} \qquad \|A\| = \operatorname{ess-sup}_{\theta\in\Theta_{\mathbf{n}\cdot\mathbf{h}}} \|\hat{a}(\theta)\| . \tag{3.25}$$

Proof. Let $\widehat{A}$ be the Fourier representation of A and $\hat{a}$ the Fourier matrix symbol of A. Then Proposition 2.17 states that $\sigma(A) = \sigma(\widehat{A})$. We can write $\widehat{A}$ as

$$\widehat{A} = \mathcal{R}_{\mathbf{m}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}} .$$

Thus, $\widehat{A} - \lambda I$ is invertible if and only if $\mathcal{R}_{\mathbf{m}}^{-1}(\hat{a} - \lambda I)\mathcal{R}_{\mathbf{n}}$ is invertible, which is the case if and only if $\hat{a} - \lambda I$ is invertible, implying that $\sigma(A) = \sigma(\widehat{A}) = \sigma(\hat{a})$.

Let us turn to the norm of A. We know from Proposition 2.16 that $\|A\| = \|\widehat{A}\|$. Furthermore

$$\|\widehat{A}\| = \max_{\|\hat{u}\|=1} \|\widehat{A}\| = \max_{\|\hat{u}\|=1} \|\mathcal{R}_{\mathbf{m}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}}\hat{u}\| .$$

As $\mathcal{R}$ is an isometry, we can substitute $\mathcal{R}_{\mathbf{n}}\hat{u}$ by $\hat{u}'$, and

$$\|\widehat{A}\| = \max_{\|\hat{u}'\|=1} \|\mathcal{R}_{\mathbf{m}}^{-1}\hat{a} \cdot \hat{u}'\| .$$

By using again that $\mathcal{R}$ is an isometry, it follows that

$$\|\widehat{A}\| = \max_{\|\hat{u}'\|=1} \|\hat{a} \cdot \hat{u}'\| . \qquad \square$$

The remainder of this section is dedicated to the proof of Theorem 3.25. For this purpose let μ be the Lebesgue measure and $X := \Theta_{\mathbf{h}}$. (However, (X, Σ, μ) can be any σ-finite measure space.) Recall that the spectrum of a scalar multiplication operator is given by the essential range of its symbol (2.18). The essential union of the pointwise spectra of the matrix symbol plays a similar role for matrix multiplication operators. The essential union of the *pointwise* spectra $\sigma(a(x))$ is given by

$$\operatorname{ess-}\bigcup_{x\in X} \sigma(a(x)) := \bigcap_{b\in[a]} \overline{\bigcup_{x\in X} \sigma(b(x))},$$

where $[a]$ is the equivalence class of functions that are almost everywhere equal to a. It can be shown that the essential union of the pointwise spectra is given by

$$\text{ess-}\bigcup_{x\in X}\sigma(a(x)) = \{z\in\mathbb{C} : \forall\varepsilon > 0\,.\,\mu(\{x\in X : \sigma(a(x))\cap U_\varepsilon(z)\neq\emptyset\} > 0\}\,. \tag{3.26}$$

The essential union can be used to compute the spectrum of matrix multiplication operators.

Proposition 3.27. *Let $A : L^2(X)^n \to L^2(X)^n$ be a matrix multiplication operator with non-empty resolvent set $\rho(A)$ and matrix symbol a. Then*

$$\sigma(A) = \text{ess-}\bigcup_{x\in X}\sigma(a(x))\,.$$

Proof. See [39]. □

The proposition states that the spectrum of the matrix multiplication operator is the essential union of the pointwise spectra. We derive a formula for the spectral radius of a multiplication operator using this result.

Lemma 3.28. *Let A be defined like in Proposition 3.27. The spectral radius $r(A)$ of A is the essential supremum of the pointwise computed spectral radii of $a(\vartheta)$, i.e.,*

$$r(A) = \text{ess-sup}_{\vartheta\in X}\, r(a(\vartheta))\,.$$

Proof. We set

$$C := \text{ess-sup}_{\vartheta\in X}\, r(a(\vartheta)) = \inf\{\alpha > 0 : \mu(\{x\in X : r(a(x)) > \alpha\}) = 0\}\,. \tag{3.27}$$

It is known that

$$r(A) = \max_{\lambda\in\sigma(A)}|\lambda|\,.$$

Thus, we can prove the assertion by showing that

1. for every $\varepsilon' > 0$ and all $|\lambda| \geq C + \varepsilon'$ we have $\lambda\in\rho(A)$, and
2. for every $\varepsilon' > 0$ there exists $\lambda\in\sigma(A)$ with $|\lambda| \geq C - \varepsilon'$.

(1) We choose $\lambda\in\mathbb{C}$ such that $|\lambda| \geq C + \varepsilon'$. Then if we can find $\varepsilon > 0$ such that

$$\mu(\{x\in X : \sigma(a(x))\cap U_\varepsilon(\lambda)\neq\emptyset\}) = 0 \tag{3.28}$$

the first assertion follows from Proposition 3.27 and the formula for the essential union of the pointwise spectra (3.26). Let us show that this equation holds.

Let us chose $\varepsilon = \varepsilon'$. If the set $\{x\in X : \sigma(a(x))\cap U_\varepsilon(\lambda)\neq\emptyset\}$ is empty then its measure is clearly zero (equation (3.28) holds). Thus, we consider the case where this set is not empty. For any $x_0\in X$ such that $\sigma(a(x_0))\cap U_\varepsilon(\lambda)\neq\emptyset$, we find a $\lambda_0\in\sigma(a(x_0))\cap U_\varepsilon(\lambda)$. Since $\lambda_0\in U_\varepsilon(\lambda)$,

$$|\tilde{\lambda}| = |\lambda - (\lambda - \tilde{\lambda})| \geq |\lambda| - |\lambda - \tilde{\lambda}| > C + \varepsilon' - \varepsilon = C\,.$$

Furthermore, as $\lambda_0\in\sigma(a(x_0))$, i.e., λ_0 is an eigenvalue of x_0,

$$r(a(x_0)) \geq |\lambda_0| > C\,.$$

Since x_0 was an arbitrary element from $\sigma(a(x_0)) \cap U_\varepsilon(\lambda)$, we have shown that

$$\{x \in X : \sigma(a(x)) \cap U_\varepsilon(\lambda) \neq \emptyset\} \subseteq \{x \in X : r(a(x)) > C\}\,.$$

By the definition of the essential supremum and the constant C in (3.27) the measure of the right hand side is zero. Thus

$$\mu(\{x \in X : \sigma(a(x)) \cap U_\varepsilon(\lambda) \neq \emptyset\}) \leq \mu(\{x \in X : r(a(x)) > C\}) = 0$$

for all $|\lambda| > C + \varepsilon'$, and we arrive at (3.28), which establishes the first assertion.

(2) We want to prove the second assertion by contradiction. Assume that there exists $\varepsilon' > 0$ such that $|\lambda| < C - \varepsilon'$ implies $\lambda \in \rho(A)$. Consider the set $\{x \in X : r(a(x)) > C - \varepsilon\}$. We want to show that our assumption implies that the set has measure zero, which would be a contradiction to the definition of C in (3.27). For this purpose, define the set

$$M := \{\lambda \in \mathbb{C} : C - \varepsilon' \leq |\lambda| \leq C\}\,.$$

Then

$$\{x \in X : r(a(x)) > C - \varepsilon\} \subseteq \{x \in X : r(a(x)) > C\} \cup \{x \in X : \sigma(a(x)) \cap M \neq \emptyset\}\,.$$

The first set on the right hand size has zero measure, due to the definition of C in (3.27). Thus, it remains to show that the second set has also zero measure. By the definition of the essential union of the pointwise spectra (3.26) we have that for every $\lambda \in \rho(A)$ there exists $\varepsilon(\lambda) > 0$ such that

$$\mu(\{x \in X : \sigma(a(x)) \cap U_{\varepsilon(\lambda)}(\lambda) \neq \emptyset\}) = 0\,. \tag{3.29}$$

We have that $M \subseteq (\bigcup_{\lambda \in M} U_{\varepsilon(\lambda)}(\lambda))$, which is a family of open sets that cover set M. Since M is compact, we can find finitely many $\lambda_1, \ldots, \lambda_m \in M$ such that

$$M \subseteq \bigcup_{i=1}^{m} U_{\varepsilon(\lambda_i)}(\lambda_i),$$

giving that

$$\{x \in X : \sigma(a(x)) \cap M \neq 0\} \subseteq \bigcup_{i=1}^{m} \{x \in X : \sigma(a(x)) \cap U_{\varepsilon(\lambda_i)}(\lambda_i) \neq \emptyset\},$$

and as the sets on the right hand side have all zero measure due to (3.29) the assertion (ii.) is proven. □

With this lemma at hand, we can conduct the final proof of this section.

Proof of Theorem 3.25. We have

$$r(L) = \text{ess-sup}_{\vartheta \in \Theta_{\mathbf{h}\cdot\mathbf{n}}} r(\widehat{L}(\vartheta))$$

by Lemma 3.28. Thus

$$\|L\|^2 = \|L^*L\| = r(L^*L) = \text{ess-sup}_{\vartheta \in \Theta_{\mathbf{h}\cdot\mathbf{n}}} r(\widehat{L}(\vartheta)^*\widehat{L}(\vartheta))\,.$$ □

3.6. Periodic Stencils

In this section we introduce *periodic stencils* and show that an operator that is given by a periodic stencil has a Fourier matrix symbol. Periodic stencils can be used to analyze various applications and, therefore, form the basis for the analysis the problems in Chapter 4.

Definition 3.29 (Periodic Stencil Operator)**.** Let $0 < \mathbf{n} \in \mathbb{N}^d$ be given. If A is given by a stencil $\{s\}_{\mathbf{x}\in G_{\mathbf{h}}}$ and

$$s_{\mathbf{x}} = s_{\mathbf{x}+\mathbf{k}\cdot(\mathbf{h}\cdot\mathbf{n})} \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{h}},\ \mathbf{k} \in \mathbb{Z}^d\,, \tag{3.30}$$

then A is called a *periodic stencil operator*, $\{s\}_{\mathbf{x}\in G_{\mathbf{h}}}$ a *periodic stencil*, and $\mathbf{n}$ its *period*.

3.6.1. Red-Black Jacobi Method

As an example of a method that can be expressed in terms of periodic stencils, we introduce the red-black Jacobi method. It is a variant of the Jacobi method introduced in Section 1.3.1.

In every iteration the Jacobi method computes a set of corrections $v_1, \dots, v_n$. Then the new iterate u_{k+1} is obtained from the old iterate u_k by

$$u_{k+1} = u_k + \omega \cdot (v_1 + \cdots + v_n)\,. \tag{1.17 revisited}$$

The correction v_i is constructed such that the ith component of the correction residual is zero, i.e., $r_{u_k+v_i,i} = 0$. If $\omega = 1$ and all corrections v_i change only the corresponding ith component of the residual, then after one iteration the residual is zero, and we have found the solution of the linear system. This is, however, only the case when the matrix A is a diagonal matrix. In all other cases the correction v_i changes various, if not all, components of the residual. In that sense the corrections interfere with each other. They are, however, constructed completely independently from each other, as they only use the residual before the iteration as source of information. In other words, the ith correction is construction without using any information about how the other corrections change the residual.

The idea of the red-black Jacobi method is to increase the exchange of information between the corrections, which is done by partitioning the indices $i = 1, \dots, n$ into two set R, and B. Then an intermediate iterate is computed by

$$u_{k+1/2} = u_k + \omega \cdot \sum_{i\in R} v_i\,,$$

and an intermediate residual is computed based on the intermediate iterate. This residual is then used to construct the remaining corrections v_i for $i \in B$; the new iterate is

$$u_{k+1} = u_{k+1/2} + \omega \cdot \sum_{i\in B} v_i\,.$$

The *black corrections* v_i for $i \in B$ are computed from the intermediate residual. By that they use some information about the *red corrections* v_i for $i \in R$. Thus, we expect them to be better corrections than the corrections based on the initial residual.

We give the red-black Jacobi method in the Red-Black-Iteration procedure.

RED-BLACK-ITERATION()

1 **while** u is a bad approximation
2 **for** $i \leftarrow 1$ **to** n
3 $$u_{k+1/2,i} \leftarrow u_{k,i} + \begin{cases} \frac{\omega}{a_{ii}} r_{u_k,i} & \text{if } i \in R \\ 0 & \text{otherwise} \end{cases}$$
4 **for** $i \leftarrow 1$ **to** n
5 $$u_{k+1,i} \leftarrow u_{k+1/2,i} + \begin{cases} \frac{\omega}{a_{ii}} r_{u_k+1/2,i} & \text{if } i \in B \\ 0 & \text{otherwise} \end{cases}$$

It remains to choose the partition of the indices into the red and the black ones. Assume that we have a grid based problem, i.e., every index is assigned to a point in $G_{\mathbf{h}}$. For many applications the corrections corresponding to a point in the grid have a strong effect on the residuals corresponding to neighboring points. Thus, it makes sense to assign different colors to indices that correspond to neighboring grid points. We, therefore, define the set of red and the set of black grid points by

$$R_{G_{\mathbf{h}}} := \{h \cdot z : z \in \mathbb{Z}^d, z_1 + \cdots + z_d \text{ even}\} \quad \text{and} \tag{3.31a}$$
$$B_{G_{\mathbf{h}}} := \{h \cdot z : z \in \mathbb{Z}^d, z_1 + \cdots + z_d \text{ odd}\}\,. \tag{3.31b}$$

Then we can define the set of red/black *indices* as all indices that correspond to red/black *grid points*.

Let us analyze the red-black Jacobi method, starting with the derivation of its error propagator. To do so, let us define the two matrices

$$Z_{R,ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in R, \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad Z_{B,ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in B, \\ 0 & \text{otherwise.} \end{cases}$$

These matrices allow us to rewrite line 2-3 of RED-BLACK-ITERATION as

$$u_{k+1/2} = u_k + \omega Z_R D^{-1} u_k \tag{3.32a}$$

and line 4-5 as

$$u_{k+1} = u_{k+1/2} + \omega Z_B D^{-1} u_{k+1/2}\,, \tag{3.32b}$$

where D is the diagonal matrix containing the diagonal entries of A. Thus, the red-black Jacobi method consists of two successive error correction steps. Recall that one iteration of a stationary method is given by

$$u_{k+1} = u_k + B^{-1} r_{u_k}\,,$$

which was stated in (1.24). Comparing this equation to the first step (3.32a) of the red-black Jacobi method, yields that the first step corresponds to $B^{-1} = \omega Z_R D^{-1}$. The error propagator of a stationary method is given by (1.32), implying that the error propagator of the first step fulfills that

$$E_R = I - \omega Z_R D^{-1} A\,. \tag{3.33}$$

Similarly the second step has the error propagator

$$E_B = I - \omega Z_B D^{-1} A \,.$$

The two correction steps are applied successively. Consequently, the error propagator of the red-black Jacobi method is

$$E_{RB} = E_B E_R = (I - \omega Z_B D^{-1} A)(I - \omega Z_R D^{-1} A) \,. \tag{3.34}$$

Let us now make a local Fourier analysis of the red-black Jacobi method. For this purpose, we need to extend all operators involved in the computation of E_{RB} to the infinite grid $G_{\mathbf{h}}$. If we compare the error propagator (3.34) to the error propagator of the Jacobi method (1.33) we see that the only new component are the operators Z_R and Z_B. Thus, all we have to do is to extend Z_R and Z_B to the infinite grid $G_{\mathbf{h}}$ and give their Fourier matrix symbol.

On the infinite grid $G_{\mathbf{h}}$ the operator Z_R is given by the stencil s_R with

$$s_{R,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } y = 0 \text{ and } x \in R_{G_{\mathbf{h}}}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.35a}$$

The operator Z_B is given by the stencil s_B with

$$s_{B,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } y = 0 \text{ and } x \in B_{G_{\mathbf{h}}}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.35b}$$

By comparing with the definition of $R_{G_{\mathbf{h}}}$ and $B_{G_{\mathbf{h}}}$ (3.31), we see that the stencils s_R and s_B are periodic with period $(2 \cdot \mathbf{1})$.

3.6.2. Fourier Matrix Symbol

Now that we have an example of a method that can be written in terms of periodic stencils, we want to show that every periodic stencil operator has a Fourier matrix symbol and how to compute it.

Theorem 3.30. *Let $A \in L(\ell_2(G_h))$ be a periodic stencil operator with stencil $s_{\mathbf{x}}$ and period $\mathbf{n}$. Let $A^{(\mathbf{k})}$ be the (constant) stencil operator corresponding to the stencil $(s^{(\mathbf{k})})_{\mathbf{x}} := s_{\mathbf{k}\cdot\mathbf{h}}$ (for all $\mathbf{x} \in G_{\mathbf{h}}$). Then A has a Fourier matrix symbol $\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{h}\cdot\mathbf{n}})$ with*

$$\hat{a} = F^* \cdot g \,, \quad \textit{where} \quad g_{\mathbf{kj}}(\theta) = F_{\mathbf{kj}} \cdot \hat{a}^{(\mathbf{k})}(\vartheta + \mathbf{s_j}) \,,$$

and $F \in \mathbb{C}^{\mathbf{n}\times\mathbf{n}}$ is the Fourier matrix*, i.e.,*

$$F_{\mathbf{kj}} := \tfrac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} e^{\mathrm{i}2\pi\langle \mathbf{k}/\mathbf{n}, \mathbf{j}\rangle} \quad \textit{for} \quad \mathbf{k}, \mathbf{j} = \mathbf{0}, \ldots, \mathbf{n} - \mathbf{1} \,. \tag{3.36}$$

To prove Theorem 3.30, we need to make a few preparations, but let us first outline the idea of the proof. The idea is based on the following observation. We define the operator $T_{\mathbf{k}} : \ell_2(G_{\mathbf{h}}) \to \ell_2(G_{\mathbf{h}\cdot\mathbf{n}})$ by

$$(T_{\mathbf{k}} u)(\mathbf{x}) = u(\mathbf{x} + \mathbf{t_k}) \quad \text{for} \quad \mathbf{x} \in G_{\mathbf{h}\cdot\mathbf{n}} \,,$$

where $\mathbf{t_k}$ for $\mathbf{k} = \mathbf{0}, \dots, \mathbf{n-1}$ are the space shifts, defined by

$$\mathbf{t_k} := \mathbf{k} \cdot \mathbf{h} \quad \text{for} \quad \mathbf{k} = \mathbf{0}, \dots, \mathbf{n-1}. \tag{3.8 revisited}$$

If A is a periodic stencil operator, we have that

$$T_{\mathbf{k}} A = T_{\mathbf{k}} A^{(\mathbf{k})}, \tag{3.37}$$

where $A^{(\mathbf{k})}$ is the constant stencil operator given by $s_{\mathbf{t_k}}$. This equation implies that at the points $(\mathbf{t_k} + G_{\mathbf{h}\cdot\mathbf{n}})$ the result $f = Au$ will behave like it has been computed by a constant stencil operator. Now $T_{\mathbf{k}} Au$ is a grid function on $G_{\mathbf{h}\cdot\mathbf{n}}$. Thus, we can apply the DTFT to $T_{\mathbf{k}} Au$ to get $\hat{g}_{\mathbf{k}} := \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} T_{\mathbf{k}} Au$. We will show that due to observation (3.37) the frequency function $\hat{g}_{\mathbf{k}}$ can be computed from $\hat{u}$ (instead of u) by the knowledge of the symbol $\hat{a}^{(\mathbf{k})}$, which is easily obtained by (1.55) as $A^{(\mathbf{k})}$ is a constant stencil operator. The next step is to show that the functions $\hat{g}_{\mathbf{k}}$ can be combined to obtain $\hat{f} = R_{\mathbf{n}} \mathcal{F}_{\mathbf{h}} Au$. Thus, in the end we can compute $\hat{f}$ from $\hat{u}$.

In the following it will be helpful to combine the operators $T_{\mathbf{k}}$ into a single operator $T : \ell_2(G_{\mathbf{h}}) \to (\ell_2(G_{\mathbf{h}\cdot\mathbf{n}}))^{\mathbf{n}}$ by $(Tu)_{\mathbf{k}} = T_{\mathbf{k}} u$. The combined operator T will be called *space splitting* operator. On the space $(\ell_2(G_{\mathbf{h}\cdot\mathbf{n}}))^{\mathbf{n}}$ a scalar product can be defined by

$$\langle f, g \rangle := \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \langle f_{\mathbf{k}}, g_{\mathbf{k}} \rangle .$$

If we chose this scalar product, then T is an isometry between $\ell_2(G_{\mathbf{h}})$ and $(\ell_2(G_{\mathbf{h}\cdot\mathbf{n}}))^{\mathbf{n}}$.

Lemma 3.31. *Let $d \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$ be the diagonal matrix symbol with $d_{\mathbf{kj}}(\theta) = \delta_{\mathbf{kj}} \cdot e^{\mathrm{i}\langle\theta, \mathbf{t_k}\rangle}$. Furthermore, let $F \in \mathbb{C}^{\mathbf{n}\times\mathbf{n}}$ be the Fourier matrix* (3.36)*. We define the matrix symbol*

$$\hat{t} := d \cdot F .$$

Then the kth row $\hat{t}_{\mathbf{k},:}$ of $\hat{t}$ is the Fourier matrix symbol of $T_{\mathbf{k}}$. Furthermore

$$\widehat{T} := \dot{\mathcal{F}}_{\mathbf{n}\cdot\mathbf{h}} T \mathcal{F}_{\mathbf{h}}^{-1} \quad \textit{fulfills} \quad \widehat{T} = \hat{t} \mathcal{R}_{\mathbf{n}}, \tag{3.38}$$

where we define for $u \in \ell_2(G_{\mathbf{h}\cdot\mathbf{n}})^{\mathbf{n}}$ the element-wise DTFT $\dot{\mathcal{F}}_{\mathbf{h}\cdot\mathbf{n}} : \ell_2(G_{\mathbf{h}\cdot\mathbf{n}})^{\mathbf{n}} \to L_2(\Theta_{\mathbf{h}\cdot\mathbf{n}})^{\mathbf{n}}$ by the equation $(\dot{\mathcal{F}}_{\mathbf{h}\cdot\mathbf{n}} u)_{\mathbf{j}} := \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} u_{\mathbf{j}}$.

Proof. The operator $T_{\mathbf{0}}$ is the injection restriction operator R_{inj} from (3.1). Furthermore, if we define the shift operators $(Z_{\mathbf{k}} u)(\mathbf{x}) := u(\mathbf{x} + \mathbf{t_k})$ for $\mathbf{k} = \mathbf{0}, \dots, \mathbf{n-1}$, it is easy to see that $T_{\mathbf{k}} = R Z_{\mathbf{k}}$. The Fourier matrix symbol of the restriction is

$$\hat{r}_{\mathbf{0j}} = \tfrac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}}, \tag{3.19 revisited}$$

which is a $\mathbf{1}\times\mathbf{n}$-matrix symbol. The Fourier symbol of the shift operator (cf. Lemma 2.19) is

$$\hat{z}_{\mathbf{k}}(\theta) = e^{\mathrm{i}\langle\theta, \mathbf{t_k}\rangle} .$$

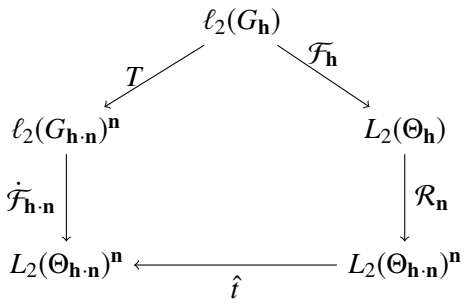


Figure 3.4.: Commuting diagram that relates $\dot{\mathcal{F}}_{\mathbf{h}} T$ and $\mathcal{R}_{\mathbf{n}} \mathcal{F}_{\mathbf{h}}$.

Theorem 3.22 shows that we can expand the symbol $\hat{z}_{\mathbf{k}}$ to a $\mathbf{n} \times \mathbf{n}$-matrix symbol $\operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}})$. Thus, we can multiply the last equation from the left by $\hat{r}$ to get the Fourier matrix symbol of $R_{\text{inj}} Z_{\mathbf{k}}$:

$$\hat{r} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}}) \,.$$

To prove the assertion of this lemma, we have to show that this term equals $\hat{t}_{\mathbf{k},:}$.

We compute the matrix product

$$(\hat{r} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}}))_{\mathbf{0j}} = \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \hat{r}_{\mathbf{0l}} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}})_{\mathbf{lj}} \,.$$

We have that $\operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}})_{\mathbf{lj}}(\theta) = \delta_{\mathbf{lj}} \hat{z}_{\mathbf{k}}(\theta + \mathbf{s}_{\mathbf{l}}) = \delta_{\mathbf{lj}} \cdot e^{\mathrm{i}\langle \theta + \mathbf{s}_{\mathbf{l}}, \mathbf{t}_{\mathbf{k}} \rangle}$, which we plug into th previous equation. It follows that

$$\begin{aligned}
(\hat{r} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{z}_{\mathbf{k}}))_{\mathbf{0j}}(\theta) &= \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \tfrac{1}{\operatorname{vol}_{\mathbf{n}}^{1/2}} \cdot \delta_{\mathbf{lj}} \cdot e^{\mathrm{i}\langle \theta + \mathbf{s}_{\mathbf{l}}, \mathbf{t}_{\mathbf{k}} \rangle} = \tfrac{1}{\operatorname{vol}_{\mathbf{n}}^{1/2}} \cdot e^{\mathrm{i}\langle \theta + \mathbf{s}_{\mathbf{j}}, \mathbf{t}_{\mathbf{k}} \rangle} \\
&= e^{\mathrm{i}\langle \theta, \mathbf{t}_{\mathbf{k}} \rangle} \cdot \tfrac{1}{\operatorname{vol}_{\mathbf{n}}^{1/2}} \cdot e^{\mathrm{i}\langle \mathbf{s}_{\mathbf{j}}, \mathbf{t}_{\mathbf{k}} \rangle} = e^{\mathrm{i}\langle \theta, \mathbf{t}_{\mathbf{k}} \rangle} \cdot F_{\mathbf{kj}} \\
&= \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \delta_{\mathbf{kl}} \cdot e^{\mathrm{i}\langle \theta, \mathbf{t}_{\mathbf{k}} \rangle} \cdot F_{\mathbf{lj}} = \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} d_{\mathbf{kl}} \cdot F_{\mathbf{lj}} = \hat{t}_{\mathbf{kj}} \,.
\end{aligned}$$

Thus, we have proven that the Fourier matrix symbol of $\hat{T}_{\mathbf{k}} = R_{\text{inj}} Z_{\mathbf{k}}$ is $\hat{t}_{\mathbf{k},:}$.

Let us turn to the operator T. It is defined by $(Tu)_{\mathbf{k}} = T_{\mathbf{k}} u$, implying that

$$(\dot{\mathcal{F}}_{\mathbf{n}\cdot\mathbf{h}} T \mathcal{F}_{\mathbf{h}} \hat{u})_{\mathbf{k}} = \mathcal{F}_{\mathbf{n}\cdot\mathbf{h}} T_{\mathbf{k}} \mathcal{F}_{\mathbf{h}}^{-1} \hat{u} = \widehat{T}_{\mathbf{k}} \hat{u} = \mathcal{R}_{\mathbf{1}}^{-1} \hat{t}_{\mathbf{k},:} \mathcal{R}_{\mathbf{n}} \hat{u} = \hat{t}_{\mathbf{k},:} \mathcal{R}_{\mathbf{n}} \hat{u} = (\hat{t} \mathcal{R}_{\mathbf{n}})_{\mathbf{k}} \hat{u} \,. \qquad \square$$

Equation (3.38) basically states that the diagram in Figure 3.4 commutes. From the diagram we see that T, $\mathcal{F}_{\mathbf{h}}$, $\mathcal{F}_{\mathbf{h}\cdot\mathbf{n}}$ and $R_{\mathbf{n}}$ are isometries, which implies that $\hat{t}$ is also an isometry. We can now prove the main result of this section.

Proof of Theorem 3.30. Let $u \in \ell_2(G_{\mathbf{h}})$. We begin by computing the vector $\hat{g} := \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} T A u$.

We use the definition of the element-wise DTFT and observation (3.37), i.e., the fact that a periodic stencil operator looks like a constant stencil operator when multiplied from the left by $T_{\mathbf{k}}$, to obtain

$$\hat{g}_{\mathbf{k}} := \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} T_{\mathbf{k}} A u = \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} T_{\mathbf{k}} A^{(\mathbf{k})} u \,.$$

Using Lemma 3.31 and then Theorem 3.22, we can express $\hat{g}$ in terms of $\hat{u} := \mathcal{F}_{\mathbf{h}} u$ by

$$\hat{g}_{\mathbf{k}} = \mathcal{F}_{\mathbf{h}\cdot\mathbf{n}} T_{\mathbf{k}} A^{(\mathbf{k})} \mathcal{F}_{\mathbf{h}}^{-1} \hat{u} = \widehat{T}_{\mathbf{k}} \widehat{A^{(\mathbf{k})}} \hat{u} \,.$$

The operator $\widehat{T}_{\mathbf{k}}$ has the $\mathbf{1} \times \mathbf{n}$ matrix symbol $\hat{t}_{\mathbf{k},:}$ and $\widehat{A^{(\mathbf{k})}}$ has the $\mathbf{n} \times \mathbf{n}$ matrix symbol $\operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})})$. Thus, we can write

$$\hat{g}_{\mathbf{k}} = \hat{t}_{\mathbf{k},:} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})}) \cdot (\mathcal{R}_{\mathbf{n}} \hat{u}) \,.$$

Recall that $\hat{t}_{\mathbf{k},:}$ is a row vector and $\operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})})_{\mathbf{lj}}(\theta) = \delta_{\mathbf{lj}} \hat{a}^{(\mathbf{k})}(\theta + \mathbf{s}_{\mathbf{l}})$ is a diagonal matrix. Hence, $\hat{t}_{\mathbf{k}} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})})$ is a row vector with

$$\begin{aligned}(\hat{t}_{\mathbf{k},:} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})}))_{\mathbf{0j}}(\theta) &= \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \hat{t}_{\mathbf{kl}} \cdot \delta_{\mathbf{lj}} \cdot \hat{a}^{(\mathbf{k})}(\theta + \mathbf{s}_{\mathbf{l}}) = \hat{t}_{\mathbf{kj}} \cdot \hat{a}^{(\mathbf{k})}(\theta + \mathbf{s}_{\mathbf{j}}) \\ &= d_{\mathbf{kk}} \cdot F_{\mathbf{kj}} \cdot \hat{a}^{(\mathbf{k})}(\theta + \mathbf{s}_{\mathbf{j}}) = d_{\mathbf{kk}} \cdot g_{\mathbf{kh}} \,.\end{aligned}$$

By using that d is diagonal, we have that

$$(\hat{t}_{\mathbf{k},:} \cdot \operatorname{expand}_{\mathbf{n}}(\hat{a}^{(\mathbf{k})}))_{\mathbf{0j}}(\theta) = \sum_{\mathbf{l}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} d_{\mathbf{kl}} \cdot g_{\mathbf{lj}} \,.$$

Thus,

$$\hat{g}_{\mathbf{k}} = (dg)_{\mathbf{k},:} \cdot (\mathcal{R}_{\mathbf{n}} \hat{u}),$$

and the whole vector $\hat{g}$ can be computed by $\hat{g} = dg \cdot (\mathcal{R}_{\mathbf{n}} \hat{u})$. By using that $\hat{g}$ was defined as $\hat{g} := \dot{\mathcal{F}}_{\mathbf{n}\cdot\mathbf{h}} T A \mathcal{F}_{\mathbf{h}}^{-1} \hat{u}$, we get

$$\dot{\mathcal{F}}_{\mathbf{n}\cdot\mathbf{h}} T A \mathcal{F}_{\mathbf{h}} = dg \mathcal{R}_{\mathbf{n}} \,.$$

Now we use Lemma 3.31 again (cf. Figure 3.4) to obtain that

$$\mathcal{R}_{\mathbf{n}} \mathcal{F}_{\mathbf{h}} A \mathcal{F}_{\mathbf{h}}^{-1} \mathcal{R}_{\mathbf{n}}^{-1} = (dF)^{-1} \dot{\mathcal{F}}_{\mathbf{n}\cdot\mathbf{h}} T A \mathcal{F}_{\mathbf{h}}^{-1} \mathcal{R}_{\mathbf{n}}^{-1} = (dF)^{-1} dg = F^{-1} d^{-1} dg = F^* g \,.$$

The last equality holds, as F is the DFT matrix and therefore unitary. Finally, the last equation yields that $F^* g$ is the Fourier matrix symbol of A. □

3.6.3. Symbol of the Red-Black Jacobi Method

We have seen that the red-black Jacobi method can be analyzed in terms of periodic stencils and that every periodic stencil operator has a Fourier matrix symbol. The formula for these symbols is given in Theorem 3.30. We shall combine these results to derive the Fourier matrix symbol of the error propagator of the red-black Jacobi method, providing us with a simple

example of local Fourier analysis of periodic stencil operators. For notational simplicity we shall restrict ourselves to the 2D case in this section. The calculation we present, however, is also applicable in arbitrary dimensions. Thus, Theorem 3.30 is a generalization of known results [47, 68, 74].

The first step is to compute the Fourier matrix symbol of the filtering operators Z_R and Z_B. The error propagator of the red-black Jacobi method consists of different operators. All of them except Z_R and Z_B are also present in the error propagator of the plain Jacobi method (see Section 3.6.1), which we already analyzed. Hence, to derive the Fourier matrix symbol of the error propagator of the red-black Jacobi method, it just remains to compute the Fourier matrix symbols of Z_R and Z_B.

For the derivation let us introduce a convenient notation for matrix symbols. The entry of a matrix symbol is referenced by two multi-indices if we provide an ordering of the multi-indices we can write the matrix symbols as an ordinary matrix. For this purpose, let $\mathbf{n} = (2,2)^T$, $\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_\mathbf{h})$, and the order of the indices be $(0,0)^T$, $(1,0)^T$, $(0,1)^T$, followed by $(1,1)^T$. Then we can write $\hat{a}$ as

$$\hat{a} = \begin{pmatrix} \hat{a}_{00,00} & \hat{a}_{00,10} & \hat{a}_{00,01} & \hat{a}_{00,11} \\ \hat{a}_{10,00} & \hat{a}_{10,10} & \hat{a}_{10,01} & \hat{a}_{10,11} \\ \hat{a}_{01,00} & \hat{a}_{01,10} & \hat{a}_{01,01} & \hat{a}_{01,11} \\ \hat{a}_{11,00} & \hat{a}_{11,10} & \hat{a}_{11,01} & \hat{a}_{11,11} \end{pmatrix}.$$

Using this notation, we can start computing the Fourier matrix symbol of the red-black Jacobi method.

We split the computation of the Fourier matrix symbol of the Z_R into several parts. We need to compute the matrix F and the matrix g (see Theorem 3.30) to obtain the matrix symbol by

$$\hat{a} = F^* \cdot g\,.$$

Let us start with the matrix F, which is defined as

$$F_{\mathbf{kj}} := \frac{1}{\mathrm{vol}_\mathbf{n}^{1/2}} e^{\mathrm{i}2\pi\langle \mathbf{k}/\mathbf{n},\mathbf{j}\rangle} \quad \text{for} \quad \mathbf{k},\mathbf{j} = \mathbf{0},\ldots,\mathbf{n}-\mathbf{1}. \tag{3.36 revisited}$$

To compute F we first compute the matrix B defined by $B_{\mathbf{kj}} := \langle \mathbf{k}/\mathbf{n},\mathbf{j}\rangle$. Recall that in the case of a 2D red-black Jacobi method $\mathbf{n} = (2,2)$. Thus

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 & 1 \end{pmatrix}.$$

Since we can write $F_{\mathbf{kj}} = \frac{1}{\mathrm{vol}_\mathbf{n}^{1/2}} e^{\mathrm{i}2\pi\langle \mathbf{k}/\mathbf{n},\mathbf{j}\rangle} = \frac{1}{2} e^{\mathrm{i}2\pi B_{\mathbf{kn}}}$, the previous equation yields that

$$F = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

We turn to the computation of the matrix symbol g. It is given by

$$g_{\mathbf{kj}}(\theta) = F_{\mathbf{kj}} \cdot \hat{a}^{(\mathbf{k})}(\theta + s_{\mathbf{j}}) .$$

Thus, if we define the matrix C by $C_{\mathbf{kj}} := \hat{a}^{(\mathbf{k})}(\theta + s_{\mathbf{j}})$ then g is the elementwise product of F and C. The symbols $\hat{a}^{(\mathbf{k})}$ for $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{n} - \mathbf{1}$ are determined by the periodic stencil which defines the filtering operator Z_R. The periodic stencil is the identity stencil on the red points and zero on the black points (3.35a). The symbol $\hat{a}^{(\mathbf{k})}$ is the Fourier symbol of the stencil operator that is applied at the grid point $\mathbf{t}_{\mathbf{k}}$. Thus, for the red points we have $\hat{a}^{(\mathbf{k})} = 1$ and for the black points $\hat{a}^{(\mathbf{k})} = 0$. The red points are $\mathbf{k} = (0, 0)^T$ and $\mathbf{k} = (1, 1)^T$ and the black points are $\mathbf{k} = (1, 0)^T$ and $\mathbf{k} = (0, 1)^T$. Combining these facts gives that

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} .$$

By multiplying F and C elementwise, we obtain

$$g = \tfrac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{pmatrix} .$$

Thus, we obtain the Fourier matrix symbol of Z_R by

$$\hat{z}_R = F^* \cdot g = \tfrac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} .$$

In the same way we can compute the Fourier matrix symbol of Z_B:

$$\hat{z}_B = \tfrac{1}{2} \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} .$$

Using the Fourier matrix symbols we can compute the Fourier matrix symbol of the error propagator of the red-black Jacobi method as a whole. The error propagator is a combination of a red and a black error propagator. The red error propagator is given by

$$E_R = I - \omega Z_R D^{-1} A . \qquad \text{(3.33 revisited)}$$

Let us assume that the operator A and D are both given by Fourier symbols $\hat{a}$ and $\hat{d}$. As $\hat{z}_R$ is a $\mathbf{n} \times \mathbf{n}$-matrix symbol, we need to expand $\hat{a}$ and $\hat{d}$ to be able to multiply them. Thus, the Fourier matrix symbol of E_R is

$$\hat{e}_R = 1 - \omega \hat{z}_R \cdot \mathrm{expand}_{\mathbf{n}}(\hat{d})^{-1} \cdot \mathrm{expand}_{\mathbf{n}}(\hat{a}) .$$

Let us denote $\hat{a}_{\mathbf{k}} := \hat{a}(\theta + \mathbf{s}_{\mathbf{k}})$ and $\hat{d}_{\mathbf{k}} := \hat{d}(\theta + \mathbf{s}_{\mathbf{k}})$. Then we can write

$$\text{expand}_{\mathbf{n}}(\hat{a}) = \begin{pmatrix} \hat{a}_{00} & & & \\ & \hat{a}_{10} & & \\ & & \hat{a}_{01} & \\ & & & \hat{a}_{11} \end{pmatrix} \quad \text{and} \quad \text{expand}_{\mathbf{n}}(\hat{d}) = \begin{pmatrix} \hat{d}_{00} & & & \\ & \hat{d}_{10} & & \\ & & \hat{d}_{01} & \\ & & & \hat{d}_{11} \end{pmatrix}.$$

Computing the matrix products we obtain

$$\begin{aligned} \hat{e}_R &= \tfrac{2}{1} \begin{pmatrix} -\omega\frac{a_{00}}{d_{00}} + 2 & 0 & 0 & -\omega\frac{a_{11}}{d_{11}} \\ 0 & -\omega\frac{a_{10}}{d_{10}} + 2 & -\omega\frac{a_{01}}{d_{01}} & 0 \\ 0 & -\omega\frac{a_{10}}{d_{10}} & -\omega\frac{a_{01}}{d_{01}} + 2 & 0 \\ -\omega\frac{a_{00}}{d_{00}} & 0 & 0 & -\omega\frac{a_{11}}{d_{11}} + 2 \end{pmatrix} \\ &= \tfrac{2}{1} \begin{pmatrix} (1 - \omega\frac{a_{00}}{d_{00}}) + 1 & 0 & 0 & (1 - \omega\frac{a_{11}}{d_{11}}) - 1 \\ 0 & (1 - \omega\frac{a_{10}}{d_{10}}) + 1 & (1 - \omega\frac{a_{01}}{d_{01}}) - 1 & 0 \\ 0 & (1 - \omega\frac{a_{10}}{d_{10}}) - 1 & (1 - \omega\frac{a_{01}}{d_{01}}) + 1 & 0 \\ (1 - \omega\frac{a_{00}}{d_{00}}) - 1 & 0 & 0 & (1 - \omega\frac{a_{11}}{d_{11}}) + 1 \end{pmatrix}. \end{aligned}$$

Note that $\hat{e} = 1 - \omega\hat{a}/\hat{d}$ is the symbol of the Jacobi method. Thus,

$$\hat{e}_R = \tfrac{2}{1} \begin{pmatrix} \hat{e}_{00} + 1 & 0 & 0 & \hat{e}_{11} - 1 \\ 0 & \hat{e}_{10} + 1 & \hat{e}_{01} - 1 & 0 \\ 0 & \hat{e}_{10} - 1 & \hat{e}_{01} + 1 & 0 \\ \hat{e}_{00} - 1 & 0 & 0 & \hat{e}_{11} + 1 \end{pmatrix}.$$

Similarly, we obtain the Fourier matrix symbol of E_B; we have that

$$\hat{e}_B = \tfrac{2}{1} \begin{pmatrix} \hat{e}_{00} + 1 & 0 & 0 & -\hat{e}_{11} + 1 \\ 0 & \hat{e}_{10} + 1 & -\hat{e}_{01} + 1 & 0 \\ 0 & -\hat{e}_{10} + 1 & \hat{e}_{01} + 1 & 0 \\ -\hat{e}_{00} + 1 & 0 & 0 & \hat{e}_{11} + 1 \end{pmatrix}.$$

The Fourier matrix symbol of $E_{RB} = E_B E_R$ is then easily computed by $\hat{e}_{RB} = \hat{e}_B \hat{e}_R$.

The Fourier matrix symbol of the Red-Black Jacobi method is well known for 1D, 2D, and 3D [63, 68, 74]. The reason we presented it here is to show that the concept of periodic stencils and their Fourier symbols can be used to derive these known result in a unified fashion. The 1D, 3D, and dD case can be obtained in a similar way. It is even possible to compute the Fourier matrix symbol of multi-coloring relaxation schemes. All one needs to do is to define a filtering matrix Z_c for every color c and compute its Fourier matrix symbol, which just requires the computation of the formulas in Theorem 3.30. Then all that is left to do is to multiply the individual matrix symbols.

3.7. Block Shift Invariance

It is sometimes useful to know whether an operator has a periodic stencil without actually computing its stencil. For constant stencil operators we can check if an operator is shift

invariant to see whether it has a constant stencil representation (Theorem 2.38). We shall see that we can use *block shift invariance* for the case of periodic stencils. Block shift invariant operators are defined as follows.

Definition 3.32. An operator $A \in L(\ell_2(G_{\mathbf{h}}))$ is *block shift invariant* with block size $\mathbf{n} \in \mathbb{Z}^d$ if

$$S_{\mathbf{z}} A = S_{\mathbf{z}} A \quad \text{for all} \quad \mathbf{z} \in G_{\mathbf{n}\cdot\mathbf{h}},$$

where $S_{\mathbf{z}}$ is the shift operator (2.6).

Note that the difference between shift and block shift invariant operators is subtle. Comparing this definition to the definition to the definition of shift invariance (2.6), we see that in the definition of shift invariance we require that $S_{\mathbf{z}} A = A S_{\mathbf{z}}$ for all $\mathbf{z} \in G_{\mathbf{h}}$ while in the definition of block shift invariance we restrict $\mathbf{z}$ to the coarser grid $G_{\mathbf{n}\cdot\mathbf{h}}$. Thus, only shifts by a multiple of $\mathbf{n}$ are considered.

Using this definition we can generalize a result from the non-block case. We know that operators that have Fourier symbols are shift invariant (Theorem 2.38). A similar statement is true for operators that have Fourier matrix symbols.

Theorem 3.33. *Let $A \in L(\ell_2(G_{\mathbf{h}}))$ with Fourier matrix symbol $\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}})$. Then A is block shift invariant with block size $\mathbf{n}$.*

Proof. Similarly to Proposition 2.26, it is sufficient to show that the Fourier representations fulfill

$$\widehat{S}_{\mathbf{z}} \widehat{A} = \widehat{A} \widehat{S}_{\mathbf{z}} \quad \text{for all} \quad \mathbf{z} \in G_{\mathbf{n}\cdot\mathbf{z}}\,.$$

Recall

$$\widehat{S}_{\mathbf{z}} \hat{f} = \left\{ \theta \mapsto e^{-\mathrm{i}\langle\theta,\mathbf{z}\rangle} \cdot \hat{f}(\theta) \right\}. \tag{2.17 revisited}$$

We use this definition and the definition of the matrix multiplication operator to obtain that

$$\begin{aligned}
[\widehat{A}\widehat{S}_{\mathbf{z}}\hat{u}](\theta) &= [\widehat{A}\widehat{S}_{\mathbf{z}}\hat{u}](\theta_b + \mathbf{s_k}) \\
&= \sum_{\mathbf{j}=1}^{\mathbf{n}} \hat{a}_{\mathbf{kj}} \cdot [\widehat{S}_{\mathbf{z}}\hat{u}](\theta_b + \mathbf{s_j}) = \sum_{\mathbf{j}=1}^{\mathbf{n}} \hat{a}_{\mathbf{kj}} \cdot e^{-\mathrm{i}\langle\theta_b+\mathbf{s_j},\mathbf{z}\rangle} \cdot \hat{u}(\theta_b + \mathbf{s_j})\,.
\end{aligned}$$

The frequency θ_b is an $\mathbf{n}$,$\mathbf{h}$-harmonic of $(\theta_b + \mathbf{s_j})$ and $\mathbf{z} \in G_{\mathbf{n}\cdot\mathbf{h}}$ thus $e^{-\mathrm{i}\langle\theta_b+\mathbf{s_j},\mathbf{z}\rangle} = e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle}$, and therefore

$$\begin{aligned}
[\widehat{A}\widehat{S}_{\mathbf{z}}\hat{u}](\theta) &= e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle} \cdot \sum_{\mathbf{j}=\mathbf{1}}^{\mathbf{n}} \hat{a}_{\mathbf{kj}} \cdot \hat{u}(\theta_b + \mathbf{s_j}) \\
&= e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle} \cdot [\widehat{A}\hat{u}](\theta_b + \mathbf{s_j})\,.
\end{aligned}$$

On the other hand, let us compute that

$$\begin{aligned}
[\widehat{S}_{\mathbf{z}}\widehat{A}\hat{u}](\theta) &= [\widehat{S}_{\mathbf{z}}\widehat{A}\hat{u}](\theta_b + \mathbf{s_k}) \\
&= e^{-\mathrm{i}\langle\theta_b+\mathbf{s_k},\mathbf{z}\rangle} \cdot [\widehat{A}\hat{u}](\theta_b + \mathbf{s_k})\,.
\end{aligned}$$

Again, the frequency θ_b is an $\mathbf{n}$,$\mathbf{h}$-harmonic of $(\theta_b + \mathbf{s_j})$ and $\mathbf{z} \in G_{\mathbf{n}\cdot\mathbf{h}}$, which implies that $e^{-\mathrm{i}\langle\theta_b+\mathbf{s_j},\mathbf{z}\rangle} = e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle}$. Therefore

$$[\widehat{S}_{\mathbf{z}}\widehat{A}\hat{u}](\theta) = e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle} \cdot [\widehat{A}\hat{u}](\theta_b)\,.$$

Thus, we have shown that

$$[\widehat{S}_{\mathbf{z}}\widehat{A}\hat{u}](\theta) = e^{-\mathrm{i}\langle\theta_b,\mathbf{z}\rangle} \cdot [\widehat{A}\hat{u}](\theta_b) = [\widehat{A}\widehat{S}_{\mathbf{z}}\hat{u}](\theta)\,. \qquad \square$$

Another generalization of the non-block case is the following relation between block shift invariant and periodic stencil operators.

Theorem 3.34. *Let A be a block shift invariant operator with block size $\mathbf{n}$. Then A is a periodic stencil operator with block size $\mathbf{n}$.*

Proof. Let $e_{\mathbf{z}}$ be the canonical basis vector $e_{\mathbf{z}}(\mathbf{x}) = \delta_{\mathbf{xy}}$. Recall that then

$$[Ae_{\mathbf{z}}](\mathbf{x}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s_{\mathbf{x}}(\mathbf{y}) \cdot e_{\mathbf{z}}(\mathbf{x}+\mathbf{y}) = s_{\mathbf{x}}(\mathbf{z}-\mathbf{x}). \qquad \text{(2.7 revisited)}$$

Let $\mathbf{v} \in G_{\mathbf{n}\cdot\mathbf{h}}$ and $\mathbf{w}, \mathbf{u} \in G_{\mathbf{h}}$. Then

$$[AS_{-\mathbf{v}}e_{\mathbf{v}+\mathbf{w}+\mathbf{u}}](\mathbf{u}) = [Ae_{\mathbf{w}+\mathbf{u}}](\mathbf{u}) = s_{\mathbf{u}}(\mathbf{w})\,.$$

As A is block shift invariant, this equation implies that

$$[S_{-\mathbf{v}}Ae_{\mathbf{v}+\mathbf{w}+\mathbf{u}}](\mathbf{u}) = [Ae_{\mathbf{v}+\mathbf{w}+\mathbf{u}}](\mathbf{u}+\mathbf{v}) = s_{\mathbf{u}+\mathbf{v}}(\mathbf{w})\,.$$

Thus,

$$s_{\mathbf{u}}(\mathbf{w}) = s_{\mathbf{u}+\mathbf{v}}(\mathbf{w}) \quad \text{for all} \quad \mathbf{v} \in G_{\mathbf{n}\cdot\mathbf{h}} \text{ and } \mathbf{w}, \mathbf{u} \in G_{\mathbf{h}}. \qquad \square$$

We can now combine the fact that every periodic stencil operator has a Fourier matrix symbol (Theorem 3.30) with further results from this chapter.

Theorem 3.35. *Let $A \in L(\ell_2(G_{\mathbf{h}}))$. Then the following statements are equivalent:*

1. *The operator A is a periodic stencil operator with period $\mathbf{n}$.*
2. *The operator A has a Fourier $\mathbf{n}\times\mathbf{n}$-matrix symbol.*
3. *The operator A is block shift invariant with block size $\mathbf{n}$.*

Proof. Theorem 3.30 yields that statement 1 implies 2. Theorem 3.33 states that statement 2 implies 3. Furthermore, Theorem 3.34 yields that statement 3 implies 1. $\square$

3.8. Expansion

We shall show that the product of two operators that have Fourier matrix symbols always has a Fourier matrix symbol. Let us consider the operators $A, B \in L(\ell_2(G_{\mathbf{h}}))$ with Fourier matrix symbols

$$\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n}}(\Theta_{\mathbf{h}}) \quad \text{and} \quad \hat{b} \in L_\infty^{\mathbf{m}\times\mathbf{m}}(\Theta_{\mathbf{h}}),$$

where $\mathbf{n} \neq \mathbf{m}$. We cannot obtain the Fourier symbol of AB by multiplying $\hat{a}$ and $\hat{b}$, because their dimensions do not match.

There must be, however, a Fourier matrix symbol of AB for the following reason. We know from Theorem 3.35 that A has a periodic stencil s with period $\mathbf{n}$, meaning that

$$s_{\mathbf{x}} = a_{\mathbf{x}+\mathbf{y}} \quad \text{for all} \quad \mathbf{y} \in G_{\mathbf{n}\cdot\mathbf{h}}.$$

We have for every $0 < \mathbf{p} \in \mathbb{Z}^d$ that $G_{\mathbf{p}\cdot\mathbf{n}\cdot\mathbf{h}} \subseteq G_{\mathbf{h}\cdot\mathbf{n}}$. Thus,

$$s_{\mathbf{x}} = a_{\mathbf{x}+\mathbf{y}} \quad \text{for all} \quad \mathbf{y} \in G_{\mathbf{p}\cdot\mathbf{n}\cdot\mathbf{h}}.$$

In other words, A is also represented by a periodic stencil with period $\mathbf{p} \cdot \mathbf{n}$, and thus by Theorem 3.35 we have that A has a Fourier matrix symbol

$$\acute{a} \in L_\infty^{\mathbf{p}\cdot\mathbf{n}\times\mathbf{p}\cdot\mathbf{n}}(\Theta_{\mathbf{h}}).$$

Analogously, for every $0 < \mathbf{q} \in \mathbb{Z}^d$ the operator B has a Fourier matrix symbol

$$\acute{b} \in L_\infty^{\mathbf{q}\cdot\mathbf{m}\times\mathbf{q}\cdot\mathbf{m}}(\Theta_{\mathbf{h}}).$$

By computing the (pointwise) least common multiple of $\mathbf{m}$ and $\mathbf{n}$, we find $\mathbf{p}$ and $\mathbf{q}$ such that $\mathbf{p} \cdot \mathbf{n} = \mathbf{q} \cdot \mathbf{m}$. In this case the Fourier matrix symbol of AB can be computed by the product $\acute{a}\acute{b}$.

We want to know how to compute $\acute{a}$ and $\acute{b}$ from $\hat{a}$ and $\hat{b}$. We start by showing a connection between the frequency splittings $\mathcal{R}_{\mathbf{n}}$ and $\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}$ that we shall need later.

Lemma 3.36. *Let* $0 < \mathbf{n}, \mathbf{p} \in \mathbb{Z}^d$ *be given and* $\mathbf{j} \in \mathbb{Z}^d$ *be related with* $\mathbf{k} \in \mathbb{Z}^d$ *and* $\mathbf{r} \in \mathbb{Z}^d$ *by*

$$\mathbf{j} = \mathbf{k} \cdot \mathbf{p} + \mathbf{r} \quad \textit{with} \quad 0 \leq \mathbf{r} < \mathbf{p}\,.$$

Then the frequency splitting operator defined in (3.3) *fulfills*

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\hat{u}]_{\mathbf{j}} = \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}}.$$

Proof. From the definition of the frequency splitting operator (3.3), we have that

$$\begin{aligned}
[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\hat{u}]_{\mathbf{j}}(\theta) &= \hat{u}(\theta + \mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{p}\cdot\mathbf{n}}) \\
&= \hat{u}\big(\theta + 2\pi\mathbf{j}/(\mathbf{p}\cdot\mathbf{n}\cdot\mathbf{h})\big) \\
&= \hat{u}\big(\theta + 2\pi(\mathbf{k}\cdot\mathbf{p} + \mathbf{r})/(\mathbf{p}\cdot\mathbf{n}\cdot\mathbf{h})\big) \\
&= \hat{u}\big(\theta + 2\pi\mathbf{k}/(\mathbf{n}\cdot\mathbf{h}) + 2\pi\mathbf{r}/(\mathbf{p}\cdot\mathbf{n}\cdot\mathbf{h})\big) \\
&= \hat{u}\big(\theta + \mathbf{s}_{\mathbf{k}}^{\mathbf{h},\mathbf{n}} + \mathbf{s}_{\mathbf{r}}^{\mathbf{h}\cdot\mathbf{n},\mathbf{p}}\big) \\
&= [\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{k}}\big(\theta + \mathbf{s}_{\mathbf{r}}^{\mathbf{h}\cdot\mathbf{n},\mathbf{p}}\big) \\
&= \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}}(\theta).
\end{aligned}$$

□

We can now use Lemma 3.36, to extend the Fourier matrix symbol of an operator to larger matrix dimensions.

Theorem 3.37. *Let A be an operator with Fourier matrix symbol $\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n}'}(\Theta_{\mathbf{h}})$ and $0 < \mathbf{p} \in \mathbb{Z}^d$. Then $\acute{a} \in L_\infty^{\mathbf{p}\cdot\mathbf{n}\times\mathbf{p}\cdot\mathbf{n}'}(\Theta_{\mathbf{p}\cdot\mathbf{h}})$, given by*

$$\acute{a}_{\mathbf{j}\mathbf{j}'} = \begin{cases} [\mathcal{R}_{\mathbf{p}}\hat{a}_{\mathbf{k}\mathbf{k}'}]_{\mathbf{r}} & \text{if } \mathbf{r} = \mathbf{r}', \\ 0 & \text{otherwise,} \end{cases}$$

is also a Fourier matrix symbol of A, where $\mathbf{j}$, $\mathbf{j}'$, $\mathbf{k}$, $\mathbf{k}'$, $\mathbf{r}$, and $\mathbf{r}' \in \mathbb{Z}^d$ are related via

$$\begin{aligned} \mathbf{j} &= \mathbf{k}\cdot\mathbf{p} + \mathbf{r} \quad \text{with} \quad \mathbf{0} \le \mathbf{r} < \mathbf{p} \\ \mathbf{j}' &= \mathbf{k}'\cdot\mathbf{p} + \mathbf{r}' \quad \text{with} \quad \mathbf{0} \le \mathbf{r}' < \mathbf{p}\,. \end{aligned}$$

Proof. From the definition of Fourier matrix symbols (3.16) it follows that it is sufficient to show

$$\mathcal{R}_{\mathbf{q}\cdot\mathbf{n}}\widehat{A} = \acute{a}\mathcal{R}_{\mathbf{q}\cdot\mathbf{n}'}\,.$$

Thus, we start by computing $[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}}$. Lemma 3.36 yields

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}} = \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}}.$$

Using the fact that $\widehat{A}$ has the Fourier matrix symbol $\hat{a}$, we get that

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}} = \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}}\mathcal{R}_{\mathbf{n}}^{-1}\hat{a}\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}} = \left[\mathcal{R}_{\mathbf{p}}[\hat{a}\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}}. \tag{3.39}$$

We inspect the term on the right; by definition of the matrix-vector-product (3.11), we have

$$[\hat{a}\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}} = \sum_{\mathbf{k}'=\mathbf{0}}^{\mathbf{n}'-\mathbf{1}} \hat{a}_{\mathbf{k}\mathbf{k}'} \cdot [\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}'}\,.$$

Note that $\mathcal{R}$ is linear and in general $\mathcal{R}_{\mathbf{p}}(\hat{f}\cdot\hat{g}) = [\mathcal{R}_{\mathbf{p}}\hat{f}]\cdot[\mathcal{R}_{\mathbf{p}}\hat{g}]$, which is easy to see from the definition of $\mathcal{R}_{\mathbf{p}}$. Thus,

$$\left[\mathcal{R}_{\mathbf{p}}[\hat{a}\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}}\right]_{\mathbf{r}} = \sum_{\mathbf{k}'=\mathbf{0}}^{\mathbf{n}'-\mathbf{1}} [\mathcal{R}_{\mathbf{p}}\hat{a}_{\mathbf{k}\mathbf{k}'}]_{\mathbf{r}} \cdot \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}'}\right]_{\mathbf{r}}.$$

If we plug this relation into (3.39), we get

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}} = \sum_{\mathbf{k}'=\mathbf{0}}^{\mathbf{n}'-\mathbf{1}} [\mathcal{R}_{\mathbf{p}}\hat{a}_{\mathbf{k}\mathbf{k}'}]_{\mathbf{r}} \cdot \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}'}\right]_{\mathbf{r}}$$

To rewrite the right hand side into a $\mathbf{p}\cdot\mathbf{n}\times\mathbf{p}\cdot\mathbf{n}'$-matrix-vector-product, we need to sum over $\mathbf{j}' = \mathbf{0}, \ldots, \mathbf{p}\cdot\mathbf{n}'$ instead of $\mathbf{k}'$. As we defined $\acute{a}_{\mathbf{j}\mathbf{j}'}$ to be zero for $\mathbf{r} \neq \mathbf{r}'$, we can write

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}} = \sum_{\mathbf{j}'=\mathbf{0}}^{\mathbf{p}\cdot\mathbf{n}'-\mathbf{1}} \acute{a}_{\mathbf{j}\mathbf{j}'} \cdot \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}'}\hat{u}]_{\mathbf{k}'}\right]_{\mathbf{r}}.$$

Again, as $\acute{a}_{\mathbf{jj'}}$ is zero for $\mathbf{r} \neq \mathbf{r'}$, we can replace $\mathbf{r}$ by $\mathbf{r'}$ to get

$$[\mathcal{R}_{\mathbf{p}\cdot\mathbf{n}}\widehat{A}\hat{u}]_{\mathbf{j}} = \sum_{\mathbf{j'}=\mathbf{0}}^{\mathbf{p}\cdot\mathbf{n'}-\mathbf{1}} \acute{a}_{\mathbf{jj'}} \cdot \left[\mathcal{R}_{\mathbf{p}}[\mathcal{R}_{\mathbf{n}}\hat{u}]_{\mathbf{k'}}\right]_{\mathbf{r'}} = \sum_{\mathbf{j'}=\mathbf{0}}^{\mathbf{p}\cdot\mathbf{n'}-\mathbf{1}} \acute{a}_{\mathbf{jj'}} \cdot [\mathcal{R}_{\mathbf{p}\cdot\mathbf{n'}}\hat{u}]_{\mathbf{j'}} \,. \qquad \square$$

In our previous discussion we assumed that the Fourier matrix symbols of A and B were square. This is, however, not required. Assume that the operator $A \in L(\ell_2(G_{\mathbf{h''}}); \ell_2(G_{\mathbf{h'}}))$ and $B \in L(\ell_2(G_{\mathbf{h'}}); \ell_2(G_{\mathbf{h}}))$ with Fourier matrix symbols $\hat{a} \in L_\infty^{\mathbf{n}\times\mathbf{n'}}(\Theta_{\mathbf{n'}\cdot\mathbf{h'}})$ and $\hat{b} \in L_\infty^{\mathbf{m}\times\mathbf{m'}}(\Theta_{\mathbf{m}\cdot\mathbf{h'}})$. Then we can find $\mathbf{0} < \mathbf{p}, \mathbf{q} \in \mathbb{Z}^d$ such that $\mathbf{p}\cdot\mathbf{n'} = \mathbf{q}\cdot\mathbf{m}$. According to Theorem 3.37 there exists Fourier matrix symbols $\acute{a} \in L_\infty^{\mathbf{pn}\times\mathbf{pn'}}(\Theta_{\mathbf{p}\cdot\mathbf{n'}\cdot\mathbf{h'}})$ and $\acute{b} \in L_\infty^{\mathbf{qm}\times\mathbf{qm'}}(\Theta_{\mathbf{q}\cdot\mathbf{m}\cdot\mathbf{h'}})$ of A and B. Thus, the Fourier matrix symbol of AB is given by $\acute{a}\cdot\acute{b}$.

This is an important result. It means that the set of operators that have Fourier matrix symbols is closed under multiplication. Thus, whenever we multiply two operators that have Fourier matrix symbols we are sure that the result will also have a Fourier matrix symbol.

In a similar way, we can show that the set of Fourier matrix symbols is closed under addition. Let $A \in L(\ell_2(G_{\mathbf{h'}}); \ell_2(G_{\mathbf{h}})$ and $B \in L(\ell_2(G_{\mathbf{h'}}); \ell_2(G_{\mathbf{h}}))$ be operators with Fourier matrix symbols. Then the operator $A + B$ also has a Fourier matrix symbol.

3.9. Smoothing Factor

If the error propagator S of a smoother has an ordinary Fourier symbol $\hat{s}$ then the smoothing factor, as discussed in Section 2.5, is given by

$$\text{smf}(S, \mathbf{n}) := r(S^{\nu_2}Q_{\mathbf{n}}S^{\nu_1}) = r(\widehat{S}^{\nu_2}\widehat{Q}_{\mathbf{n}}\widehat{S}^{\nu_1}) = \|\hat{s}^{\nu_2}\cdot\hat{q}_{\mathbf{n}}\cdot\hat{s}^{\nu_1}\|_\infty \,. \qquad \text{(2.20 revisited)}$$

When $\hat{s}$ is not a Fourier symbol but a Fourier *matrix* symbol of S we have two problems. First, the we cannot multiply $\hat{s}$ and $\hat{q}_{\mathbf{n}}$, as their dimensions do not match. Second, the right most equality does not hold anymore.

The first problem can be addressed by expanding the operators $\hat{s}$ and $\hat{q}_{\mathbf{n}}$ to $\acute{s}$ and $\acute{q}_{\mathbf{n}}$ such that their dimensions match. Then the product

$$\hat{e} := \acute{s}^{\nu_2}\cdot\acute{q}_{\mathbf{n}}\cdot\acute{s}^{\nu_1}$$

is defined and is the Fourier matrix symbol of $S^{\nu_2}Q_{\mathbf{n}}S^{\nu_1}$.

The second problem can be addressed by using Theorem 3.26. It gives a way to compute the spectral radius from the Fourier matrix symbol of an operator. Thus, if the smoother is represented by a Fourier matrix symbol, we can define the smoothing factor as

$$\text{smf}(S, \mathbf{n}) := r(S^{\nu_2}Q_{\mathbf{n}}S^{\nu_1}) = \text{ess-sup}_{\theta\in\Theta_{\mathbf{n}\cdot\mathbf{h}}}\, r(\acute{s}^{\nu_2}(\theta)\cdot\acute{q}_{\mathbf{n}}(\theta)\cdot\acute{s}^{\nu_1}(\theta)) \,. \qquad (3.40)$$

3.10. Three- and n-Grid Analysis

To this point we only considered the analysis of the two-grid method. Using the expansion of Fourier matrix symbols, we can compute the Fourier matrix symbol of any sum or

composition of operators if the operators themselves have Fourier matrix symbols. With this ability, we can easily analyze a multigrid method consisting of any desired number of levels.

Theorem 1.10 states that the error propagator of a multigrid method with L levels is given recursively by

$$\begin{aligned} E_i &= S_i^{\nu_2}(I - P(I - E_{i+1}^{\eta})A_{i+1}^{-1}RA_i)S_i^{\nu_1} && \text{for } i < L, \\ E_i &= 0 && \text{for } i = L. \end{aligned} \tag{1.37 revisited}$$

Thus, we can compute the Fourier matrix symbol of E_1 by writing down the Fourier matrix symbol of all involved operators and then, after a proper expansion of the symbols, they can be added and multiplied as required by the formula. Therefore, we can perform an analysis of multigrid methods using an arbitrary number of grids.

3.11. Literature and Contributions

We started this chapter with a discussion about the Fourier representation of the injection restriction and interpolation, followed by an introduction to (Fourier) matrix symbols. These results are well known [11, 63, 68, 74].

After this discussion, we considered the interpretation and visualization of matrix symbols (Section 3.4), which has not been examined in the literature so far.

This discussion was followed by the description of the spectral properties of matrix multiplication operators. The formulas for the norm and the spectrum of these operators have been known for some time (cf. [8] and [39], respectively). The formula for the spectral radius, however, (Lemma 3.28) has been missing from the literature so far.

These results were followed by the discussion of the Fourier matrix symbols of periodic stencil operators, a theorem about block shift invariance, and the description of the general expansion of operators with Fourier matrix symbols. These results are original to this thesis.

With the introduction of periodic stencils and their Fourier analysis we were able to widen the applicability of LFA. There are, of course, still problems where LFA has not been applied successfully, which can be found, e.g., in [23–25, 52].

4. Applications

Fourier matrix symbols allow us to represent periodic stencil operators and therefore analyze problems that can be represented on an infinite grid by periodic stencil operators. In this chapter we analyze two advanced problems—a PDE with jumping coefficient, and block smoothers.

4.1. PDEs with Jumping Coefficients

Consider the PDE

$$-\nabla \cdot (a \nabla u) = f \quad \text{on } \Omega \tag{4.1a}$$

$$u = g \quad \text{on } \partial\Omega\,. \tag{4.1b}$$

In this equation $a : \Omega \to \mathbb{R}$ and $a > 0$. Furthermore, ∇u for $u : \Omega \to \mathbb{R}$ is the *gradient* of u, defined by

$$\nabla u := (\tfrac{\partial u}{\partial x_1}, \dots, \tfrac{\partial u}{\partial x_d})^T\,,$$

and $\nabla \cdot w$ for $w : \Omega \to \mathbb{R}^d$ is the *divergence* of w, defined by

$$\nabla \cdot w := \tfrac{\partial w_1}{\partial x_1} + \cdots + \tfrac{\partial w_d}{\partial x_d}\,.$$

We assume that the domain Ω is partitioned into several sub-domains, and a is constant on every of these sub-domains. The function a, however, is allowed to vary by several orders of magnitude between different sub-domains, e.g., as depicted in Figure 4.1.

This PDE models, e.g., the stationary temperature distribution u in some material that is heated or cooled. The heating or cooling is described by the function f while the temperature at the boundary is held fixed at the value described by g. The coefficient a then is the *thermal*

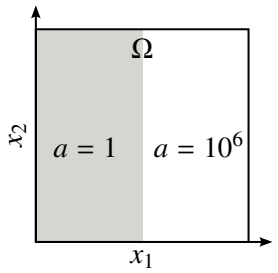


Figure 4.1.: The domain Ω is split into two sub-domains where the value of a is different.

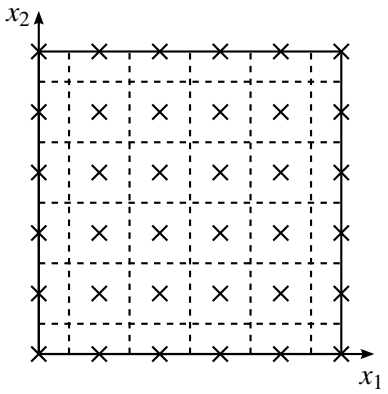


Figure 4.2.: The domain Ω split into control volumes Ω_i. The crosses mark the degree of freedom of the control volume.

conductivity [78]. When a is large at position x, then the material conducts heat very well at this point, i.e., heat is transferred fast through the material. If a is small, then the material insulates heat well, i.e., the heat is transferred slowly. The value of a jumps by several orders of magnitude, e.g., when the domain Ω is filled with different materials that have different thermal conductivities.

The jump in the coefficient a causes problems when discretizing the PDE (4.1); it is well known that the solution ∇u will not be continuous in general. Thus, approximating the derivatives by ordinary finite differences leads to large errors. In this case the *finite volume method* [44, 68, 73] is a better approximation.

4.1.1. Finite Volume Method

The finite volume method can be applied to a wide variety of domains. However, since the principles of the method can be well explained by deriving the method for a specific domain, let us for simplicity assume that $\Omega \subseteq \mathbb{R}^d$ and that Ω is a rectangle. The main idea of the finite volume method is to partition the domain Ω into small (non-overlapping) sub-domains Ω_i—the control volumes. Again, for simplicity assume that the control volumes Ω_i are rectangles as depicted in Figure 4.2 and that the coefficient a is constant within a control volume. We assign a degree of freedom to every control volume Ω_i. In our case, if the control volume lies in the interior of the domain we choose the function value $u(x_i)$ at the center x_i of the control volume; if the control volume lies at the boundary of the domain we locate x_i at the boundary. The degrees of freedom are marked by crosses in Figure 4.2. These are the positions where we want to compute the function values of u.

To determine $u(x_i)$ for $i = 1, \dots, n$ we need a set of n equations—one equation for every degree of freedom, i.e., for every control volume Ω_i. We obtain n equations by integrating both sides of the PDE (4.1a) over Ω_i for $i = 1, \dots, n$. Thus, we demand that

$$-\int_{\Omega_i} \nabla \cdot (a \nabla u) \, \mathrm{d}V = \int_{\Omega_i} f \, \mathrm{d}V \quad \text{for} \quad i = 1, \dots, n \,. \tag{4.2}$$

This set of equations still involves all function values of u. Since, we want to compute u *only* at the points x_i, we need to approximate the integrals in a way that involves the function values of u only at the points x_i. Furthermore, we need to approximate the integral on the right hand side, as its exact value will be usually not available.

The integral on the right hand side of (4.2) is readily approximated by the midpoint rule. Thus, if we denote h_1 as the width and h_2 as the height of the control volume then

$$\int_{\Omega_i} f \,\mathrm{d}V \approx h_1 h_2 \cdot f(x_{i,1}, x_{i,2})\,. \tag{4.3}$$

The first step is to relate neighboring control volumes by using Gauss' divergence theorem [4, 20, 43], which states that for a vector field $F : \Omega \to \mathbb{R}$ we have that

$$\int_{\Omega} \nabla \cdot F \,\mathrm{d}V = \int_{\partial\Omega} \langle F, n\rangle \,\mathrm{d}S\,,$$

where n is the outward-pointing unit normal of Ω. Applying this equation to the left hand side of (4.2) yields that

$$-\int_{\Omega_i} \nabla \cdot (a\nabla u)\,\mathrm{d}V = -\int_{\partial\Omega_i} \langle a\nabla u, n\rangle \,\mathrm{d}S\,.$$

In our case the control volumes Ω_i are rectangles with sides $\mathcal{E}_i = \{e_{i,\mathrm{n}}, e_{i,\mathrm{w}}, e_{i,\mathrm{s}}, e_{i,\mathrm{e}}\}$, which are shown in Figure 4.3. Thus, the integral over the boundary of Ω_i becomes a sum over the integral over the edges, i.e.,

$$-\int_{\Omega_i} \nabla \cdot (a\nabla u)\,\mathrm{d}V = -\sum_{e\in\mathcal{E}_i} \int_e \langle a\nabla u, n\rangle \,\mathrm{d}S\,. \tag{4.4}$$

Consequently, we are left with the approximation of the integrals over the edges.

Let us approximate the integrals over the edges. We start with the *east* edge $e_{i,\mathrm{e}}$ of the ith control volume Ω_i. In the following we focus only on the edge of one control volume. We, therefore, suppress the index i, i.e., x is the center of the control volume we consider and e_e is its east edge. First of all, since the normal of the edge is pointing in the x_1-direction,

$$\int_{e_\mathrm{e}} (a\nabla u)\cdot n \,\mathrm{d}S = \int_{e_\mathrm{e}} a \cdot \tfrac{\partial u}{\partial x_1} \,\mathrm{d}S\,.$$

If we denote h_1 as the width of the control volume and h_2 as the height we have

$$\int_{e_\mathrm{e}} (a\nabla u)\cdot n \,\mathrm{d}S = \int_{x_2-h_2/2}^{x_2+h_2/2} a\tfrac{\partial u}{\partial x_1}\left(x_1 + \tfrac{h_1}{2}, x_2\right) \mathrm{d}x_2\,.$$

Recall that we assumed a to be constant on Ω_i. Thus, application of the midpoint rule yields that

$$\int_{e_\mathrm{e}} (a\nabla u)\cdot n \,\mathrm{d}S \approx h_2 \cdot a(x_1, x_2) \cdot \tfrac{\partial u}{\partial x_1}(x_1 + \tfrac{h_1}{2}, x_2), \tag{4.5}$$

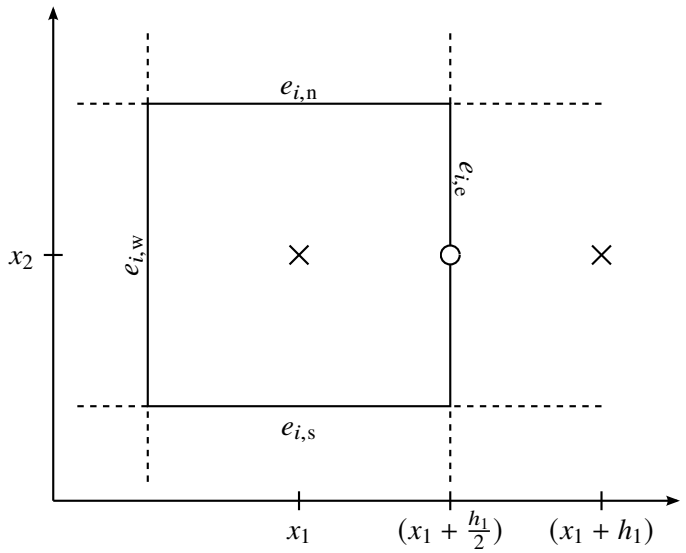


Figure 4.3.: The edges and edge mid-points of a control volume.

where we used that $a(x_1 + \frac{h_1}{2}, x_2) = a(x_1, x_2)$. It remains to approximate the partial derivative of u in x_1-direction at the point $(x_1 + \frac{h_1}{2}, x_2)$.

The approximation of the partial derivative requires some special care. The coefficient a can be discontinuous at the point $(x_1 + \frac{h_1}{2}, x_2)$. Hence, the partial derivative $\frac{\partial u}{\partial x_1}$ can be discontinuous at that point, too, implying that a Taylor series expansion of u is only an approximation of u to the point of discontinuity, but not beyond. Thus, we approximate the partial derivative by

$$\frac{\partial u}{\partial x_1}\left(x_1 + \tfrac{h_1}{2}, x_2\right) \approx \frac{u\left(x_1 + \frac{h_1}{2}, x_2\right) - u\left(x_1, x_2\right)}{h_1/2}. \tag{4.6}$$

Therefore, we need to find a way to approximate $u(x_1 + \frac{h_1}{2}, x_2)$.

To approximate the function value at the control volume midpoint we use the following relation. We know that the flux $a\nabla u$ is continuous, in particular that

$$\lim_{\bar{x}_1 \nearrow \left(x_1 + \frac{h_1}{2}\right)} a(\bar{x}_1, x_2) \cdot \frac{\partial u}{\partial x_1}(\bar{x}_1, x_2) = \lim_{\bar{x}_1 \searrow \left(x_1 + \frac{h_1}{2}\right)} a(\bar{x}_1, x_2) \cdot \frac{\partial u}{\partial x_1}(\bar{x}_1, x_2).$$

If we plug in the approximation (4.6) for the left hand side and a similar finite difference for the right hand side we get the equation

$$a(x_1, x_2) \cdot \frac{u\left(x_1 + \frac{h_1}{2}, x_2\right) - u\left(x_1, x_2\right)}{h_1/2} = a(x_1 + h_1, x_2) \cdot \frac{u\left(x_1 + h_1, x_2\right) - u\left(x_1 + \frac{h_1}{2}, x_2\right)}{h_1/2}.$$

Solving for $u\left(x_1 + \frac{h_1}{2}, x_2\right)$ gives that

$$u\left(x_1 + \tfrac{h_1}{2}, x_2\right) = \frac{a(x_1, x_2)u(x_1, x_2) + a(x_1 + h_1, x_2)u(x_1 + h_1, x_2)}{a(x_1, x_2) + a(x_1 + h_1, x_2)},$$

which can be plugged into (4.6) to yield after a few algebraic simplifications that

$$\frac{\partial u}{\partial x_1}\left(x_1 + \tfrac{h_1}{2}, x_2\right) \approx \frac{2}{h_1} \cdot \frac{a(x_1 + h_1, x_2)}{a(x_1, x_2) + a(x_1 + h_1, x_2)}\Big[u(x_1 + h_1, x_2) - u(x_1, x_2)\Big]. \tag{4.7}$$

We have arrived at an expression that involves the values of u only at x_i for $i = 1, \ldots, n$. Hence, it remains to put the individual pieces together.

To obtain the final set of equations we start by substituting (4.7) back into (4.5), which gives that

$$\int_{e_\mathrm{e}} (a\nabla u) \cdot n \,\mathrm{d}S \approx \omega_\mathrm{e} \cdot \Big[u(x_\mathrm{e}) - u(x)\Big] \quad \text{where}$$

$$x_\mathrm{e} = (x_1 + h_1, x_2) \quad \text{and} \quad \omega_\mathrm{e} = \frac{2h_2}{h_1} \cdot \frac{a(x) \cdot a(x_\mathrm{e})}{a(x) + a(x_\mathrm{e})}. \tag{4.8a}$$

Similarly if we define

$$x_\mathrm{w} = (x_1 - h_1, x_2), \qquad \omega_\mathrm{w} = \frac{2h_2}{h_1} \cdot \frac{a(x) \cdot a(x_\mathrm{w})}{a(x) + a(x_\mathrm{w})}, \tag{4.8b}$$

$$x_\mathrm{n} = (x_1, x_2 + h_2), \qquad \omega_\mathrm{n} = \frac{2h_1}{h_2} \cdot \frac{a(x) \cdot a(x_\mathrm{n})}{a(x) + a(x_\mathrm{n})}, \tag{4.8c}$$

$$x_\mathrm{s} = (x_1, x_2 - h_2), \qquad \omega_\mathrm{s} = \frac{2h_1}{h_2} \cdot \frac{a(x) \cdot a(x_\mathrm{s})}{a(x) + a(x_\mathrm{s})} \tag{4.8d}$$

then

$$\int_{e_d} (a\nabla u) \cdot n \,\mathrm{d}S \approx \omega_d \cdot \Big[u(x_d) - u(x)\Big] \quad \text{for} \quad d \in \{\mathrm{n}, \mathrm{e}, \mathrm{s}, \mathrm{w}\}.$$

Using that the integral over the control volume can be computed by the sum over the integrals over the edges, i.e,

$$-\int_{\Omega_i} \nabla \cdot (a\nabla u) \,\mathrm{d}V = -\sum_{e \in \mathcal{E}_i} \int_e \langle a\nabla u, n\rangle \,\mathrm{d}S, \tag{4.4 revisited}$$

we obtain

$$-\int_{\Omega_i} \nabla \cdot (a\nabla u) \,\mathrm{d}V \approx -\sum_{d \in \{\mathrm{n},\mathrm{e},\mathrm{s},\mathrm{w}\}} \omega_d \Big[u(x_d) - u(x)\Big]. \tag{4.9}$$

We can now plug this approximation and the approximation of the right hand side (4.3) into

$$-\int_{\Omega_i} \nabla \cdot (a\nabla u) \,\mathrm{d}V = \int_{\Omega_i} f \,\mathrm{d}V \quad \text{for} \quad i = 1, \ldots, n, \tag{4.2 revisited}$$

and obtain the set of equations

$$-\sum_{d\in\{\mathrm{n,e,s,w}\}} \omega_d \Big[u(x_{i,d}) - u(x_i))\Big] = h_1 h_2 \cdot f(x_{i,1}, x_{i,2}) \quad \text{for} \quad i = 1, \dots, n\,.$$

This is a linear set of equations which can be solved to obtain the approximations of u at the points x_i. Note that the left hand side can be represented by the stencil

$$\begin{bmatrix} & -\omega_\mathrm{n} & \\ -\omega_\mathrm{w} & (\omega_\mathrm{n}+\omega_\mathrm{e}+\omega_\mathrm{s}+\omega_\mathrm{w}) & -\omega_\mathrm{e} \\ & -\omega_\mathrm{s} & \end{bmatrix}_{\mathbf{h}}\,. \tag{4.10}$$

4.1.2. Fourier Analysis I

We want to construct a multigrid method that solves the linear system from the finite volume discretization of the PDE (4.1a), whose coefficient a has a jump. To construct a multigrid method, we need to pick a smoother and a coarse grid correction that work well together. Let us start with the smoother.

We consider the weighted Jacobi method and want to know whether it is a suitable smoother for this problem. To answer this question we perform an LFA for the error propagator of the Jacobi method given by

$$E = I - \omega D^{-1} A\,. \tag{1.33 revisited}$$

In this equation A is the matrix constructed by the finite volume method and D its diagonal entries.

To perform an LFA we need to extend all operators to the infinite grid $G_{\mathbf{h}}$. The operator A can be easily extended to the whole grid by using the stencil of the finite volume method (4.10) for the whole grid. Then D is also defined on $G_{\mathbf{h}}$. The stencil, however, depends on the coefficient a. Thus, a has to be defined on the whole space $\mathbb{R}^2$.

Let us consider the situation where a is constant along vertical stripes in the domain $\mathbb{R}^2$, and the coefficient a alternates between $a = 1$ and $a = 10^6$ as shown in Figure 4.4. To use the finite volume discretization, the jumps in the coefficient have to be aligned with the boundary of a control volume. Furthermore, if we pick a to be periodic, then A is given by a periodic stencil, which can be shown by an easy calculation. Thus, we choose $\mathbf{p} = \mathbf{n} \cdot \mathbf{h}$ (with n_1 even) and a $\mathbf{p}$-periodic with

$$\begin{aligned} a(\mathbf{x}) &= 1 \qquad \text{for all} \quad 0 \le x_1 + \tfrac{h_1}{2} < \tfrac{p_1}{2} \\ a(\mathbf{x}) &= 10^6 \quad \text{for all} \quad \tfrac{p_1}{2} \le x_1 + \tfrac{h_1}{2} < p_1\,. \end{aligned}$$

In this case we know how to perform an LFA.

We compute the Fourier matrix symbols $\hat{a}$ and $\hat{d}$ of A and D using Theorem 3.30. Then

$$\hat{e} = 1 - \omega \hat{d}^{-1} \hat{a}$$

is the Fourier matrix symbol of the error propagator of the Jacobi method. Figure 4.5 shows the frequency emission for $\omega = 0.8$. The large values of the frequency emission are located in

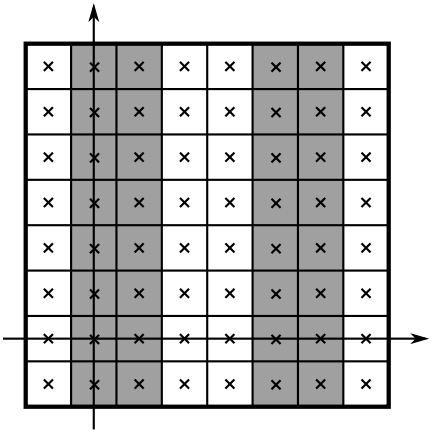


Figure 4.4.: The pattern for the jumping coefficient. In the gray region $a = 1$ and in the white region $a = 10^6$.

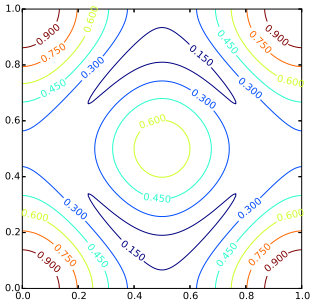


Figure 4.5.: The values of the frequency emission of the error propagator of the weighted Jacobi method. The method is applied to the jumping coefficient problem and $\omega = 0.8$.

the corners of the plot, where the low frequencies are located. In other words, the frequencies that dominate the output of E are the low frequencies. Thus, the output should be slowly varying, and the weighted Jacobi method is therefore a suitable smoother for the jumping coefficient problem.

Let us now turn to the choice of an effective coarse grid correction. We need to choose: a coarse step-size H, an interpolation $P : \ell_2(G_H) \to \ell_2(G_h)$, a restriction $R : \ell_2(G_h) \to \ell_2(G_H)$, and a coarse grid approximation $A_H : \ell_2(G_H) \to \ell_2(G_H)$. For the coarse step-size we pick $H = 2h$. We have seen that the Jacobi method produces a slowly varying error when applied to the jumping coefficient problem. Hence, as a slowly varying error is well approximated by linear interpolation, we choose P as linear interpolation. For the restriction we then choose $R = P^*$.

Rediscretizing the PDE on the coarse grid with step-size H does not give a good coarse grid approximation. Recall that the finite volume method requires that the jump in the coefficient be located as the boundary of the control volume. Even if this condition is fulfilled for the step-size h, it might be violated for the step-size H. Hence, we cannot choose A_H as the finite volume discretization of the problem on the coarse grid G_H but rather choose A_H to be the *Galerkin coarse grid approximation*, i.e.,

$$A_H := RA_hP\,.$$

This completes the two-grid method.

We can now analyze the two-grid method by using LFA. The error propagator of the method is

$$E_{\mathrm{TG}} = S^{\nu_2}(I - PA_H^{-1}RA_h)S^{\nu_1}. \qquad \text{(1.35 revisited)}$$

Since all operators on the right hand side of the equation have Fourier matrix symbols, the Fourier matrix symbol of E_{TG} is

$$\hat{e}_{\mathrm{TG}} = \hat{s}^{\nu_2}(1 - \hat{p}\hat{a}_H^{-1}\hat{r}\hat{a}_h)\hat{s}^{\nu_1}\,.$$

From the symbol we compute the norm and spectral radius of the error propagator E_{TG}. We have

$$r(E) = 0.46 \quad \text{and} \quad \|E\| = 0.91,$$

i.e., on the one hand the spectral radius is small, indicating an efficient method, on the other hand the norm is large, indicating an inefficient method.

The difference between the spectral radius and the norm can be interpreted as follows. The *spectral radius* describes the asymptotic behavior of the method, and a small spectral radius means that the method works well when many iterations are performed. The *norm* describes the worst case behavior of the method when just a single iteration is performed, and a large norm means that for just a few iterations the method can indeed be inefficient.

This behavior can be explained by LFA. We have already seen that the Jacobi method works well as a smoother for this problem; let us take a look at the coarse grid correction. The error propagator of the coarse grid correction is

$$E_{\mathrm{CGC}} = I - PA_H^{-1}RA_h. \qquad \text{(1.34 revisited)}$$

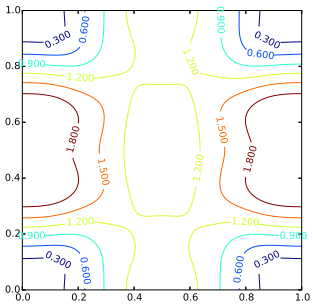


Figure 4.6.: Frequency damping of the coarse grid correction applied to the jumping coefficient problem.

Thus, its Fourier matrix symbol is

$$\hat{e}_{\mathrm{CGC}} = 1 - \hat{p}\hat{a}_H^{-1}\hat{r}\hat{a}_h \,.$$

Figure 4.6 displays the frequency damping of the coarse grid correction $\hat{e}_{\mathrm{CGC}}$. We see that some high frequencies are amplified and that the largest amplification factor is 1.8. This observation explains the behavior of the method.

During the first few iterations the error is oscillatory, and the coarse grid correction is applied to an oscillatory function, which amplifies certain parts of its frequencies. Hence, we expect a bad convergence rate in the beginning. During later iterations the error becomes slowly varying and the coarse grid correction works efficiently. So why does the coarse grid correction behaves that way?

Linear interpolation is not a good choice when we have a jump in the coefficient. As the coefficient is discontinuous, we expect the solution u to have a kink at the line of discontinuity. Thus, in the beginning, when the error is just the solution, linear interpolation should be able to interpolate functions with a kink. Linear interpolation, however, approximates a point badly if the kink appears between two neighboring coarse grid points. Therefore, we need a better interpolation, which we derive in the next section.

4.1.3. Adaptive Interpolation

We shall describe the operator dependent interpolation introduced in [3, 17]. As the name suggests, this interpolation operator is derived from the stencil of the jumping coefficient differential operator in (4.1). We first describe how linear interpolation can be derived from the Laplace operator, then we show how this idea carries over to the differential operator from (4.1).

Linear Interpolation

Linear interpolation is not well suited for the jumping coefficient problem. If the coefficient a is constant then we know that ∇u is continuous. If a is discontinuous, however, ∇u is not constant—only $a\nabla u$ is. We shall see, why this fact is a problem for linear interpolation.

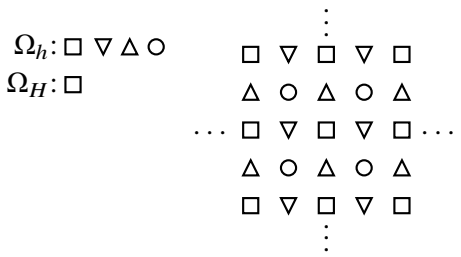


Figure 4.7.: The splitting of the grid into coarse and fine grid points.

Let us show how the assumption that ∇u is continuous, naturally leads to linear interpolation. Consider the following 1D interpolation problem. Let $x_0 < x_1$ be given. We want to approximate a function u on the interval $[x_0, x_1]$ by a function $\tilde{u}$, and we know the values $u(x_0)$, $u(x_1)$, and that u' is continuous. That u' is continuous means that it can be approximated locally by a constant. Thus, we want that our approximation $\tilde{u}$ fulfills that $\tilde{u}'$ is constant, $\tilde{u}(x_0) = u(x_0)$, and $\tilde{u}(x_1) = u(x_1)$. This constraint, however, completely determines $\tilde{u}$ and $\tilde{u}$ is linear interpolation of the points $(x_0, u(x_0))$ and $(x_1, u(x_1))$.

The assumption that u' is continuous is crucial for $\tilde{u}$ being a good approximation to u. If u' is discontinuous then the graph of u is in general not well approximated by a straight line.

Let us now construct an approximation $\tilde{u}$ of a function u defined on $G_{\mathbf{h}}$ using only the function values at $G_{2\mathbf{h}}$. In other words, we know the function value of u at the squares, shown in Figure 4.7, and want to approximate u at the triangles and circles. The values at the triangles can easily be interpolated by the one dimensional interpolation discussed above. The values at the upward pointing triangles can be interpolated from the values at the squares above and below the triangle. The values at the downward pointing triangles can interpolated from the values at the squares to their left and right. Approximating the values at the circles requires a bit more work.

The function values at the circles could be approximated by the 1D interpolation from the already interpolated values at upward pointing triangles or from the values at the downward pointing triangles. We show, however, another approach that generalizes to the jumping coefficient case. Let us assume that there is an approximation function $\tilde{u}$. We know that ∇u is continuous. Hence, it is reasonable to require that $\frac{\partial \tilde{u}}{\partial x_1}$ and $\frac{\partial \tilde{u}}{\partial x_2}$ are constant, which implies that $\frac{\partial^2 \tilde{u}}{\partial x_1^2} = 0$ and $\frac{\partial^2 \tilde{u}}{\partial x_2^2} = 0$. Therefore,

$$\frac{\partial^2 \tilde{u}}{\partial x_1^2} + \frac{\partial^2 \tilde{u}}{\partial x_2^2} = 0\,.$$

This is the Poisson equation and we have the approximation

$$\Delta_{\mathbf{h}} \tilde{u}(\mathbf{x}) = 0$$

given in (1.49). Thus,

$$\tfrac{1}{h^2}\Big(\tilde{u}(\mathbf{x}_{\mathrm{n}}) + \tilde{u}(\mathbf{x}_{\mathrm{e}}) - 4\tilde{u}(\mathbf{x}) + \tilde{u}(\mathbf{x}_{\mathrm{s}}) + \tilde{u}(\mathbf{x}_{\mathrm{w}})\Big) = 0\,,$$

where we used the notation from Section 1.5.2. The last equation implies

$$\tilde{u}(\mathbf{x}) = \tfrac{1}{4}\Big(\tilde{u}(\mathbf{x}_\mathrm{n}) + \tilde{u}(\mathbf{x}_\mathrm{e}) + \tilde{u}(\mathbf{x}_\mathrm{s}) + \tilde{u}(\mathbf{x}_\mathrm{w})\Big)\,.$$

A simple calculation shows that this formula is the (bi-)linear interpolation introduced in (1.50).

Operator Dependent Interpolation

In the case of a jumping coefficient we cannot assume that ∇u is continuous. However, $a\nabla u$ can be assumed to be continuous. We shall derive an approximation to $\tilde{u}$ along the lines of the previous section.

Let us start with the 1D case. We assume that

1. we are given the points $x_1 < \bar{x} < x_2$ and function values $u(x_1)$ and $u(x_2)$,
2. the coefficient a is piecewise constant and has one jump at $\bar{x}$, i.e.,

$$a(x) = \begin{cases} a_1 & \text{for } x < \bar{x}, \\ a_2 & \text{for } \bar{x} \le x, \end{cases}$$

   with $a_1, a_2 \in \mathbb{R}$, and
3. the functions u and au' are continuous.

Let us construct an interpolation function $\tilde{u}$. We know that au' is continuous, consequently we assume that if $a\tilde{u}'$ is equal to a constant c, then $\tilde{u}$ is a good approximation for u. This condition is fulfilled if

$$\tilde{u}'(x) = \begin{cases} c/a_1 & \text{for } x < \bar{x}, \\ c/a_2 & \text{for } x \ge \bar{x}. \end{cases}$$

Integrating $\tilde{u}'$ yields

$$\tilde{u}(x) = \begin{cases} d_1 + c/a_1 \cdot x & \text{for } x < \bar{x}, \\ d_2 + c/a_2 \cdot x & \text{for } x \ge \bar{x}. \end{cases}$$

We want $\tilde{u}(x_1) = u(x_1)$, $\tilde{u}(x_2) = u(x_2)$, and $\tilde{u}$ to be continuous, i.e., $\lim_{\nearrow \bar{x}} \tilde{u}(x) = \lim_{\swarrow \bar{x}} \tilde{u}(x)$. These equations can be written as

$$\begin{aligned} d_1 + \frac{c}{a_1} \cdot x_1 &= u(x_1)\,, \\ d_2 + \frac{c}{a_2} \cdot x_2 &= u(x_2)\,, \\ d_1 + \frac{c}{a_1} \cdot \bar{x} &= d_2 + \frac{c}{a_2} \cdot \bar{x}\,. \end{aligned}$$

These three equations form a linear system w.r.t. the variables d_1, d_2, and c. Solving the system for c gives

$$c = \frac{a_1 \cdot a_2}{(\bar{x} - x_1)a_2 + (x_2 - \bar{x})a_1}\Big[u(x_2) - u(x_1)\Big]\,. \tag{4.11}$$

Knowing c, we can easily compute d_1 and d_2.

With this knowledge we can turn to the case of the finite volume method. In this case, the coefficient can be discontinuous at the left *and* at the right boundary of the control volume. Thus, if we want to interpolate the function value at the center of the control volume, we need to construct an interpolation function where a is allowed to have two discontinuities instead of one.

Let $x_0, h_1 \in \mathbb{R}$, and let us assume that we know $u(x_0 - h_1)$ and $u(x_0 + h_1)$. Furthermore the coefficient a is given by

$$a(x) = \begin{cases} a_1 & \text{for } x < x_1 - h_1\,, \\ a_2 & \text{for } x_1 - h_1 \le x < x_0 + h_1\,, \\ a_3 & \text{for } x_0 + h_1 \le x\,. \end{cases}$$

We want to find an approximation $\tilde{u}$ for u with

$$\tilde{u}(x_0 - h_1) = u(x_0 - h_1), \quad \tilde{u}(x_0 + h_1) = u(x_0 + h_1), \quad \text{and} \quad a\tilde{u}' \text{ constant.}$$

These are nearly the same conditions as for the approximation of the beginning of this section. The only difference is the number of jumps of a. We can, however, reduce this problem in the following way to the case considered before. Let

$$\tilde{u}(x) := \begin{cases} \tilde{u}_{\mathrm{w}}(x) & \text{for } x < x_0\,, \\ \tilde{u}_{\mathrm{e}}(x) & \text{for } x \ge x_0\,, \end{cases}$$

where

$$\begin{aligned} &\tilde{u}_{\mathrm{w}}(x_0 - h_1) = u(x_0 - h_1), && \tilde{u}_{\mathrm{w}}(x_0) = y_0\,, && a\tilde{u}' \text{ constant for } x < x_0, \\ &\tilde{u}_{\mathrm{e}}(x_0) = y_0\,, && \tilde{u}_{\mathrm{w}}(x_0 + h_1) = u(x_0 + h_1), && a\tilde{u}' \text{ constant for } x < x_0\,. \end{aligned}$$

In other words, the function u_{w} approximates the left half of u while the function u_{e} approximates the right half of u. In order to guarantee that $a\tilde{u}'$ is constant, we need that

$$a\tilde{u}'_{\mathrm{w}}(x_0) = a\tilde{u}'_{\mathrm{e}}(x_0)\,. \tag{4.12}$$

Note that $a\tilde{u}'_{\mathrm{w}} = c$, which we can use in (4.11) to obtain that

$$a\tilde{u}'_{\mathrm{w}}(x_0) = \frac{2}{h_1} \cdot \frac{a_1 \cdot a_2}{a_1 + a_2} \cdot \Big[y_0 - u(x_0 - h_1) \Big]\,.$$

We realize that the factor on the right hand side is $\frac{\omega_{\mathrm{w}}}{h_2}$ from the finite volume method (4.8). Thus,

$$a\tilde{u}'_{\mathrm{w}}(x_0) = \frac{\omega_{\mathrm{w}}}{h_2} \cdot \Big[y_0 - u(x_0 - h_1) \Big]\,.$$

In a similar way,

$$a\tilde{u}'_{\mathrm{e}}(x_0) = \frac{2}{h_1} \cdot \frac{a_2 \cdot a_3}{a_2 + a_3} \cdot \Big[u(x_0 + h_1) - y_0 \Big] = \frac{\omega_{\mathrm{e}}}{h_2} \cdot \Big[u(x_0 + h_1) - y_0 \Big]\,.$$

Plugging these formulas for $a\tilde{u}'_{\mathrm{w}}(x_0)$ and $a\tilde{u}'_{\mathrm{e}}(x_0)$ into the equation (4.12), we have that

$$\frac{\omega_{\mathrm{w}}}{h_2} \cdot \Big[y_0 - u(x_0 - h_1) \Big] = \frac{\omega_{\mathrm{e}}}{h_2} \cdot \Big[u(x_0 + h_1) - y_0 \Big] .$$

This equation can be solved for y_0, which gives that

$$y_0 = \frac{\omega_{\mathrm{w}}}{\omega_{\mathrm{w}} + \omega_{\mathrm{e}}} \cdot u(x_0 - h_1) + \frac{\omega_{\mathrm{e}}}{\omega_{\mathrm{w}} + \omega_{\mathrm{e}}} \cdot u(x_0 + h_1) . \tag{4.13}$$

From this equation we see that the interpolation weights for the interpolation operator, needed for the finite volume method, can be derived from the stencil of the finite volume method. Therefore, this interpolation is also called *operator dependent interpolation*.

Let us now consider the case of a 2D grid. We want to construct an interpolation that takes the function values $u(\mathbf{x})$ for $\mathbf{x} \in G_{2\mathbf{h}}$ and gives an approximation $\tilde{u}(\mathbf{x})$ for $\mathbf{x} \in G_{\mathbf{h}}$. From Figure 4.7 we see that we need to consider four cases.

(1) We consider a point $\mathbf{x} \in G_{2\mathbf{h}}$, denoted by a square in Figure 4.7. Since we know the function value $u(\mathbf{x})$ for $\mathbf{x} \in G_{2\mathbf{h}}$, we set

$$\tilde{u}(\mathbf{x}) = u(\mathbf{x}) .$$

(2) We consider a point $\mathbf{x}$, denoted by a downward pointing triangle in Figure 4.7 that is located on the horizontal line between two points from $G_{2\mathbf{h}}$. As these two points and $\mathbf{x}$ lie on a line, we can use the 1D interpolation formula (4.13), i.e.,

$$\tilde{u}(\mathbf{x}) = \frac{\omega_{\mathrm{w}}}{\omega_{\mathrm{w}} + \omega_{\mathrm{e}}} \cdot u(x_1 - h_1, x_2) + \frac{\omega_{\mathrm{e}}}{\omega_{\mathrm{w}} + \omega_{\mathrm{e}}} \cdot u(x_1 + h_1, x_2) .$$

(3) We consider a point $\mathbf{x}$, denoted by a upward pointing triangle in Figure 4.7 that is located on the vertical line between two points from $G_{2\mathbf{h}}$. Like in the second case we set

$$\tilde{u}(\mathbf{x}) = \frac{\omega_{\mathrm{s}}}{\omega_{\mathrm{s}} + \omega_{\mathrm{n}}} \cdot u(x_1, x_2 - h_2) + \frac{\omega_{\mathrm{n}}}{\omega_{\mathrm{s}} + \omega_{\mathrm{n}}} \cdot u(x_1, x_2 + h_2) .$$

(4) We consider a point $\mathbf{x}$, denoted by a circle in Figure 4.7 that is at the center of four coarse grid points. In this case we compute $\tilde{u}(\mathbf{x})$ by using the points computed in the second and third case. As we stated earlier, under the assumption that $a\nabla u$ is continuous, it is reasonable to assume that $\tilde{u}$ is a good approximation if $a\frac{\partial \tilde{u}}{\partial x_1}$ and $a\frac{\partial \tilde{u}}{\partial x_2}$ are constant, implying that

$$\frac{\partial}{\partial x_1}\left(a \frac{\partial \tilde{u}}{\partial x_1} \right) = 0 \quad \text{and} \quad \frac{\partial}{\partial x_2}\left(a \frac{\partial \tilde{u}}{\partial x_2} \right) = 0 .$$

Adding the two equations yields

$$\nabla \cdot (a \nabla \tilde{u}) = 0 . \tag{4.14}$$

Recall that

$$-\int_{\Omega_i} \nabla \cdot (a \nabla u) \, \mathrm{d}V \approx -\sum_{d \in \{\mathrm{n},\mathrm{e},\mathrm{s},\mathrm{w}\}} \omega_d \Big[u(x_d) - u(x) \Big] \tag{4.9 revisited}$$

and the volume of the control volume Ω_i is $h_1 \cdot h_2$. If we assume that the control volume is small, then

$$\nabla \cdot (a \nabla \tilde{u}) \approx \tfrac{1}{h_1 h_2} \sum_{d \in \{\mathrm{n,e,s,w}\}} \omega_d \left[\tilde{u}(x) - \tilde{u}(x_d) \right] .$$

Plugging this approximation into (4.14) and solving for $\tilde{u}(\mathbf{x})$ then gives

$$\tilde{u}(\mathbf{x}) = \frac{1}{\omega_\mathrm{n} + \omega_\mathrm{e} + \omega_\mathrm{s} + \omega_\mathrm{w}} \cdot \left[\omega_\mathrm{n} \tilde{u}(x_\mathrm{n}) + \omega_\mathrm{e} \tilde{u}(x_\mathrm{e}) + \omega_\mathrm{s} \tilde{u}(x_\mathrm{s}) + \omega_\mathrm{w} \tilde{u}(x_\mathrm{w}) \right] .$$

Remark 4.1. This interpolation is only a good choice for a two-grid method. Using this interpolation in the Galerkin coarse grid approximation leads to a 9-point stencil. Thus, for the third level the interpolation should be adapted. For the adapted version of the interpolation see [3, 17].

4.1.4. Fourier Analysis II

The adaptive interpolation, introduced in the previous section, allows us to construct a new two-grid method. Let P be the adaptive interpolation, let the restriction by given by $R = P^*$, and let the coarse grid approximation fulfill $A_H = RA_hP = P^*A_hP$. Furthermore, we choose the Jacobi method for smoothing the error. These choices determine the two-grid method.

We can analyze the effectiveness of the new method by inspecting the Fourier matrix symbols of the involved error propagators. We start with the symbol of the error propagator of the coarse grid correction, which is given by

$$\hat{e}_\mathrm{CGC} = 1 - \hat{p}(\hat{p}^* \hat{a}_h \hat{p})^{-1} \hat{p}^* \hat{a}_h ,$$

where $\hat{p}$ and $\hat{a}_h$ are the Fourier matrix symbols of P and A. Figure 4.8 shows the frequency damping of $\hat{e}_\mathrm{CGC}$. When comparing this Figure to Figure 4.6, we see that the large amplifications of some of the oscillatory modes are gone when using the adaptive interpolation. Thus, the new coarse grid correction can be used with oscillatory functions. In this case the slowly varying part of the error is reduced, while the oscillatory part remains unchanged.

The Fourier matrix symbol of the error propagator of the whole two-grid method gives further information about the effectiveness of the method. The symbol of the two-grid method is

$$\hat{e}_\mathrm{TG} = \hat{s}^{\nu_2} \cdot \hat{e}_\mathrm{CGC} \cdot \hat{s}^{\nu_1} ,$$

where $\hat{s}$ is the symbol of the error propagator of the Jacobi method. From the symbol $\hat{e}_\mathrm{TG}$ we compute the norm and spectral radius of the error propagator. The results for the new two-grid method and the two-grid method that uses linear interpolation are listed in Table 4.1. In there we see that the adaptive interpolation reduces the norm of E_TG substantially, meaning that the new method works better during the first iterations than the old one. Furthermore, we see that the spectral radius is also reduced, i.e., the asymptotic convergence rate improves. All in all, the method is improved by using the adaptive interpolation.

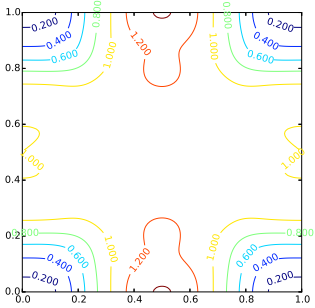


Figure 4.8.: Frequency damping of the coarse grid correction with adaptive interpolation applied to the jumping coefficient problem.

	Linear Interpolation	Adaptive
$r(E_{\mathrm{TG}})$	0.46	0.36
$\|E_{\mathrm{TG}}\|$	0.91	0.53

Table 4.1.: Comparison of linear interpolation and adaptive two-grid method.

4.1.5. Numerical Evaluation

Local Fourier analysis is an idealized analysis. It assumes that the problem is posed on an infinite grid, while in real computations we can, of course, only work with finite grids. Thus, we want to compare our prediction, that the spectral radius of the adaptive two-grid method $\rho(E_{\mathrm{TG}}) = 0.36$ (see Table 4.1), to an actual run of the method.

For the finite grid we choose the domain by $\Omega = (0,1) \times (0,1)$, as the unit square, and apply Dirichlet boundary conditions on $\partial\Omega$. We discretize the domain by a regular 33×33 grid. Furthermore, we choose the coefficient $a(\mathbf{x})$ in the following way. In the center of the domain we let $a(\mathbf{x})$ alternate in the x_1 direction between 1 and 10^6 eleven times. More precisely, we change the value of $a(\mathbf{x})$ in this area changes every two control volumes. Thus, in this area, the coefficient a consists of eleven strips of width $2h$. For this case we compute the spectral radius of the adaptive two-grid method with Jacobi ($\omega = 0.8$) and obtain $\rho(E_{\mathrm{TG}}^{\mathrm{real}}) = 0.36$, which is in perfect agreement with our prediction.

When running the method on a finite grid, we observe that the number of jumps does not influence the convergence rate. Hence, we wonder whether the Fourier analysis is also able to predict the behavior of a domain that contains just one jump. Thus, we run the same experiment again with

$$a(\mathbf{x}) = \begin{cases} 10^6 & \text{if } x_1 < 1/2 \\ 1 & \text{if } x_1 \geq 1/2 \end{cases}.$$

When we compute the spectral radius, we obtain $\rho(E_{\mathrm{TG}}^{\mathrm{real}}) = 0.36$, which is still in perfect agreement with the prediction, even though we have just one jump in the domain and our analysis assumes that we have infinitely many jumps.

4.2. Block Smoothers

In every iteration pointwise smoothers solve a large number of linear systems in one variable; block smoothers solve fewer linear systems, but each system has more than one variable. We shall show, using Fourier matrix symbols, how this difference affects the smoothing properties of the methods.

4.2.1. Block Jacobi

The *block Jacobi method* is a generalization of the Jacobi method. It has often a better convergence rate than the plain Jacobi method but requires a larger amount of arithmetic operations per iteration.

Derivation of the Block Jacobi Method

Recall the basic idea of the weighted Jacobi method. The method produces a sequence of iterates $u_0, u_1, \ldots$. To compute the $(k+1)$th iterate the method computes corrections $v_1, \ldots, v_n$. Then the new iterate is computed by adding the corrections to the old iterate, i.e.,

$$u_{k+1} = u_k + \omega \cdot (v_1 + \cdots + v_n)\,. \qquad \text{(1.17 revisited)}$$

Let us for simplicity of the discussion assume that $\omega = 1$. Thus,

$$u_{k+1} = u_k + v_1 + \cdots + v_n\,.$$

From this equation we obtain that the $(k+1)$th residual is given by

$$r_{u_{k+1}} = f - A(u_k + v_1 + \cdots + v_n) = r_{u_k} - Av_1 - \cdots - Av_n\,.$$

The correction v_i is chosen such that the ith component of the correction residual, which is given by

$$r_{u_k + v_i} = b - A(u_k + v_i) = r_{u_k} - Av_i\,,$$

is zero. Thus, the ith component of Av_i annihilates the ith component of the residual r_{u_k}. If the ith component of the residual is nonzero then the ith component of Av_i is nonzero. The vector Av_i, however, has in general further nonzero entries. If we would just apply the ith correction this fact would not be a problem. After the correction the ith component of the residual would be zero. We apply, however, further corrections v_j and for some of them the ith component of Av_j is nonzero, which changes the ith entry of the residual back to a nonzero value; the corrections interfere.

In other words, the purpose of the ith correction v_i is a to annihilate the ith component of the residual. As a side-effect, however, it alters other components of the residual as well. Thus, if the side-effect is small, the method converges fast. If the side effect is large, the method converges slowly or even diverges. The idea of the block Jacobi method is to reduce the side-effects by constructing corrections that annihilate multiple components of the residual at once. Using this construction, less corrections per iteration are needed which leads to less interference between the corrections and is done as follows.

Let $S_1, \dots, S_m$ be non-empty subsets of $\{1, \dots, n\}$. The ith correction should annihilate the components of the correction residual whose indices are contained in S_i, i.e.,

$$0 = r_{u_k + v_i, j} = (f - A(u_k + v_i))_j \quad \text{for} \quad j \in S_i \,. \tag{4.15}$$

In the Jacobi method we required that the correction had the form αe_i where $\alpha \in \mathbb{R}$ and e_i was the ith unit vector. Consequently α was the only degree of freedom. One degree of freedom, however, is in general not enough to annihilate multiple components of the correction residual. Hence, we require the components of the correction v_i to be zero for all indices not in S_i and allow arbitrary values for all indices in S_i, i.e.,

$$v_i \in \operatorname{span}\{e_\ell : \ell \in S_i\} \,. \tag{4.16}$$

When the corrections have been computed we can update the iterate. The new iterate is computed from the old iterate by

$$u_{k+1} = u_k + \omega \cdot (v_1 + \cdots + v_m) \,. \tag{4.17}$$

The difference to the update (1.17) is that we now apply only m corrections instead of n.

The method that we described so far is called *block Jacobi method*, if the sets $S_1, \dots, S_m$ form a partition of the index set $\{1, \dots, n\}$, i.e.,

1. $S_1 \cup S_2 \cup \cdots \cup S_m = \{1, \dots, n\}$ and
2. $S_i \cap S_j \neq \emptyset$ implies $i = j$.

Other choices of $S_1, \dots, S_m$ are possible and lead to different methods. For example, Vanka [70] formulated an efficient smoother for the Navier-Stokes equation. This smoother uses a partition with $S_i \cap S_j \neq \emptyset$ for $i \neq j$. If we, however, restrict ourselves to the case of the block Jacobi method, the method can be written in a compact form.

Theorem 4.2. *The block Jacobi method given by equations* (4.15)*,* (4.16)*,* (4.17) *and the partition $S_1, \dots, S_m$ can be written in the form*

$$u_{k+1} = u_k + \omega D^{-1} r_{u_k} \,, \tag{4.18}$$

where D is defined as follows. Let $b : \{1, 2, \dots, n\} \to \{1, 2, \dots, m\}$ the function that assigns every index $i \in \{1, \dots, n\}$ the index of the partition the index i is in, i.e.,

$$i \in S_{b(i)} \,.$$

Then the entries of D are give by

$$d_{ij} = \begin{cases} a_{ij} & \text{if } j \in S_{b(i)}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

Thus, the error propagator of the block Jacobi method is

$$E = I - \omega D^{-1} A \,. \tag{4.20}$$

Furthermore, there exists a permutation of the indices such that D is block diagonal.

Proof. Let us define $\bar{v} := v_1 + \cdots + v_m$. Then the iteration update (4.17) can be written as

$$u_{k+1} = u_k + \omega \bar{v}. \tag{4.21}$$

We shall see that the vector $\bar{v}$ can be computed without computing the individual $v_1, \ldots, v_m$.

Since the sets $S_1, \ldots, S_m$ form a partition of the index set and v_ℓ is non-zero only for the components whose index is in S_ℓ, for every i there is precisely one vector, the vector $v_{b(i)}$, that has a (potential) non-zero in position i. It follows that

$$\bar{v}_i = v_{b(i),i}\,. \tag{4.22}$$

With this knowledge we can compute $\bar{v}$.

From (4.15) we have

$$0 = \Big[f - A(u_k + v_\ell)\Big]_i = \Big[r_{u_k} - Av_\ell\Big]_i \quad \text{for all} \quad i \in S_\ell \text{ and } \ell = 1, \ldots, m.$$

This equation is equivalent to

$$r_{u_k,i} = \Big[Av_\ell\Big]_i = \sum_{j=1}^{n} a_{ij} v_{\ell,j} \quad \text{for all} \quad i \in S_\ell \text{ and } \ell = 1, \ldots, m.$$

The entry $v_{\ell,j}$ is zero if $j \notin S_\ell$, which implies

$$r_{u_k,i} = \sum_{j \in S_\ell} a_{ij} v_{\ell,j} \quad \text{for all} \quad i \in S_\ell \text{ and } \ell = 1, \ldots, m.$$

Substituting $b(i)$ for ℓ and then using (4.22), we obtain that

$$r_{u_k,i} = \sum_{j \in S_{b(i)}} a_{ij} v_{b(j),j} = \sum_{j \in S_{b(i)}} a_{ij} \bar{v}_j = \sum_{j=1}^{n} d_{ij} \bar{v}_j \quad \text{for} \quad i = 1, \ldots, n.$$

The right hand side of this equation is a matrix vector product, meaning we can write the equation as $r_{u_k} = D\bar{v}$, which implies $\bar{v} = D^{-1} r_{u_k}$. Plugging this form of $\bar{v}$ into (4.21) gives the desired iteration update (4.18).

The equation for the error propagator (4.20) follows from equation (1.32). Thus, it remains to show the statement about the block diagonal permutation of D.

It can be shown that a matrix A can be written in block diagonal form, if there exists a partition $S_1, \ldots, S_m$ of the index set $\{1, \ldots, n\}$ such that

$$i \in S_\ell \text{ and } j \notin S_\ell \quad \text{implies} \quad a_{ij} = 0\,.$$

The matrix entry $d_{ij} = 0$ if $j \notin S_{b(i)}$. Now $i \in S_{b(i)}$ by definition of $b(i)$. Thus, D can be written in block diagonal form. □

In case that we consider a grid-based problem we can give the stencil representation of the matrix D. Recall that in a grid based-problem every index i corresponds to a position $\mathbf{x}$ on the grid $\overline{\Omega_\mathbf{h}}$. In this case the stencil form s_D of D is given by

$$s_{D,\mathbf{x}}(\mathbf{y}) = \begin{cases} s_{A,\mathbf{x}}(\mathbf{y}) & \text{if } (\mathbf{x} + \mathbf{y}) \in S_{b(\mathbf{x})}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.23}$$

where s_A is the stencil of A.

Analysis of the Block Jacobi Method

Let us use LFA to analyze the block Jacobi method and compare it to other smoothers. Especially, we want to know whether the block Jacobi method is a better smoother than the plain Jacobi method. Furthermore, we investigate how the block size affects the smoothing effect of the method.

To analyze a problem by LFA we need to formulate the problem on an infinite grid. We see from the error propagator of the block Jacobi method (4.20) that we need to define the operators A and D on the infinite grid $G_{\mathbf{h}}$.

Let us assume for simplicity that the operator A can be given by a constant stencil on $G_{\mathbf{h}}$. We can then derive the operator D from the stencil of A. To define D we need to partition the grid $G_{\mathbf{h}}$. Now that we have infinitely many grid points we can partition the grid into infinitely many subsets. Thus, we assume that we have a partition S_i for $i \in \mathcal{I}$ for some index set $\mathcal{I}$. Also in this case there exists a map b with $\mathbf{x} \in S_{b(\mathbf{x})}$ for all $\mathbf{x} \in G_{\mathbf{h}}$, and the stencil of D is given by (4.23) and the block Jacobi method by (4.20).

We can analyze the block Jacobi method only for special choice of the partition. The partition we are interested in splits the grid into regular, rectangular blocks, i.e., it is given by

$$S_{\mathbf{i}} = \mathbf{i} \cdot \mathbf{n} + T_{\mathbf{n}} \quad \text{where} \quad T_{\mathbf{n}} = \{\mathbf{h} \cdot \mathbf{k} : \mathbf{0} \leq \mathbf{k} < \mathbf{n}\}, \tag{4.24}$$

which is a reasonable choice for the following reason.

The convergence rate is often bad, when the corrections v_i interfere strongly with each other. Furthermore, in many applications the corrections corresponding to neighboring grid points interfere strongly, i.e., often the components of Av_i that correspond to neighboring grid points of $\mathbf{x}$, are non-zero. Thus, computing corrections v_i, where each correction collectively annihilates the components of the residual which correspond to neighboring grid points, is reasonable.

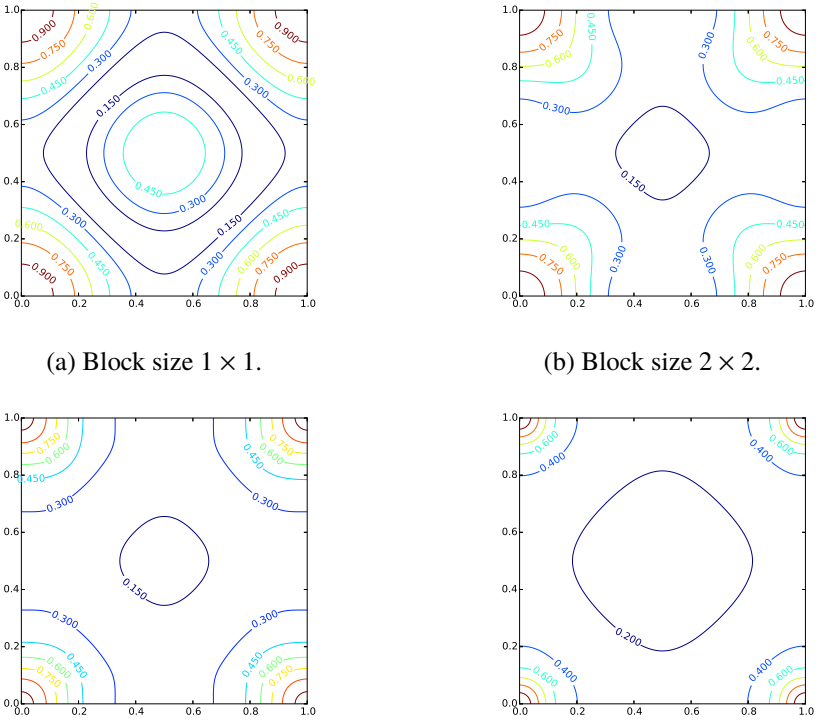
The choice of the partition (4.24) ensures that the stencil s_D is periodic. Thus, by using Theorem 3.30, we have that A and D have Fourier matrix symbols $\hat{a}$ and $\hat{d}$, and from formula for the error propagator (4.20), we have the Fourier matrix symbol of the error propagator of the block Jacobi method,

$$\hat{e} = 1 - \hat{d}^{-1}\hat{a}\,.$$

We can now use this formula to examine a concrete example.

Consider the application of the block Jacobi method to the discrete Poisson equation in 2D (1.49). We wonder whether the block Jacobi method is a better smoother for this problem than the plain Jacobi method. Hence, we plot the frequency emissions (see Section 3.4.2) of the error propagator. The result is shown in Figure 4.9 for different block sizes $\mathbf{n}$. We see that the large values of the frequency emission are located in the corners of the plot. As the low frequencies are located in the corners we conclude that the block Jacobi method reduces the high frequencies well. Thus, the method is a suitable smoother. Furthermore, when the block size increases, the large values of the frequency emission move closer to the corners of the plot. To make use of this observation, consider the following.

The smoothing factor indicates the quality of a smoother w.r.t. a coarsening range $\mathbf{c}$. To compute the smoothing factor, the smoother is combined with an idealized coarse grid



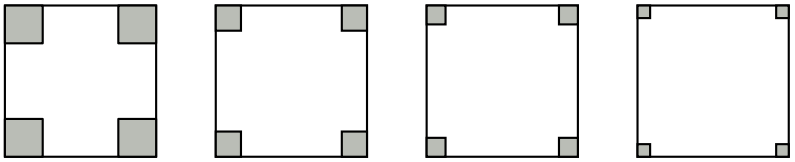
(a) Block size 1×1.

(b) Block size 2×2.

(c) Block size 4×4.

(d) Block size 8×8.

Figure 4.9.: Frequency emission of the block Jacobi method applied to the discrete Poisson equation in 2D.



(a) $\mathbf{c} = (2, 2)$ (b) $\mathbf{c} = (3, 3)$ (c) $\mathbf{c} = (4, 4)$ (d) $\mathbf{c} = (6, 6)$

Figure 4.10.: Low **c**,**h**-harmonics for different coarsening ranges **c**.

coarsening	block size				
	1	2	4	6	8
2	0.60 (0.80)	0.40 (0.80)	0.39 (0.85)	0.39 (0.85)	0.38 (0.86)
4	0.86 (0.93)	0.77 (0.88)	0.63 (0.86)	0.60 (0.86)	0.58 (0.87)
6	0.93 (0.96)	0.88 (0.94)	0.81 (0.90)	0.73 (0.90)	0.70 (0.89)
8	0.96 (0.98)	0.93 (0.96)	0.87 (0.93)	0.84 (0.92)	0.77 (0.91)

Table 4.2.: Smoothing factor of the block Jacobi method for the optimal weight (parenthesis).

correction. This coarse grid correction eliminates the low **c**,**h**-harmonics. Thus, a smoothing method whose output mainly consists of low **c**,**h**-harmonics has a small smoothing factor, which means that it is a good smoother. Now, as the low **c**,**h**-harmonics depend on **c**, so does the smoothing factor. When we increase **c** the low **c**,**h**-harmonics move closer to the corners of $\Theta_{\mathbf{h}}$, as Figure 4.10 illustrates. Therefore, to be effective for large **c**, a smoother has to return functions whose frequencies are closer to the corners, than for small **c**. Now, this is the behavior of the block Jacobi method for large block sizes b.

Let us consider the question, whether the block Jacobi method is a good smoother for large coarsening ranges **c** and how large the block size **b** has to be. We start by taking a look at the smoothing factors for different values of **c** and **b**. Recall that the block Jacobi method has a parameter, the weight, that needs to be chosen. We pick the weight that minimizes the smoothing factor[1] and list the corresponding values in Table 4.2. In the table we see that picking the block **b** equal to the coarsening range **c** yields small smoothing factors. Furthermore, with this choice, the smoothing factor grows only moderately. Thus, we conclude that the growth of the smoothing factor caused by increasing **c**, can be partly compensated by increasing the block size of the block Jacobi method.

4.2.2. Aggressive Coarsening

The smoothing analysis from the previous section suggest that block smoothers allow us to use larger coarsening ranges. In case that the coarsening range **c** is large, we say that the

[1] The smoothing factor is determined using LFA.

method uses *aggressive coarsening* [7, 13, 14, 50]. We shall introduce and analyze a two-grid method for the Poisson equation that uses aggressive coarsening and block smoothers.

Aggressive coarsening is useful, as it reduces the size of the coarse grid system. Consequently, the cost of solving the coarse grid system is reduced. The downside is that in general the convergence rate becomes worse. There can be, however, an overall benefit.

In parallel multigrid computations aggressive coarsening can be used to increase the overall performance of the multigrid method. In a parallel computation the grid is distributed along the available processors. Thus, on a very coarse grid each processor is only responsible for the computation of a few grid points. As communication is usually more expensive than computation, the time for computation is dominated by communication efforts. This situation then leads to a bad utilization of the available processing power. Aggressive coarsening can reduce this effect, because fewer coarse grids are needed, and most of the computational work is done on the finer grids where the hardware is utilized well.

We consider a two-grid method for the Poisson equation that uses aggressive coarsening. Under these conditions we want to analyze the effect of the block Jacobi method and its block size. To determine the method we need to give the smoother, interpolation, restriction and coarse approximation that we want to use. As already mentioned, we use the block Jacobi method as a smoother, then we use bi-linear interpolation, the restriction operator that is induced by linear interpolation (Definition 1.7), and rediscretization for the coarse approximation.

Until now, we have only given the formula for linear interpolation when $\mathbf{c} = \mathbf{2}$, see (1.50). In the next section we derive the formula for larger coarsening ranges.

Linear Interpolation

We derive the formula of the (bi-)linear interpolation in 2D for arbitrary coarsening ranges.

Lemma 4.3. *Bi-linear interpolation of the values* y_{00}, y_{10}, y_{01}, y_{11} *at the points* $(0,0)$, $(c_1,0)$, $(0,c_2)$, (c_1,c_2) *gives the function* f *that is described by*

$$f(\mathbf{x}) = \tfrac{(c_1-x_1)(c_2-x_2)}{c_1c_2} \cdot y_{00} + \tfrac{x_1(c_2-x_2)}{c_1c_2} \cdot y_{10} + \tfrac{(c_1-x_1)x_2}{c_1c_2} \cdot y_{01} + \tfrac{x_1x_2}{c_1c_2} \cdot y_{11} \,.$$

Proof. Bi-linear interpolation is a combination of three linear interpolations. First the values $\tilde{y}_0$ and $\tilde{y}_1$ at the points $(x_1,0)$, (x_1,c_2) are computed by linear interpolation in x_1-direction. Then the value $f(\mathbf{x})$ is computed by linear interpolation in x_2-direction from $\tilde{y}_0$ and $\tilde{y}_1$, as Figure 4.11 shows. A well known formula for linear interpolation gives $\tilde{y}_0$ and $\tilde{y}_0$;

$$\tilde{y}_0 = y_{00} \cdot \tfrac{c_1-x_1}{c_1} + y_{10} \cdot \tfrac{x_1}{c_1} \quad \text{and} \quad \tilde{y}_1 = y_{01} \cdot \tfrac{c_1-x_1}{c_1} + y_{11} \cdot \tfrac{x_1}{c_1} \,.$$

The function value of the interpolant f can be computed using the same formula; we obtain that

$$f(\mathbf{x}) = \tilde{y}_0 \cdot \tfrac{c_2-x_2}{c_2} + \tilde{y}_1 \cdot \tfrac{x_2}{c_2} \,.$$

Plugging the formulas for $\tilde{y}_0$ and $\tilde{y}_1$ into the last equation, we get that

$$f(\mathbf{x}) = \left(y_{00} \cdot \tfrac{c_1-x_1}{c_1} + y_{10} \cdot \tfrac{x_1}{c_1}\right) \cdot \tfrac{c_2-x_2}{c_2} + \left(y_{01} \cdot \tfrac{c_1-x_1}{c_1} + y_{11} \cdot \tfrac{x_1}{c_1}\right) \cdot \tfrac{x_2}{c_2} \,.$$

Expanding yields the desired equation. □

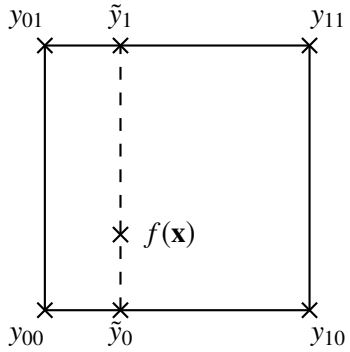


Figure 4.11.: Construction used in bi-linear interpolation. The points $\tilde{y}_0$ and $\tilde{y}_1$ are obtained by linear interpolation. From these points linear interpolation gives $f(\mathbf{x})$.

Let us now derive the stencil representation of the (bi-)linear interpolation. This representation allows us to analyze this interpolation by LFA.

Theorem 4.4. *Linear interpolation is given by $P_{\text{st}}P_{\text{inj}}$, where P_{st} is give by the stencil*

$$s'(\mathbf{i}\cdot\mathbf{h}) = \tfrac{1}{c_1c_2}\cdot\begin{cases}(c_1-|i_1|)(c_2-|i_2|) & \text{if } |i_1|<c_1 \text{ and if } |i_2|<c_2,\\ 0 & \text{otherwise.}\end{cases}$$

Proof. We want to write the interpolation in the form

$$[Pu](\mathbf{x}_b+\mathbf{k}\cdot\mathbf{h}) = \sum_{y\in G_{\mathbf{c}\cdot\mathbf{h}}} s_{\mathbf{k}}(\mathbf{y})\cdot u(\mathbf{x}_b+\mathbf{y}), \tag{4.25}$$

where $\mathbf{x}_b \in G_{\mathbf{c}\cdot\mathbf{h}}$ and $\mathbf{k} = \mathbf{0},\dots,\mathbf{c}-\mathbf{1}$, such that we can apply Theorem 3.8. Bi-linear interpolation commutes with shifting the origin and stretching the coordinate axes. Thus, without loss of generality we can compute the interpolant in the coordinate system given by $\mathbf{k}$ and assume that we are looking for the interpolant at position $\mathbf{k}$ determined by the function values at the points $\mathbf{k} = (0,0), (c_1,0), (0,c_2), (c_1,c_2)$. These points are the coarse grid points. If we define the stencil

$$s_{\mathbf{k}}(\mathbf{i}\cdot\mathbf{h}) = \begin{cases}\frac{(c_1-k_1)(c_2-k_2)}{c_1c_2} & \text{if } \mathbf{i}=(0,0),\\ \frac{k_1(c_2-k_2)}{c_1c_2} & \text{if } \mathbf{i}=(c_1,0),\\ \frac{(c_1-k_1)k_2}{c_1c_2} & \text{if } \mathbf{i}=(0,c_2),\\ \frac{k_1k_2}{c_1c_2} & \text{if } \mathbf{i}=(c_1,c_2)\\ 0 & \text{otherwise.}\end{cases}$$

then (4.25) equals the formula for bi-linear interpolation from Lemma 4.3. We see that for every value of $\mathbf{k}$ we have a different stencil. Thus, we want to use Theorem 3.8 to write the interpolation using only one combined stencil.

As we have written the interpolation in the form of equation (4.25), we can apply Theorem 3.8, which states that the combined stencil s' is given by

$$s'(\mathbf{j}\cdot\mathbf{c}\cdot\mathbf{h}-\mathbf{k}\cdot\mathbf{h}) = s_{\mathbf{k}}(\mathbf{j}\cdot\mathbf{c}\cdot\mathbf{h}),$$

where $\mathbf{j}\in\mathbb{Z}^d$ and $\mathbf{k}=\mathbf{0},\dots,\mathbf{c}-\mathbf{1}$. Plugging the formula for $s_{\mathbf{k}}$ into the current equation gives that

$$s'(\mathbf{j}\cdot\mathbf{c}\cdot\mathbf{h}-\mathbf{k}\cdot\mathbf{h}) = \begin{cases} \frac{(c_1-k_1)(c_2-k_2)}{c_1c_2} & \text{if } \mathbf{j}\cdot\mathbf{c}=(0,0),\\ \frac{k_1(c_2-k_2)}{c_1c_2} & \text{if } \mathbf{j}\cdot\mathbf{c}=(c_1,0),\\ \frac{(c_1-k_1)k_2}{c_1c_2} & \text{if } \mathbf{j}\cdot\mathbf{c}=(0,c_2),\\ \frac{k_1k_2}{c_1c_2} & \text{if } \mathbf{j}\cdot\mathbf{c}=(c_1,c_2),\\ 0 & \text{otherwise.}\end{cases} \tag{4.26}$$

We would, however, like to have a formula for $s'(\mathbf{i}\cdot\mathbf{h})$, where $\mathbf{i}=\mathbf{j}\cdot\mathbf{c}-\mathbf{k}$.

Using the relation $\mathbf{i}=\mathbf{j}\cdot\mathbf{c}-\mathbf{k}$ we can express $\mathbf{k}$ and the conditions in (4.26) in terms of $\mathbf{i}$. We have:

1. All values of i_d for which $j_d\cdot c_d=0$ holds are given by

   $$i_d = -k_d \quad \text{for} \quad k_d = 0,\dots,c_d-1\,.$$

   Thus

   $$k_d=-i_d \quad\text{and}\quad j_d\cdot c_d=0 \quad\text{if}\quad -c_d<i_d\le 0\,. \tag{4.27a}$$

2. All values of i_d for which $j_d\cdot c_d=c_d$ holds are given by

   $$i_d = c_d-k_d \quad \text{for} \quad k_d = 0,\dots,c_d-1\,.$$

   Thus

   $$k_d=c_d-i_d \quad\text{and}\quad j_d\cdot c_d=c_d \quad\text{if}\quad 0<i_d\le c_d. \tag{4.27b}$$

Therefore, using (4.27) we can rewrite (4.26) to obtain

$$\begin{aligned} s'(\mathbf{i}\cdot\mathbf{h}) &= \begin{cases} \frac{(c_1+i_1)(c_2+i_2)}{c_1c_2} & \text{if } -c_1<i_1\le 0 \text{ and } -c_2<i_2\le 0,\\ \frac{(c_1-i_1)(c_2+i_2)}{c_1c_2} & \text{if } 0<i_1\le c_1 \text{ and } -c_2<i_2\le 0,\\ \frac{(c_1+i_1)(c_2-i_2)}{c_1c_2} & \text{if } -c_1<i_1\le 0 \text{ and } 0<i_2\le c_2,\\ \frac{(c_1-i_1)(c_2-i_2)}{c_1c_2} & \text{if } 0<i_1\le c_1 \text{ and } 0<i_2\le c_2,\\ 0 & \text{otherwise}\end{cases}\\ &= \begin{cases} \frac{(c_1-|i_1|)(c_2-|i_2|)}{c_1c_2} & \text{if } -\mathbf{c}<\mathbf{i}\le\mathbf{c},\\ 0 & \text{otherwise.}\end{cases} \end{aligned}$$

□

The restriction which is induced by the (bi-)linear interpolation is easily obtained by using Theorem 4.4. Recall the definition of the induced restriction from Section 1.3.7, and let s'_{P} be the stencil given in Theorem 4.4. Then a little calculation show that the stencil of the induced restriction s'_{R} is given by

$$s'_{\mathrm{R}}(\mathbf{y}) = \tfrac{1}{\sum_{\mathbf{z}\in G_{\mathbf{h}}} s'_{\mathrm{P}}(\mathbf{z})} s'_{\mathrm{P}}(\mathbf{y})\,.$$

Remark 4.5. The results from Lemma 4.3 and Theorem 4.4 can be generalized to arbitrary dimensions. The proofs for the dD case are very similar to the 2D case, however, they are notationally intense. Therefore, we shall only state the results.

Assume there exists a point $\mathbf{c} > 0$ and values $y_{\mathbf{i}}$ for $\mathbf{i} = \mathbf{0}, \ldots, \mathbf{1}$. Let

$$p_{\mathbf{i},j} := \begin{cases} 0 & \text{if } i_j = 0, \\ c_i & \text{if } i_j = 1. \end{cases}$$

The multilinear interpolation that interpolates the values $y_{\mathbf{i}}$ at $p_{\mathbf{i}}$ gives the function f with

$$f(\mathbf{x}) = \frac{1}{\sum_{j=1}^{d} c_j} \cdot \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{1}} \left(\prod_{j=1}^{d} \begin{Bmatrix} c_j - x_j & \text{if } i_j = 0 \\ x_j & \text{if } i_j = 1 \end{Bmatrix} \right) \cdot y_{\mathbf{i}}\,.$$

The stencil of the multilinear interpolation is

$$s'(\mathbf{i}\cdot\mathbf{h}) = \begin{cases} \frac{(c_1-|i_1|)\cdot(c_2-|i_2|)\cdots(c_d-|i_d|)}{c_1\cdot c_2\cdots c_d} & \text{if } |\mathbf{i}| < \mathbf{c}, \\ 0 & \text{otherwise,} \end{cases}$$

which is obtained from the formula above.

Analysis

We use two-grid LFA to analyze the multigrid method for the Poisson equation described above.

Figure 4.12 shows the frequency damping plot of a two-grid method and aggressive coarsening applied to the 2nd order finite difference discretization of Poisson's equation. It demonstrates that the combination of (pointwise) Jacobi and aggressive coarsening fails to reduce some of the high frequency errors. However, this problem disappears if we use the block Jacobi method with a sufficiently large block size.

In Table 4.3 the spectral radii of the two-grid error propagator for different coarsenings $\mathbf{c}$ and different block sizes of the block Jacobi method with $\omega = 0.8$ are presented. It turns out that choosing the block size equal to the coarsening range can compensate for the growth in the spectral radius caused by aggressive coarsening to some extend. Furthermore, we observe that not choosing the block size as a multiple of the coarsening range has negative effect on the convergence rate.

A similar effect can be seen in Table 4.4. This table shows the spectral radii of the error propagator of the two-grid method for different block sizes and coarsening ranges. However, we computed for every parameter combination the weight of the block Jacobi method that minimizes the spectral radius of the error propagator. We can see that the results are improved compared to the previous table. The improvement, however, is only minor. We therefore conclude that for this setting a choice of $\omega = 0.8$ leads to good results.

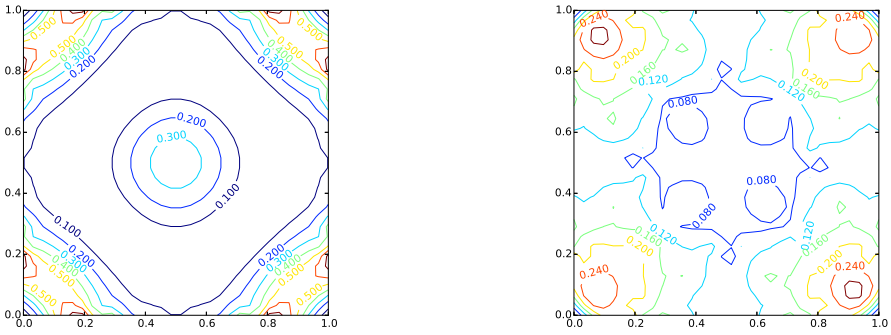


Figure 4.12.: Frequency damping plot of a two-grid method with aggressive 4×4 coarsening and a Jacobi smoother (left) and a 4×4 block Jacobi smoother (right).

coarsening	block size				
	1	2	4	6	8
2	0.36	0.32	0.27	0.26	0.25
4	0.76	0.62	0.38	0.45	0.35
6	0.88	0.79	0.68	0.47	0.58
8	0.92	0.85	0.76	0.74	0.55

Table 4.3.: Spectral radius of the two-grid method with aggressive coarsening and block Jacobi smoothing.

coarsening	block size				
	1	2	4	6	8
2	0.36 (0.80)	0.25 (0.74)	0.22 (0.75)	0.22 (0.75)	0.22 (0.75)
4	0.75 (0.93)	0.61 (0.88)	0.38 (0.79)	0.43 (0.84)	0.35 (0.80)
6	0.88 (0.96)	0.78 (0.94)	0.67 (0.90)	0.47 (0.83)	0.56 (0.87)
8	0.93 (0.98)	0.86 (0.96)	0.77 (0.93)	0.72 (0.92)	0.55 (0.86)

Table 4.4.: Two-grid convergence factor of the block Jacobi method for the optimal weight (parenthesis).

4.2.3. Red-Black Block Jacobi

The *red-black block Jacobi method* is a variant of the block Jacobi method. It is derived from the block Jacobi method in a similar way as the red-black Jacobi method, introduced in Section 3.6.1, is derived from the Jacobi method.

Derivation of the Red-Black Block Jacobi Method

The block Jacobi method computes a new iterate u_{k+1} from the old iterate u_k by

$$u_{k+1} = u_k + \omega \cdot (v_1 + \cdots + v_m), \tag{4.17 revisited}$$

where $v_1, \ldots, v_m$ is a set of corrections, which are computed from the residual r_{u_k}. Furthermore, each correction v_i corresponds to an index set S_i. These index sets form a partition of the indices $\{1, \ldots, n\}$, and the jth component of v_i is only non-zero if $j \in S_i$. Using these relations, we construct the red-black block Jacobi method.

Like in the derivation of the red-black Jacobi method, the update step is split into two individual steps, where each step applies half of the corrections. For this purpose we partition the blocks into *red* and *black* ones, by listing the indices of the red blocks in the set R_I and the indices of the black blocks in B_I. More precisely, the sets R_I and B_I are a partition of the set $\{1, \ldots, m\}$. The first step is to compute the intermediate iterate $u_{k+1/2}$, by

$$u_{k+1/2} = u_k + \omega \sum_{i \in R_I} v_i \,.$$

In the second step the new iterate is computed by the formula

$$u_{k+1/2} = u_{k+1/2} + \omega \sum_{i \in B_I} v_i \,.$$

For the second step, however, the corrections v_i are computed based on the intermediate residual $r_{u_{k+1/2}}$. To analyze this method we need a formula for its error propagator.

Theorem 4.6. *Let $R = \bigcup_{i \in R_I} S_i$ and $B = \bigcup_{i \in B_I} S_i$. The red-black block Jacobi method can be written as*

$$u_{k+1/2} = u_k + \omega Z_R D^{-1} r_{u_k} \,, \tag{4.28a}$$

$$u_{k+1} = u_{k+1/2} + \omega Z_B D^{-1} r_{u_{k+1/2}} \,, \tag{4.28b}$$

where D is given in (4.19)*,*

$$Z_{R,ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in R, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad Z_{B,ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in B, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the error propagator of the red-black Jacobi method is $E_{\mathrm{RBBJ}} = E_B E_R$, where

$$E_R = I - \omega Z_R D^{-1} A \quad \text{and} \quad E_B = I - \omega Z_B D^{-1} A \,. \tag{4.29}$$

Proof. We define the vector

$$\bar{v}_R := \sum_{i \in R_I} v_i \,. \tag{4.30}$$

With this definition we can write the first step of the red-black block Jacobi method as

$$u_{k+1/2} = u_k + \omega \cdot \bar{v}_R \,.$$

Thus, to show (4.28a) we need to show that $\bar{v}_R = Z_R D^{-1} r_{u_k}$.

To simplify the sum (4.30) we take a closer look at the vectors $v_1, \ldots, v_m$. The jth component of the vector v_i is non-zero only if $j \in S_i$. As the sets $S_1, \ldots, S_m$ form a partition of $\{1, \ldots, n\}$, there is for every j exactly one index $b(j)$ such that $j \in S_{b(j)}$. Therefore, for every j there is exactly one vector $v_{b(i)}$ whose jth component is (potentially) non-zero. Hence, when computing the jth entry of $\bar{v}_{R,j}$, by

$$\bar{v}_{R,j} = \sum_{i \in R_I} v_{i,j} \,,$$

the term $v_{b(j),j}$ is the only non-zero one, if the vector $v_{b(j)}$ is part of the sum, i.e., if $b(j) \in R_I$. Otherwise, all summands are zero. Thus,

$$\bar{v}_{R,j} = \begin{cases} v_{b(j),j} & \text{if } b(j) \in R_I, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $b(j) \in R_I$ if and only if $j \in R$, giving

$$\bar{v}_{R,j} = \begin{cases} v_{b(j),j} & \text{if } j \in R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

This formula can be simplified by using Theorem 4.2, which describes a representation of the block Jacobi method.

In the proof of Theorem 4.2 we defined a vector $\bar{v}$ by

$$\bar{v} = \sum_{i=1}^{m} v_i \,.$$

This vector has the property that $\bar{v}_j = v_{b(j),j}$. Combining this fact with equation (4.31) we find that

$$\bar{v}_{R,j} = \begin{cases} \bar{v}_j & \text{if } j \in R, \\ 0 & \text{otherwise.} \end{cases}$$

Using the definition of the matrix Z_R we obtain that

$$\bar{v}_R = Z_R \bar{v} \,.$$

We know from Theorem 4.2 that $\bar{v} = D^{-1} u_k$; simplifying the previous equation to

$$\bar{v}_R = Z_R D^{-1} u_k \,,$$

which proves (4.28a).

Equation (4.28b) can be shown in a similar way and the other statements of this theorem follow from (4.28). □

We now want to analyze the red-black block Jacobi method. For this reason, we give the stencil representations for all operators that occur in the error propagators of the two steps of the method (4.29) assuming that we are dealing with a grid-based problem. The involved operators are the identity I, the diagonal operator D, the filtering operators Z_R and Z_B, and the operator A. All operators have already been considered and their stencils are known, except the operators Z_R and Z_B. Their stencils are

$$s_{Z_R,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } y = 0 \text{ and } \mathbf{x} \in R, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad s_{Z_B,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } y = 0 \text{ and } \mathbf{x} \in B, \\ 0 & \text{otherwise,} \end{cases} \tag{4.32}$$

which is easily obtained from the definition of Z_R and Z_B. Using these formulas we can perform an LFA of the red-black block Jacobi method.

Analysis of the Red-Black Block Jacobi Method

We want to analyze the red-black block Jacobi method by LFA. As already discussed in Section 4.2.1, which was about the analysis of the plain block Jacobi method, we can only analyze the method for certain choices of the partition $S_\mathbf{i}$. We choose the blocks to correspond to regular rectangles of grid points, i.e.,

$$S_\mathbf{i} = \mathbf{i} \cdot \mathbf{n} + T_\mathbf{n} \quad \text{where} \quad T_\mathbf{n} = \{\mathbf{h} \cdot \mathbf{k} : \mathbf{0} \leq \mathbf{k} < \mathbf{n}\}, \tag{4.24 revisited}$$

where $\mathbf{n}$ is the block size. The same choice was used for analyzing the block Jacobi method. In addition, we define the sets R_I and B_I by

$$\begin{aligned} R_I &= \{k \in \mathbb{Z}^d : k_1 + \cdots + k_d \text{ even}\} \quad \text{and} \\ B_I &= \{k \in \mathbb{Z}^d : k_1 + \cdots + k_d \text{ odd}\}. \end{aligned}$$

These set are very similar to the definition of the red and black points for the point-wise red-black Jacobi method (3.31). The difference is that the present definition assigns a color to a whole block of grid point, instead of to individual grid points.

The set of all individual red grid points is $R = \bigcup_{\mathbf{i} \in R_I} S_\mathbf{i}$, while the set of all black points is $B = \bigcup_{\mathbf{i} \in R_B}$. Figure 4.13 shows the red and black grid points for $\mathbf{n} = (2, 3)$. We see that the red and black points form red and black rectangles of grid points. Furthermore, the pattern is periodic with period $2\mathbf{n}$. Having these definitions in place, we can now build an LFA for this method.

We start with the error propagator of the method

$$E_{\text{RBBJ}} = (I - \omega Z_B D^{-1} A)(I - \omega Z_R D^{-1} A), \tag{4.33}$$

which we derived in Theorem 4.6. For our analysis we need to formulate all involved operators on the infinite grid $G_\mathbf{h}$. Like in the case where we analyzed the block Jacobi method, we assume that the operator A is given by a constant stencil on the whole grid. In this case D is represented by the stencil (4.23), and the choice of the partition ensured that the stencil of D is periodic, as discussed in the section about the block Jacobi method. It remains to find a representation for Z_R and Z_B.

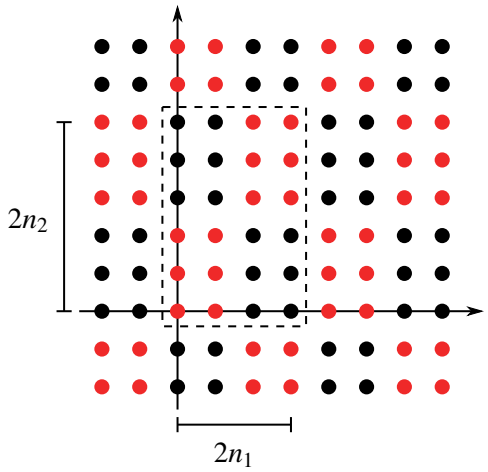


Figure 4.13.: Red and black blocks used in the red-black block Jacobi method.

We let Z_R and Z_B be given on the whole grid $G_{\mathbf{h}}$ by the stencils from (4.32). From this equation we see that the stencils s_{Z_R} and s_{Z_B} consist of two constant stencils: the identity stencil, which is applied at all red/black points, and the zero stencils, which is applied at all black/red points. We noted earlier that the red-black pattern, introduced above, is periodic with period $2\mathbf{n}$. Therefore, the stencils of Z_R and Z_B are periodic with the same period.

In summary all operators in the error propagator 4.33 given by a constant stencil or a periodic stencil. Consequently, all operators have a Fourier matrix symbol. The operator A has a $\mathbf{1} \times \mathbf{1}$, the operator D has a $\mathbf{n} \times \mathbf{n}$, and the operators Z_R and Z_B have a $2\mathbf{n} \times 2\mathbf{n}$ Fourier matrix symbol. Therefore, using a proper expansion of the matrix symbols (Theorem 3.37) gives that E_{RBBJ} has a $2\mathbf{n} \times 2\mathbf{n}$ Fourier matrix symbol. We can now compute this symbol to analyze a concrete application.

Results

We consider the discrete Poisson equation in 2D (1.48) and ask the question whether the red-black block Jacobi method is a suitable smoother for this problem. Furthermore, we want to know how increasing the block size $\mathbf{n}$, affects the method. We start with the first question.

A plot of the frequency emission for $\omega = 1$ is show in Figure 4.14. We see a similar behavior as with the block Jacobi method. The badly damped frequencies move closer to the corners when we increase the block size. As the frequencies in the corners correspond to wave functions that are slowly varying, we conclude that the red-black block Jacobi method is a suitable smoother. Furthermore, as increasing the block size improves the smoothing effect, we suppose that the method is also well suited for aggressive coarsening.

We check this hypothesis by conducting a two-grid analysis. The spectral radii of the two-grid error propagator for $\omega = 1$ can be found in Table 4.5. We see that the red-black block Jacobi method has a behavior similar to the block Jacobi method. The increase of the spectral radius that occurs when the coarsening range grows can be partially compensated

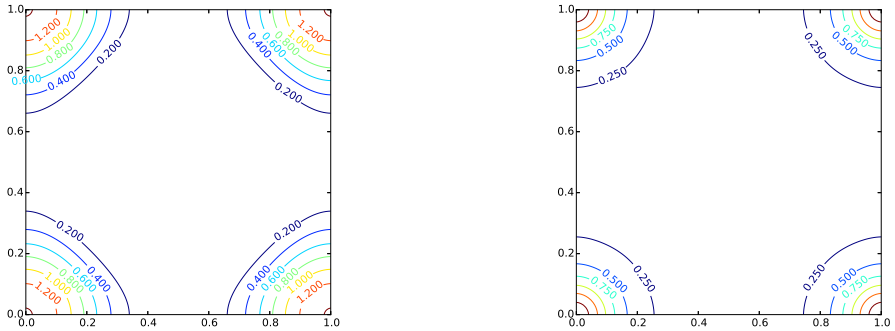


Figure 4.14.: Frequency emission of the red-black Jacobi method (left) and the 4×4 red-black block Jacobi method (right).

coarsening	block size				
	1	2	4	6	8
2	0.072	0.033	0.028	0.024	0.015
4	0.51	0.3	0.11	0.18	0.1
6	0.73	0.55	0.39	0.18	0.3
8	0.81	0.67	0.51	0.47	0.25

Table 4.5.: Spectral radius of the two-grid method with aggressive coarsening and red-black block Jacobi smoothing.

by increasing the block size of the smoother. In addition, we see that the red-black block Jacobi method with fixed ω has better convergence rates than the block Jacobi method with optimally chosen ω.

4.2.4. Numerical Evaluation

Similar to Section 4.1.5 we want to evaluate the quality of the local Fourier analysis predictions. Our analysis assumes an infinite grid and in real applications we only compute with a finite grid.

We pick a 96×96 grid with periodic boundary conditions and run the two-grid method with different smoothing configurations. Table 4.6 shows the results for the Jacobi smoother with weight $\omega = 0.8$, Table 4.7 shows the results for the block Jacobi smoother with optimal weight, and Table 4.8 shows the results for the red-black block Jacobi smoother. Comparing these Tables to the LFA predictions in Table 4.3, 4.4 and 4.5, we see that the predictions are quite accurate.

Coarsening	Block Size				
	1	2	4	6	8
2	0.34	0.29	0.24	0.24	0.23
4	0.72	0.60	0.34	0.41	0.32
6	0.82	0.75	0.65	0.44	0.54
8	0.84	0.80	0.73	0.71	0.53

Table 4.6.: Spectral radius of the two-grid method with aggressive coarsening and block Jacobi smoothing for a finite grid.

Coarsening	block size				
	1	2	4	6	8
2	0.34 (0.80)	0.23 (0.74)	0.21 (0.75)	0.20 (0.75)	0.22 (0.75)
4	0.70 (0.93)	0.61 (0.88)	0.34 (0.79)	0.38 (0.84)	0.32 (0.80)
6	0.81 (0.96)	0.74 (0.94)	0.60 (0.90)	0.42 (0.83)	0.49 (0.87)
8	0.87 (0.98)	0.80 (0.96)	0.71 (0.93)	0.67 (0.92)	0.50 (0.86)

Table 4.7.: Spectral radius of the two-grid method with aggressive coarsening and block Jacobi smoother with optimal weight (parenthesis) for a finite grid.

Coarsening	Block Size				
	1	2	4	6	8
2	0.06	0.02	0.01	0.01	0.01
4	0.50	0.30	0.06	0.14	0.05
6	0.71	0.53	0.38	0.14	0.26
8	0.78	0.68	0.54	0.46	0.22

Table 4.8.: Spectral radius of the two-grid method with aggressive coarsening and red-black block Jacobi smoothing for a finite grid.

4.3. Literature and Contributions

We considered two applications in this chapter: a PDE with a jumping coefficient and block smoothers.

The PDE with a jumping coefficient was discretized using the finite volume method. The monographs [44, 68, 73] contain introductions to this method. Furthermore, the multigrid method to solve the finite volume system was introduced in [3, 17].

The LFA of the jumping coefficient PDE multigrid method is an original result of this thesis. Nevertheless, the method has already been analyzed using SAMA [23], which is related to LFA. SAMA considers a tensor product of a finite and an infinite position space. Then, the infinite space is Fourier transformed, while the finite space remains unchanged. Hence, the problem is partly analyzed in Fourier space and partly analyzed in position space. In contrast, the analysis of this thesis is a pure Fourier space analysis.

Block smoothers are, e.g., treated in [59]. The Fourier analysis of these smoothers, also in combination with aggressive coarsening, is an original contribution of this thesis.

5. Automating Fourier Analysis

There are applications in which Fourier matrix symbols have many rows and columns, which has been the case, for example, for some of the applications described in Chapter 4. Computations involving such large symbols require a lot of work. Consequently, we want to automate these computations via software, in the following way.

We describe the error propagator E by entering a formula that describes it. This formula contains a set of further operators, and each of these then needs to be specified, e.g., by giving a stencil. Then the software computes the Fourier matrix symbol[1] $\hat{e}$ of the error propagator. After this computation we can ask the software to use the symbol to give us $r(E)$, $\|E\|$, and to plot $\mathcal{R}_{\mathbf{m}}^{-1}\mathrm{rn}(\hat{e})$, or $\mathcal{R}_{\mathbf{n}}^{-1}\mathrm{cn}(\hat{e})$.

As an example, consider the two-grid method below, given by

$$E_{\mathrm{TG}} = E_{\mathrm{J}}^{\nu_2}(I - PA_H^{-1}RA_h)E_{\mathrm{J}}^{\nu_1}\,,$$

where E_{J} is the error propagator of the Jacobi method, i.e.,

$$E_{\mathrm{J}} = I - \omega D^{-1}A_h\,.$$

These are the two formulas that we need to enter. They involve the scalars ω, ν_1, ν_2, and the operators A_h, A_H, D^{-1}, P, and R. The scalars are easily specified by just giving their values. The operators A_h and A_H are given by their stencils, and the operator D^{-1} is obtained from the stencil of A_h. We assume for the operators R and P that they can be written as a composition of a stencil operator and a injection interpolation or restriction, i.e., $R = R_{\mathrm{inj}}R_{\mathrm{st}}$, and $P = P_{\mathrm{st}}P_{\mathrm{inj}}$. Thus, additionally we need to specify these two formulas and the stencils of R_{st}, and P_{st}, which completes the specification of the problem and the software should be able to compute the Fourier matrix symbol $\hat{e}_{\mathrm{TG}}$ of E_{TG}. We shall now discuss how to realize such a software.

5.1. Constant Stencils

Many operators can be described by a stencil. Therefore, we need a way to store and manipulate stencils on a computer.

A stencil s is a mapping from $G_{\mathbf{h}}$ to $\mathbb{C}$, which assigns to every *offset* from $G_{\mathbf{h}}$ a *coefficient*. As there are infinitely many offsets we would need to store infinitely many coefficients if we were to store arbitrary stencils. All applications that we have considered so far, however, do not need arbitrary stencils. In all applications all stencil entries, except a few, were zero. Consequently, we only need to consider stencils with finitely many non-zero entries; implying

[1] Actually, the software computes only an approximation to the symbol for reasons we give in Section 5.3.

that we only need to store the non-zero ones. To work with such stencils, we specify a basic set of operations.

We need to read and write the non-zero entries, we need to iterate over the non-zero entries, and we need to obtain the step-size of the stencil. More precisely, when s is the storage of a stencil, we denote the access of the stencil entries by $s(\mathbf{y})$, where $\mathbf{y}$ is the offset. We denote the iteration of the non-zero entries by:

```
1  for all (y, v) in NonZeros(s)
2      // use the offset y and coefficient v
```

We denote the step-size of a by $a.\mathit{step\text{-}size}$. Furthermore, we assume the existence of a function ZeroStencil that returns a stencil whose entries are all zero. Using these definitions we can write down some basic stencil algorithms.

The first basic algorithm is the addition of two stencils. If s_A and s_B are the stencils of operators A and B, then the stencil s_C is the stencil of the operator $C := A + B$. From the definition of stencil operators (1.52) we get that

$$s_C(\mathbf{y}) = s_A(\mathbf{y}) + s_B(\mathbf{y}) \quad \text{for} \quad \mathbf{y} \in G_{\mathbf{h}}\,.$$

Making use of this formula and the fact that both stencils have only finitely many entries, we obtain the algorithm for the stencil addition.

```
Stencil-Add(s_A, s_B)
1  s_C ← ZeroStencil(s_A.step-size)
2  for all (y, v) in s_A
3      s_C(y) ← s_A(y)
4  for all (y, v) in s_B
5      s_C(y) ← s_A(y) + s_C(y)
```

The second basic algorithm that we want to discuss, is the stencil multiplication. If s_A and s_B are the constant stencils of two operators A and B, then there is a stencil s_C that represents the operator $C := BA$, as the next theorem shows.

Theorem 5.1. *Let A and B be stencil operators with stencils s_A and s_B. Then $C := BA$ is a stencil operator with stencil*

$$s_C(\mathbf{z}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s_B(\mathbf{y}) \cdot s_A(\mathbf{z}-\mathbf{y})\,. \tag{5.1}$$

We call s_C the product of the stencils s_A and s_B.

Proof. We shall prove the assertion by writing BAu in the form of (1.52), i.e.,

$$[BAu](\mathbf{x}) = \sum_{\mathbf{z}\in G_{\mathbf{h}}} s_C(\mathbf{z}) \cdot u(\mathbf{x}+\mathbf{z})\,. \tag{5.2}$$

We start by using the definition of the operator B:

$$[B(Au)](\mathbf{x}) = \sum_{\mathbf{y}\in G_{\mathbf{h}}} s_B(\mathbf{y}) \cdot [Au](\mathbf{x}+\mathbf{y})$$

In this equation we insert the definition of A, and obtain that

$$\begin{aligned}[B(Au)](\mathbf{x}) &= \sum_{\mathbf{y}\in G_\mathbf{h}} s_B(\mathbf{y}) \cdot \left(\sum_{\mathbf{y}'\in G_\mathbf{h}} s_A(\mathbf{y}') \cdot u(\mathbf{x}+\mathbf{y}+\mathbf{y}') \right) \\ &= \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_B(\mathbf{y}) \cdot s_A(\mathbf{y}') \cdot u(\mathbf{x}+\mathbf{y}+\mathbf{y}') \,.\end{aligned}$$

Substituting $\mathbf{y}'$ by $\mathbf{z}-\mathbf{y}$ gives

$$\begin{aligned}[B(Au)](\mathbf{x}) &= \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{z}\in G_\mathbf{h}} s_B(\mathbf{y}) \cdot s_A(\mathbf{z}-\mathbf{y}) \cdot u(\mathbf{x}+\mathbf{z}) \\ &= \sum_{\mathbf{z}\in G_\mathbf{h}} \left(\sum_{\mathbf{y}\in G_\mathbf{h}} s_B(\mathbf{y}) \cdot s_A(\mathbf{z}-\mathbf{y}) \right) \cdot u(\mathbf{x}+\mathbf{z}) \,.\end{aligned}$$

By comparing the last equation with the stencil formula for C (5.2), we see that the term in parenthesis is the stencil s_C of C. □

We cannot directly implement the formula for the stencil multiplication (5.1), as it involves the sum over infinitely many grid points. To overcome this problem let us rewrite the equation; giving

$$s_C(\mathbf{z}) = \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_B(\mathbf{y}) \cdot s_A(\mathbf{y}'-\mathbf{y}) \cdot \delta_{\mathbf{z}\mathbf{y}'} \,.$$

Substituting $\mathbf{y}'+\mathbf{y}$ for $\mathbf{y}'$ yields

$$s_C(\mathbf{z}) = \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_B(\mathbf{y}') \cdot s_A(\mathbf{y}) \cdot \delta_{\mathbf{z}(\mathbf{y}+\mathbf{y}')} \,.$$

We see that the summands are only non-zero if $s_A(\mathbf{y})$, $s_B(\mathbf{y}')$, and $\delta_{\mathbf{z}(\mathbf{y}+\mathbf{y}')}$ are simultaneously non-zero. Thus, to compute $s_C(\mathbf{z})$, we just need to loop over the non-zero entries of a_A and s_B in a nested loop. However, to compute all non-zero entries of s_C, we have to make some further manipulations.

Let $s_\mathbf{z}$ be the stencil that is one at $\mathbf{z}$ and zero everywhere else, i.e., $s_\mathbf{z}(\mathbf{y}) = \delta_{\mathbf{y}\mathbf{z}}$. With this definition we can write s_C as

$$\begin{aligned}s_C &= \sum_{\mathbf{z}\in G_\mathbf{h}} s_\mathbf{z} \cdot s_C(\mathbf{z}) \\ &= \sum_{\mathbf{z}\in G_\mathbf{h}} s_\mathbf{z} \cdot \left(\sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_B(\mathbf{y}') \cdot s_A(\mathbf{y}) \cdot \delta_{\mathbf{z}(\mathbf{y}+\mathbf{y}')} \right) \\ &= \sum_{\mathbf{z}\in G_\mathbf{h}} \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_\mathbf{z} \cdot s_B(\mathbf{y}') \cdot s_A(\mathbf{y}) \cdot \delta_{\mathbf{z}(\mathbf{y}+\mathbf{y}')} \,.\end{aligned}$$

The summands of this sum are only non-zero if $\mathbf{z} = \mathbf{y}+\mathbf{y}'$. Consequently,

$$s_C = \sum_{\mathbf{y}\in G_\mathbf{h}} \sum_{\mathbf{y}'\in G_\mathbf{h}} s_{\mathbf{y}+\mathbf{y}'} \cdot s_B(\mathbf{y}') \cdot s_A(\mathbf{y}) \,.$$

	list	dense	balanced trees
random read access	$O(n)$	$O(1)$	$O(\log(n))$
random write access	$O(n)$	$O(n)$	$O(\log n))$
iterate elements	$O(n)$	$O(n)$	$O(n)$

Table 5.1.: Running time of the stencil operations for different implementations in dependence of the number of entries of the stencil n.

The last equation implies that in order to compute the stencil s_C, we just have to loop over the non-zero entries of s_A and s_B in a nested loop and add the product $s_B(\mathbf{y}') \cdot s_A(\mathbf{y})$ to the stencil s_C at offset $\mathbf{y} + \mathbf{y}'$. This procedure is written down in Stencil-Multiply.

```
Stencil-Multiply(s_B, s_A)
1  s_C ← Zero-Stencil(s_A.step-size)
2  for all (y', d') in NonZeros(s_B)
3      for all (y, d) in NonZeros(s_A)
4          s_C(y + y') ← s_C(y + y') + d · d'
5  return s_C
```

Until now, we have not discussed how to actually implement the stencil interface that we discussed earlier. There are different ways for the implementation, and we shall briefly comment on their advantages and disadvantages.

The first possible implementation uses just a list to store the offsets together with their coefficients. The advantage of this implementation is that it is easy to implement and iterating over all entries is fast. The disadvantage is that accessing an element with a random offset is rather slow, as it requires the iteration though all elements. However, for a small number of entries this behavior is not a problem; the running time per operation is still very low, due to the simple implementation.

The second possible implementation needs a bounding box that surrounds all non-zero entries of the stencil. The entries inside the bounding box are then stored in a dense dD array. The advantage is that accessing an element with a random offset is fast, as long as the offset is inside the bounding box. The disadvantage is that a write access outside the bounding box requires reallocating the memory, which is slow. Furthermore, when the bounding box contains only a few elements, then a lot of storage space is wasted and iterating over the non-zero entries is relatively slow, as most of the stored elements are zero but need to be considered.

The third possible implementation uses balanced binary trees such as red-black trees [16]. We use the offsets as keys and the coefficients as data elements. The advantage of this data structure is that it is reasonably fast for all operations. The disadvantage is that it is more complicated to implement and the running time for a single operation is larger than for the list implementation.

In summary, the list implementation is suitable for situations with a low number of entries, while the balanced tree implementation is a good choice for large numbers of elements. The

dense array implementation is only useful in special situations. Table 5.1 gives an overview of the running times for the different operations and implementations. As we now have a set of operations for constant stencil operators, we can use them to work with periodic stencils.

5.2. Periodic Stencils

Periodic stencils are useful for the analysis of various applications, as we saw in Chapter 4. A periodic stencil is a family $\{s\}_{\mathbf{x}\in G_{\mathbf{h}}}$ of constant stencils $s_{\mathbf{x}}$ that is periodic with a period $\mathbf{n} \in \mathbb{Z}^d$, i.e.,

$$s_{\mathbf{x}} = s_{\mathbf{x}+\mathbf{k}\cdot(\mathbf{h}\cdot\mathbf{n})} \quad \text{for all} \quad \mathbf{x} \in G_{\mathbf{h}},\ \mathbf{k} \in \mathbb{Z}^d\,. \tag{3.30 revisited}$$

To work with periodic stencils on a computer we need a way to store and manipulate them on a machine.

To store periodic stencils, we make use of the fact that a periodic stencil $\{s\}_{\mathbf{x}\in G_{\mathbf{h}}}$ with period $\mathbf{n}$ is completely determined by the values $s_{\mathbf{x}}$ for $\mathbf{0} \le \mathbf{x} < \mathbf{n}$. Thus, to store a periodic stencil, it is sufficient to store a dD array that contains the constant stencils of $s_{\mathbf{x}}$ for those values of $\mathbf{x}$. More precisely, if a is such a dD stencil, we would access the entry $s_{\mathbf{x}}$ by accessing the entry $a_{(\mathbf{x} \bmod (\mathbf{n}\cdot\mathbf{h}))}$ of the array a.

To use periodic stencils in algorithms, we need to introduce some notation. If s is the handle to the storage of some periodic stencil, we denote the period of s by $s.\mathit{period}$, the step-size of s by $s.\mathit{step\text{-}size}$, and the constant stencil at $\mathbf{x}$ by $s_{\mathbf{x}}$. Furthermore, we assume that there exists a procedure Allocate-Periodic-Stencil($\mathbf{h}, \mathbf{n}$) that returns a handle to a free memory location, which stores a periodic stencil with step-size $\mathbf{h}$ and period $\mathbf{n}$.

A simple example for using this notation is the Periodic-Stencil-Add procedure that adds two periodic stencils. For simplicity we assume that the two stencils have the same period.

```
Periodic-Stencil-Add(s_A, s_B)
1  s_C ← Allocate-Periodic-Stencil(s_A.step-size, s_A.period)
2  for k ← 0 to s_A.period − 1
3      s_{C,k} ← s_{A,k} + s_{B,k}
```

Note that the addition in Line 3 is actually the addition of two constant stencils. Thus, we would have to use the addition algorithm that was discussed in Section 5.1.

5.3. Approximating Fourier Symbols

We shall discuss the representation of symbols on a computer and how to perform computations involving them. A symbol is a function $a \in L_\infty(\Theta_{\mathbf{h}})$. These symbols can be applied to functions $f \in L_2(\Theta_{\mathbf{h}})$ by pointwise multiplication. The problem is, however, that the spaces $L_\infty(\Theta_{\mathbf{h}})$ and $L_2(\Theta_{\mathbf{h}})$ both have infinite dimensions. Thus, even if we know a basis for these spaces we need to store infinitely many complex numbers to represent a general function from these spaces. Such an amount of storage, of course, is not available on typical computers. Therefore, we content ourselves by approximating functions from these spaces.

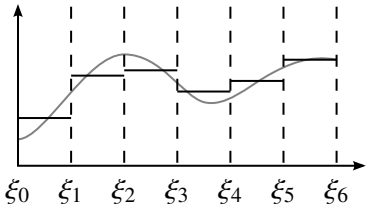


Figure 5.1.: Approximating a function by a function from T_r with $r = 6$.

5.3.1. Representing Symbols

For simplicity, let us first introduce an approximation to symbols in 1D, and show later how this idea can be generalized to approximate symbols of arbitrary dimensions. We start by choosing an integer $r \in \mathbb{N}$, which we call the *resolution* of the approximation. Then we split the interval Θ_h into r equally sized subintervals, i.e., we define

$$\xi_k := \tfrac{2\pi}{hr} \cdot k \tag{5.3}$$

and consider the intervals $Q_k := [\xi_k, \xi_{k+1})$ for $k = 0, \dots, r-1$. Finally, we approximate a function f from $L_\infty(\Theta_\mathbf{h})$ or $L_2(\Theta_\mathbf{h})$ by a function $\bar{f}$ that is constant on every interval Q_k, as shown in Figure 5.1.

Definition 5.2. A function $\bar{f} : \Theta_h \to \mathbb{C}$ has the *resolution* $r \in \mathbb{N}$ if $\bar{f}$ restricted to Q_k is constant for all $i = 0, \dots, r-1$. We denote the set of all functions with resolution r by T_r, i.e.,

$$T_r := \{\bar{f} : \Theta_h \to \mathbb{C} : \bar{f} \text{ restricted to } Q_k \text{ is constant}\}.$$

Furthermore, we say that a function $\bar{f} : \Theta_h \to \mathbb{C}$ has a *finite resolution* if there exists an $r \in \mathbb{N}$ such that $\bar{f}$ has the resolution r. Otherwise we say that the *resolution of $\bar{f}$ is infinite*.

Functions with resolution r can be represented by a finite set of numbers, which we can store on a computer. To show this fact, consider the functions

$$q_k(\theta) := \begin{cases} 1 & \text{if } \theta \in Q_k, \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 0, \dots, r-1.$$

It is easy to see that every function $\bar{f} \in T_r$ can be uniquely written as

$$\bar{f} = \sum_{k=0}^{r-1} f_k \cdot q_k \tag{5.4}$$

for some $f_0, \dots, f_{r-1} \in \mathbb{C}$. This representation implies that the functions q_k for $k = 0, \dots, r-1$ form a basis of T_r, and therefore every function in T_r is completely determined by the coefficients $f_0, \dots, f_{r-1}$. Now, these coefficients just have to be stored on a computer to represent functions from T_r.

Definition 5.3. Let $\bar{f} \in T_r$. The coefficients $f_0, \ldots, f_{r-1}$ from (5.4) are the *resolution r representation* of the function $\bar{f}$.

To compute the formulas derived in previous chapters, we need the ability to add two functions from T_r, to multiply a function from T_r with a scalar, and to multiply two functions from T_r. These operations can be realized purely in terms of their resolution r representations. Assume that we have $\lambda \in \mathbb{C}$, $\bar{f}, \bar{g} \in T_r$ with corresponding coefficients f and g. Then

$$\lambda \bar{f} \quad \text{corresponds to} \quad \left(\lambda f_k\right)_{k=0}^{r-1}, \tag{5.5a}$$

$$\bar{f} + \bar{g} \quad \text{corresponds to} \quad \left(f_k + g_k\right)_{k=0}^{r-1}, \tag{5.5b}$$

$$\bar{f} \cdot \bar{g} \quad \text{corresponds to} \quad \left(f_k \cdot g_k\right)_{k=0}^{r-1}. \tag{5.5c}$$

The first two relations (5.5a) and (5.5b) are, of course, the usual vector operations, while the third relation (5.5c) is the coefficient-wise product of two vectors. These three relations allow us to perform computations with functions from T_r on the computer.

Let us address the question if the functions from the space T_r are suitable to approximate functions from $L_\infty(\Theta_h)$ and $L_2(\Theta_h)$. It is easy to see that $T_r \subseteq L_\infty(\Theta_h)$ and $T_r \subseteq L_2(\Theta_h)$. For this reason we are certain to work only with functions that are permitted within our framework. We continue by considering the quality of the approximation. It is characterized by the theorem below.

Theorem 5.4. *The set $\bigcup_{r=0}^{\infty} T_r$ is dense in $L_2(\Theta_h)$ and in $L_\infty(\Theta_h)$.*

Theorem 5.4 implies that for a given function from $L_\infty(\Theta_h)$ or $L_2(\Theta_h)$ we can find a function from T_r that is arbitrarily close to the first one, under the assumption that we choose a suitable, i.e., large, value for r.

Proof of Theorem 5.4. It is known that the set of continuous functions is dense in $L_2(\Theta_h)$. This fact implies that it is sufficient to show that $\bigcup_{r=0}^{\infty} T_r$ is dense in the set of continuous functions.

To show this assertion, we let $\epsilon > 0$ be given and assume that $f : \Theta_h \to \mathbb{C}$ is continuous. The set Θ_h is compact and therefore f is uniformly continuous. Thus, there exists δ such that for all θ, θ' with $|\theta' - \theta| < \delta$ the bound $|f(\theta') - f(\theta)| < \epsilon$. We use the value of δ to construct an approximation for f.

We choose $r \in \mathbb{N}$ such that the length of the interval Q_i is smaller than δ. Furthermore, we define $\bar{g} \in T_r$ by

$$\bar{g}(\theta) = f(\xi_i) \quad \text{for} \quad \theta \in Q_i\,,$$

where ξ_i was defined in (5.3). Then

$$|\bar{g}(\theta) - f(\theta)| = |f(\xi_i) - f(\theta)| \quad \text{for} \quad \theta \in Q_i\,.$$

As $\theta, \xi_i \in Q_i$ and the length of the interval Q_i is smaller than δ, we have

$$|\xi_i - \theta| < \delta \quad \text{for} \quad \theta \in Q_i\,.$$

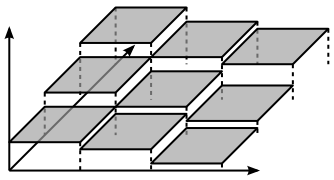


Figure 5.2.: A function from the set $T_{\mathbf{r}}$ with $\mathbf{r} = (3,3)^T$.

Using the fact that f is uniformly continuous, this inequality implies that

$$|\bar{g}(\theta) - f(\theta)| = |f(\xi_i) - f(\theta)| < \epsilon \quad \text{for} \quad \theta \in Q_i\,.$$

Hence, $\|\bar{g} - f\|_2 \leq \|\bar{g} - f\|_\infty < \epsilon$. □

Let us now extend this approximation to higher dimensions. Here, we cannot split the set $\Theta_{\mathbf{h}}$ into intervals. Instead, we partition $\Theta_{\mathbf{h}}$ into equally sized rectangles in 2D, rectangular cuboids in 3D, and their generalizations in higher dimensions. In this case, the resolution is given by a vector $\mathbf{r}$, whose entry r_j is the resolution in θ_j-direction. Thus, the subsets are defined by

$$Q_{\mathbf{k}} := \{\theta \in \mathbb{R}^d : \theta_j \in \tfrac{2\pi}{h_j r_j}[k_j, k_j + 1) \text{ for } j = 1, \ldots, d\} \quad \text{for} \quad \mathbf{k} = \mathbf{0}, \ldots, \mathbf{r} - \mathbf{1}\,.$$

Introducing the notation

$$\xi_{\mathbf{k}} := \frac{2\pi}{\mathbf{h} \cdot \mathbf{r}} \cdot \mathbf{k}, \tag{5.6}$$

we can write the previous equation more compactly as

$$Q_{\mathbf{k}} = \{\theta \in \mathbb{R}^d : \xi_{\mathbf{k}} \leq \theta < \xi_{\mathbf{k+1}}\}\,. \tag{5.7}$$

With this definition, we can generalize the set of finite resolution functions to arbitrary dimensions.

Definition 5.5. Let $\mathbf{r} \in \mathbb{N}^d$ and $Q_{\mathbf{k}}$ be given by (5.7). A function $\bar{f} : \Theta_{\mathbf{h}} \to \mathbb{C}$ has the *resolution* $\mathbf{r}$ if $\bar{f}$ restricted to $Q_{\mathbf{k}}$ is constant for all $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{r} - \mathbf{1}$. We denote the set of all functions with resolution $\mathbf{r}$ by $T_{\mathbf{r}}$, i.e.,

$$T_{\mathbf{r}} := \{\bar{f} : \Theta_h \to \mathbb{C} : \bar{f} \text{ restricted to } Q_{\mathbf{k}} \text{ is constant}\}\,.$$

Furthermore, we say that a function $\bar{f} : \Theta_{\mathbf{h}} \to \mathbb{C}$ has a *finite resolution* if there exists an $\mathbf{r} \in \mathbb{N}$ such that $\bar{f}$ has the resolution $\mathbf{r}$. Otherwise we say that the *resolution of $\bar{f}$ is infinite*.

An example for a function from $T_{\mathbf{r}}$ is shown in Figure 5.2. In the 1D case it was useful to have a basis representation for such a function.

The generalization of the basis functions for higher dimensions is straightforward. They are defined as

$$q_{\mathbf{k}}(\theta) := \begin{cases} 1 & \text{if } \theta \in Q_{\mathbf{k}}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

Thus, every function $\bar{f} \in T_{\mathbf{r}}$ can be written uniquely as

$$\bar{f} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} f_{\mathbf{k}} \cdot q_{\mathbf{k}} \,. \tag{5.9}$$

Note that we now use multi-indices for the coefficients.

Definition 5.6. Let $\bar{f} \in T_{\mathbf{r}}$. The coefficients $f_{\mathbf{0}}, \ldots, f_{\mathbf{r}-\mathbf{1}}$ from (5.8) are the *resolution* $\mathbf{r}$ *representation* of the function $\bar{f}$.

The relations between the resolution $\mathbf{r}$ representation and operations on the resolution $\mathbf{r}$ functions are also very similar to the 1D case (5.5). Let $\lambda \in \mathbb{C}$, $\bar{f}, \bar{g} \in T_{\mathbf{r}}$ with corresponding coefficients f and g. Then

$$\lambda \bar{f} \quad \text{corresponds to} \quad \left(\lambda f_{\mathbf{k}}\right)_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}}, \tag{5.10a}$$

$$\bar{f} + \bar{g} \quad \text{corresponds to} \quad \left(f_{\mathbf{k}} + g_{\mathbf{k}}\right)_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}}, \tag{5.10b}$$

$$\bar{f} \cdot \bar{g} \quad \text{corresponds to} \quad \left(f_{\mathbf{k}} \cdot g_{\mathbf{k}}\right)_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}}. \tag{5.10c}$$

The only difference to (5.5) is the use of multi-indices.

Note that in higher dimensions it is also true that $T_{\mathbf{r}} \subseteq L_\infty(\Theta_{\mathbf{h}})$, $T_{\mathbf{r}} \subseteq L_2(\Theta_{\mathbf{h}})$, and that $T_{\mathbf{r}}$ is dense in $L_\infty(\Theta_{\mathbf{h}})$ and in $L_2(\Theta_{\mathbf{h}})$. The last statement can be shown by modifying the proof of Theorem 5.4.

In the remainder of this chapter, we want to discuss algorithms that implement the Fourier analysis that we introduced in the previous chapters. This analysis involves the computation with functions from $L_\infty(\Theta_{\mathbf{h}})$, and $L_2(\Theta_{\mathbf{h}})$ that we shall replace by computations with functions of finite resolution. We, therefore, need an algorithmic notation for working with resolution $\mathbf{r}$ functions.

Our algorithms need to store, access, and manipulate finite resolution functions. More precisely, they need to store the step-size $\mathbf{h}$, resolution $\mathbf{r}$, and coefficients of the resolution $\mathbf{r}$ representation (5.9). We assume the existence of a function Allocation-Function($\mathbf{h}, \mathbf{r}$) that takes the step-size and the resolution and returns allocated storage for a resolution $\mathbf{r}$ representation. If f is the return value of Allocate-Function($\mathbf{h}, \mathbf{r}$) then

- $f.$*step-size* should be the step-size $\mathbf{h}$,
- $f.$*resolution* should be the resolution $\mathbf{r}$, and
- $f_{\mathbf{k}}$ for $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{r} - \mathbf{1}$ should be the coefficients of the basis representation (5.9).

Using this definition, we can write our first algorithm.

The function Constant-Function returns a resolution $\mathbf{r}$ representation that is equal to a constant $c \in \mathbb{C}$ on the whole domain.

Constant-Function($\mathbf{h}, \mathbf{r}, c$)

```
1  f ← Allocate-Function(h, r)
2  for k ← 0 to r − 1
3      f_k ← c
4  return f
```

Using the relations (5.10), we can implement the basic operations on resolution **r** functions: the multiplication by a scalar, the addition of two functions, and the pointwise multiplication of two functions.

The SCALAR-MULTIPLY-FUNCTION procedure expects a scalar $\lambda \in \mathbb{C}$ and a resolution **r** representation of a function f, and it returns the resolution **r** representation of λf.

SCALAR-MULTIPLY-FUNCTION(λ, f)

```
1  g ← ALLOCATE-FUNCTION(f.step-size, f.resolution)
2  for k ← 0 to f.resolution
3      g_k ← λ · f_k
4  return g
```

The ADD-FUNCTIONS procedure expects two resolution **r** representations of two functions f and g, and it returns the resolution **r** representation of $f + g$.

ADD-FUNCTIONS(f, g)

```
1  u ← ALLOCATE-FUNCTION(f.step-size, f.resolution)
2  for k ← 0 to f.resolution
3      u_k ← f_k + g_k
4  return u
```

The POINTWISE-MULTIPLY-FUNCTIONS procedure expects two resolution **r** representations of two functions f and g, and it returns the resolution **r** representation of the pointwise product $f \cdot g$.

POINTWISE-MULTIPLY-FUNCTIONS(f, g)

```
1  u ← ALLOCATE-FUNCTION(f.step-size, f.resolution)
2  for k ← 0 to f.resolution
3      u_k ← f_k · g_k
4  return u
```

Note that all these algorithms work on resolution **r** *representations* and not on resolution **r** *functions*. Any function with resolution **r** also has the resolution $\mathbf{r} \cdot \mathbf{n}$ for any $\mathbf{0} < \mathbf{n} \in \mathbb{N}^d$. However, a resolution **r** representation is not automatically a resolution $\mathbf{r} \cdot \mathbf{n}$ representation. There is, of course, a resolution $\mathbf{r} \cdot \mathbf{n}$ representation that represents the same function, however, its computation is usually not desirable for the following reason.

In general, a function with low resolution approximates a desired function with less accuracy than a function with a high resolution. Thus, when we add a low resolution and a high resolution function, their sum will generally approximate the desired function equally well as if we computed with two low resolution functions. Thus, the computation of a high resolution function was a waste of resources, and therefore we shall only perform computations where the resolutions of the involved representations match appropriately.

For the following algorithms we shall simplify our notation. Calling the functions above by their name will make most algorithms verbose. Hence, we shall write

- $\lambda \cdot f$ instead of SCALAR-MULTIPLY-FUNCTIONS(λ, f),

- $f + g$ instead of ADD-FUNCTIONS(f, g), and
- $f \cdot g$ instead of POINTWISE-MULTIPLY-FUNCTIONS(f, g).

Using this notation, we can write the application of a symbol a to a function u, which is just the pointwise multiplication of the two functions, simply as:

APPLY-SYMBOL(a, u)
1 **return** $a \cdot u$

Another useful algorithm is the evaluation of the resolution $\mathbf{r}$ function $\bar{f}$ at a specific value of θ. It is, e.g., useful when we want to plot the function. To determine the function value at θ, we need to find the index $\mathbf{k}$ of the set $Q_{\mathbf{k}}$ such that $\theta \in Q_{\mathbf{k}}$. In this case, the function value is the coefficient $f_{\mathbf{k}}$. The index $\mathbf{k}$ is determined in the following lemma.

Lemma 5.7. *Let $\theta \in \Theta_{\mathbf{r}\cdot\mathbf{h}}$. If*

$$\mathbf{k} = \left\lfloor \tfrac{1}{2\pi}\theta \cdot \mathbf{r} \cdot \mathbf{h} \right\rfloor \tag{5.11}$$

then

$$\xi_{\mathbf{k}} \leq \theta < \xi_{\mathbf{k}+\mathbf{1}},$$

where $\xi_{\mathbf{k}}$ is given in (5.6)*.*

Proof. The choice of $\mathbf{k}$ in (5.11) implies

$$k_j \leq \tfrac{\theta_j r_j h_j}{2\pi} < k_j + 1.$$

From this inequality we get

$$\tfrac{2\pi}{r_j h_j} k_j \leq \theta_j < \tfrac{2\pi}{r_j h_j}(k_j + 1).$$

Note that the term on left-hand side and on the right-hand side are $\xi_{\mathbf{k},j}$ and $\xi_{(\mathbf{k}+\mathbf{1}),j}$. □

Using the formula for the index (5.11) we can define the EVALUATE-FUNCTION procedure.

EVALUATE-FUNCTION(f, θ)
1 $\mathbf{h} \leftarrow f.$*step-size*
2 $\mathbf{r} \leftarrow f.$*resolution*
3 $\mathbf{k} \leftarrow \left\lfloor \tfrac{1}{2\pi}\theta \cdot \mathbf{r} \cdot \mathbf{h} \right\rfloor$
4 **return** $f_{\mathbf{k}}$

5.3.2. Constant Stencils

Computing the Fourier symbol of a constant stencil can already give insight into the behavior of an iterative method. For example, in Section 1.5.3 we used it to show the smoothing properties of the Jacobi iteration when applied to the Poisson equation. Furthermore, the computation of Fourier symbols is an important building block in computing the Fourier *matrix* symbols of periodic stencils. Thus, we consider the approximation of Fourier symbols.

To begin with, recall Theorem 2.18, which states that the Fourier symbol of a constant stencil operator $\hat{a}$ is given by

$$\hat{a}(\theta) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{y} \rangle} . \qquad \text{(1.55 revisited)}$$

Note that we assumed that the stencil s is finite, i.e., the stencil has finitely many non-zero entries. It is easy to see that in this case $\hat{a}$ is continuous. With this fact in mind, we can discuss the approximation of the symbol $\hat{a}$ by finite resolution functions.

Consider the sets $Q_{\mathbf{k}}$ for $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{r} - \mathbf{1}$ corresponding to the resolution $\mathbf{r}$. If we increase the components of the resolution $\mathbf{r}$, then the sets $Q_{\mathbf{k}}$ become smaller. Since $\hat{a}$ is continuous, $\hat{a}$ is nearly constant when restricted to a set $Q_{\mathbf{k}}$ that is small enough. Hence, we can approximate $\hat{a}$ by a function $\bar{a}$ that is equal to a constant $a_{\mathbf{k}}$ on $Q_{\mathbf{k}}$, i.e., $\bar{a} \in T_{\mathbf{r}}$, which has the representation

$$\bar{a} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a_{\mathbf{k}} \cdot q_{\mathbf{k}} . \qquad (5.12)$$

As $\hat{a}$ is nearly constant on $Q_{\mathbf{k}}$, any function value $\hat{a}(\theta)$ for $\theta \in Q_{\mathbf{k}}$ is a reasonable choice for $a_{\mathbf{k}}$, which is the function value of $\bar{a}$ on $Q_{\mathbf{k}}$.

We choose $a_{\mathbf{k}}$ in the following way. Let $\theta_0 \in \Theta_{\mathbf{r}\cdot\mathbf{h}}$, we set

$$a_{\mathbf{k}} := \hat{a}(\xi_{\mathbf{k}} + \theta_0) \qquad (5.13)$$

for $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{r} - \mathbf{1}$. Figure 5.3 shows the function arguments $\xi_{\mathbf{k}} + \theta_0$ for a specific choice of θ_0. A small computation shows that the definition of $\xi_{\mathbf{k}}$ (5.6) and the definition of $Q_{\mathbf{k}}$ (5.7) imply

$$\xi_{\mathbf{k}} + \theta_0 \in Q_{\mathbf{k}} .$$

The vector θ_0 is called the *sampling offset*. Using (1.55), we can rewrite (5.13);

$$a_{\mathbf{k}} = \sum_{\mathbf{k} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \xi_{\mathbf{k}} + \theta_0, \mathbf{y} \rangle} . \qquad (5.14)$$

The procedure Stencil-Symbol computes an approximation to a Fourier symbol given by a constant stencil based on this equation.

```
Stencil-Symbol(s, r, θ_0)
1  a ← Allocate-Function(s.step-size, r)
2  for k ← 0 to r − 1
3      a_k ← 0
4      for all (y, v) in NonZeros(s)
5          a_k ← a_k + v · e^{i⟨ξ_k + θ_0, y⟩}
6  return a
```

In this procedure we make especially use of the assumption that s is finite.

We evaluate the quality of the approximation $\bar{a}$, given by the coefficients (5.14). We shall answer this question by deriving a bound on the difference between $\bar{a}$ and $\hat{a}$. We start by showing an auxiliary bound.

×	×	×
Q_{01} ×	Q_{11} ×	Q_{21} ×
Q_{00} ×	Q_{10} ×	Q_{20} ×

Figure 5.3.: The points at which the symbol $\hat{a}$ is evaluated. As θ_0 is the same for all set $Q_{\mathbf{k}}$, the distance to the corners of $Q_{\mathbf{k}}$ are the same giving a regular pattern at which $\hat{a}$ is evaluated.

Lemma 5.8. *For $t \in \mathbb{R}$ we have*

$$|1 - e^{\mathrm{i}t}| \leq |t| \,. \tag{5.15}$$

Proof. Interpreting $\mathbb{C}$ as a subset of $\mathbb{R}^2$, we can write

$$|1 - e^{\mathrm{i}t}| = \| \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} \| \,.$$

In other words, $|1 - e^{\mathrm{i}t}|$ is the distance of the points $(1,0)^T$ and $(\cos(t), \sin(t))^T$, as illustrated in Figure 5.4. This distance, however, is the *length* of the *shortest* path between these two points (see, e.g., [71]). Consequently, the length of *any* path connecting the two points must be greater or equal to the distance of the two points, i.e.,

$$|1 - e^{\mathrm{i}t}| \leq \left| \int_0^t \|\alpha'(t_0)\| \, \mathrm{d}t_0 \right|$$

for any curve α with $\alpha(0) = (1,0)^T$ and $\alpha(t) = (\cos(t), \sin(t))^T$. It is easy to see that the curve

$$\alpha(t) := \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}$$

fulfills this requirement. Hence, we have that

$$|1 - e^{\mathrm{i}t}| \leq \left| \int_0^t \|\alpha'(t_0)\| \, \mathrm{d}t_0 \right| = \left| \int_0^t \sqrt{\sin(t_0)^2 + \cos(t_0)^2} \, \mathrm{d}t_0 \right| = \left| \int_0^t 1 \, \mathrm{d}t_0 \right| = |t| \,,$$

where $|t|$ is the length of the curve α. □

Using this lemma, we can show a result that characterizes how fast the function $\hat{a}$ changes, which is done by deriving a bound for the difference of $\hat{a}$ evaluated for two different arguments.

Lemma 5.9. *Let $\hat{a}$ be the Fourier symbol of the operator given by the stencil s. Then*

$$|\hat{a}(\theta) - \hat{a}(\theta')| \leq \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(y)| \cdot |\langle \theta' - \theta, \mathbf{y} \rangle| \tag{5.16}$$

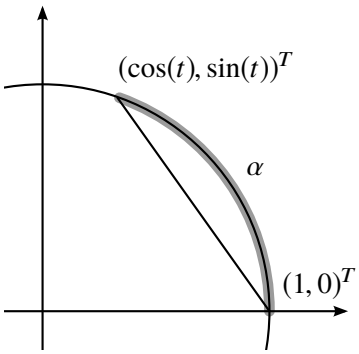


Figure 5.4.: The distance of the points $(1, 0)^T$ and $(\cos(t), \sin(t))^T$) is bounded by the length of the curve α.

Proof. We have from the definition of $\hat{a}$ (1.55) that

$$\begin{aligned} |\hat{a}(\theta) - \hat{a}(\theta')| &= |\sum_{\mathbf{y}\in G_{\mathbf{h}}} s(y) \cdot e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} - \sum_{\mathbf{y}\in G_{\mathbf{h}}} s(y) \cdot e^{\mathrm{i}\langle\theta',\mathbf{y}\rangle}| \\ &= |\sum_{\mathbf{y}\in G_{\mathbf{h}}} s(y) \cdot (e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} - e^{\mathrm{i}\langle\theta',\mathbf{y}\rangle})| \,. \end{aligned} \tag{5.17}$$

We can rewrite the term in parenthesis giving that

$$\begin{aligned} e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} - e^{\mathrm{i}\langle\theta',\mathbf{y}\rangle} &= e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} - e^{\mathrm{i}\langle\theta-\theta+\theta',\mathbf{y}\rangle} = e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} - e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} \cdot e^{\mathrm{i}\langle\theta'-\theta,\mathbf{y}\rangle} \\ &= e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} \cdot (1 - e^{\mathrm{i}\langle\theta'-\theta,\mathbf{y}\rangle}) \,. \end{aligned}$$

If we substitute this relation back into (5.17), we obtain:

$$\begin{aligned} |\hat{a}(\theta) - \hat{a}(\theta')| &= |\sum_{\mathbf{y}\in G_{\mathbf{h}}} s(y) \cdot e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle} \cdot (1 - e^{\mathrm{i}\langle\theta'-\theta,\mathbf{y}\rangle})| \\ &\leq \sum_{\mathbf{y}\in G_{\mathbf{h}}} |s(y)| \cdot |e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle}| \cdot |1 - e^{\mathrm{i}\langle\theta'-\theta,\mathbf{y}\rangle}| \,. \end{aligned}$$

Using that $|e^{\mathrm{i}\langle\theta,\mathbf{y}\rangle}| = 1$ and that we can bound $|1 - e^{\mathrm{i}\langle\theta'-\theta,\mathbf{y}\rangle}|$ by using (5.15), we obtain that

$$|\hat{a}(\theta) - \hat{a}(\theta')| \leq \sum_{\mathbf{y}\in G_{\mathbf{h}}} |s(y)| \cdot |\langle\theta' - \theta, \mathbf{y}\rangle| \,. \qquad \square$$

It is noteworthy that the bound (5.16) depends on the difference between θ and θ' but not on the absolute value of θ. Using this bound, we can bound the difference between our approximation $\bar{a}$ and the Fourier symbol $\hat{a}$.

Theorem 5.10. *Let $\hat{a}$ be the Fourier symbol of the stencil operator that is given by the stencil s. Furthermore, let $\bar{a} \in T_{\mathbf{r}}$ be given by the coefficients* (5.13)*. Then*

$$\|\hat{a} - \bar{a}\|_\infty \leq \|C(\theta_0)\| \cdot \sum_{\mathbf{y}\in G_{\mathbf{h}}} |s(\mathbf{y})| \cdot \|\mathbf{y}\| \,, \quad \textit{where} \quad C(\theta_0)_k := \tfrac{\pi}{h_k r_k} + |\tfrac{\pi}{h_k r_k} - \theta_{0,k}| \,. \tag{5.18}$$

Proof. As $\bar{a}$ is constant on $Q_{\mathbf{j}}$, we bound the difference of $\hat{a}$ and $\bar{a}$ for every $Q_{\mathbf{j}}$. For $\theta \in Q_{\mathbf{j}}$ we have

$$\bar{a}(\theta) = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a_{\mathbf{j}} \cdot q_{\mathbf{j}}(\theta) = a_{\mathbf{j}} = \hat{a}(\xi_{\mathbf{j}} + \theta_0),$$

by the definition of $\bar{a}$ (5.12), the basis functions $q_{\mathbf{j}}$ (5.8), and the coefficients $a_{\mathbf{j}}$ (5.13). Thus,

$$|\hat{a}(\theta) - \bar{a}(\theta)| = |\hat{a}(\theta) - \hat{a}(\xi_{\mathbf{j}} + \theta_0)| \quad \text{for} \quad \theta \in Q_{\mathbf{j}}\,.$$

Using this equation and Lemma 5.9, we obtain that

$$|\hat{a}(\theta) - \bar{a}(\theta)| \leq \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(\mathbf{y})| \cdot |\langle \xi_{\mathbf{j}} + \theta_0 - \theta, \mathbf{y} \rangle| \quad \text{for} \quad \theta \in Q_{\mathbf{j}}\,.$$

Applying the Cauchy-Schwarz inequality yields:

$$|\hat{a}(\theta) - \bar{a}(\theta)| \leq \sum_{\mathbf{y} \in G_{\mathbf{h}}} |s(\mathbf{y})| \cdot \|\xi_{\mathbf{j}} + \theta_0 - \theta\| \cdot \|\mathbf{y}\| \quad \text{for} \quad \theta \in Q_{\mathbf{j}}\,.$$

Thus, we need to find a bound for $\|\xi_{\mathbf{j}} + \theta_0 - \theta\|$.

This bound is obtained by considering

$$\|\xi_{\mathbf{j}} + \theta_0 - \theta\|^2 = \sum_{k=1}^{d} |\xi_{\mathbf{j},k} + \theta_{0,k} - \theta_k|^2\,, \tag{5.19}$$

because an upper bound for this sum is obtained by deriving an upper bound for the individual summands $|\xi_{\mathbf{j},k} + \theta_{0,k} - \theta_k|$.

As $\theta \in Q_{\mathbf{j}}$, we have

$$\theta_k \in \left[\xi_{\mathbf{j},k}, \xi_{\mathbf{j},k} + \tfrac{2\pi}{h_k r_k}\right),$$

and since the absolute value is a convex function, the largest function value is located at the boundary of the interval, i.e., for $\theta_k = \xi_{\mathbf{j},k}$ or $\theta_k = \xi_{\mathbf{j},k} + \frac{2\pi}{h_k r_k}$. Thus,

$$\begin{aligned} |\xi_{\mathbf{j},k} + \theta_{0,k} - \theta_k| &\leq \max\{|\xi_{\mathbf{j},k} + \theta_{0,k} - \xi_{\mathbf{j},k}|, |\xi_{\mathbf{j},k} + \theta_{0,k} - (\xi_{\mathbf{j},k} + \tfrac{2\pi}{h_k r_k})|\} \\ &= \max\{|\theta_{0,k}|, |\theta_{0,k} - \tfrac{2\pi}{h_k r_k}|\}\,. \end{aligned}$$

Recall that $\theta_{0,k} \in [0, \frac{2\pi}{h_k r_k})$. From this fact, we obtain that

$$|\xi_{\mathbf{j},k} + \theta_{0,k} - \theta_k| \leq \max\{\theta_{0,k}, \tfrac{2\pi}{h_k r_k} - \theta_{0,k}\}\,.$$

Furthermore, for general $a, b \in \mathbb{R}$ we have $\max\{a, b\} = \frac{1}{2}(b - a) + \frac{1}{2}|b - a|$, and therefore

$$|\xi_{\mathbf{j},k} + \theta_{0,k} - \theta_k| \leq \tfrac{\pi}{h_k r_k} + |\tfrac{\pi}{h_k r_k} - \theta_{0,k}|\,.$$

We can combine this equation with (5.19) to obtain that

$$\|\xi_{\mathbf{j}} + \theta_0 - \theta\|^2 \leq \sum_{k=1}^{d} (\tfrac{\pi}{h_k r_k} + |\tfrac{\pi}{h_k r_k} - \theta_{0,k}|)^2 = \|C(\theta_0)\|^2\,. \qquad \square$$

The bound (5.18) involves the stencil coefficients $s(\mathbf{y})$, the norm of the stencil offsets $\mathbf{y}$, and the vector $C(\theta_0)$. Let us discuss the meaning of these constituents.

The size of the coefficients $s(\mathbf{y})$ provides the order of magnitude of the symbol $\hat{a}$. When $\hat{a}$ has, in general, large values, we expect the *absolute* approximation error to be large, too. Conversely, when we are interested in the relative error in the approximation, we can ignore the size of the coefficients.

The norm of the stencil offsets $\mathbf{y}$ gives the amount of oscillation a symbol $\hat{a}$ has, as we can see from

$$\hat{a}(\theta) := \sum_{\mathbf{y} \in G_{\mathbf{h}}} s(\mathbf{y}) \cdot e^{\mathrm{i}\langle \theta, \mathbf{y} \rangle} . \qquad \text{(1.55 revisited)}$$

When $\mathbf{y}$ is large, and $s(\mathbf{y}) \neq 0$, then $\hat{a}$ involves a highly oscillatory component, because the corresponding exponential is rapidly oscillating. In this case, we need a high resolution function to capture the rapidly changing function by our approximation.

The formula for the vector $C(\theta_0)$ has two consequences. First, we note that $\|C(\theta_0)\|$ has a minimal value when θ_0 is the mid-point of $\Theta_{\mathbf{r}\cdot\mathbf{h}}$, i.e., when $\theta_{0,k} = \frac{\pi}{h_k r_k}$. Thus, the bound (5.18) indicated that the best approximation is obtained when choosing θ_0 as the mid-point of $\Theta_{\mathbf{r}\cdot\mathbf{h}}$. Second, for this choice of θ_0 the value of $\|C(\theta_0)\|$ is

$$\left(\sum_{k=0}^{d} \left(\frac{\pi}{r_k h_k} \right)^2 \right)^{1/2} .$$

Thus, we can make $\|C(\theta_0)\|$ small by making the entries of the resolution $\mathbf{r}$ large. Hence, increasing the resolution should improve the quality of the approximation.

Note that the bound (5.18) is easily evaluated on the computer. Therefore, it can be used to get a bound on the approximation quality during computation.

Furthermore, note that choosing θ_0 as the mid-point of $\Theta_{\mathbf{r}\cdot\mathbf{h}}$ is only a good choice when we are interested in the approximation error. In case we are interested in, e.g., the supremum of the symbol, a different choice of θ_0 can lead to better results.

5.3.3. Norm and Spectral Radius using Symbols

The norm and spectral radius of error propagators give us insight into stationary iterative methods; we know from Theorem 2.39 that if an operator A has a Fourier symbol $\hat{a}$, then

$$r(A) = \|A\| = \|\hat{a}\|_\infty . \qquad \text{(2.19 revisited)}$$

Therefore, we can compute the desired quantities by computing the infinity norm of the symbol $\hat{a}$.

The infinity norm of a function f is given by

$$\|f\|_\infty = \inf\{c \geq 0 : \mu(|f| > c) > 0\} .$$

It is easy to see that if f is an finite resolution function, i.e.,

$$\bar{f} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} f_{\mathbf{k}} \cdot q_{\mathbf{k}} , \qquad \text{(5.9 revisited)}$$

we have

$$\|\bar{f}\|_\infty = \max_{\mathbf{k}} |a_{\mathbf{k}}| \,.$$

The procedure Infinity-Norm implements this formula to compute the infinity norm of an finite resolution function.

Infinity-Norm(f)
1 **return** $\max\{|f_{\mathbf{k}}| : \mathbf{k} = \mathbf{0}, \ldots, f.\mathit{resolution} - \mathbf{1}\}$

5.4. Approximating Fourier Matrix Symbols

Fourier matrix symbols allow us to analyze the two- and multigrid method and periodic stencil operators. Like ordinary Fourier symbols, Fourier *matrix* symbols cannot be stored exactly on a computer; they need to be approximated. We shall discuss how to define, store, and manipulate these approximations.

5.4.1. Frequency Splitting

The frequency splitting operator $\mathcal{R}$ was needed to define Fourier matrix symbols[2] and in the computation of symbols of periodic stencil operators[3]. We want to approximate Fourier matrix symbols by using finite resolution functions. As a first step, we consider the application of the frequency splitting operator to finite resolution functions.

The next lemma gives a formula for the function values of the individual parts that a finite resolution function is split into. The lemma requires that the resolution $\mathbf{r}$ should be a pointwise multiple of the splitting factor $\mathbf{n}$, i.e., there should be a $\mathbf{c} \in \mathbb{N}^d$ such that $\mathbf{r} = \mathbf{c} \cdot \mathbf{n}$. This makes the computations easier, as it ensures that the frequency splitting operator does not cut the individual steps of a finite resolution function into parts, as shown in Figure 5.5. This requirement is not really a restriction, as we can always slightly increase the resolution in a computation to meet the requirement.

Lemma 5.11. *Let* $\bar{f} \in T_{\mathbf{r}}$. *Let* $\mathbf{r}$ *be a (pointwise) multiple of* $\mathbf{n}$. *For* $\mathbf{j} \le \frac{\mathbf{r}}{\mathbf{n}} - \mathbf{1}$ *we have*

$$\left(\mathcal{R}_{\mathbf{n}} \bar{f}\right)_{\mathbf{k}} (\xi_{\mathbf{j}} + \theta_0) = \bar{f}(\xi_{\mathbf{j}+\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})} + \theta_0) \quad \textit{for} \quad \theta_0 \in \Theta_{\mathbf{r}\cdot\mathbf{h}} \,, \tag{5.20}$$

where

$$\xi_{\mathbf{k}} := \frac{2\pi}{\mathbf{h} \cdot \mathbf{r}} \cdot \mathbf{k} \,. \tag{5.6 revisited}$$

Proof. The first thing we should check is that (5.20) is actually well-defined, i.e., that $\xi_{\mathbf{j}} + \theta_0$ is in the domain $\Theta_{\mathbf{n}\cdot\mathbf{h}}$ of $(\mathcal{R}_{\mathbf{n}} \bar{f})_{\mathbf{k}}$. The requirement $\mathbf{j} \le \mathbf{r}/\mathbf{n} - \mathbf{1}$ implies

$$0 \le \xi_{\mathbf{j}} \le \frac{2\pi}{\mathbf{r} \cdot \mathbf{h}} \cdot \left(\frac{\mathbf{r}}{\mathbf{n}} - \mathbf{1}\right) = \frac{2\pi}{\mathbf{n} \cdot \mathbf{h}} - \frac{2\pi}{\mathbf{h} \cdot \mathbf{r}} \,,$$

[2] Definition 3.19 on page 82.
[3] Theorem 3.30 on page 95.

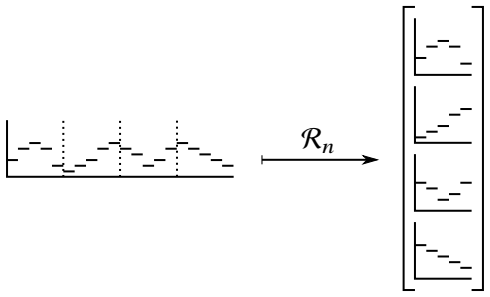


Figure 5.5.: When splitting a resolution $\mathbf{r}$ function, $\mathbf{r}$ should be a multiple of $\mathbf{n}$ to avoid cutting the individual steps of the function into parts.

while the definition of $\Theta_{\mathbf{r}\cdot\mathbf{h}}$ in equation (1.56) implies

$$0 \leq \theta_0 < \frac{2\pi}{\mathbf{r}\cdot\mathbf{h}}\,.$$

Adding the two previous equations gives

$$0 \leq \xi_{\mathbf{j}} + \theta_0 < \frac{2\pi}{\mathbf{n}\cdot\mathbf{h}}\,.$$

Thus, $\xi_{\mathbf{j}} + \theta_0 \in \Theta_{\mathbf{n}\cdot\mathbf{h}}$, and (5.20) is well-defined. With this knowledge, we can now prove the formula itself.

Using the definition of $\mathcal{R}_{\mathbf{n}}$ (3.3), we get

$$\left(\mathcal{R}\bar{f}\right)_{\mathbf{k}}(\xi_{\mathbf{j}} + \theta_0) = \bar{f}(\xi_{\mathbf{j}} + \theta_0 + \mathbf{s}_{\mathbf{k}})\,.$$

Replacing $\xi_{\mathbf{j}}$ and $\mathbf{s}_{\mathbf{k}}$ by their definitions, equation (5.6) and (3.2), the last equation can be written as

$$\begin{aligned}\left(\mathcal{R}\bar{f}\right)_{\mathbf{k}}(\xi_{\mathbf{j}} + \theta_0) &= \bar{f}\left(\tfrac{2\pi\mathbf{j}}{\mathbf{h}\cdot\mathbf{r}} + \theta_0 + \tfrac{2\pi\mathbf{k}}{\mathbf{h}\cdot\mathbf{n}}\right) = \bar{f}\left(\tfrac{2\pi\mathbf{j}}{\mathbf{h}\cdot\mathbf{r}} + \theta_0 + \tfrac{2\pi\mathbf{k}}{\mathbf{h}\cdot\mathbf{r}}\cdot\tfrac{\mathbf{r}}{\mathbf{n}}\right)\\ &= \bar{f}\left(\tfrac{2\pi}{\mathbf{h}\cdot\mathbf{r}}\left(\mathbf{j} + \mathbf{k}\cdot\tfrac{\mathbf{r}}{\mathbf{n}}\right) + \theta_0\right) = \bar{f}(\xi_{\mathbf{j}+\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})} + \theta_0)\,.\end{aligned}$$ □

On a computer, we represent finite resolution functions by the coefficients of their basis representation. For that reason, we want to represent the frequency splitting operator in terms of the coefficients.

Theorem 5.12. *Let $\bar{f} \in T_{\mathbf{r}}$ be given by*

$$\bar{f} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} f_{\mathbf{k}} \cdot q_{\mathbf{k}}, \qquad \text{(5.9 revisited)}$$

Then $[\mathcal{R}_{\mathbf{n}}\bar{f}]_{\mathbf{k}} \in T_{\mathbf{r}/\mathbf{n}}$, and we have

$$\left[\mathcal{R}_{\mathbf{n}}\bar{f}\right]_{\mathbf{k}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{r}/\mathbf{n}-\mathbf{1}} f_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}} \cdot q_{\mathbf{j}}\,. \qquad (5.21)$$

Note that the frequency splitting operator maps from $L_2(\Theta_{\mathbf{h}})$ to $L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})^{\mathbf{n}}$. Hence, we have that $\bar{f} \in L_2(\Theta_{\mathbf{h}})$ and $\left[\mathcal{R}_{\mathbf{n}}\bar{f}\right]_{\mathbf{k}} \in L_2(\Theta_{\mathbf{n}\cdot\mathbf{h}})$ for $\mathbf{j} = \mathbf{0}, \ldots, \mathbf{n} - \mathbf{1}$.

Proof. Let $\mathbf{j} < \mathbf{r}/\mathbf{n}$. As $\bar{f}$ is a resolution $\mathbf{r}$ function,

$$\bar{f}(\theta_{\mathbf{j}+\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})} + \theta_0) = f_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}} \quad \text{for} \quad \theta_0 \in \Theta_{\mathbf{r}\cdot\mathbf{h}}\,.$$

Plugging this relation into (5.20) gives that

$$\left[\mathcal{R}_{\mathbf{n}}\bar{f}\right]_{\mathbf{k}}(\xi_{\mathbf{j}} + \theta_0) = f_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}} \quad \text{for} \quad \theta_0 \in \Theta_{\mathbf{r}\cdot\mathbf{h}}\,.$$

Combining the definition of $\xi_{\mathbf{j}}$ and $\Theta_{\mathbf{r}\cdot\mathbf{h}}$ gives after a short computation that

$$\left[\mathcal{R}_{\mathbf{n}}\bar{f}\right]_{\mathbf{k}}(\theta) = f_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}} \quad \text{for all} \quad \theta \in Q_{\mathbf{j}}\,.$$

Thus, $\left[\mathcal{R}_{\mathbf{n}}\bar{f}\right]_{\mathbf{k}} \in T_{\mathbf{r}/\mathbf{n}}$, and it can be written in the form of (5.21). □

The frequency splitting operator splits a function into multiple parts. The sum (5.21) gives the coefficients of these parts. Hence, we can write down an algorithm for the frequency splitting operator applied to finite resolution functions.

The frequency splitting operator returns a vector whose entries are functions. To give an algorithm that computes the splitting, we need a way to denote such vectors. For this reason, we assume the existence of a function Allocate-Vector($\mathbf{n}$) that returns a handle to the storage of a vector of size $\mathbf{n}$, i.e., a vector whose entries can be indexed by the indices $\mathbf{0}, \ldots, \mathbf{n} - \mathbf{1}$. If v is such a handle, we assume that we can access the entries by $v_{\mathbf{j}}$ and the size by $v.\mathit{size}$. With this notation, we can now give the function Frequency-Splitting.

```
Frequency-Splitting(f, n)
1  r ← f.resolution
2  h ← f.step-size
3  g ← Allocate-Vector(n)
4  for k ← 0 to n − 1
5      g_k ← Allocate-Function(n · h, r/n)
6      for j ← 0 to (r/n) − 1
7          g_kj ← f_k·(r/n)+j
8  return g
```

Note that the Frequency-Splitting procedure expects a resolution $\mathbf{r}$ representation, where $\mathbf{r}$ is a multiple of $\mathbf{n}$. It returns a vector of shape $\mathbf{n}$ of resolution $\mathbf{r}/\mathbf{n}$ representations.

The definition of Fourier matrix symbols also involves the inverse $\mathcal{R}_{\mathbf{n}}^{-1}$ of the frequency splitting operator. Consequently, we also need an algorithm to compute the inverse, which we call Frequency-Join. It is a direct consequence of (5.21).

FREQUENCY-JOIN(f)
1 $\mathbf{n} \leftarrow f.\mathit{size}$
2 $\mathbf{r} \leftarrow f_{\mathbf{0}}.\mathit{resolution} \cdot \mathbf{n}$
3 $\mathbf{h} \leftarrow f_{\mathbf{0}}.\mathit{step\text{-}size}/\mathbf{n}$
4 $g \leftarrow$ ALLOCATE-FUNCTION($\mathbf{h}, \mathbf{r}$)
5 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\mathbf{n} - \mathbf{1}$
6 **for** $\mathbf{j} \leftarrow \mathbf{0}$ **to** $(\mathbf{r}/\mathbf{n}) - \mathbf{1}$
7 $f_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}} \leftarrow g_{\mathbf{kj}}$
8 **return** g

With these two algorithms in place, we can now turn to algorithms for matrix symbols.

5.4.2. Matrix Symbols

We want to work with Fourier matrix symbols on a computer, so that we can automatically analyze, e.g., periodic stencil operators[4]. Therefore, we need a way to store and manipulate matrix symbols on a computer, which we shall discuss now.

We can think of matrix symbols as matrices where each entry is a function from $L_\infty(\Theta_{\mathbf{h}})$ instead of a number, as is the case for ordinary matrices. We discussed in Section 5.3.1 that these functions cannot be represented exactly on a computer; we need to approximate them. For this approximation we chose the spaces of resolution $\mathbf{r}$ functions $T_{\mathbf{r}}$. The idea for approximating matrix symbols is to approximate a matrix symbol on a computer by a matrix, where each entry is a resolution $\mathbf{r}$ function from the space $T_{\mathbf{r}}$. We denote the set of $\mathbf{m} \times \mathbf{n}$-*resolution* $\mathbf{r}$ *function matrices* by

$$T_{\mathbf{r}}^{\mathbf{m}\times\mathbf{n}} := \{\bar{a} : \bar{a}_{\mathbf{ij}} \in T_{\mathbf{r}} \text{ for } \mathbf{i} = \mathbf{0}, \ldots, \mathbf{m}-\mathbf{1}, \mathbf{j} = \mathbf{0}, \ldots, \mathbf{n}-\mathbf{1}\}\,.$$

These matrices can be stored on a computer.

To store them, we need to find a suitable representation. By exploiting the fact that each entry $\bar{a}_{\mathbf{ij}}$ of an $\mathbf{m} \times \mathbf{n}$-resolution $\mathbf{r}$ function matrix $\bar{a}$ has a $\mathbf{r}$-resolution representation $a_{\mathbf{ij}}$, we can write

$$\bar{a}_{\mathbf{ij}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a_{\mathbf{ij},\mathbf{k}} \cdot q_{\mathbf{k}}\,, \tag{5.22}$$

where $q_{\mathbf{k}}$ is the basis functions defined in (5.8). We call the coefficients $a_{\mathbf{ij},\mathbf{k}}$ for $\mathbf{i} = \mathbf{0}, \ldots, \mathbf{m}-\mathbf{1}$, $\mathbf{j} = \mathbf{0}, \ldots, \mathbf{n}-\mathbf{1}$, $\mathbf{k} = \mathbf{0}, \ldots, \mathbf{r}-\mathbf{1}$ the $\mathbf{m} \times \mathbf{n}$-*resolution* $\mathbf{r}$ *representation matrix* of the matrix symbol $\bar{a}$.

All operations on resolution $\mathbf{r}$ representation matrices $T_{\mathbf{r}}^{\mathbf{m}\times\mathbf{n}}$ can be defined in terms of their representation. In many cases, however, we can reuse the procedures for resolution $\mathbf{r}$ functions that we introduced in Section 5.3.1, and work with the representation (5.22) only indirectly. For example, to allocate storage for a resolution $\mathbf{r}$ function representation, we just call the ALLOCATE-FUNCTION procedure for every entry of the matrix and organize the function handles properly. For this purpose, we assume the existence of a function

[4] Theorem 3.30 shows that every periodic stencil gives rise to a Fourier matrix symbol.

ALLOCATE-MATRIX($\mathbf{m}, \mathbf{n}$) that returns a handle to a storage for a matrix. If a is such a handle, then $a.\mathit{rows}$ should be the number of rows $\mathbf{m}$, and $a.\mathit{cols}$ should be the number of columns $\mathbf{n}$. Furthermore, $a_{\mathbf{ij}}$ should give access to the storage of the element with index $(\mathbf{i}, \mathbf{j})$, where $\mathbf{i} = \mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}$ and $\mathbf{j} = \mathbf{0}, \ldots, \mathbf{n} - \mathbf{1}$.

As an example, let us consider the function MS-ALLOCATE($\mathbf{h}, \mathbf{r}, \mathbf{n}, \mathbf{m}$) that returns a handle to a storage for a finite resolution function matrix.

```
MS-Allocate(h, r, m, n)
1  a ← Allocate-Matrix(m, n)
2  a.step-size ← h
3  a.resolution ← r
4  for i = 0 to n − 1
5      for j = 0 to n − 1
6          a_ij ← Allocate-Function(h, r)
```

In the following, we often shall abbreviate matrix symbol by the letters MS.

Let us now consider operations on matrix symbols. Our final goal is to evaluate a formula that describes an error propagator. As these formulas often involve matrix symbols, we want to be able to perform different operations with them, e.g., addition, scalar multiplication, etc. We start by considering the addition of two matrix symbols.

The addition of two matrix symbols $a \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_\mathbf{h})$ and $b \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_\mathbf{h})$ is given in Definition 3.10 by

$$(a + b)_{\mathbf{ij}} = a_{\mathbf{ij}} + b_{\mathbf{ij}} \,. \tag{3.10 revisited}$$

Recall that the entries $a_{\mathbf{ij}}$ and $b_{\mathbf{ij}}$ are functions from $L_\infty(\Theta_\mathbf{h})$. Thus, the sum is actually the pointwise sum of two functions. In our case, a and b are resolution $\mathbf{r}$ representation matrices $T_\mathbf{r}^{\mathbf{m}\times\mathbf{n}}$. Consequently, $a_{\mathbf{ij}}$ and $b_{\mathbf{ij}}$ are resolution $\mathbf{r}$ representations, and their addition was already discussed in Section 5.3.1. Using this knowledge, the implementation of MS-ADD, which adds two resolution $\mathbf{r}$ representation matrices, is straightforward.

```
MS-Add(a, b)
1  c ← MS-Allocate(a.step-size, a.resolution, a.rows, a.cols)
2  for i ← 0 to a.rows − 1
3      for j ← 0 to a.cols − 1
4          c_ij ← a_ij + b_ij
5  return c
```

Note that the sum in line 4 is computed by the ADD-FUNCTIONS procedure from Section 5.3.1. In a similar way, we can implement the matrix-matrix product.

The matrix-matrix product of two matrix symbols $a \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_\mathbf{h})$ and $b \in L_\infty^{\mathbf{n}\times\mathbf{p}}(\Theta_\mathbf{h})$ is given by

$$c_{\mathbf{ij}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{ik}} \cdot b_{\mathbf{kj}} \,. \tag{3.12 revisited}$$

For computations on a computer we want that the resolutions of the involved functions match appropriately. Thus, we require that $a_{\mathbf{ij}}$ and $b_{\mathbf{ij}}$ are both resolution $\mathbf{r}$ representations. The

equation for the matrix-matrix product (3.12) implies that the entries of the product c are obtained just by adding and multiplying resolution $\mathbf{r}$ representations. Therefore, the entries of the result c are also resolution $\mathbf{r}$ representations. This discussion proves the following lemma.

Lemma 5.13. *If $a \in T_{\mathbf{r}}^{\mathbf{m}\times\mathbf{n}}$ and $b \in T_{\mathbf{r}}^{\mathbf{n}\times\mathbf{p}}$, then the result of the matrix-matrix product $a \cdot b$ fulfills*

$$a \cdot b \in T_{\mathbf{r}}^{\mathbf{m}\times\mathbf{p}} .$$

With the same reasoning that we used to prove this lemma, we conclude that we can implement the matrix-matrix product just by using the procedures that operate on resolution $\mathbf{r}$ representations introduced in Section 5.3.1.

MS-Matrix-Multiply(a, b)

```
1  h ← a.step-size
2  r ← a.resolution
3  c ← MS-Allocate(h, r, a.rows, b.cols)
4  for i ← 0 to c.rows − 1
5      for j ← 0 to c.cols − 1
6          c_ij ← Constant-Function(h, r, 0)
7          for k ← 0 to a.cols − 1
8              c_ij ← c_ij + a_ik · b_kj
9  return c
```

Note that in Line 8, we are actually using the function Add-Functions and the function Pointwise-Multiply-Functions. In a similar way, we can define a function called MS-Vector-Multiply that multiplies a matrix of finite resolution representations with a vector of finite resolution representations.

The scalar multiplication, the matrix-vector symbol multiplication, and the computation of the adjoint can be implemented in the same way. First, we use the corresponding formula from Definition 3.10, then we use the operations on resolution $\mathbf{r}$ representations from Section 5.3.1 to compute the formula. There are, however, further operations on matrix symbols that cannot be implemented in this way. These operations are the inversion of matrix symbols and the computation of the norm and spectral radius.

The inversion of matrix symbols is a pointwise operation, i.e., to compute the inverse of a matrix symbol a, we have to compute the inverse of the matrix $a(\theta)$ for all $\theta \in \Theta_{\mathbf{h}}$. As a consequence, the inverse of a is given by

$$a^{-1}(\theta) = (a(\theta))^{-1} \quad \text{a.e.}, \tag{3.13 revisited}$$

which we use to compute the inverse of resolution $\mathbf{r}$ function matrices.

Assume that a is a resolution $\mathbf{r}$ representation matrix, i.e., it represents the resolution $\mathbf{r}$ function matrix

$$\bar{a}_{\mathbf{ij}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a_{\mathbf{ij},\mathbf{k}} \cdot q_{\mathbf{k}} . \tag{5.22 revisited}$$

From the definition of $q_{\mathbf{k}}$, we have that

$$\bar{a}_{\mathbf{ij}}(\theta) = a_{\mathbf{ij},\mathbf{k}} \quad \text{for} \quad \theta \in Q_{\mathbf{k}} .$$

If we define the auxiliary matrices $b_{\mathbf{k}} \in \mathbb{C}^{\mathbf{m}\times\mathbf{n}}$ by

$$b_{\mathbf{k},\mathbf{ij}} = a_{\mathbf{ij},\mathbf{k}} ,$$

we have that

$$\bar{a}(\theta) = b_{\mathbf{k}} \quad \text{for} \quad \theta \in Q_{\mathbf{k}} . \tag{5.23}$$

Using the pointwise inverse formula (3.13), we get

$$(\bar{a}^{-1})(\theta) = b_{\mathbf{k}}^{-1} \quad \text{for} \quad \theta \in Q_{\mathbf{k}} .$$

Note that in the last equation $b_{\mathbf{k}}^{-1}$ is the ordinary inverse of the matrix $b_{\mathbf{k}}$, which can be computed by standard methods like the LU-decomposition [16, 30, 60]. We can use this equation to give a basis representation of the inverse;

$$(\bar{a}^{-1})_{\mathbf{ij}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} (b_{\mathbf{k}}^{-1})_{\mathbf{ij}} \cdot q_{\mathbf{k}} .$$

From this formula we can deduce an algorithm for computing the inverse of a resolution $\mathbf{r}$ function matrix.

For this purpose, we need a way to compute the auxiliary matrices $b_{\mathbf{k}}$ which the procedure MS-At$(a, \mathbf{k})$ computes.

MS-At$(a, \mathbf{k})$

```
1  b ← Allocate-Matrix(a.rows, a.cols)
2  for i = 0 to a.rows − 1
3      for j = 0 to a.cols − 1
4          b_ij ← a_ij,k
5  return b
```

Then we can use this procedure in the MS-Invert(a) procedure, which computes the representation of the inverse of a matrix symbol a.

MS-Invert(a)

```
1  h ← a_00.step-size
2  r ← a_00.resolution
3  n ← a.rows
4  b ← MS-Allocate(h, r, n, n)
5  for k ← 0 to r − 1
6      t ← MS-At(a, k)
7      MS-At(b, k) ← t^{-1}
8  return b
```

Let us now turn to the computation of the norm and spectral radius of resolution **r** function matrices. Consider the statement of Theorem 3.25; if a is the matrix symbol of a matrix multiplication operator A, then

$$r(A) = \text{ess-sup}_{\theta \in \Theta_{\mathbf{h} \cdot \mathbf{n}}} r(a(\theta)) \qquad \text{and} \qquad \|A\| = \text{ess-sup}_{\theta \in \Theta_{\mathbf{h} \cdot \mathbf{n}}} \|a(\theta)\| . \quad \text{(3.24 revisited)}$$

Using the auxiliary matrices $b_{\mathbf{k}}$, especially (5.23), the computation of the spectral radius for a resolution **r** function matrix $\bar{a}$ simplifies to

$$r(\bar{A}) = \max_{\mathbf{k}} r(b_{\mathbf{k}}) ,$$

which gives rise to the algorithm for the MS-Norm procedure.

```
MS-Norm(a)
1  norm ← −∞
2  for k ← 0 to a.resolution − 1
3       m ← MS-At(k)
4       if ||m|| > norm
5            norm ← ||m||
```

Also by applying (5.23), the formula for the norm of the operator simplifies to

$$\|\bar{A}\| = \max_{\mathbf{k}} \|b_{\mathbf{k}}\| .$$

This computation is implemented in the MS-Spectral-Radius procedure.

```
MS-Spectral-Radius(a)
1  radius ← −∞
2  for k ← 0 to a.resolution − 1
3       m ← MS-At(k)
4       if r(m) > radius
5            radius ← r(m)
```

With these operations for resolution **r** representation matrices, we can now turn to the computation of Fourier matrix symbols and the symbols of periodic stencils.

5.4.3. Fourier Matrix Symbols

Fourier matrix symbols allow us to write certain Fourier representations in terms of matrix symbols, e.g., the multigrid method of the jumping coefficient problem from Section 4.1 or block smoothers, as discussed in Section 4.2. Recall that if A has a Fourier matrix symbol $\hat{a} \in L_{\infty}^{\mathbf{m} \times \mathbf{n}}$, then its Fourier representation $\widehat{A}$ can be written as

$$\widehat{A}\hat{u} = \mathcal{R}_{\mathbf{m}}^{-1} \hat{a} \mathcal{R}_{\mathbf{n}} \hat{u} . \quad \text{(3.16 revisited)}$$

Using the results from Section 5.4.1 and Section 5.4.2, we shall discuss how to compute finite resolution representation matrices of Fourier matrix symbols for periodic stencil, interpolation, and restriction operators.

Resolution of Fourier Matrix Symbols

We want that our computations involve only finite resolution functions with matching resolutions, which has some consequences for the computation with Fourier matrix symbols.

Consider the following algorithm that applies a finite resolution function $\hat{u}$ to the Fourier representation $\widehat{A}$ of an operator A that has a Fourier matrix symbol $\hat{a}$.

MATRIX-SYMBOL-APPLY($\hat{a}, \hat{u}$)

1 $\hat{u}' \leftarrow$ FREQUENCY-SPLITTING($\hat{u}, a.cols$)
2 $\hat{f}' \leftarrow$ MS-VECTOR-MULTIPLY($\hat{a}, \hat{u}'$)
3 **return** FREQUENCY-JOIN($\hat{f}'$)

It follows from the definition of Fourier matrix symbols (3.16) that the algorithm computes the desired result. However, the function $\hat{u}$ must have a proper resolution.

Let us now consider the question, which requirements the resolutions of $\hat{u}$ and $\hat{f}$ need to fulfill. Let $\hat{a}$ be an $\mathbf{m} \times \mathbf{n}$ resolution $\mathbf{r}$ representation matrix. The MS-VECTOR-MULTIPLY procedure then requires that $\hat{u}'$ is a vector of length $\mathbf{n}$ of resolution $\mathbf{r}$ representations. The vector $\hat{u}'$ is the return value of the FREQUENCY-SPLITTING procedure, and this procedure returns a vector of length $\mathbf{n}$ of resolution $\mathbf{r}$ representations only if $\hat{u}$ is a resolution $\mathbf{n} \cdot \mathbf{r}$ representation. Similarly, $\hat{f}'$ has to be a vector of length $\mathbf{m}$ of resolution $\mathbf{r}$ representations and therefore $\hat{f}$ is a resolution $\mathbf{r} \cdot \mathbf{m}$ representation. This discussion motivates the following definition.

Definition 5.14. Let $\hat{a}$ be an $\mathbf{m} \times \mathbf{n}$ resolution $\mathbf{r}$ function matrix that represents a Fourier matrix symbol. We then say that $\mathbf{r}_{\text{in}} := \mathbf{n} \cdot \mathbf{r}$ is the *input resolution* and $\mathbf{r}_{\text{out}} := \mathbf{m} \cdot \mathbf{r}$ is the *output resolution* of $\hat{a}$.

The definition directly implies the relation

$$\mathbf{r}_{\text{in}}/\mathbf{n} = \mathbf{r}_{\text{out}}/\mathbf{m}\,.$$

We can compare this relation to the relation between the input step-size and the output step-size of a Fourier matrix symbol, which is

$$\mathbf{n} \cdot \mathbf{h}_{\text{in}} = \mathbf{m} \cdot \mathbf{h}_{\text{out}}\,. \tag{3.15 revisited}$$

From these two equations a little algebra gives

$$\mathbf{h}_{\text{out}}/\mathbf{h}_{\text{in}} = \mathbf{r}_{\text{in}}/\mathbf{r}_{\text{out}}\,. \tag{5.24}$$

Thus, when an operator maps from a space with a small step-size to a space with a step-size that is larger by a factor, the resolution is reduced by the same factor and vice versa.

5.4.4. Periodic Stencil Operators

For the computation of the Fourier matrix symbol of periodic stencils we use Theorem 3.30, which states the following. Let $\{s_{\mathbf{x}}\}_{\mathbf{x} \in G_{\mathbf{h}}}$ be a periodic stencil with period $\mathbf{n}$. If $\hat{a}^{(\mathbf{k})}$ is the Fourier symbol of the *constant* stencil operator with stencil $s_{\mathbf{k} \cdot \mathbf{h}}$, we can compute the matrix symbol in three steps.

1. We compute the matrix F by

$$F_{\mathbf{kj}} := \frac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} e^{\mathrm{i}2\pi\langle \mathbf{k}/\mathbf{n},\mathbf{j}\rangle} \quad \text{for} \quad \mathbf{k},\mathbf{j} = \mathbf{0},\ldots,\mathbf{n}-\mathbf{1}\,. \tag{3.36 revisited}$$

2. We compute the matrix symbol g by

$$g_{\mathbf{kj}} = F_{\mathbf{kj}} \cdot [\mathcal{R}_{\mathbf{n}} \hat{a}^{(\mathbf{k})}]_{\mathbf{j}}\,. \tag{5.25}$$

3. We compute the Fourier matrix symbols of A by

$$\hat{a} = F^* \cdot g\,. \tag{5.26}$$

On a computer, however, we have to approximate these computations.

The computation of F requires no approximation[5], because F is just a regular matrix. The symbols $\hat{a}^{(\mathbf{k})}$, however, need to be approximated, which we do by using the Stencil-Symbol procedure. Let us denote this approximation by $\bar{a}^{(\mathbf{r})}$. Then we have to compute $\mathcal{R}_{\mathbf{n}}\bar{a}^{(\mathbf{k})}$, which implies that the resolution $\bar{\mathbf{r}}$ of $\bar{a}^{(\mathbf{k})}$ has to be a multiple of $\mathbf{n}$, i.e., $\bar{\mathbf{r}} = \mathbf{n}\cdot\mathbf{r}$ for some $\mathbf{r} \in \mathbb{N}^d$. Approximating g using the approximations $\bar{a}^{(\mathbf{k})}$ in (5.25), i.e., computing

$$g_{\mathbf{kj}} = F_{\mathbf{kj}} \cdot [\mathcal{R}_{\mathbf{n}} \bar{a}^{(\mathbf{k})}]_{\mathbf{j}}\,,$$

yields that $g \in T_{\mathbf{r}}^{\mathbf{n}\times\mathbf{n}}$. Finally, we apply (5.26) and obtain that $\hat{a} \in T_{\mathbf{r}}^{\mathbf{n}\times\mathbf{n}}$. Hence, as soon as we have computed the approximations $\bar{a}^{(\mathbf{k})}$, all computations involve only finite resolution functions. With $\mathbf{r}_{\text{in}} := \mathbf{n}\cdot\mathbf{r}$ we can write down the Periodic-Stencil-Symbol procedure.

Periodic-Stencil-Symbol$(s, \mathbf{r}_{\text{in}}, \theta_0)$

```
1  n ← s.period
2  for k ← 0 to n − 1
3      a^(k) ← Stencil-Symbol(s_{k·h}, r_in, θ_0)
4      ã^(k) ← Frequency-Splitting(a^(k), n)
5      for j ← 0 to n − 1
6          F_kj ← 1/vol_n^{1/2} e^{i2π⟨k/n,j⟩}
7          g_kj ← F_kj · ã_j^(k)
8  return F* · g
```

5.4.5. Restriction and Interpolation

To analyze two- and three-grid methods by LFA, we need to compute the Fourier matrix symbols of interpolation and restriction operators. As the Fourier matrix symbol of the injection restriction and the injection interpolation operators are constant with respect to the frequency θ, they are already finite resolution function matrices. Therefore, they fit well into the framework that we have been discussing so far. Furthermore, as many interpolation and restriction operators can be expressed by composition of injection and stencil operators, we

[5] Except for the approximation that occurs by the use of finite precision numbers, which we shall ignore in this discussion.

can analyze many methods in this way. There are, however, a few things to take into account when implementing operators that map between grids with different step-sizes.

We start with the injection restriction. It maps from a grid with step-size $\mathbf{h}_{\text{in}}$ to a grid with step-size $\mathbf{h}_{\text{out}} = \mathbf{n} \cdot \mathbf{h}_{\text{in}}$, where $\mathbf{n}$ is the coarsening range. Its Fourier matrix symbol $\hat{r} \in L_\infty^{\mathbf{1}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})$ and it has the entries

$$\hat{r}_{\mathbf{0j}}(\theta) = \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \quad \text{for} \quad \theta \in \Theta_{\mathbf{h}\cdot\mathbf{n}} \,. \tag{3.17 revisited}$$

We now want to construct an algorithm that computes a finite resolution representation matrix for $\hat{r}$.

Assume the input resolution $\mathbf{r}_{\text{in}}$ is given. Then the resolution $\mathbf{r}$ of the finite resolution function matrix that represents $\hat{r}$ is given by $\mathbf{r} = \mathbf{r}_{\text{in}}/\mathbf{n}$. The symbol of the restriction can be written as

$$\hat{r}_{\mathbf{0j}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \cdot q_{\mathbf{k}} \,.$$

From this equation we obtain that the resolution $\mathbf{r}$ representation matrix is

$$\left\{ \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \right\}_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}},$$

which is computed by the Injection-Restriction procedure.

Injection-Restriction($\mathbf{h}_{\text{in}}, \mathbf{r}_{\text{in}}, \mathbf{n}$)

1 $\hat{r} \leftarrow$ MS-Allocate($\mathbf{n} \cdot \mathbf{h}_{\text{in}}, \mathbf{r}_{\text{in}}/\mathbf{n}, \mathbf{1}, \mathbf{n}$)
2 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\mathbf{n} - \mathbf{1}$
3 $\quad \hat{r}_{\mathbf{0k}} \leftarrow$ Constant-Function $\left(\mathbf{n} \cdot \mathbf{h}_{\text{in}}, \mathbf{r}_{\text{in}}/\mathbf{n}, \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \right)$
4 **return** $\hat{r}$

In a similar way, we can derive from the formula for the Fourier matrix symbol of the injection interpolation

$$\hat{p}_{\mathbf{k0}} = \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \tag{3.20 revisited}$$

the procedure Injection-Interpolation that computes the resolution $\mathbf{r}$ representation matrix.

Injection-Interpolation($\mathbf{h}_{\text{out}}, \mathbf{r}_{\text{out}}, \mathbf{n}$)

1 $\hat{p} \leftarrow$ MS-Allocate($\mathbf{n} \cdot \mathbf{h}, \mathbf{r}_{\text{out}}/\mathbf{n}, \mathbf{n}, \mathbf{1}$)
2 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\mathbf{n} - \mathbf{1}$
3 $\quad \hat{p}_{\mathbf{k0}} \leftarrow$ Constant-Function $\left(\mathbf{h}_{\text{out}} \cdot \mathbf{n}, \mathbf{r}_{\text{out}}/\mathbf{n}, \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} \right)$
4 **return** $\hat{p}$

Consider now the stencil restriction operator, which is the composition of a stencil operator and the injection restriction operator. Its Fourier matrix symbol $\hat{r} \in L_\infty^{\mathbf{1}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}_{\text{in}}})$ is given by

$$\hat{r}_{\mathbf{0j}} = \frac{1}{\text{vol}_{\mathbf{n}}^{1/2}} [\mathcal{R}_{\mathbf{m}} \hat{r}_{\text{st}}]_{\mathbf{j}} \,. \tag{3.23 revisited}$$

In this equation, $\hat{r}_{\mathrm{st}}$ is the Fourier symbol of a stencil operator. To approximate $\hat{r}$ on a computer, we replace the symbol $\hat{r}_{\mathrm{st}}$ by a resolution $\mathbf{r}_{\mathrm{in}}$ function that approximates the actual symbol. Then equation (3.23) can be carried out on a computer using known algorithms. This leads to the definition of the Stencil-Restriction procedure.

Stencil-Restriction($\mathbf{r}_{\mathrm{in}}$, $\mathbf{n}$, s, θ_0)

1 $\mathbf{h}_{\mathrm{in}} \leftarrow s.\mathit{step\text{-}size}$
2 $\hat{r} \leftarrow$ MS-Allocate($\mathbf{n} \cdot \mathbf{h}_{\mathrm{in}}, \mathbf{r}_{\mathrm{in}}/\mathbf{n}, \mathbf{1}, \mathbf{n}$)
3 $\hat{a} \leftarrow$ Stencil-Symbol($s, \mathbf{r}_{\mathrm{in}}, \theta_0$)
4 $\hat{a}' \leftarrow$ Frequency-Splitting($\hat{a}, \mathbf{n}$)
5 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\mathbf{n} - \mathbf{1}$
6 $\quad \hat{r}_{\mathbf{0k}} \leftarrow \frac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} \cdot \hat{a}'_{\mathbf{k}}$
7 **return** $\hat{r}$

Note that to compute $\mathcal{R}_{\mathbf{n}}\hat{r}_{\mathrm{st}}$, the input resolution $\mathbf{r}_{\mathrm{in}}$ has to be a multiple of $\mathbf{n}$. Hence, the procedure Stencil-Restriction has the same requirement.

In a similar way, we obtain the procedure Stencil-Interpolation from

$$\hat{p}_{\mathbf{k0}} = \frac{1}{\mathrm{vol}_{\mathbf{m}}^{1/2}} [\mathcal{R}_{\mathbf{m}} \hat{p}_{\mathrm{st}}]_{\mathbf{k}} \,. \qquad \text{(3.22 revisited)}$$

This procedure computes an approximation to the stencil interpolation.

Stencil-Interpolation($\mathbf{r}_{\mathrm{out}}$, $\mathbf{n}$, s, θ_0)

1 $\mathbf{h}_{\mathrm{out}} \leftarrow s.\mathit{step\text{-}size}$
2 $\hat{p} \leftarrow$ MS-Allocate($\mathbf{n} \cdot \mathbf{h}_{\mathrm{out}}, \mathbf{r}_{\mathrm{out}}, \mathbf{n}, \mathbf{1}$)
3 $\hat{a} \leftarrow$ Stencil-Symbol($s, \mathbf{r}_{\mathrm{out}}, \theta_0$)
4 $\hat{a}' \leftarrow$ Frequency-Splitting($\hat{a}, \mathbf{n}$)
5 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\mathbf{n} - \mathbf{1}$
6 $\quad \hat{p}_{\mathbf{k0}} \leftarrow \frac{1}{\mathrm{vol}_{\mathbf{n}}^{1/2}} \cdot \hat{a}'_{\mathbf{k}}$
7 **return** $\hat{p}$

5.4.6. Expansion

The expansion of Fourier matrix symbols, that we discussed in Section 3.8, was essential for proving that the set of Fourier matrix symbols is closed under multiplication and addition. More precisely, to add or multiply Fourier matrix symbols, it is often necessary to expand them before the desired operation can be carried out. Consequently, for the implementation of general multiplication and addition of Fourier matrix symbols on a computer we need to implement the expansion of these symbols.

The information needed to create the expansion algorithm can be found in Theorem 3.37. It states that the expansion by a factor $\mathbf{p}$ maps a symbol $\hat{a} \in L_{\infty}^{\mathbf{n}\times\mathbf{n}'}(\Theta_{\mathbf{h}})$ to a symbol $\acute{a} \in L_{\infty}^{\mathbf{p}\cdot\mathbf{n}\times\mathbf{p}\cdot\mathbf{n}'}(\Theta_{\mathbf{p}\cdot\mathbf{h}})$. Both symbols represent the same operator. Furthermore, the entries of $\acute{a}$ are given by

$$\acute{a}_{\mathbf{k}\cdot\mathbf{p}+\mathbf{t},\mathbf{k}'\cdot\mathbf{p}+\mathbf{t}'} = [\mathcal{R}_{\mathbf{p}} \hat{a}_{\mathbf{k}\mathbf{k}'}]_{\mathbf{t}} \cdot \delta_{\mathbf{t}\mathbf{t}'} \,.$$

From this equation we can also deduce the resolution of the representation that we can compute.

Assume that $\hat{a}_{\mathbf{k}\mathbf{k}'}$ is a resolution $\mathbf{r}$ representation. We have to use the FREQUENCY-SPLITTING procedure on this representation, which returns a resolution $\mathbf{r}/\mathbf{p}$ representation. Hence, the entries of $\acute{a}$ also have to be resolution $\mathbf{r}/\mathbf{p}$ representations. As a consequence, we require $\mathbf{r}$ to be a multiple of $\mathbf{p}$.

As we now know how to compute the rows, columns, step-size, resolution, and the entries of the result of the expansion, we can define the EXPAND procedure that computes the expansion of an $\mathbf{n} \times \mathbf{n}'$ resolution $\mathbf{r}$ representation matrix.

EXPAND($\hat{a}$, $\mathbf{p}$)
1 $\acute{a} \leftarrow$ MS-ALLOCATE($\mathbf{p} \cdot \hat{a}.\mathit{step\text{-}size}, \hat{a}.\mathit{resolution}/\mathbf{p}, \mathbf{p} \cdot \hat{a}.\mathit{rows}, \mathbf{p} \cdot \hat{a}.\mathit{cols}$)
2 **for** $\mathbf{k} \leftarrow \mathbf{0}$ **to** $\hat{a}.\mathit{rows} - \mathbf{1}$
3 **for** $\mathbf{k}' \leftarrow \mathbf{0}$ **to** $\hat{a}.\mathit{cols} - \mathbf{1}$
4 $\tilde{a} \leftarrow$ FREQUENCY-SPLITTING($\hat{a}_{\mathbf{k}\mathbf{k}'}$, $\mathbf{p}$)
5 **for** $\mathbf{t} \leftarrow \mathbf{0}$ **to** $\mathbf{p} - \mathbf{1}$
6 **for** $\mathbf{t}' \leftarrow \mathbf{0}$ **to** $\mathbf{p} - \mathbf{1}$
7 $\acute{a}_{\mathbf{k}\cdot\mathbf{p}+\mathbf{t},\mathbf{k}'\cdot\mathbf{p}+\mathbf{t}'} \leftarrow \tilde{a}_{\mathbf{t}} \cdot \delta_{\mathbf{t}\mathbf{t}'}$
8 **return** $\acute{a}$

Let us sum up the properties of this procedure.

Proposition 5.15. *Let $\hat{a}$ be an $\mathbf{n} \times \mathbf{n}'$ resolution $\mathbf{r}$ representation matrix. The* EXPAND *procedure requires that the resolution $\mathbf{r}$ is a multiple of the expansion factor $\mathbf{p}$. It returns a $\mathbf{p} \cdot \mathbf{n} \times \mathbf{p} \cdot \mathbf{n}'$ resolution $\mathbf{r}/\mathbf{p}$ representation matrix that represents the same operator as $\hat{a}$.*

Combining this proposition with the definition of input and output resolution[6] gives the following corollary.

Corollary 5.16. *The* EXPAND *procedure does not change the input and output resolution of the operator, i.e., it returns a representation matrix with the same input and output resolution as $\hat{a}$.*

This corollary gives another justification for the definition of input and output resolution of finite resolution representation matrices. With the definition of the expansion procedure we can now construct the multiplication of finite resolution representation matrices with non-matching dimensions.

Assume that A and B have Fourier matrix symbols that have finite resolution representation matrices $\hat{a}$ and $\hat{b}$. Then to compute the Fourier matrix symbol of $B \cdot A$, we require that the output of A maps to a grid with the same step-size as the input of B, and that the output resolution of $\hat{a}$ is equal to the input resolution of $\hat{b}$. Furthermore, if we require a certain condition on the resolution of the representations, there exists an algorithm that computes a finite resolution representation matrix that represents $B \cdot A$, as the following theorem shows.

[6] Definition 5.14

Theorem 5.17. *Let* $A : \ell_2(G_{\mathbf{h}_{\text{in}}}) \to \ell_2(G_{\mathbf{h}_{\text{out}}})$ *and* $B : \ell_2(G_{\mathbf{h}'_{\text{in}}}) \to \ell_2(G_{\mathbf{h}'_{\text{out}}})$ *that have Fourier matrix symbols with finite resolution representations* $\hat{a}$ *and* $\hat{b}$*. Furthermore, let* $\hat{a}$ *be an* $\mathbf{m} \times \mathbf{n}$ *resolution* $\mathbf{r}$ *representation matrix and* $\hat{b}$ *be an* $\mathbf{m}' \times \mathbf{n}'$ *resolution* $\mathbf{r}'$ *representation matrix. If*

1. $\mathbf{h}_{\text{out}} = \mathbf{h}'_{\text{in}}$,
2. $\mathbf{r}_{\text{out}} = \mathbf{r}'_{\text{in}}$, *and*
3. $\mathbf{r}_{\text{out}}$ *is a multiple of* $\mathbf{m}$ *and* $\mathbf{n}'$,

then the Fourier matrix symbol of $C := B \cdot A$ *has a finite resolution representation matrix* $\hat{c}$ *and* $\hat{c}$ *is an* $\mathbf{m}' \cdot \mathbf{q}' \times \mathbf{n} \cdot \mathbf{q}$ *resolution* $\mathbf{r}_{\text{out}}/\text{lcm}(\mathbf{m}, \mathbf{n}')$ *representation, where* $\mathbf{q} = \text{lcm}(\mathbf{m}, \mathbf{n}')/\mathbf{m}$ *and* $\mathbf{q}' = \text{lcm}(\mathbf{m}, \mathbf{n}')/\mathbf{n}'$*. In these equations* lcm *denotes the pointwise least common multiple.*

Proof. Let $\acute{a}$ and $\acute{b}$ the factor $\mathbf{q}$ and factor $\mathbf{q}'$ expansions of $\hat{a}$ and $\hat{b}$. This definition, however, is only well defined if $\mathbf{r}$ is a multiple of $\mathbf{q}$ and $\mathbf{r}'$ is a multiple of $\mathbf{q}'$, which can be proven as follows.

The output resolution $\mathbf{r}_{\text{out}}$ is a multiple of $\mathbf{m}$ and $\mathbf{n}'$ and therefore a multiple of $\text{lcm}(\mathbf{m}, \mathbf{n}')$. As $\mathbf{r}_{\text{out}}$ is a multiple of $\text{lcm}(\mathbf{m}, \mathbf{n}')$, $\mathbf{r} = \mathbf{r}_{\text{out}}/\mathbf{m}$ is a multiple of $\text{lcm}(\mathbf{m}, \mathbf{n}')/\mathbf{m} = \mathbf{q}$, and therefore $\mathbf{r}$ is a multiple of $\mathbf{q}$. We can show that $\mathbf{r}'$ is a multiple of $\mathbf{q}'$ in a similar way.

Now, $\acute{a}$ and $\acute{b}$ have matching resolutions and therefore $\hat{c}$ is obtained by the product of the representations of $\acute{b}$ with $\acute{a}$. □

The proof of Theorem 5.17 describes the way how the finite representation matrix of $B \cdot A$ can be computed; leading to the definition of the General-Multiply procedure.

General-Multiply($\hat{b}, \hat{a}$)

```
1  q ← lcm(â.rows, b̂.cols)/â.rows
2  q′ ← lcm(â.rows, b̂.cols)/b̂.cols
3  á ← Expand(â, q)
4  b́ ← Expand(b̂, q′)
5  return MS-Multiply(b́, á)
```

To use this procedure, the requirements of Theorem 5.17 have to be met.

The first requirement, the step-sizes match, is a natural one. The composition of A and B is just not defined if the codomain of A does not match the domain of B. Thus, the user of the computer program, which computes the analysis, is required to make sure that the step-sizes match. The second and third requirement, that put restrictions on the resolution, can be fulfilled by a proper choice of resolutions. This proper choice of resolutions can be done automatically, as we shall discuss in a later section of this chapter.

Similar to Theorem 5.17 and the General-Multiply procedure, one can derive a general addition of finite resolution representation matrices.

Theorem 5.18. *Let* $A : \ell_2(G_{\mathbf{h}_{\text{in}}}) \to \ell_2(G_{\mathbf{h}_{\text{out}}})$ *and* $B : \ell_2(G_{\mathbf{h}'_{\text{in}}}) \to \ell_2(G_{\mathbf{h}'_{\text{out}}})$ *that have Fourier matrix representations* $\hat{a}$ *and* $\hat{b}$*. Furthermore, let* $\hat{a}$ *be an* $\mathbf{m} \times \mathbf{n}$ *resolution* $\mathbf{r}$ *representation matrix and* $\hat{b}$ *be an* $\mathbf{m}' \times \mathbf{n}'$ *resolution* $\mathbf{r}'$ *representation matrix. If*

1. $\mathbf{h}_{\text{in}} = \mathbf{h}'_{\text{in}}$,
2. $\mathbf{h}_{\text{out}} = \mathbf{h}'_{\text{out}}$,
3. $\mathbf{r}_{\text{in}} = \mathbf{r}'_{\text{in}}$, *and*

4. $\mathbf{r}_{\text{in}}$ *is a multiple of* $\mathbf{n}$ *and* $\mathbf{n}'$,

then the Fourier matrix symbol of $C := B + A$ *has a finite resolution representation matrix* $\hat{c}$ *and* $\hat{c}$ *is an* $\mathbf{m} \cdot \mathbf{q} \times \mathbf{n} \cdot \mathbf{q}$ *resolution* $\mathbf{r}_{\text{in}}/\text{lcm}(\mathbf{n}, \mathbf{n}')$ *representation, where* $\mathbf{q} = \text{lcm}(\mathbf{n}, \mathbf{n}')/\mathbf{n}$.

5.4.7. Smoothing Factor

Having the expansion procedure at hand, we can automate the computation of the smoothing factor. Let the smoother be given by its error propagator S with Fourier matrix symbols $\hat{s}$, and let $\hat{q}_{\mathbf{n}}$ be the Fourier symbol of the idealized coarse grid correction $Q_{\mathbf{n}}$. The smoothing factor is defined by

$$\text{smf}(S, \mathbf{n}) := r(S^{\nu_2} Q_{\mathbf{n}} S^{\nu_1}) = \text{ess-sup}_{\theta \in \Theta_{\mathbf{n} \cdot \mathbf{h}}} r(\acute{s}^{\nu_2}(\theta) \cdot \acute{q}_{\mathbf{n}}(\theta) \cdot \acute{s}^{\nu_1}(\theta)), \qquad \text{(3.40 revisited)}$$

where $\acute{s}$ and $\acute{q}_{\mathbf{n}}$ are properly expanded versions of $\hat{s}$ and $\hat{q}_{\mathbf{n}}$.

Let us assume that we have a finite resolution representation matrix for $\hat{s}$. The only thing missing to compute the smoothing factor is a finite resolution representation of $\hat{q}_{\mathbf{n}}$.

Combining (1.61) and (1.62) yields that the Fourier symbol of $Q_{\mathbf{n}}$ is given by

$$\hat{q}_{\mathbf{n}}(\theta) = \begin{cases} 0 & \text{if } 0 \le \theta_k < \frac{\pi}{n_k h_k} \quad \text{or} \quad \frac{(2n_k - 1)\pi}{n_k h_k} \le \theta_k < \frac{2\pi}{h_k} \quad \text{for all} \quad k = 1, \ldots, d, \\ 1 & \text{otherwise.} \end{cases}$$

This is a resolution $2\mathbf{n}$ function. Hence, for any resolution $\mathbf{r}$ that is a multiple of $2\mathbf{n}$ there is a resolution $\mathbf{r}$ *representation* of $\hat{q}_{\mathbf{n}}$.

It turns out that we can give a Fourier *matrix* symbol which represents $Q_{\mathbf{n}}$ in a more elegant way. Combining the definition of the frequency splitting operator (3.3) with the previous equation, we obtain

$$[\mathcal{R}_{2\mathbf{n}} \hat{q}_{\mathbf{n}}]_{\mathbf{i}}(\theta) = \begin{cases} 0 & \text{if } i_k = 0 \quad \text{or} \quad i_k = 2s_k - 1 \quad \text{for all} \quad k = 1, \ldots, d, \\ 1 & \text{otherwise.} \end{cases}$$

We see that the entries of the vector $\mathcal{R}_{2\mathbf{n}} \hat{q}_{\mathbf{n}}$ are constant functions. Applying the operator $\hat{q}_{\mathbf{n}}$ to a test function $\hat{u}$ gives

$$\begin{aligned} [\mathcal{R}_{2\mathbf{n}} [\widehat{Q}_{\mathbf{n}} \hat{u}]]_{\mathbf{i}} &= [\mathcal{R}_{2\mathbf{n}} (\hat{q}_{\mathbf{n}} \cdot \hat{u})]_{\mathbf{i}} \\ &= [\mathcal{R}_{2\mathbf{n}} \hat{u}]_{\mathbf{i}} \cdot \begin{cases} 0 & \text{if } i_k = 0 \quad \text{or} \quad i_k = 2s_k - 1 \quad \text{for all} \quad k = 1, \ldots, d, \\ 1 & \text{otherwise.} \end{cases} \end{aligned}$$

This equation can be written as

$$[\mathcal{R}_{2\mathbf{n}} [\widehat{Q}_{\mathbf{n}} \hat{u}]]_{\mathbf{i}} = \sum_{\mathbf{j}} \acute{q}_{\mathbf{n},\mathbf{ij}} \cdot [\mathcal{R}_{2\mathbf{n}} \hat{u}]_{\mathbf{j}},$$

where

$$\acute{q}_{\mathbf{n},\mathbf{ij}} = \delta_{\mathbf{ij}} \cdot \begin{cases} 1 & \text{if } i_k = 0 \quad \text{or} \quad i_k = 2s_k - 1 \quad \text{for all} \quad k = 1, \ldots, d, \\ 0 & \text{otherwise,} \end{cases}$$

leading to

$$\widehat{Q}_{\mathbf{n}}\hat{u} = \mathcal{R}_{2\mathbf{n}}^{-1}\acute{q}_{\mathbf{n}}\mathcal{R}_{2\mathbf{n}}\hat{u}\,.$$

Hence, $\acute{q}_{\mathbf{n}}$ is the $2\mathbf{n} \times 2\mathbf{n}$ Fourier matrix symbol of the operator $\widehat{Q}_{\mathbf{n}}$, and each entry of $\acute{q}$ is a constant function, which can be represented by any resolution $\mathbf{r} > \mathbf{0}$. Therefore, to multiply $\acute{q}_{\mathbf{n}}$ with another matrix symbol, we only need to expand the two operators to proper dimensions; the resolution of $\acute{q}_{\mathbf{n}}$ adds no further constraints, as it can be chosen arbitrarily.

5.4.8. Matrix Representation

Consider an $\mathbf{m} \times \mathbf{n}$ resolution $\mathbf{r}$ function matrix $\bar{a}$ that is the Fourier matrix symbol of an operator A, i.e., the Fourier representation $\widehat{A}$ of A is given by $\widehat{A} = \mathcal{R}_{\mathbf{m}}^{-1}\bar{a}\mathcal{R}_{\mathbf{n}}$. Restricting the Fourier representation to $T_{\mathbf{n}\cdot\mathbf{r}}$ gives an operator $\widehat{A} : T_{\mathbf{n}\cdot\mathbf{r}} \to T_{\mathbf{m}\cdot\mathbf{r}}$, which is a linear operator that maps between finite dimensional spaces. Hence, there exists a matrix that represents the action of A given a basis of $T_{\mathbf{n}\cdot\mathbf{r}}$ and of $T_{\mathbf{m}\cdot\mathbf{r}}$. The following theorem shows how to obtain this matrix from the representation of the function matrix $\bar{a}$ for the basis (5.8) of $T_{\mathbf{n}\cdot\mathbf{r}}$ and $T_{\mathbf{m}\cdot\mathbf{r}}$.

Theorem 5.19. *Let $\bar{a}$ be an $\mathbf{m} \times \mathbf{n}$ resolution $\mathbf{r}$ function matrix, with representation a and $\widehat{A} : T_{\mathbf{n}\cdot\mathbf{r}} \to T_{\mathbf{m}\cdot\mathbf{r}}$ given by $\widehat{A} = \mathcal{R}_{\mathbf{m}}^{-1}\bar{a}\mathcal{R}_{\mathbf{n}}$. Furthermore, let $\bar{f} \in T_{\mathbf{r}}^{\mathbf{m}}$ and $\bar{u} \in T_{\mathbf{r}}^{\mathbf{n}}$ such that $\bar{f} = \widehat{A}\bar{u}$. If we define f and u by the basis representations*

$$\bar{f} = \sum_{\mathbf{j},\mathbf{k}} f_{\mathbf{j}\mathbf{k}} \cdot q_{\mathbf{k}\cdot(\mathbf{r}/\mathbf{m})+\mathbf{j}} \quad \textit{and} \quad \bar{u} = \sum_{\mathbf{j}',\mathbf{k}'} u_{\mathbf{j}'\mathbf{k}'} \cdot q_{\mathbf{k}'\cdot(\mathbf{r}/\mathbf{n})+\mathbf{j}'}\,,$$

then $\widehat{A}$ can be represented by a matrix a', i.e.,

$$f_{\mathbf{j}\mathbf{k}} = \sum_{\mathbf{j}',\mathbf{k}'} a'_{\mathbf{j}\mathbf{k},\mathbf{j}'\mathbf{k}'} \cdot u_{\mathbf{j}'\mathbf{k}'}\,, \quad \textit{where} \quad a'_{\mathbf{j}\mathbf{k},\mathbf{j}'\mathbf{k}'} = a_{\mathbf{i}\mathbf{j},\mathbf{k}} \cdot \delta_{\mathbf{k}\mathbf{k}'}\,. \tag{5.27}$$

Proof. We start with $\bar{f} = \widehat{A}\bar{u}$ and obtain that

$$\bar{f} = \mathcal{R}_{\mathbf{m}}^{-1}\bar{a}\mathcal{R}_{\mathbf{n}}\bar{u}\,.$$

Multiplying by $\mathcal{R}_{\mathbf{m}}$ from the left gives that

$$\mathcal{R}_{\mathbf{m}}\bar{f} = \bar{a}\mathcal{R}_{\mathbf{n}}\bar{u}\,,$$

and the definition of matrix symbols yields that

$$[\mathcal{R}_{\mathbf{m}}\bar{f}]_{\mathbf{j}} = \sum_{\mathbf{j}'=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \bar{a}_{\mathbf{j}\mathbf{j}'} \cdot [\mathcal{R}_{\mathbf{n}}\bar{u}]_{\mathbf{j}'}\,. \tag{5.28}$$

We shall now rewrite this equation by using the resolution $\mathbf{r}$ representations of the involved finite resolution functions.

Using Theorem 5.12, we see that $\{f_{\mathbf{j}\mathbf{k}}\}_{\mathbf{k}=\mathbf{0}}^{\mathbf{r}-\mathbf{1}}$ and $\{u_{\mathbf{j}'\mathbf{k}'}\}_{\mathbf{k}'=\mathbf{0}}^{\mathbf{r}-\mathbf{1}}$ are the resolution $\mathbf{r}$ representations of $[\mathcal{R}_{\mathbf{m}}\bar{f}]_{\mathbf{j}}$ and $[\mathcal{R}_{\mathbf{n}}\bar{u}]_{\mathbf{j}'}$. Thus, writing (5.28) in terms of these representations gives

$$f_{\mathbf{j}\mathbf{k}} = \sum_{\mathbf{j}'=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} a_{\mathbf{j}\mathbf{j}',\mathbf{k}} \cdot u_{\mathbf{j}'\mathbf{k}} = \sum_{\mathbf{j}'=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \sum_{\mathbf{k}'=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a_{\mathbf{j}\mathbf{j}',\mathbf{k}} \cdot \delta_{\mathbf{k}\mathbf{k}'} \cdot u_{\mathbf{j}'\mathbf{k}'} = \sum_{\mathbf{j}'=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \sum_{\mathbf{k}'=\mathbf{0}}^{\mathbf{r}-\mathbf{1}} a'_{\mathbf{j}\mathbf{k},\mathbf{j}'\mathbf{k}'} \cdot u_{\mathbf{j}'\mathbf{k}'}\,. \qquad \square$$

In this theorem we used two multiindices $\mathbf{jk}$ and $\mathbf{j'k'}$ to index the coefficients $f_{\mathbf{jk}}$ and $u_{\mathbf{j'k'}}$. Thus, the matrix relating f and u has four multiindices. This choice of indices made the formulation and proof of Theorem 5.19 easier. Furthermore, it reveals that the matrix a' is a block diagonal matrix.

From (5.27) we see that the entry $a'_{\mathbf{jk},\mathbf{j'k}}$ is zero if $\mathbf{k} \neq \mathbf{k'}$. Thus, if we order the row indices by $\mathbf{k}$ and column indices by $\mathbf{k'}$ the matrix a' is block diagonal. Hence, we can also interpret finite representation matrices that represent Fourier symbols as block matrices. It can be shown that operations on these representations directly map to operations on block matrices.

5.5. A Language for LFA

LFA analyzes the properties of iterative methods by examining the error propagator of the method. The error propagator is, in general, given by a formula and a set of (periodic) stencils that describe the involved operators. The idea of LFA is to compute the error propagator by evaluating the formula for the Fourier matrix symbols corresponding to the involved operators. We know how to carry out these computations; however, we need a way to turn a formula that a user enters into a sequence of operations on Fourier matrix symbols.

5.5.1. Evaluating Formulas

We shall briefly discuss the algorithmic evaluation of formulas [1, 2, 65]. Let us start with a simple example, the computation of

$$6 \cdot (3 + 4)\,.$$

The evaluation is, of course, done by first evaluating the sum of three and four to seven, and then evaluating the product of six and seven, i.e.,

$$6 \cdot (3 + 4) \quad \rightsquigarrow \quad 6 \cdot 7 \quad \rightsquigarrow \quad 42\,.$$

This is a straightforward procedure. There is, however, an issue with the notation that we used.

Mathematically, $6 \cdot 7$ is equal to 42; the two are indistinguishable. As we want to talk about the term "$6 \cdot 7$" and not its value, which is 42, we need a way to describe terms in a mathematical way without the risk of confusing the term with its value. One way to do this, is to use *prefix notation* [1]. A term in prefix notation is a tuple, where the first element of the tuple is the operator and the remaining elements are the arguments. For example the term "$3 + 4$" written in prefix notation is

$$(+, 3, 4)$$

and the term "$6 \cdot (3 + 4)$" is

$$(\cdot, 6, (+, 3, 4))\,.$$

In this notation we can write the evaluation of "$6 \cdot (3 + 4)$" as

$$(\,\cdot\,, 6, (+, 3, 4)) \quad \rightsquigarrow \quad (\,\cdot\,, 6, 7) \quad \rightsquigarrow \quad 42\,.$$

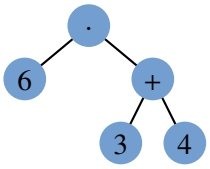


Figure 5.6.: Expression tree of the term "$6 \cdot (3 + 4)$".

From this representation of the formula, we can easily derive the corresponding *expression tree*[7].

We can construct an expression tree from a term in prefix notation by creating a node for every tuple and label it with the operator, the first element of the tuple. Then we create a node for every number and label it with the number itself. As a last step, for each tuple we create an edge that connects the operator node with the node of its arguments. It can easily be seen that the resulting graph is a tree. Figure 5.6 shows an example.

We shall now discuss the evaluation procedure in an abstract way, i.e., we want to describe the algorithm for the *evaluation* of an *expression* to a *value*. To define this algorithm, we need three sets: the set of *values*, the set of *operators*, and the set of *expressions*. The evaluation algorithm takes an element of the set of expressions and returns an element of the set of values, according to some rules. This procedure differs from the way a (mathematical) function is evaluated; when the algorithm is applied to the same input expression, it might return a different value for every application. This can be the case, for example, when the computation of the result depends on some user input.

The set of values can, in principle, be any arbitrary set. The members of this set, however, are the objects that we accept as a final result of the computation. In the example above we could have chosen the set of real numbers $\mathbb{R}$. Thus, we would accept "42" as a result, but not $(\cdot, 6, 7)$, as this is a whole term and not just a number.

The set of operators contains the actions or computations that we can perform. More precisely, we assume there exists an Apply procedure that, given an *operator* and a list of *values*, returns a new value.

The set of expressions is a set that contains objects which describe the actions that should be carried out by the evaluation algorithm. This set is defined recursively, starting with the set of values.

$$e \text{ is an expression if } \begin{cases} e \text{ is a value, or} \\ e = (op, e_1, e_2, \dots), \quad \text{where} \quad op \text{ is an operation and} \\ \qquad\qquad\qquad\qquad\quad\ e_1, e_2, \dots \text{ expressions.} \end{cases}$$

Consider the evaluation of an expression of the form $(op, e_1, e_2, \dots)$. The meaning of the operator op is given by the Apply procedure. This procedure, however, can only be applied to the operator and a list of values and not a list of expressions. Thus, to apply the operator to its arguments, we first have to compute the value of the argument expressions $e_1, e_2, \dots$, which is what the evaluation procedure is supposed to do. To evaluate an expression, the

[7] For a definition of trees see [16].

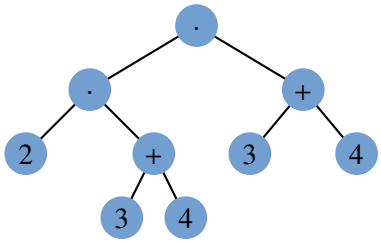


Figure 5.7.: Expression tree of the term $(2 \cdot (3 + 4)) \cdot (3 + 4)$.

evaluation procedure is, therefore, recursively applied to all argument expressions. Then the operator is applied to all argument values. The Eval procedure is defined below.

```
Eval(expr)
1  if expr in values
2      return expr
3  else
4      (op, e1, e2, ...) ← expr
5      v1 ← Eval(e1)
6      v2 ← Eval(e2)
7      ...
8      return Apply(op, v1, v2, ...)
```

When an expression contains the same sub-expression multiple times, the Eval procedure performs the same actions for each of these sub-expressions. These sub-expressions often[8] evaluate to the same value. Thus, computing the sub-expression multiple times is unnecessary.

Consider the expression

$$(\cdot, (\cdot, 2, (+, 3, 4)), (+, 3, 4)),$$

which computes $(2 \cdot (3 + 4)) \cdot (3 + 4)$ and contains the sub-expression $(+, 3, 4)$ two times. Figure 5.7 shows the corresponding expression tree. This tree contains the sub-tree corresponding to the sub-expression $(+, 3, 4)$ twice, which can be avoided if we connect the nodes that connect to the $(+, 3, 4)$ to the same sub-tree instead of two independent sub-trees, as shown in Figure 5.8. This graph, however, is not a tree anymore. Though, it is acyclic, which allows us to apply the Eval procedure to the graph without changes.

When the Eval procedure is applied to the directed acyclic graph from Figure 5.8, it performs the same operations as when it is applied to the tree from Figure 5.7. To actually avoid the repeated evaluation of the sub-expression, we need to modify the Eval procedure to store the intermediate results, i.e., the values of the sub-expressions. One way to do this, is to replace the node that represents a sub-expression by a node representing its value, as soon as it is computed. This is, what the Eval-Dag procedure does.

[8] This is the case when the sub-expression does not depend on a state that changes during the computation.

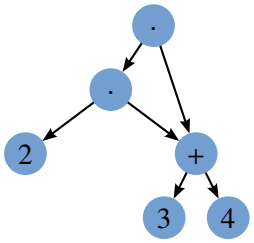


Figure 5.8.: Directed acyclic graph of the term $(2 \cdot (3 + 4)) \cdot (3 + 4)$.

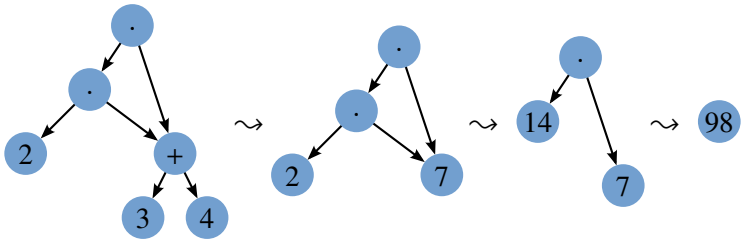


Figure 5.9.: Application of the EVAL-DAG procedure to the directed acyclic graph representing the expression $(2 \cdot (3 + 4)) \cdot (3 + 4)$.

EVAL-DAG(*node*)

```
1  if node in values
2      return node
3  else
4      (op, e1, e2, ...) ← node
5      v1 ← EVAL(e1)
6      v2 ← EVAL(e2)
7      ...
8      result ← APPLY(op, v1, v2, ...)
9      replace node by result
10     return result
```

Its application to the directed acyclic graph from Figure 5.8 is visualized in Figure 5.9.

5.5.2. Building Expression Trees

If a user wants to evaluate a formula on a computer, he/she needs a way to enter the formula into the computer. Since the computer can so far only evaluate expression trees, we need a procedure that converts the user input into an expression tree.

A formula, for example, can be specified in a textual way; the user enters a text that defines the formula. For example the text "`6 * (3 + 4)`" is a textual description of the expression tree from Figure 5.6. A program that turns such a text into a formula is called a *parser* [2]. For a parser to be able to perform its task, the text has to follow a certain set of rules. Writing

a parser is laborious, however, many programming languages allow us to reuse their parser for our purpose.

If a programming language allows *operator overloading*, we can often use the parser of the programming language to build an expression tree. Operator overloading enables us to (re-) define the meaning of operators for user-defined types, allowing us to define a type which represents an expression. Operations on this type then create new expressions instead of performing computations. In that way we obtain an expression tree instead of a value. We shall demonstrate this approach by a small example.

The example is a program written in the Python [69] programming language. The program turns a formula into an expression tree, which is then used to print the prefix notation of the corresponding formula.

We start by creating a class called Expression and define its addition and multiplication operators by overwriting the `__add__` and `__mul__` methods. The operators, instead of computing a value, just construct an instance of the classes Sum and Product, which store the left hand side (self) and the right hand side (other) of the addition and multiplication.

```
1  #!/usr/bin/env python3
2
3  class Expression:
4      def __add__(self, other):
5          return Sum(self, other)
6
7      def __mul__(self, other):
8          return Product(self, other)
```

Furthermore, we define a class Operator which is an Expression and classes Sum and Product which are Operators. The Operator class implements the `__str__` method which describes how an operator will be displayed. It will be represented by an opening parenthesis, followed by its name, followed by the representation of its arguments, followed by a closing parenthesis.

The Sum and Product class just store their arguments and set their name, such that they can be printed by the method defined in the Operator class.

```
10 class Operator(Expression):
11     def __str__(self):
12         s =  '(' + self.name
13         for d in self.arguments:
14             s += ', ' + str(d)
15         return s + ')'
16
17 class Sum(Operator):
18     def __init__(self, lhs, rhs):
19         self.name = '+'
20         self.arguments = [lhs, rhs]
21
22 class Product(Operator):
23     def __init__(self, lhs, rhs):
24         self.name = '*'
25         self.arguments = [lhs, rhs]
```

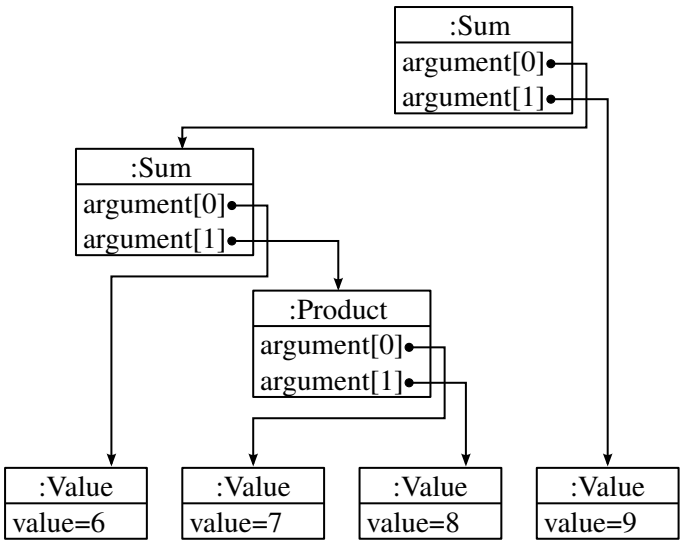


Figure 5.10.: Object diagram of the expression from the example program.

Note that the Sum and the Product class are themselves Expressions, as the Operator class is an Expression. The sum of two Expressions returns an instance of the Sum class. As this instance is itself an Expression, something else can be added to this class, which will create a new instance of the Sum class. This will produce a tree, in which the inner nodes are instances of the Operator class and the children of the nodes are the arguments of the operator.

There needs to be a way to represent values. Therefore, we define a class called Value. This class just stores the value it is given. However, it is an Expression and therefore the addition and multiplication operators of the Expression class are used. Thus, whenever adding two instances of the Value class, an instance of the Sum class is returned which references the two values in its arguments.

The Value class also implements the `__str__` method which just returns the corresponding value as a string.

```
27 class Value(Expression):
28     def __init__(self, v):
29         self.value = v
30 
31     def __str__(self):
32         return str(self.value)
```

We can test our implementation with the following code.

```
34 expr = Value(6) + Value(7) * Value(8) + Value(9)
35 print(expr)
```

The objects that are referenced in the `expr` variable are pictured in Figure 5.10. From this structure, we can easily see that the output of this code will be:

```
(+, (+, 6, (*, 7, 8)), 9)
```

5.5.3. Matching Resolutions

To compute the product or sum of two Fourier matrix symbols with finite resolution representations and matching step-sizes, the resolutions need to fulfill the conditions of Theorem 5.17 or Theorem 5.18, respectively. Assume that we have a set of Fourier matrix symbols with finite resolution representations and a formula that combines them. We shall show how to assign a resolution to every Fourier matrix symbol such that all operations which are required to evaluate the formula combine only representations with matching resolution.

To choose the resolutions appropriately, we shall make use of the next theorem. It states that for every formula that involves Fourier matrix symbols there is a *common step-size*, in the sense that all other step-sizes are integer multiples of the common step-size.

Theorem 5.20. *Let $A \in L(\ell_2(G_{\mathbf{h}_{\text{in}}}); \ell_2(G_{\mathbf{h}_{\text{out}}}))$ be given by an expression. The set of values should be the set of Fourier matrix symbols, and the set of operators should consist of the addition, multiplication, scalar multiplication, inverse, and adjoint. Then the step-sizes of the domain and the steps-sizes of the codomain of all Fourier matrix symbols in the expression are the integer multiples of a common step-size $\mathbf{h}_0$.*

Proof. We prove the assertion by induction. First, we show that the set of values fulfills the assertion. Then we show that an operator applied to expressions that fulfill the assertion returns a Fourier matrix symbol, which also fulfills the assertion.

Let us start with the set of values. Let $B \in L(\ell_2(G_{\mathbf{h}_1}); \ell_2(G_{\mathbf{h}_2}))$ be an operator that has a Fourier matrix symbol $\hat{b} \in L_\infty^{\mathbf{m}\times\mathbf{n}}(\Theta_{\mathbf{n}\cdot\mathbf{h}_1})$. Then the step-sizes have to fulfill the requirement (3.15), i.e.,

$$\mathbf{n} \cdot \mathbf{h}_1 = \mathbf{m} \cdot \mathbf{h}_2 \,.$$

Thus, if we define the common step-size $\mathbf{h}_0 = \mathbf{h}_1/\mathbf{m} = \mathbf{h}_2/\mathbf{n}$, $\mathbf{c}_1 = \mathbf{m}$, and $\mathbf{c}_2 = \mathbf{n}$, then $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{N}^d$,

$$\mathbf{h}_1 = \mathbf{c}_1 \cdot \mathbf{h}_0 \,, \quad \text{and} \quad \mathbf{h}_2 = \mathbf{c}_2 \cdot \mathbf{h}_0 \,.$$

We continue with the addition of an operator $B \in L(\ell_2(G_{\mathbf{h}_1}); \ell_2(G_{\mathbf{h}_2}))$ and an operator $B' \in L(\ell_2(G_{\mathbf{h}'_1}); \ell_2(G_{\mathbf{h}'_2}))$. The induction hypothesis tells us that B and B' can be computed using common step-sizes $\mathbf{h}_0$ and $\mathbf{h}'_0$. In particular, this statement implies that there exist $\mathbf{c}_1, \mathbf{c}'_1 \in \mathbb{N}^d$ such that

$$\begin{aligned} \mathbf{h}_1 &= \mathbf{c}_1 \cdot \mathbf{h}_0 \,, \\ \mathbf{h}'_1 &= \mathbf{c}'_1 \cdot \mathbf{h}'_0 \,. \end{aligned}$$

The fact that we are able to add B and B' implies that $\mathbf{h}_1 = \mathbf{h}'_1$, otherwise the formula would not make sense. Combining this equation with the previous one yields that

$$\mathbf{c}_1 \cdot \mathbf{h}_0 = \mathbf{c}'_1 \cdot \mathbf{h}'_0 \,.$$

Thus, if we define $\mathbf{h}''_0 = \mathbf{h}_0/\mathbf{c}'_1 = \mathbf{h}'_0/\mathbf{c}_1$, then $\mathbf{h}_0$ and $\mathbf{h}'_0$ are integer multiples of $\mathbf{h}''_0$, and all integer multiples of $\mathbf{h}_0$ and $\mathbf{h}'_0$ can be written as integer multiples of $\mathbf{h}''_0$. Therefore, $\mathbf{h}''_0$ is the new common step-size.

We consider the product $B' \cdot B$ of an operator $B \in L(\ell_2(G_{\mathbf{h}_1}); \ell_2(G_{\mathbf{h}_2}))$ and an operator $B' \in L(\ell_2(G_{\mathbf{h}'_1}); \ell_2(G_{\mathbf{h}'_2}))$. The induction hypothesis tells us that B and B' can be computed using common step-sizes $\mathbf{h}_0$ and $\mathbf{h}'_0$. In particular, this statement implies that there exist $\mathbf{c}_1, \mathbf{c}'_1 \in \mathbb{N}^d$ such that

$$\begin{aligned} \mathbf{h}_2 &= \mathbf{c}_2 \cdot \mathbf{h}_0 \,, \\ \mathbf{h}'_1 &= \mathbf{c}'_1 \cdot \mathbf{h}'_0 \,. \end{aligned}$$

As we are able to multiply B and B', we have that $\mathbf{h}_2 = \mathbf{h}'_1$. Combining this equation with the previous one, we obtain that

$$\mathbf{c}_2 \cdot \mathbf{h}_0 = \mathbf{c}'_1 \cdot \mathbf{h}'_0 \,.$$

Thus, if we define $\mathbf{h}''_0 = \mathbf{h}_0/\mathbf{c}'_1 = \mathbf{h}'_0/\mathbf{c}_2$, then $\mathbf{h}_0$ and $\mathbf{h}'_0$ are integer multiples of $\mathbf{h}''_0$, implying that all integer multiples of $\mathbf{h}_0$ and $\mathbf{h}'_0$ can be written as integer multiples of $\mathbf{h}''_0$. Therefore, $\mathbf{h}''_0$ is the new common step-size.

It remains to prove the assertion for operators that involve just one Fourier matrix symbol, i.e., the scalar multiplication, the inverse and the adjoint. However, as these operators pose no constraints on the step-sizes, the common step-size is the same before and after application of the operator. □

For computational reasons it can be desired to choose $\mathbf{h}_0$ as large as possible, i.e., to make $\mathbf{h}_0$ as small as necessary but not smaller. Assume that in the computation we need the step-sizes $\mathbf{h}_1, \ldots, \mathbf{h}_k$, and we have a common step-size $\mathbf{h}_0$ such that

$$\mathbf{h}_j = \mathbf{c}_j \cdot \mathbf{h}_0 \quad \text{for} \quad j = 1, \ldots, k \,.$$

The smallest step-size $\mathbf{h}'_0$ is obtained by letting

$$\mathbf{c}'_j = \mathbf{c}_j / \gcd(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k) \quad \text{and} \quad \mathbf{h}'_0 = \mathbf{h}_0 \cdot \gcd(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k) \,.$$

With this definition,

$$\mathbf{h}_j = \mathbf{c}'_j \cdot \mathbf{h}'_0$$

and $\mathbf{c}_j \in \mathbb{N}^d$, showing that all step-sizes $\mathbf{h}_j$ are indeed integer multiples of $\mathbf{h}'_0$.

Using the fact that when evaluating an expression of Fourier matrix symbols, there exists only one step-size that defines all the others, we can show that for an expression of finite resolution representation matrices there is just one resolution, fixing all other resolutions. Recall that the input and output resolution and the input and output step-size of a finite resolution representation are related by

$$\mathbf{h}_{\text{out}}/\mathbf{h}_{\text{in}} = \mathbf{r}_{\text{in}}/\mathbf{r}_{\text{out}} \,. \qquad \text{(5.24 revisited)}$$

First of all note that if $\mathbf{h}_{\text{out}} = \mathbf{h}_{\text{in}}$, the previous equation implies

$$\mathbf{r}_{\text{in}} = \mathbf{r}_{\text{out}} \,.$$

Thus, in order to have matching resolutions of operators that map grid functions with a given step-size to a grid function with the same step-size, both grid functions must have the

same resolution, implying that there is only *one resolution per grid*. Let $\mathbf{r}_0$ be the resolution assigned to the grid with step-size $\mathbf{h}_0$.

We know that there exists a common step-size $\mathbf{h}_0$ such that

$$\mathbf{h}_{\text{in}} = \mathbf{c}_{\text{in}} \cdot \mathbf{h}_0 \quad \text{and} \quad \mathbf{h}_{\text{out}} = \mathbf{c}_{\text{out}} \cdot \mathbf{h}_0 \,.$$

Combining the last two equations with (5.24), we get that

$$\mathbf{r}_{\text{in}}/\mathbf{r}_{\text{out}} = \mathbf{h}_{\text{out}}/\mathbf{h}_{\text{in}} = (\mathbf{c}_{\text{out}} \cdot \mathbf{h}_0)/(\mathbf{c}_{\text{in}} \cdot \mathbf{h}_0) = \mathbf{c}_{\text{out}}/\mathbf{c}_{\text{in}} \,.$$

If we let $\mathbf{h}_{\text{in}} = \mathbf{h}_0$, i.e., $\mathbf{c}_{\text{in}} = \mathbf{1}$ and $\mathbf{r}_{\text{in}} = \mathbf{r}_0$, then

$$\mathbf{r}_{\text{in}}/\mathbf{r}_{\text{out}} = \mathbf{r}_0/\mathbf{r}_{\text{out}} = \mathbf{c}_{\text{out}} \,.$$

Thus,

$$\mathbf{r}_{\text{out}} = \mathbf{r}_0/\mathbf{c}_{\text{out}} \,.$$

In a similar way, by assuming that $\mathbf{c}_{\text{out}} = \mathbf{1}$, we get

$$\mathbf{r}_{\text{in}} = \mathbf{r}_0/\mathbf{c}_{\text{in}} \,.$$

In other words, the input and output resolution are completely determined by the resolution $\mathbf{r}_0$ that we assigned to the grid with step-size $\mathbf{h}_0$. The choice of $\mathbf{r}_0$, however, is not arbitrary.

Assume that we are evaluating an expression containing the operators

$$A_j : \ell_2(G_{\mathbf{c}_{\text{in},j} \cdot \mathbf{h}_0}) \to \ell_2(G_{\mathbf{c}_{\text{out},j} \cdot \mathbf{h}_0})$$

with corresponding $\mathbf{m}_j \times \mathbf{n}_j$ finite resolution representations matrices $\hat{a}_j$, for $j = 1, \ldots, k$. It is easy to see that if we define

$$\mathbf{r}_{0,\min} = \operatorname{lcm}(\mathbf{c}_{\text{in},1} \cdot \mathbf{n}_1, \ldots, \mathbf{c}_{\text{in},k} \cdot \mathbf{n}_k, \mathbf{c}_{\text{out},1} \cdot \mathbf{m}_1, \ldots, \mathbf{c}_{\text{out},k} \cdot \mathbf{m}_k), \tag{5.29}$$

any integer multiple of $\mathbf{r}_{0,\min}$ is an appropriate choice for $\mathbf{r}_0$. By appropriate we mean that the requirements of Theorem 5.17 and Theorem 5.18 are fulfilled for all pairs of operators $(A_j, A_{j'})$ whose step-sizes allow their multiplication or addition.

5.5.4. Evaluating Fourier Matrix Symbols

We shall now discuss how to compute a finite resolution approximation to a Fourier matrix symbol given by a formula. In contrast to Section 5.5.1, we need to enrich the expression tree of the formula before we can actually evaluate it.

The formula is evaluated by first approximating all Fourier matrix symbols, which appear in the formula, by finite resolution representation matrices, and then combining them according to the formula. In the previous section we have seen that we need a suitable resolution, and to compute this resolution, we need the input and output step-size and the rows and columns of all matrix symbols of the formula. Let us call these quantities the *properties of the matrix*

symbols. From these properties we can compute a suitable resolution and then evaluate the formula.

If we have the properties of the arguments of an operator, like the addition of multiplication operators, we can compute the properties that the resulting matrix symbol would have. For example, consider the general multiplication of finite resolution representations of Fourier matrix symbols, as defined in the General-Multiply procedure. In this case, Theorem 5.17 gives the properties of the product in terms of the properties of the two factors. The computation of the properties for the general multiplication is given in the Compute-Properties-Multiply procedure.

Compute-Properties-Multiply(p_1, p_2)

1 $p.\textit{out-step-size} \leftarrow p_1.\textit{out-step-size}$
2 $p.\textit{in-step-size} \leftarrow p_2.\textit{in-step-size}$
3 $q_1 \leftarrow \operatorname{lcm}(p_1.\textit{cols}, p_2.\textit{rows})/p_1.\textit{cols}$
4 $q_2 \leftarrow \operatorname{lcm}(p_1.\textit{cols}, p_2.\textit{rows})/p_2.\textit{rows}$
5 $p.\textit{rows} \leftarrow p_1.\textit{rows} \cdot q_1$
6 $p.\textit{cols} \leftarrow p_2.\textit{cols} \cdot q_2$
7 **return** p

For every operator we can define such a function that computes the properties of the resulting matrix symbol when given the properties of the arguments. Let us assume we have a function Compute-Properties($op, p_1, p_2, \dots$). Given the operator and the properties of its arguments, this function computes the properties of the matrix symbol that the operator would produce when applied to arguments with the properties p_1, p_2, ... Then we can recursively traverse the expression tree and enrich it with the information about the properties of the resulting symbol representations, which is what the Set-Properties procedure does.

Set-Properties(*expr*)

1 **if** *expr* **not in** *values*
2 $\quad (op, e_1, e_2, \dots) \leftarrow \textit{expr}$
3 $\quad$ Set-Properties(e_1)
4 $\quad$ Set-Properties(e_2)
5 $\quad \dots$
6 $\quad \textit{expr}.\textit{properties} \leftarrow$ Compute-Properties($op, e_1.\textit{properties}, e_2.\textit{properties}, \dots$)

Let us discuss an example. We consider the block Jacobi method in 1D and with block size 2. Furthermore, the operator of the linear system should be given by a constant stencil. The method was discussed in Section 4.2.1, and its error propagator is given by

$$E = I - \omega D^{-1} A\,. \tag{4.20 revisited}$$

In this case, the operator D has a corresponding 2×2 and the operator A a corresponding 1×1 Fourier matrix symbol. The enriched expression tree for this formula, as produced by the Set-Properties method, is shown in Figure 5.11. The procedure computes the properties

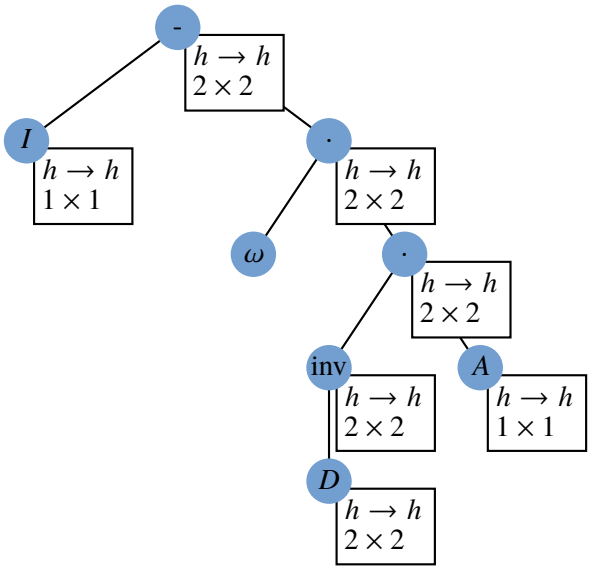


Figure 5.11.: Enriched expression tree of the expression $I - \omega D^{-1} A$.

from the bottom to the top of the tree. Whenever two operators are combined in a node and their rows or columns do not match, the rows and columns of this node have to be computed by the rules of Theorem 5.17 and Theorem 5.18, i.e., the matrix symbols have to be expanded properly. The expansion is necessary, for example, when D^{-1} and A are multiplied. The rows of A do not match the columns of D^{-1}. Consequently, the result has more rows and columns than A. From this enriched expression tree we can compute the suitable resolutions for the evaluate of the expression.

The resolution of all representations is determined by the resolution on the finest grid, as discussed in Section 5.5.3. Let $\mathbf{h}_0$ be the step-size of the finest grid. A feasible resolution is an integer multiple of $r_{0,\min}$, which is determined as follows. Assume that we have k nodes in the expression tree, and the jth node has the following properties: the input step-size $\mathbf{h}_{j,\text{in}} = \mathbf{c}_{j,\text{in}} \cdot \mathbf{h}_0$, the output step-size $\mathbf{h}_{j,\text{out}} = \mathbf{c}_{j,\text{out}} \cdot \mathbf{h}_0$, the rows $\mathbf{m}_j$, and the columns $\mathbf{n}_j$. Then as discussed in Section 5.5.3,

$$\mathbf{r}_{0,\min} = \operatorname{lcm}(\mathbf{c}_{\text{in},1} \cdot \mathbf{n}_1, \ldots, \mathbf{c}_{\text{in},k} \cdot \mathbf{n}_k, \mathbf{c}_{\text{out},1} \cdot \mathbf{m}_1, \ldots, \mathbf{c}_{\text{out},k} \cdot \mathbf{m}_k)\,. \qquad \text{(5.29 revisited)}$$

This value can be computed by traversing all nodes of the expression tree. We can use this value to determine the resolution on the finest grid.

The resolution determines the quality of the computed result. Therefore, a user wants to specify a *minimal resolution* to ensure a minimum quality of the result. Thus, we choose the smallest multiple of $r_{0,\min}$ that is larger than the resolution which the user requested.

This choice for the resolution ensures that when evaluating the expression tree, all operations involve matching resolutions. Hence, after choosing the resolution, we can evaluate the expression tree using the Eval procedure from Section 5.5.1 where the Apply procedure

uses the algorithms from Section 5.3. The result is a finite resolution representation matrix, which approximates the Fourier matrix symbol of the operator described by the expression.

5.6. Literature and Contributions

This chapter dealt with the automation of LFA. Another approach of automating LFA is described in [74], where an LFA software with a graphical user interface is presented. The main difference between this reference and this thesis is, that in this thesis we allow for the computation of arbitrary combinations of operators, while the software described in the reference only allows the evaluation of a fixed set of formulas.

For the purpose of allowing for arbitrary operator combinations, we combined the idea of approximating functions by finite resolution functions—a concept well known in applied mathematics—with the idea of evaluating expression trees—a concept will known in computer science.

To evaluate arbitrary formulas involving operators with Fourier matrix symbols, we need to compute general sums and products of these symbols. These operations require the proper expansion, which was introduced in this thesis. Furthermore, it is reasonable to fulfill certain requirements about the resolution of the functions that are used in the computation. These considerations and the proof that a proper choice of resolution is always possible is an original result of this thesis.

The algorithm for the computation of Fourier matrix symbols, introduced in Section 3.6, is also a new result. Furthermore, Theorem 5.10, which gives an estimate for the accuracy of the finite resolution approximation, is also new.

6. Conclusions

In this thesis, we have developed a general framework for local Fourier analysis of multigrid methods that is versatile and well suited for computer implementation. The two main results that made this framework possible are the equivalence of periodic stencil operators and operators with Fourier matrix symbols (Theorem 3.35), and that operators with Fourier matrix symbols are closed under many operations (Theorem 3.37). These results were shown by exploiting the fact that the operators given in position space have a corresponding representation in Fourier space.

More precisely, a discrete function on an infinite grid $G_{\mathbf{h}}$ is represented in Fourier space by a continuous function on the compact set $\Theta_{\mathbf{h}}$. These two representations are related by the discrete time Fourier transform giving a relation between the spaces $\ell_2(G_{\mathbf{h}})$ and $L_2(\Theta_{\mathbf{h}})$. Consequently, linear, bounded operators on these spaces are similarly related; every operator in $L(\ell_2(G_h); \ell_2(G_{h'}))$ has a Fourier representation in $L(L_2(\Theta_h); L_2(\Theta_{h'}))$. It turned out that there are two subsets of operators on grid functions whose Fourier representations can be written in a useful way—constant stencil and periodic stencil operators.

We have proven that the Fourier representation of constant stencil and periodic stencil operators can be written as symbols and matrix symbols. This form is important, as it allows us to compute the norm and spectral radius of the corresponding operators via

$$r(A) = \text{ess-sup}_{\theta \in \Theta_{\mathbf{n}\cdot\mathbf{h}}} r(\hat{a}(\theta)) \qquad \text{and} \qquad \|A\| = \text{ess-sup}_{\theta \in \Theta_{\mathbf{n}\cdot\mathbf{h}}} \|\hat{a}(\theta)\|, \quad \text{(3.25 revisited)}$$

where $\hat{a}$ is the Fourier matrix symbol of the operator A. These relations have been used in local Fourier analysis literature (see, e.g., [11, 74]). Still, we could not find the constraints or proofs for these statements in the literature. To close this gap, we decided to proof these important statements in this thesis.

Using the Fourier matrix symbols of periodic stencil operators, we were able to analyze multigrid methods which have not been considered to this point. We analyzed a multigrid method for a diffusion problem with jumping coefficients, and we analyzed various block smoothers. Furthermore, we showed that known results, like the analysis of the point-wise red-black Jacobi method, can be performed using periodic stencil operators.

As mentioned above, for the practical implementation of LFA in a computer software, it is important that Fourier matrix symbols are closed under many operations for the following reason. We know how to approximate Fourier matrix symbols on a computer. Hence, if we perform a computation, we want to be sure that the result is a Fourier matrix symbol. If the result would be something else, we would not be able to store and represent it on the machine.

We showed how to create a flexible software for the automation of local Fourier analysis. This flexibility was achieved by choosing approximations to Fourier matrix symbols as primitive components that could be combined into complicated expressions. In this way, many problems can be described, which are then analyzed by the software.

In summary, we have described a theoretical framework that allows for the analysis of complicated problems and a flexible computer implementation. This implementation permits the reuse of code and therefore simplifies the applicability of local Fourier analysis.

In the future, the software should allow us to analyze further complicated problems that have not yet been considered by local Fourier analysis. Furthermore, it is straightforward to widen the analysis to include systems of operators. In addition, the software uses an expression tree to represent the computations that should be performed. By applying algorithms that rewrite this tree, it should be possible to optimize the computation of the Fourier analysis, e.g., by removing redundant information or simplifying expressions.

A. Exponential Basis

Orthonormal bases are useful for different analytical tasks. In this chapter, we discuss an orthonormal basis of $L_2([0, 2\pi/h_1) \times \cdots \times [0, 2\pi/h_2); \mathbb{C})$, where $0 < \mathbf{h} \in \mathbb{R}d$, that uses the complex wave functions e^{it} ($t \in \mathbb{R}$). These are classical results that can be found, e.g., in the textbooks [4, 21, 42, 61, 72].

A.1. The one-dimensional Case

Let us begin with the one dimensional case and the space $L_2([0, 2\pi); \mathbb{C})$. We claim that the functions

$$\psi_k(s) := \tfrac{1}{\sqrt{2\pi}} e^{\mathrm{i}ks} \quad (\text{for } k \in \mathbb{Z})$$

form an orthonormal basis of $L_2([0, 2\pi); \mathbb{C})$. The orthonormality is easily verified by computing the integral

$$\langle \psi_k, \psi_j \rangle = \int_0^{2\pi} \tfrac{1}{\sqrt{2\pi}} e^{-\mathrm{i}js} \cdot \tfrac{1}{\sqrt{2\pi}} e^{\mathrm{i}ks} \, \mathrm{d}s = \delta_{kj} \, .$$

From this fact we already know that for a function that is given by

$$f := \sum_{k=-\infty}^{\infty} a_k \psi_k \, , \quad \text{where} \quad a_k \in \mathbb{C} , \tag{A.1}$$

the coefficients a_k fulfill $a_k = \langle f, \psi_k \rangle$.

It is also true that every function $f \in L_2([0, 2\pi); \mathbb{C})$ can be written as the sum (A.1). This statement, however, is more difficult to prove.

Lemma A.1. *Every function $f \in L_2([0, 2\pi); \mathbb{C})$ can be written as the sum* (A.1).

Sketch of the proof. This proof can be found in standard textbooks [4, 21, 42, 72]. Therefore, we will just outline one possible way to prove the assertion.

It can be proven that every step function on $[0, 2\pi)$ can be written as the sum (A.1) [21]. Thus, the set $\{\psi_k\}_{k=-\infty}^{\infty}$ is dense in the set of the step functions. As this set is dense in $L_2([0, 2\pi); \mathbb{C})$ also is $\{\psi_k\}_{k=-\infty}^{\infty}$. □

A.2. The d-dimensional Case

In the next step, let us generalize this basis to the space $L_2([0, 2\pi)^2; \mathbb{C})$. By using the results from section A.1, we shall show that

$$\psi_{\mathbf{k}}(\mathbf{s}) := \tfrac{1}{(2\pi)^{d/2}} e^{\mathrm{i}\langle \mathbf{k}, \mathbf{s} \rangle} \quad (\text{for } \mathbf{k} \in \mathbb{Z}^d)$$

is an orthonormal basis for $L_2([0,2\pi)^d;\mathbb{C})$.

We can use the exponentiation law to obtain

$$\begin{aligned}\psi_{\mathbf{k}}(\mathbf{s}) &= \tfrac{1}{(2\pi)^{d/2}} e^{\mathrm{i}\langle \mathbf{k},\mathbf{s}\rangle} \\ &= \tfrac{1}{(2\pi)^{d/2}} e^{\mathrm{i}k_1 s_1} e^{\mathrm{i}k_2 s_2} \dots e^{\mathrm{i}k_d s_d} \\ &= \psi_{k_1}(s_1)\cdot\psi_{k_2}(s_2)\cdots\psi_{k_d}(s_d)\,.\end{aligned}$$

Thus, the functions $\psi_{\mathbf{k}}$ are just the product of the basis functions from section A.1. By computing

$$\begin{aligned}\langle \psi_{\mathbf{k}}, \psi_{\mathbf{j}}\rangle &= \int_{[0,2\pi)^d} \overline{\psi_{j_1}(s_1)\cdots\psi_{j_d}(s_d)}\cdot\psi_{k_1}(s_1)\cdots\psi_{j_d}(s_d)\,\mathrm{d}\mathbf{s} \\ &= \langle \psi_{k_1}, \psi_{j_1}\rangle\cdots\langle \psi_{k_d}, \psi_{j_2}\rangle \\ &= \delta_{\mathbf{kj}}\,,\end{aligned}$$

we verify the orthonormality of $\{\psi_{\mathbf{k}}\}_{\mathbf{k}=-\infty}^{\infty}$.

Furthermore, $\{\psi_{\mathbf{k}}\}_{\mathbf{k}=-\infty}^{\infty}$ is also a basis of $L_2([0,2\pi)^d;\mathbb{C})$, as we will see in a minute. For this purpose, pick a function $f\in L_2([0,2\pi)^d;\mathbb{C})$. For almost all $x_2,\dots,x_d\in\mathbb{R}$ it is true that the mapping $s_1\mapsto f(s_1,x_2\dots,x_d)$ is a univariate function in $L_2([0,2\pi);\mathbb{C})$. Thus, if we use Lemma A.1, we have that there exists $a_{k_1}(x_2,\dots,x_d)\in\mathbb{C}$ such that

$$f(s_1,x_2,\dots,x_d) = \sum_{k_1=-\infty}^{\infty} a_{k_1}(x_2,\dots,x_d)\cdot\psi_{k_1}(s_1)\,. \tag{A.2}$$

The same argument applies to the functions a_{k_1}. We can write them as

$$a_{k_1}(s_2,x_3,\dots,x_d) = \sum_{k_2=-\infty}^{\infty} a_{k_1k_2}(x_3,\dots,x_d)\cdot\psi_{k_2}(s_2) \quad \text{for } k_1\in\mathbb{Z}\,.$$

We can substitute this relation into the first sum (A.2) and obtain that

$$f(s_1,s_2,x_3\dots,x_d) = \sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} a_{k_1k_2}(x_3,\dots,x_d)\cdot\psi_{k_2}(s_2)\psi_{k_1}(s_1)\,.$$

We can now repeat this with $a_{k_1k_2}$, $a_{k_1k_2k_3}$, ... and finally arrive at

$$f = \sum_{\mathbf{k}=-\infty}^{\infty} a_{\mathbf{k}}\cdot\psi_{k_1}\cdots\psi_{k_d} = \sum_{\mathbf{k}=-\infty}^{\infty} a_{\mathbf{k}}\cdot\psi_{\mathbf{k}}\,.$$

Therefore, $\{\psi_{\mathbf{k}}\}_{\mathbf{k}=-\infty}^{\infty}$ is an orthonormal basis of $L_2([0,2\pi)^d;\mathbb{C})$.

A.3. Changing the Domain

We have an orthonormal basis for $\Psi := [0, 2\pi)^d$, and we would like to have an orthonormal basis for $\Theta_{\mathbf{h}} := [0, 2\pi/h_1) \times \cdots \times [0, 2\pi/h_d)$, because this is the space we use in local Fourier analysis. To obtain this basis, we construct an isometry between $L_2(\Psi; \mathbb{C})$ and $L_2(\Theta_{\mathbf{h}}; \mathbb{C})$.

Lemma A.2. *The mapping $T : \Psi \to \Theta_{\mathbf{h}}$ given by*

$$(Tf)(\mathbf{x}) := \mathrm{vol}_{\mathbf{h}}^{1/2} \cdot f(\gamma(\mathbf{x}))$$

is an isometry.

Proof. Let $f, g \in L_2(\Psi; \mathbb{C})$. As T is linear, we just need to show that

$$\int_{\Psi} \overline{f} g \, \mathrm{d}\mathbf{s} =: \langle f, g \rangle = \langle Tf, Tg \rangle := \int_{\Theta_{\mathbf{h}}} \overline{Tg} \cdot Tf \, \mathrm{d}\mathbf{t} \,.$$

We prove this by constructing the mapping T from a variable substitution. Let $\gamma : \Theta_{\mathbf{h}} \to \Psi$ and

$$\gamma(\mathbf{x}) := h_1 x_2 \, \mathrm{d}x_1 + \cdots + h_d x_d \, \mathrm{d}x_d \,.$$

Then the variable substitution given by γ yields

$$\int_{\Psi} \overline{g(\mathbf{s})} \cdot f(\mathbf{s}) \, \mathrm{d}s = \int_{\gamma^{-1}(\Psi)} \overline{g(\gamma(\mathbf{t}))} \cdot f(\gamma(\mathbf{t})) \cdot |\det \gamma'(\mathbf{t})| \, \mathrm{d}\mathbf{t} \,.$$

A simple computation yields that $\det(\gamma'(t)) = \mathrm{vol}_{\mathbf{h}}$. By distributing this factor onto the two functions f and g, we obtain that

$$\int_{\Psi} \overline{g(\mathbf{s})} \cdot f(\mathbf{s}) \, \mathrm{d}s = \int_{\Theta_{\mathbf{h}}} \overline{\mathrm{vol}_{\mathbf{h}}^{1/2} g(\gamma(\mathbf{t}))} \cdot \mathrm{vol}_{\mathbf{h}}^{1/2} f(\gamma(\mathbf{t})) \, \mathrm{d}t \,.$$

Therefore, the mapping T is an isometry. □

By using the mapping, T can transform the orthonormal basis $\psi_{\mathbf{k}}$ of Ψ into an orthonormal basis $T(\psi_{\mathbf{k}})$ of $\Theta_{\mathbf{h}}$, proving the following theorem.

Theorem A.3. *The set of functions*

$$(T\psi_{\mathbf{k}})(\mathbf{t}) = \frac{\mathrm{vol}_{\mathbf{h}}^{1/2}}{(2\pi)^{d/2}} e^{\mathrm{i}\langle \mathbf{k}, \mathbf{h} \cdot \mathbf{t} \rangle}$$

is an orthonormal basis of the space $L_2(\Theta_{\mathbf{h}}; \mathbb{C})$.

Nomenclature

$\mathbb{N}$	The natural numbers.
$\mathbb{Z}$	The integer numbers.
$\mathbb{R}$	The real numbers.
$\mathbb{C}$	The complex numbers.
i	The imaginary unit.
$\overline{a}$	The complex conjugate of the complex number a.
$\#S$	Cardinality of the set S.
χ_M	The characteristic function on the set M.
$\delta_{\mathbf{ij}}$	The Kronecker delta.
$U_\epsilon(x)$	The open unit sphere of radius ϵ with center x.
$[a]$	The equivalence class of a w.r.t. some equivalence relation.
$\mathrm{vol}_{\mathbf{n}}$	The volume of the hyper-cube with sides $n_1, \ldots, n_d$.
$\mathrm{lcm}(a, b)$	The least common multiple of a and b.
$\gcd(a, b)$	The greatest common divisor of a and b.
I	The identity matrix/operator.
$\mathrm{span}\{v_1, \ldots, v_n\}$	The span of the vectors $v_1, \ldots, v_n$.
$\langle a, b\rangle$	The scalar product of a and b.
$\|a\|$	The norm of a.
$\mathrm{diag}(w_1, \ldots, w_n)$	The diagonal matrix with diagonal entries $w_1, \ldots, w_n$.
$\overline{X}$	The closure of the set X.
$C^p[a, b]$	The space of p-times continuously differentiable functions on the interval $[a, b]$.
μ	The Lebesgue measure.
$L_2(X)$	The square Lebesgue integrable functions on X.
$L_\infty(X)$	The essentially bounded functions on X.
$L(H)$	The set of linear, bounded operators $A : H \to H$.
$L(H_1; H_2)$	The set of linear, bounded operators $A : H_1 \to H_2$.
A^*	The adjoint operator of A.
$\sigma(A)$	The spectrum of A.
$\rho(A)$	The resolvent set of A.
ess-im	The essential range.
ess-sup	The essential supremum.
$L_\infty^{\mathbf{m}\times\mathbf{n}}(X)$	The set of $\mathbf{m} \times \mathbf{n}$ matrices with entries in $L_\infty(X)$.
$L_2^{\mathbf{n}}(X)$	The set of vectors of length $\mathbf{n}$ with entries in $L_2(X)$.
Δ	The Laplace operator.
$\Delta_{\mathbf{h}}$	The discrete Laplace operator.

∇u	The gradient of the function u.
$\nabla \cdot w$	The divergence of the field w.
Ω	The domain of a PDE.
$\partial\Omega$	The boundary of the domain of a PDE.
$\Omega_{\mathbf{h}}$	Inner points of the discrete domain.
$\overline{\Omega}_{\mathbf{h}}$	The whole discrete domain.
$\partial\Omega_{\mathbf{h}}$	The boundary of the discrete domain.
$G_{\mathbf{h}}$	The infinite grid with step-size $\mathbf{h}$.
$\ell_2(G_{\mathbf{h}})$	Bounded functions on the infinite grid $G_{\mathbf{h}}$.
$\Theta_{\mathbf{h}}$	The visible frequencies on a grid with step-size $\mathbf{h}$.
$L_2(\Theta_{\mathbf{h}})$	Bounded functions on the frequencies.
ϕ_θ	The wave function with frequency θ.
$[\theta]$	The set of all $\mathbf{n}$,$\mathbf{h}$-harmonics of θ.
θ_ℓ	The low $\mathbf{n}$,$\mathbf{h}$-harmonic corresponding to θ.
θ_b	The base frequency of the frequency θ.
$\mathcal{F}_{\mathbf{h}}$	The discrete time Fourier transform.
$\dot{\mathcal{F}}_{\mathbf{h}}$	The element-wise discrete time Fourier transform.
$\widehat{A}$	The Fourier representation of the operator A.
$\hat{a}$	The Fourier (matrix) symbol of the operator A.
$\mathbf{s}_{\mathbf{j}}^{\mathbf{h},\mathbf{n}}$	The frequency shifts.
$\mathbf{s}_{\mathbf{j}}$	The frequency shifts.
$\mathbf{t}_{\mathbf{k}}$	The space shifts.
$\mathcal{R}_{\mathbf{n}}$	The frequency splitting operator.
$T_{\mathbf{r}}$	The set of all resolution $\mathbf{r}$ functions.
$T_{\mathbf{r}}^{\mathbf{m}\times\mathbf{n}}$	The set of all $\mathbf{m} \times \mathbf{n}$ matrices with entries in $T_{\mathbf{r}}$.

Index

Bibliography

[1] H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 2nd edition, 1996.

[2] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman. *Compiler: Prinzipien, Techniken und Werkzeuge*. Addison-Wesley, 2008.

[3] R. Alcouffe, A. Brandt, J. Dendy, Jr., and J. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Statist. Comput.*, 2(4):430–454, 1981. doi: 10.1137/0902035.

[4] H. W. Alt. *Lineare Funktionalanalysis*. Springer, 6th edition, 2012.

[5] P. J. Antsaklis and A. N. Michel. *Linear Systems*, volume 2. Birkhäuser, 2006.

[6] W. Arendt and K. Urban. *Partielle Differentialgleichungen. Eine Einführung in analytische und numerische Methoden*. Spektrum, 2010.

[7] A. Baker, R. Falgout, T. Kolev, and U. Yang. Multigrid smoothers for ultraparallel computing. *SIAM J. Sci. Comput.*, 33(5):2864–2887, 2011. doi: 10.1137/100798806.

[8] A. Böttcher and B. Silbermann. *Analysis of Toeplitz Operators*. Springer Monographs in Mathematics. Springer, 2nd edition, 2006.

[9] H. Bin Zubair, C. W. Oosterlee, and R. Wienands. Multigrid for high-dimensional elliptic partial differential equations on non-equidistant grids. *SIAM J. Sci. Comput.*, 29 (4):1613–1636, 2007. doi: 10.1137/060665695.

[10] D. Braess. *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer Spektrum, 5th edition, 2013.

[11] A. Brandt. Rigorous quantitative analysis of multigrid, I: Constant coefficients two-level cycle with L_2-norm. *SIAM J. Numer. Anal.*, 31(6):1695–1730, 1994. doi: 10.1137/0731087.

[12] A. Brandt and O. E. Livne. *Multigrid Techniques. 1984 Guide with Applications to Fluid Dynamics*. SIAM, 2011.

[13] J. Brannick, X. Hu, C. Rodrigo, and L. Zikatanov. Local Fourier analysis of multigrid methods with polynomial smoothers and aggressive coarsening. *Numer. Math. Theory Methods Appl.*, 8(1):1–21, 2015. doi: 10.4208/nmtma.2015.w01si.

[14] M. Brezina, P. Vaněk, and P. S. Vassilevski. An improved convergence analysis of smoothed aggregation algebraic multigrid. *Numer. Linear Algebra Appl.*, 19(3):441–469, 2012. doi: 10.1002/nla.775.

[15] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, 2nd edition, 2000.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[17] J. E. Dendy, Jr. Black box multigrid. *J. Comput. Phys.*, 48(3):366 – 386, 1982. doi: 10.1016/0021-9991(82)90057-2.

[18] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 2nd edition, 2010.

[19] S. J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Dover Publications, 1993.

[20] O. Forster. *Analysis 3. Maß- und Integrationstheorie, Integralsätze im* $\mathbb{R}^n$ *und Anwendungen*. Springer Spektrum, 7th edition, 2012.

[21] O. Forster. *Analysis 1. Differential- und Integralrechnung in einer Veränderlichen*. Springer Spektrum, 11th edition, 2013.

[22] A. P. French. *Vibrations and Waves*. CBS Publishers & Distributors, 2003.

[23] S. Friedhoff and S. MacLachlan. A generalized predictive analysis tool for multigrid methods. *Numer. Linear Algebra Appl.*, 22(4):618–647, 2015. doi: 10.1002/nla.1977.

[24] S. Friedhoff, S. MacLachlan, and C. Börgers. Local Fourier analysis of space-time relaxation and multigrid schemes. *SIAM J. Sci. Comput.*, 35(5):S250–S276, 2013. doi: 10.1137/120881361.

[25] M. J. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. Sci. Comput.*, 38(4):A2173–A2208, 2016. doi: 10.1137/15M1046605.

[26] F. J. Gaspar, J. L. Gracia, and F. J. Lisbona. Fourier analysis for multigrid methods on triangular grids. *SIAM J. Sci. Comput.*, 31(3):2081–2102, 2009. doi: 10.1137/080713483.

[27] F. J. Gaspar, J. L. Gracia, F. J. Lisbona, and C. Rodrigo. On geometric multigrid methods for triangular grids using three-coarsening strategy. *Appl. Numer. Math.*, 59 (7):1693–1708, 2009. doi: 10.1016/j.apnum.2009.01.003.

[28] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Dover Publications, 2003.

[29] H. Goldstein, C. P. Poole, and J. L. Safko. *Classical Mechanics*. Prentice Hall, 3rd edition, 2013.

[30] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[31] W. Hackbusch. On the regularity of difference schemes. *Ark. Mat.*, 19(1):71–95, 1981. doi: 10.1007/BF02384470.

[32] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag, 1985.

[33] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen*. Springer Spektrum, 4th edition, 2017. doi: 10.1007/978-3-658-15358-8.

[34] R. W. Hamming. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, 2nd edition, 1973.

[35] P. W. Hemker. Fourier analysis of gridfunctions, prolongations and restrictions. CWI Technical Report NW 93/80, Stichting Mathematisch Centrum. Numerieke Wiskunde., Kruislaan 413, 1098 SJ, Amsterdam, November 1980. `http://oai.cwi.nl/oai/asset/8975/8975A.pdf`.

[36] P. W. Hemker and M. H. van Raalte. Fourier two-level analysis for higher dimensional discontinuous Galerkin discretisation. *Comput. Vis. Sci.*, 7(3-4):159–172, 2004. doi: 10.1007/s00791-004-0136-1.

[37] P. W. Hemker, W. Hoffmann, and M. H. van Raalte. Two-level Fourier analysis of a multigrid approach for discontinuous Galerkin discretization. *SIAM J. Sci. Comput.*, 25 (3):1018–1041, 2003. doi: 10.1137/S1064827502405100.

[38] P. W. Hemker, W. Hoffmann, and M. H. van Raalte. Fourier two-level analysis for discontinuous Galerkin discretization with linear elements. *Numer. Linear Algebra Appl.*, 11(5-6):473–491, 2004. doi: 10.1002/nla.356.

[39] A. Holderrieth. Matrix multiplication operators generating one parameter semigroups. *Semigroup Forum*, 42(2):155–166, 1991. doi: 10.1007/BF02573417.

[40] M. H. Holmes. *Introduction to Numerical Methods in Differential Equations*. Texts in Applied Mathematics. Springer, 2007.

[41] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Publications, 2009.

[42] W. Kaballo. *Grundkurs Funktionalanalysis*. Spektrum, 2011.

[43] K. Königsberger. *Analysis 2*. Springer, 2nd edition, 1997.

[44] P. Knabner and L. Angermann. *Numerik partieller Differentialgleichungen. Eine anwendungsorientierte Einführung*. Springer, 2000.

[45] C.-C. J. Kuo and B. C. Levy. Two-color Fourier analysis of the multigrid method with red-black Gauss-Seidel smoothing. *Appl. Math. Comput.*, 29(1):69–87, 1989. doi: 10.1016/0096-3003(89)90040-4.

[46] S. Larsson and V. Thomée. *Partial Differential Equations with Numerical Methods*. Texts in Applied Mathematics. Springer, 2009. doi: 10.1007/978-3-540-88706-5.

[47] O. E. Livne and A. Brandt. Formal fourier analysis of multicolor and composite relaxation schemes. Technical Report GMC-14, The Carl F. Gauss Minerva Center for Scientific Computation, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, 234 Herzl Street, Rehovot 7610001 Israel, Feb. 2002. `http://www.wisdom.weizmann.ac.il/~achi/gmc.html`.

[48] O. E. Livne and A. Brandt. Local mode analysis of multicolor and composite relaxation schemes. *Comput. Math. Appl.*, 47(2-3):301–317, 2004. doi: 10.1016/S0898-1221(04)90026-0.

[49] S. P. MacLachlan and C. W. Oosterlee. Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs. *Numer. Linear Algebra Appl.*, 18 (4):751–774, 2011. doi: 10.1002/nla.762.

[50] U. Meier Yang. On long-range interpolation operators for aggressive coarsening. *Numer. Linear Algebra Appl.*, 17(2-3):453–472, 2010. doi: 10.1002/nla.689.

[51] C.-D. Munz and T. Westermann. *Numerische Behandlung gewöhnlicher und partieller Differentialgleichungen. Ein interaktives Lehrbuch für Ingenieure*. Springer, 3rd edition, 2012.

[52] A. Niestegge and K. Witsch. Analysis of a multigrid Stokes solver. *Appl. Math. Comput.*, 35(3):291–303, 1990. doi: 10.1016/0096-3003(90)90048-8.

[53] Y. Notay and P. S. Vassilevski. Recursive krylov-based multigrid cycles. *Numer. Linear Algebra Appl.*, 15(5):473–487, 2008. doi: 10.1002/nla.542.

[54] R. H. Risch. The problem of integration in finite terms. *Trans. Amer. Math. Soc.*, 139: 167–189, 1969. doi: 10.1090/S0002-9947-1969-0237477-8.

[55] R. H. Risch. The solution of the problem of integration in finite terms. *Bull. Amer. Math. Soc.*, 76:605–608, 1970. doi: 10.1090/S0002-9904-1970-12454-5.

[56] T. J. Rivlin. *An Introduction to the Approximation of Functions*. Dover Publications, 1981.

[57] W. Rudin. *Functional Analysis*. McGraw-Hill Series in Higher Mathematics. McGraw-Hill, 1973.

[58] J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987. doi: 10.1137/1.9781611971057.ch4.

[59] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

[60] H. R. Schwarz and N. Köckler. *Numerische Mathematik*. Vieweg+Teubner, 8th edition, 2011.

[61] B. Schweizer. *Partielle Differentialgleichungen. Eine anwendungsorientierte Einführung*. Springer Spektrum, 2013.

[62] K. Stüben. Algebraic multigrid (AMG): An introduction with applications. Technical Report GMD-Report 70, German National Research Center for Information Technology (GMD), Institute for Algorithms and Scientific Computing (SCAI), Schloss Birlinghoven, D-53754, St. Augustin, Germany, November 1999.

[63] K. Stüben and U. Trottenberg. Multigrid methods: fundamental algorithms, model problem analysis and applications. In *Multigrid methods (Cologne, 1981)*, volume 960 of *Lecture Notes in Math.*, pages 1–176. Springer, Berlin-New York, 1982.

[64] R. Stevenson. *On the Validity of Local Mode Analysis of Multi-Grid Methods*. phdthesis, Utrecht University, December 1990.

[65] C. Strachey. Fundamental concepts in programming languages. *High.-Order Symb. Comput.*, 13(1-2):11–49, 2000. doi: 10.1023/A:1010000313106.

[66] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. W. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–532. Academic Press, 2001.

[67] J. R. Taylor. *Classical Mechanics*. University Science Books, 2004.

[68] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[69] G. van Rossum. Python reference manual. Technical Report CS-R9525, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, April 1995.

[70] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65(1):138–158, 1986. doi: 10.1016/0021-9991(86)90008-2.

[71] R. Weinstock. *Calculus of Variations. With Applications to Physics & Engineering*. Dover Publications, 1974.

[72] D. Werner. *Funktionalanalysis*. Springer, 7th edition, 2011.

[73] P. Wessling. *Principles of Computational Fluid Dynamics*. Number 29 in Springer Series in Computational Mathematics. Springer, 2001.

[74] R. Wienands and W. Joppich. *Practical Fourier Analysis For Multigrid Methods*. Chapman Hall/CRC Press, 2005.

[75] R. Wienands and C. W. Oosterlee. On three-grid Fourier analysis for multigrid. *SIAM J. Sci. Comput.*, 23(2):651–671, 2001. doi: 10.1137/S106482750037367X.

[76] R. Wienands, C. W. Oosterlee, and T. Washio. Fourier analysis of GMRES(m) preconditioned by multigrid. *SIAM J. Sci. Comput.*, 22(2):582–603, 2000. doi: 10.1137/S1064827599353014.

[77] I. Yavneh. Multigrid smoothing factors for red-black Gauss-Seidel relaxation applied to a class of elliptic operators. *SIAM J. Numer. Anal.*, 32(4):1126–1138, 1995. doi: 10.1137/0732051.

[78] H. D. Young and R. A. Freedman. *Sears & Zemansky's University Physics. With Modern Physics*. Pearson, 14th edition, 2016.