2020

Optimizing multigrid smoothers using GLT theory

Denhanh Huynh

Department of Numerical Analysis, Lund University
Master's thesis supervised by Philipp Birken

LUND
UNIVERSITY

Abstract

Multigrid algorithms are algorithms used to find numerical solutions to differential equations using a hierarchy of grids of different coarseness. This exploits the fact that short-wavelength components of the solutions converges at a faster rate than the long-wavelength components when using some basic iterative methods, such as the Jacobi method or the Gauss-Seidel method. These iterative methods are also called smoothers since they have the effect of smoothing out the errors.

In this thesis, two classes of smoothers based on time integration methods are studied using the one dimensional linear advection equation as model problem. The first class is a class based on explicit Runge-Kutta methods, and the other class is derived from considering implicit Runge-Kutta methods. For both classes it is possible to derive a matrix $\mathbf{M}$, dependent on the coefficient function in the linear advection problem and the parameters of the smoother, which describes the evolution of the error. To optimize the smoothers parameters are chosen so that the eigenvalues of $\mathbf{M}$ are optimized.

If the coefficient function is constant it is possible to derive a closed form expression for the eigenvalues of the resulting matrix $\mathbf{M}$. However, in the variable coefficient case it is not possible, and for large matrices it is impractical to compute the eigenvalues using iterative methods. Therefore, the theory of *Generalized Locally Toeplitz (GLT) sequences* is used to instead approximate the distribution of the eigenvalues. This results in an approximate optimization problem. The results show that this is an effective method for obtaining parameters for the smoothers in the variable coefficient case.

Populärvetenskaplig sammanfattning

Många fysiska problem, så som väderprognoser, luftflöden runt en flygplansvinge och vattenvågor, kan modelleras av differentialekvationer. Dessa kan inte alltid lösas analytiskt, men idag är våra datorer kraftfulla nog att lösa många av dessa modeller numeriskt, dvs. genom att diskretisera problemet och approximera lösningen på ett rutnät. Lösningarna för modellerna kan beskrivas som summor av vågor med långa och korta våglängder. När man använder simpla iterativa lösare, också kallade för *smoothers*, så konvergerar de korta våglängderna fortare än de långa. Problemet med dessa lösare är just den långsamma konvergensen av de långa våglängderna. Dessa konvergerar snabbare om man använder grövre nät. I *multigrid-algoritmer* använder man sig därför av flera rutnät av olika täthet för att få snabb konvergens av både korta och långa våglängder av lösningen. Syftet med det här arbetet är att studera hur parametrar kan väljas för två andra typer av smoothers med hjälp av något kallat GLT-teori (Generalized Locally Toeplitz theory).

Acknowledgements

I would like to offer my sincere thanks to my supervisor Professor Philipp Birken at the Centre for Mathematical Sciences at Lund University. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. He also helped improving my writing by offering valuable feedback.

I would also like to thank Dr. Fabio Durastante for helping me understand the article he coauthored, on which a lot of this thesis was based on.

Finally, I would like to thank my dear friend MSc Johan Book for his help, especially with the formulation in the popular science summary in Swedish.

Contents

Chapter 1

Introduction

To find the numerical solution of unsteady flow problems one can use a multigrid strategy. When using basic iterative methods to solve these problems, such as the Jacobi method or the Gauss-Seidel method, the short-wavelength components of the solutions converge faster than the long-wavelength components. Since this has the effect of smoothing out the error, these methods are called smoothers. The bottleneck of these methods is the slow convergence of the long-wavelength components. To circumvent this problem, multigrid algorithms uses several grids of different coarseness to speed up the convergence of the long-wavelength components.

These methods were shown by Caughey and Jameson [1] to be able to solve steady Euler flows in three to five multigrid cycles, which can be executed in a matter of seconds on a PC. However, with the multigrid strategy tuned for steady flow problems, the convergence rate for unsteady flow problems is deteriorated.

In [2], Birken showed that the convergence rate could be improved by optimizing the smoother of the multigrid method to unsteady flow problems. For this Birken used a class of smoothers based on explicit Runge-Kutta (RK) schemes, referred to as RK smoothers, which have low storage costs and scale well in parallel. It is possible to derive a matrix $\mathbf{M}$ created from the stability polynomial of the Runge-Kutta scheme which describes how the error evolves with each smoothing step. The RK smoothers are optimized by choosing parameters for the RK scheme such that the eigenvalues of $\mathbf{M}$ (corresponding to the short-wavelength components) are minimized. Thus, the optimization of these smoothers requires knowledge about the eigenvalues of $\mathbf{M}$.

The constant coefficient linear advection equation, considered in [2], results in a matrix $\mathbf{M}$ for which an explicit expression for the eigenvalues can be determined analytically. However, this is not possible in the variable coefficient case, and if very fine grids are considered in the discretization of the problem, it is impractical to solve for the eigenvalues iteratively, since the matrices are large. Therefore, Bertaccini et al. proposed using the theory of Generalized Locally Toeplitz (GLT) matrix sequences [3] to instead approximate the distribution of

the eigenvalues and thus generalizing the strategy to variable coefficient convection–diffusion equations [4].

GLT matrix sequences are built up by combining Locally Toeplitz sequences which in turn are generalizations of Toeplitz matrix sequences. A GLT sequence is described by a function, called the symbol of the GLT sequence. This symbol describes the asymptotic singular value distribution of the matrix sequence. With some additional assumptions on the matrices in the sequence, the symbol also describes the asymptotic eigenvalue distribution of the sequence.

The optimization of two different classes of smoothers is studied, using the GLT theory, with the one dimensional variable coefficient linear advection equation as model problem. The first one is the class of explicit Runge-Kutta smoothers as mentioned above, and the other one is a class of W-smoothers (the convective part of the additive W-methods in [5]) which is a smoother developed from considering implicit Runge-Kutta schemes. There is no unique set of optimal parameters for these smoothers that work for all problems [2], so improving the method of solving for the parameters plays a role in the efficiency of these smoothers.

In chapter 2 the model problem and the discretization of the problem is described in detail. The discretization of the model problem results in a linear system of the form $\mathbf{Au} = \mathbf{b}$. Chapter 3 explains how to solve this system using multigrid methods. The RK smoothers are described in chapter 4 as well as how to optimize them in the constant coefficient case. In chapter 5 the GLT theory is explained. Here, the eigenvalue distribution of matrices of different sizes from the same GLT sequence are compared to the symbol of the matrix sequence to see how well the symbol approximates the distribution of the eigenvalues of the matrices. Different coefficient functions result in different GLT sequences and symbols. Therefore, this is done for some different coefficient functions. The GLT theory is used to set up an approximate optimization problem for the parameters of the RK smoothers in the variable coefficient case in chapter 6. Here, a method for creating a lookup table for the parameters of the smoothers is also proposed. Chapter 7 introduces the W-smoothers, and describes how to optimize these smoothers, both in the constant coefficient case and the variable coefficient case. Numerical results are given with some discussion in chapter 8. Finally, some conclusions are given in chapter 9.

Chapter 2

Model problem and discretization

The model problem used in this thesis is the one dimensional linear advection equation with periodic boundary conditions, i.e.

$$u_t(x,t) + (a(x)u(x,t))_x = 0, \qquad a(x) > 0,\ a(x) \in \mathbb{R}\ \forall\ x \in [x_{min}, x_{max}], \tag{2.1}$$

$$u(x = x_{min}, t) = u(x = x_{max}, t), \tag{2.2}$$

which describes the bulk motion of a wave, and can be solved given an initial value

$$u(x, t = 0) = u_0. \tag{2.3}$$

In the above equation, the subscripts t and x refers to partial derivatives with respect to the time variable t and spatial variable x.

2.1 Constant coefficient advection equation

Here, the boundaries of x are set to $x_{min} = 0$ and $x_{max} = 1$ for simplicity, as this can be done without loss of generality. To solve (2.1) numerically, the grid is discretized using $n + 1$ equidistant grid points creating n grid cells of width $\Delta x = 1/n$ (Figure 2.1). Here the grid points $x_{i+1/2}$ are defined as $x_{i+1/2} = i\Delta x$, where half integers are used so that grid cell i ranges from $x_{i-1/2}$ to $x_{i+1/2}$. The boundary grid points corresponding to 0 and 1 are thus $x_{1/2} = 0$ and $x_{n+1/2} = 1$.

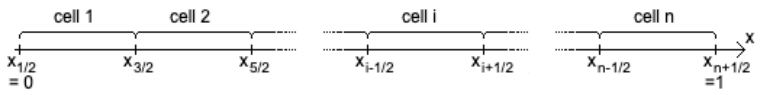


Figure 2.1: Discretization of the grid.

First, the case where $a(x) \equiv a$ is constant is considered. To get a finite volume scheme, the advection equation (2.1) is integrated over each grid cell:

$$\begin{aligned} 0 &= \int_{x_{i-1/2}}^{x^{i+1/2}} u_t(x,t)dx + \int_{x_{i-1/2}}^{x^{i+1/2}} (au(x,t))_x dx \\ &= \int_{x_{i-1/2}}^{x^{i+1/2}} u_t(x,t)dx + a(u(x_{i+1/2},t) - u(x_{i-1/2},t)) \Leftrightarrow \\ 0 &= (u_i(t))_t + \frac{a}{\Delta x}(u(x_{i+1/2},t) - u(x_{i-1/2},t)), \qquad i = 1,2,...,n, \end{aligned} \tag{2.4}$$

where u_i is the cell average of cell i, i.e. $u_i(t) := \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t)dx$, $i = 1,...,n$. Let the solution vector be given by $\mathbf{u} = (u_1, u_2, ..., u_n)^T = (u_1(t), u_2(t), ..., u_n(t))^T$, which represents a step function (Figure 2.2).

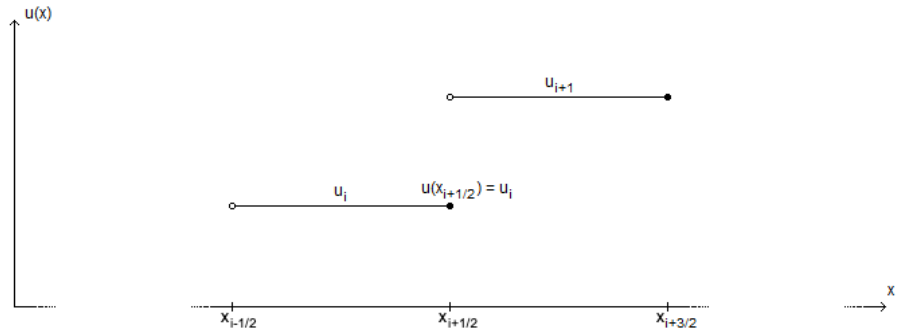


Figure 2.2: Part of the solution vector $\mathbf{u} = (u_1, u_2, ..., u_n)^T$. The value of $u(x_{i+1/2})$ is chosen to be represented by the value of the step function on the left side, i.e. u_i.

To represent $u(x_{i+1/2})$ and $u(x_{i-1/2})$ in terms of the values of this step function, one must choose to either use the value of the step function on the left or right side of the grid point. Using an upwind scheme, the function evaluations are weighted towards the side where the information is coming from. Since $a > 0$, the wave is travelling in the positive x-direction. Therefore, the left value is chosen. This leads to the system of ODEs

$$(u_i)_t + \frac{a}{\Delta x}(u_i - u_{i-1}) = 0, \qquad i = 1,2,...,n, \tag{2.5}$$

which is an approximation of (2.4). Here $u_0 := u_n$ because of the periodic boundary conditions. This system of ODEs can be written in matrix form

$$\mathbf{u}_t + \frac{a}{\Delta x}\tilde{\mathbf{B}}_n\mathbf{u} = \mathbf{0}, \tag{2.6}$$

where $\tilde{\mathbf{B}}_n$ is an $n \times n$ matrix given by

$$\tilde{\mathbf{B}}_n = \begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}, \tag{2.7}$$

and $\mathbf{u} = (u_1, u_2, ..., u_n)^T$ as defined above. Let superscripts denote the current time step. To get the fully discretized form of the linear advection equation, (2.6) is discretized in time using an implicit Euler step of size Δt. This leads to the linear system

$$\begin{aligned} \mathbf{u}^{k+1} &= \mathbf{u}^k + \Delta t \mathbf{u}_t^{k+1} \\ &= \mathbf{u}^k - \frac{a\Delta t}{\Delta x}\tilde{\mathbf{B}}_n \mathbf{u}^{k+1} \Leftrightarrow \\ \left(\mathbf{I}_n + \frac{a\Delta t}{\Delta x}\tilde{\mathbf{B}}_n\right)\mathbf{u}^{k+1} &= \mathbf{u}^k \Leftrightarrow \\ \tilde{\mathbf{A}}_n \mathbf{u}^{k+1} &= \mathbf{u}^k, \end{aligned} \tag{2.8}$$

where

$$\tilde{\mathbf{A}}_n := \mathbf{I}_n + \frac{a\Delta t}{\Delta x}\tilde{\mathbf{B}}_n. \tag{2.9}$$

Here $\mathbf{I}_n$ denotes the $n \times n$ unit matrix.

2.2 Variable coefficient advection equation

If a is not assumed to be constant (still with $a(x) > 0 \; \forall \; x$), the finite volume upwind scheme for (2.1) becomes

$$\begin{aligned} &(u_1)_t + \frac{1}{\Delta x}(a_{1+1/2}u_1 - a_{n+1/2}u_n) = 0 \\ &(u_i)_t + \frac{1}{\Delta x}(a_{i+1/2}u_i - a_{i-1/2}u_{i-1}) = 0, \qquad i = 2, ..., n, \end{aligned} \tag{2.10}$$

where $a_{i+1/2} := a(x_{i+1/2})$, and the corresponding matrix form is given by

$$\mathbf{u}_t + \frac{1}{\Delta x}\mathbf{B}_n \mathbf{u} = \mathbf{0}, \tag{2.11}$$

where

$$\mathbf{B}_n = \begin{bmatrix} a_{3/2} & 0 & & \cdots & -a_{n+1/2} \\ -a_{3/2} & a_{5/2} & & & \\ & -a_{5/2} & a_{7/2} & & \\ & & \ddots & \ddots & \\ & & & -a_{n-1/2} & a_{n+1/2} \end{bmatrix}. \tag{2.12}$$

As in the constant coefficient case, (2.11) is discretized in time with implicit Euler and time step Δt, which leads to the fully discretized form

$$\begin{aligned} \mathbf{u}^{k+1} &= \mathbf{u}^k + \Delta t \mathbf{u}_t^{k+1} \\ &= \mathbf{u}^k - \frac{\Delta t}{\Delta x}\mathbf{B}_n \mathbf{u}^{k+1} \Leftrightarrow \\ \mathbf{A}_n \mathbf{u}^{k+1} &= \mathbf{u}^k, \end{aligned} \tag{2.13}$$

where

$$\mathbf{A}_n = \mathbf{I}_n + \frac{\Delta t}{\Delta x}\mathbf{B}_n. \tag{2.14}$$

Note that for a constant a, the matrix $\mathbf{B}_n$ is simply $a\tilde{\mathbf{B}}_n$, which results in $\mathbf{A}_n = \tilde{\mathbf{A}}_n$ as above.

Chapter 3

Multigrid methods

To solve the linear system (2.13) for the next step $\mathbf{u}^{k+1}$, one can use *multigrid methods* [6, 7]. The aim of this section is to explain what multigrid methods are and why they are used.

3.1 Iterative methods

At each time step, a linear system of the form

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{3.1}$$

has to be solved for $\mathbf{u}$ where $\mathbf{u} = \mathbf{u}^{k+1}$, and $\mathbf{b} = \mathbf{u}^k$ is known from the previous time step. The subscript of $\mathbf{A} = \mathbf{A}_n$ describing the size of the matrix has been removed to simplify the derivations. Here the focus will be on solving this system using iterative methods. Let $\mathbf{u}$ denote the exact solution of (3.1) and let $\mathbf{x}$ denote the current approximation of $\mathbf{u}$. The error is then defined as

$$\mathbf{e} = \mathbf{u} - \mathbf{x}. \tag{3.2}$$

For the constant coefficient case (2.9), the eigenvectors of $\mathbf{A} = \tilde{\mathbf{A}}$ are given by the *Fourier modes* $\tilde{\mathbf{v}}_j = (\tilde{v}_{j,1}, ..., \tilde{v}_{j,n})^T$ with $\tilde{v}_{j,l} = e^{\mathrm{i}j\frac{2\pi l}{n}}$. The error can be described in terms of these Fourier modes

$$\mathbf{e} = \sum_{j=-n/2+1}^{n/2} c_j \tilde{\mathbf{v}}_j, \tag{3.3}$$

and it can be shown that, when $\mathbf{A}$ represents the discretization of a differential equation, iterating toward $\mathbf{u}$ using a simple method such as the Jacobi method or the Gauss-Seidel method, the error components with high frequencies ($c_j\tilde{\mathbf{v}}_j$ where $|j|$ is high) are damped considerably faster than components with low frequencies ($c_j\tilde{\mathbf{v}}_j$ where $|j|$ is low) [6, 7].

Since $|j| \in [0, n/2]$, let *high frequency components* of the error be defined as the components $c_j \tilde{\mathbf{v}}_j$ with $|j| \geq n/4$ and let *low frequency components* of the error be defined as the components with $|j| < n/4$, so that (approximately) half the components are defined as high frequency components and (approximately) half are defined as low frequency components. One way to see that it is reasonable that the high frequency components are damped faster, in both the constant coefficient and the variable coefficient case, is through equations (2.5) and (2.10). High frequency error components corresponds to short distance errors (short wavelengths) and the low frequency error components correspond to long distance errors (long wavelengths). At each step of the iteration, the time derivative $(u_i)_t$ of a cell is only affected by the values of the neighbouring cells. Thus, it takes fewer iterations for the information to travel between cells with just a few cells between them and correct the local errors, than between cells with a large number of cells between them and correct the global error.

Since the high frequency components of the errors are eliminated faster than low frequency components, this has the effect of smoothing out the error. These iterative solvers are therefore also referred to as *smoothers*.

3.2 Coarse grid correction

The main idea behind multigrid algorithms is to speed up the convergence of the low frequency error components by the use of coarser grids, i.e. grids with less nodes, where the information travels farther with each step of the iteration. Some of the low frequency components of the error on the fine grid become high frequency components on the coarse grid.

Multiplying (3.2) with $\mathbf{A}$ from the left results in the *residual equation*

$$\begin{aligned} &\mathbf{Ae} = \mathbf{Au} - \mathbf{Ax} = \mathbf{b} - \mathbf{Ax} \Leftrightarrow \\ &\mathbf{Ae} = \mathbf{r}, \end{aligned} \tag{3.4}$$

showing that the error satisfies the same equation as $\mathbf{u}$ but with the residual $\mathbf{r} := \mathbf{b} - \mathbf{Ax}$ on the right hand side instead of $\mathbf{b}$. If the error was known exactly, then the solution of $\mathbf{Au} = \mathbf{b}$ could be determined exactly since $\mathbf{u} = \mathbf{x} + \mathbf{e}$. One method for improving the approximation $\mathbf{x}$ is thus to get a good approximation $\bar{\mathbf{e}}$ of the error $\mathbf{e}$ through the residual equation, and then updating the approximation $\mathbf{x}$ by

$$\mathbf{x} \leftarrow \mathbf{x} + \bar{\mathbf{e}}. \tag{3.5}$$

To speed up the convergence of the low frequency error components, instead of iterating on the fine grid, the residual is transferred to the coarse grid (explained in the next subsection) where the approximation of the error is computed. The error is then transferred back to the fine grid and $\mathbf{x}$ is updated using (3.5).

3.2.1 Restriction and prolongation

Let $h = \Delta x$ denote the cell width for the original grid and create a coarser grid by removing every other grid point from the fine grid so that the coarse grid

has half as many grid cells and the cell width $2h$. To distinguish between the two grids, let the superscript of a matrix or vector denote the cell width of the grid on which it is evaluated. The vectors are transferred between the grids using the $\frac{n}{2} \times n$ restriction matrix $\mathbf{R}$ (from fine to coarse grid) and the $n \times \frac{n}{2}$ prolongation matrix $\mathbf{P}$ (from coarse to fine grid) defined by

$$\mathbf{R} = \frac{1}{2} \begin{bmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & \ddots & \ddots & & \\ & & & & & & 1 & 1 \end{bmatrix}, \tag{3.6}$$

and

$$\mathbf{P} = 2\mathbf{R}^T = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{bmatrix}. \tag{3.7}$$

The vector $\mathbf{x}^h$ is thus transferred from the fine grid to the coarse grid by

$$\mathbf{x}^{2h} = \mathbf{R}\mathbf{x}^h,$$

which corresponds to joining two adjacent cells by taking their average, and the vector $\mathbf{x}^{2h}$ is transferred from the coarse grid to the fine grid by

$$\mathbf{x}^h = \mathbf{P}\mathbf{x}^{2h},$$

which corresponds to splitting one cell into two by duplicating the value of the cell. These transfer operations conserves the integral of the corresponding step functions over the spatial domain.

3.3 Two-grid algorithm

A multigrid method with two grids or *levels*, one fine grid (the top level) and one coarse grid (bottom level), could be implemented as follows:

---

1. Choose an initial guess $\mathbf{x}^h$.
2. **Presmoothing:** Perform a number of smoothing steps on $\mathbf{x}^h$.
3. Calculate the residual $\mathbf{r}^h = \mathbf{b}^h - \mathbf{A}^h\mathbf{x}^h$.
4. **Coarse grid correction**:

   (a) Restrict the residual to the coarse grid: $\mathbf{r}^h \rightarrow \mathbf{r}^{2h} = \mathbf{R}\mathbf{r}^h$.
   (b) Restrict the matrix $\mathbf{A}^h$ to the coarse grid: $\mathbf{A}^h \rightarrow \mathbf{A}^{2h}$.
   (c) Solve for the error $\mathbf{e}^{2h}$ on the coarse grid through the residual equation $\mathbf{A}^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$.
   (d) Prolong the error back to the fine grid: $\mathbf{e}^{2h} \rightarrow \mathbf{e}^h = \mathbf{P}\mathbf{e}^{2h}$.
   (e) Update $\mathbf{x}^h$ through $\mathbf{x}^h \leftarrow \mathbf{x}^h + \mathbf{e}^h$
5. **Postsmoothing:** Perform a number of smoothing steps on $\mathbf{x}^h$.

Here, the matrix $\mathbf{A}^{2h}$ is defined as the advection equation discretized on the coarse grid with $\Delta x = 2h$.

3.4 V-cycle

To approximate the error $\mathbf{e}^{2h}$ in step 4c one can use the same iterative method used for the smoothing in the presmoothing and postsmoothing steps. A good initial guess for the error is $\bar{\mathbf{e}}^{2h,(0)} = \mathbf{0}^{2h}$, since performing one smoothing step on $\mathbf{x}$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$ with initial guess $\mathbf{x}_l$ is equivalent to performing one smoothing step on the error $\mathbf{e}_l$ in $\mathbf{A}\mathbf{e}_l = \mathbf{r}_l$ with the specific initial guess $\bar{\mathbf{e}}_l^{(0)} = \mathbf{0}$ and updating $\mathbf{x}_l$ using (3.5) [6].

To show this for a linear method where $\mathbf{A} = \mathbf{N} - \mathbf{N}\mathbf{M}$ and the next step $\mathbf{x}^{l+1}$ is given by

$$\mathbf{x}^{l+1} = \mathbf{M}\mathbf{x}^l + \mathbf{N}^{-1}\mathbf{b} \tag{3.8}$$

(which is the case for methods such as the Jacobi method and the Gauss-Seidel method), let $\bar{\mathbf{e}}_l^{(1)}$ denote the approximation of the error $\mathbf{e}$ after one smoothing step on $\bar{\mathbf{e}}_l^{(0)} = \mathbf{0}$, i.e. $\bar{\mathbf{e}}_l^{(1)} = \mathbf{M}\bar{\mathbf{e}}_l^{(0)} + \mathbf{N}^{-1}\mathbf{r}^l = \mathbf{N}^{-1}\mathbf{r}^l$. Then the next step $\mathbf{x}^{l+1}$ when instead performing the smoothing step on the error and updating according to (3.5) is

$$\begin{aligned} \mathbf{x}^{l+1} &= \mathbf{x}^l + \bar{\mathbf{e}}_l^{(1)} \\ &= \mathbf{x}^l + \mathbf{N}^{-1}\mathbf{r}^l \\ &= \mathbf{x}^l + \mathbf{N}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^l) \\ &= \mathbf{x}^l - \mathbf{N}^{-1}\mathbf{A}\mathbf{x}^l + \mathbf{N}^{-1}\mathbf{b} \\ &= (\mathbf{I} - \mathbf{N}^{-1}\mathbf{A})\mathbf{x}^l + \mathbf{N}^{-1}\mathbf{b} \\ &= \mathbf{M}\mathbf{x}^l + \mathbf{N}^{-1}\mathbf{b} \end{aligned}, \tag{3.9}$$

which is the same as (3.8), proving that the two methods of updating $\mathbf{x}^l$ are equivalent.

Using recursion, it is possible to speed up the convergence of the low frequencies of the error on the coarse grid by moving to even coarser grids with mesh widths $4h, 8h, ...$ and so on until the grid is coarse enough, i.e. $\mathbf{A}$ is small enough that $\mathbf{A}\mathbf{e} = \mathbf{r}$ can be solved quickly with a direct method. This results in the following multigrid algorithm called a *V-cycle*.

Algorithm 1: v_cycle($\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$, ν_1, ν_2, *level*)

Result: Improved approximation of $\mathbf{x}$.
if *level* > 0 **then**
  Presmoothing: $\mathbf{x}$ = smoother($\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$), ν_1 times
  Calculate residual: $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
  Restrict residual: $\mathbf{r}_{rest}$ = restrict($\mathbf{r}$)
  Restrict matrix: $\mathbf{A}_{rest}$ = restrict_matrix($\mathbf{A}$)
  Initiate $\mathbf{e}$ on the coarse grid: $\mathbf{e} = \mathbf{0}$
  Compute approximation of $\mathbf{e}$ recursively:
    $\mathbf{e}$ = v_cycle($\mathbf{e}$, $\mathbf{A}_{rest}$, $\mathbf{r}_{rest}$, ν_1, ν_2, $level - 1$)
  Prolong and update: $\mathbf{x} = \mathbf{x}$ + prolong($\mathbf{e}$)
  Postsmoothing: $\mathbf{x}$ = smoother($\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$), ν_2 times
else
  Solve directly for $\mathbf{x}$ in $\mathbf{A}\mathbf{x} = \mathbf{b}$

Again, the matrices $\mathbf{A}^{4h}, \mathbf{A}^{8h}, ...$ on the coarser grids are created by discretizing the problem on those grids. For effective use of algorithm 1 it is assumed that the coarsest grid has few enough grid points, i.e. that $\mathbf{A}$ is small enough, that $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be solved quickly with a direct method. If this is not the case, the bottom level of the V-cycle could instead consist of a number of smoothing steps.

3.5 Storage and computational cost

The amount of storage needed on the finest grid scales linearly with the grid size. Assume that the finest grid requires N bytes of storage. Each subsequent grid requires half as much storage as the previous one, so if M grids are used, the total storage cost is given by

$$N\left(1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^{M-1}}\right). \tag{3.10}$$

This is a geometric sum with the upper bound

$$\frac{N}{1 - \frac{1}{2}} = 2N. \tag{3.11}$$

The computational cost of the multigrid algorithm is commonly measured in *work units* (WU) [6, 7]. 1 WU is defined as the cost of performing one smoothing step on the finest grid. The cost of intergrid transfer operations typically amounts to 10-20% of the cost of an entire cycle, and is usually neglected. If ν_1 is the number of presmoothing steps and ν_2 is the number of postsmoothing steps, then the computational cost of the finest grid is $(\nu_1 + \nu_2)$WU. Each subsequent grid requires half as much computational work as the previous, resulting in a geometric sum with the upper bound

$$\frac{(\nu_1 + \nu_2)\text{WU}}{1 - \frac{1}{2}} = 2(\nu_1 + \nu_2)\text{WU} \tag{3.12}$$

Thus, the cost of storage and computation for a V-cycle is less than twice the cost of storage and computation of the finest grid alone regardless of the number of grids used. These computations were made for the one dimensional case (since the model problem used here is the one dimensional linear advection equation). For more dimensions, the bounds are even better [6]. For example, the factor 2 in (3.11) and (3.12) become 4/3 for two dimensions and 8/7 for three dimensions.

Chapter 4

Explicit Runge-Kutta smoothers

Now, except for the number of smoothing steps and the number of grids, the only thing left to define is what smoother to use, which is the main subject of this thesis. One class of smoothers studied in this thesis is a class of *explicit Runge-Kutta methods* (*RK methods*) following [2, 4].

4.1 Pseudo time stepping

Consider the initial value problem given by

$$\begin{aligned} \mathbf{x}_{t^*} &= \mathbf{f}(\mathbf{x}), \\ \mathbf{x}_0 &= \mathbf{x}(t_0^*). \end{aligned} \tag{4.1}$$

The time t^* is written here with a star as superscript to show that this is a pseudo time introduced only for iterating towards the solution of $\mathbf{Au} = \mathbf{b}$. To use this to find the solution of $\mathbf{Au} = \mathbf{b}$, let $\mathbf{f}(\mathbf{x}) = \mathbf{b} - \mathbf{Ax}$. If $\mathbf{x}_{t^*} = \mathbf{f}(\mathbf{x}) \to \mathbf{0}$ as $t^* \to \infty$, then $\mathbf{Ax} \to \mathbf{b}$ as $t^* \to \infty$. This is the case if the real part of the eigenvalues of $\mathbf{A}$ are positive [8], which they are for the matrices here. This can be proven using the Gershgorin Circle Theorem [9], which states that each eigenvalue of a complex square matrix $\mathbf{M}$ with elements m_{ij} are located in at least one of the *Gershgorin disks*

$$D_i := \{z : |z - m_{ii}| \leq \sum_{j \neq i} |m_{ij}|\}. \tag{4.2}$$

Lemma 1. The real part of the eigenvalues of the matrices $\mathbf{A} = \mathbf{A}_n$ defined in (2.14) are positive.

Proof. First of all, with $\mathbf{B} := \mathbf{B}_n$ from (2.12), Let $\lambda_{B,j}$ be an eigenvalue of $\mathbf{B}$

and let $\mathbf{v}_j$ be the corresponding eigenvector. Then

$$\begin{aligned} \mathbf{A} &= \mathbf{I} + \frac{\Delta t}{\Delta x}\mathbf{B} \Leftrightarrow \\ \mathbf{A}\mathbf{v}_j &= \left(\mathbf{I} + \frac{\Delta t}{\Delta x}\mathbf{B}\right)\mathbf{v}_j \\ &= \left(1 + \frac{\Delta t}{\Delta x}\lambda_{B,j}\right)\mathbf{v}_j, \end{aligned} \tag{4.3}$$

and thus, the eigenvalues $\lambda_{A,j}$ of $\mathbf{A}$ are given by

$$\lambda_{A,j} = 1 + \frac{\Delta t}{\Delta x}\lambda_{B,j} \qquad j = 1, 2, ..., n. \tag{4.4}$$

Now, by looking at the Gershgorin disks of $\mathbf{B}^T$ (with $\mathbf{B} := \mathbf{B}_n$ from (2.12)) one can deduce that the real part of the eigenvalues of $\mathbf{B}^T$, and therefore also of $\mathbf{B}$ (since $\mathbf{B}$ and $\mathbf{B}^T$ have the same characteristic polynomial

$$\det(\mathbf{B}^T - \lambda\mathbf{I}) = \det(\mathbf{B}^T - \lambda\mathbf{I}^T) = \det([\mathbf{B} - \lambda\mathbf{I}]^T) = \det(\mathbf{B} - \lambda\mathbf{I}), \tag{4.5}$$

they have the same set of eigenvalues), are non-negative: since the radii of the disks $|-a_{i+1/2}| = a_{i+1/2}$ have the same value as the the centers of the disks on the real axis, all disks are located in the right complex half-plane and intersect the imaginary axis at the origin. In particular, all eigenvalues of $\mathbf{B}$ are located on or inside the circle with radius $a_{max} := \max_x a(x) > 0$ and centered at a_{max} since all other Gershgorin disks are inside this circle (Figure 4.1). Therefore, since $\Delta t/\Delta x > 0$, it follows from (4.4) that the real part of the eigenvalues of $\mathbf{A}$ are positive. □

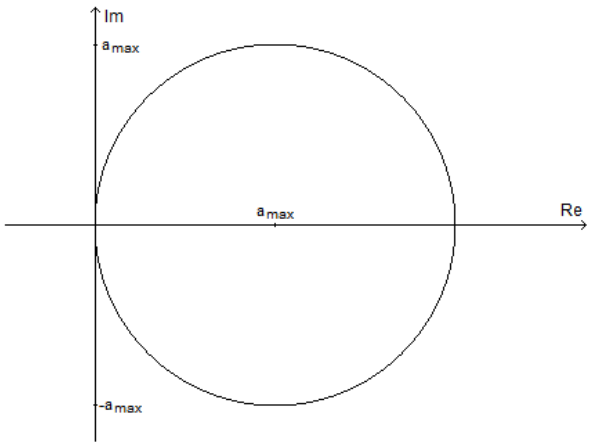


Figure 4.1: The circle of radius $a_{max} := \max_x a(x)$ centered at a_{max}.

The solution of (4.1) can be approximated with the explicit s-stage RK scheme given by

$$\begin{aligned}\mathbf{x}_l^{(0)} &= \mathbf{x}_l \\ \mathbf{x}_l^{(j)} &= \mathbf{x}_l + \alpha_j \Delta t^* \mathbf{f}(\mathbf{x}_l^{(j-1)}), \quad j = 1, ..., s-1 \\ \mathbf{x}_{l+1} &= \mathbf{x}_l + \Delta t^* \mathbf{f}(\mathbf{x}_l^{(s-1)})\end{aligned} \tag{4.6}$$

where $\Delta t^* \in \mathbb{R}$ and $\alpha_j \in [0, 1] \; \forall \; j$.

One iteration of the RK smoother consists of taking one step in pseudo time t^* of size Δt^* using the RK scheme (4.6). Let $\alpha_s = \alpha_{s+1} = 1$, and let the *stability polynomial*

$$P_s(z) = \sum_{r=0}^{s} \left(\prod_{m=s-r+1}^{s+1} \alpha_m \right) z^r, \tag{4.7}$$

and the polynomial

$$S_s(z) = \sum_{r=0}^{s-1} \left(\prod_{m=s-r}^{s} \alpha_m \right) z^r, \tag{4.8}$$

be defined. Then (4.6) can be described by

$$\mathbf{x}_{l+1} = P_s(-\Delta t^* \mathbf{A}) \mathbf{x}_l + S_s(-\Delta t^* \mathbf{A}) \Delta t^* \mathbf{b}, \tag{4.9}$$

which can be seen by working backwards:

$$\begin{aligned}\mathbf{x}_{l+1} &= \mathbf{x}_l + \Delta t^* (\mathbf{b} - \mathbf{A}\mathbf{x}_l^{(s-1)}) \\ &= \mathbf{x}_l + \Delta t^* (\mathbf{b} - \mathbf{A}(\mathbf{x}_l + \alpha_{s-1} \Delta t^* (\mathbf{b} - \mathbf{A}\mathbf{x}_l^{(s-2)}))) \\ &= ... \\ &= \left[\mathbf{x}_l + (-\Delta t^* \mathbf{A}) \mathbf{x}_l + \alpha_{s-1} (-\Delta t^* \mathbf{A})^2 \mathbf{x}_l + ... + \left(\prod_{m=1}^{s-1} \alpha_m \right) (-\Delta t^* \mathbf{A})^s \mathbf{x}_l \right] \\ &\quad + \left[\Delta t^* \mathbf{b} + \alpha_{s-1} (-\Delta t^* \mathbf{A}) \Delta t^* \mathbf{b} + ... + \left(\prod_{m=1}^{s-1} \alpha_m \right) (-\Delta t^* \mathbf{A})^{s-1} \Delta t^* \mathbf{b} \right] \\ &= \left[\mathbf{I} + (-\Delta t^* \mathbf{A}) + \alpha_{s-1} (-\Delta t^* \mathbf{A})^2 + ... + \left(\prod_{m=1}^{s-1} \alpha_m \right) (-\Delta t^* \mathbf{A})^s \right] \mathbf{x}_l \\ &\quad + \left[\mathbf{I} + \alpha_{s-1} (-\Delta t^* \mathbf{A}) + ... + \left(\prod_{m=1}^{s-1} \alpha_m \right) (-\Delta t^* \mathbf{A})^{s-1} \right] \Delta t^* \mathbf{b} \\ &= P_s(-\Delta t^* \mathbf{A}) \mathbf{x}_l + S_s(-\Delta t^* \mathbf{A}) \Delta t^* \mathbf{b}.\end{aligned} \tag{4.10}$$

From this it is possible to deduce that the RK smoothers satisfy (3.8) with $\mathbf{M} = P_s(-\Delta t^* \mathbf{A})$ and $\mathbf{N}^{-1} = S_s(-\Delta t^* \mathbf{A}) \Delta t^*$. To show this, $\mathbf{A} = \mathbf{N} - \mathbf{N}\mathbf{M}$ has to be proven. First, note that the relation

$$\mathbf{I} - P_s(-\Delta t^* \mathbf{A}) = \Delta t^* S_s(-\Delta t^* \mathbf{A}) \mathbf{A} \tag{4.11}$$

holds. This relation is shown as follows.

$$\begin{aligned}
P_s(-\Delta t^*\mathbf{A}) &= \sum_{r=0}^{s}\left(\prod_{m=s-r+1}^{s+1}\alpha_m\right)(-\Delta t^*\mathbf{A})^r \\
&= a_{s+1}(-\Delta t^*\mathbf{A})^0 + \left[\sum_{r=1}^{s}\left(\prod_{m=s-r+1}^{s+1}\alpha_m\right)(-\Delta t^*\mathbf{A})^{r-1}\right](-\Delta t^*\mathbf{A}) \\
&= \mathbf{I} - \Delta t^*\left[\sum_{r=1}^{s}\left(\prod_{m=s-r+1}^{s+1}\alpha_m\right)(-\Delta t^*\mathbf{A})^{r-1}\right]\mathbf{A} \\
&= \mathbf{I} - \Delta t^*\left[\sum_{r=0}^{s-1}\left(\prod_{m=s-r}^{s}\alpha_m\right)(-\Delta t^*\mathbf{A})^{r}\right]\mathbf{A} \\
&= \mathbf{I} - \Delta t^* S_s(-\Delta t^*\mathbf{A})\mathbf{A} \Leftrightarrow \\
\mathbf{I} - P_s(-\Delta t^*\mathbf{A}) &= \Delta t^* S_s(-\Delta t^*\mathbf{A})\mathbf{A}.
\end{aligned} \tag{4.12}$$

From this follows that

$$\begin{aligned}
\mathbf{N} - \mathbf{NM} &= \mathbf{N}(\mathbf{I} - \mathbf{M}) \\
&= \frac{1}{\Delta t^*} S_s(-\Delta t^*\mathbf{A})^{-1}(\mathbf{I} - P_s(-\Delta t^*\mathbf{A})) \\
&= \frac{1}{\Delta t^*} S_s(-\Delta t^*\mathbf{A})^{-1}\Delta t^* S_s(-\Delta t^*\mathbf{A})\mathbf{A} \\
&= \mathbf{A},
\end{aligned} \tag{4.13}$$

where (4.11) was used in the third equality.

4.2 Evolution of the error

Let $\mathbf{u}$ be the solution to $\mathbf{Au} = \mathbf{b}$ and let $\mathbf{x}_l$ be the current approximation of $\mathbf{u}$ as above. Also, let the error $\mathbf{e}_l$ be defined as in (3.2), i.e. $\mathbf{e}_l = \mathbf{u} - \mathbf{x}_l$. Since $\mathbf{x}_{l+1} = \mathbf{Mx}_l + \mathbf{N}^{-1}\mathbf{b}$ and $\mathbf{A} = \mathbf{N} - \mathbf{NM} \Leftrightarrow \mathbf{N}^{-1} = \mathbf{A}^{-1} - \mathbf{MA}^{-1}$, taking one step with the RK smoother results in

$$\begin{aligned}
\mathbf{e}_{l+1} &= \mathbf{u} - \mathbf{x}_{l+1} \\
&= \mathbf{u} - (\mathbf{Mx}_l + \mathbf{N}^{-1}\mathbf{b}) \\
&= \mathbf{u} - \mathbf{Mx}_l - \mathbf{N}^{-1}\mathbf{b} \\
&= \mathbf{u} - \mathbf{M}(\mathbf{u} - \mathbf{e}_l) - (\mathbf{A}^{-1} - \mathbf{MA}^{-1})\mathbf{b} \\
&= \mathbf{u} - \mathbf{Mu} + \mathbf{Me}_l - (\mathbf{u} - \mathbf{Mu}) \\
&= \mathbf{Me}_l.
\end{aligned} \tag{4.14}$$

4.3 Optimizing M

Let $\mathbf{v}_j$ denote the eigenvectors of $\mathbf{M}$ and let λ_j denote the corresponding eigenvalues. If the error at step k is described in terms of the eigenvectors

$$\mathbf{e}_k = \sum_{j=-n/2+1}^{n/2} c_{j,k}\mathbf{v}_j \tag{4.15}$$

then the error after m iterations is

$$\begin{aligned} \mathbf{e}_{k+m} &= \mathbf{M}^m \mathbf{e}_k \\ &= \sum_{j=-n/2+1}^{n/2} c_{j,k}\lambda_j^m \mathbf{v}_j. \end{aligned} \tag{4.16}$$

Here the eigenvectors of $\mathbf{A}$ created from the variable coefficient case are assumed to approximately behave as the eigenvectors of $\tilde{\mathbf{A}}$ in the constant coefficient case, so the components $c_{j,k}\mathbf{v}_j$ will be referred to as high frequency components if $|j| \geq n/4$ and low frequency components if $|j| < n/4$. To optimize the overall convergence of the smoother, one would need to minimize $\max_j |\lambda_j|$ with respect to the parameters of the RK method, i.e. the coefficients $\alpha_i,\ i = 1, ..., s-1$, and the pseudo time step Δt^*. However, since the low frequency components are handled by moving to a coarser grid in the multigrid method, the smoother is instead optimized for the convergence of the high frequency error components, i.e. to minimize the *smoothing factor* $\max_{|j|\geq n/4} |\lambda_j|$, and the optimization problem becomes

$$\min_{\Delta t^*, \alpha_1, ..., \alpha_{s-1}} \left[\max_{|j|\geq n/4} |\lambda_j| \right] \tag{4.17}$$

with $\Delta t^* \in \mathbb{R}$ and $\alpha_j \in [0, 1]\ \forall\ j$. Thus, to optimize the explicit RK smoother, the eigenvalues of $\mathbf{M}$ are needed.

4.4 Eigenvalues of M

Let $\mathbf{M}_n$ denote the matrix $\mathbf{M}$ of size $n \times n$, i.e. $\mathbf{M}_n = P_s(-\Delta t^* \mathbf{A}_n)$. Since the eigenvectors of $\mathbf{M}_n$ are the same as the eigenvectors for $\mathbf{A}_n$, and since $\mathbf{A}_n = \mathbf{I}_n + \frac{\Delta t}{\Delta x}\mathbf{B}_n$ is constructed using the matrix $\mathbf{B}_n$, the eigenvalues of $\mathbf{B}_n$ are computed first.

4.4.1 Constant coefficient

In the constant coefficient case, the eigenvalues of the matrix $\tilde{\mathbf{B}}_n$ defined in (2.7) can be determined analytically. Let $\mathbf{D}_{1,n}$ be the matrix created by removing

the first row and the last column from $\tilde{\mathbf{B}}_n - \tilde{\lambda}_B \mathbf{I}_n$, i.e.

$$\mathbf{D}_{1,n} = \begin{bmatrix} -1 & 1-\lambda & & & \\ & -1 & 1-\lambda & & \\ & & \ddots & \ddots & \\ & & & & -1 \end{bmatrix} \tag{4.18}$$

Then the determinant of this matrix is

$$\det(\mathbf{D}_{1,n}) = (-1)^{n-1}. \tag{4.19}$$

Thus, by expanding $\det(\tilde{\mathbf{B}}_n - \tilde{\lambda}_B \mathbf{I}_n)$ along the first row:

$$\begin{aligned} \det(\tilde{\mathbf{B}}_n - \tilde{\lambda}_B \mathbf{I}_n) = 0 \Leftrightarrow \\ (1-\tilde{\lambda}_B)^n + (-1)^{n+1}(-1)\det(\mathbf{D}_{1,n}) = 0 \Leftrightarrow \\ (1-\tilde{\lambda}_B)^n - 1 = 0 \Leftrightarrow \\ \tilde{\lambda}_{B,j} = 1 - \exp\left(-\mathrm{i}\frac{2\pi j}{n}\right), \quad j \in \mathbb{Z} : j \in (-\frac{n}{2}, \frac{n}{2}], \end{aligned} \tag{4.20}$$

where i is the imaginary unit. In this text, to distinguish the imaginary unit from the index, the imaginary unit i will always be non-italic.

Let $\mathbf{v}_j$ be the jth eigenvector of $\tilde{\mathbf{B}}_n$. Then

$$\begin{aligned} \tilde{\mathbf{A}}_n \mathbf{v}_j &= \left(\mathbf{I}_n + \frac{a\Delta t}{\Delta x}\tilde{\mathbf{B}}_n\right)\mathbf{v}_j \\ &= \left(1 + \frac{a\Delta t}{\Delta x}\tilde{\lambda}_{B,j}\right)\mathbf{v}_j, \end{aligned} \tag{4.21}$$

and from this follows that the eigenvalues of $\tilde{\mathbf{A}}_n$ are given by

$$\begin{aligned} \tilde{\lambda}_{A,j} &= 1 + \frac{a\Delta t}{\Delta x}\tilde{\lambda}_{B,j} \\ &= 1 + \frac{a\Delta t}{\Delta x}\left[1 - \exp\left(-\mathrm{i}\frac{2\pi j}{n}\right)\right], \quad j \in \mathbb{Z} : j \in (-\frac{n}{2}, \frac{n}{2}]. \end{aligned} \tag{4.22}$$

Finally, since

$$\mathbf{M}_n \mathbf{v}_j = P_s(-\Delta t^* \tilde{\mathbf{A}}_n)\mathbf{v}_j = P_s(-\Delta t^* \tilde{\lambda}_{A,j})\mathbf{v}_j, \tag{4.23}$$

the eigenvalues of $\mathbf{M}_n$ are given by

$$\begin{aligned} \tilde{\lambda}_{P,j} &= P_s(-\Delta t^* \tilde{\lambda}_{A,j}) \\ &= P_s\left(-\Delta t^*\left[1 + \frac{a\Delta t}{\Delta x}\left[1 - \exp\left(-\mathrm{i}\frac{2\pi j}{n}\right)\right]\right]\right), \quad j \in \mathbb{Z} : j \in (-\frac{n}{2}, \frac{n}{2}]. \end{aligned} \tag{4.24}$$

4.4.2 Variable coefficient

In the variable coefficient case, it is not as easy to determine the eigenvalues. By following the same process above for the eigenvalues of $\mathbf{B}_n$ defined in (2.12), one would end up at the following expression:

$$\det(\mathbf{B}_n - \lambda_B \mathbf{I}_n) = \prod_{i=1}^{n}(a_{i+1/2} - \lambda_B) - \prod_{i=1}^{n} a_{i+1/2} = 0, \tag{4.25}$$

for which there is no general solution in radicals for $n \geq 5$ (Abel's impossibility theorem). While it is possible to approximate the eigenvalues using iterative methods, it becomes computationally expensive as the size of the matrix gets large.

In the next chapter it will be shown that it is possible to get an explicit expression that approximates how the eigenvalues of the matrices $\mathbf{B}_n$ and $\mathbf{A}_n$ from (2.14) are distributed, assuming that n is large.

Chapter 5

Generalized Locally Toeplitz Sequences

The theory of *generalized locally Toeplitz (GLT) sequences* [3] can be used to get information about the *distribution of singular values*, and given some additional assumptions, also the *distribution of eigenvalues*, of large matrices.

In this text, a *matrix sequence* is a sequence of the form $\{A_n\}_n$, where $A_n \in \mathbb{C}^{n\times n}$, and $n \in \mathbb{N}$. All definitions in this chapter are taken from [3].

Definition 2. Let $\{A_n\}_n$ be a matrix sequence, and let $f : D \subset \mathbb{R}^k \to \mathbb{C}$ be a measurable function defined on a set D with $0 < \mu_k(D) < \infty$. Also let $\sigma_i(A_n)$ denote the ith singular value and $\lambda_i(A_n)$ denote the ith eigenvalue of the matrix A_n. Then

- $\{A_n\}_n$ has a *singular value distribution described by* f, denoted by $\{A_n\}_n \sim_\sigma f$, if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} F(\sigma_i(A_n)) = \frac{1}{\mu_k(D)} \int_D F(|f(\boldsymbol{x})|)d\boldsymbol{x}, \qquad \forall F \in C_c(\mathbb{R}). \quad (5.1)$$

  ($C_c(\mathbb{C})$ (resp., $C_c(\mathbb{R})$) refers to the space of complex-valued continuous functions defined on $\mathbb{C}$ (resp., $\mathbb{R}$) with bounded support.) f is then called the *singular value distribution symbol* of $\{A_n\}_n$.

- $\{A_n\}_n$ has a *spectral (or eigenvalue) distribution described by* f, denoted by $\{A_n\}_n \sim_\lambda f$, if

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} F(\lambda_i(A_n)) = \frac{1}{\mu_k(D)} \int_D F(f(\boldsymbol{x}))d\boldsymbol{x}, \qquad \forall F \in C_c(\mathbb{C}). \quad (5.2)$$

  f is then called the *spectral (or eigenvalue) distribution symbol* of $\{A_n\}_n$.

What this means is that if $\{A_n\}_n$ has a singular value distribution described by f and if n is large enough, then the singular values $\sigma_i(A_n)$ are approximately equal to the samples of $|f|$ over a uniform grid in D, and equivalently for the eigenvalues $\lambda_i(A_n)$ and f, if $\{A_n\}_n$ has an eigenvalue distribution described by f.

The aim of this chapter is to give an understanding of GLT sequences. An explanation of *Toeplitz sequences* is given in section 5.1. This is then generalized into *locally Toeplitz (LT) sequences* in section 5.2, and finally into GLT sequences in section 5.3. Some properties, including asymptotic eigenvalue and singular value distributions of GLT sequences are given in subsection 5.3.1.

In this text, a *matrix sequence* is a sequence of the form $\{A_n\}_n$, where $A_n \in \mathbb{C}^{n\times n}$, and $n \in \mathbb{N}$.

5.1 Toeplitz sequences

Recall the matrix

$$\tilde{\mathbf{B}}_n = \begin{bmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}$$

that arose in the semi-discretized form of the constant coefficient case (2.6). This is a special type of matrix called a *Toeplitz matrix*.

Definition 3. A matrix $\mathbf{A}$ where each diagonal is constant:

$$\mathbf{A} = [a_{j-i}]_{i,j}^n = \begin{bmatrix} a_0 & a_1 & a_2 & \dots & a_{n-1} \\ a_{-1} & a_0 & a_1 & & \\ a_{-2} & a_{-1} & a_0 & \ddots & \\ \vdots & & \ddots & \ddots & a_1 \\ a_{-(n-1)} & & & a_{-1} & a_0 \end{bmatrix} \tag{5.3}$$

is called a *Toeplitz matrix*.

A Toeplitz matrix can be defined by a function f using Fourier analysis. The reason for doing this is that the function f also contains information about the eigenvalues of the Toeplitz matrix, as will be seen in section 5.3.

Definition 4. Let $f : [-\pi, \pi] \to \mathbb{C}$ be a function in $L^1([-\pi, \pi])$, and let $n \in \mathbb{N}$.

The *nth Toeplitz matrix associated with* f, denoted $T_n(f)$, is defined as

$$T_n(f) = [f_{j-i}]_{i,j=1}^{n} = \begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_{n-1} \\ f_{-1} & f_0 & f_1 & & \\ f_{-2} & f_{-1} & f_0 & \ddots & \\ \vdots & & \ddots & \ddots & f_1 \\ f_{-(n-1)} & & & f_{-1} & f_0 \end{bmatrix}, \tag{5.4}$$

where f_k is the kth Fourier coefficient of f:

$$f_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\theta) e^{-\mathrm{i}k\theta} d\theta. \tag{5.5}$$

The matrix sequence $\{T_n(f)\}_n$ is called the *Toeplitz sequence generated by* f.

Given a Toeplitz matrix, the function f can be easily found as a Fourier series with Fourier coefficients given by the elements of the matrix. For example, the matrix $\tilde{\mathbf{B}}_n$ is the nth Toeplitz matrix associated with $f(\theta) = 1 - e^{-\mathrm{i}\theta} - e^{\mathrm{i}(n-1)\theta}$.

5.2 Locally Toeplitz sequences

In the variable coefficient case, the matrix describing the semi-discrete form (2.11) was

$$\mathbf{B}_n = \begin{bmatrix} a_{3/2} & 0 & & \cdots & -a_{n+1/2} \\ -a_{3/2} & a_{5/2} & & & \\ & -a_{5/2} & a_{7/2} & & \\ & & \ddots & \ddots & \\ & & & -a_{n-1/2} & a_{n+1/2} \end{bmatrix}.$$

If $a(x)$ is not constant, then this is not a Toeplitz matrix. However, as $n \to \infty$ it resembles the structure of a toeplitz matrix locally in the sense that small subblocks of $\mathbf{B}_n$ are close to being Toeplitz matrices. To define *Locally Toeplitz sequences* rigorously, a number of other definitions first have to be made.

Definition 5. Let $a : [0,1] \to C$, and let $n \in \mathbb{N}$. The *nth diagonal sampling matrix generated by* a, denoted $D_n(a)$, is defined as

$$D_n(a) = \underset{i=1,\ldots,n}{\operatorname{diag}} a(\frac{i}{n}) = \begin{bmatrix} a(\frac{1}{n}) & & & \\ & a(\frac{2}{n}) & & \\ & & \ddots & \\ & & & a(\frac{1}{n}) \end{bmatrix}. \tag{5.6}$$

Definition 6. Let $m, n \in \mathbb{N}$, $a : [0,1] \to \mathbb{C}$, and $f \in L^1([-\pi,\pi])$. The *locally Toeplitz operators* of a and f are defined as the following $n \times n$ matrices:

$$LT_n^m(a,f) = D(a) \otimes T_{\lfloor \frac{n}{m} \rfloor}(f) \oplus O_{n \bmod m}, \tag{5.7}$$

where the tensor (Kronecker) product "$\otimes$" is applied before the direct sum "$\oplus$", i.e.

$$LT_n^m(a,f) = \begin{bmatrix} a(\frac{1}{m})T_{\lfloor \frac{n}{m} \rfloor}(f) & & & & \\ & a(\frac{2}{m})T_{\lfloor \frac{n}{m} \rfloor}(f) & & & \\ & & \ddots & & \\ & & & a(1)T_{\lfloor \frac{n}{m} \rfloor}(f) & \\ & & & & O_{n \bmod m} \end{bmatrix}. \tag{5.8}$$

Here O_n denotes the $n \times n$ zero matrix, and $\lfloor \cdot \rfloor$ denotes the floor function.

Note that each block $a(\frac{i}{m})T_{\lfloor \frac{n}{m} \rfloor}(f)$ in (5.8) is a Toeplitz matrix, and that every Toeplitz matrix $T_n(f)$ is also a locally Toeplitz operator with $T_n(f) = LT_n^m(1,f)$ for any $m \in \mathbb{N}$.

The matrix $\mathbf{B}_n$ is not a locally Toeplitz operator which can be easily seen by noting that the matrix $\mathbf{B}_n$ is not a blockdiagonal matrix. However, it resembles a locally Toeplitz operator as $n \to \infty$. This resemblance can be defined using the concept of *approximating classes of sequences*.

Definition 7. Let $\{A_n\}_n$ be a matrix sequence and let $\{\{B_{n,m}\}_n\}_m$ be a sequence of matrix sequences. Then $\{\{B_{n,m}\}_n\}_m$ is an *approximating class of sequences (or a.c.s.)* for $\{A_n\}_n$ if the following property holds: $\forall\, m\ \exists\, n_m : n \geq n_m \Rightarrow$

$$A_n = B_{n,m} + R_{n,m} + N_{n,m}, \qquad \operatorname{rank}(R_{n,m}) \leq c(m)n,\ ||N_{n,m}|| \leq \omega(m), \tag{5.9}$$

where the quantities $n_m, c(m)$, and $\omega(m)$ only depend on m, and

$$\lim_{m\to\infty} c(m) = \lim_{m\to\infty} \omega(m) = 0.$$

This is denoted by $\{\{B_{n,m}\}_n\}_m \xrightarrow{\text{a.c.s.}} \{A_n\}_n$.

Thus, if $\{\{B_{n,m}\}_n\}_m$ is an approximating class of sequences for $\{A_n\}_n$ and n and m are large, then the matrix A_n is equal to the matrix $B_{n,m}$ plus two matrices that are close to the zero matrix in two different ways:

- The matrix $R_{n,m}$ has a low rank relative to the size of the matrix.
- The matrix $N_{n,m}$ has a small norm. (Any norm, since all norms are equivalent.)

Given these definitions, *Locally Toeplitz sequences* can now be defined.

Definition 8. Let $\{A_n\}_n$ be a matrix sequence, let $a : [0,1] \to \mathbb{C}$ be Riemann integrable, and let $f \in L^1([-\pi,\pi])$. Then $\{A_n\}_n$ is a *locally Toeplitz (LT) sequence* with symbol $a \otimes f$, denoted by $\{A_n\}_n \sim_{LT} a \otimes f$, if

$$\{LT_n^m(a,f)\}_n \xrightarrow{a.c.s.} \{A_n\}_n.$$

Here $a \otimes f$ is the tensor-product of a and f, i.e. $(a \otimes f)(x,\theta) := a(x)f(\theta)$.

The matrix sequence $\{\mathbf{B}_n\}_n$ created by $\mathbf{B}_n$ is an LT sequence with symbol

$$\kappa_B(x,\theta) = a(x)(1 - \exp(-\mathrm{i}\theta)), \tag{5.10}$$

i.e. $\{\mathbf{B}_n\}_n \sim_{LT} \kappa_B(x,\theta)$ as shown in [4]. (The matrices in the matrix sequence this was shown for had a different first row. However, the difference between the matrices could be added into the matrix $R_{n,m}$ in the definition of an approximating class of sequences.)

5.3 Generalized Locally Toeplitz sequences

The concept of LT sequences can be further generalized by combining different LT sequences.

Definition 9. Let $\{A_n\}_n$ be a matrix sequence, and let $\kappa : [0,1] \times [-\pi,\pi] \to \mathbb{C}$ be a measurable function. Then $\{A_n\}_n$ is a *generalized locally Toeplitz (GLT) sequence* with *symbol* κ, denoted by $\{A_n\}_n \sim_{GLT} \kappa$, if the following property holds: $\forall\, m \in \mathbb{N}$ there exists a finite number of LT sequences $\{A_n^{(i,m)}\}_n \sim_{LT} a_{i,m} \otimes f_{i,m}, i = 1, ..., N_m$ such that

- $\sum_{i=1}^{N_m} a_{i,m} \otimes f_{i,m} \to \kappa$ in measure, and
- $\sum_{i=1}^{N_m} \{A_n^{(i,m)}\}_n \xrightarrow{a.c.s.} \{A_n\}_n$.

Any linear combination of LT sequences is a GLT sequence, since then the above condition holds with N_m constant for all m (note that an LT sequence multiplied by a scalar is an LT sequence). In particular, every LT sequence $\{A_n\}_n \sim_{LT} \kappa = a \otimes f$ is also a GLT sequence.

5.3.1 Properties of GLT sequences

If a sequence of matrices is a GLT sequence and the symbol of the GLT sequence is known, then the *singular value distribution*, and with additional assumptions also the *eigenvalue distribution*, of the matrix sequence is known. Here follow some useful properties of GLT sequences given in [3].

Proposition 10. The following properties hold for GLT sequences:

GLT1 If $\{A_n\}_n \sim_{GLT} \kappa$ then $\{A_n\}_n \sim_\sigma \kappa$. If in addition the matrices A_n are Hermitian, then $\{A_n\}_n \sim_\lambda \kappa$.
GLT2 If $\{A_n\}_n \sim_{GLT} \kappa$ and each matrix A_n can be separated into $A_n = X_n + Y_n$ where

- each X_n is Hermitian,
- $||X_n||, ||Y_n|| \leq C$ for some constant C independent of n, and
- $||Y_n||_1/n \to 0$ as $n \to \infty$,

then $\{A_n\}_n \sim_\lambda \kappa$.
GLT3 The following hold:

- $\{T_n(f)\} \sim_{GLT} \kappa(x,\theta) = f(\theta)$ for $f \in L^1([-\pi,\pi])$,
- $\{D_n(a)\}_n \sim_{GLT} \kappa(x,\theta) = a(x)$ for $a : [0,1] \to \mathbb{C}$ here a is continuous a.e., and
- $\{Z_n\}_n \sim_{GLT} 0$ if and only if $\{Z_n\}_n \sim_\sigma 0$.

GLT4 If $\{A_n\}_n \sim_{GLT} \kappa$ and $\{B_n\}_n \sim_{GLT} \xi$, then

- $\{A_n^*\}_n \sim_{GLT} \bar{\kappa}$
- $\{\alpha A_n + \beta B_n\}_n \sim_{GLT} \alpha\kappa + \beta\xi \quad \forall\, \alpha, \beta \in \mathbb{C}$
- $\{A_n B_n\}_n \sim GLT\kappa\xi$

GLT4 is equivalent to stating that the space of GLT sequences forms a *-algebra of matrix sequences. **GLT3** and **GLT4** can be used to determine the symbol of a matrix sequence, and **GLT1** and **GLT2** are used for getting information about the singular values and eigenvalues of matrix sequences for which the symbol is known. Points 2 and 3 from **GLT4** can be used to give the following useful corollary.

Corollary 11. Let $\{A_n\}_n \sim_{GLT} \kappa$ and let P be a polynomial. Then the sequence $\{P(A_n)\}_n$ satisfies $\{P(A_n)\}_n \sim_{GLT} P(\kappa)$.

Proof. Applying the third point from GLT4 multiple times leads to the formula

$$\{A_n^k\}_n \sim_{GLT} \kappa^k. \tag{5.11}$$

Let the degree of the polynomial P be denoted by s and let $a_k, k = 0, ..., s$ denote the coefficients of the polynomial. Then applying the second point from GLT4 (linearity) in addition to (5.11) leads to

$$\{P(A_n)\}_n = \{\sum_{k=0}^{s} a_k A_n^k\}_n \sim_{GLT} \sum_{k=0}^{s} a_k \kappa^k = P(\kappa). \tag{5.12}$$

□

Thus, if the symbol of the matrix sequence $\{A_n\}_n$ is known, the symbol of $\{P(A_n)\}_n$ is also known.

To optimize a smoother, one would preferably like to know the eigenvalue distribution of the matrix sequence in addition to knowing the singular value distribution (which will be seen to be the case for the matrix sequences in this thesis). However, since the eigenvalues are bounded by the largest singular value

$$|\lambda_j| \leq \max_i \sigma_i \quad \forall\, j \in [1, n], \tag{5.13}$$

it is still possible to get useful information about the eigenvalues even if only the singular value distribution is known.

5.4 Symbols of $\{\mathbf{A}_n\}_n$ and $\{\mathbf{B}_n\}_n$

As stated above, the matrix sequence $\{\mathbf{B}_n\}_n$ with $\mathbf{B}_n$ defined as in (2.12) is an LT sequence with symbol

$$\kappa_B(x, \theta) = a(x)(1 - \exp(-\mathrm{i}\theta)). \tag{5.14}$$

Since all LT sequences are also GLT sequences, this is also a GLT sequence. From this follows that $\{\mathbf{B}_n\}_n \sim_\sigma \kappa_B(x, \theta)$ by GLT1 in proposition 10. Since these matrices are not Hermitian, the sequence does not satisfy the second part of GLT1. However, it does satisfy GLT2 by seperating each $\mathbf{B}_n$ into $\mathbf{B}_n = \mathbf{X}_n + \mathbf{Y}_n$ where

$$\mathbf{X}_n := \begin{bmatrix} a_{3/2} & & & & \\ & a_{5/2} & & & \\ & & a_{7/2} & & \\ & & & \ddots & \\ & & & & a_{n+1/2} \end{bmatrix} \tag{5.15}$$

is Hermitian since $a(x) \in \mathbb{R}$, and

$$Y_n := \begin{bmatrix} 0 & & & & & -a_{n+1/2} \\ -a_{3/2} & 0 & & & & \\ & -a_{5/2} & 0 & & & \\ & & \ddots & \ddots & & \\ & & & -a_{n-1/2} & 0 & \end{bmatrix}. \tag{5.16}$$

Since all norms are equivalent, it is enough that $||\mathbf{X}_n||$ and $||\mathbf{Y}_n||$ are bounded with respect to the 1-norm, which they are with the bound $C = \max_x a(x)$. Since $||\mathbf{Y}_n||_1$ is bounded by C it also follows that $||\mathbf{Y}_n||_1/n \xrightarrow{n\to\infty} 0$. Thus

$$\{\mathbf{B}_n\}_n \sim_\lambda \kappa_B(x,\theta) \tag{5.17}$$

by GLT2.

By GLT3 the symbol of the identity matrix $\mathbf{I}_n$ is simply 1. If $\Delta t/\Delta x$ is assumed to be constant, then by GLT4 the matrix sequence $\{\mathbf{A}_n\}_n$ defined in (2.14), i.e.

$$\{\mathbf{A}_n\}_n = \left\{\mathbf{I}_n + \frac{\Delta t}{\Delta x}\mathbf{B}_n\right\}_n,$$

is a GLT sequence with the symbol

$$\begin{aligned} \kappa_A(x,\theta) &= 1 + \frac{\Delta t}{\Delta x}\kappa_B(x,\theta) \\ &= 1 + \frac{\Delta t}{\Delta x}a(x)(1-\exp(-\mathrm{i}\theta)). \end{aligned} \tag{5.18}$$

Even if $\{\mathbf{A}_n\}_n$ did not satisfy the additional assumptions given in GLT1 or GLT2 for the symbol to describe the eigenvalue distribution, it is still true that $\{\mathbf{A}_n\}_n \sim_\lambda \kappa_A(x,\theta)$. This can be seen by looking at the eigenvalues of $\mathbf{A}_n$ for a fixed n and then letting n tend to infinity. As was deduced in (4.4), the eigenvalues of $\mathbf{A}_n$ are given by

$$\lambda_{A,j} = 1 + \frac{\Delta t}{\Delta x}\lambda_{B,j} \qquad j = 1, 2, ..., n.$$

Since this is true for all n, it follows that the asymptotic eigenvalue distribution of $\mathbf{A}_n$ is given by

$$1 + \frac{\Delta t}{\Delta x}\kappa_B(x,\theta) = \kappa_A(x,\theta). \tag{5.19}$$

Note that this would be true whether $\{\mathbf{A}_n\}_n$ was a GLT sequence or not.

For the different levels of the multigrid algorithm, the value of $\Delta t/\Delta x$ is doubled and halved when moving between levels (since Δt is fixed and Δx is doubled when moving to a coarser grid). To get the approximation of the eigenvalue distribution of a matrix $\mathbf{A}_n$ for a given level, one would have to fix Δx and use the symbol for the corresponding matrix sequence. Here, with some abuse of notation, the eigenvalue distribution of the sequence $\{A_n\}_n$ with $\Delta x = 1/n$ dependent on n will be said to have an eigenvalue distribution described by κ_A in (5.18), where κ_A is dependent on Δx.

5.5 Numerical examples

To see how closely the eigenvalues are distributed according to the symbol for various n, or at least how well the eigenvalues covers the image of the symbol, some numerical tests were done. The values

$$\begin{aligned} x_{min} &= 0, \\ x_{max} &= 2, \text{ and} \\ \Delta t &= 3/2^5, \end{aligned}$$

were used for the discretization in all examples below. For each test, defined by which function $a(x)$ that was used, matrices $\mathbf{A}_n$ of size $n = 128, 1024, 4096, 8192$ were created. The eigenvalues were then computed in python using the function scipy.linalg.eig from the SciPy library.

5.5.1 Sine function

The first test was done with the function

$$a(x) = 1 + 0.7 \sin\left(\frac{4\pi x}{x_{max} - x_{min}}\right) \tag{5.20}$$

(Figure 5.1). Figure 5.2 shows plots of the eigenvalues of $\mathbf{A}_n$ for the various n. The orange, green and red points show the eigenvalues for the matrix in the constant coefficient case, where the constant $a = a_{min}$, $a = a_{mean}$, and $a = a_{max}$ respectively. Remember that these eigenvalues are given exactly by the explicit expression in (4.22). The area between the orange and the red circles corresponds to the image of the symbol $\kappa_A(x, \theta)$. Note that since $\Delta x = 2/n$ is changed with increasing n, the ranges for the eigenvalues are also changed with n.

As expected from the GLT theory, the distribution of the eigenvalues gets closer to the image with increasing n. For values of $n < 128$, all eigenvalues were located inside the green circle created from the eigenvalues for $\mathbf{A}_n$ with $a(x) \equiv a_{mean}$.

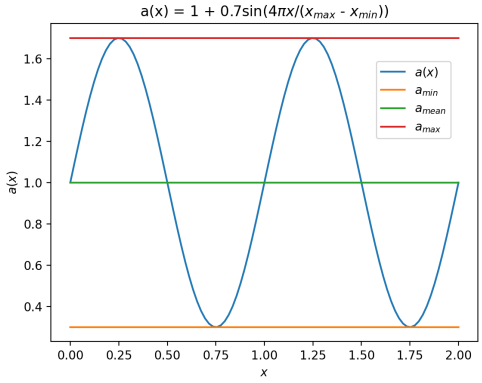


Figure 5.1: The function $a(x) = 1 + 0.7 \sin\left(\frac{4\pi x}{x_{max} - x_{min}}\right)$ along with lines representing the maximum, the minimum, and the mean of $a(x)$.

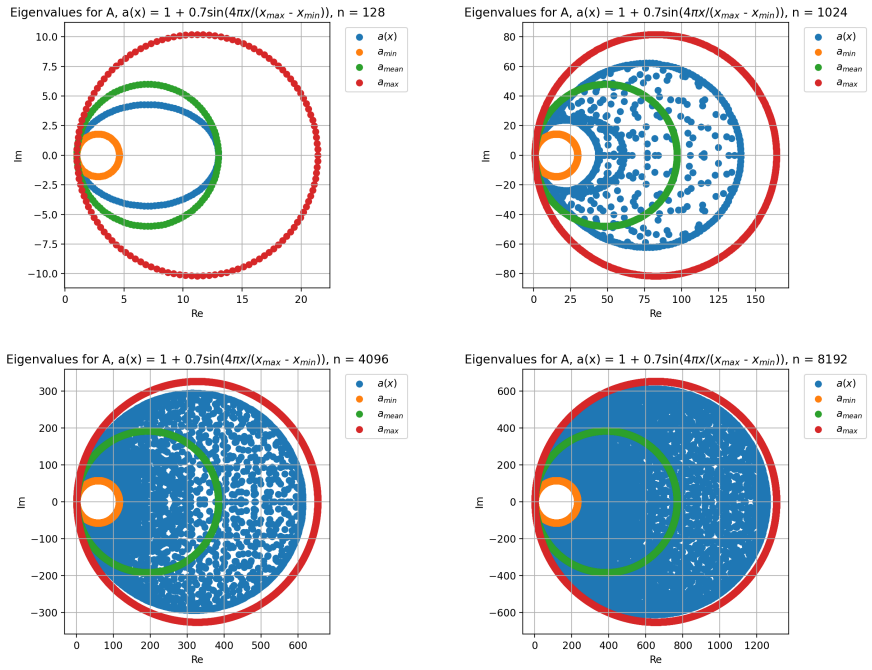


Figure 5.2: Eigenvalue plots for $\mathbf{A}_{128}$, $\mathbf{A}_{1024}$, $\mathbf{A}_{4096}$, and $\mathbf{A}_{8192}$, created by $a(x) = 1 + 0.7 \sin\left(\frac{4\pi x}{x_{max} - x_{min}}\right)$.

5.5.2 Monotone convex function

The second test was done with the function

$$a(x) = 1 + x^2 \tag{5.21}$$

(Figure 5.3). The result is shown in Figure 5.4. As expected, the distribution get closer to the image of $\kappa_A(x, \theta)$ as n is increased in this case as well.

For this coefficient function, the eigenvalues are outside of the green circle even for low values of n (even for n as low as $n = 4$). Another result that differs from the sine function case is that the eigenvalue distribution converges more slowly towards the red circle. A possible reason for these results is that the mean of $a(x)$

$$a_{mean} = \frac{1}{2}\int_0^2 1 + x^2 dx = \frac{1}{2}\left[x + \frac{x^3}{3}\right]_0^2 = \frac{7}{3} \approx 2.33 \tag{5.22}$$

shown as the green line in Figure 5.3 is below the midpoint 3 of the range of the function $a(x) \in [1, 5]$. This means that the point evaluations $a_{i+1/2}$ for the discretized version of $a(x)$ are weighted towards the lower half of the range. For this reason, a third test was done with a concave function.

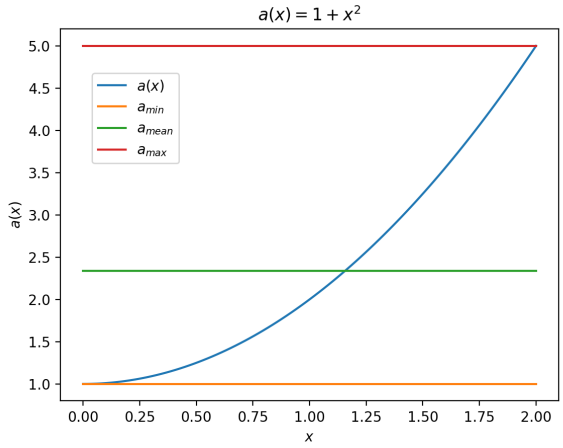


Figure 5.3: The function $a(x) = 1 + x^2$ along with lines representing the maximum, the minimum, and the mean of $a(x)$.

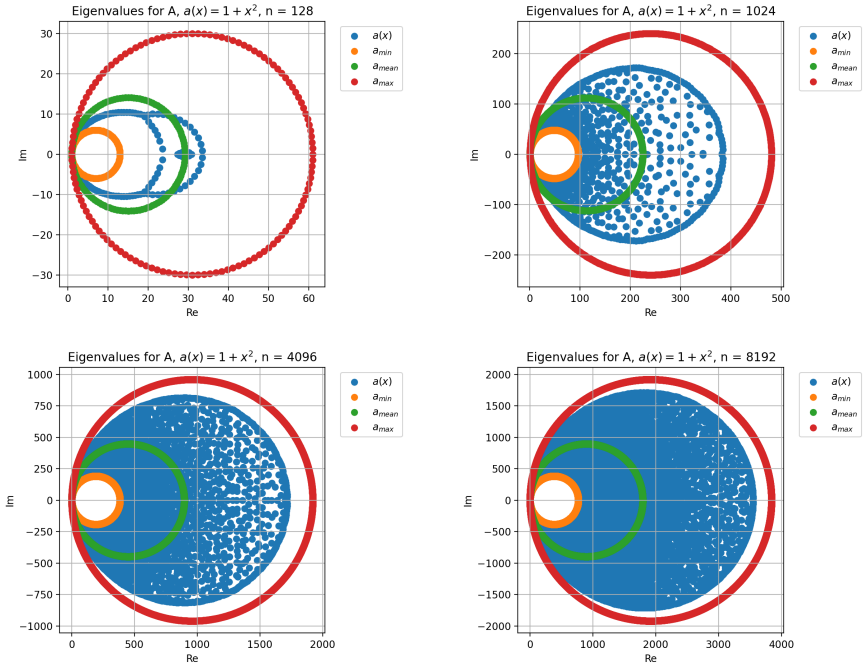


Figure 5.4: Eigenvalue plots for $\mathbf{A}_{128}$, $\mathbf{A}_{1024}$, $\mathbf{A}_{4096}$, and $\mathbf{A}_{8192}$, created by $a(x) = 1 + x^2$.

5.5.3 Concave function

The third test was done with the function

$$a(x) = 1 + 8x - 4x^2 \tag{5.23}$$

(Figure 5.5). This function is a concave function that was chosen to have the same range as the monotone convex function $a(x) \in [1, 5]$. Here, the mean of $a(x)$

$$a_{mean} = \frac{1}{2}\int_0^2 1 + 8x - 4x^2 dx = \frac{1}{2}\left[x + 4x^2 - \frac{4x^3}{3}\right]_0^2 = \frac{11}{3} \approx 3.67 \tag{5.24}$$

is above the midpoint 3 of the range. The eigenvalue distributions for this function are shown in Figure 5.6. As suspected, the distribution in this case converges faster towards the red circle and slower towards the orange circle compared to both the sine function, and the monotone convex function case. Overall the convergence seems to be the fastest for the concave function case. This is not necessarily true in general for GLT sequences, but is the case here for this discretization of the linear advection equation. The key point here is that the convergence of the eigenvalues towards the image of the domain through the symbol depends on $a(x)$ even if they result in the same image of the symbol.

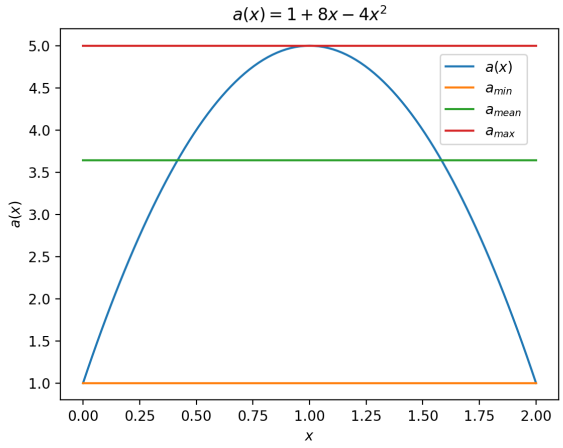


Figure 5.5: The function $a(x) = 1 + 2x - x^2$ along with lines representing the maximum, the minimum, and the mean of $a(x)$.

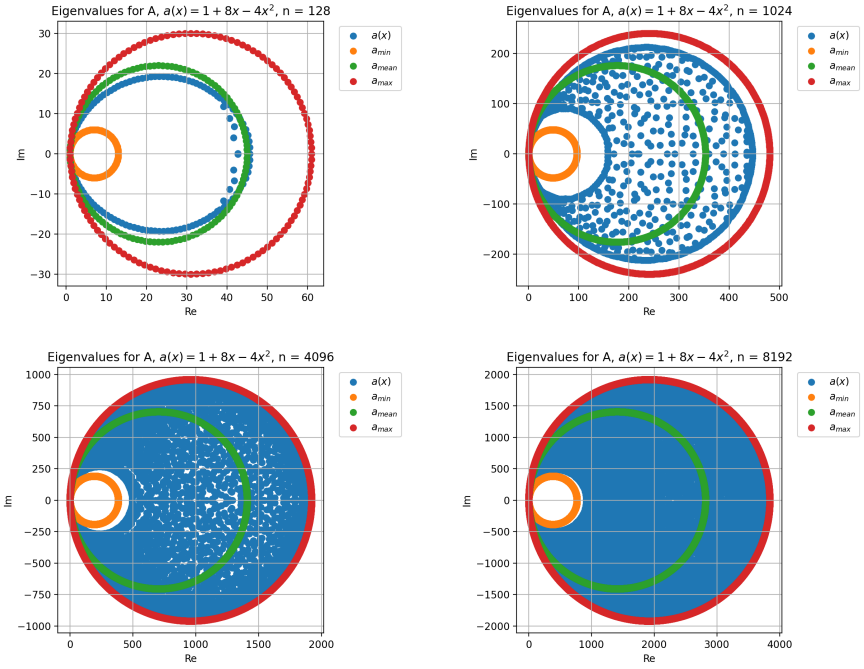


Figure 5.6: Eigenvalue plots for $\mathbf{A}_{128}$, $\mathbf{A}_{1024}$, $\mathbf{A}_{4096}$, and $\mathbf{A}_{8192}$, created by $a(x) = 1 + 2x - x^2$.

Remark. A question that arises is whether it can be assumed that all eigenvalues are located within the image of the symbol as they appear to do for the examples given here. That all eigenvalues are located on or inside the red circles in Figures 5.2, 5.4, and 5.6, can be proven by again turning to the Gershgorin circle theorem. Given a coefficient function $a(x)$, the eigenvalues of the resulting matrix $\mathbf{B}_n$ are located on or inside the circle centered at a_{max}, as was stated in section 4.1. Since the eigenvalues of $\mathbf{A}_n$ are given by (4.4), i.e.

$$\lambda_{A,j} = 1 + \frac{\Delta t}{\Delta x}\lambda_{B,j} \qquad j = 1, 2, ..., n,$$

the eigenvalues of $\mathbf{A}_n$ must all be located on or inside the circle centered at

$$1 + \frac{\Delta t}{\Delta x}a_{max}$$

with radius a_{max}, which describes the red circles in the figures.

Chapter 6

Optimizing explicit Runge-Kutta smoothers

Using the theory of GLT sequences it is now possible to approximate the distribution of the eigenvalues of $\mathbf{M}_n = P_s(-\Delta t^* \mathbf{A}_n)$ with $\mathbf{A}_n$ defined as in (2.14).

6.1 Symbol of $\{\mathbf{M}_n\}_n$

By Corollary 11, the symbol of the matrix sequence $\{\mathbf{M}_n\}_n$ is

$$\begin{aligned} \kappa_M(x,\theta) &= P_s(-\Delta t^* \kappa_A(x,\theta)) \\ &= P_s\left(-\Delta t^* \left[1 + \frac{\Delta t}{\Delta x} a(x)(1-\exp(-\mathrm{i}\theta))\right]\right). \end{aligned} \tag{6.1}$$

If $a(x) \equiv a$ is constant and if θ only takes the discrete values $\theta = 2\pi j/n$, $j \in \mathbb{Z} : j \in (-\frac{n}{2}, \frac{n}{2}]$, then this becomes the same expression as (4.24), which is the exact expression for the eigenvalues of $\mathbf{M}_n$ in the constant coefficient case.

Since $\mathbf{x}_{t^*} = \mathbf{b} - \mathbf{A}\mathbf{x} \to 0$ as $t^* \to \infty$, one would ideally want to take as large pseudo time steps Δt^* as possible. Since these smoothers are created from explicit schemes, the pseudo time step Δt^* is restricted by a CFL condition [10] dependent on the coefficient function and the discretization of the function.

Let

$$r := \frac{\Delta t}{\Delta x}, \tag{6.2}$$

$$c := \frac{\Delta t^* \Delta t}{\Delta x}, \tag{6.3}$$

and define

$$z(c,x,\theta;r) := -\frac{c}{r} - ca(x)(1-\exp(-\mathrm{i}\theta)). \tag{6.4}$$

The symbol can then be written as

$$\kappa_M(x,\theta) = P_s(z(c,x,\theta;r)). \tag{6.5}$$

Using the variables r and c avoids some divisions and multiplications with potentially very small numbers in addition to simplifying the expression. Thus, instead of optimizing directly for Δt^* here, Δt^* is instead solved from the optimized value of c in (6.3), which is a value closely related to the CFL condition. With the same idea as in subsection 5.4, to see that the eigenvalue distribution of $\{\mathbf{M}_n\}_n$ is given by its symbol, let $\mathbf{v}_A$ be an eigenvector of $\mathbf{A}_n$ with n fixed and let λ_A be the corresponding eigenvalue. Then the following holds

$$\mathbf{M}_n\mathbf{v}_A = P_s(-\Delta t^*\mathbf{A}_n)\mathbf{v}_A = P_s(-\Delta t^*\lambda_A)\mathbf{v}_A. \tag{6.6}$$

Since this is true for each fixed n, it follows that (by letting n tend to infinity)

$$\{\mathbf{M}_n\}_n \sim_\lambda P_s(-\Delta t^*\kappa_A(x,\theta)) = P_s(z(c,x,\theta;r)). \tag{6.7}$$

Hence, convergence of the RK smoother requires that parameters are chosen so that

$$\max_{x,\theta} |P_s(z(c,x,\theta;r))| < H_n, \tag{6.8}$$

for a bound H_n defined by

$$H_n := \frac{\max_{x,\theta} |P_s(z(c,x,\theta;r))|}{|\lambda_{max}(\mathbf{M}_n)|} \xrightarrow{n\to\infty} 1, \tag{6.9}$$

as this would imply that $|\lambda_{max}(\mathbf{M}_n)| < 1$.

6.2 Approximate optimization problem

The parameters of the s-stage RK smoother to optimize are the coefficients $\alpha_i, i = 1, ..., s$ and c. The approximate optimization problem becomes

$$\min_{c,\alpha_1,...,\alpha_s} \left[\max_{(x,\theta)\in[x_{min},x_{max}]\times[\pi/2,\pi]} |P_s(z(c,x,\theta;r))| \right], \tag{6.10}$$

where Δt^* is solved from c by $\Delta t^* := \frac{c\Delta x}{\Delta t}$. The domain for θ is cut down to $[\pi/2, \pi]$ for the following reasons. Firstly, to optimize for the convergence of high frequency components, the domain $[-\pi, \pi]$ is cut in half into $[-\pi, -\pi/2]\cup[\pi/2, \pi]$ (this corresponds to $|j| \geq n/4$ in the discrete case). Secondly, since $|P_s(z)| = |P_s(\overline{z})|$, where $\overline{z}$ is the complex conjugate of z, and since the complex conjugate of $\exp(-\mathrm{i}\theta)$ is $\exp(\mathrm{i}\theta)$, the domain $[-\pi, -\pi/2] \cup [\pi/2, \pi]$ is cut in half once again into $[\pi/2, \pi]$.

6.2.1 Parameters from optimization problem on coarse grids

Since the eigenvalue distribution results from the GLT theory are asymptotic, there is no guarantee that the parameters $\Delta t^*, \alpha_1, ..., \alpha_s$ obtained from the optimization problem above are accurate for grids with few grid points. However,

if all of the eigenvalues are located within the image given by the symbol, as they were in the numerical examples above, then

$$\max_{x,\theta} |P_s(z(c,x,\theta;r))| \geq |\lambda_{max}(P_s(-\Delta t^* \mathbf{A}_n)| \tag{6.11}$$

and the parameters should at least guarantee convergence (of the high frequency components) if the value of $\max_{x,\theta} |P_s(z(c,x,\theta;r))|$ is below one, i.e. $H_n \geq 1$ in (6.8). That all eigenvalues are at least located on or inside the red circles in Figures 5.2, 5.4, and 5.6, was proven in the remark in section 5.5.

6.2.2 Invariance of parameters on fine grids

One thing to note is that if the grids are fine enough, i.e. if n is large enough, so that $1/r = \Delta x/\Delta t \propto 1/n$ is small relative to $a(x)(1 - \exp(-i\theta))$, then

$$z(c,x,\theta;r) := -\frac{c}{r} - ca(x)(1 - \exp(-\mathrm{i}\theta) \approx -ca(x)(1 - \exp(-\mathrm{i}\theta)). \tag{6.12}$$

Since $z(c,x,\theta;r)$ is then approximately independent of r, and therefore the cell width Δx, the optimal parameters $c, \alpha_1, ..., \alpha_s$ are approximately constant on different levels of the multigrid algorithm. Thus, if a fine enough grid is used for the discretization of the problem, it is potentially not necessary to solve the problem for every level of the algorithm. The only parameter that needs to be changed for the RK smoother between levels with a large n is then

$$\Delta t^* = \frac{c\Delta x}{\Delta t} \tag{6.13}$$

which is doubled and halved when moving between levels.

6.2.3 Using linear functions

With the optimization problem defined as in (6.10), it is not necessary to know the exact distribution when solving the optimization problem. It is only necessary to know the image of the domain $[x_{min}, x_{max}] \times [\pi/2, \pi]$ through the distribution function. Let c and r be fixed. Since $a(x) \in \mathbb{R} \ \forall \ x$, the image of $[x_{min}, x_{max}] \times [\pi/2, \pi]$ through

$$z(c,x,\theta;r) = -\frac{c}{r} - ca(x)(1 - \exp(-\mathrm{i}\theta))$$

is equivalent to the image of $[a_{min}, a_{max}] \times [\pi/2, \pi]$ through

$$z(c,x,\theta;r) = -\frac{c}{r} - cx(1 - \exp(-\mathrm{i}\theta)),$$

where $a_{min} := \min_x a(x)$ and $a_{max} := \max_x a(x)$.

Therefore, solving (6.10) with the spatial domain $[a_{min}, a_{max}]$ and the linear coefficient function $a(x) = x$ is equivalent to solving (6.10) using any function $a(x)$ with the same extrema a_{min} and a_{max}. From this follows that, at least in

theory, the parameters obtained from solving (6.10) using a linear function solves the optimization problem for the set of all functions with the same extrema a_{min} and a_{max} (even if they have different spatial domains). In practice, if this problem is solved using a method such as grid search, as is done in this thesis, the solutions obtained can be slightly different. However, given a fine enough grid when performing the grid search, the results should be approximately the same, since asymptotically the values should cover the entire image of the domain through the symbol.

6.2.4 Lookup table

The above idea can be further developed. If it is necessary to quickly get parameters for problems with different coefficient functions $a_k(x)$, where the extrema $(a_k)_{min}$ and $(a_k)_{max}$ are not necessarily the same, an idea is to solve the problem (6.10) using the linear coefficient function $a(x) = x$, and the spatial domain $[C_{min}, C_{max}]$ that covers all the ranges $[(a_k)_{min}, (a_k)_{max}]$ created by the extrema, i.e. where

$$C_{min} \leq (a_k)_{min} \leq (a_k)_{max} \leq C_{max}, \qquad \forall\, k \tag{6.14}$$

and the function $a(x) = x$. Although the parameters obtained from this optimization problem are not optimal for each given problem, they will guarantee convergence of the high frequency components of the errors for the different problems given that

$$\max_{(x,\theta)\in[C_{min},C_{max}]\times[\pi/2,\pi]} |P_s(z(c,x,\theta;r))| < 1$$

holds. To see this, let

$$D_1 := [C_{min}, a_{min}] \times [\pi/2, \pi],$$

$$D_2 := [a_{min}, a_{max}] \times [\pi/2, \pi], \text{ and}$$

$$D_3 := [a_{max}, C_{max}] \times [\pi/2, \pi].$$

Then

$$\begin{aligned} \max_{(x,\theta)\in[a_{min},a_{max}]\times[\pi/2,\pi]} |P_s(z)| &= \max_{(x,\theta)\in D_2} |P_s(z)| \\ &\leq \max_{(x,\theta)\in(D_1\cup D_2\cup D_3)} |P_s(z)| \\ &= \max_{(x,\theta)\in[C_{min},C_{max}]\times[\pi/2,\pi]} |P_s(z)| \\ &< 1, \end{aligned} \tag{6.15}$$

where the variables for $z = z(c, x, \theta; r)$ were left out to shorten the expressions in the derivation.

Now if the extrema $(a_k)_{min}$ and $(a_k)_{max}$ are not known in advance, as is the case for example when solving the discretized inviscid Burgers' equation

where the coefficient function is changed at each time step, one can create a lookup table where parameters have been found for different ranges, for example $[0, 1], [0, 2], [0, 3], ...$, and then choose parameters from the smallest range in the lookup table that covers $[a_{min}, a_{max}]$. Zero has been used as the lower boundary for all ranges in this example since $a(x) > 0 \, \forall \, x$. More generally, the lookup table can be two dimensional with different lower and upper bounds, and given that this table can be precalculated, it can use a lot more accurate ranges.

Chapter 7

W-smoothers

Another class of smoothers studied in this thesis is the class of W-methods. These were derived as additive W-methods for solving convection-diffusion equations in [5]. Here, only the convective (advective) part of the additive W-methods is considered and the derivation is slightly different here to be consistent with the definitions in the derivation of the Runge-Kutta smoothers in chapter 4. This is a class of methods which comes from considering implicit Runge-Kutta methods for solving the initial value problem (4.1) instead of the class of explicit Runge-Kutta methods.

7.1 Pseudo time stepping

The initial value problem (4.1) is restated here for clarity:

$$\begin{aligned} \mathbf{x}_{t^*} &= \mathbf{f}(\mathbf{x}), \\ \mathbf{x}_0 &= \mathbf{x}(t_0^*). \end{aligned}$$

W-methods are derived by first considering a *singly diagonally implicit Runge-Kutta (SDIRK) method* given by

$$\begin{aligned} \mathbf{k}^{(0)} &= \mathbf{f}(\mathbf{x}_l) \\ \mathbf{k}^{(j)} &= \mathbf{f}(\mathbf{x}_l + \Delta t^*(\alpha_{j-1}\mathbf{k}^{(j-1)} + \eta\mathbf{k}^{(j)})), \qquad j = 1, ..., s, \\ \mathbf{x}_{l+1} &= \mathbf{x}_l + \Delta t^*\mathbf{k}^{(s)}, \end{aligned} \tag{7.1}$$

where, in general, s nonlinear equation systems have to be solved for the *stage derivatives* $\mathbf{k}^{(j)}$. If $\eta = 0$ this would be equivalent to the explicit RK method considered in chapter 4 with

$$\begin{aligned} \mathbf{x}_l^{(0)} &:= \mathbf{x}_l, \quad \text{and} \\ \mathbf{x}_l^{(j)} &:= \mathbf{x}_l + \alpha_j \Delta t^* \mathbf{k}^{(j)}. \end{aligned} \tag{7.2}$$

The s nonlinear equations are not solved exactly here, but instead approximated using one Newton step each with initial guess zero, resulting in what is called *Rosenbrock methods*. Approximating stage derivative $\mathbf{k}^{(j)}$ is equivalent to approximating $\mathbf{k}$ in

$$\mathbf{k} - \mathbf{f}(\mathbf{x}_l + \Delta t^*(\alpha_{j-1}\mathbf{k}^{(j-1)} + \eta\mathbf{k})) = \mathbf{0}. \tag{7.3}$$

Let

$$\mathbf{g}_j(\mathbf{k}) = \mathbf{k} - \mathbf{f}(\mathbf{x}_l + \Delta t^*(\alpha_{j-1}\mathbf{k}^{(j-1)} + \eta\mathbf{k})) \tag{7.4}$$

so that the problem becomes $\mathbf{g}_j(\mathbf{k}) = \mathbf{0}$. Then

$$\frac{\mathrm{d}\mathbf{g}_j(\mathbf{k})}{\mathrm{d}\mathbf{k}} = \mathbf{I} - \eta\Delta t^* \frac{\mathrm{d}\mathbf{f}(\mathbf{x}_l + \Delta t^*(\alpha_{j-1}\mathbf{k}^{(j-1)} + \eta\mathbf{k}))}{\mathrm{d}\mathbf{x}}. \tag{7.5}$$

Let $\mathbf{k}_1^{(j)}$ denote the value after one Newton step and let the initial guess be denoted by $\mathbf{k}_0^{(j)}$. Then $\mathbf{k}^{(j)}$ is defined as

$$\begin{aligned}
\mathbf{k}^{(j)} &:= \mathbf{k}_1^{(j)} \\
&= \mathbf{k}_0^{(j)} - \left(\frac{\mathrm{d}\mathbf{g}_j(\mathbf{k}_0^{(j)})}{\mathrm{d}\mathbf{k}}\right)^{-1} \mathbf{g}_j(\mathbf{k}_0^{(j)}) \\
&= -\left(\frac{\mathrm{d}\mathbf{g}_j(\mathbf{0})}{\mathrm{d}\mathbf{k}}\right)^{-1} \mathbf{g}_j(\mathbf{0}) \\
&= \left(\mathbf{I} - \eta\Delta t^* \frac{\mathrm{d}\mathbf{f}(\mathbf{x}_l + \alpha_{j-1}\Delta t^*\mathbf{k}^{(j-1)})}{\mathrm{d}\mathbf{x}}\right)^{-1} \mathbf{f}(\mathbf{x}_l + \alpha_{j-1}\Delta t^*\mathbf{k}^{(j-1)}) \\
&= (\mathbf{I} - \eta\Delta t^*\mathbf{J}_j)^{-1}\mathbf{f}(\mathbf{x}_l + \alpha_{j-1}\Delta t^*\mathbf{k}^{(j-1)}), \qquad j = 1, ..., s,
\end{aligned} \tag{7.6}$$

where

$$\begin{aligned}
\mathbf{J}_j &:= \frac{\mathrm{d}\mathbf{f}(\mathbf{x}_l + \alpha_{j-1}\Delta t^*\mathbf{k}^{(j-1)})}{\mathrm{d}\mathbf{x}} \\
&= \frac{\mathrm{d}\mathbf{f}(\mathbf{x}_l^{(j-1)})}{\mathrm{d}\mathbf{x}}.
\end{aligned} \tag{7.7}$$

Now the problem of solving s nonlinear equation systems has turned into a problem of solving s linear equation systems. Finally, the matrices $\mathbf{I} - \eta\Delta t^*\mathbf{J}_j$ are approximated by a matrix $\mathbf{W}$. This results in the W-methods with stage derivatives

$$\mathbf{k}^{(j)} = \mathbf{W}^{-1}\mathbf{f}(\mathbf{x}_l + \alpha_{j-1}\Delta t^*\mathbf{k}^{(j-1)}), \qquad j = 1, ..., s, \tag{7.8}$$

which can be rewritten in the same form as the explicit RK methods

$$\begin{aligned}
\mathbf{x}_l^{(0)} &= \mathbf{x}_l, \\
\mathbf{x}_l^{(j)} &= \mathbf{x}_l + \alpha_j\Delta t^*\mathbf{W}^{-1}\mathbf{f}(\mathbf{x}_l^{(j-1)}), \quad j = 1, ..., s-1 \\
\mathbf{x}_{l+1} &= \mathbf{x}_l + \Delta t^*\mathbf{W}^{-1}\mathbf{f}(\mathbf{x}_l^{(s-1)}).
\end{aligned} \tag{7.9}$$

In the same way as with the RK smoother, to solve the equation system $\mathbf{Au} = \mathbf{b}$, the time derivative is set to $f(x) = \mathbf{b} - \mathbf{Ax}$, and one iteration of the W-smoother consists of taking one step in pseudo time t^* of size Δt^*.

Here the focus is to test the GLT theory for the optimization of the W-smoothers rather than deciding for a good approximation matrix $\mathbf{W}$, so the matrix $\mathbf{W}$ is defined to be $\mathbf{W} = \mathbf{I} - \eta\Delta t^*\mathbf{J}_1$. With $f(x) = \mathbf{b} - \mathbf{Ax} \Rightarrow \mathbf{J}_1 = -\mathbf{A}$ this results in

$$\mathbf{W} = \mathbf{I} + \eta\Delta t^*\mathbf{A}. \tag{7.10}$$

Since $\mathbf{J}_j = -\mathbf{A}$ for all $j = 1, ..., s$ here, this effectively results in Rosenbrock smoothers.

7.2 Evolution of the error

Since the equations in (7.9) with $f(x) = \mathbf{b} - \mathbf{Ax}$ are equivalent to the explicit RK scheme with $\mathbf{A}$ changed to $\mathbf{W}^{-1}\mathbf{A}$, and with $\mathbf{b}$ changed to $\mathbf{W}^{-1}\mathbf{b}$, the evolution of the error for the W-smoother is the same as the evolution of the error for the explicit RK-smoother with $\mathbf{A}$ changed to $\mathbf{W}^{-1}\mathbf{A}$, i.e.

$$\mathbf{e}_{l+1} = P_s(-\Delta t^*\mathbf{W}^{-1}\mathbf{A})\mathbf{e}_l, \tag{7.11}$$

with $P_s(z)$ defined as in (4.7):

$$P_s(z) = \sum_{r=0}^{s} \left(\prod_{m=s-r+1}^{s+1} \alpha_m \right) z^r.$$

Let $\mathbf{M}^{(W)} := P_s(-\Delta t^*\mathbf{W}^{-1}\mathbf{A})$ denote the matrix describing the error evolution of the W-smoother, so that

$$\mathbf{e}_{l+1} = \mathbf{M}^{(W)}\mathbf{e}_l. \tag{7.12}$$

7.3 Optimization problem

The optimization problem for the W-smoother is found by following the same procedure as in chapter 4. With the error evolution given by (7.12), the exact optimization problem becomes

$$\min_{\Delta t^*, \alpha_1, ..., \alpha_{s-1}, \eta} \left[\max_{|j| \geq n/4} |\lambda_j| \right] \tag{7.13}$$

where λ_j are the eigenvalues of the matrix $\mathbf{M}^{(W)}$. Note that η, which is a parameter for $\mathbf{W}$, is an additional parameter to optimize here.

7.3.1 Eigenvalues of $\mathbf{W}^{-1}\mathbf{A}$

If $\tilde{\lambda}_j$ is an eigenvalue of $\mathbf{W}^{-1}\mathbf{A}$, then $\lambda_j = P_s(-\Delta t^* \tilde{\lambda}_j)$ is an eigenvalue of $\mathbf{M}^{(W)}$. Therefore, the eigenvalues of $\mathbf{W}^{-1}\mathbf{A}$ have to be determined. Let $\mathbf{v}_j$ be an eigenvector of $\mathbf{A}$, and let $\lambda_{A,j}$ be the corresponding eigenvalue. Then with $\mathbf{W}$ defined as in (7.10)

$$\begin{aligned} \mathbf{v}_j &= \mathbf{W}^{-1}\mathbf{W}\mathbf{v}_j \\ &= \mathbf{W}^{-1}(\mathbf{I} + \eta\Delta t^* \mathbf{A})\mathbf{v}_j \\ &= (1 + \eta\Delta t^* \lambda_{A,j})\mathbf{W}^{-1}\mathbf{v}_j \\ \Leftrightarrow \mathbf{W}^{-1}\mathbf{v}_j &= \frac{1}{1 + \eta\Delta t^* \lambda_{A,j}}\mathbf{v}_j, \end{aligned} \tag{7.14}$$

and from this follows that

$$\mathbf{W}^{-1}\mathbf{A}\mathbf{v}_j = \lambda_{A,j}\mathbf{W}^{-1}\mathbf{v}_j = \frac{\lambda_{A,j}}{1 + \eta\Delta t^* \lambda_{A,j}}\mathbf{v}_j. \tag{7.15}$$

Thus the eigenvalues $\tilde{\lambda}_j$ of $\mathbf{W}^{-1}\mathbf{A}$ are given by the expression

$$\tilde{\lambda}_j = \frac{\lambda_{A,j}}{1 + \eta\Delta t^* \lambda_{A,j}}, \qquad j = 1, 2, ..., n, \tag{7.16}$$

where n denotes the size $n \times n$ of the matrix $\mathbf{A}$.

7.3.2 Eigenvalue distribution of $\mathbf{M}^{(W)}$

As before, let the subscript n denote the size of the matrices. Since the eigenvalues of $\mathbf{W}_n^{-1}\mathbf{A}_n$ are given by (7.16), the eigenvalues of $\mathbf{M}_n^{(W)} = P_s(-\Delta t^* \mathbf{W}_n^{-1}\mathbf{A}_n)$ are given by

$$P_s(-\Delta t^* \tilde{\lambda}_j) = P_s\left(\frac{-\Delta t^* \lambda_{A,j}}{1 + \eta\Delta t^* \lambda_{A,j}}\right). \tag{7.17}$$

Therefore, the asymptotic eigenvalue distribution of $P_s(-\Delta t^* \mathbf{W}_n^{-1}\mathbf{A}_n)$ is given by

$$P_s\left(\frac{-\Delta t^* \kappa_A(x,\theta)}{1 + \eta\Delta t^* \kappa_A(x,\theta)}\right) = P_s\left(\frac{-\Delta t^* \left[1 + \frac{\Delta t}{\Delta x} a(x)(1 - \exp(-\mathrm{i}\theta))\right]}{1 + \eta\Delta t^* \left[1 + \frac{\Delta t}{\Delta x} a(x)(1 - \exp(-\mathrm{i}\theta))\right]}\right), \tag{7.18}$$

where $\kappa_A(x,\theta)$ is the eigenvalue distribution of A given by (5.18):

$$\kappa_A(x,\theta) = 1 + \frac{\Delta t}{\Delta x} a(x)(1 - \exp(-\mathrm{i}\theta)).$$

By introducing the variables r, c, and z as in (6.2), (6.3), and (6.4):

$$\begin{aligned} r &:= \frac{\Delta t}{\Delta x}, \\ c &:= \frac{\Delta t^* \Delta t}{\Delta x}, \\ z(c, x, \theta; r) &:= -\frac{c}{r} - c a(x)(1 - \exp(-\mathrm{i}\theta)), \end{aligned}$$

and the additional variable

$$z_W(\eta, c, x, \theta; r) = \frac{z(c, x, \theta; r)}{1 - \eta z(c, x, \theta; r)} \tag{7.19}$$

The expression in (7.18) can be written as:

$$P_s\left(\frac{z(c, x, \theta; r)}{1 - \eta z(c, x, \theta; r)}\right) = P_s(z_W(\eta, c, x, \theta; r)) \tag{7.20}$$

Since the matrix sequence created by taking the inverse of the matrices in a GLT sequence is not necessarily a GLT sequence, $\{\mathbf{W}_n^{-1}\mathbf{A}_n\}_n$, and thus $\{\mathbf{M}_n^{(W)}\}_n$, are not necessarily GLT sequences. Therefore, even though the eigenvalue distribution is given by (7.20), it is in general not the symbol of the matrix sequence $\{\mathbf{M}_n^{(W)}\}_n$.

7.3.3 Approximate optimization problem

With the asymptotic eigenvalue distribution of $\mathbf{M}_n^{(W)}$ given by (7.20), the approximate optimization problem becomes

$$\min_{c,\alpha_1,\ldots,\alpha_s,\eta} \left[\max_{(x,\theta)\in[x_{min},x_{max}]\times[\pi/2,\pi]} |P_s(z_W(\eta, c, x, \theta; r))| \right]. \tag{7.21}$$

7.3.4 Invariance of parameters

As was shown in subsection 6.2.2, if the grid is fine enough, i.e. if n is large enough, then

$$z(c, x, \theta; r) := -\frac{c}{r} - ca(x)(1 - \exp(-\mathrm{i}\theta) \approx -ca(x)(1 - \exp(-\mathrm{i}\theta)).$$

From this follows that

$$z_W(\eta, c, x, \theta; r) = \frac{z(c, x, \theta; r)}{1 - \eta z(c, x, \theta; r)} \approx \frac{-ca(x)(1 - \exp(-\mathrm{i}\theta))}{1 + \eta ca(x)(1 - \exp(-\mathrm{i}\theta))} \tag{7.22}$$

is approximately independent on r, so that the parameters $c, \alpha_1, ..., \alpha_s, \eta$ are approximately equal for different levels of the multigrid algorithm.

7.3.5 Using linear functions and lookup tables

The image of $z_W(\eta, c, x, \theta; r)$ using an arbitrary coefficient function $a(x)$ with the extrema a_{min} and a_{max} is the same as the image of $z_W(\eta, c, x, \theta; r)$ using the linear coefficient function $a(x) = x$ with spatial domain $[a_{min}, a_{max}]$. Thus, following the same reasoning as in subsection 6.2.3 one can optimize the coefficients using linear functions with the domain $[a_{min}, a_{max}]$ instead of the coefficient function in the problem formulation. From this follows that the idea of a lookup table can be used for the W-smoothers as well.

Chapter 8

Results and discussion

8.1 Description of example problems

The smoothers were tested on three different linear advection problems (2.1), that only differed on the coefficient functions. The three different coefficient functions used in these problems were the functions (5.20), (5.21), and (5.23) used in the numerical examples in chapter 5, i.e. the *sine function*

$$a(x) = 1 + 0.7 \sin\left(\frac{4\pi x}{x_{max} - x_{min}}\right),$$

the *monotone function*

$$a(x) = 1 + x^2,$$

and the *concave function*

$$a(x) = 1 + 8x - 4x^2.$$

For these tests the boundaries 0 and 2 were chosen so that $x \in [0, 2]$. With this spatial domain, the image of the sine function is the range [0.3, 1.7], and the images of the monotone function and the concave function are both the range [1, 5]. This resulted in the problem

$$\begin{aligned} &u_t(x,t) + (a(x)u(x,t))_x = 0, \quad a(x) > 0,\ a(x) \in \mathbb{R}\ \forall\ x \in [0,2], \\ &u(x=0,t) = u(x=2,t). \end{aligned} \tag{8.1}$$

The problems all used the step function

$$u_0 = \begin{cases} 5, & x \in [0,1) \\ 1, & x \in [1,2) \end{cases} \tag{8.2}$$

as initial value, and the problems were discretized so that the fine grid (top level) had $n = 2^{13} = 8192$ grid cells so that $\Delta x = 2/2^{13} = 2^{-12}$. For each

problem, one step forward in time was taken using the multigrid method with step size $\Delta t = 3/2^5$. With this cell width and this time step size, the CFL number for the sine problem is

$$\text{CFL} = \frac{a_{max}\Delta t}{\Delta x} = \frac{1.7 \cdot 3/2^5}{2^{-12}} = 652.8, \tag{8.3}$$

and the CFL number for the monotone problem and the concave problem is

$$\text{CFL} = \frac{5 \cdot 3/2^5}{2^{-12}} = 1920. \tag{8.4}$$

Since implicit Euler was used for the time steps, the solutions are stable even if the CFL numbers are large.

8.2 Construction of multigrid algorithm

Since the optimization problems (6.10) and (7.21) are independent of the number of pre- and postsmoothing steps, not too much focus was put into optimizing these numbers. For the RK smoothers, five presmoothing steps and five postsmoothing steps were used in the multigrid algorithm ($\nu_1 = 5, \nu_2 = 5$), and for the W-smoothers one presmoothing step and one postsmoothing step were used ($\nu_1 = 1, \nu_2 = 1$). On the lowest level consisted of one smoothing step (i.e. no direct solve was done). These were empirically found to be reasonable values, albeit not necessarily optimal.

8.3 Determining coefficients for the smoothers

Parameters for the RK-smoothers and W-smoothers were determined using the optimization problems (6.10) and (7.21) respectively. These problems are too difficult to solve exactly. Therefore, approximate solutions were computed using grid search, i.e. by discretizing the parameter space and exhaustively searching through the discretized space for the best parameters.

8.4 Explicit Runge-Kutta smoothers

The first smoothers studied here were explicit Runge-Kutta smoothers with 2 and 3 stages, here referred to as RK2 and RK3. To perform the grid search for the coefficients, a bounded region was required. The α_i's are bounded by $[0, 1]$ by definition, and the search space for the eigenvalue distribution function is defined in the approximate optimization problem. The only parameter left is c. Here $c \in [0, 2]$ was chosen. Since $\mathbf{Ax} \to \mathbf{b}$ as $t^* \to \infty$ and $\Delta t^* \propto c$, the optimal value of c should be as large as possible without making the solution become unstable. Thus, the region for c is large enough if the calculated optimal value of c is not on or close to (due to numerical errors) the upper boundary 2. Otherwise a larger region is needed.

For each optimization problem, a uniform grid of $400 \times 200 \times 200 \times 200$ points was chosen for the parameter space $c \times \alpha_1 \times x \times \theta$ for RK2 and a uniform grid of $400 \times 200 \times 200 \times 200 \times 200$ points was chosen for the parameter space $c \times \alpha_1 \times \alpha_2 \times x \times \theta$ for RK3.

8.4.1 Simplified Runge-Kutta smoother

Using the idea from section 6.2.2, the implementation of the RK smoothers was simplified by letting the α_is be the ones calculated for the finest grid for all levels of the algorithm and only change Δt^* by doubling it when moving down levels in the multigrid algorithm. This way, only two values need to be stored for RK2 (c and α_1, or alternatively Δt^* and α_1, for the top level) and only three values need to be stored for RK3 (c, α_1, and α_2 for the top level). Since the matrices here are relatively small, the coefficients might turn out to be too inaccurate for the lower levels of the algorithm.

The parameters for the finest grid for the different problems can be found in Table 8.1. Both the values of c and Δt^* are included in the table, as c was the value optimized in the optimization problem and Δt^* is the value used in the smoother. The values of $\max_{x,\theta} |P_s(z)|$ are also included in the table as this shows an upper bound for the convergence rate of the high frequency error components on the top level.

The monotone and the concave functions resulted in the same values for the coefficients (of RK2 and of RK3), as was expected from the discussion in subsection 6.2.3, i.e. that since both coefficient functions were designed to have the same image $[a_{min}, a_{max}] = [1, 5]$, they both result in the same optimization problem. They also resulted in similar values of $\max_{x,\theta} |P_s(z)|$, i.e. they should have similar convergence rate for the high frequency error components for a given level of the algorithm, given that the grid is fine enough that the distribution from the GLT theory approximates the eigenvalue distribution accurately.

Table 8.1: Coefficients of RK2 and RK3 for the top level of the three problems (referred to by their function).

smoother	**function**	c	Δt^*	α_1	α_2	$\max \lvert P_s \rvert$
RK2	**sine**	1.01	2.630e-3	0.265	-	0.7397
	monotone	0.335	8.724e-4	0.27	-	0.7187
	concave	0.335	8.724e-4	0.27	-	0.7187
RK3	**sine**	1.585	4.128e-3	0.105	0.3	0.6313
	monotone	0.525	1.367e-3	0.11	0.315	0.6039
	concave	0.525	1.367e-3	0.11	0.315	0.6039

Using the coefficients in Table 8.1, the smoothers were tested on the three example problems. In the initial tests, 13 grids were used in the multigrid algorithm so that the bottom level had 2 grid cells. The error $\mathbf{e}$ was calculated after each V-cycle by comparing the estimation from the multigrid algorithm to the solution given by the solver scipy.sparse.linalg.spsolve() from the SciPy

library in python (sparse matrices were used to optimize the performance). The program was terminated either after 1000 V-cycles or if the tolerance 10^{-12} had been reached, i.e. if $||\mathbf{e}||_2 \leq 10^{-12}$. The number of V-cycles to reach the tolerance for the different problems are shown in Table 8.2.

Table 8.2: Number of V-cycles until the tolerance $||\mathbf{e}||_2 \leq 10^{-12}$ was reached. For the sine problem, the error diverged when using RK3. However, when the number of grids was lowered to 12, the multigrid algorithm converged after 21 V-cycles.

smoother	**function**	**Number of V-cycles**
RK2	**sine**	20
	monotone	16
	concave	23
RK3	**sine**	did not reach - (24 with 12 levels)
	monotone	17
	concave	26

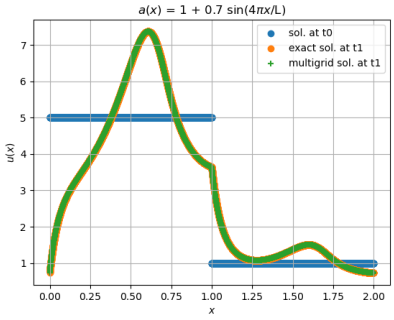
The error diverged when solving the sine problem using RK3. The reason for this is that the parameters from the finest grid are too inaccurate for the coarsest level. These parameters, i.e. $\alpha_1 = 0.11$, $\alpha_2 = 0.315$, and $\Delta t^* = 4.128 \cdot 10^{-3} \cdot 2^{12}$, give the value

$$\max_{(x,\theta)\in[0,2]\times[\pi/2,\pi]} |P_s(z(c, x, \theta; r))| \approx 249 \tag{8.5}$$

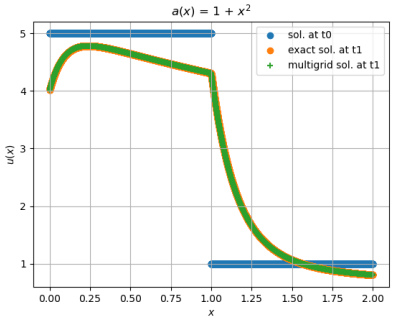
for the coarsest level (to guarantee convergence, a value of less than one is needed). When the number of grids was changed to 12 instead, the algorithm converged after 24 V-cycles.

The initial value, the solution after one time step from the SciPy solver, and the solution after one time step from the multigrid algorithm using RK2 are shown for each of the example problems in Figure 8.1. The solutions from the multigrid algorithm overlap with the solutions from the SciPy solver showing that the multigrid algorithm computes the correct solution.

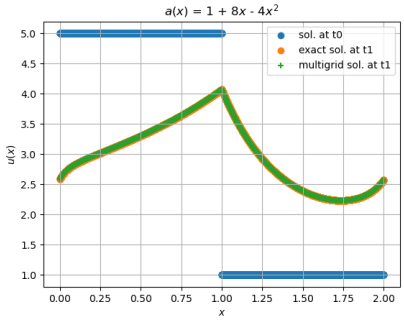
Except for the sine problem, lowering the number of grids for the methods did not improve upon the number of V-cycles. This is shown in Table 8.3 and Figure 8.2.



(a) Sine function.



(b) Monotone function.

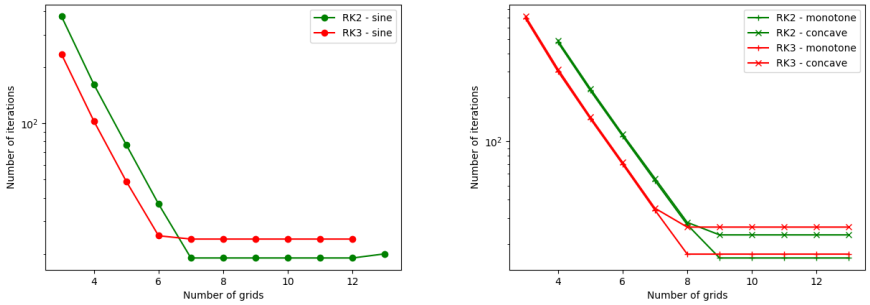


(c) Concave function.

Figure 8.1: Solutions of the example problems after one time step. The blue lines show the initial values, the orange lines show the solutions calculated using a solver from the SciPy library in python and the green lines show the solutions from the multigrid algorithm.

Table 8.3: Number of V-cycles to reach tolerance for the three different problems using the multigrid method with different number of grids. A dash represents that the algorithm did not reach the tolerance within 1000 V-cycles.

Number of grids		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
RK2	**sine**	-	-	375	162	77	37	19	19	19	19	19	19	20
	monotone	-	-	-	475	223	109	54	27	16	16	16	16	16
	concave	-	-	-	488	230	112	56	28	23	23	23	23	23
RK3	**sine**	-	-	235	103	49	25	24	24	24	24	24	24	-
	monotone	-	-	690	301	143	70	34	17	17	17	17	17	17
	concave	-	-	713	312	147	72	35	26	26	26	26	26	26



(a) Sine function.

(b) Monotone and concave function.

Figure 8.2: Lin-log plots of the results from Table 8.3. The green lines represent RK2 and the red lines represent RK3.

In general, increasing the number of grids improved the convergence rate. When using few grids, the three stage smoother RK3 performed better than the two stage smoother RK2. However, when a lot of grids are used, RK2 performed slightly better than RK3. The likely reason for this is that, since the degree of the stability polynomial for RK3 is higher than the degree of the stability polynomial for RK2, the value of the polynomial is more affected by the $1/r$ term of

$$z = -\frac{c}{r} - ca(x)(1 - \exp(-i\theta)) = -c\left[\frac{1}{r} + a(x)(1 - \exp(-i\theta))\right],$$

which changes between levels.

Approximately the same number of V-cycles are needed for solving the monotone function and solving the concave function when only using few grids (for both RK2 and RK3). This was a likely outcome since both problems have similar upper bounds for the convergence rate for the high frequency error components on the finer grids, as was discussed above.

8.4.2 Using linear functions

Using the idea from subsection 6.2.3, the optimization problem (6.10) was solved with the coefficient functions changed to linear functions

$$a(x) \to x$$

and the range set to $[a_{min}, a_{max}]$

$$[x_{min}, x_{max}] \to [a_{min}, a_{max}].$$

For the sine function the range is $a \in [0.3, 1.7]$, and for the monotone and concave functions the range is $a \in [1, 5]$. The results are shown in Table 8.4.

Table 8.4: Coefficients of RK2 and RK3 for the top level. Here linear functions were used instead of the coefficient functions. The range $[0.3, 1.7]$ corresponds to the sine function and the range $[1, 5]$ corresponds to the monotone function and the concave function.

smoother	**range**	c	Δt^*	α_1	α_2	$\max \lvert P_s \rvert$
RK2	**[0.3, 1.7]**	1.01	2.630e-3	0.265	-	0.7397
	[1, 5]	0.335	8.724e-4	0.27	-	0.7187
RK3	**[0.3, 1.7]**	1.585	4.128e-3	0.105	0.3	0.6313
	[1, 5]	0.525	1.367e-3	0.11	0.315	0.6039

As expected, the results in this table are the same as the results in Table 8.1 for the corresponding coefficient functions $a(x)$.

8.4.3 Lookup table

Since a lookup table can be calculated in advance, the regions used in a look up table can be very accurate. However, the more accurate the regions, the more time is needed to calculate the coefficients for the lookup table (and the more storage is needed). Thus, the effect of using a somewhat inaccurate lookup table was tested by calculating coefficients for regions a bit larger than $[a_{min}, a_{max}]$. If a lookup table was created for the ranges $[0, n]$ for $n = 1, 2, ...N$ for some integer $N \geq 5$, then the smallest range covering $[0.3, 1.7]$ corresponding to the sine problem would be the range $0, 2$ and the smallest range covering $[1, 5]$ corresponding to the other two problems would be $[0, 5]$. Therefore, coefficients were solved for for the ranges $[0, 2]$ and $[0, 5]$ (Table 8.5).

Table 8.5: Coefficients for RK2 and RK3 for the top level (calculated using linear functions).

smoother	**range**	c	Δt^*	α_1	α_2	$\max \lvert P_s \rvert$
RK2	**[0, 2]**	0.995	2.591e-3	0.24	-	0.9974
	[0, 5]	0.395	1.029e-3	0.23	-	0.9990
RK3	**[0, 2]**	1.685	4.388e-3	0.085	0.245	0.9956
	[0, 5]	0.675	1.758e-3	0.085	0.245	0.9982

These coefficients were used to solve the three example problems (Table 8.6 and Figure 8.3). With RK2, the algorithm converges. However, when using RK3 with the coefficients from Table 8.5 the error diverged, even when only using few grids.

Table 8.6: Number of V-cycles to reach tolerance when using coefficients from Table 8.5.

Number of grids		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
RK2	**sine**	-	-	378	165	78	38	21	21	21	21	21	21	21
	monotone	-	-	913	400	190	93	46	25	24	23	23	23	24
	concave	-	-	-	-	217	96	78	76	76	76	75	76	76
RK3	**sine**	-	-	-	-	-	-	-	-	-	-	-	-	-
	monotone	-	-	-	-	-	-	-	-	-	-	-	-	-
	concave	-	-	-	-	-	-	-	-	-	-	-	-	-

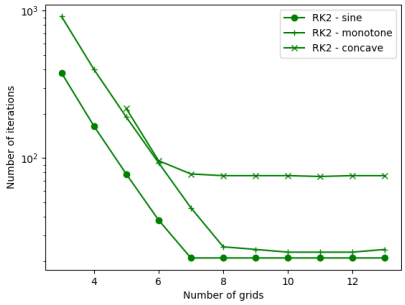


Figure 8.3: Lin-log plot of the results for RK2 from Table 8.6.

Since setting the lower boundary to 0 might be too limiting for the performance, a test was done with 0.1 as the lower boundary, so that coefficients were solved for the ranges [0.1, 2] and [0.1, 5]. The coefficients are shown in Table 8.7, and the results from solving the problems using these coefficients are shown in Table 8.8 and Figure 8.4.

Table 8.7: Coefficients for RK2 and RK3 for the top level.

smoother	**range**	c	Δt^*	α_1	α_2	$\max \lvert P_s \rvert$
RK2	**[0.1, 2]**	0.95	2.474e-3	0.255	-	0.9072
	[0.1, 5]	0.39	1.016e-3	0.25	-	0.9608
RK3	**[0.1, 2]**	1.545	4.023e-3	0.095	0.275	0.8538
	[0.1, 5]	0.645	1.680e-3	0.09	0.265	0.9360

Table 8.8: Number of V-cycles to reach tolerance using coefficients from Table 8.7.

number of grids		1	2	3	4	5	6	7	8	9	10	11	12	13
RK2	**sine**	-	-	396	172	82	39	20	19	19	19	19	19	19
	monotone	-	-	924	404	191	94	47	23	19	19	19	19	19
	concave	-	-	957	419	199	96	73	72	71	71	71	71	71
RK3	**sine**	-	-	-	106	50	36	36	36	36	36	36	36	36
	monotone	-	-	-	-	-	-	-	-	-	-	-	-	-
	concave	-	-	-	-	-	-	-	-	-	-	-	-	-

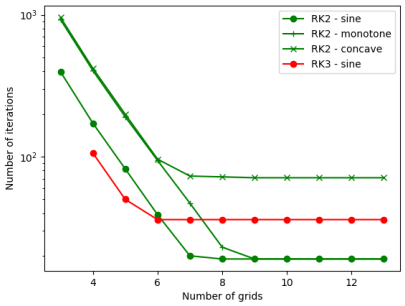


Figure 8.4: Lin-log plot of the results from Table 8.8. The green lines represent the results for RK2 and the red line represents the results for the sine problem for RK3.

When using many grids, the convergence rate improved. When using RK2 with 13 grids, the improvement was most noticeable for the monotone problem, where the solution was found in 19 V-cycles instead of 24, about 21% less. For RK3, the algorithm converged for the sine problem. However, the errors of the other problems still diverged. When using few grids, the improvement is not so clear. For example, the convergence rate was a bit worse for the sine and monotone problems when using the RK2 smoother with only 3 grids. Thus, the performance can potentially be improved by using a two dimensional lookup table, with different upper and lower boundaries. However, no clear conclusion can be drawn from the examples with 0.1 as lower boundary for the ranges. (In practice, the regions used for the lookup table can be considerably more accurate.)

If a lookup table is used, RK3 requires more accurate regions than RK2, which is more likely to converge if the parameters of the smoother are sub-optimal.

8.5 W-smoothers

As with the Runge-Kutta smoothers, only W-smoothers with 2 and 3 stages, referred to as W2 and W3, were studied. The optimization problem is given by (7.21). The only additional parameter for the W-smoothers is η, chosen to have the range $[0, 1]$. Since these methods are derived from implicit Runge-Kutta schemes, they can use a lot larger pseudo step sizes. By using a very large fixed value of c, one can optimize for one less parameter. Here c was set to $c = 10^{16}$, giving $\Delta t^* \approx 2.604 \cdot 10^{13}$. For a given level, a grid of $200 \times 200 \times 200 \times 200$ points was chosen for the parameter space $\alpha_1 \times \eta \times x \times \theta$ for W2 and a grid of $200 \times 200 \times 200 \times 200 \times 200$ points was chosen for the parameter space $\alpha_1 \times \alpha_2 \times \eta \times x \times \theta$ for W3.

8.5.1 Simplified W-smoother

The implementation of the W-smoothers was simplified in the same way as the RK smoothers, i.e. by only using the coefficients calculated for the finest grid, and multiplying Δt^* by two when moving down levels. The coefficients obtained for the different smoothers for the fine grid of $n = 8192$ grid cells are shown in Table 8.9.

Table 8.9: Coefficients of W2 and W3 for the top level of the three problems.

smoother	**function**	Δt^*	α_1	α_2	η	$\max \lvert P_s \rvert$
W2	**sine**	2.604e13	0.25	-	0.5	2.958e-31
	monotone	2.604e13	0.25	-	0.5	9.861e-32
	concave	2.604e13	0.25	-	0.5	9.861e-32
W3	**sine**	2.604e13	0.1	0.32	0.4	2.220e-16
	monotone	2.604e13	0.045	0.275	0.45	1.863e-17
	concave	2.604e13	0.045	0.275	0.45	1.813e-17

As expected from subsection 7.3.4, the coefficients calculated for the monotone and concave function were the same. The values of $\max |P_s|$ in this table are worse for the three stage smoother W3 than they are for the two stage smoother W2. This could mean that a finer grid is needed for the search space for the parameters α_1, α_2, and η.

Since the values of $\max |P_s|$ are so small for both W2 and W3, meaning that the high frequency error components should converge very fast, at least for the top level, the number of presmoothing steps and postsmoothing steps in the multigrid algorithm were set to $\nu_1 = \nu_2 = 1$. The results are shown in Table 8.10. The results were slightly better for W2 here, which was somewhat expected considering the values of $\max |P_s|$ in Table 8.9. Since the algorithm converges in one V-cycle here even without using multiple grids, it is likely better to use these iterative algorithms not as a part of a multigrid scheme but instead by themselves (note that the smoothers here are effectively Rosenbrock

smoothers, and that this statement is not necessarily true in general for W-smoothers, where other approximation matrices **W** are used). In this case it is better to solve for coefficients in the entire theta-range $\theta \in [-\pi, \pi]$ instead of optimizing only for the high frequency error components.

Table 8.10: Number of V-cycles to reach tolerance using W2 and W3 with parameters from Table 8.9.

Number of grids		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
W2	**sine**	1	1	1	1	1	1	1	1	1	1	1	1	1
	monotone	1	1	1	1	1	1	1	1	1	1	1	1	1
	concave	1	1	1	1	1	1	1	1	1	1	1	1	1
W3	**sine**	3	1	1	1	1	1	1	1	1	1	1	1	1
	monotone	2	1	1	1	1	1	1	1	1	1	1	1	1
	concave	6	1	1	1	1	1	1	1	1	1	1	1	1

8.5.2 Using linear functions

The optimization problem was solved using linear functions with the ranges $[0.3, 1.7]$ and $[1, 5]$. The same grids were used for the parameter space as above to compare the values to Table 8.9. The results are shown in Table 8.11.

Table 8.11: Coefficients for W2 and W3 for the top level. The range [0.3, 1.7] corresponds to the sine function and the range [1, 5] corresponds to the monotone function and the concave function.

smoother	**range**	Δt^*	α_1	α_2	η	$\max \lvert P_s \rvert$
W2	**[0.3, 1.7]**	2.604e13	0.25	-	0.5	2.958e-31
	[1, 5]	2.604e13	0.25	-	0.5	9.861e-32
W3	**[0.3, 1.7]**	2.604e13	0.055	0.275	0.55	1.536e-16
	[1, 5]	2.604e13	0.045	0.275	0.45	1.959e-17

The only parameters that differed here compared to the corresponding problem in Table 8.9 were the ones calculated for W3 for the range $[0.3, 1.7]$, which corresponds to the sine problem, further suggesting that a finer grid was needed for the W3 smoother. Since the parameters differed, the sine problem was solved using W3 with parameters from this table. The results are shown in Table 8.12.

Table 8.12: Number of V-cycles to reach tolerance solving the sine problem with W3 with coefficients from Table 8.11.

number of grids		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
W3	**sine**	2	1	1	1	1	1	1	1	1	1	1	1	1

With parameters from Table 8.11, the sine function was still solved in 1 V-cycle when using multiple grids. When no additional grids were used, the

problem was solved in two V-cycles, which is fewer V-cycles than when coefficients from Table 8.9 were used. Thus, it is not necessarily worse to use linear functions when solving for coefficients.

8.5.3 Lookup table

The idea of a lookup table was tested for W2. Since W3 performed worse than W2 in the tests above, and likely need needs finer grids for the parameter search space, this was not tested for W3. The coefficients calculated for the ranges [0, 2], [0.1, 2], [0, 5], [0.1, 5] are shown in Table 8.13.

Table 8.13: Coefficients for W2 for the top level.

smoother	**range**	Δt^*	α_1	η	$\max \lvert P_s \rvert$
W2	**[0, 2]**	2.604e13	0.25	0.5	1.483e-28
	[0, 5]	2.604e13	0.25	0.5	2.840e-29
	[0.1, 2]	2.604e13	0.25	0.5	2.367e-30
	[0.1, 5]	2.604e13	0.25	0.5	2.367e-30

The coefficients calculated for these ranges are the same as the ones for the original coefficient functions, which shows that the lookup table idea also works for these problems for the smoother W2.

Chapter 9

Conclusion

In this thesis two different classes of iterative methods, or smoothers, were studied in the context of multigrid algorithms with the one dimensional linear advection equation as model problem. The first one was a class of explicit Runge-Kutta smoothers (RK smoothers). These smoothers can be optimized by optimizing the eigenvalues of the matrix $\mathbf{M} := P_s(-\Delta t^* \mathbf{A})$ where $P_s(z)$ is the stability polynomial of the RK scheme, Δt^* is a parameter of the RK smoother, and $\mathbf{A}$ comes from the equation system $\mathbf{Au} = \mathbf{b}$ to be solved. The variable coefficient linear advection problem results in a matrix $\mathbf{M}$ where there is no general expression for the eigenvalues, and for large matrices it becomes costly to solve for the eigenvalues iteratively.

Generalized Locally Toeplitz (GLT) theory can be used to find a function, called the symbol, of a GLT sequence, that describes the asymptotic singular value distribution, and given some extra assumptions also the asymptotic eigenvalue distribution, of the matrices in the sequence. The larger the matrices, the more accurately the distributions are described by the symbol. The GLT theory can be used to set up an approximate optimization problem to find good parameters for the RK smoothers.

For the two stage RK smoother, RK2, the coefficients (excluding the pseudo time step Δt^*) did not change considerably between the grids and the algorithm converged even when the coefficients were a bit sub-optimal. The three stage RK smoother, RK3, was more affected by the size of the grid, and in all cases here, when a lot of grids were used in the multigrid algorithm, it was outperformed by RK2. This might be less of an issue, however, if the bottom level of the algorithm is fine enough (so that the term $1/r = \Delta t/\Delta x$ is small). In addition, when using RK3, the error was more prone to diverge when the coefficients were sub-optimal.

Given the structure of the optimization problem, it is not necessary to use the coefficient function $a(x)$ from the problem. Instead one can use a linear function defined to have the same range as $a(x)$. Therefore, given a range, solving the optimization problem for a linear function solves the optimization problem for all coefficient functions with the same range. This makes it possible

to create a lookup table for the coefficients that can be created in advance given the discretization of the problem. For this, using RK2 rather than RK3 has some benefits. Firstly, solving for the parameters is less costly in terms of computing time and storage needed, meaning that more accurate regions may be used. Secondly, as mentioned above, if the coefficients are a bit sub-optimal, with RK2 the algorithm is more likely to still converge.

The other class of smoothers studied here was the class of W-smoothers that was developed from considering implicit Runge-Kutta schemes. These smoothers are similar to the RK smoothers and are optimized in a similar way. The eigenvalues to optimize here are the eigenvalues of the matrix $\mathbf{M}^{(W)} := P_s(-\Delta t^* \mathbf{W}^{-1}\mathbf{A})$. The ideas of using linear functions and creating a lookup table for the coefficients work here as well.

Since these smoothers are developed from implicit schemes, the pseudo time step taken can be significantly larger than for the explicit RK smoothers. This means that much better convergence rates can be achieved, and for the problems here the algorithm was able to converge in one V-cycle for the two stage smoother W2 even when only one grid was used. The three stage smoother, W3, needed at least two grids to be able to converge in one V-cycle. A possible reason for this worse performance is that a finer grid might have been needed in the grid search for the parameters of the smoother.

Bibliography

[1] A. Jameson and D. Caughey, "How many steps are required to solve the euler equations of steady, compressible flow-in search of a fast solution algorithm," in *15th AIAA Computational Fluid Dynamics Conference*, p. 2673, 2001.

[2] P. Birken, "Optimizing Runge-Kutta smoothers for unsteady flow problems," *Electronic Transactions on Numerical Analysis*, vol. 39, no. 1, pp. 298–312, 2012.

[3] C. Garoni and S. Serra-Capizzano, *Generalized Locally Toeplitz sequences: theory and applications*, vol. 1. Springer, 2017.

[4] D. Bertaccini, M. Donatelli, F. Durastante, and S. Serra-Capizzano, "Optimizing a multigrid Runge–Kutta smoother for variable-coefficient convection–diffusion equations," *Linear Algebra and its Applications*, vol. 533, pp. 507–535, 2017.

[5] P. Birken, J. Bull, and A. Jameson, "Preconditioned smoothers for the full approximation scheme for the RANS equations," *Journal of Scientific Computing*, vol. 78, no. 2, pp. 995–1022, 2019.

[6] W. L. Briggs, S. F. McCormick, *et al.*, *A multigrid tutorial*, vol. 72. Siam, 2000.

[7] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of computation*, vol. 31, no. 138, pp. 333–390, 1977.

[8] H. M. Moya-Cessa and F. Soto-Eguibar, *Differential equations: an operational approach*. Rinton Press, Incorporated, 2011.

[9] R. S. Varga, *Geršgorin and his circles*, vol. 36. Springer Science & Business Media, 2010.

[10] R. Courant, K. Friedrichs, and H. Lewy, "On the partial difference equations of mathematical physics," *IBM journal of Research and Development*, vol. 11, no. 2, pp. 215–234, 1967.