Fast Explicit Methods for PDE-Based Image Analysis

Dissertation
zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Sven Grewenig

Saarbrücken

2013

Tag des Kolloquiums:	29.05.2013
Dekan:	Prof. Dr. Mark Groves
Ausschussvorsitzender:	Prof. Dr. Matthias Hein
Berichterstatter:	Prof. Dr. Joachim Weickert
	Prof. Dr. Tobias Preußer
Protokollführer:	Dr. Christian Schmaltz

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den

Sven Grewenig

Kurzzusammenfassung

Diese Arbeit stellt ein neues schnelles explizites Diffusionsschema (FED) vor, das wechselnde Zeitschrittweiten nutzt, um parabolische Probleme effizient zu lösen. Für die effiziente Lösung von elliptischen Problemen bauen wir FED und auch einen auf FED basierenden Löser für lineare Gleichungssysteme, das schnelle Jacobi-Verfahren, in ein kaskadiertes Mehrgitterverfahren ein. Sowohl FED als auch das schnelle Jacobi-Verfahren sind sehr gut für einfache parallele Implementierungen geeignet.

FED wird durch eine Zerlegung von aus der Signalverarbeitung bekannten Mittelwertfiltern im Sinne von expliziten Schemata für eindimensionale lineare Diffusionsprobleme motiviert. Die entsprechenden Zeitschrittweiten verletzen die Stabilitätsbedingungen gewöhnlicher expliziter Schemata in bis zu 50 Prozent der Fälle. Anders als das bekannte Superzeitschrittverfahren (STS) benötigt es dank der auf der Signalverarbeitung basierenden Herleitung keinen zusätzlichen Dämpfungsparameter.

Zur Verbesserung der numerischen Stabilität präsentieren wir ein FED Runge-Kutta Schema und eine semi-iterative Version des schnellen Jacobi-Verfahrens. Im Bezug auf nichtlineare Probleme erlauben diese Verfahren Updatestrategien, die sowohl die Genauigkeit als auch die Effizienz weiter verbessern können.

Schließlich zeigen wir, dass FED mit Extrapolationsverfahren kombiniert werden kann. Dies liefert stabile explizite Schemata höherer Ordnung zur Lösung parabolischer Probleme.

Short Abstract

This thesis introduces a novel fast explicit diffusion (FED) scheme that uses varying time step sizes to efficiently solve parabolic problems. For the efficient solution of elliptic problems, we embed FED as well as an FED-based solver for linear systems of equations, the Fast-Jacobi method, in a cascadic coarse-to-fine approach. Both FED and Fast-Jacobi are very well-suited for simple, parallel implementations.

FED is motivated from a decomposition of box filters, which is a well-known signal processing tool, in terms of explicit schemes for one-dimensional linear diffusion problems. The corresponding time step sizes violate the stability restrictions of usual explicit schemes in up to 50 percent of all cases. Unlike the known Super Time Stepping (STS) approach, it does not require an additional damping parameter due to the derivation based on signal processing.

To improve the numerical stability, we present an FED Runge-Kutta scheme and a semi-iterative version of the Fast-Jacobi method. Regarding nonlinear problems, these approaches allow update strategies that can further improve both the accuracy and efficiency.

Finally, we show that FED can be combined with extrapolation methods. This yields stable, higher order explicit schemes for the solution of parabolic problems.

Abstract

The main topic of this thesis is the efficient numerical solution of partial differential equations (PDEs) in the context of image analysis. Many techniques in this field use diffusion processes that require the solution of linear or nonlinear heat equations. With an increasing amount of data, this can become very cumbersome.

To this end, we introduce a novel Fast Explicit Diffusion (FED) scheme. It is as simple as an usual explicit scheme, but uses cycles with varying time step sizes that violate stability restrictions in up to 50 percent of all cases. Compared to the related and known Super Time Stepping (STS) method, the time step sizes do not have to be modified by an additional damping parameter. However, the FED time steps can be very large and thus FED can reach large stopping times with a relatively small number of steps. More precisely, in contrast to the linear dependence between the stopping time and the number of time steps for usual explicit schemes with constant time step size, the stopping time of an FED cycle depends quadratically on the number of explicit steps.

For elliptic problems, we propose to embed FED in a cascadic coarse-to-fine approach. Moreover, we introduce a Fast-Jacobi method with varying relaxation parameters that are based on the FED time steps. This method can also be used in combination with a cascadic approach.

FED schemes are derived with the help of an interesting connection between linear, symmetric filters and one-dimensional explicit linear diffusion schemes. In this context, we prove a theorem stating that each linear, symmetric filter can be interpreted as an explicit linear diffusion scheme, where the time step sizes may vary. Thus, we can relate the STS approach to a specific linear filter that has problems with high frequency components such as noise, and derive the FED scheme from the well-known box filter kernel.

Since the box filter has much fewer problems concerning higher frequencies, the FED scheme does not require additional parameters improving the behaviour with respect to such frequency components. The transfer of FED to arbitrary diffusion problems is straightforward. One only has to take care of the corresponding time step size limits. The implementation is very easy and requires marginal additional effort compared to usual explicit schemes. Moreover, it is very well suited for parallel computing. However, because of numerical rounding errors, we have to apply techniques for rearranging the sequence of the time step sizes. In this thesis, we consider so-called κ-cycles that have been already proposed in the context of STS, the rearrangement using Leja ordering, and another method proposed by

Lebedev and Finogenov. These approaches are known to work well for rearrangements of varying parameters.

Fortunately, the problem regarding this rearrangement can be solved by rewriting FED in terms of a recursive box filter scheme or, in other words, a Runge-Kutta scheme. To this end, we present a recurrence relation for box filters that avoids the decomposition into explicit linear diffusion steps. On the one hand this requires more computational effort, but on the other hand provides robustness against numerical rounding errors without any requirement for rearrangements. Furthermore, in contrast to FED, it is possible to perform updates within a cycle, which can improve both the accuracy and the efficiency. Such inner updates are also useful for efficient predictor-corrector strategies, because they require good approximations with respect to the predictor.

Finally, we combine FED schemes with the so-called Richardson extrapolation that allows the construction of higher order schemes. We show that the resulting FED extrapolation scheme is stable, whereas the stability of an STS extrapolation method depends on the above mentioned damping parameter.

Our numerical experiments cover comparisons with popular solvers for nonlinear diffusion problems such as additive operator splitting (AOS) for isotropic or semi-implicit schemes for anisotropic processes, and demonstrate the state-of-the-art performance of FED. For the elliptic case it turns out that the Fast-Jacobi method can be more efficient than a parabolic FED approach. Furthermore, we discuss the advantages of having no damping parameter anymore. Overall, these experiments shall illustrate the universality of the proposed methods in the context of PDE-based image analysis.

Acknowledgements

At this point, I want to express my thanks to all the people who have contributed to this PhD thesis. Without them, it would have been impossible to complete my work.

First of all, I would like to thank Prof. Dr. Joachim Weickert for supervising my thesis and giving me the opportunity to do my PhD in his group. Furthermore, my thanks go to the cluster of excellence “Multimodal Computing and Interaction” and the German Research Foundation (DFG) for partially funding my PhD with the project We 2602/7-1 and a Gottfried Wilhelm Leibniz Prize We 2602/9-1. I also like to thank Prof. Dr. Tobias Preußer for agreeing to become an external reviewer.

My very special thanks go to the whole Mathematical Image Analysis (MIA) group, including all former members, for interesting scientific discussions and providing a great atmosphere that made me enjoy working here. Without this great team spirit, it would have been not possible to win the “DAGM Main Prize” and to finish my PhD. In particular, I would like to thank Prof. Dr. Andrés Bruhn, Oliver Demetz, Dr. Pascal Gwosdek, Kai-Uwe Hagenburg, Markus Mainberger, Christopher Schroers, Dr. Simon Setzer and Dr. Henning Zimmer for fruitful discussions on FED and assistance with respect to implementation issues.

Regarding the administrative problems, I am very grateful to our secretary Ellen Wintringer and the system administrator Marcus Hargarter.

Last but not least, I wish to thank my whole family, in particular my parents Anne and Rolf, my brother Marc and my girlfriend Jennifer who supported me during my PhD studies and always believe in me.

Saarbrücken, Mai 2013 Sven Grewenig

Contents

Chapter 1

Introduction

There are things known
and things unknown,
and in between are the doors.
Jim Morrison (The Doors)

1.1 Motivation

In recent years, digital image processing has become, and still becomes, increasingly important for industrial as well as medical applications, and of course in our daily life. The main task of image processing is the quality enhancement of images such that they are more useful to a human observer or a computer vision system. In this context, enhancement means for instance the removal of noise, smoothing an image or the filling-in of missing (corrupted) data.

If we consider, for example, the development of digital compact cameras, a maximum resolution larger than 10 megapixels is today's standard, which means that one has to store and process lots of data. Thus, the software used for the enhancement of the recorded images requires a better and better hardware. Besides these higher requirements with respect to the camera chips, it is essential to develop fast and clever algorithms that provide good filtering results within a reasonable time. They should be simple to implement and well-suited for parallel computing, due to the widespread availability of low cost parallel computing hardware such as GPUs.

Two important classes of methods for image processing that we are going to consider in this thesis are linear filters and diffusion filtering. They both enhance images by computing weighted averages of grey or colour values. In contrast to linear filters, diffusion filters are based on partial differential equations (PDEs) and can be adapted to the local image structure. Thus, they are able to preserve e.g. image edges that are very important for our visual system. However, these filter classes are not disjoint, since the convolution with the Gaussian function is a linear filter and at the same time equivalent to a diffusion process.

Besides the direct calculation of the above mentioned Gaussian convolution, it is also popular to iteratively apply linear filters as an approximation. A well-known example for such an iterative application is the so-called box or mean filter that simply computes mean values of symmetric neighbourhoods with a fixed size. It has nice properties that allow a fast computation independent of the size of the neighbourhood, even in higher dimensions. On the other hand, multi-dimensional diffusion filtering can get very expensive, since large linear systems of equations with band matrices, whose bandwidths increase with the number of dimensions, must be solved. Furthermore, if one wants to avoid linear systems, the alternative, i.e. explicit numerical schemes, suffers from smaller time step size restrictions. Therefore, explicit schemes can be less efficient. However, they are much easier to implement and are well-suited for parallel computing. Moreover, one does not have to take care of iterative solvers for linear systems of equations and their corresponding stopping criteria or other parameters, which makes explicit approaches less parameter-sensitive.

As we have mentioned above, box filtering is easy to implement and represents a very efficient linear filter. Actually, we are looking for the same in the case of PDE-based problems, i.e. easy and efficient numerical schemes. Concerning linear diffusion or equivalently Gaussian convolution, we can use an iterative application of the box filter. On the other hand, the numerical solution of linear diffusion processes can be seen as an iterative application of a numerical scheme with a fixed time step size. This gives rise to the questions whether a box filter or, in general, a linear filter is in some sense related to numerical diffusion schemes, and if yes, how can we build the bridge between them.

The first goal of the thesis is it to build such a bridge and shed some light on this connection. With the help of this, we are going to encounter a special class of explicit diffusion schemes, introduce a novel scheme that is based on discrete box filtering and has some advantages compared to existing methods of the class. In this context, the second goal is to extend the novel numerical scheme to solve arbitrary parabolic problems in PDE-based image analysis. Since many applications in image analysis like e.g.

inpainting require the solution of elliptic problems, we also look for methods that allow the efficient treatment of these tasks.

The following sections will give a short overview about diffusion filtering and linear filters. Furthermore, we deal with the representation of continuous linear filters in terms of differential operators. At the end of this introductory chapter, we conclude with an outline.

1.2 Diffusion Filtering

Diffusion filtering is widely used in digital image processing and a well-understood technique for image simplification, denoising or interpolation. Actually, it is a physical process that equilibrates concentration differences and at the same time preserves mass.

One main ingredient is *Fick's law*, which states that the flux j depends on the concentration gradient $\boldsymbol{\nabla} u$ in terms of

$$\boldsymbol{j} \;=\; -\boldsymbol{D}\,\boldsymbol{\nabla} u\;, \tag{1.1}$$

where the so-called *diffusion tensor* $\boldsymbol{D}$ describes the relation between j and the spatial gradient $\boldsymbol{\nabla} u$. It is a symmetric positive definite matrix. We distinguish between the *isotropic* case, where j and $\boldsymbol{\nabla} u$ are parallel, and the *anisotropic* one without this parallelism. Note that in the isotropic case, $\boldsymbol{D}$ can be replaced by a positive real-valued *diffusivity*.

To guarantee the preservation of mass, we also require the well-known *continuity equation*

$$\partial_t u \;=\; -\operatorname{div} \boldsymbol{j}\;, \tag{1.2}$$

with the time variable t. The divergence on the right hand side is a spatial operator. Putting Fick's law and the continuity equation together yields the *diffusion equation*

$$\partial_t u \;=\; \operatorname{div}\left(\boldsymbol{D}\,\boldsymbol{\nabla} u\right)\,. \tag{1.3}$$

In image processing, we identify the concentration with grey or colour values. If the diffusion tensor $\boldsymbol{D}$ is constant over the whole image domain, the corresponding process is called *homogeneous* diffusion. In the case of a space-dependent tensor, we speak of *inhomogeneous*. *Nonlinear diffusion* processes require a diffusion tensor that depends on the evolving image itself. However, at first, we want to discuss *linear diffusion*, where the diffusion tensor does not depend on the evolving image. Since we present just a short overview, we recommend e.g. [147] for a more detailed discussion of diffusion filters including the Gaussian scale space [78, 79, 152, 153, 158].

1.2.1 Linear Diffusion

One of the simplest methods for smoothing images is linear diffusion filtering. Here, the diffusion tensor is the identity matrix, which means $\boldsymbol{D} = \boldsymbol{I}$. Eq. (1.3) then simplifies to the linear diffusion equation

$$\partial_t u \;=\; \operatorname{div}(\boldsymbol{\nabla} u) \;=\; \Delta u \;, \tag{1.4}$$

with the spatial Laplacian Δ. If our initial data is given by f, i.e. we assume $u(\boldsymbol{x},0) = f(\boldsymbol{x})$, then the solution u of the linear diffusion process is (see e.g. [73])

$$u(\boldsymbol{x},t) \;=\; \begin{cases} f(\boldsymbol{x}) & ,\; t \;=\; 0 \\ (G_{\sqrt{2t}} * f)(\boldsymbol{x}) & ,\; t \;>\; 0 \;, \end{cases} \tag{1.5}$$

where G_σ denotes the d-dimensional Gaussian with the standard deviation $\sigma > 0$,

$$G_\sigma(\boldsymbol{x}) \;:=\; (2\pi\sigma^2)^{-\frac{d}{2}} \cdot \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma^2}\right) \;, \tag{1.6}$$

and the symbol $*$ means the continuous convolution

$$(g * h)(\boldsymbol{x}) \;=\; \int_{\mathbb{R}^d} g(\boldsymbol{x}-\boldsymbol{y}) \cdot h(\boldsymbol{y}) \; d\boldsymbol{y} \;. \tag{1.7}$$

Note that the stopping time T of the diffusion process is related to the standard deviation σ of the Gaussian via

$$T \;=\; \frac{1}{2}\sigma^2 \;. \tag{1.8}$$

The preceding theory is completely continuous. However, images or data are given in a discrete form. To this end, we have to discretise the diffusion process and apply numerical schemes.

Basic Numerical Schemes

Using a spatial discretisation, i.e. the Laplacian is replaced by a discrete version, transforms the partial differential equation (PDE) (1.4) into a time-continuous system of ordinary differential equations (ODEs)

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{A}\boldsymbol{u} \;, \tag{1.9}$$

where $\boldsymbol{u} \in \mathbb{R}^N$ is the spatial discretisation of u and the symmetric negative semi-definite matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ represents the discrete Laplacian. This is the so-called *method of lines* [116, 120], which we will discuss later in more detail. If the original image is given by $\boldsymbol{u}^0 \in \mathbb{R}^N$, the exact solution of Eq. (1.9) corresponds to

$$\boldsymbol{u}(t) \;=\; \exp(t \cdot \boldsymbol{A})\, \boldsymbol{u}^0 \;, \tag{1.10}$$

with the matrix exponential $\exp(\cdot)$.

To approximate this solution, one often uses first order explicit or implicit schemes. In both cases, the left hand side of Eq. (1.9) is discretised by a first order finite difference $\frac{\boldsymbol{u}^{k+1}-\boldsymbol{u}^k}{\tau}$ with the approximation $\boldsymbol{u}^m \approx \boldsymbol{u}(m \cdot \tau)$ and a time step size $\tau > 0$. If the right hand side is approximated by $\boldsymbol{A}\boldsymbol{u}^k$, this yields the explicit finite difference scheme

$$\boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k \,+\, \tau \cdot \boldsymbol{A}\boldsymbol{u}^k \;=\; (\boldsymbol{I} \,+\, \tau\, \boldsymbol{A})\, \boldsymbol{u}^k \;, \tag{1.11}$$

where τ is limited with respect to the largest modulus of the eigenvalues of $\boldsymbol{A}$. Thus, the explicit scheme corresponds to a simple matrix-vector multiplication. Using $\boldsymbol{A}\boldsymbol{u}^{k+1}$ instead yields the implicit method

$$(\boldsymbol{I} \,-\, \tau\, \boldsymbol{A})\, \boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k \;. \tag{1.12}$$

It is stable for all time step sizes $\tau > 0$, but requires the solution of a linear system.

1.2.2 Nonlinear Isotropic Diffusion

The main disadvantage of linear diffusion filtering is the delocalisation and blurring of image edges that are important for human vision. To this end, Perona and Malik have proposed a nonlinear diffusion method that avoids these problems [105, 106]. Their proposed inhomogeneous process reduces the diffusivity at locations that have a large likelihood to be image edges. This likelihood depends on the magnitude of the image gradient $|\boldsymbol{\nabla} u|^2$. The larger this magnitude is, the more likely is the existence of an edge at the corresponding location. The underlying diffusion equation uses a diagonal tensor $\boldsymbol{D} = g\left(|\boldsymbol{\nabla} u|^2\right) \cdot \boldsymbol{I}$ with a diffusivity function such as

$$g\left(s^2\right) \;=\; \frac{1}{1 \,+\, \frac{s^2}{\lambda^2}} \qquad (\lambda \,>\, 0) \;. \tag{1.13}$$

Thus, they consider

$$\partial_t u \;=\; \operatorname{div}\left(g\left(|\boldsymbol{\nabla} u|^2\right)\, \boldsymbol{\nabla} u\right) \;. \tag{1.14}$$

The parameter $\lambda > 0$ is called *contrast parameter*. It separates smoothing *forward diffusion* from edge-enhancing *backward diffusion*. The resulting forward-backward diffusion behaviour can also appear for other diffusivities than (1.13). An overview of several common diffusivities for the nonlinear isotropic Perona-Malik model can be found e.g. in [11].

Unfortunately, the Perona-Malik filter has theoretical problems with respect to well-posedness. For the continuous setting, Kichenassamy has shown that it is not well-posed [81]. However, a former result discovered by Höllig has already indicated problems with forward-backward diffusion processes [75]: He has found a forward-backward process (different from Perona-Malik) that has infinitely many solutions.

To overcome such problems, Catté et al. [25] have proposed to replace the diffusivity $g\left(|\boldsymbol{\nabla} u|^2\right)$ by a Gaussian-smoothed version $g\left(|\boldsymbol{\nabla} u_\sigma|^2\right)$ with $u_\sigma := G_\sigma * u$:

$$\partial_t u \;=\; \operatorname{div}\left(g\left(|\boldsymbol{\nabla} u_\sigma|^2\right)\,\boldsymbol{\nabla} u\right)\;. \tag{1.15}$$

They have proved existence, uniqueness and regularity of a solution for $\sigma > 0$ [25]. Besides these nice theoretical results, the Gaussian convolution yields more robustness against noise. However, one should mention that spatial discretisations of the original Perona-Malik filter also work like a regularisation and give well-posedness, which has been shown by Weickert and Benhamouda [151].

Numerical Schemes

The construction of schemes for the solution of the regularised nonlinear diffusion equation (1.15) is actually very similar to the linear case. We also perform a spatial discretisation yielding the ODE system

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{A}(\boldsymbol{u})\,\boldsymbol{u}\;. \tag{1.16}$$

If we just replace $\boldsymbol{A}$ in the linear schemes by $\boldsymbol{A}\left(\boldsymbol{u}^k\right)$ for each time step, we get the explicit scheme

$$\boldsymbol{u}^{k+1} \;=\; \left(\boldsymbol{I} \,+\, \tau\,\boldsymbol{A}\left(\boldsymbol{u}^k\right)\right)\boldsymbol{u}^k\;, \tag{1.17}$$

as well as the semi-implicit method

$$\left(\boldsymbol{I} \,-\, \tau\,\boldsymbol{A}\left(\boldsymbol{u}^k\right)\right)\boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k\;. \tag{1.18}$$

For multi-dimensional problems, the semi-implicit method requires the solution of huge linear systems with more than three non-zero entries per row. In this case, the popular *Thomas algorithm* (tridiagonal matrix algorithm) [131], whose complexity is linear in the number of unknowns, can not be applied. Thus, so-called *additive operator splitting (AOS) schemes* have been proposed [93, 155]. The main idea is to decompose the multi-dimensional problem into several 1-D problems, which yields linear systems of equations with tridiagonal matrices. As mentioned above, these systems can be solved very efficiently with the help of the Thomas algorithm. Concerning a d-dimensional problem, the inverse matrix of $(\boldsymbol{I} - \tau \boldsymbol{A})$ is replaced by a first order Taylor approximation:

$$\left(\boldsymbol{I} - \tau \boldsymbol{A}\left(\boldsymbol{u}^k\right)\right)^{-1} \approx \frac{1}{d} \sum_{\ell=1}^{d} \left(\boldsymbol{I} - d \cdot \tau \boldsymbol{A}_\ell\left(\boldsymbol{u}^k\right)\right)^{-1} , \tag{1.19}$$

where the matrices $\boldsymbol{A}_\ell(\cdot)$ represent the 1-D problems and $\boldsymbol{A}(\cdot) = \sum_{\ell=1}^{d} \boldsymbol{A}_\ell(\cdot)$. The AOS scheme is more efficient than explicit and usual semi-implicit approaches. Furthermore, it works well together with parallelisation strategies [17, 149, 156]. However, it induces a splitting error that increases with the magnitude of the time step size τ.

So far, we have only discussed isotropic diffusion processes, where the flux is parallel to the gradient. If one wants to bias the flux towards the orientation of interesting features, the introduction of an anisotropic model is necessary.

1.2.3 Nonlinear Anisotropic Diffusion

Nonlinear anisotropic diffusion filters have been proposed by Cottet and Germain [34] as well as Weickert [145]. Since the flux does not have to be parallel to the image gradient, these filters allow not only to steer the amount of diffusion, but also its direction. Thus, anisotropic diffusion filters can yield a better edge enhancement than their isotropic counterparts, and may also be employed to enhance flow-like structures.

Now we review two specific 2-D anisotropic diffusion filters that we are going to use for the numerical experiments in this thesis.

Edge-Enhancing Diffusion (EED)

Edge-enhancing anisotropic diffusion [146] inhibits diffusion across edges and instead prefers smoothing within the image regions. It follows the evolution equation

$$\partial_t u = \operatorname{div}\left(\boldsymbol{D}\left(\boldsymbol{\nabla} u_\sigma\right) \boldsymbol{\nabla} u\right) , \tag{1.20}$$

where $\boldsymbol{D} \in \mathbb{R}^{2\times 2}$ is the symmetric positive definite diffusion tensor, and u_σ is the image u convolved with a 2-D Gaussian of standard deviation $\sigma > 0$. Since we want to inhibit diffusion across edges, i.e. parallel to the gradient $\boldsymbol{\nabla} u_\sigma$, the corresponding eigenvalue should be very small at image edges, which means that we consider the value $g\left(|\boldsymbol{\nabla} u_\sigma|^2\right)$. For the diffusion along the image edge, i.e. perpendicular to the image gradient, we assume the eigenvalue to be equal to 1. Thus, the diffusion tensor can be written as

$$\boldsymbol{D}\left(\boldsymbol{\nabla} u_\sigma\right) \;=\; g\left(|\boldsymbol{\nabla} u_\sigma|^2\right) \cdot \frac{\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top}{|\boldsymbol{\nabla} u_\sigma|^2} + 1 \cdot \frac{\boldsymbol{\nabla} u_\sigma^\perp \boldsymbol{\nabla} u_\sigma^{\perp\,\top}}{|\boldsymbol{\nabla} u_\sigma^\perp|^2}\;, \tag{1.21}$$

where $\cdot^\top$ means the usual matrix transposition and $\binom{a}{b}^\perp := \binom{-b}{a}$, i.e. the corresponding orthogonal vector. In fact, besides the regularisation, the Gaussian convolution is important to achieve anisotropy. Without the convolution, we would have

$$\begin{aligned}
\boldsymbol{D}\left(\boldsymbol{\nabla} u\right) \boldsymbol{\nabla} u \;&=\; g\left(|\boldsymbol{\nabla} u|^2\right) \cdot \frac{\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top}{|\boldsymbol{\nabla} u|^2} \boldsymbol{\nabla} u \;+\; 1 \cdot \frac{\boldsymbol{\nabla} u^\perp \boldsymbol{\nabla} u^{\perp\,\top}}{|\boldsymbol{\nabla} u^\perp|^2} \boldsymbol{\nabla} u \\
&=\; g\left(|\boldsymbol{\nabla} u|^2\right)\, \boldsymbol{\nabla} u \cdot \underbrace{\frac{\boldsymbol{\nabla} u^\top \boldsymbol{\nabla} u}{|\boldsymbol{\nabla} u|^2}}_{=1} + \boldsymbol{\nabla} u^\perp \cdot \underbrace{\frac{\boldsymbol{\nabla} u^{\perp\,\top} \boldsymbol{\nabla} u}{|\boldsymbol{\nabla} u^\perp|^2}}_{=0} \\
&=\; g\left(|\boldsymbol{\nabla} u|^2\right)\, \boldsymbol{\nabla} u\;,
\end{aligned} \tag{1.22}$$

which corresponds to an isotropic diffusion process.

In our experiments we shall use EED with the so-called *Charbonnier diffusivity* function [29]

$$g\left(s^2\right) \;=\; \left(1 + \tfrac{s^2}{\lambda^2}\right)^{-1/2}\;. \tag{1.23}$$

It has proven to be highly useful for image interpolation purposes such as the compression method in [54].

Coherence-Enhancing Diffusion (CED)

Coherence-enhancing diffusion filtering [148] enhances line- and flow-like structures. Its diffusion tensor has the same eigenvectors as the so-called *structure tensor* [51]

$$\boldsymbol{J}_\rho\left(\boldsymbol{\nabla} u_\sigma\right) := G_\rho * \left(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top\right)\;, \tag{1.24}$$

with the 2-D Gaussian G_ρ of standard deviation ρ, and its eigenvalues are given by

$$\lambda_1 \;:=\; \alpha \tag{1.25}$$

$$\lambda_2 \;:=\; \begin{cases} \alpha, & \text{if } \mu_1 = \mu_2 \,, \\ \alpha + (1-\alpha)\exp\left(\frac{-\lambda}{(\mu_1-\mu_2)^2}\right), & \text{else} \end{cases} \,, \tag{1.26}$$

where μ_1 and μ_2 are the eigenvalues of the structure tensor. We assume that they satisfy $\mu_1 \geq \mu_2$. For further details we refer to [147, 148]. As a space discretisation for CED, we have used the one in [157], which has low dissipativity.

Numerical Schemes

The system of ODEs resulting from the spatial discretisation is very similar to the isotropic Eq. (1.16). Since the averaging neighbourhood is larger due to the mixed derivatives, the system matrix $\boldsymbol{A}(\boldsymbol{u})$ is less sparse. Unfortunately, there is no efficient full operator splitting like AOS in the general anisotropic case. Instead, there has been a number of proposals for numerical schemes for anisotropic diffusion processes (cf. e.g. [31, 41, 107, 108, 154, 157]), although probably the two most popular ways to implement anisotropic diffusion filters are explicit and semi-implicit finite difference schemes as described above for the isotropic case.

However, in contrast to the isotropic case, we use a modified semi-implicit scheme that is numerically more robust [150]. To this end, we introduce the vector $\boldsymbol{v} := \frac{\boldsymbol{u}^{k+1}-\boldsymbol{u}^k}{\tau}$ and rewrite the scheme

$$\left(\boldsymbol{I} - \tau \boldsymbol{A}\left(\boldsymbol{u}^k\right)\right) \boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k \tag{1.27}$$

by means of the new linear system

$$\left(\boldsymbol{I} - \tau \boldsymbol{A}\left(\boldsymbol{u}^k\right)\right) \boldsymbol{v} \;=\; \boldsymbol{A}\left(\boldsymbol{u}^k\right) \boldsymbol{u}^k \,. \tag{1.28}$$

Following the definition of $\boldsymbol{v}$, the solution $\boldsymbol{u}^{k+1}$ is then given by $\boldsymbol{u}^k + \tau \boldsymbol{v}$.

1.3 Linear Filtering

In this section we want to consider linear filters $\mathbb{L}_k$ that are convolutions with appropriate kernel functions $k : \mathbb{R}^d \to \mathbb{R}$. However, the definition of the function k indicates that we first restrict ourselves to the continuous setting. Given an arbitrary function $f : \mathbb{R}^d \to \mathbb{R}$, we compute a filtered version $\mathbb{L}_k[f]$ via

$$\mathbb{L}_k[f](\boldsymbol{x}) \;:=\; (k * f)(\boldsymbol{x}) \;=\; \int_{\mathbb{R}^d} k(\boldsymbol{x}-\boldsymbol{y}) \cdot f(\boldsymbol{y})\, d\boldsymbol{y}\;. \tag{1.29}$$

Note that we have already presented a special case of a linear filter in the last section: Linear diffusion filtering corresponds to a convolution with the Gaussian from Eq. (1.6), i.e. $k = G_\sigma$ [73].

Actually, continuous convolutions can be replaced by so-called *pseudo-differential operators*, which means that the continuous linear filtering is equivalent to the application of a pseudo-differential operator [38]. In order to clarify this, we need some technical details like e.g. the popular *Fourier transform*. It has many nice properties that we shortly discuss in the following.

1.3.1 Fourier Transform and Convolution Theorem

At first, we want to introduce some function spaces, namely the *Schwartz space* (see e.g. [129, 130]), on which the Fourier transform can be declared, and the *space of p-integrable functions*. Therefore, we need the multi-index notation:

Definition 1.1. *For a d-dimensional multi-index* $\boldsymbol{\alpha} = (\alpha_1, \ldots \alpha_d) \in \mathbb{N}_0^d$ *and* $\boldsymbol{x} = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$ *we define the* ***power***

$$\boldsymbol{x}^{\boldsymbol{\alpha}} \;:=\; x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \cdots \cdot x_d^{\alpha_d}\;, \tag{1.30}$$

and the ***higher order partial derivative***

$$\mathcal{D}^{\boldsymbol{\alpha}} \;:=\; \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial_{x_1}^{\alpha_1}\,\partial_{x_2}^{\alpha_2}\,\ldots\,\partial_{x_d}^{\alpha_d}}\;, \tag{1.31}$$

with $|\boldsymbol{\alpha}| := \sum_{i=1}^{d} \alpha_i\,.$

With the help of this notation, we are able to define the above mentioned function spaces:

Definition 1.2. *The* ***Schwartz space*** *or the* ***space of rapidly decreasing functions*** $\mathcal{S} = \mathcal{S}\left(\mathbb{R}^d\right)$ *on* $\mathbb{R}^d$ *is the function space*

$$\mathcal{S} = \left\{ f \in C^\infty\left(\mathbb{R}^d\right) \mid \|f\|_{\boldsymbol{\alpha},\boldsymbol{\beta}} := \sup_{\boldsymbol{x}\in\mathbb{R}^d} |\boldsymbol{x}^{\boldsymbol{\alpha}} \cdot \mathcal{D}^{\boldsymbol{\beta}} f(\boldsymbol{x})| < \infty \ \ \forall\, \boldsymbol{\alpha}, \boldsymbol{\beta} \right\}, \tag{1.32}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta}$ *are multi-indices and* $C^\infty\left(\mathbb{R}^d\right)$ *the set of smooth functions from* $\mathbb{R}^d$ *to* $\mathbb{C}$.
Furthermore we define the ***space of p-integrable functions*** *on* $\mathbb{R}^d$ *for* $p \geq 1$,

$$\mathcal{L}_p\left(\mathbb{R}^d\right) = \left\{ f : \mathbb{R}^d \to \mathbb{C} \ \mid \ \|f\|_p := \left(\int_{\mathbb{R}^d} |f(\boldsymbol{x})|^p \, d\boldsymbol{x} \right)^{\frac{1}{p}} < \infty \right\}. \tag{1.33}$$

Such spaces are very important, if one wants to define operators like for instance integral transformations with a kernel Ψ,

$$\mathcal{A}[f](\boldsymbol{x}) = \int\limits_E \Psi(\boldsymbol{x}, \boldsymbol{y}) \cdot f(\boldsymbol{y}) \, d\boldsymbol{y} \quad , \ \boldsymbol{x} \in G \subset \mathbb{R}^d \,, \tag{1.34}$$

where $E \subset \mathbb{R}^d$. It is clear that the domain of the operator $\mathcal{A}$ depends on the kernel $\Psi : G \times E \to \mathbb{R}$. In the case of a bounded kernel $|\Psi| \leq c$, the integral in Eq. (1.34) exists for $f \in \mathcal{L}_1$. A very popular integral transform with a bounded kernel is the Fourier transform (see e.g. [13, 20, 159] for further details):

Definition 1.3. *Let* $f \in \mathcal{S}\left(\mathbb{R}^d\right)$ *or* $f \in \mathcal{L}_1\left(\mathbb{R}^d\right)$ *and* $\boldsymbol{\xi} \in \mathbb{R}^d$. *Then*

$$\mathcal{F}[f](\boldsymbol{\xi}) = \hat{f}(\boldsymbol{\xi}) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} f(\boldsymbol{x}) \cdot \exp(-i\, \boldsymbol{x}^T \boldsymbol{\xi}) \, d\boldsymbol{x} \tag{1.35}$$

is the ***Fourier transform*** *of* f.
The ***inverse Fourier transform*** *is given by*

$$f(\boldsymbol{x}) = \mathcal{F}^{-1}\big[\hat{f}\big](\boldsymbol{x}) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \hat{f}(\boldsymbol{\xi}) \cdot \exp(i\, \boldsymbol{x}^T \boldsymbol{\xi}) \, d\boldsymbol{\xi} \,. \tag{1.36}$$

The Fourier transform decomposes the function $f(\boldsymbol{x})$ into its frequency components $\left\{\hat{f}(\boldsymbol{\xi}) \mid \boldsymbol{\xi} \in \mathbb{R}^d\right\}$. In the multi-dimensional case, the frequency $\boldsymbol{\xi} \in \mathbb{R}^d \backslash \{\mathbf{0}\}$ can be interpreted as a normal vector of a hyperplane on which the kernel function $w_{\boldsymbol{\xi}}(\boldsymbol{x}) = \exp(-i\, \boldsymbol{x}^T \boldsymbol{\xi})$ is constant. This can be seen as follows: If $\boldsymbol{x}$ is on a hyperplane with normal vector $\boldsymbol{\xi} \neq \mathbf{0}$, i.e. $\boldsymbol{x} = \alpha \cdot \boldsymbol{\xi} + \boldsymbol{\eta}$ with $\boldsymbol{\eta}^T \boldsymbol{\xi} = 0$, we have

$$w_{\boldsymbol{\xi}}(\boldsymbol{x}) = \exp(-i\,(\alpha\boldsymbol{\xi} + \boldsymbol{\eta})^T\boldsymbol{\xi}) = \exp(-i\,\alpha\,|\boldsymbol{\xi}|^2)\,. \tag{1.37}$$

For $\boldsymbol{\xi} = \mathbf{0}$, the Fourier transform simplifies to a scaled mean value of the function f:

$$\mathcal{F}[f](\mathbf{0}) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} f(\boldsymbol{x})\, d\boldsymbol{x}\,. \tag{1.38}$$

Since $\hat{f}(\boldsymbol{\xi}) \in \mathbb{C}$, we can express it in the (complex) polar coordinate system:

$$\hat{f}(\boldsymbol{\xi}) = |\hat{f}(\boldsymbol{\xi})| \cdot \exp(i\,\theta(\boldsymbol{\xi}))\,. \tag{1.39}$$

The magnitude $|\hat{f}(\boldsymbol{\xi})|$ denotes the so-called Fourier spectrum and $\theta(\boldsymbol{\xi})$ the phase angle. This Fourier spectrum describes the importance or influence of the frequency $\boldsymbol{\xi}$ for the function f. Hence, it is often used for visualisation purposes.

At this point, we want to compute the Fourier transform of a 1-D Gaussian and another popular function that is also very important for signal as well as image processing applications, namely a box function with finite support.

Examples

(1) 1-D Gaussian with $\sigma > 0$, $G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \in \mathcal{S}(\mathbb{R})$:

$$\begin{aligned}
\mathcal{F}[G_\sigma](\xi) &= \frac{1}{2\pi\sigma} \int_{\mathbb{R}} \exp\left(-\tfrac{x^2}{2\sigma^2}\right) \cdot \exp(-i\,x\xi)\, dx \\
&= \frac{1}{2\pi\sigma} \exp\left(-\tfrac{\sigma^2\xi^2}{2}\right) \int_{\mathbb{R}} \exp\left(-\tfrac{(x+i\,\sigma^2\xi)^2}{2\sigma^2}\right) dx \\
&= \frac{1}{2\pi\sigma} \exp\left(-\tfrac{\sigma^2\xi^2}{2}\right) \underbrace{\int_{\mathbb{R}} \exp\left(-\tfrac{x^2}{2\sigma^2}\right) dx}_{=\sqrt{2\pi}\sigma} \\
&= \frac{1}{\sqrt{2\pi}} \exp\left(-\tfrac{\sigma^2\xi^2}{2}\right) = \sigma^{-1} \cdot G_{\sigma^{-1}}(\xi)\,.
\end{aligned} \tag{1.40}$$

(2) Normalised 1-D box function (or characteristic function) with the radius $r > 0$, $b_r(x) = \frac{1}{2r} \cdot \chi_{[-r,r]}(x) \in L_1(\mathbb{R})$:

$$\begin{aligned}
\mathcal{F}[b_r](\xi) &= \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{2r} \int_{-r}^{r} \exp(-i\, x\xi)\, dx = \frac{1}{\sqrt{8\pi} r} \int_{-r}^{r} \cos(x\xi)\, dx \\
&= \frac{1}{\sqrt{2\pi} r} \int_{0}^{r} \cos(x\xi)\, dx = \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{r\xi} \sin(r\xi) \\
&= \frac{1}{\sqrt{2\pi}} \cdot \operatorname{sinc}(r\xi)\,, \qquad (1.41)
\end{aligned}$$

where we have used the definition of the sinc-function, i.e.

$$\operatorname{sinc}(x) = \frac{sin(x)}{x}\,. \qquad (1.42)$$

These two examples are very interesting from a theoretical point of view. The first one shows that the Fourier transform of a one-dimensional Gaussian remains a Gaussian, but with inverse standard deviation. This is also valid for higher dimensions. For the special case $\sigma = 1$, the Gaussian does not change at all. Hence, it is an eigenfunction of the Fourier transform with eigenvalue 1. The second example with the box function demonstrates that the Fourier transform of a function with finite support can have an infinite support.

Another important property is that derivatives of f in the spatial domain correspond to multiplications with polynomials in the Fourier domain:

Proposition 1.4 (Derivative Property of Fourier Transform). *Let $f \in \mathcal{S}\left(\mathbb{R}^d\right)$. Then we have*

$$\mathcal{F}[\mathcal{D}^\alpha f](\boldsymbol{\xi}) = i^{|\alpha|} \cdot \boldsymbol{\xi}^\alpha \cdot \mathcal{F}[f](\boldsymbol{\xi}) \qquad (1.43)$$

for all multi-indices $\alpha \in \mathbb{N}_0^d$.

This can be very useful for the solution of linear differential equations, since they correspond to polynomial equations in the Fourier domain. Having computed a solution of the polynomial equation, the application of the inverse transform yields the solution of the differential equation.

Besides this differentiation rule, one of the main results regarding the connection between operations in the spatial and Fourier domain is the well-known *convolution theorem*. It states that the spatial convolution of two functions corresponds to a pointwise multiplication of their Fourier transforms:

Theorem 1.5 (Convolution Theorem). *Let* $f, g \in \mathcal{S}\left(\mathbb{R}^d\right)$*. The Fourier transform of a convolution fulfils*

$$\mathcal{F}[f * g](\boldsymbol{\xi}) \;=\; (2\pi)^{\frac{d}{2}} \cdot \mathcal{F}[f](\boldsymbol{\xi}) \cdot \mathcal{F}[g](\boldsymbol{\xi}) \,. \tag{1.44}$$

Proof. With the substitution $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ we obtain

$$\begin{aligned}
\mathcal{F}[f * g](\boldsymbol{\xi}) \;&=\; (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} f(\boldsymbol{x} - \boldsymbol{y}) \cdot g(\boldsymbol{y}) \cdot \exp(-i\, \boldsymbol{x}^T \xi)\; d\boldsymbol{y}\; d\boldsymbol{x} \\
&=\; (2\pi)^{\frac{d}{2}} \cdot (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} f(\boldsymbol{z}) \cdot \exp(-i\, \boldsymbol{z}^T \boldsymbol{\xi})\; d\boldsymbol{z} \\
&\qquad \cdot (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} g(\boldsymbol{y}) \cdot \exp(-i\, \boldsymbol{y}^T \boldsymbol{\xi})\; d\boldsymbol{y} \\
&=\; (2\pi)^{\frac{d}{2}} \cdot \mathcal{F}[f](\boldsymbol{\xi}) \cdot \mathcal{F}[g](\boldsymbol{\xi}) \,.
\end{aligned} \tag{1.45}$$

□

Such a convolution theorem also exists in the discrete case and hence allows us to avoid expensive discrete convolutions with large filter kernels. If the Fourier transforms are known, one can use the cheap multiplication and apply the inverse Fourier transform to get the result of the convolution.

Now we have all necessary ingredients in order to represent convolutions as pseudo-differential operators.

1.3.2 Filter Kernels and Pseudo-Differential Operators

We have already mentioned that the convolution with e.g. a Gaussian kernel is an important smoothing operator in digital signal or image processing. It can improve the corresponding data by removing noise or artifacts. We distinguish mainly between three different basic types: Lowpass, highpass and bandpass filters.

Lowpass filters are designed as convolutions with averaging kernels. They attenuate high frequency components like noise. Typical examples are the above mentioned convolutions with a Gaussian or a box function.

In contrast to lowpass filters, *highpass filters* amplify high frequencies and attenuate low frequencies. The goal is to remove low-frequent background perturbations or to sharpen blurry structures. They can be easily constructed by computing the difference between the original data and a low-pass filtered version. Concerning image processing, an important class of highpass filters consists of derivative filters that can be used for detecting edges within an image.

To keep structures within a specific frequency band, one applies so-called *bandpass filters*. These filters are less important for image enhancement. Nevertheless, they can be useful for the extraction of interesting structures on certain scales. Their construction is possible by subtracting two lowpass filters.

The following thoughts are based on the work by Didas and Weickert [38]. Let us assume we have given a filter kernel k whose Fourier transform exists and want to convolve it with a function $f \in \mathcal{S}(\mathbb{R})$. If $\hat{k}$ is additionally analytic, then it has a power series representation

$$\hat{k}(\xi) \;=\; \sum_{m=0}^{\infty} a_m \, \xi^m \, . \tag{1.46}$$

Using this together with the convolution theorem, we get

$$\mathcal{F}[k * f](\xi) \;=\; \sqrt{2\pi} \cdot \sum_{m=0}^{\infty} a_m \, \xi^m \cdot \hat{f}(\xi) \, . \tag{1.47}$$

Considering Proposition 1.4, this can be seen as a sum of arbitrary order derivatives of f in the Fourier domain. The application of the inverse transform yields

$$k * f \;=\; \sqrt{2\pi} \cdot \mathcal{F}^{-1} \left[\sum_{m=0}^{\infty} a_m \, \xi^m \cdot \hat{f}(\xi) \right] \, . \tag{1.48}$$

Under the assumption of sufficient convergence conditions for the power series, we may interchange the sum and the inverse Fourier transform:

$$\mathcal{F}^{-1}\left[\sum_{m=0}^{\infty} a_m\, \xi^m \cdot \hat{f}(\xi)\right] \;=\; \sum_{m=0}^{\infty} a_m \cdot \mathcal{F}^{-1}\left[\xi^m \hat{f}(\xi)\right]$$
$$\;=\; \sum_{m=0}^{\infty} a_m \left(-i\,\frac{d}{dx}\right)^m f\,. \tag{1.49}$$

Using the pseudo-differential operator

$$\hat{k}\left(-i\,\frac{d}{dx}\right) \;:=\; \sum_{m=0}^{\infty} a_m \left(-i\,\frac{d}{dx}\right)^m, \tag{1.50}$$

the convolution with the kernel k, i.e. the application of the linear filter $\mathbb{L}_k[f]$, can finally be written as [38]

$$\mathbb{L}_k[f] \;=\; k * f \;=\; \sqrt{2\pi}\cdot \hat{k}\left(-i\,\frac{d}{dx}\right) f\,. \tag{1.51}$$

After the description of the general idea, we now want to show two examples of filter kernels: The first example is a Gaussian kernel, which means that we rewrite linear diffusion filtering as a pseudo-differential operator. Another basic operation in image processing is to take the mean of a symmetric neighbourhood of pixels. This corresponds to a convolution with a normalised box kernel that is part of our second example. Further examples are given in [38].

Examples

(1) Convolution with a Gaussian G_σ (linear diffusion): We have shown that the Fourier transform is again a Gaussian. With the help of the exponential series we have

$$\sigma^{-1}\cdot G_{\sigma^{-1}}(\xi) \;=\; \frac{1}{\sqrt{2\pi}}\cdot \sum_{m=0}^{\infty} \underbrace{(-1)^m}_{=i^{2m}} \frac{(\sigma\xi)^{2m}}{2^m\, m!}$$
$$\;=\; \frac{1}{\sqrt{2\pi}}\cdot \sum_{m=0}^{\infty} \frac{\sigma^{2m}}{2^m\, m!}\,(i\,\xi)^{2m}\,. \tag{1.52}$$

Hence, we write the convolution in terms of a pseudo-differential operator [38]

$$\begin{aligned} f * G_\sigma &= \exp\left(\frac{\sigma^2}{2} \cdot \frac{d^2}{dx^2}\right) f = \sum_{m=0}^{\infty} \frac{\sigma^{2m}}{2^m\, m!} \cdot \frac{d^{2m}}{dx^{2m}} f \\ &= f + \frac{\sigma^2}{2} \cdot f'' + \frac{\sigma^4}{8} \cdot f'''' + \dots . \end{aligned} \tag{1.53}$$

(2) Convolution with a box kernel: As mentioned above, the Fourier transform of $b_r(x)$ is given by

$$\hat{b}_r(\xi) = \frac{1}{\sqrt{2\pi}} \cdot \operatorname{sinc}(r\xi) . \tag{1.54}$$

With the Taylor expansion of the sinc-function we obtain

$$\begin{aligned} \hat{b}_r(\xi) &= \frac{1}{\sqrt{2\pi}} \cdot \sum_{m=0}^{\infty} (-1)^m \frac{(r\xi)^{2m}}{(2m+1)!} \\ &= \frac{1}{\sqrt{2\pi}} \cdot \sum_{m=0}^{\infty} \frac{r^{2m}}{(2m+1)!} (i\,\xi)^{2m} . \end{aligned} \tag{1.55}$$

Thus, we can represent the convolution with the box kernel b_r without any integral [38]:

$$\begin{aligned} f * b_r &= \operatorname{sinc}\left(-i\, r \cdot \frac{d}{dx}\right) f = \sum_{m=0}^{\infty} \frac{r^{2m}}{(2m+1)!} \cdot \frac{d^{2m}}{dx^{2m}} f \\ &= f + \frac{r^2}{6} \cdot f'' + \frac{r^4}{120} \cdot f'''' + \dots . \end{aligned} \tag{1.56}$$

If we reconsider the second example, it is also possible to write the sinc function as an infinite product [55]:

$$\operatorname{sinc}\left(-i\, r \cdot \frac{d}{dx}\right) = \prod_{m=1}^{\infty} \left(1 + \frac{r^2}{(m\,\pi)^2} \frac{d^2}{dx^2}\right) . \tag{1.57}$$

On the other hand, an explicit time discretisation of the 1-D linear diffusion equation

$$\partial_t u(x,t) = \partial_{xx} u(x,t) \tag{1.58}$$

with a time step size $\tau > 0$ yields the scheme

$$\begin{aligned} u(x,(\ell+1)\tau) &= u(x,\ell\tau) \;+\; \tau\,\frac{d^2}{dx^2}u(x,\ell\tau) \\ &= \left(1 \,+\, \tau\,\frac{d^2}{dx^2}\right)\, u(x,\ell\tau)\;. \qquad (1.59) \end{aligned}$$

Thus, a convolution with a box kernel can be represented as an infinite series of discrete explicit time steps with the varying step sizes $\tau_m = \frac{r^2}{(m\pi)^2}$ $(m = 1, 2, \dots)$. Interestingly, the step sizes depend quadratically on the radius r of the box function and become arbitrarily small for $m \to \infty$.

1.4 Outline of the Thesis

We have seen that the concept of pseudo-differential operators allows us to see a connection between the continuous convolution with a box kernel and a time-discrete diffusion process using infinitely many varying time steps.

However, our actual goal is to find such a representation in the discrete setting. To this end, we are going to develop a similar concept to pseudo-differential operators for discrete linear filters in Chapter 2. It will help us to express them in terms of numerical diffusion schemes with cycles of varying explicit time steps that partially violate stability restrictions. Fortunately, in contrast to the continuous case, the number of the time steps within a cycle is finite. Chapter 3 focuses in particular on the novel cyclic diffusion scheme that is based on 1-D discrete box filters. We are going to analyse the scheme with respect to theoretical as well as numerical stability, and show how it can be used to solve arbitrary parabolic problems. Based on this scheme, we construct a Jacobi-like linear system solver for elliptic problems. In Chapter 4, we present a recursion formula for box filters that helps us to improve the numerical stability of the novel methods. In this context, we also examine the relation to Runge-Kutta schemes and use it for the development of further schemes. We will also see that the recursion relation can be transferred to the Fast-Jacobi method. Afterwards, Chapter 5 shows that the novel cyclic explicit scheme can be used for time extrapolation, which is a way to construct second order methods. The thesis is concluded in Chapter 6, and we also give an outlook that contains some aspects for further research.

Chapter 2

A New Perspective on Discrete Linear Filters

Everything we hear is an opinion, not a fact.
Everything we see is a perspective, not the truth.
Marcus Aurelius

We are now going to derive and analyse the equivalence between 1-D linear symmetric filters and explicit diffusion schemes with varying time step sizes. The derivation makes use of a factorisation of one-dimensional linear symmetric filter kernels. To this end, we first consider the series expansions of such filters, which means that we represent them as a sum of discrete derivatives, i.e. finite differences. This chapter is mainly based on [165] and provides a more detailed analysis.

2.1 Discrete Linear Filters

Let $\boldsymbol{f} = (f_i)_{i\in\mathbb{Z}}$ and $\boldsymbol{g} = (g_i)_{i\in\mathbb{Z}}$ be discrete real-valued 1-D signals given on an equidistant grid with mesh size $h > 0$. The discrete convolution of the two signals is declared as

$$(\boldsymbol{f} *_h \boldsymbol{g})_i := \sum_{k\in\mathbb{Z}} f_k \cdot g_{i+k} \,. \tag{2.1}$$

It is well-defined for $\boldsymbol{f}, \boldsymbol{g} \in \ell_2(\mathbb{Z})$ with

$$\ell_2(\mathbb{Z}) \;:=\; \left\{ \boldsymbol{a} = (a_j)_{j\in\mathbb{Z}} \;\middle|\; \|\boldsymbol{a}\|_2 \;:=\; \Big(\sum_{j\in\mathbb{Z}} a_j^2\Big)^{\frac{1}{2}} \;<\; \infty \right\} , \tag{2.2}$$

or if one of the two signals, w.l.o.g. $\boldsymbol{g}$, is finite. This means that there exists an index $i_0 \in \mathbb{N}$ with $g_{-i} = g_i = 0$ for $i \geq i_0$. In this case, the existence of a constant $C > 0$ with $|f_i| \leq C$ for all $i \in \mathbb{Z}$ is sufficient.

When it comes to discrete linear filtering, the discrete convolution plays a very important role. Actually, a discrete linear filter is a transformation L that maps a signal to another one, e.g. $L(\boldsymbol{f}) = \boldsymbol{g}$. It satisfies the so-called *superposition principle*

$$L(\alpha \cdot \boldsymbol{f} \,+\, \beta \cdot \boldsymbol{g}) \;=\; \alpha \cdot L(\boldsymbol{f}) \,+\, \beta \cdot L(\boldsymbol{g}) \tag{2.3}$$

for all signals $\boldsymbol{f}, \boldsymbol{g}$ and $\alpha, \beta \in \mathbb{R}$. Additionally, we assume that the filter is *shift invariant*, i.e.

$$L\mathcal{T}_m(\boldsymbol{f}) \;=\; \mathcal{T}_m L(\boldsymbol{f}) \;, \tag{2.4}$$

with $m \in \mathbb{Z}$ and the shift operator $\mathcal{T}_m(\boldsymbol{f}) \;:=\; \tilde{\boldsymbol{f}} = (f_{i-m})_{i\in\mathbb{Z}}$. If we define the signal $\boldsymbol{\delta} := (\delta_{0,i})_{i\in\mathbb{Z}}$ with the Kronecker delta

$$\delta_{i,j} \;=\; \begin{cases} 1 & , \; i \,=\, j \\ 0 & , \; \text{else} \end{cases} \;, \tag{2.5}$$

the resulting signal $L(\boldsymbol{\delta})$ is called *impulse response*. Since the signal $\boldsymbol{f}$ can be written as

$$\boldsymbol{f} \;=\; \sum_{i\in\mathbb{Z}} f_i \cdot \mathcal{T}_i(\boldsymbol{\delta}) \;, \tag{2.6}$$

we get with the help of the superposition principle and the shift invariance

$$L(\boldsymbol{f}) \;=\; \sum_{i\in\mathbb{Z}} f_i \cdot \mathcal{T}_i L(\boldsymbol{\delta}) \;. \tag{2.7}$$

This means that L is completely characterised by its impulse response $L(\boldsymbol{\delta})$. It can be shown that a linear shift invariant filter always performs a convolution [80]. The corresponding convolution kernel (mask) is given by the impulse response.

Therefore, we define a discrete filter L^h_{2n+1} of length $(2n+1)h$ with $n \in \mathbb{N}$ in terms of the discrete convolution

$$\big(L^h_{2n+1}(\boldsymbol{f})\big)_i \;:=\; \sum_{k=-n}^{n} w_k \cdot f_{i+k} \;, \tag{2.8}$$

where $w_k \in \mathbb{R}$ are the so-called weights of the convolution kernel. We should mention that the definition of the length comes from the support $\left[-(n+\frac{1}{2})h\,,\,(n+\frac{1}{2})h\right]$ of the continuous version $W(x)$ of the discrete kernel, which we obtain by sampling at the points $x_k = k \cdot h$, i.e. $w_k = W(x_k)$. Note that the points x_k are the midpoints of the corresponding intervals $\left[(k-\frac{1}{2})h\,,\,(k+\frac{1}{2})h\right]$. Since we initially focus on diffusion processes, i.e. symmetric kernels, we assume $w_{-k} = w_k$ for $k \geq 1$. More details about discrete linear filters can be found for instance in [24, 80, 103].

2.2 Examples for Symmetric Filters

In this subsection, we present some well-known examples for discrete, symmetric filter kernels and show how they are related to linear diffusion. As we have already mentioned, a linear diffusion process is equivalent to a convolution of the original signal data with a Gaussian. Instead of discretising the Gaussian and computing the corresponding discrete convolution, we can also apply arbitrary filter kernels with non-negative weights that sum up to 1. This is possible due to the well-known *central limit theorem* [49]. To this end, the filter kernels are interpreted as probability density functions. The convolution of $k \geq 2$ density functions belonging to independent random variables $X_1, \ldots, X_k$ corresponds to the probability density function of the sum $X_1 + \ldots + X_k$. Then the central limit theorem states that this density function converges to a Gaussian for $k \to \infty$.

Our first example is actually a direct approximation for a Gaussian convolution, whereas the other three examples require an iterative application.

Binomial Kernel

A popular way for the approximation of Gaussians are binomial kernels. Their weights are given by

$$w_k \;=\; \frac{1}{4^n} \cdot \binom{2n}{n+k} \tag{2.9}$$

for a filter kernel with length $(2n+1)h$. The factor $\frac{1}{4^n}$ ensures that the weights sum up to 1. Binomial kernels have some nice properties: They can be used even in integer arithmetics, because the normalisation factors are powers of two and the division corresponds to bit shifts. Moreover, the composition of two binomial kernels with length $(2n_1+1)h$ and $(2n_2+1)h$ respectively is again a binomial kernel that has the length $(2(n_1+n_2)+1)h$. Hence, they are also suitable for an iterative application. Since the variance

σ^2 of a symmetric filter kernel with the weights w_k, $k \in \{-n, \ldots, n\}$, is given by the sum

$$\sigma^2 \;=\; h^2 \cdot \sum_{k=-n}^{n} k^2 \cdot w_k \;, \tag{2.10}$$

one can show that a binomial kernel of length $(2n+1)h$ has the variance $\sigma^2 = h^2 \cdot \frac{n}{2}$. Therefore, the standard deviation σ can not be tuned continuously. Since σ is proportional to $\sqrt{n}$, in particular very large standard deviations require a substantial computational effort.

Figure 2.1 illustrates an example regarding the binomial kernel with unit grid size. Using a kernel of length five, we iterate it three times. Actually, this corresponds to a binomial kernel with length 13. We compare this kernel to a 1-D Gaussian with the same standard deviation $\sigma = \sqrt{3}$. As one can see, the approximation given by the (iterated) binomial kernel is very good.

Discrete Filter with Maximum Variance

If we assume non-negative weights that sum up to 1 and reconsider the formula in Eq. (2.10), the maximum variance for the length $(2n+1)h$ is $\sigma^2 = h^2 \cdot n^2$. It is reached for the kernel defined by the weights

$$w_k \;=\; \begin{cases} \frac{1}{2} & , \; k \;=\; \pm n \\ 0 & , \; \text{else} \quad . \end{cases} \tag{2.11}$$

We denote the corresponding linear filter kernel by V^h_{2n+1}. Obviously, the standard deviation σ can not be tuned continuously neither. While the variance of the binomial kernel depends linearly on the width n, we have in this case a quadratic dependency. It means that even a small number of iterations of this filter yields an approximation for a Gaussian with a large standard deviation. However, in contrast to the binomial kernel, it lacks in accuracy even for a large number of iterative applications. Hence, we have some kind of a trade-off between the variance σ^2 and the accuracy of the approximation.

In Fig. 2.1 we see, for example, that the three times iterated kernel with length five provides a very poor approximation for the Gaussian with standard deviation $\sigma = \sqrt{12}$.

Discrete Box Filter

At this point we consider some kind of a compromise between the two previous filter kernels, namely a box filter. It is well-known that iterated box filtering yields a good approximation for Gaussian kernels [77]. The corresponding 1-D filter kernel for the length $(2n+1)h$ consists of the uniform weights

$$w_k \;=\; \frac{1}{2n+1}\,. \tag{2.12}$$

Concerning the variance, we obtain

$$\sigma^2 \;=\; \frac{h^2}{2n+1}\cdot\sum_{k=-n}^{n} k^2 \;=\; \frac{2h^2}{2n+1}\cdot\underbrace{\sum_{k=1}^{n} k^2}_{=\frac{n(n+1)(2n+1)}{6}} \;=\; h^2\cdot\frac{n^2\,+\,n}{3}\,. \tag{2.13}$$

Thus, it also grows quadratically in n and covers only a discrete set of variances like the two previous examples. We denote a discrete box filter of length $(2n{+}1)h$ by B^h_{2n+1}. Since box filters for multi-dimensional problems are separable, i.e. each direction can be treated as a convolution with a 1-D box kernel, they are also very efficient for multi-dimensional applications. Moreover, there exist very efficient ways for the evaluation of the convolution like for instance the *sliding-window approach* [167] that uses the relation

$$\big(B^h_{2n+1}(\boldsymbol{f})\big)_i \;=\; \big(B^h_{2n+1}(\boldsymbol{f})\big)_{i-1} \;-\; \frac{1}{2n+1}\cdot f_{i-1-n} \;+\; \frac{1}{2n+1}\cdot f_{i+n}\,. \tag{2.14}$$

Hence, except for the initialisation, the complexity is independent of the filter width and grows linearly in the signal length.

Although the variance grows quadratically in n, the example in Fig. 2.1 illustrates that iterated box filtering yields a good approximation. It is much better than the result of the filter with maximum variance.

Extended Box Filter

To overcome the problem of the discrete distribution with respect to the box filter variances, Gwosdek et al. have proposed the so-called *Extended Box Filter* [167]. This approach allows to generate arbitrary variances. The main idea for an extended filter E^h_{2n+1} with length $(2n{+}1)h$ is to consider a scaled box filter whose length is $(2n{-}1)h$ and add an additional weight at the boundaries such that all weights sum up to 1. More precisely, an extended box filter E^h_{2n+1} is a convex combination of a discrete box filter B^h_{2n-1} and the filter with maximum variance V^h_{2n+1}, i.e.

$$E^h_{2n+1} \;=\; \gamma\cdot B^h_{2n-1} \;+\; (1-\gamma)\cdot V^h_{2n+1}\,, \tag{2.15}$$

with $\gamma \in (0,1)$. The choice $\gamma = \frac{2n-1}{2n+1}$ yields the usual box filter B^h_{2n+1}, which means that the extended filter can be seen as a generalisation. Its variance is just the convex sum of the variances of the two previous filters. Thus, it is given by

$$\begin{aligned}\sigma^2 &= h^2 \cdot \left(\gamma \cdot \frac{n^2 - n}{3} + (1-\gamma)\cdot n^2\right) \\ &= h^2 \cdot \left(n^2 - \gamma \cdot \frac{2n^2 + n}{3}\right) . \end{aligned} \tag{2.16}$$

Since γ is a real-valued number, the variance σ^2 can be tuned in a continuous way. Therefore, the filter allows a more accurate approximation for Gaussian kernels with arbitrary standard deviations. Furthermore, the extended version shares some nice properties with the usual box filter: It is also separable and can be computed efficiently in a sliding-window manner. For further details, see [167]. As for the other filters, an example for the iterative application is shown in Fig. 2.1 with $\gamma = 3/4$. One can see that the approximation quality is comparable to the usual box filter.

2.3 Representation as Discrete Pseudo-Differential Operators

In Chapter 1 we have seen that a continuous convolution can be represented as a weighted infinite sum of arbitrary order derivatives, a pseudo-differential operator. Now we want to show that every discrete linear, symmetric 1-D filter L^h_{2n+1} can be written as a weighted sum of discrete even order derivatives, i.e. a finite series expansion like

$$L^h_{2n+1} = \sum_{m=0}^{n} \alpha^{(n)}_m \cdot \Delta^m_h , \tag{2.17}$$

with the discrete 1-D Laplacian defined by means of the finite difference

$$(\Delta_h \boldsymbol{f})_i := \frac{f_{i+1} - 2\,f_i + f_{i-1}}{h^2} , \tag{2.18}$$

and real-valued coefficients $\alpha^{(n)}_m$. Note that this finite difference is equivalent to a convolution with the symmetric kernel $\frac{1}{h^2}(1,-2,1)$. For $m = 0$, Δ^m_h is just the identity operator and for $m > 1$ the m-times composition of the discrete Laplacian, which corresponds to a finite difference approximating the derivative of order $2m$. The following proposition states a closed-form expression for these finite differences.

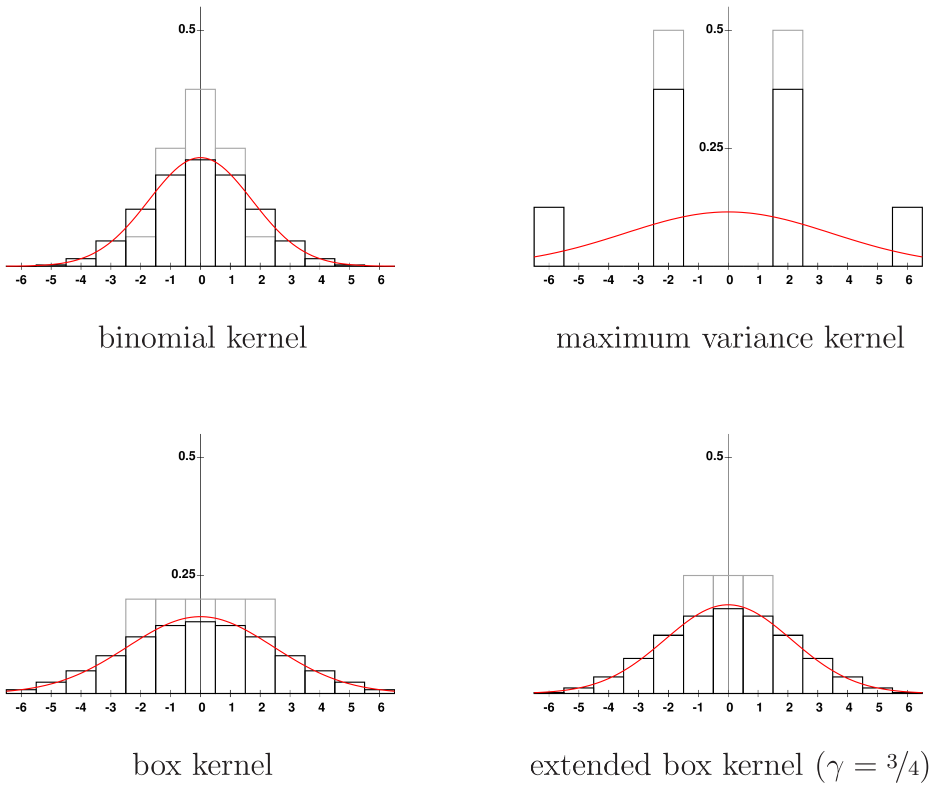


Figure 2.1: Examples for filter kernels with unit grid size. Each example consists of the original kernel with length 5 (grey), the three times iterated kernel (black), and the corresponding Gaussian to be approximated (red).

Proposition 2.1 (Closed-form expression for $\boldsymbol{\Delta_h^m}$). *Let $\boldsymbol{f}$ be a discrete signal and $m \geq 1$. Then the m-times composition of the discrete Laplacian Δ_h fulfils*

$$(\Delta_h^m \boldsymbol{f})_i \;=\; \frac{1}{h^{2m}} \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i+k} \;. \tag{2.19}$$

Proof. We use a proof by induction: The above equation is obviously valid for $m = 1$. If we assume that it holds for an arbitrary $m \geq 1$, this yields for $m+1$:

$$
\begin{aligned}
\left(\Delta_h^{m+1}(\boldsymbol{f})\right)_i \;&=\; \left(\Delta_h^m\left(\frac{f_{\cdot+1} \;-\; 2\,f_{\cdot} \;+\; f_{\cdot-1}}{h^2}\right)\right)_i \\
&\overset{(2.19)}{=} \frac{1}{h^{2m}} \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot \left(\frac{f_{i+1+k} - 2f_{i+k} + f_{i-1+k}}{h^2}\right) \\
&=\; \frac{1}{h^{2(m+1)}} \Bigg(f_{i+1+m} \;+\; \sum_{k=-m}^{m-1} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i+1+k} \\
&\qquad -\; 2 \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i+k} \\
&\qquad +\; \sum_{k=-m+1}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i-1+k} \\
&\qquad +\; f_{i-1-m} \Bigg) \;.
\end{aligned}
\tag{2.20}
$$

We perform a change of indices for both the first $(k \to k{-}1)$ and the third sum $(k \to k+1)$. Furthermore, we use the fact that

$$
\binom{2m}{-1} \;=\; \binom{2m}{2m+1} \;=\; 0\;. \tag{2.21}
$$

Thus, we get

$$
\begin{aligned}
&\frac{1}{h^{2(m+1)}} \Bigg(f_{i+1+m} \;+\; \sum_{k=-m}^{m} (-1)^{m+k-1} \binom{2m}{m+k-1} \cdot f_{i+k} \\
&\qquad -\; 2 \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i+k} \\
&\qquad +\; \sum_{k=-m}^{m} (-1)^{m+k+1} \binom{2m}{m+k+1} \cdot f_{i+k} \;+\; f_{i-1-m} \Bigg) \\
=\;& \frac{1}{h^{2(m+1)}} \Bigg(f_{i+(m+1)} \;+\; f_{i-(m+1)} \;+\; \sum_{k=-m}^{m} (-1)^{m+k+1}\, f_{i+k} \,\cdot
\end{aligned}
$$

$$\cdot \left[\binom{2m}{m+k+1} + 2\binom{2m}{m+k} + \binom{2m}{m+k-1} \right] \Bigg) . \tag{2.22}$$

Because of

$$\begin{aligned} &\binom{2m}{m+k+1} + 2\binom{2m}{m+k} + \binom{2m}{m+k-1} \\ =\ &\binom{2m}{m+k+1} + \binom{2m}{m+k} + \binom{2m}{m+k} + \binom{2m}{m+k-1} \\ =\ &\binom{2m+1}{m+k+1} + \binom{2m+1}{m+k} = \binom{2m+2}{m+k+1} , \end{aligned} \tag{2.23}$$

Eq. (2.22) finally results in

$$\frac{1}{h^{2(m+1)}} \left(\sum_{k=-(m+1)}^{m+1} (-1)^{(m+1)+k} \binom{2(m+1)}{(m+1)+k} \cdot f_{i+k} \right) . \tag{2.24}$$

□

Proposition 2.1 is very useful in order to show that Eq. (2.17) is valid. The coefficients $\alpha_m^{(n)}$ are unique and therefore the representation of a symmetric filter L_{2n+1}^h as a weighted sum of the discrete even-order derivatives Δ_h^m is also unique. In this context, we are going to prove the following proposition:

Proposition 2.2 (Unique representation as discrete pseudo-differential operator). *Every linear symmetric 1-D filter L_{2n+1}^h has a unique set of coefficients $\{\alpha_m^{(n)} \mid 0 \le m \le n\}$ that fulfils Eq.* (2.17).

Proof. If we replace Δ_h^m by its explicit formula given in Prop. 2.1, then we have for the right hand side of Eq. (2.17):

$$\sum_{m=0}^{n} \alpha_m^{(n)} \left(\Delta_h^m(\boldsymbol{f})\right)_i = \sum_{m=0}^{n} \frac{\alpha_m^{(n)}}{h^{2m}} \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} \cdot f_{i+k} . \tag{2.25}$$

On the other hand, we have the symmetric filter

$$\left(L_{2n+1}^h(\boldsymbol{f})\right)_i = \sum_{k=-n}^{n} w_{|k|} \cdot f_{i+k} , \tag{2.26}$$

with the weights $w_0, \ldots, w_n \in \mathbb{R}$. The comparison of the last two equations yields a system of linear equations with the $n+1$ unknowns $\alpha_m^{(n)}$ and $n+1$ equations:

$$\sum_{m=k}^{n} (-1)^{m+k} \cdot \frac{1}{h^{2m}} \binom{2m}{m+k} \cdot \alpha_m^{(n)} \;=\; w_k \qquad \forall\, k \in \{0, ..., n\} \;. \tag{2.27}$$

The corresponding matrix-vector notation of Eq. (2.27) is

$$\boldsymbol{B}\,\boldsymbol{\alpha}^{(n)} \;=\; \boldsymbol{w} \;, \tag{2.28}$$

where $\boldsymbol{B} \;=\; (b_{k,m})_{k,m=0}^{n} \in \mathbb{R}^{(n+1)\times(n+1)}$ with the entries

$$b_{k,m} \;=\; (-1)^{m+k} \cdot \frac{1}{h^{2m}} \binom{2m}{m+k} \;, \tag{2.29}$$

the unknown solution $\boldsymbol{\alpha}^{(n)} \;=\; \left(\alpha_0^{(n)}, \ldots, \alpha_n^{(n)}\right)^T \in \mathbb{R}^{n+1}$ and the given vector $\boldsymbol{w} \;=\; (w_0, \ldots, w_n)^T \in \mathbb{R}^{n+1}$. Since $b_{k,m} \;=\; 0$ for $k > m$ and $b_{k,k} \;=\; \frac{1}{h^{2k}} \;\neq\; 0$, $\boldsymbol{B}$ is a regular upper triangular matrix and the unique solution is given by $\boldsymbol{\alpha}^{(n)} \;=\; \boldsymbol{B}^{-1}\,\boldsymbol{w}$. □

Since we have proven the existence and uniqueness of the representation in Eq. (2.17), we can state that the set

$$\mathcal{B}_n \;:=\; \{\, \Delta_h^m \mid 0 \leq m \;\leq\; n \,\} \tag{2.30}$$

of the finite differences Δ_h^m is a basis for the set of all symmetric filters with length $(2n+1)h$. The dimension is obviously equal to $n+1$. Hence, the multiplication with the matrix $\boldsymbol{B}^{-1}$ is a change of basis, i.e. the filter L_{2n+1}^h is expressed in terms of the new basis $\mathcal{B}_n$. The next proposition gives an explicit formula for the inverse matrix $\boldsymbol{B}^{-1}$. Then we can compute the representation coefficients with respect to $\mathcal{B}_n$ by a simple matrix-vector multiplication.

Proposition 2.3 (Change-of-basis matrix $\boldsymbol{B}^{-1}$). *The entries of the inverse matrix* $\boldsymbol{B}^{-1} = \left(b_{k,m}^{-1}\right)_{k,m=0}^{n} \in \mathbb{R}^{(n+1)\times(n+1)}$ *are given by*

$$b_{k,m}^{-1} \;=\; h^{2k} \left(\binom{m+k}{2k} \;+\; \left(1 \,-\, \delta_{(m+k),0}\right) \cdot \binom{m+k-1}{2k} \right) \;, \tag{2.31}$$

with the Kronecker delta $\delta_{i,j}$ *from Eq. (2.5).*

Proof. We have to show that

$$\sum_{p=0}^{n} b_{k,p}^{-1} \cdot b_{p,m} \;=\; \delta_{k,m} \; . \tag{2.32}$$

Since $\boldsymbol{B}$ is an upper triangular matrix, its inverse $\boldsymbol{B}^{-1}$ is also an upper triangular matrix. Therefore the summation is only necessary for the indices $p \in \{k, ..., m\}$. Thus, the above equation can be simplified to

$$\sum_{p=k}^{m} b_{k,p}^{-1} \cdot b_{p,m} \;=\; \delta_{k,m} \; . \tag{2.33}$$

This equation obviously holds for the case $k > m$ because of an empty sum. If $k = m$, then it is also valid, since $b_{k,k}^{-1} = h^{2k} = \frac{1}{b_{k,k}}$.

Let now $m > k \geq 0$. Then the sum is equal to

$$h^{2(k-m)} \cdot \sum_{p=k}^{m} \left(\binom{p+k}{2k} + \left(1 - \delta_{(p+k),0}\right) \binom{p+k-1}{2k} \right) \cdot (-1)^{m+p} \binom{2m}{m+p} \; . \tag{2.34}$$

We first consider the case $k > 0$, i.e. $\delta_{(p+k),0} = 0$. The $2k$-degree polynomial

$$\begin{aligned} s(p) \;&:=\; \binom{p+k}{2k} + \binom{p+k-1}{2k} \\ &=\; \prod_{j=1}^{2k} \frac{p + (k+1-j)}{j} \;+\; \prod_{j=1}^{2k} \frac{p + (k-j)}{j} \end{aligned} \tag{2.35}$$

fulfils the symmetry condition $s(p) = s(-p)$ and regarding its roots we have $s(p) = 0$ for the indices $p \in \{-k+1, ..., k-1\}$. With the help of these facts we obtain

$$\begin{aligned} 0 \;&\overset{(2.37)}{=}\; \sum_{p=-m}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \\ &=\; \sum_{p=-m}^{-k} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \;+\; \sum_{p=k}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \\ &=\; 2 \cdot \sum_{p=k}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \; , \end{aligned} \tag{2.36}$$

where we have used a well-known property of binomial coefficients:

$$\sum_{j=0}^{r} (-1)^j \, P(j) \binom{r}{j} \;=\; 0 \tag{2.37}$$

for any polynomial $P(j)$ with degree less than $r \in \mathbb{N}$. Here $s(p)$ has degree $2k < 2m$ and thus satisfies Eq. (2.37).

In the case of $k = 0$, we have $b^{-1}_{0,0} = 1$ and $b^{-1}_{0,p} = 2$ for $p \neq 0$. Hence,

$$\begin{aligned}
\sum_{p=0}^{m} b^{-1}_{0,p} \cdot (-1)^{m+p} \binom{2m}{m+p} \;&=\; (-1)^m \binom{2m}{m} \;+\; 2 \cdot \sum_{p=1}^{m} (-1)^{m+p} \binom{2m}{m+p} \\
&=\; (-1)^m \binom{2m}{m} \;+\; \sum_{\substack{p=-m \\ p\neq 0}}^{m} (-1)^{m+p} \binom{2m}{m+p} \\
&=\; \sum_{p=-m}^{m} (-1)^{m+p} \binom{2m}{m+p} \;\overset{(2.37)}{=}\; 0 \,.
\end{aligned} \tag{2.38}$$

Considering the equations (2.36) and (2.38), it follows that

$$\sum_{p=k}^{m} b^{-1}_{k,p} \cdot b_{p,m} \;=\; 0 \tag{2.39}$$

for $m > k$. This concludes the proof. □

With this proposition we have established an explicit connection between the filter weights $w_0, ..., w_n$ and the coefficients $\alpha_0^{(n)}, ..., \alpha_n^{(n)}$ of the discrete pseudo-differential operator. Thus, we can formulate the central theorem of this subsection:

Theorem 2.4 (Coefficients of the Expansion). *Let $w_k \in \mathbb{R}$ with $k = 0, \ldots, n$ be the weights of the symmetric filter kernel. Then the coeffcients of its expansion in Eq. (2.17) are given by*

$$\begin{aligned}
\alpha_m^{(n)} \;&=\; \sum_{k=m}^{n} b^{-1}_{m,k} \cdot w_k \\
&=\; h^{2m} \sum_{k=m}^{n} \left(\binom{k+m}{2m} \;+\; \left(1 \;-\; \delta_{(k+m),0}\right) \binom{k+m-1}{2m} \right) w_k \,.
\end{aligned} \tag{2.40}$$

2.4 The Symbol of Discrete Linear Filters

We have shown that a discrete symmetric filter L^h_{2n+1} can be represented as a series expansion. It consists of finite differences that approximate even order derivatives. As mentioned in the previous chapter, a derivative in the spatial domain corresponds to a multiplication with a monomial in the Fourier domain. The degree of the monomial depends on the corresponding order of the derivative. Having applied the Fourier transform on a filter kernel, one can analyse its behaviour regarding different frequency components.

The *symbol* or *amplification factor* (see e.g. [126]) is a similar concept: We replace the operator, here the discrete 1-D Laplacian Δ_h, by the variable -z reflecting the spectrum or the eigenvalues of this operator. Then we get the symbol of L^h_{2n+1} ,

$$p_L^{[n]}(z) \;:=\; \sum_{m=0}^{n} \alpha_m^{(n)} (-z)^m \;, \tag{2.41}$$

where $z \in [0\,,\,\frac{4}{h^2}]$ in the case of the discrete 1-D Laplacian. This can be seen as follows: A signal $\boldsymbol{g}$ is an eigensignal to the eigenvalue $\lambda \in \mathbb{C}$, if we have for any arbitrary index $i \in \mathbb{Z}$

$$(\Delta_h\, \boldsymbol{g})_i \;=\; \frac{g_{i+1} \,-\, 2g_i \,+\, g_{i-1}}{h^2} \;=\; \lambda \cdot g_i \;. \tag{2.42}$$

Using the well-known theorem of Gerschgorin [58, 139] for arbitrary linear operators [84], we can state that the eigenvalue λ fulfils

$$\lambda \;\in\; \left\{ a \in \mathbb{C} \;:\; \left|a + \tfrac{2}{h^2}\right| \;\le\; \tfrac{2}{h^2} \right\} \;. \tag{2.43}$$

Due to the symmetry, or self-adjointness, of Δ_h, the eigenvalues have to be real-valued and hence we obtain $\lambda \in [-\frac{4}{h^2}\,,\,0]$.

The graph of the symbol illustrates the behaviour of the filter concerning the frequency components. More precisely, the value $p_L^{[n]}(z_0)$ indicates how the filter L^h_{2n+1} modifies the frequency component $\boldsymbol{v}_{z_0}$ of the initial signal belonging to z_0, since the corresponding component of the filtered signal is equal to $p_L^{[n]}(z_0) \cdot \boldsymbol{v}_{z_0}$. To this end, the condition $|p_L^{[n]}(z_0)| \le 1$ for all eigenvalues z_0 ensures that the frequency components are not amplified and one can show stability in the Euclidean norm [126]. Low frequencies refer to small values $z_0 \approx 0$, where larger eigenvalues near $\frac{4}{h^2}$ represent the highest frequency components. This can be seen, for example, if we consider two different eigensignals: The constant signal $\boldsymbol{v}_0 = (\dots, 1, 1, 1, 1, \dots)$ corresponds to frequency 0 and is an eigensignal of Δ_h with eigenvalue 0. On the other hand, the high frequency eigensignal $\boldsymbol{v}_{4/h^2} = (\dots, -1, 1, -1, 1, \dots)$ corresponds to the eigenvalue $-\frac{4}{h^2}$. Some examples are depicted in Fig. 2.2. Since the analytic solution of the ODE system (1.9) for linear diffusion is given by a matrix exponential, the ideal symbols should be similar to the exponential functions $\exp(-T \cdot z)$, where the real number $T > 0$ is the stopping time of the diffusion process. This ensures that the higher the frequency, the stronger is the damping, because higher frequency components are multiplied with smaller values. If we take a look at the symbols shown in Fig. 2.2, the binomial kernel has the best and the maximum variance kernel V_7^h the worst amplification factor. Even the iterative application of the maximum variance kernel does not yield a significant improvement, as we have already seen in the example shown in Fig. 2.1. The symbol of the iterated version is illustrated in Fig. 2.3 and is obviously far from approximating an exponential function.

Actually, the problem is that the amplification factor $p_V^{[3]}(z)$ of V_7^h takes the values ± 1 for $z > 0$. If we iterate V_7^h m times, the corresponding symbol is $\left(p_V^{[3]}(z)\right)^m$ and still takes the values ± 1 for odd m or only 1 for even m. Thus, even a large number of iterations does not yield better approximations, and higher frequency components might be preserved due to the multiplication with ± 1. However, in the case of the box or extended box filtering, Fig. 2.3 demonstrates how the iterative application achieves amplification factors that hardly differ from an exponential function. Therefore, the corresponding iterated kernels yield a significantly better behaviour with respect to the higher frequency components. In fact, this has already been indicated by the good results in Fig. 2.1.

For the following sections we just use the notation p_L instead of $p_L^{[n]}$, which means that we assume the kernel length to be $(2n+1)h$.

2.5 Connection to Fourier Analysis

Concerning the definition of the symbol, we replace the operator Δ_h by a real-valued, non-positive variable. As we have mentioned above, this concept is similar to a Fourier transform. More precisely, it can be seen as a modification of the so-called *Z-transform* [103]. It is defined by

$$G(z) = \sum_{j \in \mathbb{Z}} g_j \cdot z^{-j} , \tag{2.44}$$

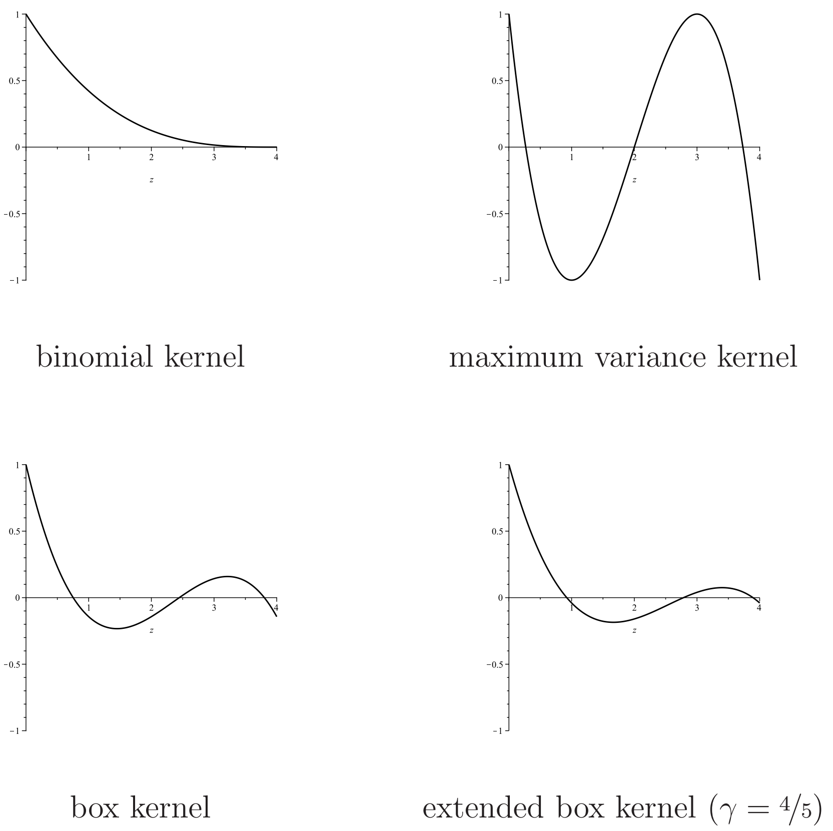


Figure 2.2: Examples for the symbols of the filter kernels with unit grid size and length 7.

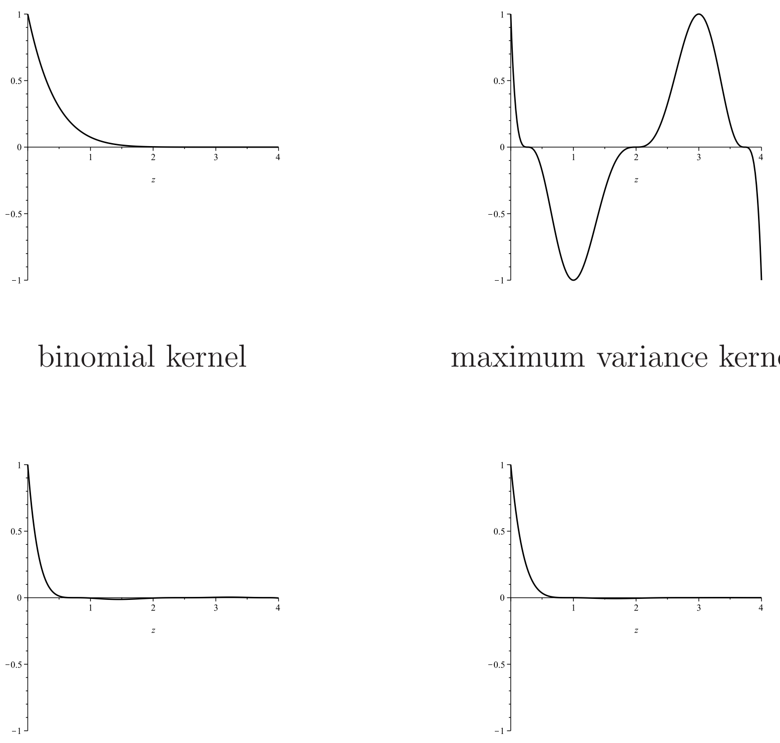
where $\boldsymbol{g} = (g_j)_{j\in\mathbb{Z}}$ denotes a signal or a filter kernel. The Z-transform $D(z)$ of Δ_h, i.e. the kernel $(\dots, 0, \frac{1}{h^2}, -\frac{2}{h^2}, \frac{1}{h^2}, 0, \dots)$, is given by

$$D(z) \;=\; \frac{1}{h^2} \cdot \left(\frac{1}{z} \;-\; 2 \;+\; z \right) \;. \tag{2.45}$$

For arbitrary $\tilde{z} \geq 0$ we obtain

$$-\, D \left(\tfrac{2 + h^2 \cdot \tilde{z}}{2} \pm \sqrt{\tfrac{(2 + h^2 \cdot \tilde{z})^2}{4} \;-\; 1} \;\right) \;=\; -\tilde{z} \;, \tag{2.46}$$

which shows that we implicitly apply a modified Z-transform when we replace Δ_h by a non-positive variable. To establish the connection between the symbol and the Fourier analysis, we now consider the definition of the well-known *Discrete Time Fourier Transform* [103]:



binomial kernel — maximum variance kernel

box kernel — extended box kernel ($\gamma = 4/5$)

Figure 2.3: Examples for the symbols of the three times iterated filter kernels with unit grid size and length 7.

Definition 2.5. *The* ***Discrete Time Fourier Transform*** *(DTFT) of a discrete signal* $\boldsymbol{f} = (f_j)_{j\in\mathbb{Z}}$ *is defined by the function*

$$\hat{\boldsymbol{f}}(\omega) \;=\; \sum_{j\in\mathbb{Z}} f_j \cdot \exp(-i\,\omega\,j)\;, \tag{2.47}$$

with the complex number i *and the continuous normalised radian frequency variable* $\omega \in [-\pi, \pi)$.

As one can see, it is a special case of the Z-transform with $z = \exp(i\,\omega) \in \mathbb{C}$, and therefore implies the connection of the symbol to the Fourier analysis. Moreover, the DTFT can be seen as a continuous generalisation of the popular *Discrete Fourier Transform* (DFT), because the DFT corresponds to the evaluation of the DTFT $\hat{\boldsymbol{f}}(\omega)$ at discrete frequencies $\omega = \omega_k$. It is also well-known that there exists a discrete convolution theorem for the DTFT [103]:

Theorem 2.6 (Convolution Theorem for DTFT). *Let* $\boldsymbol{f} = (f_j)_{j\in\mathbb{Z}}$ *and* $\boldsymbol{g} = (g_j)_{j\in\mathbb{Z}}$ *be two discrete real-valued* 1*-D signals such that their convolution* $(\boldsymbol{f} *_h \boldsymbol{g})$ *exists. Then the DTFT fulfils the equation*

$$(\boldsymbol{f} *_h \boldsymbol{g})\hat{}\,(\omega) \;=\; \hat{\boldsymbol{f}}(\omega)\cdot\hat{\boldsymbol{g}}(\omega)\;. \tag{2.48}$$

With the help of this theorem, we can analyse the behaviour of a filter L^h_{2n+1} in the Fourier domain by computing the DTFT with respect to the finite, symmetric filter kernel $\boldsymbol{w} = (w_k)_{k=-n}^{n}$. Because of the symmetry, the complex sine part vanishes, and we have

$$\hat{\boldsymbol{w}}(\omega) \;=\; \sum_{k=-n}^{n} w_k\cdot\exp(-i\,\omega\,k) \;=\; w_0 \;+\; 2\cdot\sum_{k=1}^{n} w_k\cdot\cos(k\,\omega)\;. \tag{2.49}$$

At this point, we are going to introduce the famous Chebyshev polynomials of first kind [1, 30, 97]. This helps us to get a better comprehension of Eq. (2.49).

Definition 2.7. *The* ***Chebyshev polynomials of the first kind*** *are defined by the recursion*

$$\begin{cases} T_0(x) &= 1\;,\\ T_1(x) &= x\;,\\ T_{n+1}(x) &= 2x\cdot T_n(x) \;-\; T_{n-1}(x)\;. \end{cases} \tag{2.50}$$

A closed-form representation is given by [1]

$$T_n(x) \;=\; \frac{n}{2}\cdot\sum_{m=0}^{\lfloor n/2\rfloor}\frac{(-1)^m}{n-m}\binom{n-m}{m}(2x)^{n-2m}\;. \tag{2.51}$$

If $x \in [-1,1]$, i.e. $x = \cos(\phi)$ with $\phi \in [-\pi, \pi)$, the Chebyshev polynomials T_n can be written in terms of a cosine function:

$$T_n(x) \;=\; T_n(\cos(\phi)) \;=\; \cos(n \cdot \phi)\ . \tag{2.52}$$

Thus, Eq. (2.49) can be rewritten as

$$\hat{\boldsymbol{w}}(\omega) \;=\; w_0 \;+\; 2 \cdot \sum_{k=1}^{n} w_k \cdot T_k(\cos(\omega))\ , \tag{2.53}$$

which means that the DTFT of $\boldsymbol{w}$ is a linear combination of Chebyshev polynomials. Therefore, $\hat{\boldsymbol{w}}(\omega)$ corresponds to a polynomial in $s := \cos(\omega)$ with degree n.

Assume we have given a symmetric filter L^h_{2n+1} with the filter weights $w_0, \ldots, w_n$. Since we can rewrite the filter in terms of Eq. (2.17), the application of the DTFT on both sides of this equation yields

$$w_0 \;+\; 2 \cdot \sum_{k=1}^{n} w_k \cdot T_k(\cos(\omega)) \;=\; \sum_{m=0}^{n} \alpha_m^{(n)} \cdot (\Delta_h^m)\hat{}\,(\omega)\ , \tag{2.54}$$

where we have used the linearity of the DTFT. Regarding the right hand side, we have to evaluate the DTFT of the finite difference operators Δ_h^m. This will be done by the following proposition.

Proposition 2.8 (**DTFT of $\boldsymbol{\Delta_h^m}$**). *Let $m \geq 0$. Then the following holds:*

$$(\Delta_h^m)\hat{}\,(\omega) \;=\; \left(\tfrac{2}{h^2}\cos(\omega) \;-\; \tfrac{2}{h^2}\right)^m\ . \tag{2.55}$$

Proof. The case $m = 0$ is obviously true, because the DTFT of the identity operator is just a constant function with the value 1. Hence, we can assume that Eq. (2.55) is valid for an arbitrary $m \geq 0$. For the induction step $m \to m+1$, we can state that the operator Δ_h^{m+1} is a discrete convolution between the symmetric kernel mask of Δ_h^m (cf. Prop. 2.1) and the mask of Δ_h, $(\frac{1}{h^2}\,,\, -\frac{2}{h^2}\,,\, \frac{1}{h^2})$. In the case of the operator Δ_h, the corresponding DTFT is

$$\begin{aligned}(\Delta_h)\hat{}\,(\omega) \;&=\; \frac{1}{h^2}\exp(i\,\omega) \;-\; \frac{2}{h^2} \;+\; \frac{1}{h^2}\exp(-i\,\omega) \\ &=\; -\frac{2}{h^2} \;+\; \frac{2}{h^2}\cos(\omega)\ . \end{aligned} \tag{2.56}$$

Using the convolution theorem 2.6, we obtain for $m+1$

$$\begin{aligned}\left(\Delta_h^{m+1}\right)\hat{}\,(\omega) &= \left(\Delta_h^m\right)\hat{}\,(\omega)\cdot\left(\Delta_h\right)\hat{}\,(\omega)\\ &\overset{(2.55)}{=} \left(\tfrac{2}{h^2}\cos(\omega) - \tfrac{2}{h^2}\right)^m\cdot\left(-\tfrac{2}{h^2} + \tfrac{2}{h^2}\cos(\omega)\right)\\ &= \left(\tfrac{2}{h^2}\cos(\omega) - \tfrac{2}{h^2}\right)^{m+1}\,. \qquad (2.57)\end{aligned}$$

This concludes the proof. $\square$

Thus, by means of Eq. (2.55) we can rewrite Eq. (2.54) with $s = \cos(\omega)$ as follows:

$$\begin{aligned}w_0 + 2\sum_{k=1}^{n} w_k\cdot T_k(s) &= \sum_{m=0}^{n}\alpha_m^{(n)}\cdot\left(\tfrac{2}{h^2}s - \tfrac{2}{h^2}\right)^m\\ &= p_L\left(\tfrac{2}{h^2}(1-s)\right)\,. \qquad (2.58)\end{aligned}$$

Since the right hand side of this equation is the above mentioned polynomial $q_L(s)$, the substitution $s = (1 - \frac{h^2}{2}z)$ finally results in

$$p_L(z) = w_0 + 2\cdot\sum_{k=1}^{n} w_k\cdot T_k\left(1 - \tfrac{h^2}{2}z\right)\,. \qquad (2.59)$$

Hence, the symbol $p_L(z)$ can also be represented in terms of a linear combination of Chebyshev polynomials. Moreover, one should mention that if L_{2n+1}^h is a symmetric filter with positive weights that sum up to 1, Eq. (2.59) represents a convex combination. Since all Chebyshev polynomials $T_k\left(1 - \frac{h^2}{2}z\right)$ are bounded in absolute value by 1, all convex combinations also fulfil this constraint, i.e. we have in this case $|p_L(z)| \leq 1$. On the other hand, there are filters L_{2n+1}^h with negative weights satisfying $|p_L(z)| \leq 1$ for $z \in [0\,,\,\frac{4}{h^2}]$.

As an example we consider the filter mask $(1/3, 1/4, -1/6, 1/4, 1/3)$. It can be written as a discrete convolution of the two explicit diffusion kernels $(1/4, 1/2, 1/4)$, which corresponds to the time step size $\tau = 1/4$, and $(4/3, -5/3, 4/3)$, where we have $\tau = 4/3$. The corresponding amplification factor $(1 - \frac{1}{4}z)\cdot(1 - \frac{4}{3}z)$ is bounded in absolute value by 1, as shown in Fig. 2.4.

Overall, we can state that the symbol is related to Chebyshev polynomials of the first kind. This will be very helpful for the next section, where we want to show and analyse the equivalence of L_{2n+1}^h and explicit linear diffusion schemes.

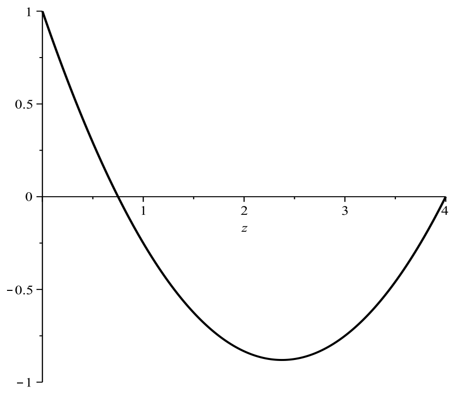


Figure 2.4: Symbol of the filter $(1/3, 1/4, -1/6, 1/4, 1/3)$ with unit grid size.

2.6 Diffusion Interpretation of Linear Filters

Having computed the coefficients $\alpha_0^{(n)}, ..., \alpha_n^{(n)}$ according to Eq. (2.40), we can reconsider the symbol of the filter L_{2n+1}^h. If the weights of the filter L_{2n+1}^h are non-negative, which means the coefficients $\alpha_m^{(n)}$ are also non-negative, then $p_L(z) \geq \alpha_0^{(n)}$ for $z \leq 0$. Thus, the real-valued roots of p_L have to be non-negative. According to the fundamental theorem of algebra, $p_L(z)$ has n roots $z_0, ..., z_{n-1} \in \mathbb{C}$. Hence, it can be written as a product of n linear factors,

$$p_L(z) \;=\; c \cdot \prod_{m=0}^{n-1} (z_m - z) \;, \tag{2.60}$$

where $c \in \mathbb{R}$ is the normalisation factor. In the case of $p_L(0) = \alpha_0^{(n)} > 0$, we have $z_m \neq 0$ for all m and therefore this factor has to fulfil

$$c \;=\; \alpha_0^{(n)} \cdot \left(\prod_{m=0}^{n-1} z_m \right)^{-1} \;. \tag{2.61}$$

In the following, we assume that the weights $w_{-n}, ..., w_n$ sum up to 1. Using the symmetry constraint $w_{-k} = w_k$, this yields

$$\begin{aligned} \alpha_0^{(n)} \;&=\; \sum_{k=0}^{n} \left(\binom{k}{0} \;+\; (1 \;-\; \delta_{k,0}) \binom{k-1}{0} \right) w_k \\ &=\; w_0 \;+\; 2 \cdot \sum_{k=1}^{n} w_k \;=\; 1 \;. \end{aligned} \tag{2.62}$$

Since $p_L(0) = \alpha_0^{(n)} = 1$, the symbol $p_L(z)$ can be rewritten as follows:

$$p_L(z) \;=\; \prod_{m=0}^{n-1} \left(1 - \frac{z}{z_m} \right) \;. \tag{2.63}$$

Replacing $(-z)$ by the discrete Laplacian Δ_h and interpreting the corresponding product as a composition of operators finally shows

$$L_{2n+1}^h \;=\; \prod_{m=0}^{n-1} \left(I \,+\, z_m^{-1}\, \Delta_h \right) \;, \tag{2.64}$$

where I denotes the identity operator, i.e. $I\,\boldsymbol{f} = \boldsymbol{f}$.

1-D Explicit Diffusion Scheme

Since the goal of this section is to show the equivalence of 1-D linear symmetric filters and 1-D explicit diffusion schemes, we are going to spend some words about such explicit methods. To this end, we consider the linear 1-D diffusion equation on a real interval $[a, b]$ with $a < b$:

$$\partial_t u(x,t) \;=\; \partial_{xx} u(x,t) \;, \tag{2.65}$$

where $(x,t) \in (a,b) \times (0,\infty)$. For $t = 0$ we have a given bounded function $u(x,0) = u_0(x)$, $x \in [a,b]$. Moreover, we assume homogeneous Neumann boundary conditions

$$\partial_x u(a,t) \;=\; \partial_x u(b,t) \;=\; 0 \tag{2.66}$$

for $t \in (0,\infty)$. It can be shown that this problem is well-posed and has a unique solution [32].

In order to solve this problem numerically, we define a spatiotemporal grid $\mathcal{G}$ with the spatial grid size $h > 0$ and the time step size $\tau > 0$,

$$\mathcal{G} \;:=\; \left\{ (x_i\,,\,t_k) \in [a,b] \times [0,T] \;\middle|\; x_i = a + \left(i - \tfrac{1}{2}\right) h\,,\; t_k = k\tau \right\} \;, \tag{2.67}$$

with suitable indices i, k. Usually, one has $h = \frac{b-a}{N}$ with the number of spatial grid points N and $\tau = \frac{T}{M}$, where M is the number of time step sizes. Then we use $1 \le i \le N$ and $0 \le k \le M$. The numerical solution is described by the approximations $u_i^k \approx u(x_i, t_k)$. Concerning the boundary conditions, we use so-called dummy variables $u_0^k := u_1^k$ and $u_{N+1}^k := u_N^k$

that model vanishing first order derivatives at the boundaries $x = a$ and $x = b$ respectively. The explicit discretisation of the evolution equation

$$\partial_t u(x,t) \;=\; \partial_{xx} u(x,t) \tag{2.68}$$

on the spatiotemporal grid yields

$$\frac{u_i^{k+1} \,-\, u_i^k}{\tau} \;=\; \frac{u_{i+1}^k \,-\, 2\,u_i^k \,+\, u_{i-1}^k}{h^2}\,, \tag{2.69}$$

with a forward finite difference discretising the time derivative and a central finite difference for the second order spatial derivative. This discretisation uses solely already known values for the right hand side. An implicit discretisation would use new unknown values from the next time step $k+1$, which implies the necessity for solving a linear system. If we define the finite signal $\boldsymbol{u}^k := (u_0^k, \ldots, u_{N+1}^k)$ including the dummy variables at the boundaries, we can rewrite the above equation with the help of the discrete Laplacian Δ_h and the identity operator I:

$$\begin{aligned} u_i^{k+1} \;&=\; u_i^k \,+\, \tau \cdot \frac{u_{i+1}^k \,-\, 2\,u_i^k \,+\, u_{i-1}^k}{h^2} \\ &=\; \big((I \,+\, \tau\Delta_h)\,\boldsymbol{u}^k\big)_i \qquad (i = 1, \ldots, N), \end{aligned} \tag{2.70}$$

Hence, the representation of the linear filter L_{2n+1}^h in Eq. (2.64) can be seen as a composition of n explicit linear diffusion steps with the varying time step sizes z_m^{-1}, $m = 0, \ldots, n{-}1$. The (total) *cycle time* θ_n that corresponds to the stopping time of this diffusion process is given by

$$\begin{aligned} \theta_n \;&=\; \sum_{m=0}^{n-1} z_m^{-1} \;=\; \alpha_1^{(n)} \\ &=\; h^2 \cdot \sum_{k=1}^{n} \left(\binom{k+1}{2} + \binom{k}{2} \right) w_k \;=\; h^2 \cdot \sum_{k=1}^{n} k^2\, w_k\,, \end{aligned} \tag{2.71}$$

where we have used Eq. (2.40). We summarise these results in the following theorem:

Theorem 2.9 (Filter Factorisation into Explicit Diffusion Steps). *Let L_{2n+1}^h be an arbitrary linear, symmetric 1-D filter kernel whose weights $w_{-n}, ..., w_n$ sum up to 1. Then L_{2n+1}^h is equivalent to a cycle of n explicit 1-D linear diffusion steps, i.e.*

$$L_{2n+1}^h \;=\; \prod_{m=0}^{n-1} (I \,+\, \tau_m \Delta_h)\;, \tag{2.72}$$

with the time step sizes

$$\tau_m \;=\; z_m^{-1} \in \mathbb{C}\,, \tag{2.73}$$

where $z_m \in \mathbb{C}\setminus\{0\}$ *are the roots of the symbol* $p_L(z)$*. The cycle time* θ_n *is given by*

$$\theta_n \;=\; h^2 \cdot \sum_{k=1}^{n} k^2\, w_k\,. \tag{2.74}$$

Note that the right hand side of Eq. (2.74) is just $\frac{1}{2}$ times the variance σ^2 of the discrete filter L^h_{2n+1}. It shows the well-known proportionality between the variance and the stopping time of the corresponding diffusion process. In this context, it makes sense to assume weights such that the variance is positive. This is in particular satisfied for filters having only non-negative weights.

2.7 Practical Examples

In this section, we discuss the application of Theorem 2.9 by means of the four discrete filters that we have presented above. To this end, we rewrite them as explicit diffusion schemes and analyse the corresponding symbols.

2.7.1 Binomial Kernel

Since we can represent the binomial kernel of length $(2n{+}1)h$ as a composition of n kernels with length $3h$, it is sufficient to consider only this special kernel. Equation (2.59) is the simplest way to compute the symbol. The weights are $w_0 = {}^1\!/\!_2\,,\; w_{-1} = w_1 = {}^1\!/\!_4$ and thus the symbol is given by

$$p^{[3]}_{Bin}(z) \;=\; \frac{1}{2} \;+\; \frac{1}{2}\cdot T_1\left(1 - \tfrac{h^2}{2}\,z\right) \;=\; 1 \;-\; \frac{h^2}{4}\,z\,. \tag{2.75}$$

If we consider the general case with length $(2n{+}1)h$, then we obtain

$$p^{[n]}_{Bin}(z) \;=\; \left(1 \;-\; \tfrac{h^2}{4}\,z\right)^n\,. \tag{2.76}$$

Because of $z \le \frac{4}{h^2}$, the symbol takes only non-negative values, which means that all frequency components do not change their sign or orientation and oscillations are impossible from the theoretical point of view.

Concerning the equivalence to explicit schemes, we can decompose a binomial kernel with arbitrary length $(2n{+}1)h$ into n explicit steps with the constant time step size $\tau = \frac{h^2}{4}$,

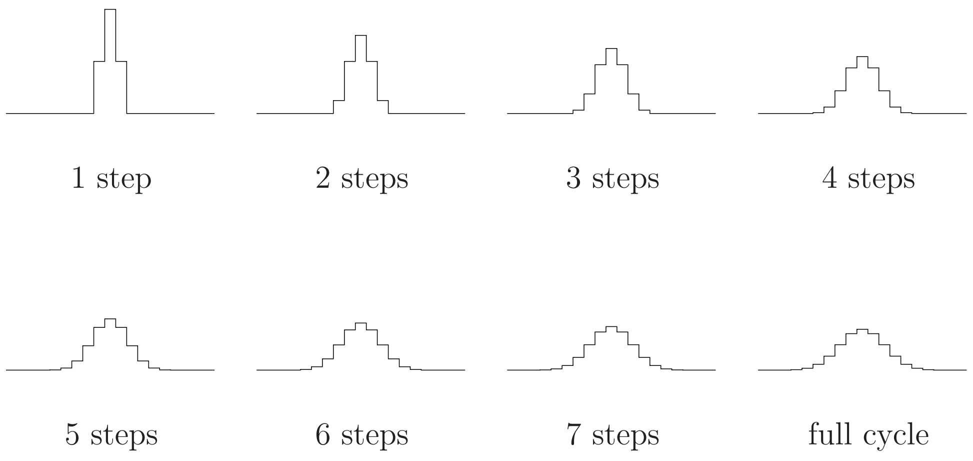


Figure 2.5: Evolution of the binomial kernel with $n = 8$ diffusion steps.

$$Bin^h_{2n+1} \;=\; \left(I \,+\, \tfrac{h^2}{4}\,\Delta_h\right)^n \,. \tag{2.77}$$

Hence, the cycle time θ_n grows linearly in the width n and is given by $\frac{h^2}{4}\cdot n$. The evolution of the binomial kernel with respect to the time steps of the explicit scheme is illustrated in Fig. 2.5. Actually, after each explicit time step we get a binomial kernel with increasing width.

2.7.2 Discrete Maximum Variance Kernel

Our second example is the linear filter V^h_{2n+1}, whose kernel reaches the maximum variance. Its weights are given by $w_k = 0$ for $|k| \leq n-1$ and $w_{-n} = w_n = {}^1\!/_2$. According to Eq. (2.40), the coefficients $\alpha^{(n)}_m$ satisfy

$$\begin{aligned}
\alpha^{(n)}_m \;&=\; \frac{h^{2m}}{2}\cdot\left(\binom{n+m}{2m} + \binom{n+m-1}{2m}\right)\\
&=\; \frac{h^{2m}}{2}\cdot\binom{n+m}{2m}\left(1 + \frac{n-m}{n+m}\right)\\
&=\; h^{2m}\cdot\frac{n}{n+m}\binom{n+m}{2m}\,.
\end{aligned} \tag{2.78}$$

Therefore, the symbol is

$$p_V^{[n]}(z) \;=\; \sum_{m=0}^{n} h^{2m} \cdot \frac{n}{n+m} \binom{n+m}{2m} (-z)^m \;, \tag{2.79}$$

and it is related to the Chebyshev polynomial T_{2n}. This can be seen as follows:

$$\begin{aligned} p_V^{[n]}(z) \;&=\; n \cdot \sum_{m=0}^{n} \frac{(-1)^m}{n+m} \binom{n+m}{2m} (h^2 \cdot z)^m \\ &=\; n \cdot \sum_{m=0}^{n} \frac{(-1)^{n-m}}{2n-m} \binom{2n-m}{2(n-m)} \left(h^2 \cdot z\right)^{n-m} \\ &=\; (-1)^n \cdot \frac{2n}{2} \cdot \sum_{m=0}^{\lfloor 2n/2 \rfloor} \frac{(-1)^m}{2n-m} \binom{2n-m}{m} \left(h \cdot \sqrt{z}\right)^{2n-2m} \\ &\overset{(2.51)}{=}\; (-1)^n \cdot T_{2n}\left(\tfrac{h\sqrt{z}}{2}\right) \;. \end{aligned} \tag{2.80}$$

Note that we have changed the order of summation ($m \to n-m$) in the second step. This result verifies Eq. (2.59):

$$p_V^{[n]}(z) \;=\; T_n\left(1 - \tfrac{h^2}{2}\, z\right) \;=\; T_n\left(-T_2\left(\tfrac{h\sqrt{z}}{2}\right)\right) \;=\; (-1)^n \cdot T_{2n}\left(\tfrac{h\sqrt{z}}{2}\right) \;, \tag{2.81}$$

where we have used the two well-known calculation rules

$$T_n(-x) \;=\; (-1)^n \cdot T_n(x) \tag{2.82}$$

and

$$T_n\left(T_m(x)\right) \;=\; T_{n \cdot m}(x) \tag{2.83}$$

for arbitrary $n, m \in \mathbb{N}_0$.

In order to construct the corresponding explicit scheme, we have to compute the roots of the Chebyshev polynomial $T_{2n}\left(\frac{h\sqrt{z}}{2}\right)$. The zeros of $T_{2n}(x)$ are known and given by

$$x_i \;=\; \cos\left(\pi \cdot \tfrac{2i+1}{4n}\right) \qquad (i = 0, \ldots, 2n-1). \tag{2.84}$$

Using the relation $z = \frac{4}{h^2} \cdot x^2$, the n roots z_m of the polynomial $T_{2n}\left(\frac{h\sqrt{z}}{2}\right)$ are given by the formula

$$z_m \;=\; \frac{4}{h^2} \cdot \cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right) \qquad (m = 0, \ldots, n-1). \tag{2.85}$$

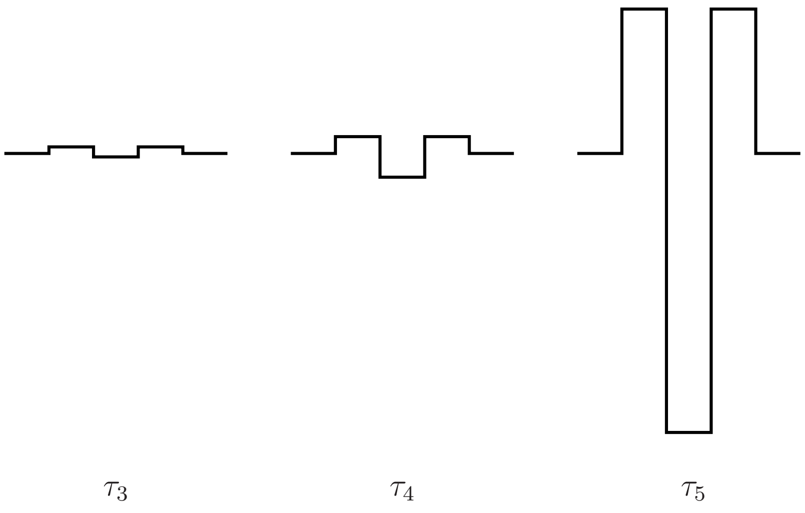


Figure 2.6: Unstable diffusion kernels of the scheme corresponding to V_{13}^h.

According to Theorem 2.9, the time step sizes τ_m for the corresponding cycle consisting of n explicit diffusion steps fulfil

$$\tau_m \;=\; z_m^{-1} \;=\; \frac{h^2}{4}\cdot\frac{1}{\cos^2\left(\pi\cdot\frac{2m+1}{4n}\right)} \qquad (m=0,\ldots,n-1). \qquad (2.86)$$

The cycle time is, due to Eq. (2.74), given by

$$\theta_n \;=\; \frac{h^2}{2}\cdot n^2\;, \qquad (2.87)$$

and therefore grows quadratically in the number of time step sizes n. This is n times the maximum diffusion time that a stable explicit scheme with constant time step size $\tau=\frac{h^2}{2}$ can reach. Hence, some of the time step sizes in Eq. (2.86) have to violate stability restrictions, though the whole cycle is stable. In this context, Fig. 2.6 illustrates the unstable convolution kernels $\left(\frac{\tau_m}{h^2}\,,\,1-2\cdot\frac{\tau_m}{h^2}\,,\,\frac{\tau_m}{h^2}\right)$, $m\geq 3$, belonging to V_{13}^h. The negative centre weights imply the violation of the stability restriction: $\tau_m>\frac{h^2}{2}$. However, the final result is a stable filter with non-negative weights. Figure 2.7 shows an example for the evolution of the filter kernel from step to step. Surprisingly, all eight intermediate kernels consist solely of non-negative weights.

Since we have $p_V^{[n]}(z)=\pm 1$ for some $z\neq 0$, this explicit scheme might have problems with higher frequencies, as we have already mentioned. It

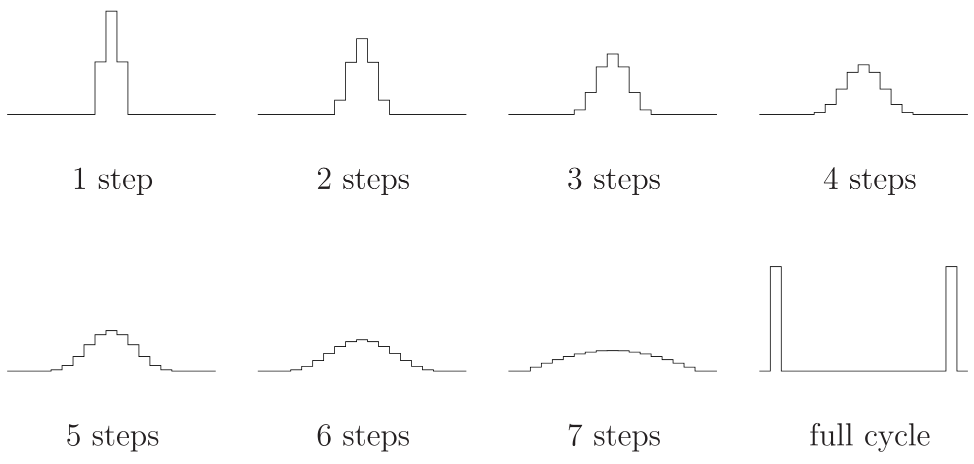


Figure 2.7: Evolution of the filter kernel for the explicit scheme using a cycle that ends up with the filter V_{17}^h.

can happen that such frequency components are not damped $(|p_V^{[n]}(z)| = 1)$ and additionally switch their orientation, i.e. are oscillating $(p_V^{[n]}(z) = -1)$. Later, we see that an explicit scheme using the time step sizes τ_m has already been introduced, but by means of another derivation.

Due to the above described problems with oscillations or insufficient damping, a so-called damping parameter $\nu > 0$ can be used. It ensures that the symbol of the method takes the value 1 only for $z = 0$. The corresponding modified polynomial is then given by

$$p_{V,\nu}^{[n]}(z) \;:=\; \frac{T_n\left(1 \,+\, \nu \,-\, \frac{h^2}{4}\,(2+\nu)\cdot z\right)}{T_n\,(1+\nu)}\;, \tag{2.88}$$

with the roots

$$z_m(\nu) \;=\; \frac{4}{h^2}\cdot\frac{\nu \,+\, 2\,\cos^2\left(\pi\cdot\frac{2m+1}{4n}\right)}{2\,+\,\nu} \qquad (m = 0,\ldots,n-1). \tag{2.89}$$

Regarding the time step sizes of the damped explicit scheme, we have

$$\tau_m(\nu) \;=\; \frac{h^2}{4}\cdot\frac{2\,+\,\nu}{\nu \,+\, 2\,\cos^2\left(\pi\cdot\frac{2m+1}{4n}\right)} \qquad (m = 0,\ldots,n-1). \tag{2.90}$$

Because of

$$\begin{aligned}\nu + 2\cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right) &= (\nu + 2)\cdot\cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right) \\ &\quad + \nu \cdot \left(1 - \cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right)\right) \\ &\geq (\nu + 2)\cdot\cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right)\ , \qquad (2.91)\end{aligned}$$

the step sizes fulfil

$$\tau_m(\nu) \leq \frac{h^2}{4}\cdot\frac{1}{\cos^2\left(\pi\cdot\frac{2m+1}{4n}\right)} \overset{(2.86)}{=} \tau_m\ . \qquad (2.92)$$

However, a larger damping parameter makes the method more robust with respect to higher frequencies. Thus, the damping parameter $\nu > 0$ can be seen as a trade-off between stability and the magnitude of the reached cycle time or variance. In this context, ν allows a continuous adjustment of the cycle time. For $\nu \to \infty$, the time step sizes $\tau_m(\nu)$ tend to $\frac{h^2}{4}$, i.e. the corresponding modified filter approximates a binomial kernel.

The influence of the damping parameter is illustrated in Fig. 2.8. It demonstrates two filter kernel evolutions with different damping parameters. As before, the intermediate kernels have non-negative weights. However, by introducing the damping parameter, the final results seem to be better approximations for Gaussians, in particular for larger ν. One can see that an increasing ν reduces the influence of the boundary weights, and thus makes the other weights larger.

2.7.3 Discrete Box Filter

Let us now reconsider the box filter B^h_{2n+1} with length $(2n+1)h$. As mentioned above, the weights are uniform and sum up to 1, $w_k = \frac{1}{2n+1}$ for all $k \in \{-n, \ldots, 0, \ldots n\}$. To compute the coefficients of the symbol, we use again Eq. (2.40): We have $\alpha_0^{(n)} = 1$ and for $m > 0$:

$$\begin{aligned}\alpha_m^{(n)} &= \frac{h^{2m}}{2n+1}\cdot\sum_{k=m}^{n}\left(\binom{k+m}{2m} + \binom{k+m-1}{2m}\right) \\ &= \frac{h^{2m}}{2n+1}\cdot\left(\sum_{k=2m}^{n+m}\binom{k}{2m} + \sum_{k=2m}^{n+m-1}\binom{k}{2m}\right) \\ &= \frac{h^{2m}}{2n+1}\cdot\left(\binom{n+m+1}{2m+1} + \binom{n+m}{2m+1}\right)\end{aligned}$$

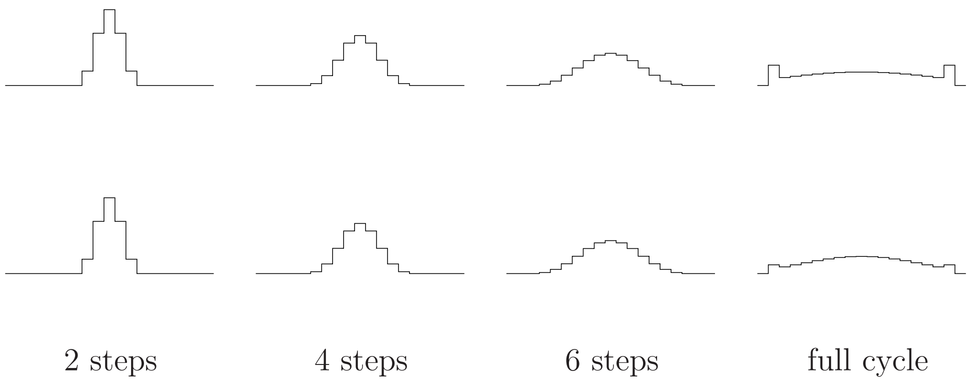


Figure 2.8: Evolution of filter kernels for the damped scheme with $n = 8$ explicit diffusion steps. **Top row**: $\nu = 1/20$. **Bottom row**: $\nu = 1/10$.

$$= \frac{h^{2m}}{2n+1} \cdot \left(\frac{n+m+1}{2m+1} \binom{n+m}{2m} + \frac{n-m}{2m+1} \binom{n+m}{2m} \right)$$

$$= \frac{h^{2m}}{2n+1} \cdot \frac{2n+1}{2m+1} \binom{n+m}{2m} = \frac{h^{2m}}{2m+1} \cdot \binom{n+m}{2m} . \tag{2.93}$$

Thus, the symbol $p_B^{[n]}(z)$ of the box filter B_{2n+1}^h is given by the polynomial

$$p_B^{[n]}(z) = \sum_{m=0}^{n} \frac{h^{2m}}{2m+1} \binom{n+m}{2m} (-z)^m . \tag{2.94}$$

Using Eq. (2.59) yields

$$p_B^{[n]}(z) = \frac{1}{2n+1} \cdot \left(1 + 2 \cdot \sum_{i=1}^{n} T_i \left(1 - \tfrac{h^2}{2} z \right) \right) \tag{2.95}$$

as a representation for the symbol. However, we can write this equation without any sum. To this end, we consider the closed-form representation of the Chebyshev polynomial $T_{2n+1}(x)$:

$$T_{2n+1}(x) = \frac{2n+1}{2} \cdot \sum_{m=0}^{n} \frac{(-1)^m}{2n+1-m} \binom{2n+1-m}{2(n-m)+1} (2x)^{2(n-m)+1}$$

$$= \frac{2n+1}{2} \cdot \sum_{m=0}^{n} \frac{(-1)^{n-m}}{n+m+1} \binom{n+m+1}{2m+1} (2x)^{2m+1}$$

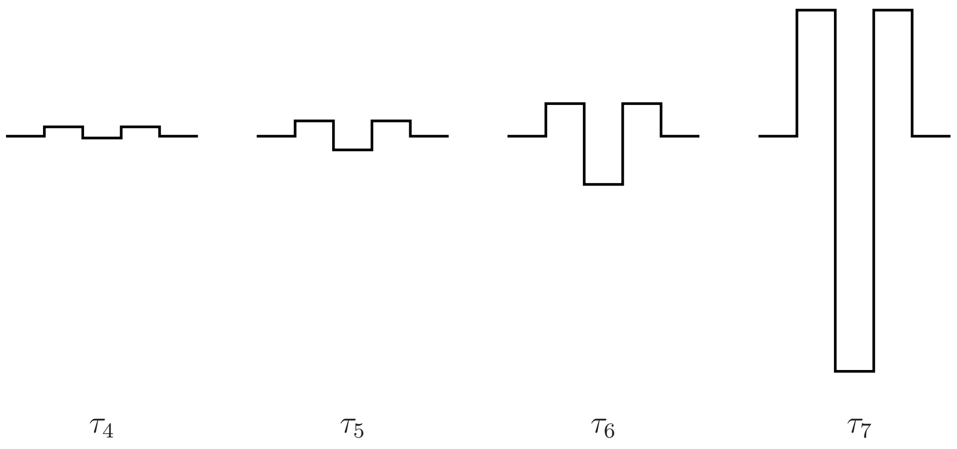


Figure 2.9: Unstable diffusion kernels of the box filter scheme ($n = 8$).

$$= (-1)^n (2n+1)x \cdot \sum_{m=0}^{n} \frac{(-1)^m}{2m+1} \binom{n+m}{2m} \left(4x^2\right)^m . \qquad (2.96)$$

With $x = \frac{h\sqrt{z}}{2}$ we obtain for $z > 0$

$$p_B^{[n]}(z) = (-1)^n \cdot \frac{2}{2n+1} \cdot \frac{T_{2n+1}\left(\frac{h\sqrt{z}}{2}\right)}{h\sqrt{z}} , \qquad (2.97)$$

which is related to the so-called *Dirichlet kernel* [39]. Note that this representation also makes sense for $z = 0$, because the limit value of the right hand side for $z \to 0$ exists and is equal to 1. Compared to the filter V_{2n+1}^h, the symbol of the box filter does not take the values ± 1 for $z \in (0, \frac{4}{h^2}]$. This can be seen with the help of the representation in Eq. (2.95). The smallest negative value that the symbol could take is $-\frac{2n-1}{2n+1} > -1$, assuming all T_i would take simultaneously the value -1. If we assume that the symbol takes the value 1, this can only happen if all Chebyshev polynomials T_i take the value 1 at the same point. Considering in particular $T_1\left(1 - \frac{h^2}{2}z\right)$ gives us only $z = 0$. Thus, $z > 0$ implies

$$|p_B^{[n]}(z)| < 1 , \qquad (2.98)$$

and an additional damping parameter is not necessary.

The n positive roots $z_0, \ldots, z_{n-1}$ of $p_B^{[n]}(z)$ coincide with the roots of the Chebyshev polynomial $T_{2n+1}\left(\frac{h\sqrt{z}}{2}\right)$, which means they have to satisfy

$$\frac{h\sqrt{z_m}}{2} = \cos\left(\pi \cdot \tfrac{2m+1}{4n+2}\right) \qquad (m = 0, \ldots, n-1). \tag{2.99}$$

This yields

$$z_m = \frac{4}{h^2} \cdot \cos^2\left(\pi \cdot \tfrac{2m+1}{4n+2}\right) \qquad (m = 0, \ldots, n-1), \tag{2.100}$$

and therefore the time step sizes of the corresponding explicit scheme are given by

$$\tau_m = \frac{h^2}{4} \cdot \frac{1}{\cos^2\left(\pi \cdot \tfrac{2m+1}{4n+2}\right)} \qquad (m = 0, \ldots, n-1). \tag{2.101}$$

The stopping time of a cycle with n time steps

$$\begin{aligned} \theta_n &= \frac{h^2}{2n+1} \cdot \sum_{k=1}^{n} k^2 \\ &= \frac{h^2}{2n+1} \cdot \frac{n(n+1)(2n+1)}{6} = h^2 \cdot \frac{n^2+n}{6} \end{aligned} \tag{2.102}$$

also grows quadratically in n and is $\frac{n+1}{3}$ times the maximum stopping time of a stable explicit scheme with the constant stable time step size $\tau = \frac{h^2}{2}$. Thus, some of the time step sizes in Eq. (2.101) have to be larger than the 1-D stability limit $\frac{h^2}{2}$. Actually, we show in Chapter 3 that half of these time step sizes are unstable, which means that every stable step allows an unstable one. An example with four unstable time steps is illustrated in Fig. 2.9. The four convolution kernels have negative centre weights that correspond to unstable explicit time steps. Figure 2.10 shows the respective kernel evolution. Despite the four unstable time steps, all intermediate kernels have non-negative weights.

To conclude this subsection, we analyse the interesting question whether the time step sizes in Eq. (2.101) can be reproduced by choosing a suitable damping factor $\nu > 0$ in Eq. (2.90). More precisely, this would mean that the box filter corresponds to a damped maximum variance filter. By equating the formulas Eq. (2.101) and Eq. (2.90) for the time step sizes, we

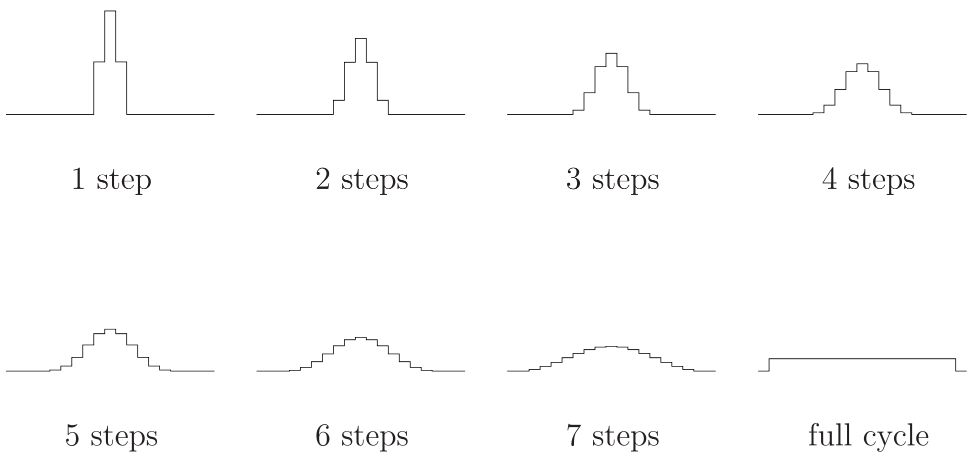


Figure 2.10: Evolution of the filter kernel for the box filter scheme with $n = 8$ explicit diffusion steps.

obtain

$$\begin{aligned}\nu &= \frac{\cos^2\left(\pi \cdot \frac{2m+1}{4n}\right) - \cos^2\left(\pi \cdot \frac{2m+1}{4n+2}\right)}{\cos^2\left(\pi \cdot \frac{2m+1}{4n+2}\right) - 1} \\ &= \frac{\sin^2\left(\pi \cdot \frac{2m+1}{4n}\right)}{\sin^2\left(\pi \cdot \frac{2m+1}{4n+2}\right)} - 1 \,, \end{aligned} \tag{2.103}$$

with $m = 0, \ldots, n-1$. This can only be valid, if the right hand side does not depend on m, i.e. is constant. In particular, the case $n = 1$ $(m = 0)$ allows the choice of a suitable damping parameter such that the damped maximum variance filter is a box filter. However, for $n > 1$ the right hand side is not constant with respect to m, which means that there is no damping parameter such that the damped maximum variance kernel turns into a box kernel. Hence, the explicit schemes corresponding to the box filter with cycle length $n \geq 2$ do not belong to the class of schemes implied by damped maximum variance filters.

2.7.4 Extended Box Filter

Our last example is the extended box filter (EBF) E^h_{2n+1} of Gwosdek et al. [167]. It is just a convex combination of the two previous examples. Depending on the parameter $\gamma \in (0,1)$, the weights of the filter E^h_{2n+1}

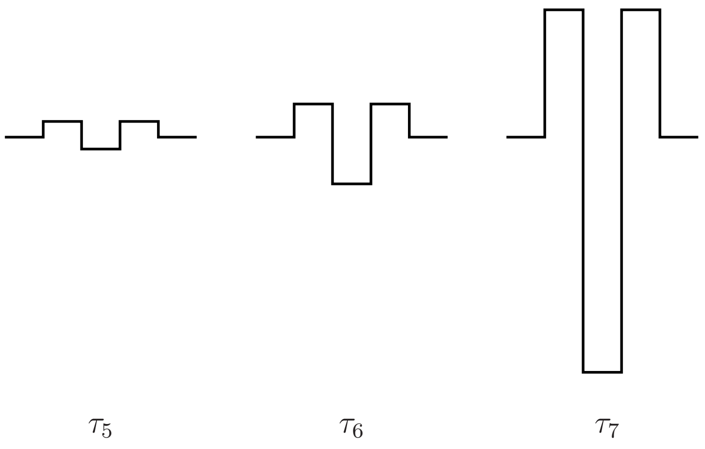


Figure 2.11: Unstable diffusion kernels of the extended box filter scheme ($\gamma = 0.94$, $n = 8$).

satisfy

$$w_k = \begin{cases} \frac{\gamma}{2n-1} & , \ |k| \le n-1 \\ \frac{1-\gamma}{2} & , \ k = \pm n \end{cases} . \tag{2.104}$$

The representation from Eq. (2.59) yields for the symbol $p_E^{[n]}(z)$:

$$\begin{aligned} p_E^{[n]}(z) &= \frac{\gamma}{2n-1} \cdot \left(1 + 2 \cdot \sum_{i=1}^{n-1} T_i\left(1 - \tfrac{h^2}{2}\, z\right)\right) \\ &\quad + (1-\gamma) \cdot T_n\left(1 - \tfrac{h^2}{2}\, z\right) \qquad (2.105) \\ &= \gamma \cdot \frac{(-1)^{n-1}\, 2 \cdot T_{2n-1}\left(\frac{h\sqrt{z}}{2}\right)}{(2n-1)h\sqrt{z}} + (1-\gamma) \cdot (-1)^n\, T_{2n}\left(\tfrac{h\sqrt{z}}{2}\right) . \end{aligned}$$

Actually, the extended box filter can be seen as a convex combination of two different explicit schemes with $n-1$ and n varying time steps, respectively. The corresponding cycle time is also a convex combination of two quadratic polynomials in n and thus grows quadratically in n. Moreover, it depends on the parameter γ and so it can be tuned in a continuous way. However, the computation is more expensive than before, since we now have $2n-1$ time steps for the two cycles. To this end, the question is whether we can reduce the effort. In this context, the following proposition states that the

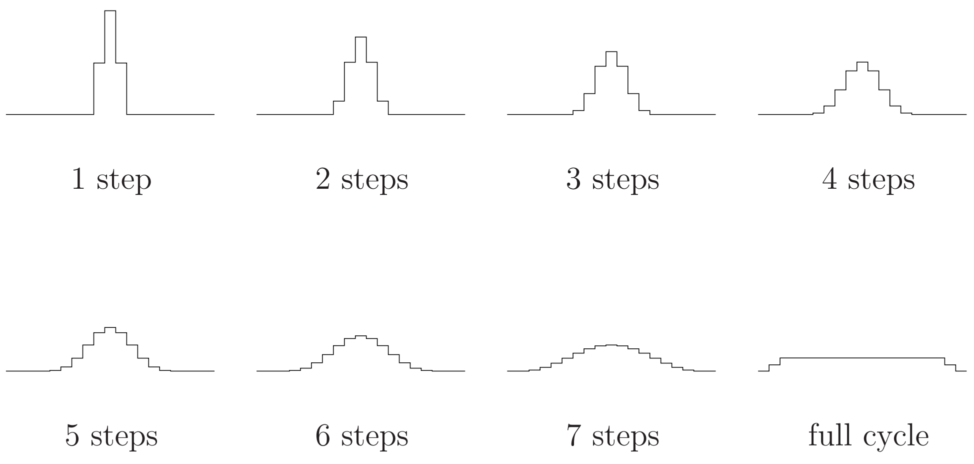


Figure 2.12: Evolution of the filter kernel for the extended box filter scheme ($\gamma = 0.94$) with $n = 8$ explicit diffusion steps.

combination of both schemes is even equivalent to a cheaper cyclic scheme with n time steps.

Proposition 2.10 (**Existence of Cyclic Scheme for EBF**). *The symbol $p_E^{[n]}(z)$ of an EBF E_{2n+1}^h has n real-valued roots $z_0, \ldots, z_{n-1}$, i.e. the EBF can be written as one cycle:*

$$E_{2n+1}^h \;=\; \prod_{m=0}^{n-1} \left(I \,+\, z_m^{-1}\,\Delta_h \right) \;. \tag{2.106}$$

Proof. Let $\gamma \in (0,1)$ and $n \geq 2$, because the case $n = 1$ is trivial. According to Eq. (2.105), the polynomial $p_E^{[n]}(z)$ fulfils

$$p_E^{[n]}(z) \;=\; \gamma \cdot q_{n-1}(z) \;+\; (1-\gamma) \cdot r_n(z) \;, \tag{2.107}$$

with the $n-1$ degree polynomial $q_{n-1}(z)$ and the n degree polynomial $r_n(z)$. We know that both polynomials have only real-valued zeros: The roots of q_{n-1} are related to the $n-1$ positive roots of the Chebyshev polynomial T_{2n-1} and the zeros of r_n to the n positive roots of T_{2n}. They are given by

$$z_k^{(q)} \;=\; \frac{4}{h^2} \cdot \cos^2\left(\pi \cdot \tfrac{2k+1}{4n-2}\right) \qquad (k = 0, \ldots, n-2) \tag{2.108}$$

for q_{n-1} and

$$z_m^{(r)} \;=\; \frac{4}{h^2} \cdot \cos^2\left(\pi \cdot \tfrac{2m+1}{4n}\right) \qquad (m = 0, \ldots, n-1) \tag{2.109}$$

in the case of r_n. For $0 \le j \le n-2$ we have

$$\frac{1}{2} \;>\; \frac{2j+3}{4n} \;>\; \frac{2j+1}{4n-2} \;>\; \frac{2j+1}{4n} \;>\; 0\;, \tag{2.110}$$

and by applying the monotonically decreasing squared cosine function $\cos^2 : \left[0, \frac{\pi}{2}\right] \to [0,1]$, we obtain

$$\cos^2\left(\pi \cdot \tfrac{2j+3}{4n}\right) \;<\; \cos^2\left(\pi \cdot \tfrac{2j+1}{4n-2}\right) \;<\; \cos^2\left(\pi \cdot \tfrac{2j+1}{4n}\right)\;. \tag{2.111}$$

This is equivalent to

$$z_{j+1}^{(r)} \;<\; z_j^{(q)} \;<\; z_j^{(r)} \tag{2.112}$$

for $j \in \{0, \ldots, n-2\}$. Since the multiplicity of all roots is equal to one, the graphs of both q_{n-1} and r_n change their sign only at those points. We have $q_{n-1}(0) = r_n(0) = 1$, which means that they have the same sign in the interval $[0, z_{n-1}^{(r)})$. If we consider the interval $(z_{n-1}^{(r)}, z_{n-2}^{(q)})$, r_n is negative and q_{n-1} still positive. However, for $z \in (z_{n-2}^{(q)}\,, z_{n-2}^{(r)})$ both $r_n(z)$ and $q_{n-1}(z)$ are negative. By continuing these thoughts, we observe that $r_n(z)$ as well as $q_{n-1}(z)$ have the same sign for $z \in (z_j^{(q)}\,, z_j^{(r)})$ and it is equal to $(-1)^{n-j-1}$. This is also valid for any convex combination of r_n and q_{n-1}, in particular for the symbol $p_E^{[n]}(z)$. Hence, its sign changes $n-1$ times. Because of the continuity, $p_E^{[n]}(z)$ has at least $n-1$ real-valued zeros. However, the polynomial $p_E^{[n]}(z)$ has degree n, and therefore it has an additional root. Due to the real-valued coefficients of $p_E^{[n]}(z)$, this root has to be also real-valued. Thus, the proposition is proven. $\square$

The biggest advantage of this explicit scheme is the continuous tuning of the cycle time. As we have mentioned above, it depends quadratically on n, i.e. unstable time steps have to exist within a cycle. However, it is very difficult to compute the time step sizes, because there seems to be no closed-form representation for the zeros of $p_E^{[n]}(z)$. Figure 2.11 depicts the three unstable explicit diffusion kernels for the case $n = 8$ and $\gamma = 0.94$. In contrast to the box filter, we have one more stable kernel with respect to this parameter setting. The corresponding kernel evolution is depicted in Fig. 2.12. Also here, the intermediate filter kernels consist solely of non-negative weights.

Finally, we show a comparison of the explicit diffusion kernels for the last step of a cycle with 8 steps to visualise the different magnitudes of the negative centre weights in Fig. 2.13. As expected, the kernel of the maximum variance filter V_{17}^h is the most unstable one.

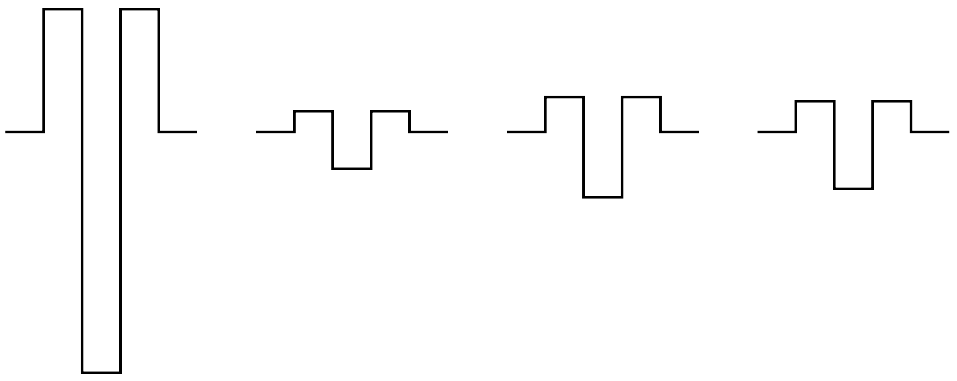


Figure 2.13: Explicit diffusion kernels with respect to the largest time step size ($n = 8$). **From left to right:** Maximum variance, damped maximum variance ($\nu = 0.1$), box filter and extended box filter kernel ($\gamma = 0.94$).

2.8 Summary

In this chapter, we have reached our first goal: We have shown for the discrete setting that 1-D linear, symmetric filters are equivalent to 1-D explicit diffusion schemes. To reach this goal, we have rewritten the filters in terms of a sum of discrete even order derivatives, or better said, polynomials in the discrete Laplacian Δ_h. The factorisation of these polynomials yields linear factors that correspond to 1-D explicit diffusion steps. More precisely, we have found a suprising access to a class of explicit schemes for solving diffusion processes by means of a signal processing background.

Replacing the discrete operator Δ_h by $-z$, which is related to a Z-transform, allows us to analyse the behaviour of the filter with respect to different frequency components by means of the symbol or amplification factor. Since this approach is very similar to Fourier analysis, we have shown a connection that has helped us to find closed-form expressions for the symbols: They are related to Chebyshev polynomials of the first kind.

With the help of this theory, we have decomposed four linear filters into explicit diffusion steps. Interestingly, three of them use unstable time step sizes within their corresponding explicit schemes. The first filter kernel V^h_{2n+1} reaches the maximum variance of a stable, symmetric filter with the length $(2n+1)h$, but it is very sensitive regarding higher frequency components of the input signal. Furthermore, we have seen that it is not very suited for an iterative approximation of a Gaussian convolution. That is why it requires an additional damping parameter $\nu > 0$ that improves the behaviour of the filter with respect to high frequencies and the accuracy for the iterative application. On the other hand, it decreases the variance of the Gaussian, or equivalently, the cycle time θ_n of the diffusion process. In this context, the parameter ν can also be used to reach arbitrary θ_n. However, the filtering results depend on this parameter, because different damping parameters ν yield different filter kernels. An increasing damping parameter decreases the weights at the boundaries of the filter kernel and improves the influence of the other weights. For $\nu \to \infty$, the maximum variance kernel tends to the well-known binomial kernel that corresponds to an explicit scheme using a constant, stable time step size.

In contrast to V^h_{2n+1}, the explicit scheme related to the box filter kernel B^h_{2n+1} does not need such a damping parameter. It also uses unstable time step sizes to reach a cycle time that grows quadratically in the number of time steps. However, the iterative application yields good approximations for a Gaussian convolution, and they are much better than the results with V^h_{2n+1}. Actually, the signal processing background has helped us here to get rid of the above mentioned damping parameter. This shows that the proposed access has a big advantage in contrast to the classical derivation which is presented in the following chapter.

The convex combination of both V^h_{2n+1} and B^h_{2n-1} yields the extended box filter kernel E^h_{2n+1}. Here, the weight $\gamma \in (0,1)$ can be seen as some kind of a damping parameter. Moreover, the extended box filter kernel allows the cycle time to be tuned continuously, since it depends on γ. Unfortunately, the corresponding time step sizes of the extended box filter scheme can be very difficult to determine, if one wants to use a more efficient cycle with n explicit steps instead of two cycles with a total of $2n-1$ time steps. However, in the next chapter, we are going to present a solution for the box filter that can yield arbitrary cycle times.

Chapter 3

Fast Explicit Diffusion (FED)

If everything seems under control,
you're just not going fast enough.
Mario Andretti

Similar to the last chapter, the current chapter is also based on our publications [164, 165]. Its goal is a detailed theoretical and experimental analysis of the explicit scheme representing the box filter kernel B^h_{2n+1}. It is a good choice, since it is more accurate than V^h_{2n+1}, does not need any additional damping or weighting parameter, and in contrast to the extended box filter kernel E^h_{2n+1} we know an easy closed-form representation for the time step sizes. The only disadvantage is the quantisation of the cycle times. However, we are going to propose a strategy for the solution of this problem. Furthermore, we show that the corresponding cyclic explicit scheme is useful for the solution of arbitrary time-dependent parabolic (diffusion-like) problems such as image enhancement. We also present cyclic methods for elliptic PDEs related to image reconstruction (inpainting) or the minimisation of energy functionals that can appear in the context of image regularisation or computer vision problems.

Actually, the idea of using varying parameters within explicit schemes can build upon existing research. Thus, we first discuss some related work.

3.1 Related Work

We have introduced explicit schemes that come from decompositions of linear filters. They use varying time step sizes, where some of them violate the theoretical stability limit. A similar method for parabolic problems has been proposed by Yuan'Chzhao-Din [162], Saul'jev [117], Franklin [53], and Guillou and Lago [64]. Later, Gentzsch and Schlüter [56, 57] have used this idea under the name *Super Time Stepping (STS)*. Contrary to our derivation, they have used a direct approach for explicit schemes: Find a set of different time step sizes, which keeps stability after each cycle, and at the same time maximises the stopping time of such a cycle. Instead of factorising a box filter, their method can be interpreted as the factorisation of V^h_{2n+1}, i.e. $(1/2, 0, \ldots, 0, 1/2)$, which we have already presented and analysed in Chapter 2.

Actually, the basic concepts that Yuan'Chzhao-Din, Saul'yev, Franklin and Guillou and Lago made use of, come from iterative methods for the solution of linear systems

$$\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{c}\,, \tag{3.1}$$

with $\boldsymbol{B} \in \mathbb{C}^{N\times N}$, the known right hand side $\boldsymbol{c} \in \mathbb{C}^N$ and the unknown solution $\boldsymbol{x} \in \mathbb{C}^N$. In 1910, L.F. Richardson [111] proposed to use a simple, explicit fixed point iteration for the equivalent system

$$\boldsymbol{x} \;=\; \boldsymbol{x} \;+\; \boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}\,. \tag{3.2}$$

It is given by

$$\boldsymbol{x}^{k+1} \;=\; \boldsymbol{x}^k \;+\; \left(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^k\right) \;=\; (\boldsymbol{I} \;-\; \boldsymbol{B})\,\boldsymbol{x}^k \;+\; \boldsymbol{c} \qquad (k \geq 0)\,, \tag{3.3}$$

where $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ denotes the identity matrix and $\boldsymbol{x}^0 \in \mathbb{C}^N$ is an arbitrary initial vector. We now assume that the eigenvalues of $\boldsymbol{B}$ are real-valued and positive. Then the convergence of the fixed point iteration is guaranteed if the moduli of the eigenvalues of $\boldsymbol{I}-\boldsymbol{B}$ are smaller than 1. This means that the largest eigenvalue $\lambda_{\max}$ of $\boldsymbol{B}$ must be smaller than 2. To speed up the convergence, Richardson proposed to use so-called relaxation parameters $\omega_k \in \mathbb{R}$ that may vary from one iteration to another:

$$\boldsymbol{x}^{k+1} \;=\; \boldsymbol{x}^k \;+\; \omega_k \cdot \left(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^k\right) \qquad (k \geq 0)\,. \tag{3.4}$$

For a constant relaxation parameter $\omega_k = \omega$, Eq. (3.4) is stable with

$$0 \;\leq\; \omega \;\leq\; \frac{2}{\lambda_{\max}}\,. \tag{3.5}$$

If the smallest eigenvalue of $\boldsymbol{B}$ is given by $\lambda_{\min} > 0$, it has been shown e.g. in [115] that the optimal convergence with a constant relaxation parameter is reached for

$$\omega = \frac{2}{\lambda_{\min} + \lambda_{\max}} . \tag{3.6}$$

However, varying parameters allow even a much better convergence. In this context, Richardson considered the homogeneous system

$$\boldsymbol{B}\boldsymbol{x} = \boldsymbol{0} \tag{3.7}$$

with the zero vector $\boldsymbol{0} \in \mathbb{R}^N$. The fixed point iteration in Eq. (3.4) then reads as

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k - \omega_k \cdot \boldsymbol{B}\boldsymbol{x}^k = (\boldsymbol{I} - \omega_k \boldsymbol{B})\, \boldsymbol{x}^k \\ &= \left(\prod_{i=0}^{k} (\boldsymbol{I} - \omega_i \boldsymbol{B}) \right) \boldsymbol{x}^0 , \end{aligned} \tag{3.8}$$

If $\boldsymbol{x}^0$ is an eigenvector $\boldsymbol{v}_m$ of $\boldsymbol{B}$, then the product at the right hand side is a polynomial of degree $k{+}1$ in the corresponding eigenvalue λ_m, i.e.

$$q(\lambda_m) := \prod_{i=0}^{k} (1 - \omega_i \lambda_m) \qquad (m = 1, \ldots, N) . \tag{3.9}$$

An optimal convergence to the solution $\boldsymbol{x} = \boldsymbol{0}$ means that the polynomial $q(\cdot)$ should be as small as possible for all eigenvalues.

In his paper, Richardson proposed to choose the relaxation parameters ω_i in a way such that the inverse values $(\omega_i)^{-1}$ are quite uniformly distributed. However, this choice might not be optimal for the convergence. It seems to be very likely that he was not aware of the works done by Chebyshev [30] in 1854 and Markoff [97] in 1892 about polynomials deviating least from zero. We have already introduced such polynomials, namely the Chebyshev polynomials of the first kind, in the previous chapter. In 1950, Flanders and Shortley [50] made use of Chebyshev polynomials to obtain better convergence for the solution of eigenvalue problems. They used an approach similar to Richardson's method. A few years later, the optimality of Chebyshev polynomials was applied to the solution of linear systems by Lanczos [86], Shortley [125] and Young [160]. In contrast to Shortley and Young, Lanczos presented an approach that differs from Richardson's method and is more complicated. Shortley proposed to use slightly modified Chebyshev polynomials $S_{k+1}(\boldsymbol{B})$ of degree $k+1$, i.e.

$$\boldsymbol{x}^{k+1} \;=\; S_{k+1}(\boldsymbol{B})\,\boldsymbol{x}^0 \;=\; \left(\sum_{i=0}^{k+1} C_i \cdot \boldsymbol{B}^i\right)\boldsymbol{x}^0\ , \tag{3.10}$$

with the coefficients $C_i \in \mathbb{R}$, $C_0 = 1$. The main disadvantage of this method is that one has to compute all $k+1$ matrix powers $\boldsymbol{B}^i$ and store the results of the matrix-vector products $\boldsymbol{B}^i\boldsymbol{x}^0$. However, the polynomial $S_{k+1}(z)$ has $k+1$ positive zeros $z_0, \ldots, z_k$ and therefore $S_{k+1}(\boldsymbol{B})$ can be written as a product of matrices

$$S_{k+1}(\boldsymbol{B}) \;=\; \prod_{i=0}^{k}\left(\boldsymbol{I} \,-\, z_i^{-1}\,\boldsymbol{B}\right)\ . \tag{3.11}$$

Hence, Eq. (3.10) is equivalent to Richardson's cyclic method (3.8) if one chooses the relaxation parameters $\omega_i = z_i^{-1}$ like Young in [160]. This means we do not have to compute and store the matrix powers $\boldsymbol{B}^i$ anymore.

Young considered linear systems (3.1) with a symmetric and positive definite matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ whose eigenvalues λ_m range in $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\max} \geq \lambda_{\min} > 0$. For $\lambda_{\max} > \lambda_{\min}$, the bijective function

$$\Phi_{\lambda_{\min},\lambda_{\max}}(z) \;:=\; \frac{\lambda_{\max} \,+\, \lambda_{\min} \,-\, 2z}{\lambda_{\max} \,-\, \lambda_{\min}} \tag{3.12}$$

maps the eigenvalues on the interval $[-1, 1]$, in which the Chebyshev polynomials are bounded by ± 1. Moreover, the polynomial $S_{k+1}(z)$ must fulfil $S_{k+1}(0) = 1$, in order to satisfy the factorisation property in Eq. (3.11). If we define for $z \in [\lambda_{\min}, \lambda_{\max}]$

$$S_{k+1}(z) \;:=\; \frac{T_{k+1}\left(\Phi_{\lambda_{\min},\lambda_{\max}}(z)\right)}{T_{k+1}\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)}\ , \tag{3.13}$$

then $S_{k+1}(0) = 1$ and $|S_{k+1}(\lambda_m)| < 1$ for all eigenvalues λ_m. More precisely, the upper bound is given by

$$\max_{m=1,\ldots,N} |S_{k+1}(\lambda_m)| \;\leq\; \frac{1}{\left|T_{k+1}\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)\right|} \;<\; 1\ . \tag{3.14}$$

The roots z_i of $S_{k+1}(z)$ fulfil

$$\Phi_{\lambda_{\min},\lambda_{\max}}(z_i) \;=\; \cos\left(\pi \cdot \tfrac{2i+1}{2k+2}\right) \qquad (i = 0, \ldots, k). \tag{3.15}$$

This is equivalent to

$$z_i \;=\; \frac{\lambda_{\max} + \lambda_{\min}}{2} \;-\; \frac{\lambda_{\max} - \lambda_{\min}}{2} \cdot \cos\left(\pi \cdot \tfrac{2i+1}{2k+2}\right) \;. \tag{3.16}$$

Thus, the optimal relaxation parameters are given by

$$\omega_i \;=\; \frac{2}{\lambda_{\max} + \lambda_{\min} \;-\; (\lambda_{\max} - \lambda_{\min}) \cdot \cos\left(\pi \cdot \tfrac{2i+1}{2k+2}\right)} \;. \tag{3.17}$$

The convergence behaviour follows from the estimation (3.14). We obtain

$$\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} \;=\; \frac{\frac{\lambda_{\max}}{\lambda_{\min}} + 1}{\frac{\lambda_{\max}}{\lambda_{\min}} - 1} \;=\; \frac{1}{2}\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1} \;+\; \frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right) \;, \tag{3.18}$$

and because of

$$T_{k+1}\left(\tfrac{1}{2}\left(s \;+\; \tfrac{1}{s}\right)\right) \;=\; \frac{1}{2}\left(s^{k+1} \;+\; \frac{1}{s^{k+1}}\right) \;, \tag{3.19}$$

we finally have

$$\begin{aligned}
\frac{1}{\left|T_{k+1}\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)\right|} \;&=\; \left|2\left(\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}\right)^{k+1} \;+\; \left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right)^{k+1}\right)^{-1}\right| \\
&\leq\; 2\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right)^{k+1} \;.
\end{aligned} \tag{3.20}$$

In the case of $\lambda_{\max} = \lambda_{\min}$, the optimal relaxation parameter is constant and it is given by $\omega_i \;=\; \frac{1}{\lambda_{\max}}$. This leads to the polynomial

$$q(z) \;=\; \left(1 \;-\; \tfrac{1}{\lambda_{\max}}\, z\right)^{k+1} \;, \tag{3.21}$$

which satisfies

$$\max_{m=1,\ldots,N} |q(\lambda_m)| \;=\; 0 \;=\; 2\left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right)^{k+1} \;. \tag{3.22}$$

Note that the ratio $\frac{\lambda_{\max}}{\lambda_{\min}}$ corresponds to the so-called *condition number* of $\boldsymbol{B}$ [115, 132]

$$\kappa_2(\boldsymbol{B}) \;:=\; \|\boldsymbol{B}\|_2 \cdot \left\|\boldsymbol{B}^{-1}\right\|_2 \;\geq\; 1 \tag{3.23}$$

with respect to the standard Euclidean norm $\|\cdot\|_2$. Linear systems with a small condition number are called *well-conditioned* and are easier to solve than *ill-conditioned* systems with very large condition numbers. According to Eq. (3.22) the convergence rate of Richardson's cyclic method with the optimal relaxation parameters depends on the square root of the condition number,

$$2\left(\frac{\sqrt{\kappa_2(\boldsymbol{B})}\,-\,1}{\sqrt{\kappa_2(\boldsymbol{B})}\,+\,1}\right)^{k+1} . \tag{3.24}$$

It is the same upper bound as for the well-known conjugate gradient method proposed by Hestenes and Stiefel [74].

The transfer to the parabolic case can be seen as follows [117]: Instead of solving an elliptic problem like Eq. (3.7) with a symmetric, positive definite matrix $\boldsymbol{B}$, one introduces an artificial time variable t transforming the problem into its parabolic analogon

$$\frac{d\boldsymbol{x}}{dt} \;=\; -\boldsymbol{B}\boldsymbol{x}\;, \tag{3.25}$$

where the vector $\boldsymbol{x} = \boldsymbol{x}(t)$ is now time-dependent. An explicit time discretisation with the time step size $\tau > 0$, i.e.

$$\frac{\boldsymbol{x}^{j+1}\,-\,\boldsymbol{x}^{j}}{\tau} \;=\; -\boldsymbol{B}\boldsymbol{x}^{j}\;, \tag{3.26}$$

with $\boldsymbol{x}^j \approx \boldsymbol{x}(j\,\tau)$ yields the scheme

$$\boldsymbol{x}^{j+1} \;=\; (\boldsymbol{I}\,-\,\tau\,\boldsymbol{B})\,\boldsymbol{x}^{j} \qquad (j \geq 0)\,. \tag{3.27}$$

It can be seen as Richardson's method for the homogeneous linear system (3.7) with the relaxation parameter τ. By using varying time step sizes $\tau_i,\ i = 0, \ldots, j$, we obtain

$$\boldsymbol{x}^{j+1} \;=\; \left(\prod_{i=0}^{j}(\boldsymbol{I}\,-\,\tau_i\,\boldsymbol{B})\right)\boldsymbol{x}^{0}\,. \tag{3.28}$$

This is equivalent to Richardson's cyclic method (3.8) with j iterations. According to Eq. (3.17) we can use the time step sizes

$$\tau_i \;=\; \frac{2}{(\lambda_{\max}+\lambda_{\min})\;-\;(\lambda_{\max}-\lambda_{\min})\cdot\cos\left(\pi\cdot\frac{2i+1}{2j+2}\right)} \qquad (i = 0, \ldots, j)\,, \tag{3.29}$$

and end up with the scheme proposed in [53, 64, 117, 162]. For the time step sizes it holds that

$$\begin{aligned}
\tau_i &= \frac{1}{\lambda_{\max} - \lambda_{\min}} \cdot \frac{2}{\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}} - \cos\left(\pi \cdot \frac{2i+1}{2j+2}\right)} \\
&= \frac{1}{\lambda_{\max}} \cdot \frac{\frac{2\cdot\lambda_{\max}}{\lambda_{\max}-\lambda_{\min}}}{1 + \frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}} - \cos\left(\pi \cdot \frac{2i+1}{2j+2}\right)} \\
&= \frac{1}{\lambda_{\max}} \cdot \frac{2 + \frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}}{\frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}} + 1 - \cos\left(\pi \cdot \frac{2i+1}{2j+2}\right)} \\
&= \frac{1}{\lambda_{\max}} \cdot \frac{2 + \frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}}{\frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}} + 2\cos^2\left(\pi \cdot \frac{2i+1}{j+1}\right)} \,.
\end{aligned} \tag{3.30}$$

With $j = n-1$, $\lambda_{\max} = \frac{4}{h^2}$ and the damping factor $\nu = \frac{2\cdot\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}$, we reobtain Eq. (2.90) that describes the time step sizes of the damped explicit scheme derived from the maximum variance filter V^h_{2n+1}. The case $\lambda_{\min} = 0$ means $\nu = 0$ and yields the time step sizes of the undamped method given by Eq. (2.86).

Very detailed experimental evaluations for the Super Time Stepping (STS) scheme have been given e.g. by Alexiades et al. [3, 4], and it has also been used to solve practical problems, for example, in [42, 92]. To improve the stability for non-symmetric parabolic problems, Gurski and O'Sullivan have proposed a modified STS scheme [65, 66, 67].

The application of Richardson's method to arbitrary linear systems, e.g. for indefinite, non-symmetric or complex matrices whose eigenvalues can be negative or range in the complex domain, has been discussed for instance in [22, 48, 102].

3.2 From Box Filtering to Fast Explicit Diffusion

This section deals with explicit linear diffusion schemes using the varying time step sizes that were motivated by the 1-D box filter kernel. We call one cycle with n varying time steps of this scheme a *Fast Explicit Diffusion (FED)* cycle. It has been proposed by Grewenig et al. [164, 165]. Because of its equivalence to 1-D box filtering, FED represents a stable explicit linear diffusion scheme.

Assume we have given a finite signal $\boldsymbol{f}^0 := (f_0^0, \dots, f_{N+1}^0)$ with dummy variables $f_0^0 = f_1^0$ and $f_{N+1}^0 = f_N$ (cf. Sec. 2.6). An explicit linear diffusion step with the time step size τ reads

$$f_k^1 \;=\; f_k^0 \,+\, \tau \cdot \frac{f_{k+1}^0 \,-\, 2\,f_k^0 \,+\, f_{k-1}^0}{h^2} \qquad (k = 1, \dots, N)\,. \tag{3.31}$$

We define the vectors

$$\tilde{\boldsymbol{f}}^i \;=\; \begin{pmatrix} f_1^i \\ \vdots \\ f_N^i \end{pmatrix} \;\in\; \mathbb{R}^N\,, \tag{3.32}$$

with $i \geq 0$ and the symmetric, tridiagonal matrix

$$\boldsymbol{A_h} \;=\; \frac{1}{h^2} \cdot \begin{pmatrix} -1 & 1 & & & & \boldsymbol{0} \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ \boldsymbol{0} & & & & 1 & -1 \end{pmatrix} \in \mathbb{R}^{N \times N}\,. \tag{3.33}$$

With the help of Gershgorin's theorem [58, 139], we can state that the N eigenvalues of the matrix $\boldsymbol{A_h}$ range in the real interval $\left[-\frac{4}{h^2}\,,\, 0\right]$. Here, the eigenvalue 0 exists in fact, and its multiplicity is one [128].

Using these definitions, we can consider Eq. (3.31) as a matrix-vector multiplication

$$\tilde{\boldsymbol{f}}^1 \;=\; (\boldsymbol{I} \,+\, \tau\,\boldsymbol{A_h})\,\tilde{\boldsymbol{f}}^0\,. \tag{3.34}$$

Thus, an FED cycle with n varying time step sizes τ_i is given by

$$\tilde{\boldsymbol{f}}^n \;=\; \left(\prod_{i=0}^{n-1} (\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{A_h})\right) \tilde{\boldsymbol{f}}^0\,. \tag{3.35}$$

However, it is important to understand that we do not explictly evaluate the product of n matrices. Instead, Eq. (3.35) means the scheme

$$\tilde{\boldsymbol{f}}^{i+1} \;=\; (\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{A_h})\,\tilde{\boldsymbol{f}}^i \qquad (i = 0, \dots, n-1)\,. \tag{3.36}$$

The Euclidean norm of the vector $\tilde{\boldsymbol{f}}^n$ satisfies

$$\begin{aligned}\left\|\tilde{\boldsymbol{f}}^n\right\|_2 &= \left\|\left(\prod_{i=0}^{n-1}(\boldsymbol{I} + \tau_i\,\boldsymbol{A_h})\right)\tilde{\boldsymbol{f}}^0\right\|_2 \\ &\le \left\|\left(\prod_{i=0}^{n-1}(\boldsymbol{I} + \tau_i\,\boldsymbol{A_h})\right)\right\|_2 \cdot \left\|\tilde{\boldsymbol{f}}^0\right\|_2 \\ &= \max_{z\in\left[0\,,\,\frac{4}{h^2}\right]}\left|\prod_{i=0}^{n-1}(1 - \tau_i\,z)\right| \cdot \left\|\tilde{\boldsymbol{f}}^0\right\|_2 \;\le\; \left\|\tilde{\boldsymbol{f}}^0\right\|_2\,, \end{aligned} \tag{3.37}$$

since we know that the symbol of a box filter B_{2n+1}^h is given by

$$p_B^{[n]}(z) \;=\; \prod_{i=0}^{n-1}(1 - \tau_i\,z) \;=\; \sum_{m=0}^{n}\frac{h^{2m}}{2m+1}\binom{n+m}{2m}(-z)^m\,, \tag{3.38}$$

and fulfils $|p_B^{[n]}(z)| \le 1$ for all $z \in \left[0\,,\,\frac{4}{h^2}\right]$. Moreover, we have used that the Euclidean norm of a symmetric matrix corresponds to the largest modulus of its eigenvalues. Thus, an FED cycle is in particular stable with respect to the Euclidean norm. The substitution $-z \to \boldsymbol{A_h}$ in Eq. (3.38) enables us to rewrite Eq. (3.35) as a matrix polynomial in $\boldsymbol{A_h}$:

$$\tilde{\boldsymbol{f}}^n \;=\; \left(\sum_{m=0}^{n}\frac{h^{2m}}{2m+1}\binom{n+m}{2m}\boldsymbol{A_h}^m\right)\tilde{\boldsymbol{f}}^0\,. \tag{3.39}$$

Such a series expansion can be very helpful for e.g. the analysis of the approximation order.

The main goal of this section is to show that FED can be applied to arbitrary multi-dimensional nonlinear diffusion processes, although it has been motivated in the 1-D setting with linear diffusion. As a first step, we analyse the distribution of the time step sizes with regard to their stability properties. Afterwards, we deal with some theoretical as well as numerical stability issues, and present common strategies to improve the robustness with respect to numerical rounding errors. Besides these stability issues, we also discuss the consistency and approximation quality of the proposed FED scheme.

3.2.1 Time Step Sizes

We have shown in the previous chapter that for a box kernel with length $(2n+1)h$ the time step sizes are given by

$$\tau_i \;=\; \frac{h^2}{2} \cdot \frac{1}{2\,\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \ldots, n-1). \tag{3.40}$$

Interestingly, the time step sizes τ_i partially violate the stability condition $\tau \leq \frac{h^2}{2}$. This is due to the acceleration factor $1/\left(2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)\right)$ that can be larger than 1. Table 3.1 shows both the smallest and largest three time step sizes for different n. Obviously, very large time steps are possible. For example with $n = 1000$, the largest inner time step size of an FED cycle is about 330000 times larger than the usual stability limit. Thus, it is not really surprising that the cycle time θ_n with n explicit time steps grows quadratically in n. As mentioned in the previous chapter, it is equal to $\frac{h^2}{6} \cdot (n^2 + n)$. An explicit scheme using only stable time steps can reach the maximum diffusion time $\frac{h^2}{2} \cdot n$. Hence, the speed-up factor is

$$\frac{\frac{h^2}{6}\,(n^2+n)}{\frac{h^2}{2} \cdot n} \;=\; \frac{1}{3}\,(n+1)\ , \tag{3.41}$$

and it is larger than 1 for $n \geq 3$.

Since we use a composition of stable and unstable time steps sizes within an FED cycle, it would be nice to know the ratio between them. In this context, the following proposition about the distribution of the time step sizes states an interesting rule: Use a stable time step and get an unstable step for free. This means that the proportion of the unstable steps is about 50%, and for even n equal to 50%. Note that $\lceil x \rceil$, where $x \in \mathbb{R}$, denotes the next largest integer $k \geq x$.

Proposition 3.1 (Distribution of the Time Step Sizes). *One FED cycle with n inner time steps consists of $\left\lceil \frac{n-1}{2} \right\rceil$ unstable steps.*

Proof. The unstable time step sizes within one FED cycle satisfy $\tau_i > \frac{h^2}{2}$, which is equivalent to

$$\cos\left(\pi \cdot \tfrac{2i+1}{4n+2}\right) \;<\; \sqrt{\frac{1}{2}}\ . \tag{3.42}$$

Because of $\pi \cdot \frac{2i+1}{4n+2} \in \left(0, \frac{\pi}{2}\right)$ for all $i \in \{0, ..., n-1\}$ and the monotonicity of the cosine function in $\left(0, \frac{\pi}{2}\right)$, the index i should meet the inequality

$$\pi \cdot \frac{2i+1}{4n+2} \;>\; \frac{\pi}{4}\ . \tag{3.43}$$

Table 3.1: First three and last three step sizes of FED (1-D) with $h = 1$.

$\boldsymbol{n}$	**50**	**100**	**250**	**500**	**1000**
$\boldsymbol{\tau_0}$	0.250060	0.250015	0.250002	0.250001	0.250000
$\boldsymbol{\tau_1}$	0.250545	0.250137	0.250022	0.250006	0.250001
$\boldsymbol{\tau_2}$	0.251518	0.250382	0.250061	0.250015	0.250004
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{\tau_{n-3}}$	28.79	113.79	706.52	2820.19	11269.25
$\boldsymbol{\tau_{n-2}}$	64.68	255.93	1589.57	6345.33	25355.72
$\boldsymbol{\tau_{n-1}}$	258.48	1023.45	6358.01	25381.06	101422.61
$\boldsymbol{\theta_n}$	425.00	1683.33	10458.33	41750.00	166833.33

This means

$$i > \frac{n}{2} - \frac{1}{4}\,, \tag{3.44}$$

i.e. the number of unstable step sizes is equal to $\frac{n}{2}$ for even n and for odd n we obtain $\frac{n-1}{2}$. The combination of both cases yields the above stated number $\lceil \frac{n-1}{2} \rceil$. □

We have already seen that the unstable time step sizes can be very large. Actually, the largest time step τ_{n-1} tends to infinity for $n \to \infty$:

$$\tau_{n-1}^{-1} = \frac{4}{h^2} \cdot \underbrace{\cos^2\left(\pi \cdot \tfrac{2n-1}{4n+2}\right)}_{\to \cos^2\left(\frac{\pi}{2}\right) = 0} \to 0\,. \tag{3.45}$$

To this end, it is necessary to clarify theoretical as well as numerical stability issues.

3.2.2 Theoretical Aspects of Stability

We only know that 1-D FED schemes reproduce 1-D box filtering and thus have to be stable after a whole cycle with n time steps. However, in Chapter 2, we have presented an exemplary evolution of a box filter with increasing number of explicit diffusion steps and we have seen that it is stable after all inner time steps. So the question is whether this example is just an exception or there is a general theoretical result that can be proven. We now analyse the stability regarding such theoretical issues.

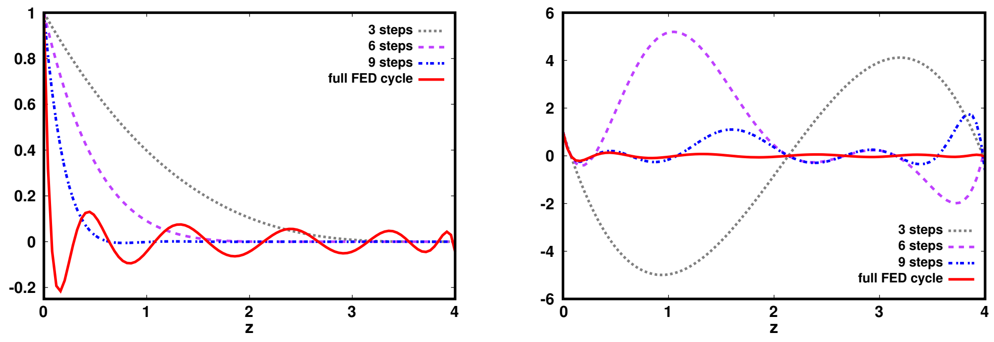


Figure 3.1: Inner symbols of FED with $n = 11$ and unit grid size $h = 1$. **(a) Left:** Original sequence. **(b) Right:** Rearranged sequence.

Inner Stability

At first, we want to prove a result concerning the inner stability and show that the inner symbols

$$p_B^{[k,n]}(z) \ := \ \prod_{i=0}^{k-1} (1 \ - \ \tau_i \, z) \qquad (k = 1, \ldots, n) \tag{3.46}$$

are bounded in absolute value by 1, provided that the time steps τ_i are used in their natural order. An example is illustrated in Fig. 3.1(a).

Proposition 3.2 (**Inner Stability of FED**). *Let $n \in \mathbb{N}$ be the length of an FED cycle with the time step sizes $\tau_0 < \tau_1 < \ldots < \tau_{n-1}$ given by Eq.* (3.40)*, and let $p_B^{[k,n]}(z)$ be the inner symbol of the k-th time step. Then these symbols satisfy for all $k \leq n$ and $z \in \left[0\,,\,\frac{4}{h^2}\right]$:*

$$|p_B^{[k,n]}(z)| \ \leq \ 1 \ . \tag{3.47}$$

Proof. We prove the proposition by contradiction: Assume we have found $n_0 \in \mathbb{N}$, $k_0 < n_0$ and $z_0 \in \left[0, \frac{4}{h^2}\right]$ with $|p_B^{[k_0,n_0]}(z_0)| > 1$. This can happen, if and only if at least one unstable time step has been used. Actually, this implies $n_0 > k_0 > \lceil \frac{n_0}{2} \rceil$ and thus $n_0 \geq 4$. Furthermore, we have

$$\begin{aligned} |p_B^{[k_0,n_0]}(z_0)| &= \prod_{i=0}^{k_0-1} |1 - \tau_i\, z_0| \\ &= \underbrace{\prod_{i=0}^{\lceil \frac{n_0}{2} \rceil - 1} |1 - \tau_i\, z_0|}_{\leq 1} \cdot \prod_{i=\lceil \frac{n_0}{2} \rceil}^{k_0-1} |1 - \tau_i\, z_0| \;>\; 1\,, \end{aligned} \tag{3.48}$$

and it follows that there exists at least one index $i_0 \geq \lceil \frac{n_0}{2} \rceil$ fulfilling the inequality $|1 - \tau_{i_0}\, z_0| > 1$. Since $i_0 \leq k_0 - 1$, we obtain

$$z_0 > \frac{2}{\tau_{i_0}} \geq \frac{2}{\tau_{k_0-1}}\,. \tag{3.49}$$

For the indices $i \geq k_0$ it holds that $\tau_i > \tau_{k_0-1}$, and therefore

$$|1 - \tau_i\, z_0| > \left|1 - \tau_i \frac{2}{\tau_{k_0-1}}\right| = \left|1 - 2\frac{\tau_i}{\tau_{k_0-1}}\right| > 1\,. \tag{3.50}$$

This yields for the symbol corresponding to the whole stable cycle:

$$\begin{aligned} |p_B^{[n_0]}(z_0)| &= |p_B^{[n_0,n_0]}(z_0)| \\ &= \underbrace{|p_B^{[k_0,n_0]}(z_0)|}_{>1} \cdot \underbrace{\prod_{i=k_0}^{n_0-1} |1 - \tau_i\, z_0|}_{>1} \;>\; 1\,, \end{aligned} \tag{3.51}$$

which is a contradiction, and thus the statement is proven. □

Actually, the sequence of the stable time steps does not matter for the proof. However, it is very important that the unstable steps come after the stable ones and are sorted by value. In the case of an arbitrary sequence, it might happen that the inner symbols $p_B^{[k,n]}$ take values with moduli much larger than 1. This is illustrated in Fig. 3.1(b). As a consequence, the corresponding inner filter kernels with length $(2k+1)h$ can have negative weights. On the other hand, the sorted sequence of unstable time steps only guarantees stability in the Euclidean norm. This can not exclude the appearance of negative filter weights. We know that such negative weights can only come from unstable steps and have to disappear at the end, because the box kernel consists of positive uniform weights. However, we conjecture that all inner filter kernels have positive weights as we have seen for example in Fig. 2.10. Now we give a formula for the weights of the inner filter kernels. It yields some interesting theoretical results.

Proposition 3.3 (Weights of the Inner Kernels). *Let $n \in \mathbb{N}$ be the length of an FED cycle, let*

$$w_k^{(0)} = \begin{cases} 1 & , k = 0 \\ 0 & , k > 0 \end{cases} , \tag{3.52}$$

and τ_m, $m \in \{0, \dots, n-1\}$, be the FED time step sizes. Moreover, we define the weights of the inner kernels by means of

$$w_k^{(m+1)} = \left(1 - 2\frac{\tau_m}{h^2}\right) \cdot w_k^{(m)} + \frac{\tau_m}{h^2} \cdot w_{k-1}^{(m)} + \frac{\tau_m}{h^2} \cdot w_{k+1}^{(m)} , \tag{3.53}$$

and let $w_{-1}^{(m)} := w_1^{(m)}$. Then $w_k^{(m)}$ fulfils

$$w_k^{(m)} = \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) . \tag{3.54}$$

Proof. First we show that Eq. (3.54) is valid for $m = 0$. In this case, the inner symbol is constant, i.e. $p_B^{[0,n]} = 1$. For $k = 0$ we have $T_0 = 1$ and thus

$$w_0^{(0)} = \frac{1}{2n+1} + \frac{2}{2n+1} \cdot n = 1 . \tag{3.55}$$

Let now $k > 0$. Since

$$1 - \frac{h^2}{2\tau_{n-s}} = \cos\left(\tfrac{2\pi \cdot s}{2n+1}\right) , \tag{3.56}$$

we obtain with $q := \left(\exp\left(i\,\frac{2\pi\cdot k}{2n+1}\right)\right) \in \mathbb{C}$:

$$
\begin{aligned}
&\sum_{s=1}^{n} T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \\
&= \sum_{s=1}^{n} \cos\left(\tfrac{2\pi\cdot k\cdot s}{2n+1}\right) \\
&= \frac{1}{2}\cdot\sum_{s=1}^{n} \left(\exp\left(i\,\tfrac{2\pi\cdot k}{2n+1}\right)\right)^s + \left(\exp\left(-i\,\tfrac{2\pi\cdot k}{2n+1}\right)\right)^s \\
&= \frac{1}{2}\cdot\left(\sum_{s=0}^{n} (q^s + \bar{q}^s) - 2\right) \\
&= -1 + \frac{1}{2}\cdot\left(\frac{1-q^{n+1}}{1-q} + \frac{1-\bar{q}^{n+1}}{1-\bar{q}}\right) \\
&= -1 + \frac{1 - \cos\left(\frac{2\pi\cdot k(n+1)}{2n+1}\right) - \cos\left(\frac{2\pi\cdot k}{2n+1}\right) + \cos\left(\frac{2\pi\cdot k\cdot n}{2n+1}\right)}{2 - 2\cos\left(\frac{2\pi\cdot k}{2n+1}\right)} \\
&= -\frac{1}{2} + \underbrace{\frac{\cos\left(\frac{2\pi\cdot k\cdot n}{2n+1}\right) - \cos\left(\frac{2\pi\cdot k\cdot(n+1)}{2n+1}\right)}{2 - 2\cos\left(\frac{2\pi\cdot k}{2n+1}\right)}}_{=0}, \qquad (3.57)
\end{aligned}
$$

where we have used

$$
\cos\left(\tfrac{2\pi\cdot k\cdot n}{2n+1}\right) = \cos\left(\tfrac{2\pi\cdot k\cdot(n+1)}{2n+1}\right) . \qquad (3.58)
$$

This verifies $w_k^{(0)} = 0$ for $k > 0$. Hence, Eq. (3.54) is valid for the level $m = 0$. Moreover, we have for $w_k^{(m+1)}$:

$$
\begin{aligned}
&\left(1 - 2\,\tfrac{\tau_m}{h^2}\right)\cdot w_k^{(m)} + \frac{\tau_m}{h^2}\cdot w_{k-1}^{(m)} + \frac{\tau_m}{h^2}\cdot w_{k+1}^{(m)} \\
&= \frac{1}{2n+1} + \frac{2}{2n+1}\cdot\sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right)\cdot\Bigg(\left(1 - 2\,\tfrac{\tau_m}{h^2}\right)\cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \\
&\quad + \frac{\tau_m}{h^2}\cdot T_{k-1}\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) + \frac{\tau_m}{h^2}\cdot T_{k+1}\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)\Bigg) \\
&= \frac{1}{2n+1} + \frac{2}{2n+1}\cdot\sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right)\cdot\Bigg(\left(1 - 2\,\tfrac{\tau_m}{h^2}\right)\cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)
\end{aligned}
$$

$$+ 2\,\frac{\tau_m}{h^2} \cdot T_1\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)\Big)$$

$$= \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot \Big(\left(1 - 2\,\tfrac{\tau_m}{h^2}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)$$

$$+ 2\,\frac{\tau_m}{h^2} \cdot \left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)\Big)$$

$$= \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} \underbrace{p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot \left(1 - \tfrac{\tau_m}{\tau_{n-s}}\right)}_{= p_B^{[m+1,n]}\left(\frac{1}{\tau_{n-s}}\right)} \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right)$$

$$= \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m-1} p_B^{[m+1,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) . \qquad (3.59)$$

This concludes the proof. □

As we have mentioned above, Eq. (3.54) helps us with the derivation of some theoretical facts that we are going to summarise in the following corollary.

Corollary 3.4. *Let $n \in \mathbb{N}$ be the length of an FED cycle and $w_k^{(m)}$ with $m \leq n$ the weights of the inner filter kernels. Then the following holds:*

(a) $w_0^{(m)} > \frac{1}{2n+1} > 0$ *and* $w_0^{(m+1)} < w_0^{(m)}$ *for* $m \in \{0, \dots, n-1\}$.

(b) $\max\limits_{0 \leq k \leq n} |w_k^{(m)}| = w_0^{(m)}$.

(c) $w_k^{(n-1)} > 0$ *and* $w_k^{(n-1)} > w_{k+1}^{(n-1)}$ *for* $k \in \{0, \dots, n-1\}$.

Proof. (a): According to Eq. (3.54) we have

$$w_0^{(m)} = \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) . \qquad (3.60)$$

Since the symbol satisfies

$$0 < p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) < 1 \qquad (3.61)$$

for $s \in \{1, \ldots, n-m\}$ with $m < n$, the sum is positive and therefore the centre weight satisfies $w_0^{(m)} > \frac{1}{2n+1}$. If $m = n$, one has the box filter weight $w_0^{(n)} = \frac{1}{2n+1}$. The monotonicity follows from the fact that

$$p_B^{[m+1,n]}\left(\tfrac{1}{\tau_{n-s}}\right) = \underbrace{\left(1 - \tfrac{\tau_m}{\tau_{n-s}}\right)}_{\in [0,1)} \cdot p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) < p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \tag{3.62}$$

for $s = 1, \ldots, n-m$ and therefore

$$\sum_{s=1}^{n-(m+1)} p_B^{[m+1,n]}\left(\tfrac{1}{\tau_{n-s}}\right) < \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) . \tag{3.63}$$

(b): Because of $|w_0^{(m)}| = w_0^{(m)}$, the maximum satisfies

$$\max_{0 \le k \le n} |w_k^{(m)}| \ \ge \ w_0^{(m)} . \tag{3.64}$$

For an arbitrary index $k \ge 0$, we use the triangle inequality:

$$\begin{aligned} |w_k^{(m)}| &= \left| \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \right| \\ &\le \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) \cdot \underbrace{\left| T_k\left(1 - \tfrac{h^2}{2\tau_{n-s}}\right) \right|}_{\le 1} \\ &\le \frac{1}{2n+1} + \frac{2}{2n+1} \cdot \sum_{s=1}^{n-m} p_B^{[m,n]}\left(\tfrac{1}{\tau_{n-s}}\right) = w_0^{(m)} . \end{aligned} \tag{3.65}$$

Thus,

$$\max_{0 \le k \le n} |w_k^{(m)}| \ \le \ w_0^{(m)} , \tag{3.66}$$

and the statement (b) is true.

(c): Proposition 3.3 yields

$$\begin{aligned} w_k^{(n-1)} &= \frac{1}{2n+1} + \frac{2}{2n+1} \cdot p_B^{[n-1,n]}\left(\tfrac{1}{\tau_{n-1}}\right) \cdot T_k\left(1 - \tfrac{h^2}{2\tau_{n-1}}\right) \\ &= \frac{1}{2n+1} + \tfrac{2}{2n+1} \cdot p_B^{[n-1,n]}\left(\tfrac{1}{\tau_{n-1}}\right) \cdot \cos\left(\tfrac{2\pi \cdot k}{2n+1}\right) , \end{aligned} \tag{3.67}$$

and because of

$$\cos\left(\tfrac{2\pi\cdot k}{2n+1}\right) \;>\; \cos\left(\tfrac{2\pi\cdot(k+1)}{2n+1}\right) \tag{3.68}$$

for $k \in \{0,\dots,n-1\}$, we get

$$w_k^{(n-1)} \;>\; w_{k+1}^{(n-1)} \;. \tag{3.69}$$

It is $w_n^{(n-1)} = 0$, and therefore (c) is proven. □

Corollary 3.4 states that the centre weights are positive for all time steps and decrease monotonically to the value $\frac{1}{2n+1}$. Moreover, the centre weight is the largest one for each step and if negative weights should appear, then their moduli are bounded by the corresponding centre weight. Another very interesting fact is the behaviour after the second last explicit step. The weights are positive and satisfy a monotonicity condition: The farther away from the centre, the smaller are the weights. However, the same condition is definitely fulfilled for all stable time steps. This gives rise to the conjecture that the condition is also satisfied for all unstable steps. Overall, this would mean that negative weights do not appear. Unfortunately, we have not found a proof yet.

At this point, we would like to mention another representation of the symbol that will be useful for further considerations. It involves the Chebyshev polynomials of the second kind:

Definition 3.5. *The Chebyshev polynomials of the second kind are defined by the recursion*

$$\begin{cases} U_0(x) &= 1\;, \\ U_1(x) &= 2x\;, \\ U_{n+1}(x) &= 2x\cdot U_n(x) \;-\; U_{n-1}(x)\;. \end{cases} \tag{3.70}$$

A closed-form representation for $n \geq 1$ is given by

$$U_n(x) \;=\; \sum_{m=0}^{\lfloor n/2\rfloor} (-1)^m \binom{n-m}{m} (2x)^{n-2m}\;. \tag{3.71}$$

The Chebyshev polynomials U_n have the nice property that they can be written as a sum of Chebyshev polynomials of the first kind, i.e.

$$U_{2k+1}(x) \;=\; 2\cdot\sum_{m=0}^{k} T_{2m+1}(x)\;, \tag{3.72}$$

and

$$U_{2k}(x) \;=\; 2 \cdot \sum_{m=0}^{k} T_{2m}(x) \;-\; 1 \,. \tag{3.73}$$

Thus, we obtain

$$\begin{aligned}
p_B^{[n]}(z) \;&=\; \frac{1}{2n+1} \cdot \left[1 \,+\, 2 \cdot \sum_{m=1}^{n} T_m \left(1 - \tfrac{h^2}{2}\, z \right) \right] \\
&=\; \frac{1}{2n+1} \cdot \left[1 \,+\, 2 \cdot \sum_{m=1}^{n} T_m \left(T_2 \left(\sqrt{1 - \tfrac{h^2}{4}\, z} \right) \right) \right] \\
&=\; \frac{1}{2n+1} \cdot \left[1 \,+\, 2 \cdot \sum_{m=1}^{n} T_{2m} \left(\sqrt{1 - \tfrac{h^2}{4}\, z} \right) \right] \\
&=\; \frac{1}{2n+1} \cdot \left[2 \cdot \sum_{m=0}^{n} T_{2m} \left(\sqrt{1 - \tfrac{h^2}{4}\, z} \right) \;-\; 1 \right] \\
&=\; \frac{1}{2n+1} \cdot U_{2n} \left(\sqrt{1 - \tfrac{h^2}{4}\, z} \right) \;,
\end{aligned} \tag{3.74}$$

where $z \in \left[0 \,,\, \frac{4}{h^2} \right]$. The advantage of this representation is the vanishing of a special treatment for the case $z = 0$. It also simplifies the description of the stability domain.

Stability Domain

The stability domain of an FED scheme with n explicit diffusion steps is given by the set [71]

$$\mathcal{S}^{(n)} \;:=\; \left\{ z \in \mathbb{C} \;:\; \left| p_B^{[n]}(-z) \right| \;\leq\; 1 \right\} \,. \tag{3.75}$$

Since the values $-z$ represent eigenvalues, the set $\mathcal{S}^{(n)}$ specifies the spectra of matrices $\boldsymbol{P}$ for which the matrix product $\prod_i (\boldsymbol{I} \,+\, \tau_i \, \boldsymbol{P})$ of the FED scheme has eigenvalues in the complex unit circle and is therefore stable. With the help of the above representation, we can state

$$\mathcal{S}^{(n)} \;=\; \left\{ z \in \mathbb{C} \;:\; \left| U_{2n} \left(\sqrt{1 + \tfrac{h^2}{4}\, z} \right) \right| \;\leq\; 2n+1 \right\} \,. \tag{3.76}$$

Figure 3.2 illustrates four different stability domains. Besides the above mentioned real interval, they also cover complex-valued numbers with an

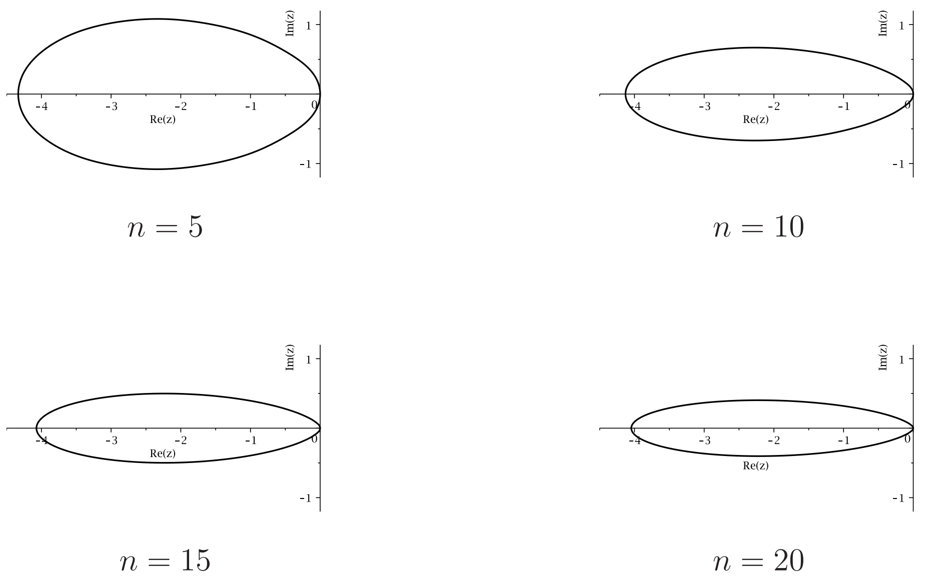


Figure 3.2: Stability domains $\mathcal{S}^{(n)}$ for different n $(h = 1)$.

imaginary part up to about ± 1. This means that the matrix $\boldsymbol{P}$ may have complex-valued eigenvalues. However, if n increases, the corresponding imaginary part decreases.

In the literature [71], it is also common to define the stability domain with the help of a modified amplification factor $\tilde{p}^{[n]}(z)$ that is given by

$$\tilde{p}^{[n]}(z) := 1 \; - \; z \; + \; \sum_{m=2}^{n} c_m \cdot (-z)^m \; . \tag{3.77}$$

Note that the coefficient of $-z$ is normalised to 1. In our case, the normalisation requires the division by the cycle time $\theta_n \; = \; \frac{h^2}{6} \cdot (n^2{+}n)$:

$$\tilde{p}_B^{[n]}(z) \;=\; p_B^{[n]}\left(\tfrac{z}{\theta_n}\right) \; . \tag{3.78}$$

Its corresponding stability domain reads

$$\tilde{\mathcal{S}}^{(n)} \;=\; \left\{ z \in \mathbb{C} \; : \; \left| U_{2n}\left(\sqrt{1 + \tfrac{3}{2\,(n^2+n)}\, z} \right) \right| \;\le\; 2n+1 \right\} \; . \tag{3.79}$$

The real part of $\tilde{\mathcal{S}}^{(n)}$ grows quadratically in n, whereas the size of the real domain for a usual explicit scheme is proportional to n. Figure 3.3 depicts some examples and illustrates the quadratic and linear dependency,

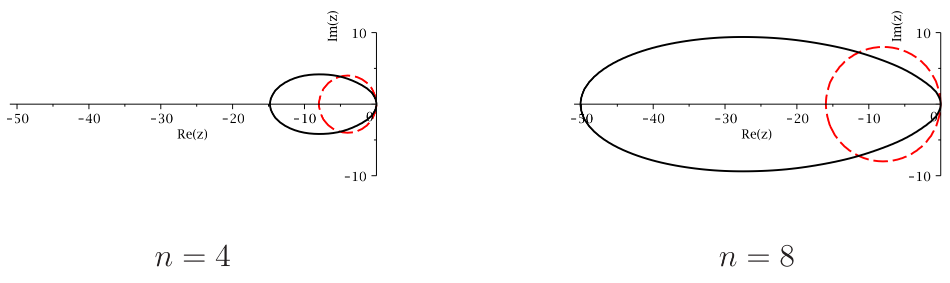


Figure 3.3: Stability domains $\tilde{\mathcal{S}}^{(4)}$ and $\tilde{\mathcal{S}}^{(8)}$ $(h = 1)$ for FED (solid line, black) and the usual explicit scheme (dashed, red).

respectively. We observe that the imaginary part seems to increase only proportionally to n in both cases, although the variable z is divided by n^2+n in the case of FED. However, this explains the decay with $\mathcal{O}(\frac{1}{n})$ of the imaginary part within $\mathcal{S}^{(n)}$ in Fig. 3.2.

3.2.3 Numerical Stability

Although the amplification factor of FED is bounded in theory, a rearrangement of the natural sequence of the time step sizes is necessary in practice due to numerical rounding errors. This can be seen as follows: Let λ_i, $i = 1, \dots, N$, be the eigenvalues of $\boldsymbol{A_h} \in \mathbb{R}^{N\times N}$ from Eq. (3.33) with their corresponding normalised eigenvectors $\boldsymbol{v}_i$. Due to the symmetry of $\boldsymbol{A_h}$, they form an orthonormal basis of $\mathbb{R}^N$. Moreover, they can be seen as frequency components, where smaller frequencies correspond to eigenvalues with small moduli and higher frequencies to larger absolute values. Referring to the FED scheme in Eq. (3.35), the initial vector $\tilde{\boldsymbol{f}}^0$ can be written in terms of this basis, which means

$$\tilde{\boldsymbol{f}}^0 \;=\; \sum_{i=1}^{N} C_i \cdot \boldsymbol{v}_i \;, \tag{3.80}$$

with the coefficients $C_i = \boldsymbol{v}_i^T \tilde{\boldsymbol{f}}^0 \in \mathbb{R}$. If we apply the FED scheme and use only the $\lfloor \frac{n+1}{2} \rfloor$ stable time steps, the corresponding solution satisfies

$$\tilde{\boldsymbol{f}}^{\lfloor \frac{n+1}{2} \rfloor} \;=\; \Big(\prod_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} (\boldsymbol{I} + \tau_i\, \boldsymbol{A_h}) \Big) \tilde{\boldsymbol{f}}^0 \;=\; \sum_{k=1}^{N} C_k \cdot \Big(\prod_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} (\boldsymbol{I} + \tau_i\, \boldsymbol{A_h}) \Big) \boldsymbol{v}_k$$

$$= \sum_{k=1}^{N} C_k \cdot \Big(\prod_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} (1 + \tau_i \, \lambda_k) \Big) \boldsymbol{v}_k \; = \; \sum_{k=1}^{N} \tilde{C}_k \cdot \boldsymbol{v}_k \; , \tag{3.81}$$

where

$$\tilde{C}_k \; := \; C_k \cdot \prod_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} (1 + \tau_i \, \lambda_k) \; . \tag{3.82}$$

For stable time step sizes τ_i and $\lambda_k < 0$ we have $q_{i,k} := |1 + \tau_i \, \lambda_k| < 1$, i.e. $|\tilde{C}_k| < |C_k|$. We assume w.l.o.g. that $\lambda_1 = 0$ and the eigenvalues are arranged such that $\lambda_N < \lambda_{N-1} < \cdots < \lambda_2 < \lambda_1$. Thus, $\tilde{C}_1 = C_1$ and

$$q_{i,k} \; \leq \; \max \Big\{ \Big| 1 + \tfrac{h^2}{4} \lambda_2 \Big| \; , \; \Big| 1 + \tfrac{h^2}{2} \lambda_N \Big| \Big\} \; =: \; q_N \; < \; 1 \; , \tag{3.83}$$

since for $i = 0, \ldots, \lfloor \frac{n-1}{2} \rfloor$ and $k = 2, \ldots, N$ it holds that

$$1 \; > \; 1 + \frac{h^2}{4} \lambda_2 \; \geq \; 1 + \tau_i \, \lambda_k \tag{3.84}$$

as well as

$$-1 \; < \; 1 + \frac{h^2}{2} \lambda_N \; \leq \; 1 + \tau_i \, \lambda_k \; . \tag{3.85}$$

Note that the maximum q_N does not depend on the time step sizes and the cycle length n. According to [128], the eigenvalues of $\boldsymbol{A_h}$ are given by

$$\lambda_k \; = \; \frac{1}{h^2} \cdot \big(2 \cos \big(\pi \cdot \tfrac{k-1}{N} \big) \; - \; 2 \big) \qquad (k = 1, \ldots, N) \, . \tag{3.86}$$

For $N \geq 3$ we obtain

$$1 + \frac{h^2}{4} \lambda_2 \; = \; \frac{1}{2} \cdot \big(1 + \cos \big(\tfrac{\pi}{N} \big) \big) \; > \; \cos \big(\tfrac{\pi}{N} \big) \; > \; 0 \; , \tag{3.87}$$

and

$$1 + \frac{h^2}{2} \lambda_N \; = \; - \cos \big(\tfrac{\pi}{N} \big) \; . \tag{3.88}$$

Hence, we can simplify the maximum by

$$q_N \; = \; \frac{1}{2} \cdot \big(1 + \cos \big(\tfrac{\pi}{N} \big) \big) \; < \; 1 \; . \tag{3.89}$$

Overall, the coefficients $\tilde{C}_k$ with $k \geq 2$ fulfil

$$\Big| \tilde{C}_k \Big| \; = \; |C_k| \cdot \prod_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} |1 + \tau_i \, \lambda_k| \; \leq \; |C_k| \cdot q_N^{\lfloor \frac{n+1}{2} \rfloor} \; , \tag{3.90}$$

and therefore $\tilde{C}_k \to 0$ for $n \to \infty$. This means an increasing cycle length n scales down the moduli of the coefficients $\tilde{C}_k$. However, regarding the numerical computations, finite precision arithmetic does not allow arbitrary small coefficients and produces numerical rounding errors. For sufficiently large n, this means that there exists an $\varepsilon > 0$ such that some numerical results $\tilde{C}_k^r$ belonging to the exact coefficients $\tilde{C}_k$ with $\tilde{C}_k \neq 0$ satisfy $|\tilde{C}_k^r| \geq \varepsilon$, although the exact coefficient is smaller: $|\tilde{C}_k| < \varepsilon$ with $\frac{|\tilde{C}_k^r|}{|\tilde{C}_k|} \gg 1$.

If we use the unstable time steps and assume again a sufficiently large n, then there exists for each eigenvalue λ_k with $k \geq 2$ an index $i_0(k)$ with $q_{i,k} > 1$ for $i \geq i_0(k)$. In this case, the absolute values of the k-th coefficient will increase from step to step and may become as large as

$$\varepsilon \cdot \prod_{i=i_0(k)}^{n-1} |1 + \tau_i \lambda_k| \ . \tag{3.91}$$

This can lead to severe inaccuracies right up to exceedance of the machine precision, because we work with huge time step sizes, i.e. $q_{i,k} \gg 1$. To illustrate these thoughts, we now present an example.

Example

We consider an oscillatory 1-D signal (vector) with length $N = 7$,

$$\boldsymbol{g}^0 \ = \ (1, -1, 1, -1, 1, -1, 1)^T \ \in \ \mathbb{R}^7 \ , \tag{3.92}$$

and apply the FED scheme (3.35) using the matrix $\boldsymbol{A_h} \in \mathbb{R}^{7\times 7}$ with unit grid size $h = 1$. The length of the FED cycle is defined by $n = 32$ and therefore corresponds to the box filter B_{65}^1. Due to the reflecting boundary condition, this yields the result

$$\boldsymbol{g}^{32} \ = \ \frac{1}{65} \cdot (9, 9, 9, 11, 9, 9, 9)^T \ . \tag{3.93}$$

Using the seven orthonormal eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_7$ of $\boldsymbol{A_h}$, we can express each intermediate result $\boldsymbol{g}^i$ in terms of these basis vectors, i.e.

$$\boldsymbol{g}^i \ := \ \left(\prod_{\ell=0}^{i-1} (\boldsymbol{I} + \tau_\ell \boldsymbol{A_h}) \right) \boldsymbol{g}^0 \ = \ \sum_{k=1}^{7} C_k^{(i)} \cdot \boldsymbol{v}_k \qquad (i = 0, \ldots, 32) \ . \tag{3.94}$$

In this numerical experiment, we focus on the coefficients $C_7^{(i)}$ of the highest frequency component $\boldsymbol{v}_7$ and observe their absolute values for different i

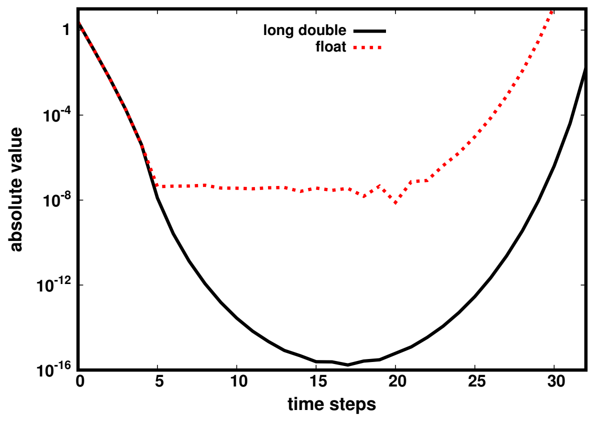


Figure 3.4: Evolution of $|C_7^{(\cdot)}|$ with respect to the number of time steps for float and long double precision.

with respect to both *float* (4 bytes) and *long double* (12 bytes) precision. The final results for float (fl) and long double (ld) precision are given by

$$\boldsymbol{g}_{fl}^{32} = 10^5 \cdot (0.654, -1.833, 2.649, -2.940, 2.649, -1.833, 0.654)^T , \quad (3.95)$$

and

$$\boldsymbol{g}_{ld}^{32} = (0.1385, 0.1385, 0.1385, 0.1692, 0.1385, 0.1385, 0.1385)^T , \quad (3.96)$$

respectively. We see that $\boldsymbol{g}_{ld}^{32}$ is a very good approximation for the exact solution $\boldsymbol{g}^{32}$, whereas $\boldsymbol{g}_{fl}^{32}$ yields a huge numerical error. This error occurs mainly due to the behaviour of the coefficients $C_7^{(i)}$ in float precision, which is illustrated in Fig. 3.4. During the first five time steps, there is virtually no difference between float and long double precision. However, the next time steps make $|C_7^{(i)}|$ too small for float precision and numerical rounding errors appear, whereas the curve of the long double precision is as we would expect it for the exact coefficients: The absolute values decrease for all stable steps, and then increase again when unstable steps come into play. In absolute terms, the numerical rounding errors are very small, but the large unstable time steps amplify them up to multiple orders of magnitudes. This induces the highly inaccurate result.

The example indicates that Prop. 3.2 is not sufficient for stability in practical applications. To make FED more robust with respect to numerical rounding errors, we have to take care of the sequence of the time step sizes.

However, if we use a rearranged sequence, it might happen that some intermediate results are unstable, which has been shown in Fig. 3.1(b). This is very important with regard to nonlinear problems, where the correct numerical solution requires stable intermediate results. Hence, there is a trade-off between theoretical and numerical stability issues.

In the context of Super Time Stepping (STS), the problem with numerical rounding errors has been addressed e.g. by Gentzsch and Schlüter [57]. However, in the case of Richardson's method, Young already mentioned this problem in 1954 and proposed an intuitive ordering that makes the method more robust with respect to rounding errors [160]. Later, Lebedev and Finogenov have presented a detailed rounding error analysis and have constructed an ordering that guarantees a computationally stable form of Richardson's method [89]. Unfortunately, this rearrangement only works for cycle lengths $n \in \{2^p \mid p \in \mathbb{N}_0\}$. Anderssen and Golub [5] have performed some numerical experiments and compared the strategies of Young as well as Lebedev and Finogenov. They state that Young's rearrangement of the time step sequence has problems with large n (in their experiment: $n = 128$), whereas the method of Lebedev and Finogenov performs well. Another strategy has been proposed by Reichel [110]. He considered the so-called Leja points [43, 90]. The main advantage of this algorithm is that it can be applied to arbitrary sets of varying parameters, i.e. $n \in \mathbb{N}$ and the parameters can also be complex-valued. Given a set, the original sequence of the elements is rearranged such that they fulfil certain constraints. This Leja ordering has been successfully applied not only to Richardson's method [21, 22, 110], but also to other problems like Newton interpolation or evaluation of polynomial coefficients [23, 109].

For our FED method, we want to use three different strategies that we have mentioned above: The first is similar to the ideas of Gentzsch and Schlüter, which means that we use so-called κ-cycles. Our second option is the Leja ordering. We have chosen these methods, since they allow arbitrary cycle lengths n and have already been successfully applied to improve the robustness with respect to numerical rounding errors. The third strategy is the Lebedev-Finogenov ordering. As mentioned above, it allows only specific cycle lengths, but it has a nice theoretical foundation and an elegant derivation. Since Lebedev and Finogenov have proven that their approach works well with Super Time Stepping (even without any damping), we conjecture that there might also exist a proof for FED. This is due to the fact that the FED time step sizes are smaller, and therefore less critical. Unfortunately, we have not found a proof yet.

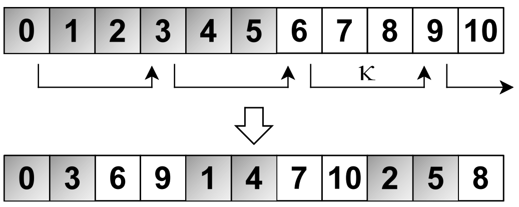


Figure 3.5: Example for the rearrangement with κ-cycles for $n = 11$ and $\kappa = 3$. The unstable time steps are white.

κ-Cycles

The rearrangement of the time step sequence with κ-cycles has been used by Gentzsch and Schlüter to improve the robustness of STS [57]. Since FED is a similar scheme, it is natural to use these κ-cycles also in our case. An example with $n = 11$ and $\kappa = 3$ is illustrated in Fig. 3.5. The first six time steps are stable, whereas steps with indices larger than five are beyond the stability restriction. As one can see, the sequence is in this case subdivided into four smaller cycles, where all cycles contain at least two stable time steps. In general, this idea can be formulated as follows: Let p be the smallest prime number with $p \geq n$ and τ_i , $i \in \{0, \ldots, n-1\}$, the time step sizes ordered in their natural sequence. We decompose the set of all τ_i into κ-cycles with $1 < \kappa < n$, and introduce a new sequence $\tilde{\tau}_0, \ldots, \tilde{\tau}_{n-1}$. Therefore, we set a counter variable $\ell = 0$ and define the mapping

$$\Phi_\kappa(m) \; := \; \big(m \cdot \kappa\big) \mod p \; . \tag{3.97}$$

For all $m = 0, \ldots, p-1$, we compute the values $\Phi_\kappa(m)$. If $\Phi_\kappa(m) \leq n-1$, then $\tilde{\tau}_\ell := \tau_{\Phi_\kappa(m)}$ and $\ell := \ell+1$. In the case of $\Phi_\kappa(m) \geq n$, we just drop this value. Then we get a feasible rearrangement of the original sequence. The algorithm is summarised in Fig. 3.6.

Unfortunately, there is no panacea for the choice of κ. However, it is possible to create a look-up table with suitable values $\kappa = \kappa(n)$ that ensure robustness regarding numerical rounding errors. To this end, we consider a 1-D test problem with an oscillatory signal $\boldsymbol{s}$ of a certain length $m \cdot h$ that is defined by $s_i := (-1)^{i-1}$ for $i \in \{1, \ldots, m\}$. If $i < 1$ or $i > m$ we assume reflecting boundary conditions. Note that it makes sense to consider high frequency signals, since they provide the biggest challenge with respect to numerical rounding errors, as shown in our above example. To find suitable

κ-values, we first compute the box filtered signal

$$\left(B^h_{2n+1}(\boldsymbol{s})\right)_i \;=\; \frac{1}{2n+1} \sum_{k=-n}^{n} s_{i+k} \;. \tag{3.98}$$

It represents the exact solution of an FED scheme with n time steps. Hence, finding a good value for $\kappa < n$ is equivalent to the minimisation of the error between the numerical result $\tilde{\boldsymbol{s}}^n_\kappa \in \mathbb{R}^m$ of the FED scheme using the corresponding sequence $\tilde{\tau}_0, \ldots, \tilde{\tau}_{n-1}$ and the box filtered signal $B^h_{2n+1}(\boldsymbol{s})$. As a measure, we use the absolute error

$$\sum_{i=1}^{m} \left| (\tilde{\boldsymbol{s}}^n_\kappa)_i \;-\; \left(B^h_{2n+1}(\boldsymbol{s})\right)_i \right| \;. \tag{3.99}$$

To create a look-up table for the numerical experiments, we have used a signal length with $m = 50$ and performed all computations in float precision. In the experimental section, we are going to see how this specific look-up table works in practice.

If we reconsider the above presented example, where we have analysed the behaviour of the result with respect to different numerical precisions, we can also examine the evolution of the coefficient for a rearranged sequence of time steps. This is illustrated in Fig. 3.7. As we can see, the numerical precision does not matter anymore, because there is no visible difference between the two curves, and the numerical results are both very accurate. The mix of stable and unstable steps prevents that the coefficient becomes too small or too large for computations with float precision. In the case of the original sequence, the absolute values range from $1.7 \cdot 10^{-16}$ to 2.4. With the rearranged sequence, they lie in $[8 \cdot 10^{-4}\,,\, 10]$, and are suited for float precision.

Leja Ordering

Since suitable κ-cycles can only be found experimentally and therefore depend on the setting of the test problems, we want to discuss another option, namely the Leja points [43, 90]. They have already been used to make Richardson's cyclic method more robust with respect to numerical rounding errors [21, 22, 110]. The corresponding algorithm works as follows: Assume that we have given an arbitrary finite set of $m{+}1$ real-valued (or complex-valued) numbers. In the case of Richardson's method, Reichel used the reciprocals of the relaxation parameters to construct this initial set [110].

```
1. Input:
   natural sequence of the time step sizes $\tau_0, \dots, \tau_{n-1}$ and $2 \le \kappa < n$
2. Initialisation:
   Compute the smallest prime number $p$ with $p \ge n$ and set $\ell := 0$.
3. Reordering: for $m = 0, 1, \dots, p-1$
   (a) Compute $\Phi_\kappa(m) = \big(m \cdot \kappa\big) \bmod p$.
   (b) If $\Phi_\kappa(m) \in \{0, \dots, n-1\}$:
         * $\tilde{\tau}_\ell := \tau_{\Phi_\kappa(m)}$
         * $\ell := \ell+1$
         * If $\ell < n$, increase $m$ by 1 and go back to (a).
   (c) If $\Phi_\kappa(m) \notin \{0, \dots, n-1\}$, then increase $m$ and go to (a).
4. Output:
   rearranged sequence $\tilde{\tau}_0, \dots, \tilde{\tau}_{n-1}$
```

Figure 3.6: Algorithm for the rearrangement with κ-cycles.

The application of the Leja algorithm yields a set $S := \{x_0, \dots, x_m\}$ which is arranged such that

$$\prod_{k=0}^{j} |x_{j+1} - x_k| = \max_{x \in S} \prod_{k=0}^{j} |x - x_k| \qquad (j = 0, ..., m-1), \tag{3.100}$$

and $|x_0| = \max\limits_{x \in S} |x|$. It might happen that two or more numbers satisfy the maximum condition. In such a case, we just take the number with the smallest modulus to get a unique result. To rearrange the sequence of n FED time step sizes, we consider the set $K_n := \{z_0, \dots, z_{n-1}\}$ with the scaled reciprocals of the time step sizes

$$z_i := \frac{h^2}{4} \cdot \tau_i^{-1} = \cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right) \in (0,1) \quad (i = 0, \dots, n-1) \tag{3.101}$$

and apply the Leja algorithm. This yields the rearranged set $\tilde{K}_n := \{\tilde{z}_0, \dots, \tilde{z}_{n-1}\}$. We know that $\tilde{z}_0 = z_0$, i.e. $\tilde{\tau}_0 = \tau_0$. The next numbers

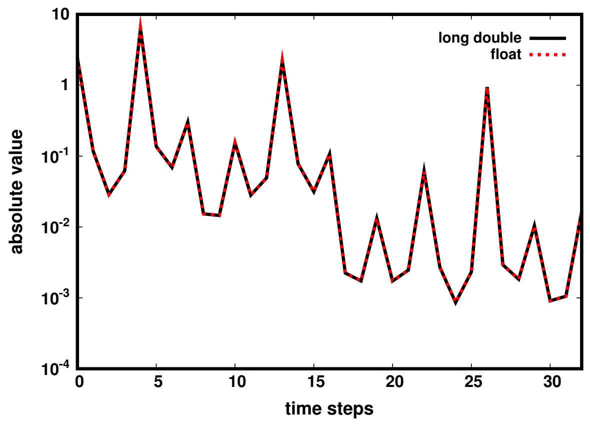


Figure 3.7: Evolution of $|C_7^{(\cdot)}|$ with respect to the number of time steps for float and long double precision. The sequence of the time step sizes $(n = 32)$ has been rearranged using a κ-cycle with $\kappa = 10$.

with $j > 0$ have to satisfy

$$\begin{aligned}
\prod_{k=0}^{j} |\tilde{z}_{j+1} - \tilde{z}_k| &= \max_{z \in K_n} \prod_{k=0}^{j} |z - \tilde{z}_k| \\
&= \prod_{\ell=0}^{j} \tilde{z}_\ell \cdot \max_{z \in K_n} \left| \prod_{k=0}^{j} \left(1 - \frac{z}{\tilde{z}_k}\right) \right| \\
&= \prod_{\ell=0}^{j} \tilde{z}_\ell \cdot \max_{z \in K_n} \left| \prod_{k=0}^{j} (1 - \tilde{\tau}_k \, z) \right| . \qquad (3.102)
\end{aligned}$$

Actually, the second product represents the inner symbol after $j+1$ time steps with respect to the reordered sequence. This means the next point $\tilde{z}_{j+1} > 0$ is an extremum point of the inner symbol in K_n. By choosing $\tilde{z}_{j+1}$ in the next step, the corresponding extremum vanishes, since $\tilde{z}_{j+1}$ is a root of the inner symbol after $j+2$ steps. A summary of the whole algorithm is given in Fig. 3.10.

The main advantage compared to κ-cycles is that the rearrangement given by the Leja ordering only depends on the input set K_n. We do not need any test problem, where we have to adjust some parameters. Concerning our example, the evolution of the coefficient with Leja ordering is shown in Fig. 3.8. Here, the absolute values range from about 10^{-2} to 48, which is

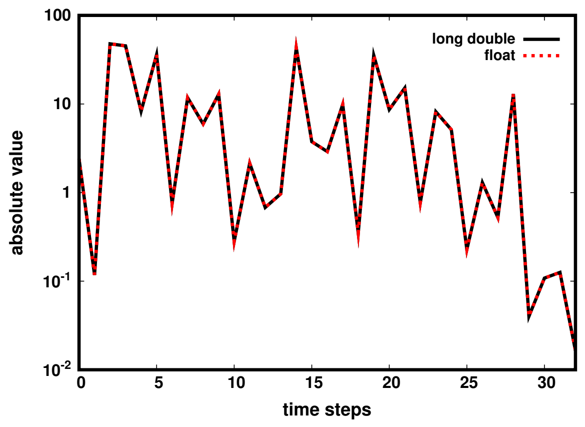


Figure 3.8: Evolution of $|C_7^{(\cdot)}|$ with respect to the number of time steps for float and long double precision. The sequence of the time step sizes $(n = 32)$ has been rearranged by the Leja algorithm.

up to two orders of magnitude larger than with κ-cycles. However, both curves have a zigzag-like course that avoids, as mentioned above, too small or too large values exceeding the numerical precision.

A disadvantage of the Leja ordering is the evaluation of the products. If some factors $|x - x_k|$ are almost equal to zero, it might happen that the whole product is wrongly evaluated due to rounding errors. The larger j is, the smaller are the single factors $|x - x_k|$. Moreover, the number of these factors increases at the same time, which means that the product is extremely small. Such problems appear in particular for large n and j. The maximum of the products can not be correctly determined and hence one has to take care that the precision is sufficient to ensure a correct Leja ordering. In our experiments, differences between float and long double precision already appear for $n = 81$ and indices $j \geq 78$. The experimental section will show that the precision is very important in the case of Leja ordering.

Lebedev-Finogenov Ordering

The last rearrangement that we want to discuss has been introduced by Lebedev and Finogenov [89]. It has also been developed in the context of STS. This reordering procedure is based on a recursion relation and only works for cycle lengths that are powers of two. Let us assume that we have given the permutation with length 2^{p-1} $(p \geq 1)$, i.e. $(j_1, \ldots, j_{2^{p-1}})$,

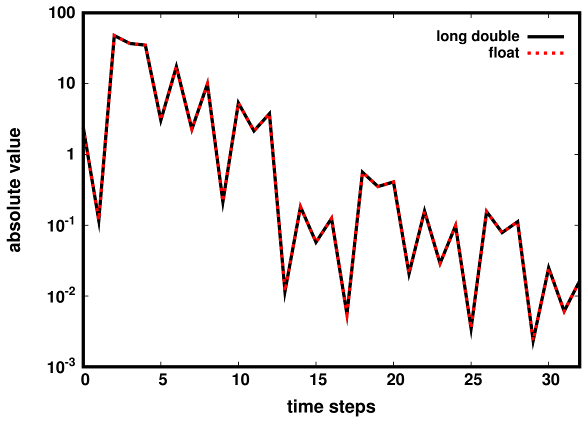


Figure 3.9: Evolution of $|C_7^{(\cdot)}|$ with respect to the number of time steps for float and long double precision. The sequence of the time step sizes $(n = 32)$ has been rearranged by the Lebedev-Finogenov ordering.

where $j_\ell \in \{0, \ldots, 2^{p-1}-1\}$ for $\ell = 1, \ldots, 2^{p-1}$. Then the permutation with length 2^p reads

$$(j_1\,,\,2^p-1-j_1\,,\,j_2\,,2^p-1-j_2\,,\ldots,\,j_{2^{p-1}}\,,2^p-1-j_{2^{p-1}})\ . \tag{3.103}$$

As one can see, the algorithm for creating such a permutation is very easy. However, the computational effort for an FED cycle might be increased by a factor of about two. More precisely, instead of e.g. a cycle length $n = 65$ we would have to use a cycle with 128 explicit steps for the Lebedev-Finogenov ordering.

Fortunately, our example indicates that this ordering works also with FED. This is shown in Fig. 3.9. Thus, the example supports the above mentioned conjecture that there might exist a theoretical result for FED. Similarly to the other rearrangements, the Lebedev-Finogenov ordering mixes stable and unstable steps such that the absolute values of the coefficient $C_7^{(\cdot)}$ range in the interval $[5 \cdot 10^{-3}\,,\,48]$, where float precision is sufficient.

Actually, all three strategies work well up to a certain cycle length, as we are going to see in the numerical experiments. At this point, we can hardly say that one of the algorithms is clearly better than the other one. Since Leja ordering is very expensive for problems with $n \gg 1000$, we recommend to use a look-up table. However, we need to store the whole optimal permutation, whereas in the case of κ-cycles, the permutation is defined only via the integer κ. On the other hand, both the Leja and

1. **Input**:
   natural sequence of the time step sizes $\tau_0, \dots, \tau_{n-1}$
2. **Initialisation**:
   Compute the scaled reciprocals $z_i := \frac{h^2}{4} \cdot \tau_i^{-1}$, and $\tilde{z}_0 := z_0$.
3. **Leja algorithm**: for $j = 1, 2, \dots, n-1$
   - Compute $\tilde{z}_j := \max\limits_{z} \prod\limits_{k=0}^{j-1} |z - \tilde{z}_k|$
     with $z \in \{z_0, \dots, z_{n-1}\} \setminus \{\tilde{z}_0, \dots, \tilde{z}_{j-1}\}$.
4. **Time step sizes**:
   Compute the new sequence of time step sizes via $\tilde{\tau}_i = \frac{h^2}{4} \cdot \tilde{z}_i^{-1}$.
5. **Output**:
   rearranged sequence $\tilde{\tau}_0, \dots, \tilde{\tau}_{n-1}$

Figure 3.10: Algorithm for the rearrangement with Leja points.

the Lebedev-Finogenov ordering have, in contrast to the κ-cycles, a better theoretical foundation. Moreover, the construction of a Lebedev-Finogenov sequence is much cheaper than the Leja ordering. A comparison of the three strategies can be found in the experimental section.

3.2.4 Consistency and Convergence

In this subsection, we want to analyse the consistency and the convergence properties of the linear FED scheme. To this end, we reconsider the linear 1-D diffusion equation

$$\partial_t u(x,t) \;=\; \partial_{xx} u(x,t) \tag{3.104}$$

with $(x,t) \in (a,b) \times (0,\infty)$, a given function $u(x,0) = u_0(x)$ for $x \in [a,b]$, and the unknown solution u. Note that we assume homogeneous Neumann boundary conditions

$$\partial_x u(a,t) \;=\; \partial_x u(b,t) \;=\; 0 \tag{3.105}$$

for $t \in (0,\infty)$. As we have mentioned in Chapter 1, this PDE can be formulated as an ODE system, if we discretise it with respect to the spatial domain, i.e. apply the method of lines [116, 120]. We use a spatial grid with N nodes, i.e.

$$\left\{x_i \;=\; a + \left(i - \tfrac{1}{2}\right) \cdot h \;\middle|\; 1 \le i \le N\right\} \;\subset\; [a,b]\;, \tag{3.106}$$

and the mesh size $h \;:=\; \frac{b-a}{N}$. The corresponding system of ODEs reads [147]

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{A_h u}\;, \tag{3.107}$$

where $\boldsymbol{A_h} \in \mathbb{R}^{N\times N}$ is defined in Eq. (3.33) and

$$\boldsymbol{u} \;=\; \boldsymbol{u}(t) \;=\; \big(u(x_1,t),\ldots,u(x_N,t)\big)^T \;\in\; \mathbb{R}^N\;. \tag{3.108}$$

With the initial value $\boldsymbol{u}^0 = \boldsymbol{u}(0)$ the exact solution of the ODE system is given by the matrix exponential

$$\begin{aligned}\boldsymbol{u}(t) \;&=\; \exp\left(t\cdot\boldsymbol{A_h}\right)\,\boldsymbol{u}^0\\ &=\; \left(\sum_{i=0}^{\infty}\frac{\left(t\cdot\boldsymbol{A_h}\right)^i}{i!}\right)\boldsymbol{u}^0\;.\end{aligned} \tag{3.109}$$

Instead of computing the matrix exponential (cf. [100]), we want to apply the FED scheme in order to approximate it. To this end, we introduce an equidistant time grid

$$\left\{t_k = \tfrac{k}{M}\cdot T \;\middle|\; 0 \le k \le M\right\} \;\subset\; [0,T]\;, \tag{3.110}$$

with $M \ge 1$ and a stopping time $T > 0$. Then we have for $k \ge 1$

$$\begin{aligned}\boldsymbol{u}(t_k) \;&=\; \exp\left(t_k\cdot\boldsymbol{A_h}\right)\,\boldsymbol{u}^0\\ &=\; \exp\left(t_{k-1}\cdot\boldsymbol{A_h}\right)\cdot\underbrace{\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right)\,\boldsymbol{u}^0}_{=\,\boldsymbol{u}(t_1)}\\ &\;\;\vdots\\ &=\; \exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right)\,\boldsymbol{u}(t_{k-1})\;.\end{aligned} \tag{3.111}$$

Thus, each cycle of an FED scheme with a total of M cycles should approximate the matrix exponential

$$\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right) \;=\; \sum_{i=0}^{\infty}\frac{T^i}{M^i\cdot i!}\cdot\boldsymbol{A}_{\boldsymbol{h}}^i\;, \tag{3.112}$$

to be consistent. Usual first order approximations are, for example,

$$\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right) \;\approx\; \boldsymbol{I} \;+\; \tfrac{T}{M}\cdot\boldsymbol{A_h}\;, \tag{3.113}$$

which corresponds to an explicit scheme, where the stability is ensured for $\frac{T}{M} \leq \frac{h^2}{2}$, and the implicit method

$$\exp\left(\tfrac{T}{M} \cdot \boldsymbol{A_h}\right) \;\approx\; \left(\boldsymbol{I} \;-\; \tfrac{T}{M} \cdot \boldsymbol{A_h}\right)^{-1} \;. \tag{3.114}$$

Concerning one FED cycle with n time steps, the comparison of the series expansion from Eq. (3.39), i.e.

$$\boldsymbol{I} \;+\; \frac{h^2}{3}\binom{n+1}{2}\boldsymbol{A_h} \;+\; \sum_{m=2}^{n} \frac{h^{2m}}{2m+1}\binom{n+m}{2m}\boldsymbol{A}_{\boldsymbol{h}}^{m} \;, \tag{3.115}$$

and the exponential series (3.112) yields

$$\frac{h^2}{3}\binom{n+1}{2} \;=\; \frac{T}{M} \;. \tag{3.116}$$

However, this condition can not be fulfilled for arbitrary $T > 0$, since the left hand side does not cover all positive real-valued numbers. Therefore we have to incorporate a real-valued adjustment factor $q \in (0,1]$ for the time step sizes of an FED cycle. More precisely, we determine the smallest integer n_0 such that the cycle time fulfils

$$\theta_{n_0} \;=\; \frac{h^2}{3}\binom{n_0+1}{2} \;\geq\; \frac{T}{M} \;. \tag{3.117}$$

This integer is given by

$$n_0 \;=\; \left\lceil \sqrt{\frac{2}{h^2} \cdot \frac{3\,T}{M} + \frac{1}{4}} - \frac{1}{2} \right\rceil \;, \tag{3.118}$$

and we define

$$q \;:=\; \frac{T}{M \cdot \theta_{n_0}} \;\leq\; 1 \;. \tag{3.119}$$

If we perform an FED cycle with the time step sizes $q \cdot \tau_i$, $i = 0, \ldots, n_0 - 1$, we obtain

$$\prod_{i=0}^{n_0-1} \left(\boldsymbol{I} \;+\; q \cdot \tau_i \, \boldsymbol{A_h}\right) \;=\; \boldsymbol{I} + \frac{T}{M}\boldsymbol{A_h} + \sum_{m=2}^{n_0} \frac{(qh^2)^m}{2m+1}\binom{n_0+m}{2m}\boldsymbol{A}_{\boldsymbol{h}}^{m} \;. \tag{3.120}$$

Thus, the coefficient of $\boldsymbol{A_h}$ is now the same as for the exponential series, and the FED scheme is consistent with at least first order. Note that we do not have any problems with the stability: The multiplication with $q \leq 1$ can also be seen as a scaling of the matrix $\boldsymbol{A_h}$, i.e.

$$\boldsymbol{I} + q \cdot \tau_i \, \boldsymbol{A_h} \;=\; \boldsymbol{I} + \tau_i \, (q \cdot \boldsymbol{A_h}) \; , \tag{3.121}$$

and the eigenvalues of this matrix range in

$$\left[-q \cdot \tfrac{4}{h^2} \, , \, 0\right] \;\subseteq\; \left[-\tfrac{4}{h^2} \, , \, 0\right] \; . \tag{3.122}$$

Since the symbol satisfies $|p_B^{[n_0]}(z)| \leq 1$ for $z \in [0 \, , \, \frac{4}{h^2}]$, this is also valid in particular for $z \in [0 \, , \, q \cdot \frac{4}{h^2}]$.

To determine whether the approximation order is larger than one, we compute the coefficient corresponding to $\boldsymbol{A_h^2}$ and compare it to the exact one of the exponential series, i.e. $\frac{T^2}{2\,M^2}$. Using Eq. (3.120), we have for this coefficient

$$\begin{aligned} \frac{q^2\,h^4}{5} \cdot \binom{n_0+2}{4} \;&=\; \frac{q\,h^2}{3} \cdot \binom{n_0+1}{2} \cdot \frac{q\,h^2}{20} \cdot (n_0+2)(n_0-1) \\ &=\; \frac{T}{M} \cdot \frac{q\,h^2}{20} \cdot (n_0+2)(n_0-1) \\ &=\; \frac{T}{M} \cdot \frac{q\,h^2}{20} \cdot \big((n_0+1)n_0 - 2\big) \\ &=\; \frac{T}{M} \cdot \left(\frac{3}{10} \cdot \frac{T}{M} \;-\; \frac{q\,h^2}{10}\right) \; . \end{aligned} \tag{3.123}$$

Thus, the condition for a second order approximation is

$$\frac{3}{10} \cdot \frac{T}{M} \;-\; \frac{q\,h^2}{10} \;=\; \frac{T}{2\,M} \; , \tag{3.124}$$

or equivalently

$$-\frac{1}{5} \cdot \frac{T}{M} \;=\; \frac{q\,h^2}{10} \; . \tag{3.125}$$

Since the left hand side is negative and the right hand side larger than zero, the condition for second order approximation can not be fulfilled.

To obtain a closed-form expression of the error and to show convergence, we want to rewrite the right hand side of Eq. (3.120) such that all coefficients of the series expansion are expressed in terms of $\frac{T}{M}$. For the coefficient of

$\boldsymbol{A}_{\boldsymbol{h}}^k$ with arbitrary $k \geq 2$ we have

$$\begin{aligned}
\frac{(q\,h^2)^k}{2k+1}\binom{n_0+k}{2k} &= \frac{(qh^2)^k}{(2k+1)!}\cdot\prod_{j=0}^{k-1}(n_0+k-j)\,(n_0-(k-j)+1) \\
&= \frac{6^k}{(2k+1)!}\cdot\prod_{j=0}^{k-1}\frac{q\,h^2}{6}\big(n_0(n_0+1)-(k-j-1)(k-j)\big) \\
&= \frac{6^k}{(2k+1)!}\cdot\prod_{j=0}^{k-1}\left(\frac{T}{M}-\frac{q\,h^2}{6}(k-j-1)(k-j)\right) \\
&= \frac{6^k}{(2k+1)!}\cdot\prod_{j=0}^{k-2}\left(1-\frac{\binom{k-j}{2}}{\binom{n_0+1}{2}}\right)\cdot\left(\frac{T}{M}\right)^k . \qquad (3.126)
\end{aligned}$$

Thus, the error concerning one cycle is given by

$$\begin{aligned}
&\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A}_{\boldsymbol{h}}\right)-\prod_{i=0}^{n_0-1}\left(\boldsymbol{I}+q\cdot\tau_i\,\boldsymbol{A}_{\boldsymbol{h}}\right) \\
=&\sum_{m=2}^{\infty}\left(\frac{1}{m!}-\frac{6^m}{(2m+1)!}\cdot\prod_{j=0}^{m-2}\left(1-\frac{\binom{m-j}{2}}{\binom{n_0+1}{2}}\right)\right)\left(\tfrac{T}{M}\cdot\boldsymbol{A}_{\boldsymbol{h}}\right)^m \\
=&\sum_{m=2}^{n_0}\left(\frac{1}{m!}-\frac{6^m}{(2m+1)!}\cdot\prod_{j=0}^{m-2}\left(1-\frac{\binom{m-j}{2}}{\binom{n_0+1}{2}}\right)\right)\left(\tfrac{T}{M}\cdot\boldsymbol{A}_{\boldsymbol{h}}\right)^m \\
&+\sum_{m=n_0+1}^{\infty}\frac{1}{m!}\left(\tfrac{T}{M}\cdot\boldsymbol{A}_{\boldsymbol{h}}\right)^m . \qquad (3.127)
\end{aligned}$$

For the estimation of the error, we use the standard Euclidean norm, and consider the difference of the symbols, i.e.

$$\exp\left(-\tfrac{T}{M}\cdot z\right)-p_B^{[n_0]}(q\cdot z)\ , \qquad (3.128)$$

with $z\in\left[0\,,\,\frac{4}{h^2}\right]$. Using the Lagrange remainder of the Taylor series and with suitable $\xi_1, \xi_2 \in \left[0\,,\,\frac{4}{h^2}\right]$ we get

$$\begin{aligned}
&\left|\exp\left(-\tfrac{T}{M}\cdot z\right) \;-\; p_B^{[n_0]}\left(q\cdot z\right)\right| \\
\le\;& \left|\exp\left(-\tfrac{T}{M}\cdot z\right) - \left(1-\tfrac{T}{M}\cdot z\right)\right| \;+\; \left|p_B^{[n_0]}\left(q\cdot z\right) - \left(1-\tfrac{T}{M}\cdot z\right)\right| \\
\le\;& \tfrac{1}{2}\left(\tfrac{T}{M}\right)^2\cdot\exp\left(-\tfrac{T}{M}\cdot\xi_1\right)\cdot z^2 \;+\; \tfrac{1}{2}\cdot\left|\left.\frac{d^2\,p_B^{[n_0]}\left(q\cdot z\right)}{dz^2}\right|_{z=\xi_2}\right|\cdot z^2 \\
\le\;& \tfrac{1}{2}\left(\tfrac{T}{M}\right)^2\cdot z^2 \;+\; \tfrac{q^2}{2}\cdot\max_{\xi\in\left[0\,,\,q\cdot\frac{4}{h^2}\right]}\left|\left(p_B^{[n_0]}\right)''(\xi)\right|\cdot z^2 \\
\le\;& \tfrac{1}{2}\left(\tfrac{T}{M}\right)^2\cdot z^2 \;+\; \tfrac{q^2}{2}\cdot\max_{\xi\in\left[0\,,\,\frac{4}{h^2}\right]}\left|\left(p_B^{[n_0]}\right)''(\xi)\right|\cdot z^2\,. \qquad (3.129)
\end{aligned}$$

In order to estimate the maximum of the second order derivative we use a result from Markoff [97]: If a polynomial p with degree n fulfils

$$\max_{x\in[-1\,,\,1]}|p(x)| \;\le\; 1\,, \qquad (3.130)$$

then its derivatives $p^{(k)}$ satisfy

$$\max_{x\in[-1\,,\,1]}|p^{(k)}(x)| \;\le\; \frac{n^2\,(n^2-1)\,\ldots\,(n^2-(k-1)^2)}{1\cdot 3\cdot\,\ldots\,\cdot(2k-1)}\,. \qquad (3.131)$$

Since we have

$$\max_{\xi\in\left[0\,,\,\frac{4}{h^2}\right]}\left|p_B^{[n_0]}(\xi)\right| \;=\; \max_{\tilde{\xi}\in[-1\,,\,1]}\left|p_B^{[n_0]}\left((\tilde{\xi}+1)\cdot\tfrac{2}{h^2}\right)\right| \;\le\; 1\,, \qquad (3.132)$$

the above result yields

$$\frac{4}{h^4}\cdot\max_{\tilde{\xi}\in[-1\,,\,1]}\left|\left(p_B^{[n_0]}\right)''\left((\tilde{\xi}+1)\cdot\tfrac{2}{h^2}\right)\right| \;\le\; \frac{n_0^2\,(n_0^2-1)}{3}\,, \qquad (3.133)$$

or equivalently

$$\max_{\xi\in\left[0\,,\,\frac{4}{h^2}\right]}\left|\left(p_B^{[n_0]}\right)''(\xi)\right| \;\le\; h^4\cdot\frac{n_0^2\,(n_0^2-1)}{12} \;=\; 3\cdot\left(\theta_{n_0}\right)^2\cdot\underbrace{\frac{n_0-1}{n_0+1}}_{\le 1}\,. \qquad (3.134)$$

Thus, we obtain

$$
\begin{aligned}
&\tfrac{1}{2}\left(\tfrac{T}{M}\right)^2 \cdot z^2 \;+\; \tfrac{q^2}{2} \cdot \max_{\xi\in\left[0\,,\,\frac{4}{h^2}\right]} \left|\left(p_B^{[n_0]}\right)''(\xi)\right| \cdot z^2 \\
\le\;& \tfrac{1}{2}\left(\tfrac{T}{M}\right)^2 \cdot z^2 \;+\; \tfrac{3}{2}\underbrace{\left(q\cdot\theta_{n_0}\right)^2}_{=\left(\frac{T}{M}\right)^2} \cdot z^2 \;=\; 2\,z^2\cdot\left(\tfrac{T}{M}\right)^2 \;. \qquad (3.135)
\end{aligned}
$$

Because of $z \in \left[0\,,\,\frac{4}{h^2}\right]$, we finally have the upper bound $C(h)\cdot\left(\frac{T}{M}\right)^2$ with $C(h) := \frac{32}{h^4}$, i.e.

$$
\left\|\exp\left(\tfrac{T}{M}\cdot \boldsymbol{A_h}\right) \;-\; \prod_{i=0}^{n_0-1}\left(\boldsymbol{I} \;+\; q\cdot\tau_i\,\boldsymbol{A_h}\right)\right\|_2 \;\le\; C(h)\cdot\left(\tfrac{T}{M}\right)^2 \;. \qquad (3.136)
$$

Now we want to estimate the global error after M FED cycles. To this end, we define for $k = 1,\ldots,M$ the numerical solutions after k cycles by

$$
\boldsymbol{u}^k \;:=\; \left(\prod_{i=0}^{n_0-1}\left(\boldsymbol{I} \;+\; q\cdot\tau_i\,\boldsymbol{A_h}\right)\right)^k \boldsymbol{u}^0\,, \qquad (3.137)
$$

and the ones after a cycle with exact data $\boldsymbol{u}(t_{k-1})$ by

$$
\tilde{\boldsymbol{u}}^k \;:=\; \left(\prod_{i=0}^{n_0-1}\left(\boldsymbol{I} \;+\; q\cdot\tau_i\,\boldsymbol{A_h}\right)\right)\boldsymbol{u}(t_{k-1}) \qquad (k=1,\ldots,M)\,, \qquad (3.138)
$$

where $t_{k-1} := (k-1)\cdot\frac{T}{M}$. To estimate the global error, we use that

$$
\begin{aligned}
&\left\|\boldsymbol{u}\left(t_M\right) \;-\; \tilde{\boldsymbol{u}}^M\right\|_2 \\
=\;& \left\|\left(\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right) \;-\; \prod_{i=0}^{n_0-1}\left(\boldsymbol{I} \;+\; q\cdot\tau_i\,\boldsymbol{A_h}\right)\right)\boldsymbol{u}\left(t_{M-1}\right)\right\|_2 \\
\le\;& \left\|\exp\left(\tfrac{T}{M}\cdot\boldsymbol{A_h}\right) \;-\; \prod_{i=0}^{n_0-1}\left(\boldsymbol{I} \;+\; q\cdot\tau_i\,\boldsymbol{A_h}\right)\right\|_2 \cdot\left\|\boldsymbol{u}\left(t_{M-1}\right)\right\|_2 \\
\overset{(3.136)}{\le}\;& \left(\tfrac{T}{M}\right)^2\cdot C(h)\cdot\left\|\boldsymbol{u}\left(t_{M-1}\right)\right\|_2\;, \qquad (3.139)
\end{aligned}
$$

as well as

$$\begin{aligned}
&\left\|\tilde{\boldsymbol{u}}^M - \boldsymbol{u}^M\right\|_2 \\
&= \left\| \left(\prod_{i=0}^{n_0-1} \left(\boldsymbol{I} + q \cdot \tau_i\, \boldsymbol{A_h}\right) \right) \left(\boldsymbol{u}\left(t_{M-1}\right) - \boldsymbol{u}^{M-1} \right) \right\|_2 \\
&\leq \underbrace{\left\| \prod_{i=0}^{n_0-1} \left(\boldsymbol{I} + q \cdot \tau_i\, \boldsymbol{A_h}\right) \right\|_2}_{\leq 1} \cdot \left\| \boldsymbol{u}\left(t_{M-1}\right) - \boldsymbol{u}^{M-1} \right\|_2 \\
&\leq \left\| \boldsymbol{u}\left(t_{M-1}\right) - \boldsymbol{u}^{M-1} \right\|_2 \,, \qquad (3.140)
\end{aligned}$$

and

$$\begin{aligned}
\left\|\boldsymbol{u}\left(t_k\right)\right\|_2 &= \left\| \exp\left(t_k \cdot \boldsymbol{A_h}\right)\, \boldsymbol{u}^0 \right\|_2 \\
&\leq \underbrace{\left\| \exp\left(t_k \cdot \boldsymbol{A_h}\right) \right\|_2}_{\leq 1} \cdot \left\|\boldsymbol{u}^0\right\|_2 \leq \left\|\boldsymbol{u}^0\right\|_2 \,. \qquad (3.141)
\end{aligned}$$

Then the global error satisfies

$$\begin{aligned}
&\left\| \boldsymbol{u}\left(t_M\right) - \boldsymbol{u}^M \right\|_2 \\
&\leq \left\| \boldsymbol{u}\left(t_M\right) - \tilde{\boldsymbol{u}}^M \right\|_2 + \left\| \tilde{\boldsymbol{u}}^M - \boldsymbol{u}^M \right\|_2 \\
&\leq \left(\tfrac{T}{M}\right)^2 \cdot C(h) \cdot \left\|\boldsymbol{u}\left(t_{M-1}\right)\right\|_2 + \left\| \boldsymbol{u}\left(t_{M-1}\right) - \boldsymbol{u}^{M-1} \right\|_2 \\
&\leq \left(\tfrac{T}{M}\right)^2 \cdot C(h) \cdot \left\|\boldsymbol{u}^0\right\|_2 + \left\| \boldsymbol{u}\left(t_{M-1}\right) - \boldsymbol{u}^{M-1} \right\|_2 \\
&\leq 2\left(\tfrac{T}{M}\right)^2 \cdot C(h) \cdot \left\|\boldsymbol{u}^0\right\|_2 + \left\| \boldsymbol{u}\left(t_{M-2}\right) - \boldsymbol{u}^{M-2} \right\|_2 \\
&\vdots \\
&\leq M\left(\tfrac{T}{M}\right)^2 \cdot C(h) \cdot \left\|\boldsymbol{u}^0\right\|_2 + \underbrace{\left\| \boldsymbol{u}\left(t_0\right) - \boldsymbol{u}^0 \right\|_2}_{=0} \\
&= \tfrac{T^2}{M} \cdot C(h) \cdot \left\|\boldsymbol{u}^0\right\|_2 \,. \qquad (3.142)
\end{aligned}$$

We can summarise the results in the following theorem:

Theorem 3.6 (Error Estimation for FED Scheme). *Let the function $\boldsymbol{u} : (0,\infty) \to \mathbb{R}^N$ be the exact solution of the system of ODEs* (3.107) *with initial data $\boldsymbol{u}^0 = \boldsymbol{u}(0)$, $T > 0$ and $M \in \mathbb{N}$ the number of FED cycles. If the cycle length $n_0 \in \mathbb{N}$ and $q \in (0,1]$ are given according to Eq.* (3.118) *and Eq.* (3.119)*, respectively, then the numerical solution after M FED cycles,*

$$\boldsymbol{u}^M \;:=\; \left(\prod_{i=0}^{n_0-1} \left(\boldsymbol{I} \,+\, q \cdot \tau_i \, \boldsymbol{A_h}\right) \right)^M \boldsymbol{u}^0 \;, \tag{3.143}$$

fulfils

$$\left\| \boldsymbol{u}(T) \,-\, \boldsymbol{u}^M \right\|_2 \;\leq\; \frac{32}{h^4} \cdot \frac{T^2}{M} \cdot \left\| \boldsymbol{u}^0 \right\|_2 \;. \tag{3.144}$$

Using this theorem, we can show that the FED scheme converges to the exact solution. If one increases the number of cycles M, then Eq. (3.144) yields

$$\lim_{M\to\infty} \left\| \boldsymbol{u}(T) \,-\, \boldsymbol{u}^M \right\|_2 \;=\; 0 \;. \tag{3.145}$$

Thus, we have the convergence

$$\boldsymbol{u}^M \;\overset{M\to\infty}{\longrightarrow}\; \boldsymbol{u}(T) \;. \tag{3.146}$$

Since the error is proportional to $\frac{1}{M}$, the corresponding approximation order is one with respect to the number of cycles M.

3.2.5 Extension to Arbitrary Diffusion Problems

The FED scheme has been motivated in the 1-D setting with explicit linear diffusion filtering. However, in this case, it does not have any practical use, since a direct computation of box filtering by means of e.g. the sliding-window approach from Eq. (2.14) is much more efficient than a factorisation. To this end, we want to show that it is actually a general paradigm which can be applied e.g. to multidimensional, nonlinear, isotropic or anisotropic diffusion processes. This can be seen as follows.

We consider the general d-dimensional diffusion equation [147]

$$\partial_t u(\boldsymbol{x},t) \;=\; \operatorname{div}\left(\boldsymbol{D}\, \boldsymbol{\nabla} u(\boldsymbol{x},t)\right) \;, \tag{3.147}$$

where $\boldsymbol{x} \in \Omega \subset \mathbb{R}^d$, $u : \Omega \times (0,\infty) \to \mathbb{R}$ is the unknown solution, $\boldsymbol{D} \in \mathbb{R}^{d\times d}$ a symmetric, positive definite diffusion tensor. Note that we assume a rectangular region

$$\Omega \;=\; [a_1, b_1] \,\times\, \ldots \,\times\, [a_d, b_d] \;, \tag{3.148}$$

with $a_k, b_k \in \mathbb{R}$. If $\boldsymbol{D}$ depends on u, e.g. $\boldsymbol{D} = \boldsymbol{D}\left(\boldsymbol{\nabla} u_\sigma(\boldsymbol{x}, t)\right)$, we call the equation nonlinear. As before, u_σ is the convolution of u and a d-dimensional Gaussian with standard deviation σ, $u_\sigma = G_\sigma * u$. Furthermore, we denote the given initial function by $u_0(\boldsymbol{x}) = u(\boldsymbol{x}, 0)$ and have to impose boundary conditions for $\boldsymbol{x} \in \partial\Omega$. We consider again homogeneous Neumann boundary conditions:

$$\boldsymbol{n}^T\big(\boldsymbol{D}\,\boldsymbol{\nabla} u(\boldsymbol{x}, t)\big) \;=\; 0 \quad , \; (\boldsymbol{x}, t) \in \partial\Omega \times (0, \infty) \tag{3.149}$$

with the outer unit normal vector $\boldsymbol{n}$ of $\partial\Omega$.

To solve Eq. (3.147), we proceed as in the linear 1-D case, and use again the method of lines. Therefore, we introduce 1-D spatial grids

$$\mathcal{G}_k \;:=\; \left\{a_k + \left(i_k - \tfrac{1}{2}\right) \cdot h_k \;\middle|\; 1 \le i_k \le N_k\right\} \qquad (k = 1, \dots d)\,, \tag{3.150}$$

with the corresponding mesh sizes

$$h_k \;=\; \frac{b_k - a_k}{N_k} \qquad (k = 1, \dots d)\,, \tag{3.151}$$

and the number of nodes N_k. Using these grids, we can construct a d-dimensional grid by means of

$$\mathcal{G}_1 \times \cdots \times \mathcal{G}_d \;\subset\; \Omega\,. \tag{3.152}$$

Each node $\boldsymbol{x}_{\boldsymbol{i}}$ is determined by its indices $\boldsymbol{i} = (i_1, \dots, i_d)$. However, for the spatial discretisation we want to use a so-called single-index notation for $d \ge 2$. To this end, we identify such a node $\boldsymbol{x}_m$ by the index

$$m(\boldsymbol{i}) \;=\; i_1 + (i_2 - 1) \cdot N_1 + (i_3 - 1) \cdot N_1 \cdot N_2 + \dots + (i_d - 1) \cdot \prod_{k=1}^{d-1} N_k\,, \tag{3.153}$$

and define the time-dependent vector $\boldsymbol{u}(t) \in \mathbb{R}^N$ with $N = \prod_{k=1}^{d} N_k$:

$$\left(\boldsymbol{u}(t)\right)_m \;\approx\; u\left(\boldsymbol{x}_m, t\right) \qquad (m = 1, \dots, N)\,. \tag{3.154}$$

Then we rewrite the PDE (3.147) taking into account the boundary conditions as the time-dependent ODE system

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{P}\boldsymbol{u}\,, \tag{3.155}$$

with a negative semi-definite system matrix $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ that represents the spatial discretisation of the divergence operator. Usually, $\boldsymbol{P}$ is a symmetric (sparse) band matrix, where the number of off-diagonals depends on the discretisation. More precisely, it corresponds to the number of spatial neighbours of a node that are used with respect to the discretisation scheme. In the case of nonlinear problems, the entries of $\boldsymbol{P}$ depend on $\boldsymbol{u}$, i.e. $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{u})$.

Our goal is to construct an FED scheme for the solution of Eq. (3.155). To this end, we have to consider the eigenvalues of $\boldsymbol{P}$. Since $\boldsymbol{P}$ is assumed to be symmetric and negative semi-definite, all eigenvalues are real-valued and non-positive. We denote the largest modulus of the eigenvalues by $\mu_{\max}$. If we would apply one FED cycle with length n and use the time step sizes τ_i from Eq. (3.40), the stability in the Euclidean norm is only guaranteed for $\mu_{\max} \leq \frac{4}{h^2}$. For $\mu_{\max} > \frac{4}{h^2}$, it might happen that

$$\left\| \prod_{i=0}^{n-1} (\boldsymbol{I} + \tau_i \boldsymbol{P}) \right\|_2 > 1 , \tag{3.156}$$

which yields an unstable scheme. To ensure that the norm of this matrix product is bounded by 1, we can decrease the time step sizes by multiplication with a suitable factor $c < 1$. This is similar to the time adjustment that we have used in the last subsection. Here we adjust the time step sizes to the eigenvalues of $\boldsymbol{P}$. We define

$$c := \frac{4}{h^2 \cdot \mu_{\max}} , \tag{3.157}$$

such that $c \cdot \mu_{\max} = \frac{4}{h^2}$, and the time step sizes are

$$\tau_i' := c \cdot \tau_i = \frac{2}{\mu_{\max}} \cdot \frac{1}{2 \cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \ldots, n-1) . \tag{3.158}$$

Actually, the constant $\frac{2}{\mu_{\max}}$ is the time step size limit for the usual explicit scheme. Hence, we have just replaced the limit in the linear 1-D case, $\tau = \frac{h^2}{2}$, by the new limit $\tau_{\text{lim}} := \frac{2}{\mu_{\max}}$. Overall, we end up with a stable matrix product

$$\prod_{i=0}^{n-1} (\boldsymbol{I} + \tau_i' \boldsymbol{P}) . \tag{3.159}$$

The corresponding cycle time is given by

$$\theta_n^{\boldsymbol{P}} = \tau_{\text{lim}} \cdot \frac{n(n+1)}{3} . \tag{3.160}$$

Note that we can also have factors $c > 1$ for matrices $\boldsymbol{P}$ with $\mu_{\max} < \frac{4}{h^2}$, i.e. $\tau_{\text{lim}} > \frac{h^2}{2}$.

So far, we have only clarified stability issues for one FED cycle. However, we have seen in the linear 1-D case that increasing the number of cycles improves the accuracy of FED. This can simply be transferred to the d-dimensional linear case, since the matrix $\boldsymbol{P}$ remains constant. Concerning nonlinear parabolic problems, it is necessary to update the matrix $\boldsymbol{P}$, because it depends on $\boldsymbol{u}(t)$. We have seen that a rearrangement of the time step sizes can lead to instabilities within a cycle. Thus, updating the nonlinearities within a cycle might be risky and yield bad results. To guarantee reasonable results, we should update the matrix $\boldsymbol{P}$ only at those points, where we can expect a stable and useful intermediate result. These requirements are met, in particular, after a whole FED cycle. Actually, this means we keep the matrix $\boldsymbol{P}$ constant over one complete cycle, update it afterwards and perform another cycle with the new $\boldsymbol{P}$. Since the range of the eigenvalues might change after each update, we have to take care of the stability and adjust the time step sizes for each cycle. However, in many cases it is possible to estimate a spectrum that is independent of $\boldsymbol{u}(t)$. This has the advantage that the set of time step sizes has to be computed only once and the stability is guaranteed for every FED cycle. In our numerical experiments, we will use this solution.

According to our notation in Eq. (3.137), the k-th cycle with $k \geq 1$ of the FED scheme then reads

$$\boldsymbol{u}^k = \left(\prod_{i=0}^{n-1} \Big(\boldsymbol{I} + q \cdot \tau_i' \, \boldsymbol{P}\big(\boldsymbol{u}^{k-1}\big) \Big) \right) \boldsymbol{u}^{k-1}, \qquad (3.161)$$

with a suitable time adjustment factor $q \leq 1$. This approach can be seen as a linearisation, similar to so-called lagged diffusivity fixed point methods [28, 143, 144]. More precisely, with $\boldsymbol{u}^{k,0} := \boldsymbol{u}^{k-1}$ the k-th cycle can be written as

$$\boldsymbol{u}^{k,\,i+1} = \Big(\boldsymbol{I} + q \cdot \tau_i' \, \boldsymbol{P}\big(\boldsymbol{u}^{k-1}\big) \Big) \boldsymbol{u}^{k,\,i} \qquad (i = 0, \ldots, n-1), \qquad (3.162)$$

and results in $\boldsymbol{u}^k := \boldsymbol{u}^{k,n}$, where the matrix $\boldsymbol{P}(\boldsymbol{u}^{k-1})$ does not change during the inner iterations $i = 0, \ldots, n-1$. In the sense of e.g. [56, 57], we may interpret such an FED cycle as a single *super time step* with the nonlinearities $\boldsymbol{P}(\boldsymbol{u}^{k-1})$.

L^∞-Stability for Isotropic Processes

So far, we have analysed the stability of FED by means of the Euclidean norm. However, when it comes to isotropic parabolic processes, it can be shown that usual semi-implicit or explicit schemes satisfy stability constraints referring to the infinity norm $\|\boldsymbol{u}\|_\infty := \max_i |u_i|$ [147], i.e. L^∞-stability. More precisely, given the current numerical solution $\boldsymbol{u}^k$, the application of an L^∞-stable scheme yields $\boldsymbol{u}^{k+1}$ fulfilling the maximum-minimum principle

$$\min_i u_i^k \;\leq\; u_j^{k+1} \;\leq\; \max_i u_i^k \tag{3.163}$$

for all j. In the context of FED, we want to find out whether the constraint

$$\left\| \prod_{i=0}^{n-1} (\boldsymbol{I} + \tau_i' \boldsymbol{P}) \right\|_\infty \;\leq\; 1 \tag{3.164}$$

is valid for matrices $\boldsymbol{P}$ discretising an isotropic divergence operator, or if there exists a counter example.

Unfortunately, we could find some 1-D counter examples. To present such a counter example, we choose $N = 6$, because the case $N \leq 5$ does not seem to violate the above constraint. Our example matrix $\boldsymbol{P} \in \mathbb{R}^{6\times 6}$ is given by

$$\begin{pmatrix} -0.8564 & 0.8564 & 0 & 0 & 0 & 0 \\ 0.8564 & -1.8357 & 0.9793 & 0 & 0 & 0 \\ 0 & 0.9793 & -1.1773 & 0.1980 & 0 & 0 \\ 0 & 0 & 0.1980 & -0.3196 & 0.1216 & 0 \\ 0 & 0 & 0 & 0.1216 & -0.9721 & 0.8506 \\ 0 & 0 & 0 & 0 & 0.8506 & -0.8506 \end{pmatrix} . \tag{3.165}$$

Furthermore, we use an FED cycle with length $n = 7$ and the usual time step sizes given in Eq. (3.40). This method is stable with respect to the Euclidean norm. However, the fourth diagonal entry of the matrix $\prod_{i=0}^{6} (\boldsymbol{I} + \tau_i \boldsymbol{P})$ is negative (≈ -0.0029), which means that the vector $\boldsymbol{v} = (1, 1, 1, -1, 1, 1)^T$ fulfils $\|\boldsymbol{v}\|_\infty = 1$, but the resulting matrix-vector product yields

$$\left\| \prod_{i=0}^{6} (\boldsymbol{I} + \tau_i \boldsymbol{P})\, \boldsymbol{v} \right\|_\infty \;\approx\; 1.0058 \;>\; 1\, . \tag{3.166}$$

Thus, this example demonstrates that FED can violate the L^∞-stability criterion for isotropic problems.

3.2.6 Implementation

Now we want to give a summary of the general FED algorithm that is shown in Fig. 3.11. It basically works like in the linear 1-D case, but one has to consider the actual time step limit τ_{lim}. According to this limit, we compute the cycle length n and the time factor q such that the given stopping time $T > 0$ of the diffusion process is reached with the desired number M of FED cycles.

If there already exists an explicit code, one has to add only a few code lines: The computation of the cycle length n, the factor q and the time step sizes for one FED cycle. Concerning the rearrangement of the step sizes, one could use an external database. For nonlinear problems, the existing update procedure can be used, but one has to take care that it is used only after complete FED cycles.

Thus, FED is essentially an explicit scheme with some overhead that is not time critical. It is very easy to implement and moreover, if one uses multicore architectures or GPU's, well-suited for parallel computing.

3.3 Cascadic FED (CFED)

Our FED scheme was designed for diffusion-like problems where we are interested in the temporal evolution. They correspond to parabolic PDEs. However, let us now explain how we can use FED ideas also for elliptic PDEs. They can appear e.g. as Euler–Lagrange equations for variational image analysis methods, or as nontrivial steady state of parabolic evolutions with additional reaction terms. We have basically two possibilities: We can approximate a solution by means of a parabolic process with a large stopping time, or we directly solve the corresponding elliptic equation.

In this section, we consider the first choice that implies the application of a parabolic FED scheme. To reach this steady state as quickly as possible, we embed our FED method into a coarse-to-fine strategy [12, 37], i.e. we use results computed on a coarse level as an initialisation for a finer scale. This can be regarded as a simple multigrid approach [14, 16, 68]. It saves a lot of computational effort, since a small or medium stopping time is already sufficient on each level. Therefore, we scale down the image data via linear interpolation to a certain coarse level and apply the FED scheme on this image. Afterwards we interpolate the corresponding solution to the next finer level and apply again FED. We use this procedure recursively until the finest (original) level is reached. We call this the *Cascadic Fast Explicit Diffusion (CFED)* approach.

1. **Input Data**:
   image f, stopping time T, number M of FED cycles, and explicit step size limit τ_{lim}

2. **Initialisation**:

   (a) Compute the length n of one FED cycle (for Lebedev-Finogenov: $n = 2^p$),

   $$n = \left\lceil \sqrt{\frac{1}{\tau_{\text{lim}}} \cdot \frac{3\,T}{M} + \frac{1}{4}} - \frac{1}{2} \right\rceil ,$$

   and the corresponding time adjustment factor

   $$q = \frac{T}{M \cdot \theta_n^{\boldsymbol{P}}} \leq 1 .$$

   (b) Compute the time step sizes $q \cdot \tau_i'$ (cf. Eq. (3.158)).

   (c) Choose a suitable ordering for the step sizes (e.g. with κ-cycles, Leja ordering or Lebedev-Finogenov ordering), i.e. $\left\{q \cdot \tilde{\tau}_0', \ldots, q \cdot \tilde{\tau}_{n-1}'\right\}$.

   (d) If the diffusivity or diffusion tensor is constant in time, compute the corresponding matrix $\boldsymbol{P}$.

3. **Filtering Loop**: (M times)

   (a) If the diffusivity or diffusion tensor is time-variant, update it and compute the corresponding matrix $\boldsymbol{P}$.

   (b) Perform one FED cycle with the above ordering of the n explicit time steps.

Figure 3.11: General FED algorithm for diffusion filtering.

However, for inpainting and PDE-based compression problems [54], the cascadic approach is a bit more complicated. This is due to the fact that besides the image data, we have to scale down also a so-called *confidence map* or *inpainting mask* that specifies Dirichlet boundary data.

Let us now discuss this problem in detail. The elliptic PDE for image inpainting is given by [54]

$$\operatorname{div}\left(\boldsymbol{D}\left(\boldsymbol{\nabla} v_\sigma(\boldsymbol{x})\right) \boldsymbol{\nabla} v(\boldsymbol{x})\right) \;=\; 0\;, \tag{3.167}$$

where we want to have the solution $v : \Omega \to \mathbb{R}$. As above, the domain Ω is assumed to be rectangular, i.e. $\Omega = [a_1, b_1] \times \ldots \times [a_d, b_d]$. The trivial solution of Eq. (3.167) is $v \equiv 0$. Thus, we specify a set (not a null set) $Z \subset \Omega$ with the Dirichlet condition

$$v(\boldsymbol{x}) \;=\; f(\boldsymbol{x}) \tag{3.168}$$

for all $\boldsymbol{x} \in Z$, where $f : \Omega \to \mathbb{R}$ represents the given data. If the confidence map is $c : \Omega \to [0,1]$, we can formulate the problem in the single equation [54]

$$c(\boldsymbol{x})\cdot(v(\boldsymbol{x}) - f(\boldsymbol{x})) - (1 - c(\boldsymbol{x}))\cdot \operatorname{div}\left(\boldsymbol{D}\left(\boldsymbol{\nabla} v_\sigma(\boldsymbol{x})\right) \boldsymbol{\nabla} v(\boldsymbol{x})\right) \;=\; 0\;. \tag{3.169}$$

The Dirichlet condition implies $c(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in Z$ and in the case of e.g. $c(\boldsymbol{x}) = 0$ this function allows to fill in new data at the corresponding locations $\boldsymbol{x}$. Hence, the value $c(\boldsymbol{x})$ should reflect e.g. the quality of the given image data $f(\boldsymbol{x})$.

We can discretise Eq. (3.169) on a spatial grid in the same manner as above (single-index notation) and get a nonlinear system of equations

$$\boldsymbol{C}\,(\boldsymbol{v} - \boldsymbol{f}) \;-\; (\boldsymbol{I} \;-\; \boldsymbol{C})\,\boldsymbol{P}(\boldsymbol{v})\,\boldsymbol{v} \;=\; \boldsymbol{0}\;, \tag{3.170}$$

with the (diagonal) confidence matrix

$$\boldsymbol{C} \;=\; \begin{pmatrix} c(\boldsymbol{x}_1) & 0 & 0 & \ldots & 0 \\ 0 & c(\boldsymbol{x}_2) & 0 & \ldots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \ldots & 0 & c(\boldsymbol{x}_N) \end{pmatrix} \in \mathbb{R}^{N\times N}\;, \tag{3.171}$$

and the vectors $\boldsymbol{v} = (v(\boldsymbol{x}_m))_{m=1}^{N} \in \mathbb{R}^N$ as well as $\boldsymbol{f} = (f(\boldsymbol{x}_m))_{m=1}^{N} \in \mathbb{R}^N$. In the following, we assume that the entries of the confidence matrix are always in $\{0, 1\}$.

We solve the nonlinear equation (3.170) using an explicit scheme that is based on a parabolic evolution with the time step size $\tau > 0$

$$\boldsymbol{v}^{k+1} \;=\; \boldsymbol{v}^k \;+\; \tau\cdot\Big(\boldsymbol{C}\,\big(\boldsymbol{f} \;-\; \boldsymbol{v}^k\big) \;+\; (\boldsymbol{I} \;-\; \boldsymbol{C})\,\boldsymbol{P}\big(\boldsymbol{v}^{\boldsymbol{k}}\big)\;\boldsymbol{v}^k\Big)\;, \tag{3.172}$$

with $k \geq 0$ and $\boldsymbol{v}^0 := \boldsymbol{f}$. Actually, if we use this initialisation, then

$$\boldsymbol{C}\boldsymbol{v}^k \;=\; \boldsymbol{C}\boldsymbol{v}^0 \;=\; \boldsymbol{C}\boldsymbol{f} \tag{3.173}$$

for all k. Thus, the scheme simplifies to

$$\boldsymbol{v}^{k+1} \;=\; \boldsymbol{v}^k \;+\; \tau \cdot (\boldsymbol{I} \,-\, \boldsymbol{C})\, \boldsymbol{P}\big(\boldsymbol{v}^k\big)\; \boldsymbol{v}^k \tag{3.174}$$

Applying the results from [104], the spectral radius of $(\boldsymbol{I} - \boldsymbol{C})\,\boldsymbol{P}(\cdot)$ is bounded by the maximum absolute value $\mu_{\max}$ of the eigenvalues of $\boldsymbol{P}(\cdot)$. Therefore, τ is limited by the parabolic time step size limit $\tau_{\text{lim}} = \frac{2}{\mu_{\max}}$. Since we want to reach the steady state

$$\boldsymbol{v} \;:=\; \lim_{k\to\infty} \boldsymbol{v}^k \tag{3.175}$$

as quickly as possible, we use the CFED scheme. To this end, let us now discuss the corresponding coarse-to-fine strategy.

3.3.1 Coarse-to-Fine Strategy for Inpainting

More precisely, we consider the image data $f^{(0)} := f$ as well as the confidence map $c^{(0)} := c$ on the original grid with mesh sizes $h_k^{(0)} := h_k$, $k = 1, \ldots, d$, and the number of grid points $N^{(0)} = \prod N_k^{(0)}$. At grid points $\boldsymbol{x}_{\boldsymbol{i}}^{(0)} := \boldsymbol{x}_{\boldsymbol{i}}$ with $c^{(0)}(\boldsymbol{x}_{\boldsymbol{i}}^{(0)}) = 0$, we assume that $f^{(0)}(\boldsymbol{x}_{\boldsymbol{i}}^{(0)}) = 0$. Then we scale down both the image $f^{(0)} \to f^{(1)}$ and the confidence map $c^{(0)} \to c^{(1)}$ with the help of a suitable *restriction operator*. The corresponding coarser grid consists of $N_k^{(1)} = \lceil \frac{1}{2} \cdot N_k^{(0)} \rceil$ nodes in each direction k and the mesh sizes according to Eq. (3.151) are given by

$$h_k^{(1)} \;=\; \frac{b_k \,-\, a_k}{N_k^{(1)}} \;>\; h_k^{(0)} \qquad (k = 1, \ldots, d)\,. \tag{3.176}$$

After the application of the restriction operator, the values of the coarser confidence map $c^{(1)}$ do not have to be limited to the set $\{0, 1\}$. Thus, we introduce a normalised map $\tilde{c}^{(1)}$ via

$$\tilde{c}^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \;:=\; \begin{cases} 1 \;,\; c^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \,>\, 0 \\[2ex] 0 \;,\; c^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \,=\, 0 \end{cases} \,, \tag{3.177}$$

and a corresponding normalised coarser image

$$\tilde{f}^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \;:=\; \begin{cases} \dfrac{f^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)})}{c^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)})} \;,\; c^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \,>\, 0 \\[3ex] f^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \;\;,\; c^{(1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(1)}) \,=\, 0 \end{cases} \,. \tag{3.178}$$

Then these normalised data are restricted to the next coarser level, afterwards again normalised and so on, until the desired or coarsest level L is reached. On this level, we apply M_L FED cycles with length n_L and suitable time step sizes $\tau_i^{(L)}$, $i = 0, \dots, n_L - 1$. They are adapted to the spectrum of the matrix coming from the discretisation of the divergence operator on the coarsest grid, $\boldsymbol{P}^{(L)}(\tilde{\boldsymbol{f}}^{(L)}) \in \mathbb{R}^{N(L) \times N(L)}$:

$$\boldsymbol{u}^{(L)} := \left(\prod_{i=0}^{n_L - 1} \left(\boldsymbol{I} + \tau_i^{(L)} \cdot \left(\boldsymbol{I} - \boldsymbol{C}^{(L)} \right) \boldsymbol{P}^{(L)}(\tilde{\boldsymbol{f}}^{(L)}) \right) \right)^{M_L} \tilde{\boldsymbol{f}}^{(L)} . \tag{3.179}$$

We still use the single index notation $\tilde{\boldsymbol{f}}^{(L)} = \left(\tilde{f}^{(L)}(\boldsymbol{x}_m^{(L)}) \right)_{m=1}^{N(L)} \in \mathbb{R}^{N(L)}$, and the confidence matrix $\boldsymbol{C}^{(L)} \in \mathbb{R}^{N(L) \times N(L)}$ is derived from the confidence map $\tilde{c}^{(L)}$. Having computed the vector $\boldsymbol{u}^{(L)}$ and the corresponding image data $u^{(L)}$, we interpolate it to next finer level with a suitable *prolongation operator*, $u^{(L)} \to v^{(L-1)}$. Afterwards we adapt $v^{(L-1)}$ to the corresponding confidence map $\tilde{c}^{(L-1)}$ and the image data $\tilde{f}^{(L-1)}$ via

$$\tilde{v}^{(L-1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(L-1)}) := \begin{cases} \tilde{f}^{(L-1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(L-1)}) & , \tilde{c}^{(L-1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(L-1)}) = 1 \\ v^{(L-1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(L-1)}) & , \tilde{c}^{(L-1)}(\boldsymbol{x}_{\boldsymbol{i}}^{(L-1)}) = 0 \end{cases} . \tag{3.180}$$

Using the single index notation for $\tilde{\boldsymbol{v}}^{(L-1)} \in \mathbb{R}^{N(L-1)}$ we can compute the matrix $\boldsymbol{P}^{(L-1)}(\tilde{\boldsymbol{v}}^{(L-1)}) \in \mathbb{R}^{N(L-1) \times N(L-1)}$ and with the n_{L-1} adjusted time step sizes $\tau_i^{(L-1)}$ the FED scheme on level $L-1$ reads

$$\boldsymbol{u}^{(L-1)} := \left(\prod_{i=0}^{n_{L-1} - 1} \left(\boldsymbol{I} + \tau_i^{(L-1)} \cdot \left(\boldsymbol{I} - \boldsymbol{C}^{(L-1)} \right) \boldsymbol{P}^{(L-1)}(\tilde{\boldsymbol{v}}^{(L-1)}) \right) \right)^{M_{L-1}} \tilde{\boldsymbol{v}}^{(L-1)} . \tag{3.181}$$

By repeating the above procedure for all further levels $L-2, \dots, 0$, we end up with $\boldsymbol{u}^{(0)}$, which is an approximation for the steady state $\boldsymbol{v}$. For simplicity, the above schemes keep the matrices $\boldsymbol{P}^{(L)}(\cdot)$ and $\boldsymbol{P}^{(L-1)}(\cdot)$ constant during all M_L and M_{L-1} cycles, respectively. However, to improve the accuracy for nonlinear problems, we recommend to update the nonlinearities before a new cycle starts. This would correspond to an overall number of $M_L + M_{L-1} + \cdots + M_0$ updates.

3.3.2 Implementation

A summary of the cascadic FED algorithm for inpainting is shown in Fig. 3.12. Note that existing FED routines can be used straightforward.

1. **Input Data**:
   image f and confidence map c, number of coarser levels L, and numbers M_j ($j = 0, \dots, L$) of FED cycles with the corresponding lengths n_j

2. **Initialisation**:

   (a) Compute initial image $f^{(0)}$ by $f^{(0)}(\boldsymbol{x}) = f(\boldsymbol{x}) \cdot c(\boldsymbol{x})$.

   (b) $\tilde{f}^{(0)} := f^{(0)}$ and $\tilde{c}^{(0)} := c$.

   (c) For $j = 0, \dots, L-1$:

   - ∗ Interpolate both $\tilde{f}^{(j)}$ and $\tilde{c}^{(j)}$ to the next coarser grid level, $\tilde{f}^{(j)} \to f^{(j+1)}$ and $\tilde{c}^{(j)} \to c^{(j+1)}$.
   - ∗ Normalise $f^{(j+1)}$ and $c^{(j+1)}$, such that the normalised confidence map $\tilde{c}^{(j+1)}$ has got values in $\{0, 1\}$, i.e. $\tilde{f}^{(j+1)}(\boldsymbol{x}) := \frac{f^{(j+1)}(\boldsymbol{x})}{c^{(j+1)}(\boldsymbol{x})}$ and $\tilde{c}^{(j+1)}(\boldsymbol{x}) = 1$ for the case $c^{(j+1)}(\boldsymbol{x}) > 0$.
   - ∗ Store $\tilde{f}^{(j+1)}$ and $\tilde{c}^{(j+1)}$.

3. **Filtering Loop**: (initialise $j = L$)

   (a) Determine the time step size limit $\tau_{\text{lim}}^{(j)}$ of the level j and compute the confidence matrix $\boldsymbol{C}^{(j)}$.

   (b) Apply the FED algorithm (cf. Fig. 3.11) with the image data $\tilde{f}^{(j)}$, cycle length n_j, M_j cycles and step size limit $\tau_{\text{lim}}^{(j)}$. Note that the divergence matrix is multiplied with $\boldsymbol{I} - \boldsymbol{C}^{(j)}$.

   (c) If $j > 0$, interpolate the FED solution to the next finer grid $j-1$ and adapt it to the map $\tilde{c}^{(j-1)}$ and the data $\tilde{f}^{(j-1)}$. Go back to (a) with $j-1$, if $j > 0$.

   (d) If $j = 0$, the original level is reached.

Figure 3.12: CFED algorithm for inpainting.

The most difficult part is the embedding of FED in the coarse-to-fine framework, in particular the implementation of the restriction and prolongation operators. In our case, we use the ones presented in [18] (see page 4).

3.4 Fast-Jacobi Solver

In Sec. 3.1 we have seen that the ideas for explicit schemes with varying time step sizes originally come from linear system solvers using varying relaxation parameters. Thus, it is natural to transfer the FED method from the explicit diffusion framework to the solution of linear systems, i.e. we introduce the so-called *Fast-Jacobi (FJ) method* [165].

3.4.1 Basic Idea

We consider a linear system

$$\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{c}\,, \tag{3.182}$$

with an invertible matrix $\boldsymbol{B} \in \mathbb{C}^{N\times N}$ and $\boldsymbol{c}, \boldsymbol{x} \in \mathbb{C}^N$, where $\boldsymbol{c}$ is the known right hand side. We assume that $\boldsymbol{B}$ has non-zero diagonal entries and require that the eigenvalues' moduli of the matrix

$$\boldsymbol{M} \;:=\; \boldsymbol{D}_{\boldsymbol{B}}^{-1}\,(\boldsymbol{D}_{\boldsymbol{B}} - \boldsymbol{B}) \;=\; \boldsymbol{I} \;-\; \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B} \tag{3.183}$$

are smaller than 1, where the diagonal matrix $\boldsymbol{D}_{\boldsymbol{B}} \in \mathbb{C}^{N\times N}$ represents the diagonal part of $\boldsymbol{B}$. Under these assumptions, the well-known Jacobi method

$$\boldsymbol{x}^{k+1} \;=\; \boldsymbol{M}\boldsymbol{x}^k \;+\; \boldsymbol{D}_{\boldsymbol{B}}^{-1}\,\boldsymbol{c} \;=\; \boldsymbol{x}^k \;+\; \boldsymbol{D}_{\boldsymbol{B}}^{-1}\left(\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}^k\right) \quad (k \geq 0)\,, \tag{3.184}$$

converges for each initial vector $\boldsymbol{x}^0$ towards the unique solution of (3.182): $\boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{c}$ [98, 115]. It is also possible to incorporate a relaxation parameter $\omega > 0$, and we obtain the Jacobi over-relaxation (JOR) method [115]

$$\begin{aligned}\boldsymbol{x}^{k+1} \;&=\; \boldsymbol{x}^k \;+\; \omega\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\left(\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}^k\right)\\ &=\; \underbrace{\left(\boldsymbol{I} - \omega\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)}_{:=\boldsymbol{M}(\omega)}\boldsymbol{x}^k \;+\; \omega\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\,\boldsymbol{c}\,. \end{aligned}\tag{3.185}$$

If all eigenvalues $-1 < \mu_1 \leq \dots \leq \mu_N < 1$ of the Jacobi iteration matrix $\boldsymbol{M}$ are real-valued, then one can show that the optimal relaxation parameter of the JOR method is given by [115]

$$\omega_{opt} \;=\; \frac{2}{2 - \mu_1 - \mu_N}\,. \tag{3.186}$$

Instead of using a constant (optimal) relaxation parameter $\omega \in \mathbb{R}^+$, we want to use varying parameters that are based on the time step sizes of the FED scheme. Since the eigenvalues of $\boldsymbol{M}$ lie in $[\mu_1\,,\,\mu_N]$, the matrix

$$\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B} \;=\; \boldsymbol{I} \;-\; \boldsymbol{M} \tag{3.187}$$

has eigenvalues in $[1-\mu_N\,,\,1-\mu_1]$. The relaxation parameter has to be chosen such that the eigenvalues' moduli of $\boldsymbol{M}(\omega)$ are smaller than 1. Thus, the JOR method converges for

$$\omega \;<\; \frac{2}{1-\mu_1}\,, \tag{3.188}$$

where $\frac{2}{1-\mu_1}$ can be seen as a stability limit. We define a constant ω_{lim} via

$$\frac{2}{2\,-\,\mu_1\,-\,\mu_N} \;\leq\; \omega_{\mathrm{lim}} \;<\; \frac{2}{1-\mu_1}\,, \tag{3.189}$$

and according to the time step sizes of an FED cycle with length n, we consider the relaxation parameters

$$\omega_i \;=\; \omega_{\mathrm{lim}}\cdot\frac{1}{2\,\cos^2\left(\pi\cdot\frac{2i+1}{4n+2}\right)} \qquad (i=0,\dots,n{-}1)\,. \tag{3.190}$$

The resulting *Fast-Jacobi* cycle is then given by the explicit iterations

$$\boldsymbol{x}^{i+1} \;=\; \left(\boldsymbol{I}\,-\,\omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)\boldsymbol{x}^i \;+\; \omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\,\boldsymbol{c} \qquad (i=0,\dots,n{-}1)\,, \tag{3.191}$$

with a given initial vector $\boldsymbol{x}^0$.

3.4.2 Convergence

To show the convergence, we consider the exact solution $\boldsymbol{x}$ of Eq. (3.182) and the fact that

$$\begin{aligned}\boldsymbol{x}\,-\,\boldsymbol{x}^{i+1} \;&=\; \boldsymbol{x}\,-\,\left(\boldsymbol{I}\,-\,\omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)\boldsymbol{x}^i\,-\,\omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\underbrace{\boldsymbol{c}}_{=\boldsymbol{B}\boldsymbol{x}}\\ &=\; \left(\boldsymbol{I}\,-\,\omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)\left(\boldsymbol{x}\,-\,\boldsymbol{x}^i\right)\;.\end{aligned} \tag{3.192}$$

Thus, the error

$$\boldsymbol{e}^i \;:=\; \boldsymbol{x}\,-\,\boldsymbol{x}^i \tag{3.193}$$

after one Fast-Jacobi cycle with n iterations fulfils

$$\boldsymbol{e}^n \;=\; \prod_{i=0}^{n-1}\left(\boldsymbol{I}\,-\,\omega_i\cdot\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)\boldsymbol{e}^0 \tag{3.194}$$

with the initial error $\boldsymbol{e}^0$. We know that the eigenvalues of $\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}$ are real-valued and larger than $1-\mu_N>0$. Since the relaxation parameters are related to roots of Chebyshev polynomials, we can represent the matrix product as the matrix polynomial

$$\prod_{i=0}^{n-1} \left(\boldsymbol{I} \,-\, \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right) \;=\; \frac{1}{2n+1} \cdot U_{2n}\left(\sqrt{\boldsymbol{I} \,-\, \tfrac{\omega_{\lim}}{2} \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}}\,\right) \;. \tag{3.195}$$

The eigenvalues of this polynomial lie in the set

$$\left\{ \frac{1}{2n+1} \cdot U_{2n}\left(\sqrt{1 \,-\, \tfrac{\omega_{\lim}}{2} \cdot z}\right) \;\middle|\; z \in [1-\mu_N \,,\, 1-\mu_1] \right\} \,, \tag{3.196}$$

and because of

$$\sqrt{1 - \frac{\omega_{\lim}}{2} \cdot z} \;\neq\; 0 \,, \tag{3.197}$$

we have

$$\left| \frac{1}{2n+1} \cdot U_{2n}\left(\sqrt{1 \,-\, \tfrac{\omega_{\lim}}{2} \cdot z}\right) \right| \;<\; 1 \tag{3.198}$$

for $z \in [1-\mu_N \,,\, 1-\mu_1]$. Hence, the absolute values of the eigenvalues are smaller than 1, i.e.

$$\rho\left(\prod_{i=0}^{n-1} \left(\boldsymbol{I} \,-\, \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right) \right) \;<\; 1 \,, \tag{3.199}$$

where $\rho(\cdot)$ denotes the largest modulus of the eigenvalues (spectral radius). For the convergence we need the following well-known theorem from linear algebra. A proof of this theorem can be found for instance in [98].

Theorem 3.7 (Existence of a Norm). *Let $\boldsymbol{F} \in \mathbb{C}^{N\times N}$ be an arbitrary matrix and $\varepsilon > 0$. Then there exists a norm $\|\cdot\|$ that satisfies*

$$\|\boldsymbol{F}\| \;\leq\; \rho(\boldsymbol{F}) \,+\, \varepsilon \,. \tag{3.200}$$

The inequality (3.199) is also valid, if one adds a sufficiently small $\varepsilon > 0$ to the left hand side. Using Theorem 3.7, there exists a norm fulfilling

$$\left\| \prod_{i=0}^{n-1} \left(\boldsymbol{I} \,-\, \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right) \right\| \;<\; 1 \,, \tag{3.201}$$

which yields $\|\boldsymbol{e}^n\| < \|\boldsymbol{e}^0\|$. Multiple cycles further decrease the norm of the error, i.e., it tends to zero. More precisely the error $\boldsymbol{e}^{m,n} := \boldsymbol{x}^{m,n} - \boldsymbol{x}$, where $\boldsymbol{x}^{m,n}$ denotes the result after m cycles, can be estimated by

$$\begin{aligned}\|\boldsymbol{e}^{m,n}\| &= \left\| \left(\prod_{i=0}^{n-1} \left(\boldsymbol{I} - \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1} \boldsymbol{B} \right) \right)^m \boldsymbol{e}^0 \right\| \\ &\leq \underbrace{\left\| \prod_{i=0}^{n-1} \left(\boldsymbol{I} - \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1} \boldsymbol{B} \right) \right\|^m}_{\overset{m\to\infty}{\longrightarrow} 0} \cdot \left\| \boldsymbol{e}^0 \right\| . \end{aligned} \tag{3.202}$$

This shows $\boldsymbol{e}^{m,n} \to \boldsymbol{0}$ for $m \to \infty$, where $\boldsymbol{0}$ is the zero vector, and the series of the vectors $\boldsymbol{x}^{m,n}$, $m \geq 1$, converges to the unique solution $\boldsymbol{x}$ of (3.182). Thus, we can state the following theorem:

Theorem 3.8 (**Convergence of Fast-Jacobi**). *Let $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$ be a linear system with an invertible matrix $\boldsymbol{B} \in \mathbb{C}^{N\times N}$ such that the eigenvalues $\mu_1, \dots, \mu_N$ of $\boldsymbol{I} - \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}$ are real-valued with moduli smaller than 1. Furthermore, we define the relaxation parameters*

$$\omega_i = \omega_{\lim} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \dots, n-1), \tag{3.203}$$

with the cycle length n and the constant $\omega_{\lim} < \frac{2}{1-\mu_1}$. Then the result $\boldsymbol{x}^{m,n}$ of the Fast-Jacobi method using $m \geq 1$ cycles converges with $m \to \infty$ for any arbitrary initial vector $\boldsymbol{x}^0 \in \mathbb{C}^N$ to the unique solution $\boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{c}$ of the linear system.

Unfortunately, the norm that we use depends on ε and one has to compute a Schur decomposition for the description of this norm. In this context, Saad [115] considers a so-called *general convergence factor* that is simply equal to the spectral radius of the iteration matrix of an iterative method. This means we consider $\boldsymbol{M}$ for the usual Jacobi, $\boldsymbol{M}(\omega)$ for JOR, or the matrix product $\prod_i \left(\boldsymbol{I} - \omega_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B} \right)$ for Fast-Jacobi. In this context, we want to estimate the (relatively abstract) global convergence factor of Fast-Jacobi,

$$\max_{z\in[1-\mu_N,\, 1-\mu_1]} \left| \frac{1}{2n+1} \cdot U_{2n}\left(\sqrt{1 - \tfrac{\omega_{\lim}}{2} \cdot z} \right) \right| , \tag{3.204}$$

which would allow us to express it without the Chebyshev polynomial of the second kind. This can e.g. simplify the optimisation of the parameters n

and m. To this end, we replace the polynomial by the original representation with the Chebyshev polynomials of the first kind given in Eq. (2.97) :

$$(-1)^n \cdot \frac{1}{2n+1} \cdot \frac{T_{2n+1}\left(\sqrt{\frac{\omega_{\lim}}{2} \cdot z}\right)}{\sqrt{\frac{\omega_{\lim}}{2} \cdot z}} . \tag{3.205}$$

Since the value 1 is an absolute upper bound of $T_{2n+1}(\cdot)$ and $z \geq 1 - \mu_N$, the absolute value of this expression is bounded by

$$\frac{1}{2n+1} \cdot \frac{1}{\sqrt{\frac{\omega_{\lim}}{2} \cdot (1 - \mu_N)}} = \frac{1}{2n+1} \cdot \sqrt{\frac{2}{\omega_{\lim}(1-\mu_N)}} . \tag{3.206}$$

If $\omega_{\lim} \approx \frac{2}{1-\mu_1}$, then

$$\sqrt{\frac{2}{\omega_{\lim}(1-\mu_N)}} \approx \sqrt{\frac{1-\mu_1}{1-\mu_N}} . \tag{3.207}$$

Thus, the convergence factor depends on the square root of the ratio between the largest and the smallest eigenvalue of $\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}$. In the case of $\omega_{\lim} = \omega_{opt}$ we obtain

$$\sqrt{\frac{2}{\omega_{\lim}(1-\mu_N)}} = \sqrt{1 + \frac{1-\mu_1}{1-\mu_N}} > \sqrt{\frac{1-\mu_1}{1-\mu_N}} , \tag{3.208}$$

and therefore this choice could degrade the convergence. To this end, we recommend to use a larger parameter $\omega_{\lim} > \omega_{opt}$.

We should mention that the estimated upper bound for the eigenvalues might be too coarse for smaller n. As an example, we consider the eigenvalues $\mu_1 = -{}^{99}/_{100}$ and $\mu_N = -\mu_1$. With $\omega_{\lim} = 1$ the upper bound is $\frac{\sqrt{200}}{2n+1}$ and this is larger than 1 for $n \leq 6$. However, for increasing n, the Chebyshev polynomial T_{2n+1} strongly oscillates between -1 and 1, which means that the estimation with the upper bound is more appropriate.

Let us now compare the convergence factors of the different Jacobi-like methods. For the usual Jacobi with the iteration matrix $\boldsymbol{M}$ we have the spectral radius $\max\{|\mu_1|, |\mu_N|\}$, and in the case of JOR the spectral radius of $\boldsymbol{M}(\omega)$ is equal to $\max_i |1-\omega+\omega \cdot \mu_i|$. If one uses the optimal relaxation parameter ω_{opt} in Eq. (3.186), its spectral radius is given by the absolute value $|1-\omega_{opt}+\omega_{opt} \cdot \mu_1|$, i.e. a JOR cycle with length n yields

$$|1-\omega_{opt}+\omega_{opt} \cdot \mu_1|^n = \left|\frac{\mu_1 - \mu_N}{2 - \mu_1 - \mu_N}\right|^n . \tag{3.209}$$

At first sight, the exponential term of the JOR method seems to be better than the upper bound of the Fast-Jacobi cycle that decreases only linearly in the cycle length n. However, instead of increasing the cycle length n, it is recommendable to use multiple Fast-Jacobi cycles, because this massively improves the convergence. More precisely, the error can also decrease exponentially, but only in the number of cycles. Let us illustrate this by means of the above example.

Example

We assume again $\mu_1 = -\,{}^{99}\!/_{100}$, $\mu_N = -\mu_1$ and $\omega_{\lim} = 1$. In order to compare the corresponding estimated convergence factors, we use an overall number of 200 JOR or Fast-Jacobi iterations. Regarding the JOR method with ω_{opt}, we have

$$\left|\frac{\mu_1 - \mu_N}{2 - \mu_1 - \mu_N}\right|^{200} = \left(\frac{99}{100}\right)^{200} \approx 0.134\,. \tag{3.210}$$

Note that the optimal parameter is $\omega_{opt} = 1$, i.e. it is actually a usual Jacobi solver. The convergence of Fast-Jacobi depends on both the length n of a cycle and the number m of cycles. As mentioned above, the estimated factor is given by $\left(\frac{\sqrt{200}}{2n+1}\right)^m$. Some values for different n and m are shown in Table 3.2. The best convergence is reached with $m = 10$ Fast-Jacobi cycles of length $n = 20$. For this setting the convergence factor is almost four orders of magnitude better than in the case of JOR. To reach such a result with the Jacobi method, one needs about 1060 iterations. This means that the effort is more than five times larger.

However, for problems with a more well-posed distribution of the eigenvalues like e.g. $\mu_1 = {}^1\!/_5$ and $\mu_N = {}^4\!/_5$, the convergence factor of JOR with the optimal parameter $\omega_{opt} = 2$ and 20 iterations is about one order of magnitude smaller than the Fast-Jacobi method with its best configuration. Hence, the Fast-Jacobi method seems to be well-suited for problems with $\mu_N \approx 1$, as shown in the above example.

If we reconsider Table 3.2, we can see that there is an optimal parameter setting (n, m) with respect to the convergence factor. In order to find this optimal setting, we have to determine

$$\min_{m \in J}\left(\frac{1}{2 \cdot \frac{k}{m} + 1} \cdot \sqrt{\frac{2}{\omega_{\lim}\,(1 - \mu_N)}}\right)^m , \tag{3.211}$$

Table 3.2: Estimated convergence factors of Fast-Jacobi with different cycle lengths n and number of cycles m such that $n \cdot m = 200$.

n	m	factor
8	25	0.0100381
10	20	0.0003681
20	10	0.0000238
25	8	0.0000350
40	5	0.0001622
50	4	0.0003844
100	2	0.0049504
200	1	0.0352672

with the set $J := \left\{m \in \mathbb{N} \mid \frac{k}{m} \in \mathbb{N}\right\}$ and the overall number of iterations k. To this end, we define the real-valued, continuous function $f : \mathbb{R}^+ \to \mathbb{R}^+$,

$$f(x) \;:=\; \left(\frac{C}{2 \cdot \frac{k}{x} + 1}\right)^x , \tag{3.212}$$

with $C := \sqrt{\frac{2}{\omega_{\lim}(1-\mu_N)}} > 1$. We have the first derivative

$$f'(x) \;=\; \underbrace{\left(\frac{C}{2 \cdot \frac{k}{x} + 1}\right)^x}_{= f(x)} \cdot \left(\ln\left(\frac{Cx}{2k+x}\right) + \frac{2k}{2k+x}\right) , \tag{3.213}$$

and thus the necessary condition for a local extremum yields the fixed point equation

$$x \;=\; \underbrace{\frac{2k+x}{C} \cdot \exp\left(-\frac{2k}{2k+x}\right)}_{:= g(x)} . \tag{3.214}$$

Because of

$$g'(x) \;=\; \frac{1}{C} \cdot \left(1 + \frac{2k}{2k+x}\right) \cdot \exp\left(-\frac{2k}{2k+x}\right) \;\leq\; \frac{1}{C} \;<\; 1 \tag{3.215}$$

for $x \geq 0$, $g : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is a contraction mapping with the Lipschitz constant $\frac{1}{C} < 1$. Hence, according to the Banach fixed point theorem [159], there exists a unique solution x^* of the fixed point equation. It can be approximated by the fixed point iteration $x_{i+1} = g(x_i)$, with e.g. $x_0 = 1$. Note that $x^* > 0$, since $g(x) > 0$ for all $x \geq 0$. The second derivative of the function f is given by

$$f''(x) = f(x) \cdot \left(\left(\ln\left(\frac{Cx}{2k+x} \right) + \frac{2k}{2k+x} \right)^2 + \frac{4k^2}{x(2k+x)^2} \right) , \tag{3.216}$$

and fulfils $f''(x) > 0$ for $x > 0$, which is the sufficient condition for local minima. Thus, $f(x^*)$ is the unique local minimum, and the two candidates of the set J are $m_- \leq x^*$ and $m_+ > x^*$ with minimum distance to x^*. The decision between m_- and m_+ can be made based on the evaluation of the estimated factors for both m_- and m_+, or the cycle length. If one prefers, for example, a smaller cycle length because of stability issues, then m_+ would be the right choice. Unfortunately, x^* can not be evaluated directly, because a closed-form expression would require the *Lambert W function* [33]. However, we can give an approximation by rewriting the fixed point equation. At first, we replace $\frac{x}{2k}$ by y, which yields

$$y = \frac{1}{C} \cdot (1+y) \cdot \exp\left(-\frac{1}{1+y} \right) . \tag{3.217}$$

Since the solution of this equation does not depend on the number k, the ratio $\frac{x}{2k}$ is constant, i.e. x^* is proportional to k. As we have mentioned above, the solution of this equation requires the Lambert W function. It inverts the function $x \cdot \exp(x)$. The ansatz

$$\tilde{y} = \frac{\exp(-1)}{C - \exp(-1)} \tag{3.218}$$

yields for the right hand side

$$\frac{1}{C} \cdot \frac{C}{C - \exp(-1)} \cdot \underbrace{\exp\left(\frac{\exp(-1) - C}{C} \right)}_{= -1 + \frac{\exp(-1)}{C}} \approx \frac{\exp(-1)}{C - \exp(-1)} , \tag{3.219}$$

and is thus an approximative solution whose approximation quality increases for large C. Overall, we get

$$x^* \approx 2k \cdot \frac{\exp(-1)}{C - \exp(-1)} . \tag{3.220}$$

The proportionality between x^* and k means that the length of a cycle actually does not depend on k. It can only change slightly because of the divisibility with respect to k.

To verify these theoretical results, we reconsider the above example with $\mu_1 = -{}^{99}/_{100}$, $\mu_N = -\mu_1$, $\omega_{\text{lim}} = 1$ and $k = 200$. In this case, we obtain $C = \sqrt{200}$, and therefore

$$x^* \;\approx\; 400 \cdot \frac{\exp(-1)}{\sqrt{200} \,-\, \exp(-1)} \;\approx\; 10.68\,. \tag{3.221}$$

This confirmes the result in Table 3.2, where the optimal number of cycles is $m = 10$.

3.4.3 Implementation

After the theoretical analysis of the Fast-Jacobi method, we give a summary of the algorithm. It is shown in Fig. 3.13. The design is very similar to FED. One has to compute the relaxation parameters and find a stable sequence. Fortunately, the stable rearrangements of the time step sequences for FED work also well with the relaxation parameters. If we know or have a good estimation for the eigenvalue μ_N, then Eq. (3.220) yields a good parameter setting for the cycle length n and the number m of Fast-Jacobi cycles.

To improve the convergence, we can combine the Fast-Jacobi method with a coarse-to-fine strategy. This means, we approximate the solution of a linear system on a coarse scale and can use the prolongated result as an initialisation for the next finer level.

3.4.4 Fast-Jacobi for Parabolic Problems

We reconsider the ODE system

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{P}\boldsymbol{u}\,, \tag{3.222}$$

with a negative semi-definite system matrix $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ and initial data $\boldsymbol{u}^0 = \boldsymbol{u}(0)$. Instead of solving it with an explicit scheme, we use an implicit scheme

$$(\boldsymbol{I} \,-\, \tau\,\boldsymbol{P})\,\boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k \qquad (k \geq 0). \tag{3.223}$$

It requires the solution of a linear system in each time step. Note that such systems can also appear in the context of variational regularisation methods [119]. In contrast to a usual explicit scheme, there exists no limit for the time step size $\tau > 0$, because all eigenvalues of the symmetric system matrix are larger than or equal to 1. Thus, it is an invertible matrix for all $\tau > 0$ and the eigenvalues of the inverse matrix are smaller than or equal

1. **Input Data**:
   linear system of equations $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$, Fast-Jacobi cycle length n, number m of Fast-Jacobi cycles, feasible $\omega_{\text{lim}} > 0$

2. **Initialisation**:

   (a) Compute the relaxation parameters ω_i from Eq. (3.190):

   $$\omega_i \;=\; \omega_{\text{lim}} \cdot \frac{1}{2\,\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \ldots, n-1).$$

   (b) Choose a suitable ordering for the relaxation parameters: $\{\tilde{\omega}_0, \ldots, \tilde{\omega}_{n-1}\}$.

   (c) Define an initial vector $\boldsymbol{x}^0$.

3. **Outer Loop**: $(k = 1, \ldots, m)$

   (a) $\boldsymbol{x}^{k,\,0} \;:=\; \boldsymbol{x}^{k-1}$.

   (b) Perform one Fast-Jacobi cycle with the above ordering of the n relaxation parameters $(i = 0, \ldots, n-1)$:

   $$\boldsymbol{x}^{k,\,i+1} \;=\; \left(\boldsymbol{I} \,-\, \tilde{\omega}_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\right)\boldsymbol{x}^{k,\,i} \;+\; \tilde{\omega}_i \cdot \boldsymbol{D}_{\boldsymbol{B}}^{-1}\,\boldsymbol{c}\,.$$

   (c) $\boldsymbol{x}^{k} \;:=\; \boldsymbol{x}^{k,\,n}$.

Figure 3.13: Fast-Jacobi algorithm.

to 1, which implies stability in the Euclidean norm. More details can be found for example in [147].

Concerning isotropic diffusion processes, it has already been shown that the above system matrix is strictly diagonally dominant [147], i.e.

$$1 \,-\, \tau \cdot p_{j,j} \;>\; \tau \cdot \sum_{\substack{k=1\\k\neq j}}^{N} |p_{j,k}| \;\geq 0 \qquad (j = 1, \ldots, N)\,. \tag{3.224}$$

Thus, according to e.g. [98, 115], JOR and in particular Fast-Jacobi can be applied to solve the linear system (3.223). Let $\boldsymbol{D}_\tau \in \mathbb{R}^{N\times N}$ be the diagonal matrix representing the diagonal entries of $\boldsymbol{I} - \tau\,\boldsymbol{P}$. A Fast-Jacobi iteration step with suitable relaxation parameters ω_i from Eq. (3.190) is then given by

$$\begin{aligned} \boldsymbol{x}^{i+1} &= \boldsymbol{x}^i + \omega_i \cdot \boldsymbol{D}_\tau^{-1} \left(\boldsymbol{u}^{\boldsymbol{k}} - (\boldsymbol{I} - \tau \boldsymbol{P}) \, \boldsymbol{x}^i \right) \\ &= \left(\boldsymbol{I} + \omega_i \cdot \tau \, \boldsymbol{D}_\tau^{-1} \boldsymbol{P} \right) \boldsymbol{x}^i + \omega_i \cdot \boldsymbol{D}_\tau^{-1} \left(\boldsymbol{u}^k - \boldsymbol{x}^i \right) . \end{aligned} \tag{3.225}$$

To find optimal cycle parameters, we are going to analyse the eigenvalues of the matrix

$$\boldsymbol{G}_\tau := \boldsymbol{D}_\tau^{-1} \left(\boldsymbol{I} - \tau \boldsymbol{P} \right) . \tag{3.226}$$

The Gershgorin cycle with respect to the j-th diagonal entry of $\boldsymbol{G}_\tau$ reads as

$$\left\{ z \in \mathbb{C} \,:\, |z - 1| \leq \sum_{\substack{k=1 \\ k \neq j}}^{N} \frac{\tau \cdot |p_{j,k}|}{1 - \tau \cdot p_{j,j}} \right\} . \tag{3.227}$$

In the isotropic case, one has $p_{j,k} \geq 0$ for $j \neq k$ and

$$\sum_{k=1}^{N} p_{j,k} = 0 \tag{3.228}$$

for all j. Since $\boldsymbol{G}_\tau$ is similar to a symmetric matrix, i.e.

$$\boldsymbol{G}_\tau = \boldsymbol{D}_\tau^{-\frac{1}{2}} \left(\boldsymbol{D}_\tau^{-\frac{1}{2}} \left(\boldsymbol{I} - \tau \boldsymbol{P} \right) \boldsymbol{D}_\tau^{-\frac{1}{2}} \right) \boldsymbol{D}_\tau^{\frac{1}{2}} , \tag{3.229}$$

its eigenvalues are real-valued and range in

$$\left[1 - \max_j \frac{\tau \cdot |p_{j,j}|}{1 + \tau \cdot |p_{j,j}|} \,,\, 1 + \max_j \frac{\tau \cdot |p_{j,j}|}{1 + \tau \cdot |p_{j,j}|} \right] \subset (0, 2) . \tag{3.230}$$

Thus, the eigenvalues of $\boldsymbol{M}_\tau := \boldsymbol{I} - \boldsymbol{G}_\tau$ are included in

$$\left[- \max_j \frac{\tau \cdot |p_{j,j}|}{1 + \tau \cdot |p_{j,j}|} \,,\, \max_j \frac{\tau \cdot |p_{j,j}|}{1 + \tau \cdot |p_{j,j}|} \right] \subset (-1, 1) . \tag{3.231}$$

Because of the monotonicity, we can rewrite the fraction by means of

$$\max_j \frac{\tau \cdot |p_{j,j}|}{1 + \tau \cdot |p_{j,j}|} = \frac{\tau \cdot p_{\max}}{1 + \tau \cdot p_{\max}} , \tag{3.232}$$

where we have used the notation $p_{\max} := \max\limits_j |p_{j,j}|$. For Fast-Jacobi, we have to consider that

$$\omega_{\lim} < \frac{2}{1 + \frac{\tau \cdot p_{\max}}{1 + \tau \cdot p_{\max}}} = \frac{2 + 2\tau \cdot p_{\max}}{1 + 2\tau \cdot p_{\max}} < 2 . \tag{3.233}$$

If this condition is satisfied, the Fast-Jacobi method converges according to Theorem 3.8. Note that the choice $\omega_{\text{lim}} = 1$ works for all time step sizes $\tau > 0$. To find the optimal number of cycles, we need the largest eigenvalue of $\boldsymbol{M}_\tau$ and use the approximation in Eq. (3.220).

In the case of a nonlinear problem, i.e. $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{u})$, one uses the semi-implicit scheme

$$\big(\boldsymbol{I} - \tau\,\boldsymbol{P}(\boldsymbol{u}^k)\big)\,\boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^k \qquad (k \geq 0)\,. \tag{3.234}$$

The application of the Fast-Jacobi method is as straightforward as for the linear implicit scheme. However, $p_{\max}$ and thus the optimal number of cycles might vary. Therefore, the application of the method can be very difficult in practice.

If we additionally consider an anisotropic diffusion process, the condition $p_{i,j} \geq 0$ for $i \neq j$ can not be guaranteed [147]. Since $\boldsymbol{P}$ is still symmetric and negative semi-definite, the eigenvalues of the corresponding matrices $\boldsymbol{G}_\tau$ as well as $\boldsymbol{M}_\tau$ are real-valued. Unfortunately, the interval in Eq. (3.231) is not valid for the anisotropic case. However, using Gershgorin's theorem, we can estimate the eigenvalues and adjust the parameters of Fast-Jacobi according to this estimation. In general, it might happen that the estimation with Gershgorin circles yields eigenvalues of $\boldsymbol{M}_\tau$ with moduli larger than or equal to 1. Since $\boldsymbol{D}_\tau^{-1}$ is positive definite with eigenvalues that are bounded by 1, the matrix product $\boldsymbol{G}_\tau$ has positive eigenvalues which can not exceed the maximum eigenvalue of $\boldsymbol{I} - \tau\boldsymbol{P}$ [104]. Thus, $\boldsymbol{M}_\tau$ has eigenvalues smaller than 1, i.e. $\mu_N < 1$, and it can only happen that the smallest eigenvalue μ_1 is smaller than or equal to -1. In this case, ω_{lim} is, in contrast to the isotropic process, smaller than 1. This means that the number of iteration steps with under-relaxation $\omega_i < 1$ increases. However, Eq. (3.198) and the upper bound in Eq. (3.206) are still valid. Hence, Fast-Jacobi converges and one can use the error estimation in Theorem 3.8.

This consideration shows that the condition $\mu_1 > -1$ is not necessary for the convergence, since ω_{lim} is adapted to μ_1 such that the convergence is guaranteed. However, a small ω_{lim} yields more under-relaxation steps and hence a slower convergence.

3.4.5 Fast-Jacobi for Inpainting Problems

In Sec. 3.3 we have presented the cascadic FED (CFED) method in order to solve elliptic inpainting problems. It is based on a parabolic evolution. However, besides this parabolic ansatz, we could also solve the problem directly. To this end, we reconsider Eq. (3.167) and its spatial discretisation

$$\boldsymbol{C}\,(\boldsymbol{v}-\boldsymbol{f}) \;-\; (\boldsymbol{I}\;-\;\boldsymbol{C})\,\boldsymbol{P}(\boldsymbol{v})\,\boldsymbol{v} \;=\; \boldsymbol{0}\,, \tag{3.235}$$

where we use the notation from Sec. 3.3. We rewrite this equation and obtain the nonlinear equation system

$$\Big(\boldsymbol{C} \;-\; (\boldsymbol{I}\;-\;\boldsymbol{C})\boldsymbol{P}(\boldsymbol{v})\Big)\boldsymbol{v} \;=\; \boldsymbol{C}\boldsymbol{f}\;. \tag{3.236}$$

The corresponding system matrix has only positive diagonal entries. In order to have a linear system, we linearise the problem via

$$\Big(\boldsymbol{C} \;-\; (\boldsymbol{I}\;-\;\boldsymbol{C})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\Big)\boldsymbol{v}^{k+1} \;=\; \boldsymbol{C}\boldsymbol{f}\;, \tag{3.237}$$

with $k \geq 0$ and the initial vector $\boldsymbol{v}^0 := \boldsymbol{C}\boldsymbol{f}$. To show that JOR or Fast-Jacobi can be applied, we are going to prove the following theorem about the system matrix:

Theorem 3.9 (Eigenvalues of $\boldsymbol{C} - (\boldsymbol{I} - \boldsymbol{C})\boldsymbol{Q}$). *Let $\boldsymbol{Q} \in \mathbb{R}^{N\times N}$ be a symmetric, negative semi-definite matrix with rank $N-1$ and zero row sum. If $\boldsymbol{C} \in \{0,1\}^{N\times N}$ is a diagonal matrix with at least one non-zero entry, then the eigenvalues of the matrix $\boldsymbol{C} - (\boldsymbol{I}-\boldsymbol{C})\boldsymbol{Q}$ are real-valued and positive.*

Proof. The case $\boldsymbol{C} = \boldsymbol{I}$ is trivial, and hence we assume $\boldsymbol{C} \neq \boldsymbol{I}$. We define two linear subspaces of $\mathbb{C}^N$:

$$\mathcal{C} := \left\{\boldsymbol{C}\boldsymbol{y} \mid \boldsymbol{y} \in \mathbb{C}^N\right\} \;\neq\; \emptyset \tag{3.238}$$

and its orthogonal complement regarding the standard inner product of $\mathbb{C}^N$:

$$\mathcal{C}^\perp := \left\{(\boldsymbol{I}\;-\;\boldsymbol{C})\boldsymbol{y} \mid \boldsymbol{y} \in \mathbb{C}^N\right\} \;\neq\; \emptyset\;. \tag{3.239}$$

Thus, each vector $\boldsymbol{z} \in \mathbb{C}^N$ can be written as $\boldsymbol{z} = \boldsymbol{v} + \boldsymbol{w}$ with unique $\boldsymbol{v} \in \mathcal{C}$ and $\boldsymbol{w} \in \mathcal{C}^\perp$. Note that $\boldsymbol{v} \in \mathcal{C}$ implies $\boldsymbol{C}\boldsymbol{v} = \boldsymbol{v}$ and $\boldsymbol{w} \in \mathcal{C}^\perp$ implies $(\boldsymbol{I}-\boldsymbol{C})\boldsymbol{w} = \boldsymbol{w}$. Putting this into the eigenvector equation

$$\big(\boldsymbol{C} \;-\; (\boldsymbol{I}\;-\;\boldsymbol{C})\boldsymbol{Q}\big)\boldsymbol{z} \;=\; \lambda\cdot\boldsymbol{z} \tag{3.240}$$

yields:

$$\underbrace{\boldsymbol{v}}_{\in\,\mathcal{C}} \;-\; \underbrace{(\boldsymbol{I}\;-\;\boldsymbol{C})\boldsymbol{Q}(\boldsymbol{v}+\boldsymbol{w})}_{\in\,\mathcal{C}^\perp} \;=\; \underbrace{\lambda\cdot\boldsymbol{v}}_{\in\,\mathcal{C}} \;+\; \underbrace{\lambda\cdot\boldsymbol{w}}_{\in\,\mathcal{C}^\perp}\;. \tag{3.241}$$

If there exits an eigenvector $\boldsymbol{v} + \boldsymbol{w}$ with $\boldsymbol{v} \neq \boldsymbol{0}$, this equation immediately states that the eigenvalue has to satisfy $\lambda = 1$.

In the case of $\boldsymbol{v} = \boldsymbol{0}$ and $\boldsymbol{w} \neq \boldsymbol{0}$ we get

$$-(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{Q}\,\boldsymbol{w} \;=\; \lambda \cdot \boldsymbol{w}\,, \tag{3.242}$$

and because of $\boldsymbol{w} = (\boldsymbol{I} - \boldsymbol{C})\boldsymbol{w}$ it follows that

$$(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{w} \;=\; -\lambda \cdot \boldsymbol{w}\,. \tag{3.243}$$

The matrix $\tilde{\boldsymbol{Q}} := (\boldsymbol{I} - \boldsymbol{C})\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{C})$ is symmetric and since $\boldsymbol{Q}$ is negative semi-definite, we have for any $\boldsymbol{z} \in \mathbb{C}^N$

$$\boldsymbol{z}^*\,\tilde{\boldsymbol{Q}}\,\boldsymbol{z} \;=\; \tilde{\boldsymbol{z}}^*\,\boldsymbol{Q}\,\tilde{\boldsymbol{z}} \;\leq\; 0\,, \tag{3.244}$$

where $\tilde{\boldsymbol{z}} := (\boldsymbol{I} - \boldsymbol{C})\boldsymbol{z} \in \mathcal{C}^\perp$ and $\boldsymbol{z}^*$ denotes the conjugate transpose of $\boldsymbol{z}$. Thus, there exists a set of N orthonormal eigenvectors of $\tilde{\boldsymbol{Q}}$ with non-positive eigenvalues. Since $\boldsymbol{Q}$ has exactly one eigenvalue equal to zero (rank $N{-}1$) and the corresponding eigenvector $\boldsymbol{s} = (1, 1, \ldots, 1)^T$ (zero row sum) is not an element of $\mathcal{C}^\perp$, the above inequality is actually strict for $\tilde{\boldsymbol{z}} \in \mathcal{C}^\perp \setminus \{\boldsymbol{0}\}$, and Eq. (3.243) can only be satisfied for real-valued $\lambda > 0$. This concludes the proof. □

Regarding the linear case, it can be even shown that the unique solution $\boldsymbol{v}$ of Eq. (3.236) satisfies a maximum-minimum principle with respect to $\boldsymbol{f}^{\boldsymbol{C}} := \boldsymbol{C}\boldsymbol{f}$ [95]:

$$\min_{\substack{j=1,\ldots,N\\ c_{j,j}\neq 0}} f_j^{\boldsymbol{C}} \;\leq\; v_i \;\leq\; \max_{\substack{j=1,\ldots,N\\ c_{j,j}\neq 0}} f_j^{\boldsymbol{C}} \qquad (i = 1, \ldots, N)\,. \tag{3.245}$$

The iteration with JOR and a suitable relaxation parameter $\omega > 0$ follows the rule

$$\boldsymbol{x}^{m+1} \;=\; \Big(\boldsymbol{I} - \omega \cdot \boldsymbol{D}_k^{-1}\big(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\big)\Big)\boldsymbol{x}^m \;+\; \omega \cdot \boldsymbol{D}_k^{-1}\boldsymbol{C}\boldsymbol{f}\,, \tag{3.246}$$

where $m \geq 0$, $\boldsymbol{D}_k$ is the diagonal matrix with the diagonal entries of the system matrix $\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}(\boldsymbol{v}^k)$, and $\boldsymbol{x}^0$ an arbitrary initial vector. These entries are either 1 (if $c_{i,i} = 1$) or $-p_{i,i} > 0$ (if $c_{i,i} = 0$). Hence, we can compute $\boldsymbol{D}_k^{1/2}$ as well as $\boldsymbol{D}_k^{-1/2}$, and it holds that

$$\boldsymbol{C}\boldsymbol{D}_k^{-\frac{1}{2}} \;=\; \boldsymbol{D}_k^{-\frac{1}{2}}\boldsymbol{C} \;=\; \boldsymbol{C}\,. \tag{3.247}$$

Regarding the eigenvalues of $\boldsymbol{D}_k^{-1}\big(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\big)$ we obtain

$$\begin{aligned}
&\boldsymbol{D}_k^{-1}\big(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\big) \\
=\;& \boldsymbol{D}_k^{-\frac{1}{2}}\left(\boldsymbol{D}_k^{-\frac{1}{2}}\boldsymbol{C}\boldsymbol{D}_k^{-\frac{1}{2}}\boldsymbol{D}_k^{\frac{1}{2}} + \boldsymbol{D}_k^{-\frac{1}{2}}(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\boldsymbol{D}_k^{-\frac{1}{2}}\boldsymbol{D}_k^{\frac{1}{2}}\right) \\
=\;& \boldsymbol{D}_k^{-\frac{1}{2}}\left(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{D}_k^{-\frac{1}{2}}\boldsymbol{P}\left(\boldsymbol{v}^k\right)\boldsymbol{D}_k^{-\frac{1}{2}}\right)\boldsymbol{D}_k^{\frac{1}{2}}\,, \qquad (3.248)
\end{aligned}$$

which is a similarity transformation and means that they are equal to the eigenvalues of $\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{D}_k^{-1/2}\boldsymbol{P}\left(\boldsymbol{v}^k\right)\boldsymbol{D}_k^{-1/2}$. The matrix $\boldsymbol{Q} := \boldsymbol{D}_k^{-1/2}\boldsymbol{P}\left(\boldsymbol{v}^k\right)\boldsymbol{D}_k^{-1/2}$ is symmetric, negative semi-definite, and its rank corresponds to the one of $\boldsymbol{P}(\boldsymbol{v}^k)$ and is equal to N–1. Moreover, its eigenvector corresponding to the eigenvalue 0 is given by $\boldsymbol{D}_k^{1/2}\boldsymbol{s}$, where $\boldsymbol{s}$ has been defined in the proof of Theorem 3.9. Since the vector $\boldsymbol{D}_k^{1/2}\boldsymbol{s}$ is not contained in the set $\mathcal{C}^{\perp}$, the proof also works in this case. Thus, Theorem 3.9 implies positive eigenvalues for $\boldsymbol{D}_k^{-1}\big(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\big)$. As a consequence, the eigenvalues of the iteration matrix $\boldsymbol{I} - \omega \cdot \boldsymbol{D}_k^{-1}\big(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}(\boldsymbol{v}^k)\big)$ are bounded in absolute value by a constant $r < 1$, provided that a suitable $\omega > 0$ is used. Therefore, the convergence of both JOR and Fast-Jacobi to the unique solution is guaranteed.

Now we simplify the above JOR iteration in order to get a better understanding of it. Multiplying the linear system (3.237) with $\boldsymbol{C}$ and using $\boldsymbol{C}^2 = \boldsymbol{C}$ yields

$$\boldsymbol{C}\boldsymbol{v}^{k+1} \;=\; \boldsymbol{C}\boldsymbol{f}\,. \qquad (3.249)$$

To ensure this property for the approximate solution after a finite number of iterations, we use an initialisation $\boldsymbol{x}^0$ with $\boldsymbol{C}\boldsymbol{x}^0 = \boldsymbol{C}\boldsymbol{f}$. By multiplying the above calculation rule of JOR with $\boldsymbol{C}$, we get

$$\boldsymbol{C}\boldsymbol{x}^{m+1} \;=\; (1-\omega)\cdot\boldsymbol{C}\boldsymbol{x}^m \;+\; \omega\cdot\boldsymbol{C}\boldsymbol{f}\,, \qquad (3.250)$$

where we have used Eq. (3.247), but with $\boldsymbol{D}_k^{-1}$. Because of the relation $\boldsymbol{C}\boldsymbol{x}^0 = \boldsymbol{C}\boldsymbol{v}^k = \boldsymbol{C}\boldsymbol{f}$, it follows that $\boldsymbol{C}\boldsymbol{x}^1 = \boldsymbol{C}\boldsymbol{f}$ and $\boldsymbol{C}\boldsymbol{x}^m = \boldsymbol{C}\boldsymbol{f}$ for all $m \geq 2$. Thus, the above property is satisfied for each iteration. With

$$\boldsymbol{D}_k^{-1} \;=\; \boldsymbol{C} \;+\; (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{D}_{\boldsymbol{P}(\boldsymbol{v}^k)}^{-1}\,, \qquad (3.251)$$

where $\boldsymbol{D}_{\boldsymbol{P}(\boldsymbol{v}^k)} := \operatorname{diag}(\boldsymbol{P}(\boldsymbol{v}^k))$, we can state that

$$\begin{aligned}
\boldsymbol{D}_k^{-1}(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right) &= \left(\boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}\right)(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right) \\
&= (\boldsymbol{C} - \boldsymbol{I})\boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right) \\
&= \boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{C} - \boldsymbol{I})^2\boldsymbol{P}\left(\boldsymbol{v}^k\right) \\
&= -\boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right) \\
&= \boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}\left(\boldsymbol{v}^k\right) . \qquad (3.252)
\end{aligned}$$

Thus, we can simplify the JOR method by means of

$$\boldsymbol{x}^{m+1} = \left(\boldsymbol{I} - \omega \cdot \boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\right)\boldsymbol{x}^m , \qquad (3.253)$$

and one Fast-Jacobi cycle with length n reads

$$\boldsymbol{x}^n = \prod_{i=0}^{n-1}\left(\boldsymbol{I} - \omega_i \cdot \boldsymbol{D}^{-1}_{\boldsymbol{P}(\boldsymbol{v}^k)}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\right)\boldsymbol{x}^0 . \qquad (3.254)$$

After $m > 1$ cycles, we have the result $\boldsymbol{x}^{m,n}$ that is an approximation for the solution $\boldsymbol{v}^{k+1}$.

The iterations look very similar to the parabolic explicit scheme that is formulated in Eq. (3.174), and the main difference is the multiplication with an inverse diagonal matrix. It can be seen as a local adjustment of the relaxation parameters. Thus, the method decides, depending on the diagonal entry, whether it uses more over- or under-relaxation steps. On the other hand, the multiplication with $\boldsymbol{D}^{-1}_{\boldsymbol{P}(\cdot)}$ can also be interpreted as a preconditioning step that improves the convergence. In this context, we are going to analyse how Fast-Jacobi is related to Richardson's method.

To improve the convergence of the Fast-Jacobi solver for inpainting problems, it can be used within a coarse-to-fine strategy already presented in Sec. 3.3.1. We can reuse the existing explicit code and have to include only the diagonal preconditioning.

3.4.6 Relation to Richardson's Method

Let us the reconsider the linear system in Eq. (3.182) with an invertible matrix $\boldsymbol{B} \in \mathbb{C}^{N\times N}$ that has non-zero diagonal entries. The JOR method for this linear system corresponds to Richardson's method for the equivalent system

$$\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{D}_{\boldsymbol{B}}^{-1}\,\boldsymbol{c}\,, \tag{3.255}$$

where $\boldsymbol{D}_{\boldsymbol{B}}$ again denotes the diagonal part of $\boldsymbol{B}$. As we have mentioned in the last subsection, this matrix multiplication is some kind of a pre-conditioning step. If $\boldsymbol{B}$ is symmetric and positive definite, its condition number concerning the Euclidean norm corresponds to the ratio between the largest and the smallest eigenvalue. In this case, the matrix product $\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}$ has positive eigenvalues, but it is not necessarily symmetric, which means that we can not compute its condition number by the above ratio. Moreover, it depends on the matrix $\boldsymbol{B}$ how the diagonal preconditioning reduces the condition number. To this end, we consider three easy 1-D diffusion examples.

Diagonal Preconditioning

The condition number of system matrices associated to diffusion processes depends on the time step size τ and the process itself. Therefore, we try different step sizes as well as processes, and show the corresponding impact of the diagonal preconditioning in Table 3.3. Our examples for the diffusion processes are:

(a) Linear homogeneous diffusion:

We consider the system matrix $\boldsymbol{I} - \tau \cdot \boldsymbol{A}_{\boldsymbol{1}} \in \mathbb{R}^{5\times 5}$ $(h = 1)$, where $\boldsymbol{A}_{\boldsymbol{h}}$ is defined in Eq. (3.33).

(b) Inhomogeneous (spatially varying) diffusion (1):

Here, we analyse $\boldsymbol{I} - \tau \cdot \boldsymbol{P}_1 \in \mathbb{R}^{5\times 5}$ where $\boldsymbol{P}_1$ is given by

$$\boldsymbol{P}_1 \;=\; \begin{pmatrix} -\frac{3}{4} & \frac{3}{4} & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{5}{4} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{5}{8} & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{8} & -\frac{3}{8} & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} & -\frac{1}{4} \end{pmatrix}. \tag{3.256}$$

(c) Inhomogeneous (spatially varying) diffusion (2):

If we allow a diffusivity function that takes values larger than 1, this could imply a system matrix $\boldsymbol{I} - \tau \cdot \boldsymbol{P}_2 \in \mathbb{R}^{5\times 5}$ with

$$\boldsymbol{P}_2 = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & -3 & \frac{5}{2} & 0 & 0 \\ 0 & \frac{5}{2} & -\frac{55}{2} & 25 & 0 \\ 0 & 0 & 25 & -34 & 9 \\ 0 & 0 & 0 & 9 & -9 \end{pmatrix} . \tag{3.257}$$

These results allow the assumption that the diagonal preconditioning is better for nonlinear diffusion processes, which imply a larger range of variation regarding the diagonal entries. The corresponding improvement of the condition number is up to almost five times larger than for homogeneous diffusion. Furthermore, the best improvement is reached for the diffusion process using strongly varying coefficients. A well-known example for such a process is the so-called total variation (TV) diffusion [6, 7]. Its diffusivity is given by

$$g\left(|\boldsymbol{\nabla} u|^2\right) = \frac{1}{|\boldsymbol{\nabla} u|} . \tag{3.258}$$

It is unbounded, but in practice we use the regularised version

$$g_\varepsilon\left(|\boldsymbol{\nabla} u|^2\right) = \frac{1}{\sqrt{\varepsilon^2 + |\boldsymbol{\nabla} u|^2}} \tag{3.259}$$

with a small regularisation parameter $\varepsilon > 0$. Unfortunately, the limit time step size for an explicit diffusion scheme is proportional to ε, i.e. can be very small. Thus, the cycle times of FED are also much smaller than in the case of a diffusivity that is bounded by e.g. 1. For elliptic problems, where a steady-state solution has to be approximated with a huge stopping time, we would require many cycles.

In the experimental section, we will show that the solution by means of the Fast-Jacobi method performs much better than a parabolic FED scheme for elliptic problems with strongly varying coefficients. We also apply a *modified Richardson* method, which means that we include a diagonal preconditioning. More precisely, we consider the Fast-Jacobi algorithm, but we use the Richardson-based relaxation parameters from Eq. (3.17).

Besides the preconditioning step, there is another interesting difference between Richardson's method and Fast-Jacobi: The convergence properties if

Table 3.3: Impact of the diagonal preconditioning for the exemplary diffusion processes. The difference between the condition number κ_2 of the original, and κ_2^* of the preconditioned matrix is computed by the relative value $\frac{\kappa_2 - \kappa_2^*}{\kappa_2}$.

	$\boldsymbol{\tau}$	**5**	**10**	**50**	**100**	**500**
	$\boldsymbol{\kappa_2}$	19.09	37.18	181.90	362.80	1810.02
(a)	$\boldsymbol{\kappa_2^*}$	17.96	35.08	172.39	344.07	1717.59
	diff.	5.9 %	5.6 %	5.2 %	5.2 %	5.1 %
	$\boldsymbol{\kappa_2}$	10.62	20.23	97.17	193.34	962.72
(b)	$\boldsymbol{\kappa_2^*}$	9.32	17.09	77.71	153.79	764.46
	diff.	12.3 %	15.5 %	20.0 %	20.5 %	20.6 %
	$\boldsymbol{\kappa_2}$	285.88	570.76	2849.79	5698.58	28488.91
(c)	$\boldsymbol{\kappa_2^*}$	218.82	435.17	2174.34	4350.29	21760.16
	diff.	23.5 %	23.8 %	23.7 %	23.7 %	23.6 %

an eigenvalue of the system matrix $\boldsymbol{B}$ is equal to 0, which in fact means that the matrix is not invertible. Let us now reconsider the homogeneous system (3.7) with a system matrix $\boldsymbol{B}$ that has real-valued non-negative eigenvalues. In this case, the estimation in Eq. (3.20) for the convergence rate of Richardson's method with cycle length $k+1$ does not work anymore ($\lambda_{\min} = 0$) and the maximum in Eq. (3.14) is 1. If this maximum is not only reached for the eigenvalue 0, but also for another $\lambda_j > 0$, $S_{k+1}(\lambda_j) = \pm 1$, the convergence to a feasible solution of the homogeneous system can not be guaranteed. More precisely, let the initial vector $\boldsymbol{x}^0$ satisfy $|\boldsymbol{v}_j^T \boldsymbol{x}^0| =: r > 0$, where $\boldsymbol{v}_j$ is the eigenvector corresponding to λ_j. Having applied $m \geq 1$ cycles of length $k+1$ with Richardson's method, the result $\boldsymbol{x}^{m,\,k+1}$ fulfils

$$\left|\boldsymbol{v}_j^T \boldsymbol{x}^{m,\,k+1}\right| \;=\; |S_{k+1}(\lambda_j)|^m \cdot r \;=\; r\,. \qquad (3.260)$$

A feasible solution of the homogeneous system has to be a linear combination of eigenvectors corresponding to the eigenvalue 0. Moreover, it should be orthogonal to all other eigenvectors with positive eigenvalues, which means in particular orthogonal to $\boldsymbol{v}_j$. However, since the vector $\boldsymbol{x}^{m,\,k+1}$ can never satisfy this property, there is no converge to $m \to \infty$. Thus, to yield convergence, one should ensure that $\boldsymbol{v}_i^T \boldsymbol{x}^0 = 0$ for all eigenvectors $\boldsymbol{v}_i$ whose eigenvalues $\lambda_i > 0$ fulfil $S_{k+1}(\lambda_i) = \pm 1$.

Since the value 0 can not be excluded as a diagonal entry of $\boldsymbol{B}$, it might happen that the Fast-Jacobi method can not be applied. In this case, we could use for instance Richardson's method with suitable FED-based relaxation parameters. However, let us assume firstly that the Jacobi method works. Since the inequality (3.198) is valid for $z > 0$, the result of m Fast-Jacobi cycles with length n, i.e. $\boldsymbol{x}^{m,n}$, satisfies the estimation

$$\left|\boldsymbol{v}_j^T \boldsymbol{x}^{m,n}\right| \;<\; C_j^m \cdot \left|\boldsymbol{v}_j^T \boldsymbol{x}^0\right| \;, \tag{3.261}$$

where $\boldsymbol{v}_j$ is an arbitrary eigenvector corresponding to the eigenvalue $\lambda_j > 0$, and $C_j < 1$ a suitable positive constant that depends on the eigenvalue. Thus, $\left|\boldsymbol{v}_j^T \boldsymbol{x}^{m,n}\right| \to 0$ for $m \to \infty$, and the convergence to a feasible solution of the homogeneous system is guaranteed.

These theoretical results are important for the elliptic inpainting case. The Fast-Jacobi iteration in Eq. (3.254) can also be interpreted as an iteration for the homogeneous system

$$(\boldsymbol{C} - \boldsymbol{I})\,\boldsymbol{P}\left(\boldsymbol{v}^k\right)\,\boldsymbol{x} \;=\; \boldsymbol{0}\,. \tag{3.262}$$

This system matrix has real-valued non-negative eigenvalues. However, at least one of the diagonal entries has to be equal to zero. To this end, the Fast-Jacobi iteration in Eq. (3.254) implicitly proposes to use the inverse diagonal entries of $-\boldsymbol{P}(\boldsymbol{v}^k)$ instead of $(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{P}(\boldsymbol{v}^k)$.

If we would apply a cycle of length p with Richardson's method using the relaxation parameters (cf. Eq. (3.17) with $\lambda_{\min} = 0$)

$$\omega_i \;=\; \frac{2}{\lambda_{\max} \;-\; \lambda_{\max} \cdot \cos\left(\pi \cdot \frac{2i+1}{2p}\right)} \qquad (i = 0, \ldots, p-1)\,, \tag{3.263}$$

where $\lambda_{\max}$ is the largest eigenvalue of $(\boldsymbol{C} - \boldsymbol{I})\,\boldsymbol{P}\left(\boldsymbol{v}^k\right)$, i.e.

$$\boldsymbol{x}^{m,p} \;=\; \left(\prod_{i=0}^{p-1}\left(\boldsymbol{I} \;-\; \omega_i \cdot (\boldsymbol{C} \;-\; \boldsymbol{I})\boldsymbol{P}\left(\boldsymbol{v}^k\right)\right)\right)^m \boldsymbol{x}^0 \;, \tag{3.264}$$

then the convergence is not guaranteed, as we have shown above. To enforce convergence, one has to regularise the problem and to compute the relaxation parameters with $\lambda_{\min} > 0$, although the smallest eigenvalue is in fact equal to 0. This strategy is equivalent to the use of a damping parameter $\nu > 0$ in the parabolic case.

3.5 FED for Hyperbolic Problems

So far, we have considered the application of FED in the context of problems whose discretisations yield symmetric matrices. Now we want to analyse whether our fast explicit scheme can be also used to solve hyperbolic equations and whether it is able to improve the efficiency of standard numerical solvers. As an example, we consider the d-dimensional linear transport equation

$$\begin{aligned} \frac{\partial u(\boldsymbol{x},t)}{\partial t} \;+\; \boldsymbol{b}^T \boldsymbol{\nabla} u(\boldsymbol{x},t) \;&=\; 0 \quad \forall\,(\boldsymbol{x},t) \in \mathbb{R}^d \times (0,\infty)\;, \\ u(\boldsymbol{x},0) \;&=\; g(\boldsymbol{x}) \quad \forall\,\boldsymbol{x} \in \mathbb{R}^d\;, \end{aligned} \tag{3.265}$$

where $g : \mathbb{R}^d \to \mathbb{R}$ represents the given initial data, $\boldsymbol{b} \in \mathbb{R}^d$ describes the velocity and $u : \mathbb{R}^d \times [0,\infty) \to \mathbb{R}$ is the unknown solution. However, there exists a closed-form solution:

$$u(\boldsymbol{x},t) \;=\; g(\boldsymbol{x} \,-\, t \cdot \boldsymbol{b})\;. \tag{3.266}$$

More details about further theoretical results can be found for example in [91]. In the following, we consider the case $d = 1$ and assume a scalar-valued velocity $b > 0$.

We discretise Eq. (3.265) on a spatiotemporal grid with the spatial mesh size $h > 0$ and the time step size $\tau > 0$, $u_i^k \approx u(ih\,,\,k\tau)$. If we discretise the spatial derivative with the help of a backward difference and the time derivative using a forward difference, we get

$$\frac{u_i^{k+1} \,-\, u_i^k}{\tau} \;+\; b \cdot \frac{u_i^k \,-\, u_{i-1}^k}{h} \;=\; 0\;, \tag{3.267}$$

which can be written as an explicit scheme, a so-called *upwind scheme*:

$$u_i^{k+1} \;=\; u_i^k \;-\; b \cdot \frac{\tau}{h} \cdot \left(u_i^k \,-\, u_{i-1}^k\right)\;. \tag{3.268}$$

The constant $c := b \cdot \frac{\tau}{h}$ is the well-known *Courant number* [35]. One can show that the upwind scheme is stable for $c \in [0,1]$. Assuming a finite number N of spatial grid points and homogeneous Dirichlet boundary conditions, we can rewrite the scheme in terms of a matrix-vector multiplication,

$$\boldsymbol{u}^{k+1} \;=\; (\boldsymbol{I} \,+\, \tau\,\boldsymbol{B_h})\,\boldsymbol{u}^k\;, \tag{3.269}$$

with the unsymmetric matrix

$$\boldsymbol{B_h} \;=\; \frac{b}{h}\cdot\begin{pmatrix} -1 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & \dots & 0 \\ & \ddots & \ddots & & \vdots \\ 0 & \dots & 1 & -1 & 0 \\ 0 & \dots & 0 & 1 & -1 \end{pmatrix} \;\in\; \mathbb{R}^{N\times N}\,. \tag{3.270}$$

Although it is unsymmetric, the eigenvalues are real-valued and given by the diagonal entries $-\frac{b}{h}$. Since the stability condition for an usual explicit scheme is $\tau \leq \frac{h}{b}$, we might use an FED cycle

$$\boldsymbol{u}^n \;=\; \left(\prod_{i=0}^{n-1} (\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{B_h})\right)\boldsymbol{u}^0\;, \tag{3.271}$$

with the time step sizes

$$\tau_i \;=\; \frac{h}{b}\cdot\frac{1}{2\,\cos^2\left(\pi\cdot\frac{2i+1}{4n+2}\right)} \qquad (i=0,\dots,n{-}1)\,. \tag{3.272}$$

Such a cycle yields the time

$$\theta_n \;=\; \frac{h}{3\cdot b}\left(n^2+n\right) \tag{3.273}$$

for the transport process. From Eq. (3.266) we know that the analytical solution regarding this point in time is given by

$$u\,(x,\theta_n) \;=\; g\,(x \,-\, \theta_n\cdot b) \;=\; g\left(x \,-\, \tfrac{h}{3}\left(n^2+n\right)\right)\;, \tag{3.274}$$

or with $x = ih$,

$$u\,(ih\,,\,\theta_n) \;=\; g\left(\left(i \,-\, \tfrac{n^2+n}{3}\right)h\right)\;. \tag{3.275}$$

Using a cycle length n such that $\frac{n^2+n}{3}\in\mathbb{N}$, we can rewrite this equation in terms of the numerical solutions $\boldsymbol{u}^k$,

$$u_i^n \;=\; u^0_{i\,-\,\frac{n^2+n}{3}}\;. \tag{3.276}$$

On the other hand, the cycle matrix $\prod_i (\boldsymbol{I} \,+\, \tau_i\cdot\boldsymbol{B_h})$ is a lower triangular matrix with bandwidth n, which means that u_i^n is connected to $u^0_{i\,-\,(n^2+n)/3}$ only if $\frac{n^2+n}{3} \leq n$ or equivalently $n \leq 2$. The problem is that the cycle time θ_n implies a shift which grows quadratically in n, but the FED or in general an explicit scheme can only connect values with the maximum shift

n. In particular for large n, the actual solution $u^0_{i-(n^2+n)/3}$ is replaced by a combination of the values u^0_{i-k}, $k = 0, \ldots, n$. They have a large spatial distance to the actual solution. Thus, the numerical solution of the FED cycle in Eq. (3.271) might be a bad approximation.

However, the biggest problem is the stability in the Euclidean norm of Eq. (3.271). Since $\boldsymbol{B_h}$ is not symmetric, the Euclidean norm of the cycle matrix does not correspond to its largest eigenvalue. Actually, we have to consider the square root of the maximum eigenvalue of the matrix product

$$\prod_{i=0}^{n-1} (\boldsymbol{I} + \tau_i \cdot \boldsymbol{B_h})^T \cdot \prod_{i=0}^{n-1} (\boldsymbol{I} + \tau_i \cdot \boldsymbol{B_h}) \ . \tag{3.277}$$

Assuming for example $b = 1$, $h = 1$, $N = 7$ and a very small cycle length $n = 2$, we already have an Euclidean norm that is larger than 1. Larger cycle or signal lengths make it even worse. Note that Theorem 3.7 provides a norm for which Eq. (3.271) is stable. However, due to the bad approximation with respect to the closed-form solution, it is questionable whether the corresponding stability concept makes sense.

Since $\boldsymbol{B_h}$ is unsymmetric, the skew-symmetric part $\frac{1}{2} \cdot (\boldsymbol{B_h}^T - \boldsymbol{B_h})$ is non-zero. Its eigenvalues are imaginary and the largest modulus scales with the ratio $\frac{b}{h}$. Moreover, assuming other boundary conditions like e.g. periodic boundary conditions, the corresponding discretisation matrix $\tilde{\boldsymbol{B}}_{\boldsymbol{h}}$ can have also complex eigenvalues with large imaginary parts. To this end, we are interested in methods having a large imaginary stability boundary. As shown in Fig. 3.3, FED prefers to maximise the real stability boundary.

In fact, there are methods with symbols or amplification factors that are optimised with respect to the imaginary stability boundary [82, 83, 134]. In contrast to the parabolic case, a result proven by Vichnevetsky [142] states that there can be only a linear dependence between the maximum imaginary boundary and the cycle length n. Despite the use of symbols optimised for such hyperbolic problems, a quadratic dependence can not be reached. Thus, a speed-up like in the parabolic case is impossible, no matter which method one uses.

3.6 Numerical Experiments

In the following numerical experiments, we want to show that the proposed cyclic methods can be applied for the efficient solution of parabolic and elliptic problems in PDE-based image analysis.

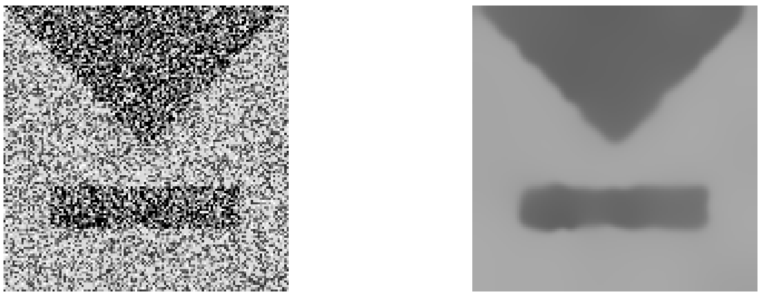


Figure 3.14: Nonlinear isotropic diffusion filtering. **(a) Left:** Original noisy image (128×128). **(b) Right:** Filtered with the AOS scheme ($\tau = 0.02$) using the parameters $\lambda = {}^5\!/_2$, $\sigma = {}^3\!/_2$ and stopping time $T = 100$.

The error measure we are going to use in the experiments is the so-called *mean squared error (MSE)*. Given two images $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ with N pixels, it is defined by

$$MSE(\boldsymbol{u_1}, \boldsymbol{u_2}) \;:=\; \frac{1}{N} \cdot \sum_{i=1}^{N} (u_{1,i} \,-\, u_{2,i})^2 \;. \tag{3.278}$$

All experiments are conducted on an *Intel Xeon* 3.2 GHz with up to four cores. The methods are implemented in *C* with float precision.

3.6.1 Isotropic Diffusion Filtering

At first we consider nonlinear isotropic diffusion filtering [106]. The corresponding evolution equation is

$$\partial_t u \;=\; \operatorname{div}\left(g\left(|\boldsymbol{\nabla} u_\sigma|^2\right) \boldsymbol{\nabla} u\right) \;, \tag{3.279}$$

where we use the Perona-Malik diffusivity given by Eq. (1.13). For the spatial discretisation we assume unit grid sizes $h_x = h_y = 1$.

Our first experiment is about the rearrangement of the time step sequence. To this end, we filter a noisy image by applying one FED cycle with length $n = 35$ such that we reach a stopping time $T = 100$. The original image and a filtered version are illustrated in Fig. 3.14(a) and (b), respectively.

If we use the natural sequence, the FED cycle performs 18 stable time steps $\tau_i \leq \frac{1}{4}$ followed by 17 unstable ones. The two largest steps are given by $\tau_{33} \approx 15.24$ and $\tau_{34} \approx 60.84$, which is more than 60 and 240 times the stability limit, respectively. Some intermediate results can be found in

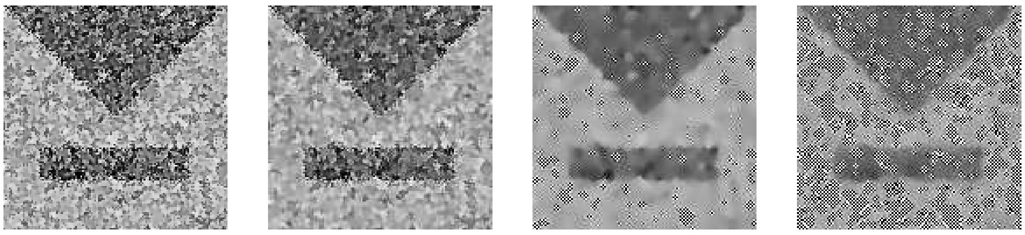


Figure 3.15: Intermediate results of the FED cycle using the natural sequence of the time steps (float precision). **From left to right:** After 10, 20, 34 and 35 steps (final result).

Fig. 3.15. As mentioned above, all computations are performed with float precision. The intermediate result after 20 time steps, i.e. two unstable steps, is still stable, since the unstable steps are relatively small and hence not critical. However, the depicted results after 34 and 35 steps are not stable anymore, due to numerical rounding errors and the very large step sizes. Note that negative grey values are set to 0 and values exceeding the limit 255 to 255. This example shows that a rearrangement of the sequence is indispensable in practice. However, some parts of the final result seem to be reasonable. If we consider the numerical scheme, the filtering result $\boldsymbol{u}$ satisfies

$$\boldsymbol{u} \;=\; \left(\prod_{i=0}^{34}\bigl(\boldsymbol{I} \,+\, \tau_i \cdot \boldsymbol{P}(\boldsymbol{f})\bigr)\right)\boldsymbol{f}\;, \tag{3.280}$$

where $\boldsymbol{f} \in \mathbb{R}^{128^2}$ is the vector whose entries are the grey values of the original image. Rewriting $\boldsymbol{f}$ as a linear combination of the eigenvectors $\boldsymbol{v}_k$, $k = 1, \ldots, 16384$, of the symmetric matrix $\boldsymbol{P}(\boldsymbol{f})$, which means

$$\boldsymbol{f} \;=\; \sum_{k=1}^{16384} C_k \cdot \boldsymbol{v}_k\;, \tag{3.281}$$

with the coefficients $C_k \in \mathbb{R}$, yields for the final result

$$\boldsymbol{u} \;=\; \sum_{k=1}^{16384} \underbrace{\prod_{i=0}^{34}\,(1 \,+\, \tau_i \cdot \mu_k \cdot C_k) \cdot \boldsymbol{v}_k}_{=:\,\tilde{C}_k}\;. \tag{3.282}$$

Here the eigenvalues $\lambda_k \leq 0$ correspond to the eigenvectors $\boldsymbol{v}_k$. Obviously, the reasonable parts of the image correspond to eigenvectors or fre-

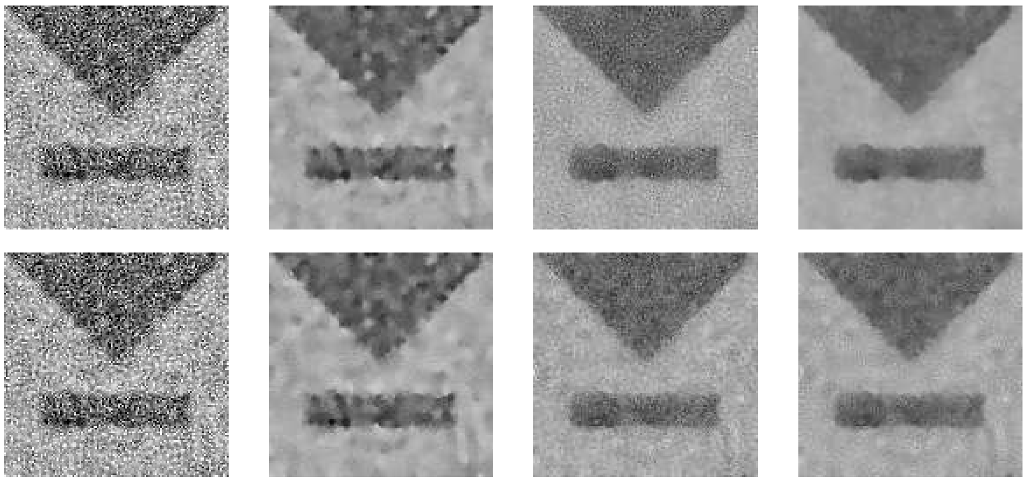


Figure 3.16: Intermediate results of the FED cycle, where the sequence of the time steps has been rearranged by κ-cycles ($\kappa = 7$). **Top row**: Without any inner update of the nonlinearities. **Bottom row**: With one inner update after 10 steps. **From left to right**: After 10, 20, 34 and 35 time steps (final result).

quency components whose coefficients $\tilde{C}_k$ are not or less affected by numerical rounding errors. On the other hand, the unstable parts come from frequency components whose numerically evaluated coefficients are much larger than the actual coefficients $\tilde{C}_k$ (cf. e.g. Fig. 3.4). If we perform the computations with long double precision, the final result is stable and in fact, the grey values of this result match many values in the reasonable parts of the result with float precision.

Some intermediate results using the rearrangement by a κ-cycle with $\kappa = 7$ are depicted in Fig. 3.16. Here the final result is stable, but we also see that there might be unstable solutions within a cycle. If we consider, for example, the result after ten explicit time steps, the minimum grey value is about -135 and the maximum one is around 338. This shows that updating the nonlinearities within a cycle can be very dangerous: Unstable inner results yield very bad nonlinearities which misdirect the evolving image. The bottom row in Fig. 3.16 shows the evolution with an inner update after 10 explicit time steps. In this case, the grey values of the final result still range in $[0, 255]$, but there is more noise.

For the rearrangement with Leja ordering we have similar observations. The final result of the top row in Fig. 3.17 is stable and equal to the one of Fig. 3.16. However, the intermediate result after 10 steps is very unstable

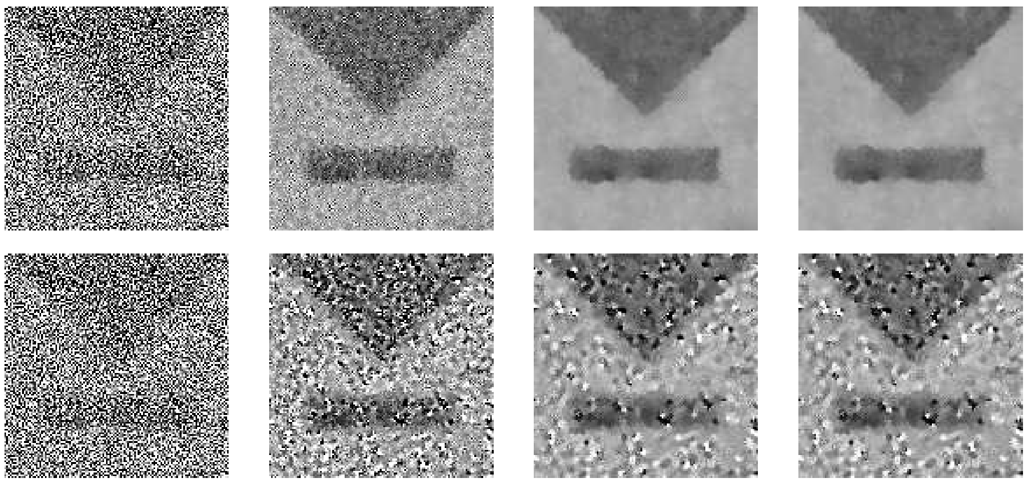


Figure 3.17: Intermediate results of the FED cycle using the Leja-ordered sequence of the time steps. **Top row:** Without any inner update. **Bottom row:** With one inner update after 10 steps. **From left to right:** After 10, 20, 34 and 35 time steps (final result).

and an update yields now even an unstable final result. It can be seen in the bottom row of Fig. 3.17.

Now we consider the Lebedev-Finogenov ordering. Since the cycle length reaching the stopping time $T = 100$ must be larger than 34, we now have to work with the length $2^8 = 64$. Actually, this allows the maximum stopping time $T \approx 346.67$, and therefore we have to use a relatively small time adjustment factor $q \approx 0.29$ yielding only 15 unstable time steps. However, the largest time step $\tau_{63} \approx 60.81$ is approximately as large as the maximum step size of the cycle with length 35. Although the cycle contains much more stable steps, a rearrangement is necessary. This is illustrated in Fig. 3.18. It demonstrates that the method of Lebedev and Finogenov works also well with FED. However, the main disadvantage of this approach is the limitation of the cycle lengths. As already mentioned, this can cause a duplication of the computational effort in the worst case.

Our last experiment with respect to the reordering strategies is about their limitation, i.e. we want to find out the maximum cycle length that yields a stable method. To this end, we perform one FED cycle with the above mentioned parameters $\lambda = 5/2$ and $\sigma = 3/2$.

Regarding the rearrangement with the κ-cycles, we use the look-up table whose parameters are mentioned in Sec. 3.2.3. This table yields stable results up to the length $n = 1200$, which corresponds to the maximum

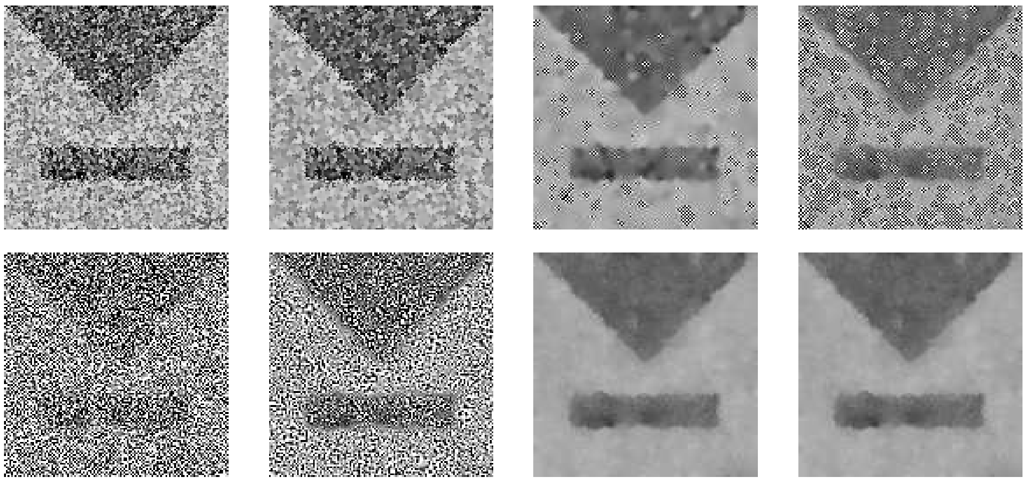


Figure 3.18: Intermediate results of the FED cycle with length 64. **Top row:** Natural sequence. **Bottom row:** Rearrangement of Lebedev and Finogenov. **From left to right:** After 20, 40, 63 and 64 time steps (final result).

stopping time $T = 120100$. However, larger cycle lengths are possible, if one uses other values for κ that differ from the ones in the look-up table. Moreover, increasing the signal length (cf. Sec. 3.2.3) further improves the maximum cycle length. This shows that the stability is very sensitive regarding the choice for κ. Thus, this strategy is more suited for cycle lengths up to about $n = 1000$.

In contrast to the κ-cycles, the Leja ordering does not need an optimisation of a parameter, because the sequence only depends on the set of the reciprocals of the time step sizes. However, as mentioned in Sec. 3.2.3, the stability of the Leja sequence massively depends on the numerical precision. If we compute it by means of float precision, then this already yields unstable cycles for small lengths $n \geq 124$. The computation of the sequence with long double precision significantly improves the stability. In this case, the maximum cycle length is about $n = 8300$ yielding the time $\theta_n = 5741525$. Thus, it is much better than the κ-cycles for cycle lengths larger than 1000. However, in contrast to κ-cycles, a look-up table requires $\mathcal{O}(n^2)$ entries.

The Lebedev-Finogenov ordering allows reasonable, stable results up to the cycle length $n = 16834$. Actually, this means that it is possible to reach a maximum cycle time $\theta_n \approx 22370987$. Hence, it seems to be the most robust strategy, and very well-suited for very huge stopping times.

For the following experiments we use solely κ-cycles to reorder the time

Table 3.4: MSE-Comparison of FED and AOS for the testbed in Fig. 3.14.

cycles/steps	**MSE**	
	FED	**AOS**
1	41.296	241.788
2	5.743	75.906
3	2.363	34.065
4	1.355	18.972
5	0.891	12.175
10	0.255	3.321
25	0.044	0.623
50	0.011	0.170

steps, because the cycle lengths do not exceed $n = 1000$ and we would like to have an optimal behaviour with respect to the computational effort, which means we want to use all cycle lengths from 1 to 1000.

At this point, we want to analyse the FED scheme for nonlinear isotropic diffusion filtering with respect to accuracy and efficiency. As we have mentioned in the introductory chapter, an efficient scheme for nonlinear isotropic diffusion filtering is the AOS scheme [93, 155]. It is also well-suited for parallel computing, which has been shown for instance in [17, 149, 156].

To evaluate the accuracy of both FED and AOS, we take the reference image $\boldsymbol{r}$ in Fig. 3.14(b) with stopping time $T = 100$ and compare it to filtering results $\boldsymbol{u}$ computed with different numbers of time steps by means of the errors $MSE(\boldsymbol{r}, \boldsymbol{u})$. We shall note that the grey values of both $\boldsymbol{r}$ and $\boldsymbol{u}$ are given in float precision. The corresponding results can be found in Table 3.4. Obviously, FED clearly outperforms the AOS scheme with respect to the MSE. The mean squared errors of FED are up to about 15 times smaller, although the reference image has been filtered with AOS. This is due to the fact that FED produces no splitting error, which increases for larger time step sizes or equivalently for a smaller number of AOS time steps.

The table also confirms the theoretical results about the approximation order of the FED scheme, which means that doubling the number of FED cycles should asymptotically decrease the Euclidean norm $\|\boldsymbol{u} - \boldsymbol{r}\|_2$ by a factor of two. If we consider the MSEs for both 25 and 50 FED cycles, the error with 50 cycles is about four times smaller than for 25. This means

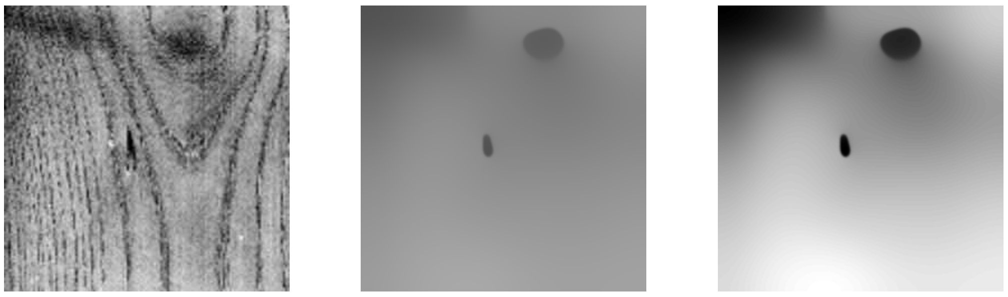


Figure 3.19: Defect detection in a wood surface with nonlinear isotropic diffusion filtering. **(a) Left:** Original image (256×256). **(b) Middle:** Filtered with the AOS scheme ($\tau = 0.05$) using $\lambda = 3/2$, $\sigma = 1$ and stopping time $T = 800$. **(c) Right:** Rescaled to $[0, 255]$ for better visualisation.

that the square root of the MSE is two times smaller and since the square root of the MSE is related to the Euclidean norm (or distance), the decrease of the MSE perfectly corresponds to the doubling of the number of FED cycles.

In our second example, nonlinear isotropic diffusion filtered is used to detect some defects in a wood surface. This is illustrated in Fig. 3.19. Here we have both a larger image and a bigger stopping time.

We again compare the filtering results of FED and AOS to the reference solution shown in Fig. 3.19(b). The MSEs are given in Table 3.5. Like before, the errors of the FED scheme are significantly smaller with a factor up to 25, and we recognize the connection between the decrease of the MSE and the number of FED cycles.

So far, we have seen that the MSEs for the AOS scheme are much larger than for FED. Thus, we can state that the FED scheme is more accurate, but in this context, it is also interesting to look into the efficiency of the methods. More precisely, we want to measure the computing times that are necessary to reach certain MSEs. To this end, we apply both FED and AOS with different numbers of cycles or time steps, compute the MSE with respect to the reference solution and determine the corresponding computing time. Since both schemes are well-suited for parallelisation, we use *OpenMP*[1] to show the benefit of parallel computing with four cores. Note that the additional effort is just one code line. However, in Sec. 3.6.5 we will also present an experiment with GPUs to fully exploit the parallelism

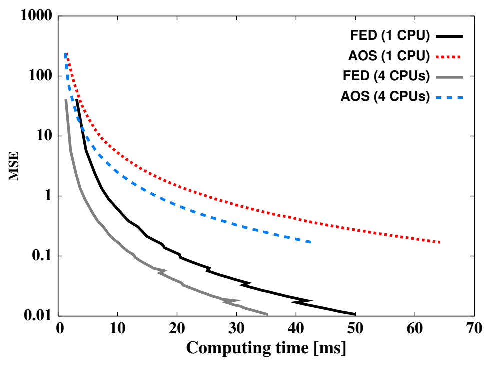
[1]see 'openmp.org' for details

Table 3.5: MSE-Comparison between FED and AOS for the defect detection depicted in Fig. 3.19.

cycles/steps	**MSE**	
	FED	**AOS**
1	123.578	268.027
2	24.110	148.356
3	8.214	91.093
5	2.932	40.922
10	0.904	10.910
25	0.146	2.092
50	0.037	0.660
100	0.009	0.188
200	0.002	0.050

of our proposed methods. In the case of FED, we just have matrix-vector multiplications that are very easy to parallelise, because each entry of the resulting vector can be treated separately. For the AOS scheme, we use the so-called *mid grain parallelism* (see e.g. [17]), which means that the linear systems of equations for each direction are decomposed into many small independent systems. For simplicity, all other components of the methods like, for example, the computation of the nonlinearities are not parallelised.

The results for the test setting in Fig. 3.14 are shown in Fig. 3.20(a). Here the number of the cycles or steps ranges from 1 to 50. As one can see, the curves for FED contain some jags that come from the time adjustment factor q: If the stopping time $T = 100$ has to be reached, for example, with 28 FED cycles, the overall step size of one cycle is about 3.57, which means that the corresponding cycle length is $n = 7$ with a relative small adjustment factor $q \approx 0.765$. The overall number of explicit time step is $28 \cdot 7 = 196$ in this case. Increasing the number of FED cycles to 29 means one more update of the nonlinearities, but a smaller cycle length $n = 6$ is sufficient, and $q \approx 0.985$. Thus, we have $29 \cdot 6 = 174$ time steps, i.e. 22 less than before. If the additional update is cheaper than 22 explicit time steps, the computing time is reduced and there is a smaller MSE because of the extra cycle. Furthermore, we shall note that the computation of the time step sizes and the rearrangement by κ-cycles are included in the computing time of the FED scheme. Thus, the compilation of a database with the rearranged sequences could further decrease the running time. However, we



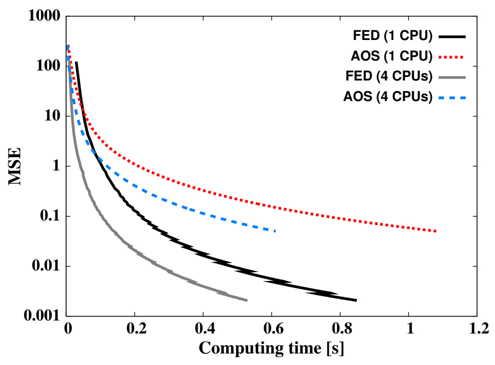


Figure 3.20: Computing time (milliseconds/seconds) versus MSE for both the AOS and FED scheme with one or four CPUs. The y-axis is log-scaled. **(a) Left:** Test setting in Fig. 3.14. **(b) Right:** Defect detection in Fig. 3.19.

can see that FED is much more efficient than the AOS scheme. Assuming the same computing time, the MSE of FED is up to more than one order of magnitude smaller. If we suppose an MSE smaller than 10, the FED scheme using one CPU core is even more efficient than AOS with four CPU cores. Overall, both schemes benefit from the parallelisation, and the improvement of the computing time is at least 30%.

Figure 3.20(b) illustrates the results for our second example. The number of cycles or steps ranges from 1 to 200. Because of both the larger image size and the larger diffusion time, the corresponding computing times increase compared to the last test setting. However, the courses of the curves look pretty similar. Like before, the FED scheme with one core can be much more efficient than AOS using the mid grain parallelism. This shows that the splitting error of AOS is very disadvantageous.

After the comparison with the widely used AOS scheme, we want to look into nonlinear isotropic diffusion filtering with Super Time Stepping (STS). Some results for both test settings are given in Table 3.6. As expected, the method without a damping factor, i.e. $\nu = 0$, has big problems with the noise removal and yields, in particular for the noisy image, a much larger MSE than its damped variant. Even with 50 cycles, the MSE is still larger than 1. However, in the case of the defect detection, a larger number of cycles can yield better results than damped STS or FED, because there is almost no noise in the original image. Due to the smaller cycle lengths of the undamped STS, it allows a faster computation. Therefore, it can be more efficient for a larger number of cycles. Regarding STS with $\nu > 0$,

Table 3.6: MSEs for STS without and with a damping factor.

cycles	**MSE** (Fig. 3.14)		**MSE** (Fig. 3.19)	
	STS ($\boldsymbol{\nu = 0}$)	**STS**	**STS** ($\boldsymbol{\nu = 0}$)	**STS**
1	2490.590	44.713	308.494	122.748
2	1226.769	5.690	134.962	23.679
3	628.415	2.465	68.145	8.699
5	178.374	0.928	16.237	3.201
10	29.941	0.269	1.662	1.019
25	3.263	0.048	0.132	0.176
50	1.076	0.012	0.028	0.044
100	0.494	0.002	0.006	0.011
200	0.001	0.000	0.001	0.002

we have chosen the damping parameters such that the cycle lengths n and the diffusion times per cycle coincide with FED. Because of $\nu > 0$, the noise removal is much better, even for smaller numbers of cycles. However, the MSEs of FED are mostly better than the ones of the damped STS, although both methods have the same cycle length and overall step size per cycle. This is due to the better smoothing properties of FED. An example is illustrated in Fig. 3.21. One can see that the result of one cycle with undamped STS is still very noisy. Introducing a damping factor $\nu > 0$ significantly improves the image, but there is still more noise than in the FED result, which might explain the larger MSE. On the other hand, the edges within the image computed with the damped STS are a bit sharper due to the weaker smoothing effect.

Therefore, a damping factor $\nu > 0$ should be used for STS. However, it is very difficult to determine an appropriate factor. If it is too small, one might have problems with noisy images and yield bad results. On the other hand, a large parameter decreases the cycle time and hence reduces the efficiency of the method. Thus, it seems to be very difficult to find an optimal damping parameter. Table 3.7 shows some damping parameters for both test settings with different numbers of cycles. They are chosen such that STS has the same cycle length and cycle time as FED. Obviously, the damping parameter does not seem to obey a certain rule. Nevertheless, we can only state that a larger number of cycles, i.e. a smaller cycle length, also requires a larger damping parameter and that the order of magnitude depends on the stopping times (here: $T = 100$ or $T = 800$) of the

Table 3.7: Damping parameters (rounded) for STS.

cycles		1	5	10	25	50	100
$\boldsymbol{\nu}$	3.14	0.0038	0.0175	0.0382	0.0996	0.2149	0.3127
	3.19	0.0005	0.0023	0.0047	0.0122	0.0240	0.0497

corresponding diffusion processes.

In conclusion, our experiments show that for isotropic parabolic problems, FED is more efficient than AOS and has better smoothing properties than STS, regardless whether one applies it with $\nu = 0$ or a positive damping parameter.

3.6.2 Anisotropic Diffusion and Inpainting Problems

As already mentioned in the introduction, a numerically more challenging scenario is given by anisotropic problems. Such processes are based on the nonlinear anisotropic diffusion equation

$$\partial_t u \;=\; \operatorname{div}\left(\boldsymbol{D}\left(\boldsymbol{\nabla} u_\sigma\right)\boldsymbol{\nabla} u\right) \tag{3.283}$$

with a symmetric, positive definite diffusion tensor $\boldsymbol{D} \in \mathbb{R}^{2\times 2}$. It can be seen as a generalisation of isotropic processes that assume a diagonal matrix. In the introductory chapter, we have already reviewed two specific filters, namely the edge-enhancing anisotropic diffusion (EED) and the coherence-enhancing diffusion (CED) filter [147].

Parabolic Problems

To evaluate FED for anisotropic diffusion problems, we first enhance a fingerprint test image with CED. As a space discretisation for CED, we have used the one in [157]. The test setting is depicted in Fig. 3.22, where the filtered reference result has been computed by the semi-implicit scheme. This scheme permits to use large time step sizes and can be more efficient than the usual explicit approach. Moreover, there is no efficient operator splitting scheme in the general anisotropic case. Thus, we want to compare FED and the semi-implicit scheme.

Since the semi-implicit method requires the solution of large, sparse linear systems of equations with positive definite system matrices, we use the conjugate gradient (CG) method [74]. It is an easy and fast iterative solver

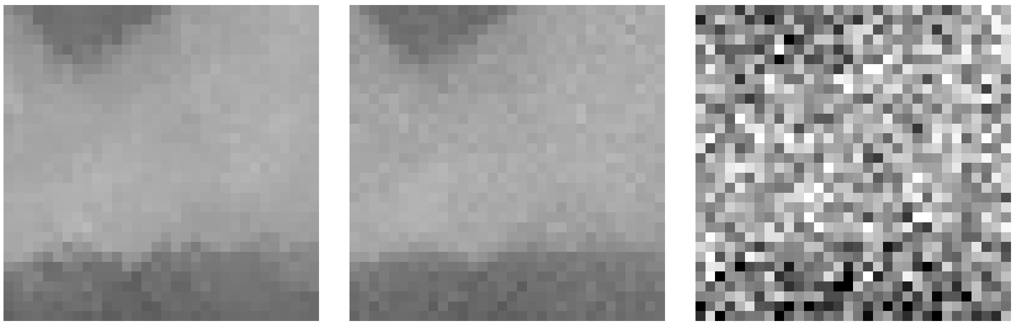


Figure 3.21: 32×32 detail of the filtering results for the noisy image with one cycle ($T = 100$). **(a) Left:** FED. **(b) Middle:** STS, $\nu = 0.0038$. **(c) Right:** STS, $\nu = 0$.

that is well-suited for parallel computing. Moreover, unlike e.g. the method of successive over-relaxation (SOR), the values at each iteration of the CG method do not depend on the order of the equations or, in terms of the implementation, the processing order of the pixels. This is very important regarding for instance rotational invariance. Another nice property is the possible preservation of the average grey value in each iteration, which means that this is independent of any stopping criteria. To guarantee the preservation, the initial vector for the CG method has to be e.g. equal to the corresponding right hand side of the linear system being solved. Concerning our numerical scheme from Eq. (1.28), we start with $\boldsymbol{v} = \mathbf{0}$.

We dispense with the preconditioned conjugate gradient (PCG) method, because common preconditioners like, for instance, the symmetric Gauss-Seidel or the symmetric SOR depend on the order of the equations and, in the latter case, can require additional parameters that have to be optimised. Moreover, these optimal values depend on the used time step size. On the other hand, the Jacobi preconditioner is an example that does not depend on the order of the equations or additional parameters, but it can not improve the efficiency for the values of the diffusion times used within the experiments. Moreover, unlike CG, the PCG method does not directly evaluate the Euclidean norm of the residuals. Thus, this evaluation requires additional computational effort.

At first, we apply FED as well as the semi-implicit scheme with different numbers of cycles or time steps, and compute the errors between the filtering results and the reference image in Fig. 3.22. Some MSEs are given in Table 3.8. For a small number of cycles or steps, the semi-implicit scheme has clearly better MSEs, and FED can beat it only for example with 50 or

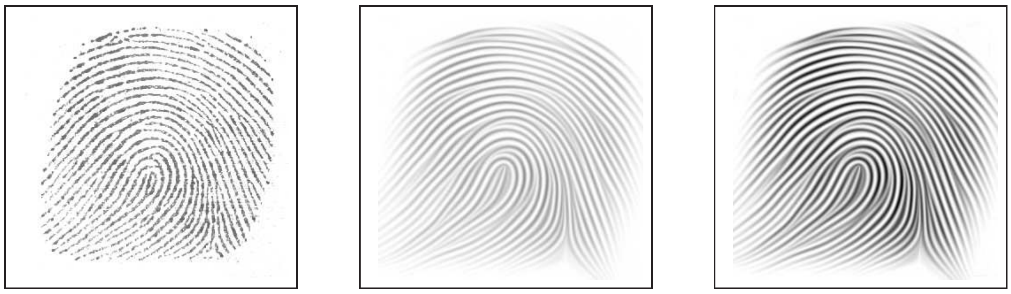


Figure 3.22: Fingerprint enhancement with CED. **(a) Left:** Original image (300×300). **(b) Middle:** Filtered with the semi-implicit scheme ($\tau = 0.05$) using the model parameters $\lambda = 1$, $\sigma = 1/2$, $\rho = 4$, $\alpha = 10^{-3}$ and stopping time $T = 300$. **(c) Right:** Rescaled to $[0, 255]$ for better visualisation.

100 cycles.

Now the question is whether FED has, in spite of the lower accuracy, a better efficiency than the semi-implicit scheme. However, the comparison is not so easy as in the case of AOS, since the CG method requires a stopping criterion that influences the efficiency. There are mainly two criteria: One can limit the number of iterations or give an error tolerance $\epsilon > 0$. More precisely, given the linear system $\boldsymbol{Bx} = \boldsymbol{c}$, the CG method stops if the residual with respect to the current iteration $\tilde{\boldsymbol{x}}$ satisfies

$$\|\boldsymbol{B}\tilde{\boldsymbol{x}} - \boldsymbol{c}\|_2 < \epsilon \cdot \|\boldsymbol{c}\|_2 \, . \tag{3.284}$$

However, the condition numbers of the system matrices $(\boldsymbol{I} - \tau \boldsymbol{P})$ depend on the time step size $\tau > 0$, i.e. they increase for larger step sizes. On the other hand, the convergence of the CG method depends on the magnitude of the condition number. Actually, this means that a stopping criterion which only includes a limit for the number of iterations is not suitable. If the limit is too small, linear systems with a large step size τ are solved inaccurately, and by using bigger limits one spends too much time for the solution of linear systems with small time step sizes. Thus, we are going to use the residual criterion. It is flexible with respect to the different magnitudes of the time step sizes.

In Fig. 3.23 we have depicted the trade-off between the CPU time and the MSE for one as well as four CPUs. The number of FED cycles and semi-implicit time steps varies from 1 to 100, respectively. For the semi-implicit method we have used $\epsilon = 10^{-4}$ and $\epsilon = 10^{-3}$ as stopping criteria. Regarding the efficiency, the latter choice should be preferred. However, FED is still more efficient. Larger values like, for example, $\epsilon = 10^{-2}$ do not

Table 3.8: MSE-Comparison between FED and the semi-implicit scheme.

cycles/steps	**MSE**	
	FED	**semi-imp.**
1	88.382	48.402
2	39.429	27.809
3	27.448	20.448
4	21.758	16.670
5	18.239	14.241
10	11.093	8.304
25	3.841	2.938
50	0.943	1.114
100	0.260	0.381

allow a reasonable numerical solution of the linear systems anymore, and smaller values would worsen the efficiency.

Besides a matrix-vector product in each iteration, the CG method requires the computation of at least two inner products, which we have also parallelised. Both FED and the semi-implicit scheme are faster, whereas the latter one benefits more from the parallel computing. This is due to the fact that the proportion of the CG method in the whole scheme is larger than the one of the FED cycles, and we do not have parallelised other components like, for instance, the update of the nonlinearities or the Gaussian convolution. Nevertheless, we can state that FED still outperforms the semi-implicit scheme with respect to the efficiency.

Elliptic Inpainting Problems

Let us now consider an elliptic inpainting problem that we are going to solve first by computing the steady-state of a parabolic evolution. As a testbed we use an inpainting problem that is relevant for image compression with EED [54, 121]. The corresponding test setting is depicted in Fig. 3.24. In this example, we store about 10% of the original image data and use this to compute a good reconstruction of the original image with the help of edge-enhancing anisotropic diffusion filtering. As mentioned in the introduction, we use the Charbonnier diffusivity given in Eq. (1.23) within the model.

In order to have a faster convergence to the steady-state of the parabolic evolution, we apply the cascadic FED scheme (CFED) and compare it with

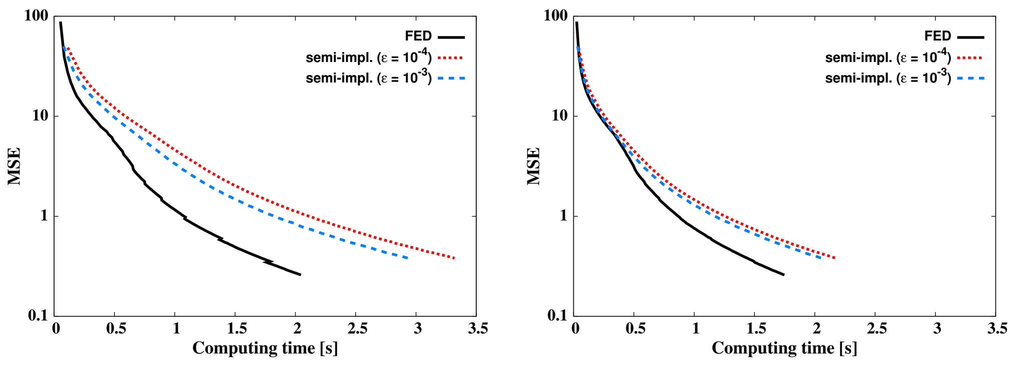


Figure 3.23: Computing time (seconds) versus MSE for FED and semi-implicit schemes with different stopping criteria. **(a) Left:** With one CPU. **(b) Right:** With four CPUs.

a cascadic semi-implicit approach. To simplify matters, we always use the same number M of FED cycles with length n for the diffusion process on each level of the Gaussian pyramid. Since we start with the unit grid sizes $h_x = h_y = 1$ and the image size 256×256, the coarser levels $\ell \geq 1$ have the mesh sizes $h_x^{(\ell)} = h_y^{(\ell)} = 2^\ell$ and the corresponding image sizes are $2^{8-\ell} \times 2^{8-\ell}$. In this case, the restriction operator just averages two neighbouring pixels to get the values with respect to the coarser grid (linear interpolation), and the prolongation operator corresponds to nearest-neighbour interpolation. Note that the time step size limit $\tau_{\text{lim}}^{(\ell)}$ at level ℓ is given by

$$\tau_{\text{lim}}^{(\ell)} = \frac{1}{\frac{2}{\left(h_x^{(\ell)}\right)^2} + \frac{2}{\left(h_y^{(\ell)}\right)^2}} = 4^{\ell-1}\,, \tag{3.285}$$

which is 4^ℓ times the limit on the original level. Thus, we adapt the time step sizes of the FED cycle to each level's limit. However, the increasing limit for the explicit scheme means also an improvement for the condition numbers of the linear systems appearing in the semi-implicit scheme. Therefore the adaptation of the time steps also makes sense for the semi-implicit method, where we use the same factor as for CFED, i.e. 4^ℓ.

For our experiments we apply a cascadic approach incorporating four levels: 256×256, 128×128, 64×64 and 32×32. In the case of 16×16, the downsampled inpainting mask already contains 254 of the 256 pixels, and other coarser levels do not require any inpainting process. We use two different stopping times $T = 250$ and $T = 1000$ that refer to the original level. Given the number of FED cycles, we can compute the corresponding cycle length n to reach these stopping times on the original grid. As

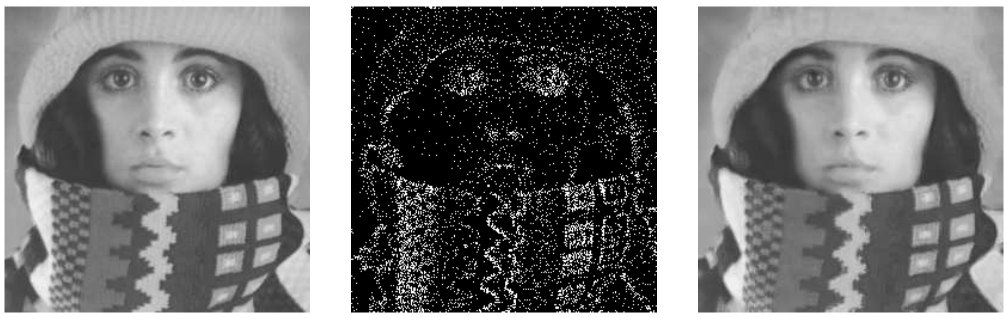


Figure 3.24: EED-based image reconstruction. **(a) Left:** Original image (*trui*, 256×256). **(b) Middle:** Inpainting mask with 10% specified pixels. **(c) Right:** Reconstruction with EED-based inpainting in the unspecified regions (semi-implicit, $T = 100000$, $\tau = 0.2$, $\lambda = 0.1$, $\sigma = 2$).

mentioned above, the stopping times on the coarse levels increase by the factor 4^ℓ, because we keep the number of time steps constant on each level and only increment the step sizes. We compare the results to the reference reconstruction depicted in Fig. 3.24(c). The errors for both CFED and the cascadic semi-implicit scheme are shown in Table 3.9. In contrast to the parabolic case, the MSEs of the cascadic FED method are smaller than the ones of the semi-implicit counterpart, especially for the smaller stopping time $T = 250$. However, the use of such relatively small stopping times requires good initialisations from the coarse levels. One can see this, for instance, in the row referring to two FED cycles. Here, the MSE increases from 2.09 to 2.19 despite the larger stopping time, which probably comes from a worse initialisation in the case of $T = 1000$.

The efficiency of both cascadic approaches with four levels is illustrated in Fig. 3.25, where the number of cycles or time steps ranges from 1 to 50. As before, the semi-implicit scheme is combined with a CG solver that uses, due to efficiency constraints, the larger residual tolerance $\epsilon = 10^{-3}$. Overall, the CFED method is more efficient, and it can be even better than a parallelised version of the cascadic semi-implicit scheme.

So far, we have solved the inpainting problem by a parabolic evolution with nonlinear diffusion filtering. For explicit schemes or FED, this is related to the direct solution of the corresponding elliptic problem. In this context, we could replace CFED by a cascadic Fast-Jacobi method that refers to the direct solution of the elliptic equation. We have already seen that the Fast-Jacobi solver can be interpreted as some kind of a preconditioned explicit scheme.

Table 3.9: MSE-Comparison of CFED and the cascadic semi-implicit scheme with four levels for the stopping times $T = 250$ and $T = 1000$.

cycles/steps per level	**MSE**			
	CFED		**casc. semi-imp.**	
	$\boldsymbol{T = 250}$	$\boldsymbol{T = 1000}$	$\boldsymbol{T = 250}$	$\boldsymbol{T = 1000}$
1	7.60	7.50	8.43	7.81
2	2.09	2.19	2.69	2.28
5	0.84	0.65	1.05	0.72
10	0.64	0.35	0.77	0.42
25	0.54	0.21	0.62	0.26
50	0.52	0.19	0.56	0.20

To this end, we now apply the Fast-Jacobi solver in a cascadic approach (CFJ). The relaxation parameters are given by

$$\omega_i \;=\; \omega_{\text{lim}} \cdot \frac{1}{2\,\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \ldots, n-1)\,, \tag{3.286}$$

where we have to define the parameter $\omega_{\text{lim}} > 0$. Following the thoughts in Sec. 3.4.4, this parameter should be chosen smaller than 1. In our experiment we use $\omega_{\text{lim}} = 0.95$, which implies a stable method. Because of the additional diagonal preconditioning, we might use smaller cycle lengths compared to CFED. To compare the methods, we apply them using the same cycle length, increase the number of cycles, and consider the MSEs with respect to the reference solution.

The results of this comparison are given in Table 3.10. For the smaller cycle length $n = 20$, the cascadic Fast-Jacobi outperforms CFED with respect to accuracy. Since the additional effort is only marginal, we can also expect a better efficiency. Note that an FED cycle with the length $n = 20$ corresponds to the very small stopping time 35 on the original grid. However, if one uses $n = 100$, the stopping time is about 25 time larger. This improves the results for CFED. On the other hand, this cycle length is not optimal for the cascadic Fast-Jacobi (CFJ) algorithm, and both methods yield similar results.

Overall, we have seen that the solution of anisotropic elliptic problems with Fast-Jacobi can be more efficient. However, as we have mentioned in Sec. 3.4.5, we expect that the diagonal preconditioning is less efficient

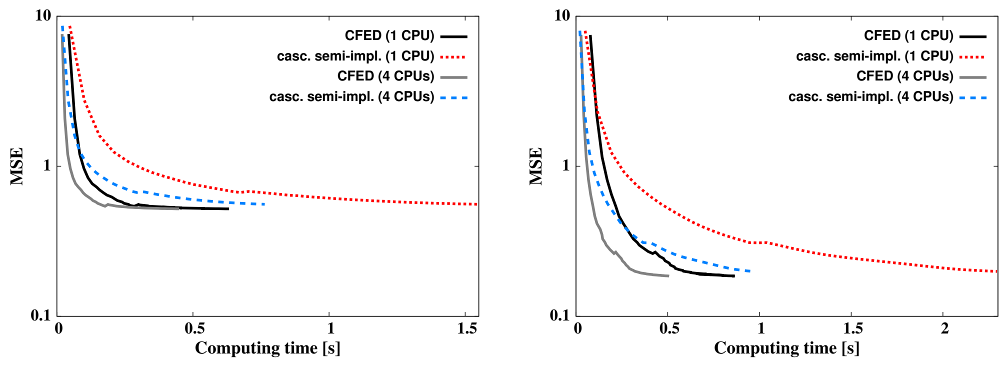


Figure 3.25: Computing time (seconds) versus MSE for CFED and the cascadic semi-implicit scheme using four levels. **(a) Left:** With stopping time 250 on the original level. **(b) Right:** Stopping time 1000.

for (linear) problems with constant coefficients. To this end, we show that a parabolic approach with CFED is already well-suited for such tasks.

3.6.3 Elliptic Problems with Constant Coefficients

As a prototype for an elliptic problem with constant coeffcients, we consider linear biharmonic image inpainting. This means we consider the problem in Eq. (3.169) and replace the divergence term by the (negative) *biharmonic* or *bilaplacian operator*:

$$c(\boldsymbol{x}) \cdot (v(\boldsymbol{x}) - f(\boldsymbol{x})) - (1 - c(\boldsymbol{x})) \cdot (-\Delta^2 v(\boldsymbol{x})) = 0 . \qquad (3.287)$$

The corresponding spatial discretisation on a grid with N nodes yields the linear system

$$\boldsymbol{C}(\boldsymbol{v} - \boldsymbol{f}) - (\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}\boldsymbol{v} = \boldsymbol{0} , \qquad (3.288)$$

where $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ is the discrete version of $-\Delta^2$. In the 1-D case, we have $\boldsymbol{P} = -\boldsymbol{A}_h^2$ with the symmetric, negative semi-definite matrix $\boldsymbol{A_h}$ given in Eq. (3.33). Thus, $\boldsymbol{P}$ is also symmetric, negative semi-definite and has the same rank as the matrix discretising the Laplacian operator, namely $N-1$. Thus, Theorem 3.9 states that the above linear system has a unique solution $\boldsymbol{v} \in \mathbb{R}^N$. This is also valid for the multi-dimensional case.

The solution of Eq. 3.288 with the help of an explicit parabolic evolution reads

$$\boldsymbol{v}^{k+1} = \Big(\boldsymbol{I} + \tau \cdot (\boldsymbol{I} - \boldsymbol{C})\,\boldsymbol{P}\Big)\boldsymbol{v}^k . \qquad (3.289)$$

Since the eigenvalues corresponding to $\boldsymbol{P}$ are the negative squares of the eigenvalues of the discrete Laplacian, the explicit time step size limit of the

Table 3.10: MSE-Comparison between CFED and CFJ using four levels for the cycle lengths $n = 20$ and $n = 100$.

length n		**number of cycles (per level)**					
		1	**2**	**3**	**4**	**5**	**10**
20	**CFED**	8.64	2.65	1.53	1.21	1.00	0.51
	CFJ	7.72	2.31	1.30	0.97	0.78	0.39
100	**CFED**	7.55	2.16	1.14	0.81	0.64	0.35
	CFJ	7.62	2.18	1.15	0.82	0.65	0.35

2-D case is now given by

$$\tau_{\text{lim}} = \frac{2}{\left(\frac{4}{h_x^2} + \frac{4}{h_y^2}\right)^2} . \tag{3.290}$$

It can be much smaller than in the case of the Laplacian, in particular for small mesh sizes $h_x \approx 0$ and $h_y \approx 0$. To this end, it is interesting to analyse whether a direct solution of Eq. (3.288) with the help of Fast-Jacobi is more efficient.

However, we have to take care of the parameter $\omega_{\text{lim}} > 0$ that is necessary for the computation of the relaxation parameters. The eigenvalues of the matrix $\boldsymbol{D}_{\boldsymbol{P}}^{-1}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}$ can be estimated with the help of Gershgorin's theorem. For $h_x = h_y$ we have the upper bound $^{10}/_{3}$ and can work with $\omega_{\text{lim}} = {}^{3}/_{5}$ in order to guarantee stability as well as convergence to the unique solution. Since the minimum eigenvalue of $\boldsymbol{D}_{\boldsymbol{P}}^{-1}(\boldsymbol{I} - \boldsymbol{C})\boldsymbol{P}$ is equal to 0, the relaxation parameters of the modified Richardson method would have to be computed with $\lambda_{\min} = 0$ and $\lambda_{\max} = {}^{10}/_{3}$. However, the results for $\lambda_{\min} = 0$ are very bad such that we prefer better damping properties with $\lambda_{\min} = 10^{-3}$.

Our test setting is shown in Fig. 3.26. The reference solution has been computed with the above explicit parabolic scheme using the small stable step size $\tau = 0.025$ and 40 million time steps. For our experiment we apply both CFED and CFJ as well as a cascadic modified Richardson (CMR), where the cascadic approach covers four levels from 256×256 to 32×32. The level 16×16 has only one unspecified pixel and the inpainting masks corresponding to coarser levels specify all pixels. Note that we use again $h_x = h_y = 1$ on the original level.

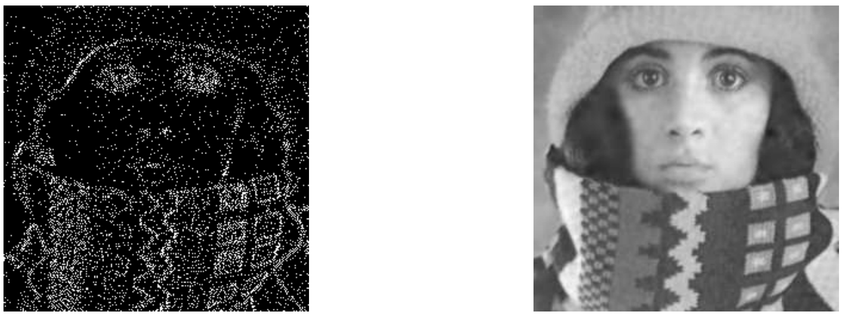


Figure 3.26: Biharmonic inpainting for *trui*. **(a) Left:** Inpainting mask with 10% specified pixels. **(b) Right:** Reconstruction with biharmonic inpainting (explicit scheme, $T = 1000000$, $\tau = 0.025$).

We assume an overall number of 1000 FED time steps or solver iterations on each level, and try different cycle lengths in our experiment. With one CPU, both FED and the Jacobi-type methods need about 0.63 seconds and the parallel versions on four CPUs finish their computations after 0.17 seconds. This corresponds to an expected speed-up factor around four. Some results are given in Table 3.11. The optimal cycle length for both FED and Fast-Jacobi seems to be 250, because the results with other cycle lengths are worse. Moreover, both methods yield almost the same small errors. For all numbers of cycles, the difference is always smaller than $2.1 \cdot 10^{-4}$, and FED provides better results in most instances. This shows that there is virtually no benefit with respect to a diagonal preconditioning. Interestingly, the cascadic modified Richardson method is more robust with respect to the cycle lengths, since the MSE is almost constant. However, the results are very sensitive with respect to the choice of $\lambda_{\min}$. With $\lambda_{\min} = 0$ the MSE varies from about 203 (one cycle) to 8.56 (twenty cycles), which is up to four orders of magnitude larger than the MSEs of the damped method with $\lambda_{\min} = 10^{-3}$. Thus, cascadic methods with Richardson-based relaxation parameters should be only applied in combination with a positive $\lambda_{\min}$. Overall, this example shows that cascadic parabolic approaches work well for elliptic problems with constant coefficients, and the direct solution with Jacobi-like methods does not pay off in this case.

3.6.4 Elliptic Problems with Strongly Varying Coefficients

In Sec. 3.4.5 we have seen that the diagonal preconditioning performs best for problems with strongly varying coefficients. We show that in this case

Table 3.11: MSE-Comparison between CFED, CFJ and cascadic modified Richardson (CMR) with 1000 time steps/iterations per level for biharmonic inpainting.

cycles per level	**MSE** $(\cdot \mathbf{10^{-3}})$		
	CFED	**CFJ**	**CMR**
1	3.39	3.60	17.19
2	0.62	0.62	17.18
4	0.35	0.32	17.23
5	1.35	1.32	17.17
10	12.25	12.46	17.26
20	42.62	44.30	19.49

Jacobi methods such as Fast-Jacobi are more efficient than FED.

Our prototypical scenario is given by an isotropic nonlinear image regularisation method. It computes a denoised version $u(\boldsymbol{x})$, $\boldsymbol{x} \in \Omega$, of the original image $f(\boldsymbol{x})$ by minimising an energy functional with a quadratic data term and with the subquadratic regulariser of Charbonnier et al. [29]:

$$E(u) \;=\; \int_\Omega \Big((u-f)^2 \;+\; \alpha \cdot 2\beta^2 \sqrt{1 \,+\, |\boldsymbol{\nabla} u|^2/\beta^2} \Big)\, d\boldsymbol{x}\;. \tag{3.291}$$

The number $\alpha > 0$ denotes the regularisation weight, and $\beta > 0$ is a contrast parameter. For the minimisation of $E(u)$, we consider its Euler-Lagrange equation

$$u - f \;-\; \alpha \operatorname{div}\big(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u\big) \;=\; 0 \tag{3.292}$$

with the Charbonnier diffusivity function

$$g(s^2) \;:=\; \frac{1}{\sqrt{1 \,+\, s^2/\beta^2}}\;. \tag{3.293}$$

It is also possible to obtain a solution of Eq. (3.292) as the steady state solution of the parabolic gradient descent equation

$$\partial_t u \;=\; \operatorname{div}\big(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u\big) \;+\; \frac{f-u}{\alpha}\;. \tag{3.294}$$

FED Scheme. An explicit discretisation of Eq. (3.294) with time step size $\tau > 0$ and implicitly stabilised fidelity term yields

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} \;=\; \boldsymbol{P}(\boldsymbol{u}^k)\, \boldsymbol{u}^k \;+\; \frac{\boldsymbol{f} - \boldsymbol{u}^{k+1}}{\alpha}\;. \tag{3.295}$$

It can be rewritten as

$$\boldsymbol{u}^{k+1} \;=\; \frac{\alpha\left(\boldsymbol{I} + \tau\,\boldsymbol{P}(\boldsymbol{u}^k)\right)\boldsymbol{u}^k \,+\, \tau\boldsymbol{f}}{\alpha+\tau}\,. \tag{3.296}$$

Note that this equation uses the expression $\boldsymbol{v}^{k+1} := (\boldsymbol{I} + \tau\,\boldsymbol{P}(\boldsymbol{u}^k))\,\boldsymbol{u}^k$. It can be seen as an explicit scheme for a diffusion equation without data fidelity term. Since (3.296) only performs a convex combination of $\boldsymbol{v}^{k+1}$ and $\boldsymbol{f}$, it has the same stability limit as this explicit diffusion scheme, namely $\tau_{\text{lim}} = 0.25$. With $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$ an FED version of the explicit scheme (3.296) is given by

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; \frac{\alpha\left(\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{P}(\boldsymbol{u}^k)\right)\boldsymbol{u}^{k,\,i} \,+\, \tau_i\,\boldsymbol{f}}{\alpha+\tau_i} \qquad (i = 0,\ldots,n{-}1)\,, \tag{3.297}$$

where the time step sizes τ_i are chosen according to Eq. (3.158).

Richardson's Method. Instead of a parabolic evolution, we now want to solve the Euler-Lagrange equation (3.292) by means of direct approaches. Our first approach is Richardson's method from Eq. (3.4) with varying relaxation parameters. The discretisation of Eq. (3.292) yields a nonlinear system of equations:

$$\left(\boldsymbol{I} \,-\, \alpha\,\boldsymbol{P}(\boldsymbol{u})\right)\boldsymbol{u} \;=\; \boldsymbol{f}\,. \tag{3.298}$$

It can be solved by the fixed point iteration

$$\underbrace{\left(\boldsymbol{I} \,-\, \alpha\,\boldsymbol{P}(\boldsymbol{u}^k)\right)}_{:=\boldsymbol{M}(\boldsymbol{u}^k)}\boldsymbol{u}^{k+1} \;=\; \boldsymbol{f} \qquad (k \geq 0)\,. \tag{3.299}$$

Since the system matrix $\boldsymbol{M}(\boldsymbol{u}^k)$ is symmetric and positive definite, we can apply Richardson's method: With $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$ we obtain

$$\begin{aligned}\boldsymbol{u}^{k+1,\,i+1} \;&=\; \boldsymbol{u}^{k+1,\,i} \,+\, \omega_i\Big(\boldsymbol{f} \,-\, \boldsymbol{M}(\boldsymbol{u}^k)\,\boldsymbol{u}^{k+1,\,i}\Big)\\ &=\; \Big(\boldsymbol{I} \,+\, \omega_i\,\alpha\,\boldsymbol{P}(\boldsymbol{u}^k)\Big)\,\boldsymbol{u}^{k+1,\,i} \,+\, \omega_i\left(\boldsymbol{f} \,-\, \boldsymbol{u}^{k+1,\,i}\right)\,,\end{aligned} \tag{3.300}$$

where the relaxation parameters are given by

$$\omega_i \;=\; \frac{2}{\lambda_{\max} + \lambda_{\min} \,-\, (\lambda_{\max} - \lambda_{\min})\cdot\cos\left(\pi\cdot\frac{2i+1}{2n}\right)} \qquad (i = 0,\ldots,n{-}1)\,. \tag{3.301}$$

We have $\lambda_{\min} = 1$, and with the help of Gershgorin's theorem we can estimate $\lambda_{\max}$ of $\boldsymbol{M}(\cdot)$ by $1{+}8\alpha$. After the complete cycle with length

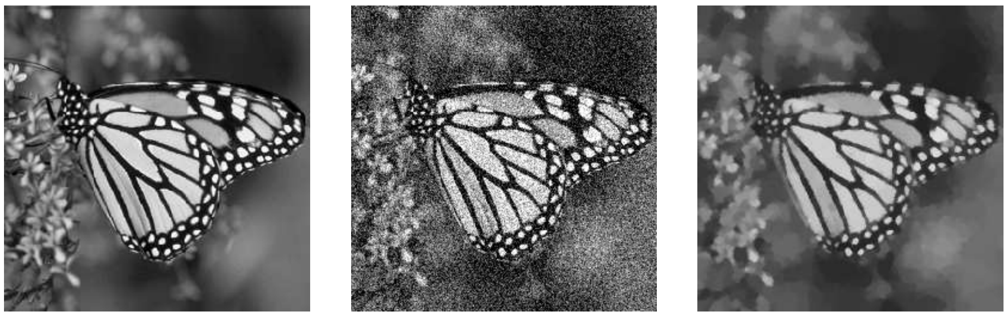


Figure 3.27: Test setting for Charbonnier regularisation. **(a) Left:** Original image (*monarch*, 256×256). **(b) Middle:** Noisy image (additive white Gaussian noise, $\sigma = 40$). **(c) Right:** Charbonnier regularisation of the noisy image with $\beta = 10^{-3}$ and $\alpha = 30000$.

n, we can set $\boldsymbol{u}^{k+1} := \boldsymbol{u}^{k+1,\,n}$ and update the nonlinearities $\boldsymbol{P}(\boldsymbol{u}^k)$ by $\boldsymbol{P}(\boldsymbol{u}^{k+1})$.

Fast-Jacobi Method. The second approach is our proposed Fast-Jacobi method. Here we get

$$\begin{aligned}\boldsymbol{u}^{k+1,\,i+1} &= \boldsymbol{u}^{k+1,\,i} + \omega_i\,\boldsymbol{D}^{-1}\Big(\boldsymbol{f} - \boldsymbol{M}(\boldsymbol{u}^k)\,\boldsymbol{u}^{k+1,\,i}\Big)\\ &= \Big(\boldsymbol{I} + \omega_i\,\alpha\,\boldsymbol{D}^{-1}\boldsymbol{P}(\boldsymbol{u}^k)\Big)\,\boldsymbol{u}^{k+1,\,i} + \omega_i\,\boldsymbol{D}^{-1}\big(\boldsymbol{f} - \boldsymbol{u}^{k+1,\,i}\big)\,,\end{aligned} \tag{3.302}$$

and we use the FED-based relaxation parameters from Eq. (3.190). Note that we can safely estimate ω_{lim} by 1 (see Sec. 3.4.4).

Our testbed is depicted in Fig. 3.27. We have degraded our test image *monarch* by additive Gaussian noise with standard deviation $\sigma = 40$. To denoise it with Charbonnier regularisation, we use the smoothness weight $\alpha = 30000$ and the contrast parameter $\beta = 10^{-3}$. The corresponding reference solution in Fig. 3.27(c) has been computed by means of the Jacobi method with 100000 iterations and nonlinear updates after each iteration.

The first experiment in Fig. 3.28 deals with the comparison of the parabolic FED scheme, Richardson's cyclic method, Jacobi, and the Fast-Jacobi approach. All approaches use a cycle length of 50. Since $\tau_{\text{lim}} = 0.25$, this corresponds to the diffusion time $T = 0.25 \cdot \frac{50\cdot 51}{3} = 212.5$ per FED cycle. As one can see in Fig. 3.28(a), the speed of convergence is significantly higher for Fast-Jacobi than for the parabolic FED approach. This shows that for elliptic problems with strongly varying coefficients, Jacobi-type methods

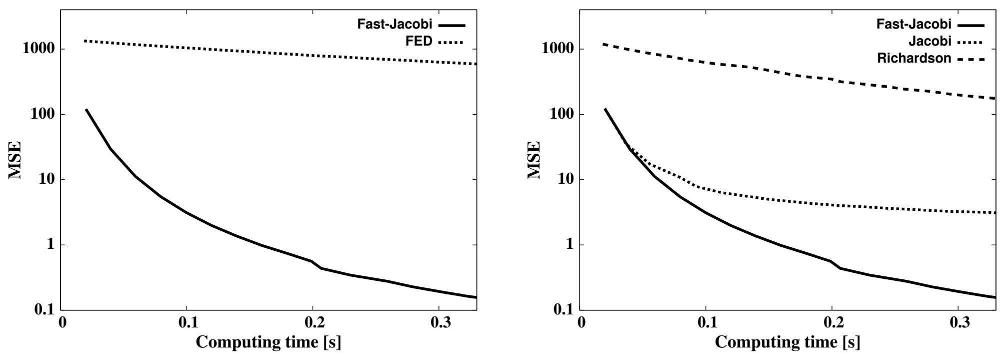
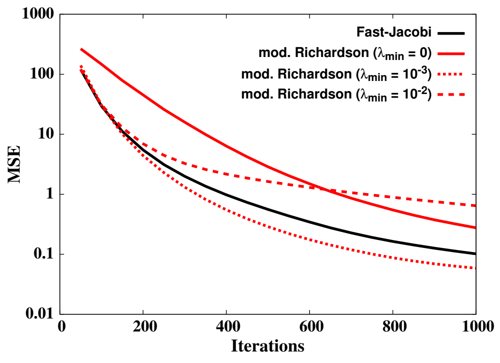


Figure 3.28: Computing time (seconds) versus MSE for FED, Richardson's method, Jacobi, and Fast-Jacobi with cycle length 50. **(a) Left:** Fast-Jacobi and FED. **(b) Right:** Fast-Jacobi, Jacobi and Richardson's method.

like Fast-Jacobi should be preferred over the parabolic FED scheme. Moreover, Fig. 3.28(b) illustrates that Fast-Jacobi is significantly better than Richardson's method which suffers from the lack of diagonal scaling that is inherent in Jacobi-like schemes. The comparison between Fast-Jacobi and Jacobi with the constant relaxation parameter $\omega = 1$ shows the usefulness of varying relaxation parameters.

For our second experiment we compare the Jacobi-type methods with the FED-based and the Richardson-based relaxation parameters, respectively. The results are given in Fig. 3.29. We have applied all methods without and with a cascadic strategy involving the three levels 64×64, 128×128 and 256×256. More precisely, we solve the problem on a coarse level and use the result as an initialisation for the next finer level. Similarly to image inpainting, this can yield a better convergence to the reference solution. If we compare, in particular, Fast-Jacobi and the modified Richardson method with an optimised damping parameter $\lambda_{\min} = 10^{-3}$, we see that Fast-Jacobi can benefit from the cascadic strategy, whereas the worse damping properties of the Richardson-based parameters deteriorate the improvement by a cascadic approach. Although the modified Richardson algorithm is still a bit better, the figure illustrates that a suboptimal choice (e.g. $\lambda_{\min} = 10^{-2}$) of the damping parameter yield results which are worse than the ones computed with the Fast-Jacobi method. Thus, we can state that the Fast-Jacobi method has a similar performance compared to the modified Richardson scheme with an optimised damping parameter. This shows the advantage of the damping parameter-free relaxation parameters that are related to FED.



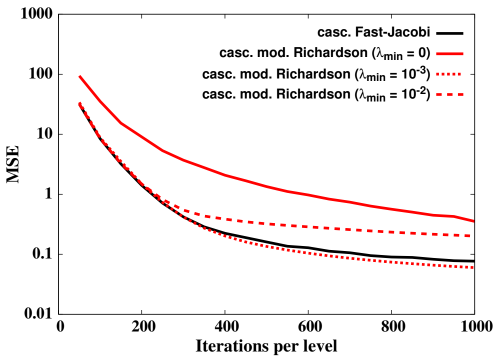


Figure 3.29: Iterations versus MSE for different Jacobi-like solvers with cycle length 50. **(a) Left:** Without a cascadic strategy. **(b) Right:** Cascadic strategy with three levels.

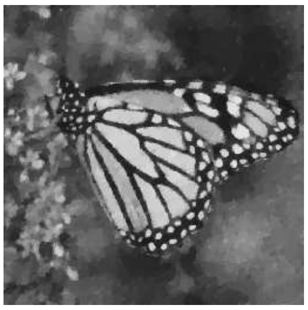
So far, we have considered a positive contrast parameter β. This is very important for the application of the above algorithms, since $\beta > 0$ implies a bounded diffusivity function in the Euler-Lagrange equation (3.292). However, we can rewrite the energy functional (3.291) by means of

$$E(u) \;=\; \int_\Omega \Big((u-f)^2 \,+\, 2\cdot\tilde{\alpha}\,\sqrt{\beta^2 \,+\, |\boldsymbol{\nabla} u|^2}\Big)\, d\boldsymbol{x}\;, \tag{3.303}$$

where we have replaced $\alpha\cdot\beta$ by a new constant $\tilde{\alpha}$. Thus, the singular case $\beta = 0$ is equivalent to total variation (TV) regularisation [114] which minimises the functional

$$E_{TV}(u) \;=\; \int_\Omega \Big((u-f)^2 \,+\, 2\cdot\tilde{\alpha}\,|\boldsymbol{\nabla} u|\Big)\, d\boldsymbol{x}\;. \tag{3.304}$$

In this case, the above algorithms can not be applied. However, there is another class of efficient algorithms that solve the TV regularisation problem, namely primal-dual methods. Recently, they have become very popular in image processing [9, 26, 27, 59] although they actually have a long history that is presented e.g. in [47]. For our comparison with Fast-Jacobi, we use the fast iterative shrinkage-thresholding algorithm (FISTA) [9, 101], because it does not require an optimisation of a parameter set. However, primal-dual methods are more complicated to implement, and converge to a solution that is different from the one with a contrast parameter $\beta > 0$. To this end, we have to compute a further reference solution that is depicted in Fig. 3.30(a). Here we have also used 100000 iterations to determine this reference result. Note that we consider the regularisation weight $\tilde{\alpha} = 30000 \cdot 10^{-3} = 30$. In order to compare the methods, we analyse the



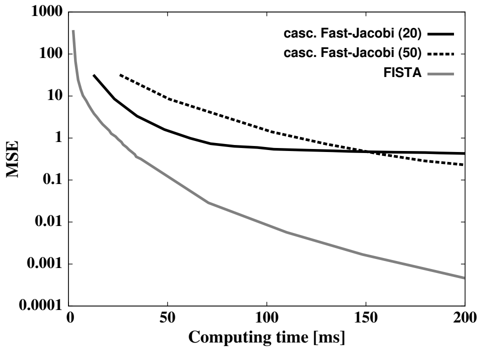


Figure 3.30: Comparison with FISTA. **(a) Left:** Reference solution for primal-dual methods with $\tilde{\alpha} = 30$. **(b) Right:** Computing time (milliseconds) versus MSE for a cascadic (three levels) Fast-Jacobi with cycle length 20 or 50, and FISTA.

trade-off between the MSE (distance to the corresponding reference) and the computing time. Although one iteration of FISTA is more expensive than a Jacobi step, FISTA benefits from a better convergence, which can be seen in Fig. 3.30(b). Even with a cascadic strategy, Fast-Jacobi can not outperform FISTA using only the original level.

3.6.5 GPU Implementations

By means of 2-D optic flow computations, it has already been demonstrated that FED is very well-suited for parallelisation on GPUs [168]. Now we illustrate that this is also the case for Fast-Jacobi in three dimensions. As an example application we have chosen range image integration, which aims at acquiring a single 3-D model from multiple range images. A range image is also referred to as depth map or 2.5-D image. It specifies the distance from the camera centre to points in the scene. Therefore it represents the visible part of the surface of a scene, which will be referred to as range surface in the remainder of this subsection. There are multiple ways how range images can be acquired. For example using devices such as the *Kinect* camera or time-of-flight cameras. By using stereo algorithms [118], range image integration can also be employed in a multiview stereo setting.

As range images will contain measurement errors in practice, integrating them is a difficult task. Curless and Levoy [36] have presented a very promising volumetric approach that can be divided into three steps. First, for each range surface the signed distance is computed within an axis aligned bounding box enclosing all range surfaces. In the second step, a cumulative

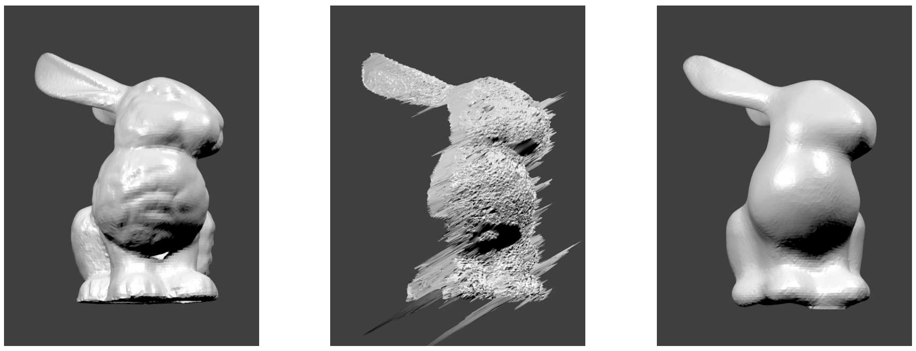


Figure 3.31: Test setting for anisotropic range image integration. **(a) Left**: One rendered image of *Stanford Bunny*. **(b) Middle**: Noisy range image. **(c) Right**: Reference reconstruction (JOR, $\omega = 0.3$, $n = 50$, 2000 cycles).

signed distance field is found by a simple averaging scheme combining all input fields. In the third and final step, the surface of the object can be found by extracting the zero level-line of the cumulative signed distance function.

Zach et al. [163] have improved this approach by computing the cumulative signed distance field as the minimiser $u : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ of a suitable energy functional:

$$E(u) \;=\; \int_\Omega \left(\sum_{i=1}^{m} w_i \, \Psi_D((u - f_i)^2) + \alpha \; \Psi_S(|\boldsymbol{\nabla} u|^2) \right) d\boldsymbol{x} \; . \tag{3.305}$$

Here $f_i : \Omega \to \mathbb{R}$ and $w_i : \Omega \to \{0, 1\}$ denote the input signed distance fields and the reliability of the measurements, respectively. The number of range images is given by m. For simplicity and to focus on the regularisation, let us choose $\Psi_D(s^2) = s^2$ and $\Psi_S(s^2) = |s|$. The corresponding Euler-Lagrange equation is given by

$$p\, u - q - \alpha \, \mathrm{div} \; (\Psi_S'(|\boldsymbol{\nabla} u|) \boldsymbol{\nabla} u) \;=\; 0 \; , \tag{3.306}$$

where the functions

$$p \;:=\; \sum_{i=1}^{m} w_i \quad \text{and} \quad q \;:=\; \sum_{i=1}^{m} w_i \cdot f_i \tag{3.307}$$

have been introduced to simplify the notation. They just denote the number of reliable measurements and the sum of the signed distance values, respectively.

In [123], the isotropic smoothing behaviour is replaced by an anisotropic one. This is achieved by modifying the Euler-Lagrange equation (3.306) accordingly:

$$p\,u - q - \alpha\,\mathrm{div}\left(\Psi'_S(\boldsymbol{J})\boldsymbol{\nabla} u\right) \;=\; 0\;. \tag{3.308}$$

Here $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ is the 3-D structure tensor and the matrix valued function Ψ'_S can be understood as an extension of a scalar valued function that is applied only to the eigenvalues of $\boldsymbol{J}$. More details can be found in [123].

After the discretisation this corresponds to a nonlinear system of equations

$$(\boldsymbol{P} - \alpha\,\boldsymbol{A}(\boldsymbol{u}))\;\boldsymbol{u} \;=\; \boldsymbol{q}\;, \tag{3.309}$$

where $\boldsymbol{P} = \mathrm{diag}(\boldsymbol{p})$ is a simple diagonal matrix. The vectors $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u} \in \mathbb{R}^N$ are obtained by discretising the functions $p(\boldsymbol{x}), q(\boldsymbol{x})$ and $u(\boldsymbol{x})$, respectively. The matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the 3-D discrete divergence operator with a diffusion tensor $\Psi'_S(\boldsymbol{J})$, where N is the number of voxels. The nonlinear system of equations can be solved in terms of a fixed-point iteration with a series of linear problems

$$\left(\boldsymbol{P} \;-\; \alpha\,\boldsymbol{A}(\boldsymbol{u}^k)\right)\;\boldsymbol{u}^{k+1} \;=\; \boldsymbol{q}\;. \tag{3.310}$$

By introducting the abbreviation

$$\boldsymbol{M}^{(k)} \;:=\; \left(\boldsymbol{P} \;-\; \alpha\,\boldsymbol{A}(\boldsymbol{u}^k)\right)\;, \tag{3.311}$$

a Fast-Jacobi cycle for computing the solution $\boldsymbol{u}^{k+1}$ can be expressed as

$$\boldsymbol{x}^{\ell+1} \;=\; \boldsymbol{x}^{\ell} \;+\; \omega_\ell\,\boldsymbol{D}^{-1}_{\boldsymbol{M}^{(k)}}\left(\boldsymbol{q} \;-\; \boldsymbol{M}^{(k)}\boldsymbol{x}^{\ell}\right) \qquad (\ell = 0,\ldots,n{-}1)\,, \tag{3.312}$$

where the relaxation parameters ω_ℓ are chosen according to Eq. (3.190), and $\boldsymbol{x}^0 := \boldsymbol{u}^k$. To get stable results, we use $\omega_{\mathrm{lim}} = {}^3\!/\!{}_{10}$.

As an experiment, we have rendered several images of the well-known *Stanford Bunny* (taken from the Stanford 3-D scanning repository) shown in Fig. 3.31(a), and used them to estimate range images based on the variational method presented by Valgaerts et al. [133]. This way, range images with a realistic amount of noise and outliers are obtained, as can be seen in Fig. 3.31(b). Although the model performs well with a huge number of range images, we have only used 15. Due to this low number of noisy range images, the reconstruction requires a high amount of smoothing, which makes it more challenging. This allows us to better illustrate the behaviour of the different solvers. Figure 3.31(c) depicts our reference reconstruction that has been computed with the JOR solver using the constant relaxation parameter $\omega = {}^3\!/\!{}_{10}$, cycle length 50 and 2000 cycles.

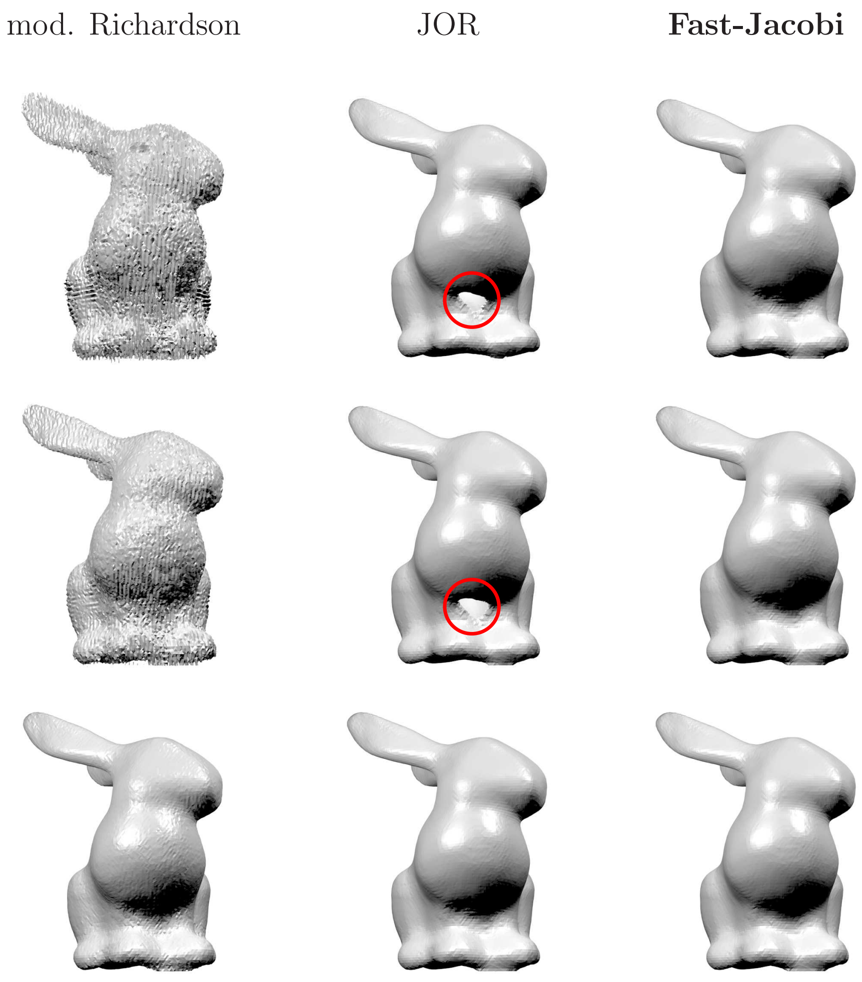


Figure 3.32: Results for the three different Jacobi-like methods with cycle length $n = 50$. **First row**: 5 cycles. **Second row**: 10 cycles. **Third row**: 100 cycles.

Figure 3.32 shows the results of the different Jacobi-like solvers with cycle length $n = 50$: The modified Richardson method, Jacobi over-relaxation (JOR) and Fast-Jacobi. As one can see, the modified Richardson method has big problems with high frequency components and thus the worst convergence to the reference reconstruction. Even with 100 cycles, it has not converged and the difference to the reference solution is clearly visible. By introducing a positive damping parameter it is likely that one can improve the results. However, as we have already seen in previous examples, the convergence can be very sensitive with respect to this parameter. The JOR solver exhibits a better convergence behaviour, but it requires more than

Table 3.12: Computing times for the sequential CPU and the parallel GPU implementation of Fast-Jacobi with 10 cycles ($n = 50$).

data size	**CPU [s]**	**GPU [s]**	**speed-up factor**
$\mathbf{64^3}$	28.16	0.34	82.8
$\mathbf{128^3}$	226.70	1.68	134.9
$\mathbf{256^3}$	1900.82	13.54	140.4

10 cycles to close the hole above the paws of the bunny (see red circles). Fast-Jacobi yields the best convergence to the reference reconstruction as it manages to close the previously mentioned hole with only 5 cycles, and is almost completely converged.

To conclude our experiment, we compare the running times of a sequential CPU and a parallel GPU implementation of the Fast-Jacobi method. This comparison is shown in Table 3.12, where we use the above mentioned 3.2 GHz *Intel Xeon* processor and a single GPU of the *NVIDIA GeForce GTX* 690, respectively. Since we are specifically interested in the solution of the nonlinear system in Eq. (3.309), the timings refer to this. Note that the number of unknowns in our example varies from $64^3 \approx 2.6 \cdot 10^5$ to $256^3 \approx 16 \cdot 10^6$, and we apply 10 cycles with length $n = 50$. As we can see in Table 3.12, the parallel implementation is up to 140 times faster, which shows that the GPU implementation is well-suited for huge data sets. The speed-up is very rewarding when considering that the parallelisation is extremely straightforward.

3.7 Summary

In this chapter we have given a detailed theoretical and experimental analysis of the explicit scheme motivated by the decomposition of the box filter: Fast Explicit Diffusion (FED). It is very easy to implement, because one can use an existing explicit method and add a few code lines. Moreover, it is well suited for parallel computing. Compared to the related known Super Time Stepping (STS) scheme, it does not need an additional damping parameter that influences the filtering results.

Since up to half of the time step sizes exceed the stability limit, the main theoretical focus has been the inner stability of the scheme. From a theoretical point of view, the natural sequence of the time step sizes allows stable intermediate solutions. However, as we have shown, this does not work in practice due to numerical rounding errors. To this end, we have recommended three different strategies for the rearrangement of the sequence that have already been used successfully in the literature to make STS and Richardson's method more robust with respect to numerical rounding errors. Unfortunately, such rearranged sequences can yield highly unstable intermediate solutions, as we have illustrated in the experimental section.

Besides these stability issues, we have analysed the convergence of the linear FED scheme, and have shown that the approximation order is one with respect to the number of FED cycles.

Furthermore, we have described the generalisation of the FED scheme to arbitrary multi-dimensional, nonlinear diffusion problems. Unfortunately, FED schemes for isotropic diffusion processes can violate the maximum-minimum principle that is valid for usual explicit, semi-implicit or additive operator splitting (AOS) methods. To solve some elliptic problems in an efficient way, we have embedded the FED scheme in a cascadic coarse-to-fine approach. Apart from a parabolic evolution, it is also possible to directly solve the elliptic equation in order to obtain the steady-state. To this end, we have proposed the so-called Fast-Jacobi method. It is a Jacobi-type solver with varying relaxation parameters that are based on the time step sizes of the FED scheme. Also in the case of Fast-Jacobi we have shown some theoretical results about the convergence.

However, when it comes to hyperbolic problems like for example the transport equation, schemes like STS or FED show their limitations. The main problem is that the stability condition for unsymmetric matrices can refer to Chebyshev polynomials with complex arguments. If the corresponding imaginary part is too large, the stability polynomials can take values with moduli larger than 1.

To confirm some theoretical results and analyse the practicability of both the FED scheme and the Fast-Jacobi method, we have presented a comprehensive experimental evaluation.

For nonlinear isotropic diffusion filtering, we have illustrated how the rearrangements with Leja ordering, κ-cycles and the strategy of Lebedev and Finogenov make the method much more robust with respect to numerical rounding errors. In this context, examples have shown how inner updates can deteriorate the final result of a cycle. Thus, we recommend to keep the nonlinearities fixed during a cycle.

When it comes to the efficient solution of isotropic parabolic problems, one usually applies the AOS scheme. However, our experiments have demonstrated that the FED scheme is much more efficient. It benefits from the omission of the splitting error. A comparison with the known

STS scheme has revealed the problems induced by the additional damping factor. On the one hand this parameter allows an optimisation, but on the other hand an optimised damping parameter again depends on other parameters such as the stopping time or the number of cycles.

For anisotropic diffusion problems, we have done a comparison between the popular semi-implicit and our FED scheme. Although the semi-implicit approach yields more accurate results with the same number of time steps (cycles), FED is faster and thus can be more efficient. This is also valid, if we embed the parabolic schemes in a cascadic coarse-to-fine approach to efficiently solve elliptic problems like for example image inpainting with edge-enhancing anisotropic diffusion. However, a cascadic Fast-Jacobi that directly solves the elliptic problem can further improve the accuracy and efficiency of cascadic FED.

Furthermore, we have applied our cyclic methods to elliptic problems with constant coefficients. As an example, we have solved an inpainting problem with the biharmonic operator. This also shows that the proposed methods are not restricted to second order PDEs: Since the discretisation of this operator yields a symmetric matrix, the transfer to such higher order PDEs is straightforward. In contrast to the anisotropic inpainting process, CFED can perform better than a cascadic Fast-Jacobi method, which means that the diagonal preconditioning is not so beneficial for elliptic problems with constant coeffcients.

However, the situation substantially changes for elliptic problems with strongly varying coefficients like e.g. Charbonnier regularisation with a small contrast parameter. In such cases, the proposed Fast-Jacobi method can provide a significantly better efficiency than a parabolic FED scheme or Richardsons's iterative method without a diagonal preconditioning. Unlike these Richardson-based approaches, the incorporation of a damping parameter is not necessary. However, as we have seen in the singular case with TV regularisation, primal-dual methods such as FISTA can provide a better convergence than Fast-Jacobi for non-singular problems.

In our last experiment, we have demonstrated the potential of a GPU implementation for the Fast-Jacobi method by means of (3-D) anisotropic range image integration. Although the parallelisation is very straightforward, we have experienced large speed-up factors up to 140 compared to a sequential CPU implementation.

Chapter 4

Recursive Fast Explicit Scheme

Study the past if you would define the future.
Confucius

So far, we have considered the decomposition of a box filter into explicit linear diffusion steps, which means that we have rewritten the box filter in terms of convolutions with stable and unstable explicit diffusion kernels. Since these unstable steps cause some problems concerning the numerical stability, we have to rearrange the sequence of all steps. In fact, this is the most crucial point.

To avoid the necessity of this rearrangement, we are going to show that there is another representation for box filters: It is based on a recursion relation. With the help of this, we propose another FED scheme that is robust against numerical rounding errors and does not require any rearrangement of time steps. Interestingly, the resulting scheme is related to a very famous class of methods for the numerical solution of differential equations: Runge-Kutta schemes.

4.1 Box Filter Recursion

In this section, we want to show that a box kernel of length $(2n{+}1)h$ can be represented as a combination of an explicit diffusion step, and two box kernels with the lengths $(2n{-}1)h$ and $(2n{-}3)h$, respectively. To this end, we derive a recursion relation for box filters.

4.1.1 Recursion Formula

Let us assume that $n \geq 2$. Given a one-dimensional signal $\boldsymbol{f} = (f_i)_{i\in\mathbb{Z}}$ on a grid with mesh size $h > 0$ that has been filtered with a box kernel of length $(2n-1)h$,

$$\left(B^h_{2n-1}(\boldsymbol{f})\right)_i \;:=\; \frac{1}{2n-1} \cdot \sum_{k=-n+1}^{n-1} f_{i+k}\ , \tag{4.1}$$

we additionally convolve it with the kernel mask $(1/2\,,\,0\,,\,1/2)$. This yields for the resulting signal $\tilde{\boldsymbol{f}}$:

$$\begin{aligned}
\tilde{f}_i \;&=\; \frac{1}{2n-1} \cdot \sum_{k=-n+1}^{n-1} \frac{1}{2}\left(f_{i+k+1} + f_{i+k-1}\right) \\
&=\; \frac{1}{4n-2} \cdot \left(\sum_{k=-n+1}^{n-1} f_{i+k+1} \;+ \sum_{k=-n+1}^{n-1} f_{i+k-1} \right) \\
&=\; \frac{1}{4n-2} \cdot \left(\sum_{k=-n+2}^{n} f_{i+k} \;+ \sum_{k=-n}^{n-2} f_{i+k} \right) \\
&=:\; \left(R^h_{2n+1}(\boldsymbol{f})\right)_i\ .
\end{aligned} \tag{4.2}$$

The corresponding filter kernel R^h_{2n+1} with length $(2n+1)h$ can be decomposed into a sum of two kernels having uniform weights with lengths $(2n+1)h$ and $(2n-3)h$, respectively:

$$\left(R^h_{2n+1}(\boldsymbol{f})\right)_i \;=\; \frac{1}{4n-2} \cdot \sum_{k=-n}^{n} f_{i+k} \;+\; \frac{1}{4n-2} \cdot \sum_{k=-n+2}^{n-2} f_{i+k}\ . \tag{4.3}$$

Since such filter kernels differ from box kernels only with respect to a multiplicative constant, we can transform the latter equation into

$$\left(R^h_{2n+1}(\boldsymbol{f})\right)_i \;=\; \frac{2n+1}{4n-2} \cdot \left(B^h_{2n+1}(\boldsymbol{f})\right)_i \;+\; \frac{2n-3}{4n-2} \cdot \left(B^h_{2n-3}(\boldsymbol{f})\right)_i\ . \tag{4.4}$$

If we solve that equation for the box kernel B^h_{2n+1}, then we obtain

$$B^h_{2n+1} \;=\; \frac{4n-2}{2n+1} \cdot R^h_{2n+1} \;-\; \frac{2n-3}{2n+1} \cdot B^h_{2n-3}\ . \tag{4.5}$$

The convolution with the mask $(1/2\,,\,0\,,\,1/2)$ corresponds to a 1-D explicit linear diffusion step with the limit time step size $\frac{h^2}{2}$, and this finally yields for $n \geq 2$:

$$B^h_{2n+1} \;=\; \alpha_n \cdot \Big(I \,+\, \tfrac{h^2}{2}\,\Delta_h\Big)\, B^h_{2n-1} \,+\, (1-\alpha_n)\cdot B^h_{2n-3}\;, \tag{4.6}$$

with $\alpha_n := \frac{4n-2}{2n+1}$. Note that Eq. (4.6) is also valid for $n = 1$, if one defines $B^h_1 = B^h_{-1} := I$. We summarise the results in the following theorem:

Theorem 4.1 (Box Filter Recursion). *A discrete 1-D box filter kernel B^h_{2n+1} fulfils the recursion relation*

$$B^h_{2n+1} \;=\; \begin{cases} I & \text{if } n < 1, \\ \alpha_n \cdot \Big(I \,+\, \tfrac{h^2}{2}\,\Delta_h\Big)\, B^h_{2n-1} \,+\, (1-\alpha_n)\cdot B^h_{2n-3} & \text{if } n \geq 1. \end{cases} \tag{4.7}$$

If we define the filtered signals

$$\boldsymbol{f}^{(k)} \;:=\; B^h_{2k+1}(\boldsymbol{f})\;, \tag{4.8}$$

we can compute them by

$$\boldsymbol{f}^{(1)} \;=\; \Big(I + \tfrac{h^2}{3}\,\Delta_h\Big)\,(\boldsymbol{f}) \qquad \text{and}$$

$$\boldsymbol{f}^{(k)} \;=\; \alpha_k \cdot \Big(I \,+\, \tfrac{h^2}{2}\,\Delta_h\Big)\,\Big(\boldsymbol{f}^{(k-1)}\Big) \,+\, (1-\alpha_k)\cdot \boldsymbol{f}^{(k-2)}\;, \tag{4.9}$$

for $k \geq 2$. To reach a box filtered signal using a filter length $(2n+1)h$, we have to perform an explicit linear diffusion step for each $k \leq n$ and simply compute a sum of two signals. At this point, we want to analyse how this relation can be useful for the improvement of FED.

4.1.2 Application to FED

In Chapter 2 we have shown that 1-D box filters can be written in terms of explicit schemes with varying time step sizes. If we assume that the filter has length $(2n+1)h$, we denote the corresponding time step sizes now by

$$\tau_i^{(n)} \;:=\; \frac{h^2}{2}\cdot\frac{1}{2\,\cos^2\Big(\pi\cdot\frac{2i+1}{4n+2}\Big)} \qquad (i = 0, ..., n{-}1)\,. \tag{4.10}$$

An FED cycle with $n \geq 1$ inner time steps for the numerical solution of the ODE system

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{A_h}\boldsymbol{u} \tag{4.11}$$

with the initial data $\boldsymbol{u}^0 \in \mathbb{R}^N$ then reads

$$\boldsymbol{u}^n \;=\; \left(\prod_{i=0}^{n-1}\Big(\boldsymbol{I} \,+\, \tau_i^{(n)}\,\boldsymbol{A_h}\Big)\right)\boldsymbol{u}^0\;. \tag{4.12}$$

We can state that the recursion relation in Eq. (4.9) implies an equivalent closed-form expression

$$\boldsymbol{u}^n \;=\; \alpha_n \cdot \Big(\boldsymbol{I} \,+\, \tfrac{h^2}{2}\,\boldsymbol{A_h}\Big)\,\boldsymbol{u}^{n-1} \;+\; (1-\alpha_n)\,\boldsymbol{u}^{n-2} \tag{4.13}$$

with $n \geq 2$. Note that for $n = 1$ we have

$$\boldsymbol{u}^1 \;=\; \Big(\boldsymbol{I} \,+\, \tfrac{h^2}{3}\,\boldsymbol{A_h}\Big)\,\boldsymbol{u}^0\;. \tag{4.14}$$

Equation (4.13) means that we can compute the result after n time steps by using the results after cycles with length $n-1$ and $n-2$, respectively. We only have to perform one explicit diffusion step that is applied to $\boldsymbol{u}^{n-1}$. The application of the recursion relation to $\boldsymbol{u}^{n-1}$, $\boldsymbol{u}^{n-2}$ and so on yields the scheme

$$\boldsymbol{u}^k \;=\; \alpha_k \cdot \Big(\boldsymbol{I} \,+\, \tfrac{h^2}{2}\,\boldsymbol{A_h}\Big)\,\boldsymbol{u}^{k-1} \;+\; (1-\alpha_k)\,\boldsymbol{u}^{k-2} \qquad (k = 2,\ldots,n)\,, \tag{4.15}$$

where the first iteration step with $k = 1$ is shown in Eq. (4.14). Compared to the cyclic FED scheme, we have the same amount of explicit steps to reach the final result $\boldsymbol{u}^n$. However, the storage of additional predecessor signals $\boldsymbol{u}^{k-2}$ and the computation of the sum of two signals is necessary for $k \geq 2$. After a complete scheme with n iteration steps, both the recursive method (4.15) and the usual FED scheme (4.12) yield the same (box filtered) result $\boldsymbol{u}^n$.

We can see that the recursive method is also well-suited for parallel computing. Besides the explicit diffusion step, we have to parallelise the computation of the sum of two vectors.

4.1.3 Recursion with the Symbol

Before we finish this section, we want to present another way for the derivation of the recursion relation in Eq. (4.6). To this end, we will use the symbol

$p_B^{[n]}(z)$ of a box filter B_{2n+1}^h and compute a recursion formula. We know that the symbol is related to a Chebyshev polynomial of second kind,

$$p_B^{[n]}(z) \;=\; \frac{1}{2n+1} \cdot U_{2n}\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right) \;, \tag{4.16}$$

where $z \in \left[0\,,\,\frac{4}{h^2}\right]$. It is a well-known result that Chebyshev polynomials satisfy the relation

$$T_m(x) \cdot U_k(x) \;=\; \frac{1}{2} \cdot \left(U_{k+m}(x) \,+\, U_{k-m}(x)\right) \;, \tag{4.17}$$

or in particular for $n \leq 2$

$$\begin{aligned} & T_2\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right) \cdot U_{2(n-1)}\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right) \\ =\; & \left(1-\tfrac{h^2}{2}\,z\right) \cdot U_{2(n-1)}\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right) \\ =\; & \frac{1}{2} \cdot \left(U_{2n}\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right) + U_{2(n-2)}\left(\sqrt{1-\tfrac{h^2}{4}\,z}\right)\right) \;. \end{aligned} \tag{4.18}$$

Thus, we obtain

$$\left(1-\tfrac{h^2}{2}\,z\right) \cdot p_B^{[n-1]}(z) \;=\; \frac{1}{2} \cdot \left(\frac{2n+1}{2n-1} \cdot p_B^{[n]}(z) + \frac{2n-3}{2n-1} \cdot p_B^{[n-2]}(z)\right) \;, \tag{4.19}$$

which can be written as the recursion formula

$$p_B^{[n]}(z) \;=\; \frac{4n-2}{2n+1} \cdot \left(1-\tfrac{h^2}{2}\,z\right) \cdot p_B^{[n-1]}(z) \;-\; \frac{2n-3}{2n+1} \cdot p_B^{[n-2]}(z) \;. \tag{4.20}$$

Since the multiplication with $\left(1-\tfrac{h^2}{2}\,z\right)$ corresponds to an explicit linear diffusion step with $\tau \,=\, \frac{h^2}{2}$ and the symbols are, according to Proposition 2.2, unique, it immediately follows that Eq. (4.20) is equivalent to the recursion relation (4.6).

4.2 Connection to Runge-Kutta Schemes

In this section, we want to show that the recursive FED scheme (4.13) is related to the well-known class of the *Runge-Kutta methods* [19, 70]. To this end, we shortly describe the basic idea of Runge-Kutta methods and give a definition for the explicit case.

Let $M \subset \mathbb{R} \times \mathbb{R}^d$, $d \in \mathbb{N}$, be an open set. We consider the initial value problem with the function $\boldsymbol{g} : M \to \mathbb{R}^d$,

$$\boldsymbol{y}'(x) \;=\; \boldsymbol{g}(x, \boldsymbol{y}(x)) \;, \quad \boldsymbol{y}(x_0) = \boldsymbol{y_0} \in \mathbb{R}^d \;, \tag{4.21}$$

where we look for the unknown solution $\boldsymbol{y} : [x_0, \infty) \to \mathbb{R}^d$. If the function $\boldsymbol{g}$ depends only on x, we can write the solution $\boldsymbol{y}$ in terms of

$$\boldsymbol{y}(x_1) \;=\; \boldsymbol{y_0} \;+\; \int\limits_{x_0}^{x_1} \boldsymbol{g}(x)\, dx \;, \tag{4.22}$$

with $x_1 > x_0$. The integral can be approximated, for example, with the help of the midpoint rule, which means

$$\boldsymbol{y}(x_1) \;\approx\; \boldsymbol{y_0} \;+\; \underbrace{(x_1 \;-\; x_0)}_{:=\Theta} \cdot \boldsymbol{g}\left(\tfrac{x_1 + x_0}{2}\right) \;. \tag{4.23}$$

For the above initial value problem, we would get

$$\boldsymbol{y_1} \;:=\; \boldsymbol{y}(x_1) \;\approx\; \boldsymbol{y_0} \;+\; \Theta \cdot \boldsymbol{g}\left(\tfrac{x_1 + x_0}{2} \,,\, \boldsymbol{y}\left(\tfrac{x_1 + x_0}{2}\right)\right) \;, \tag{4.24}$$

with the unknown vector $\boldsymbol{y}\big(\frac{1}{2} \cdot (x_1 \,+\, x_0)\big)$. However, with the help of its Taylor expansion

$$\boldsymbol{y}\left(\tfrac{x_1 + x_0}{2}\right) \;=\; \boldsymbol{y}(x_0) \;+\; \underbrace{\boldsymbol{y}'(x_0)}_{=\boldsymbol{g}(x_0, \boldsymbol{y}(x_0))} \;\cdot\; \frac{\Theta}{2} \;+\; \mathcal{O}\left(\Theta^2\right) \tag{4.25}$$

we can approximate it by using an explicit Euler step

$$\boldsymbol{y}\left(\tfrac{x_1 + x_0}{2}\right) \;\approx\; \boldsymbol{y}(x_0) \;+\; \frac{\Theta}{2} \cdot \boldsymbol{g}\left(x_0, \boldsymbol{y}(x_0)\right) \;. \tag{4.26}$$

Thus, we end up with the method

$$\begin{cases} \boldsymbol{k_1} \;=\; \boldsymbol{g}\left(x_0, \boldsymbol{y_0}\right) \\ \boldsymbol{k_2} \;=\; \boldsymbol{g}\left(x_0 \,+\, \frac{\Theta}{2} \,,\, \boldsymbol{y_0} \,+\, \frac{\Theta}{2}\, \boldsymbol{k_1}\right) \\ \boldsymbol{y_1} \;=\; \boldsymbol{y_0} \;+\; \Theta \cdot \boldsymbol{k_2} \;. \end{cases} \tag{4.27}$$

This is an *explicit Runge-Kutta scheme* using two stages. Since the midpoint rule is a better approximation (higher order) than an Euler step, this method yields better results in general. By using more accurate quadrature formulas, the number of stages can increase and the results can be further improved. We now give a general definition for such schemes:

Definition 4.2. *(cf. [70]) Let $s \in \mathbb{N}$ be the number of stages and $a_{2,1}$, $a_{3,1}$, $a_{3,2}, \ldots,$ $a_{s,1}, \ldots,$ $a_{s,s-1}$, $b_1, \ldots, b_s$, $c_2, \ldots, c_s$ be real-valued coefficients. Then the method with step size $\Theta > 0$ given by*

$$\begin{cases} \boldsymbol{k_1} = \boldsymbol{g}\,(x_0, \boldsymbol{y_0}) \\ \boldsymbol{k_2} = \boldsymbol{g}\big(x_0 + c_2 \cdot \Theta, \boldsymbol{y_0} + \Theta \cdot a_{2,1}\boldsymbol{k_1}\big) \\ \boldsymbol{k_3} = \boldsymbol{g}\big(x_0 + c_3 \cdot \Theta, \boldsymbol{y_0} + \Theta \cdot (a_{3,1}\boldsymbol{k_1} + a_{3,2}\boldsymbol{k_2})\big) \\ \quad\vdots \\ \boldsymbol{k_s} = \boldsymbol{g}\left(x_0 + c_s \cdot \Theta, \boldsymbol{y_0} + \Theta \cdot \sum\limits_{i=1}^{s-1} a_{s,i}\,\boldsymbol{k_i}\right) \\ \boldsymbol{y_1} = \boldsymbol{y_0} + \Theta \cdot \sum\limits_{i=1}^{s} b_i \boldsymbol{k_i} \end{cases} \tag{4.28}$$

*is called an **s-stage explicit Runge-Kutta method**.*

Usually, the coefficients are chosen such that

$$c_i = \sum_{j=1}^{i-1} a_{i,j} \,, \tag{4.29}$$

and

$$\sum_{i=1}^{s} b_i = 1 \,. \tag{4.30}$$

More details about (explicit) Runge-Kutta methods can be found, for instance, in [2, 19, 70, 71].

In the case of the ODE system (4.11), we have the so-called *integration steps*

$$\boldsymbol{k_1} = \boldsymbol{g}\,(x_0, \boldsymbol{y_0}) = \boldsymbol{A_h}\boldsymbol{y_0} \,, \tag{4.31}$$

as well as

$$\begin{aligned} \boldsymbol{k_2} &= \boldsymbol{A_h}\boldsymbol{y_0} + \boldsymbol{A_h}\,(\Theta \cdot a_{2,1}\boldsymbol{k_1}) \\ &= \boldsymbol{A_h}\boldsymbol{y_0} + \Theta \cdot a_{2,1}\,\boldsymbol{A_h^2}\boldsymbol{y_0} \,, \end{aligned} \tag{4.32}$$

and in general

$$\boldsymbol{k_i} = \underbrace{\left(\sum_{\ell=1}^{i} \beta_\ell^{(i)} \cdot \boldsymbol{A_h^\ell}\right)}_{=:\, q_i(\boldsymbol{A_h})} \boldsymbol{y_0} \,, \tag{4.33}$$

with real-valued coefficients $\beta_\ell^{(i)}$. Actually, $\boldsymbol{k_i}$ is a matrix-vector multiplication with a matrix polynomial $q_i(\boldsymbol{A_h})$ of degree i and the vector $\boldsymbol{y_0}$. Since the matrix polynomials are linearly independent, the set $\{q_1, q_2, \ldots, q_s\}$ is a basis, and we can choose unique coefficients $b_1, \ldots, b_s$ that satisfy

$$\begin{aligned}\boldsymbol{y_1} &= \boldsymbol{y_0} + \Theta \cdot \sum_{i=1}^{s} b_i\, \boldsymbol{k_i} \\ &= \left(\boldsymbol{I} + \sum_{\ell=1}^{s} \frac{h^{2\ell}}{2\ell+1}\binom{s+\ell}{2\ell}\boldsymbol{A_h^\ell}\right)\boldsymbol{y_0}\,. \qquad (4.34)\end{aligned}$$

Thus, the parameters of the $\boldsymbol{s}$-stage explicit Runge-Kutta method can be tuned such that it is equivalent to an FED cycle with length s. However, the determination of the parameters can become complicated. To this end, we consider a scheme that is based on the above recursion relation:

$$\left\{\begin{aligned}\boldsymbol{k_1} &= \boldsymbol{y_0} + \tau\cdot\alpha_1\cdot\tfrac{3}{s(s+1)}\,\boldsymbol{g}\left(x_0, \boldsymbol{y_0}\right) \\ \boldsymbol{k_2} &= \alpha_2\cdot\Big(\boldsymbol{k_1} + \tau\cdot\tfrac{3}{s(s+1)}\,\boldsymbol{g}\left(x_0 + c_2\cdot\tau, \boldsymbol{k_1}\right)\Big) + (1-\alpha_2)\cdot\boldsymbol{y_0} \\ &\;\vdots \\ \boldsymbol{k_s} &= \alpha_s\cdot\Big(\boldsymbol{k_{s-1}} + \tau\cdot\tfrac{3}{s(s+1)}\,\boldsymbol{g}\left(x_0 + c_s\cdot\tau, \boldsymbol{k_{s-1}}\right)\Big) \\ &\quad + (1-\alpha_s)\cdot\boldsymbol{k_{s-2}} \\ \boldsymbol{y_1} &= \boldsymbol{k_s}\,,\end{aligned}\right. \qquad (4.35)$$

where $\tau > 0$ is the total step size of the method, similar to $\Theta > 0$ in the definition. The parameters α_i are already known from the box filter recursion, i.e.

$$\alpha_i = \frac{4i-2}{2i+1} \qquad (i \geq 1)\,. \qquad (4.36)$$

Furthermore, we have $c_0 = c_1 = 0$ and

$$c_i = \alpha_{i-1}\cdot\left(c_{i-1} + \frac{3}{s(s+1)}\right) + (1-\alpha_{i-1})\cdot c_{i-2}\,, \qquad (4.37)$$

with $i \geq 2$. This yields the points in time $c_1\cdot\tau, \ldots, c_s\cdot\tau$, where the function $\boldsymbol{g}$ has to be evaluated. If we apply the above method to the ODE system in Eq. (4.11), i.e. $\boldsymbol{g}(x, \boldsymbol{y}) = \boldsymbol{A_h y}$, with the time step size $\tau = \frac{h^2}{6}\cdot s(s+1)$, then we obtain, due to the recursion relation,

$$\boldsymbol{k_i} = \left(\boldsymbol{I} + \sum_{\ell=1}^{i} \frac{h^{2\ell}}{2\ell+1}\binom{i+\ell}{2\ell}\boldsymbol{A_h^\ell}\right)\boldsymbol{y_0} \qquad (i \leq s). \qquad (4.38)$$

Thus, $\boldsymbol{k_i}$ corresponds to the result of an FED cycle with length i.

4.2.1 Stability Analysis

For the usual FED scheme, we have seen that large cycle lengths lead to numerical instabilities. This is due to the fact that FED uses unstable inner time steps. On the other hand, explicit Runge-Kutta methods with many stages can also suffer from numerical instabilities. To this end, we analyse the stability of the scheme (4.35) with $\boldsymbol{g}(x, \boldsymbol{y}) = \boldsymbol{A_h y}$, and focus in particular on the internal stability of all stages, because the final result $\boldsymbol{y_1} = \boldsymbol{k_s}$ is already stable. According to van der Houwen and Sommeijer [137], we consider the roots of the equations

$$\begin{aligned} \xi \;&=\; 1 - \alpha_1 \cdot \tfrac{3}{s(s+1)} \cdot z \\ \xi^i \;&=\; \alpha_i \cdot \Big(1 - \tfrac{3}{s(s+1)} \cdot z\Big) \cdot \xi^{i-1} + (1-\alpha_i) \cdot \xi^{i-2} \quad (s \geq i \geq 2)\,, \end{aligned} \tag{4.39}$$

where $z \in [0\,, \tau \cdot \frac{4}{h^2}]$ reflects the eigenvalues of $-\tau \cdot \boldsymbol{A_h}$. Actually, we have replaced the internal results $\boldsymbol{k_i}$ by a complex-valued number ξ^i. If the roots of the above equations are bounded in absolute value by 1, then each integration step of the *FED Runge-Kutta method* (4.35) is stable. Now we show that this is the case for suitable $\tau > 0$:

Proposition 4.3 (Location of the Roots). *If the step size $\tau > 0$ satisfies*

$$\tau \;\leq\; \frac{h^2}{2} \cdot \frac{s(s+1)}{3}\,, \tag{4.40}$$

then the roots of Eq. (4.39) lie within the complex unit circle $\mathbb{D} := \{w \in \mathbb{C} \mid |w| \leq 1\}$.

Proof. For $i = 1$ we have $\alpha_1 = 2/3$ and because of

$$z \;\leq\; \tau \cdot \frac{4}{h^2} \;\leq\; \frac{2}{3} \cdot s(s+1)\,, \tag{4.41}$$

it holds that

$$\alpha_1 \cdot \frac{3}{s(s+1)} \cdot z \;\leq\; \frac{4}{3}\,. \tag{4.42}$$

Thus, $-\frac{1}{3} \leq \xi \leq 1$.

Let us now consider the case $i \geq 2$, i.e. we have to solve

$$\xi^{i-2} \cdot \Big(\xi^2 - \alpha_i \cdot \Big(1 - \tfrac{3}{s(s+1)}\, z\Big) \cdot \xi + \alpha_i - 1\Big) \;=\; 0\,. \tag{4.43}$$

Besides $\xi_3 = 0$, we get the two solutions

$$\xi_{1,2} \;=\; \frac{\alpha_i \cdot r(z)}{2} \;\pm\; \sqrt{\frac{\left(\alpha_i \cdot r(z)\right)^2}{4} \;-\; \alpha_i \;+\; 1}\;, \tag{4.44}$$

with $r(z) \;:=\; 1 - \frac{3}{s(s+1)}\,z$. Now we assume that $\tau \;\le\; \frac{h^2}{12} \cdot s(s+1)$ or equivalently $z \le \frac{1}{3} \cdot s(s+1)$, which means $r(z) \in [0,1]$. Since $i \ge 2$, the coefficient α_i fulfils

$$\frac{6}{5} \;\le\; \alpha_i \le 2\;. \tag{4.45}$$

Let us first assume that the roots are complex-valued, i.e. $\xi_1 \;=\; \bar{\xi_2}$, where the bar denotes the complex conjugate. We obtain

$$\xi_1 \cdot \xi_2 \;=\; |\xi_1|^2 \;=\; |\xi_2|^2 \;=\; \alpha_i \;-\; 1 \;<\; 1\;, \tag{4.46}$$

and hence $\xi_1, \xi_2 \in \mathbb{D}$.
For real-valued ξ_1, ξ_2 we have, due to the non-negativity of the square root in Eq. (4.44), $r(z) \;\ge\; 2 \cdot \frac{\sqrt{\alpha_i - 1}}{\alpha_i} > 0$ and

$$\begin{aligned}
\sqrt{\frac{\left(\alpha_i \cdot r(z)\right)^2}{4} \;-\; \alpha_i \;+\; 1} \quad &= \quad \frac{r(z)}{2} \cdot \sqrt{\alpha_i^2 \;-\; \frac{4\,\alpha_i}{r^2(z)} \;+\; \frac{4}{r^2(z)}} \\
&\overset{r(z)\,\le\,1}{\le} \quad \frac{r(z)}{2} \cdot \sqrt{\alpha_i^2 \;-\; \frac{4\,\alpha_i}{r(z)} \;+\; \frac{4}{r^2(z)}} \\
&= \quad \frac{r(z)}{2} \cdot \left(\frac{2}{r(z)} \;-\; \alpha_i\right) \\
&= \quad 1 \;-\; \frac{\alpha_i \cdot r(z)}{2}\;.
\end{aligned} \tag{4.47}$$

Combining this result with Eq. (4.44) yields

$$0 \;<\; \frac{\alpha_i \cdot r(z)}{2} \;\le\; \xi_1 \;\le\; 1 \tag{4.48}$$

and

$$-1 \;<\; \alpha_i \cdot r(z) - 1 \;\le\; \xi_2 \;\le\; \frac{\alpha_i \cdot r(z)}{2} \;<\; 1\;. \tag{4.49}$$

Because of $\xi_1 \cdot \xi_2 = \alpha_i - 1 > 0$ and $\xi_1 > 0$, we can even state that $\xi_2 > 0$. Overall, we have $\xi_1, \xi_2 \in \mathbb{D}$.

The case $\tau > \frac{h^2}{12} \cdot s(s+1)$, which implies $r(z) \in [-1, 0)$, can be discussed analogously, because only the signs of both ξ_1 and ξ_2 change. □

As mentioned above, we can formulate the following stability theorem by means of Proposition 4.3 and [137]:

Theorem 4.4 (**Inner stability of the Runge-Kutta method** (4.35))**.** *Let the time step size* $\tau > 0$ *fulfil the inequality* (4.40)*. Then each integration step of the FED Runge-Kutta scheme* (4.35) *is stable.*

This theorem is a stronger statement than Proposition 3.2 about the inner stability of the usual FED scheme. From a theoretical point of view, both methods yield stable intermediate results, provided that FED uses the unstable time steps in their natural order. However, in contrast to the FED scheme, the Runge-Kutta method uses only stable time integration steps. This means that we have

$$\|\boldsymbol{y_1}\|_2 \;=\; \|\boldsymbol{k_s}\|_2 \;\le\; \|\boldsymbol{k_{s-1}}\|_2 \;\le\; \dots \;\le\; \|\boldsymbol{k_1}\|_2 \;\le\; \|\boldsymbol{y_0}\|_2\;. \tag{4.50}$$

Therefore, it is robust against numerical instabilities and we do not have to take care of the sequence of the time steps. Using the notation

$$\boldsymbol{y_1}^{(i)} \;:=\; \left(\prod_{\ell=0}^{i-1}\left(\boldsymbol{I} \,+\, \tau_\ell^{(s)}\,\boldsymbol{A_h}\right)\right)\boldsymbol{y_0} \qquad (i \le s), \tag{4.51}$$

with the time step sizes $\tau_\ell^{(s)}$ from Eq. (4.10), Prop. 3.2 only guarantees that the inner results $\boldsymbol{y_1}^{(i)}$ of the cyclic scheme satisfy

$$\left\|\boldsymbol{y_1}^{(i)}\right\|_2 \;\le\; \|\boldsymbol{y_0}\|_2 \qquad (i \le s). \tag{4.52}$$

The unstable steps can produce an increasing norm, i.e. there might exist indices i_0 with

$$\left\|\boldsymbol{y_1}^{(i_0+1)}\right\|_2 \;>\; \left\|\boldsymbol{y_1}^{(i_0)}\right\|_2\;. \tag{4.53}$$

This makes the cyclic method sensitive to numerical rounding errors, as we have shown in the last chapter.

4.2.2 Related Work

We know that the symbol or amplification factor of the Super Time Stepping method is a Chebyshev polynomial of the first kind. Because of their recursive structure, it is also possible to construct a corresponding recursive Runge-Kutta method that provides internal stability. They were developed by P. J. van der Houwen and B. P. Sommeijer in 1980 under the name *Runge-Kutta-Chebyshev schemes* [137], and allow to update possible nonlinearities after each integration step. As mentioned above, their derivation is based on the amplification factor of STS, but for the 1-D linear case, this scheme can also be seen as a recursive construction of the linear filter kernels $V_{\cdot}^{h}$ (with maximum variance) that we have already introduced and discussed in the previous chapters. Unfortunately, the method still needs a damping parameter $\nu > 0$ that influences the numerical results. Besides the above publication, van der Houwen, Sommeijer and their colleagues have done a lot of research in this field, which can be found, for example, in [71, 127, 134, 135, 140, 141]. A nice historical overview about the development of explicit Runge-Kutta methods is given in [136].

In [137], the usual Super Time Stepping method is interpreted as a Runge-Kutta method, namely a so-called *factorized scheme.* Thus, the usual FED method can also be seen as a factorized Runge-Kutta scheme.

4.2.3 Extension to Arbitrary Diffusion Problems

According to Sec. 3.2.5, we consider the nonlinear time-dependent system of ODEs

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{P}(\boldsymbol{u})\,\boldsymbol{u}\,, \tag{4.54}$$

where $\boldsymbol{P}(\cdot) \in \mathbb{R}^{N\times N}$ is an arbitrary symmetric, negative semi-definite matrix. Let us assume that we know the time step size limit τ_{lim} of the usual explicit scheme

$$\boldsymbol{u}^{k} \;=\; \big(\boldsymbol{I} \,+\, \tau\,\boldsymbol{P}\big(\boldsymbol{u}^{k-1}\big)\big)\,\boldsymbol{u}^{k-1} \qquad (k \geq 1)\,. \tag{4.55}$$

In the FED Runge-Kutta scheme (4.35) we already use a framework allowing nonlinear functions $\boldsymbol{g}$. To compute a numerical solution of the ODE system (4.54), we set

$$\boldsymbol{g}\,(t, \boldsymbol{v}) \;:=\; \boldsymbol{P}(\boldsymbol{v})\,\boldsymbol{v}\,, \tag{4.56}$$

where $\boldsymbol{v} = \boldsymbol{v}(t)$ denotes an approximation of $\boldsymbol{u}(t)$. If we replace the stability condition in Eq. (4.40) by

$$\tau \;\leq\; \tau_{\text{lim}} \cdot \frac{s(s+1)}{3}\,, \tag{4.57}$$

then we have $r(z) \in [-1, 1]$ (see the proof of Prop. 4.3). Thus, we can guarantee stability in the Euclidean norm for the corresponding FED Runge-Kutta scheme with s stages and a total time step size τ satisfying Eq. (4.57).

Although we are now able to perform inner updates with respect to nonlinear problems [137], it is recommendable to use the FED Runge-Kutta scheme in an iterative manner instead of only increasing the number of stages s. Similar to the box filter or the cyclic FED scheme, we have first order consistency, and increasing the number of outer iterations significantly improves the accuracy. In particular for nonlinear problems, the distance of the points in time where the nonlinearities have to be evaluated significantly grows with s. Thus, only increasing the number of stages can not improve the results. According to (4.35), the k-th outer iteration step of the FED Runge-Kutta scheme reads

$$\begin{aligned} \boldsymbol{u}^{k,1} &= \big(\boldsymbol{I} + \tfrac{2}{3}\cdot\tilde{\tau}\,\boldsymbol{P}\big(\boldsymbol{u}^{k,0}\big)\big)\boldsymbol{u}^{k,0}\,, \\ \boldsymbol{u}^{k,i} &= \alpha_i\cdot\big(\boldsymbol{I} + \tilde{\tau}\,\boldsymbol{P}\big(\boldsymbol{u}^{k,i-1}\big)\big)\,\boldsymbol{u}^{k,i-1} \\ &\qquad + (1-\alpha_i)\cdot\boldsymbol{u}^{k,i-2} \qquad (i=2,\dots,s)\,, \end{aligned} \tag{4.58}$$

where $\tilde{\tau} := \frac{3}{s(s+1)}\cdot\tau$ and $\boldsymbol{u}^{k,0} := \boldsymbol{u}^{k-1}$. It results in $\boldsymbol{u}^{k} := \boldsymbol{u}^{k,s}$.

4.2.4 Predictor-Corrector Scheme

If we reconsider the Runge-Kutta scheme (4.27) that is based on the midpoint rule, we can see that the idea is related to predictor-corrector strategies [40]. Actually, the first step is the computation of an intermediate solution, the so-called predictor. The second step, i.e. the corrector step, uses the nonlinearities corresponding to the predictor in order to yield a more accurate result.

Regarding the proposed method, this means we can use the FED Runge-Kutta method (4.58) to a compute a predictor, and to perform a corrector step. More precisely, given $\boldsymbol{u}^{k-1} \approx \boldsymbol{u}(t_0)$ and the total time step size $\tau > 0$, we use (4.58) with a suitable number of stages to reach the diffusion time $t_0 + \frac{\tau}{2}$. This yields a predictor $\boldsymbol{u}^{k-1/2}$, and we evaluate the matrix $\boldsymbol{P}\big(\boldsymbol{u}^{k-1/2}\big)$. For the corrector step, we apply a modified version of (4.58), which means that we keep the nonlinearities fixed and replace $\boldsymbol{P}\big(\boldsymbol{u}^{k,i-1}\big)$ for all $i = 1,\dots,s$ by $\boldsymbol{P}\big(\boldsymbol{u}^{k-1/2}\big)$.

Although the predictor-corrector strategy does not increase the first order consistency, we will see that it can improve the efficiency.

4.2.5 Implementation

The implementation of the FED Runge-Kutta (FEDRK) schemes needs a bit more effort than the usual FED scheme. In addition to an explicit time

1. **Input Data**:
   image f, stopping time T, number M of outer FED Runge-Kutta steps, and explicit step size limit τ_{lim}

2. **Initialisation**:

   (a) Compute the minimum number of stages s, such that the step size $\tau = \frac{T}{M}$ satisfies the condition (4.57).

   (b) If the diffusivity or diffusion tensor is constant in time, compute the corresponding matrix $\boldsymbol{P}$.

3. **Filtering Loop**: $(k = 1, \ldots, M)$

   – Apply the scheme (4.58). For a constant diffusivity or diffusion tensor, one simply has to replace $\boldsymbol{P}(\cdot)$ by a constant matrix $\boldsymbol{P}$. If the problem is time-variant, it is also possible to use the above-mentioned predictor-corrector strategy.

Figure 4.1: General FED Runge-Kutta algorithm for parabolic problems.

step, we have to store one more predecessor result, and compute a sum of vectors. However, the computation and rearrangement of the varying time step sizes is not necessary anymore. Moreover, in contrast to the FED scheme, we can update the nonlinearities on each stage or inner step. In the experimental section, we will analyse different update strategies, including the predictor-corrector method. A summary of the algorithm is given in Fig. 4.1.

Given the stopping time T and the desired number M of outer Runge-Kutta time steps, the corresponding number of stages s can be computed similarly to the cycle length from Chapter 3 via

$$s = \left\lceil \sqrt{\frac{1}{\tau_{\text{lim}}} \cdot \frac{3\,T}{M} + \frac{1}{4}} - \frac{1}{2} \right\rceil . \tag{4.59}$$

4.3 Semi-Iterative Methods for Linear Systems

The idea for the construction of Runge-Kutta-Chebyshev schemes comes from the so-called *Chebyshev semi-iterative method* or *Richardson's method* *of second degree* for the solution of linear systems [52, 60, 138, 161]. Such semi-iterative methods have also been used e.g. to accelerate the *Landweber iteration* [87] in the context of ill-posed problems [72].

Let us reconsider the homogeneous linear system

$$\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{0} \tag{4.60}$$

with a symmetric, positive definite system matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ and the zero vector $\boldsymbol{0} \in \mathbb{R}^N$. If the eigenvalues of the matrix $\boldsymbol{B}$ range in $[\lambda_{\min}, \lambda_{\max}]$, $\lambda_{\max} > \lambda_{\min} > 0$, and we apply Richardson's cyclic method with the relaxation parameters ω_i from Eq. (3.17), then

$$\boldsymbol{x}^{j+1} \;=\; S_{j+1}(\boldsymbol{B})\,\boldsymbol{x}^0 \;, \tag{4.61}$$

where $j+1$ is the length of the cycle, $\boldsymbol{x}^0 \in \mathbb{R}^N$ an arbitrary initial vector and $S_{j+1}(\cdot)$ a modified Chebyshev polynomial with degree $j+1$. Actually, we have constructed this polynomial with the help of linear factors, where each of the factors corresponds to a step of Richardson's method with a certain relaxation parameter:

$$S_{j+1}(\boldsymbol{B}) \;=\; \prod_{i=0}^{j} (\boldsymbol{I} \,-\, \omega_i\,\boldsymbol{B}) \;. \tag{4.62}$$

The idea of semi-iterative methods is to avoid the decomposition into linear factors or steps of Richardson's method by using the recurrence relation of the Chebyshev polynomials $S_{j+1}(\cdot)$. From Sec. 3.1, we know that

$$S_{j+1}(z) \;=\; \frac{T_{j+1}\left(\mu \,-\, \frac{2z}{\lambda_{\max}-\lambda_{\min}}\right)}{T_{j+1}\left(\mu\right)} \;, \tag{4.63}$$

where

$$\mu \;:=\; \frac{\lambda_{\max} \,+\, \lambda_{\min}}{\lambda_{\max} \,-\, \lambda_{\min}} \;. \tag{4.64}$$

Thus, we have

$$\begin{aligned}
S_{j+1}(z) \;&=\; \frac{2\Big(\mu - \frac{2z}{\lambda_{\max}-\lambda_{\min}}\Big)\, T_j\Big(\mu - \frac{2z}{\lambda_{\max}-\lambda_{\min}}\Big) \;-\; T_{j-1}\Big(\mu \;-\; \frac{2z}{\lambda_{\max}-\lambda_{\min}}\Big)}{T_{j+1}\left(\mu\right)} \\
&=\; 2\left(\mu \;-\; \frac{2z}{\lambda_{\max}-\lambda_{\min}}\right) \cdot \frac{T_j\left(\mu\right)}{T_{j+1}\left(\mu\right)} \cdot S_j(z) \;-\; \frac{T_{j-1}\left(\mu\right)}{T_{j+1}\left(\mu\right)} \cdot S_{j-1}(z) \\
&=\; 2\mu \cdot \frac{T_j\left(\mu\right)}{T_{j+1}\left(\mu\right)} \left(1 \;-\; \frac{2z}{\lambda_{\max}+\lambda_{\min}}\right) \cdot S_j(z) \\
&\quad -\; \frac{2\mu \cdot T_j\left(\mu\right) \;-\; T_{j+1}\left(\mu\right)}{T_{j+1}\left(\mu\right)} \cdot S_{j-1}(z) \\
&=\; d_j \cdot \left(1 \;-\; \frac{2}{\lambda_{\max}+\lambda_{\min}}\, z\right) \cdot S_j(z) \;+\; (1-d_j) \cdot S_{j-1}(z)\;, \qquad (4.65)
\end{aligned}$$

with the coefficients

$$d_j \;:=\; 2\mu \cdot \frac{T_j\left(\mu\right)}{T_{j+1}\left(\mu\right)}\;. \qquad (4.66)$$

Putting this result into Eq. (4.61) yields

$$\begin{aligned}
\boldsymbol{x}^{j+1} \;&=\; d_j \cdot \left(\boldsymbol{I} \;-\; \frac{2}{\lambda_{\max}+\lambda_{\min}}\,\boldsymbol{B}\right) \cdot S_j(\boldsymbol{B})\,\boldsymbol{x}^0 \;+\; (1-d_j) \cdot S_{j-1}(\boldsymbol{B})\,\boldsymbol{x}^0 \\
&=\; d_j \cdot \left(\boldsymbol{I} \;-\; \frac{2}{\lambda_{\max}+\lambda_{\min}}\,\boldsymbol{B}\right) \boldsymbol{x}^j \;+\; (1-d_j) \cdot \boldsymbol{x}^{j-1}\;, \qquad (4.67)
\end{aligned}$$

where we denote $\boldsymbol{x}^{-1} := \boldsymbol{x}^0$. This means that we not only use the last intermediate result $\boldsymbol{x}^j$, but also its predecessor $\boldsymbol{x}^{j-1}$. Interestingly, the matrix-vector multiplication in the recursion formula corresponds to an optimal iteration step of Richardson's method with the constant relaxation parameter $\omega = \frac{2}{\lambda_{\max}+\lambda_{\min}}$. As for the Runge-Kutta-Chebyshev methods, the additional effort includes the storage of $\boldsymbol{x}^{j-1}$ and the computation of a vector sum. Moreover, we do not have any problems with the numerical stability, because we do not use a sequence of varying relaxation parameters anymore. We should mention that in the case of the linear system $\boldsymbol{Bx} = \boldsymbol{c}$ with an arbitrary right hand side $\boldsymbol{c} \in \mathbb{R}^N$, the iteration reads

$$\boldsymbol{x}^{j+1} \;=\; d_j \cdot \left(\boldsymbol{x}^j \;+\; \frac{2}{\lambda_{\max}+\lambda_{\min}} \cdot \left(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^j\right)\right) \;+\; (1-d_j) \cdot \boldsymbol{x}^{j-1}\,. \qquad (4.68)$$

It is also possible to construct semi-iterative methods for linear systems with unsymmetric matrices, which is shown, for example, in [44, 45].

4.3.1 Semi-Iterative Fast-Jacobi

At this point, we want to transfer the idea to the Fast-Jacobi method presented in Chapter 3. Assuming a cycle length $j+1$, we get, according to Sec. 3.4,

$$\boldsymbol{e}^{j+1} \;=\; \frac{1}{2j+3} \cdot U_{2j+2}\left(\sqrt{\boldsymbol{I} \;-\; \tfrac{\omega_{\text{lim}}}{2}\,\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}}\,\right)\boldsymbol{e}^0 \;, \tag{4.69}$$

with the errors $\boldsymbol{e}^\ell = \boldsymbol{x} - \boldsymbol{x}^\ell$ between the exact solution $\boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{c}$ and the current iteration $\boldsymbol{x}^\ell$. The constant $\omega_{\text{lim}} > 0$ has already been defined in Sec. 3.4.1. Using the recursion formula Eq. (4.20) presented in Sec. 4.1.3 yields with the coefficients α_{j+1} from Eq. (4.36) the following result:

$$\boldsymbol{e}^{j+1} \;=\; \alpha_{j+1} \cdot \big(\boldsymbol{I} \;-\; \omega_{\text{lim}}\,\boldsymbol{D}_{\boldsymbol{B}}^{-1}\boldsymbol{B}\big)\,\boldsymbol{e}^j \;+\; (1-\alpha_{j+1}) \cdot \boldsymbol{e}^{j-1} \;, \tag{4.70}$$

or in terms of $\boldsymbol{x}^{j+1}$:

$$\boldsymbol{x}^{j+1} \;=\; \alpha_{j+1} \cdot \big(\boldsymbol{x}^j \;+\; \omega_{\text{lim}}\,\boldsymbol{D}_{\boldsymbol{B}}^{-1}\,(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^j)\big) \;+\; (1-\alpha_{j+1}) \cdot \boldsymbol{x}^{j-1} \;. \tag{4.71}$$

Thus, Eq. (4.71) is the semi-iterative version of the proposed Fast-Jacobi method (3.191). This semi-iterative scheme shares all convergence properties with the usual Fast-Jacobi method, but is much more robust with respect to numerical rounding errors, due to the stable recursion.

4.3.2 Implementation

Figure 4.2 illustrates the implementation of the proposed *semi-iterative Fast-Jacobi (SIFJ)*. It is based on the iteration in Eq. (4.71). In contrast to the usual algorithm shown in Fig. 3.13, the computation of the varying relaxation parameters and the reordering have been dropped. As before for FED Runge-Kutta schemes, inner updates of nonlinearities are possible. In this context, we refer to the experimental section.

4.4 Numerical Experiments

The two main advantages of both the FEDRK scheme (4.58) and the semi-iterative Fast-Jacobi solver (4.71) are the abolition of a rearrangement and the stability of each inner step or iteration, which allows more flexibility with respect to update strategies. Since the computational effort per inner step or iteration is a bit larger than for the usual FED or Fast-Jacobi, the question is whether the novel schemes benefit from additional updates. As testbeds we use from Chapter 3 the fingerprint enhancement for the parabolic, and the Charbonnier regularisation for the elliptic case.

```
1. Input Data:
   linear system of equations $\boldsymbol{Bx} = \boldsymbol{c}$, number of inner iterations
   $n$, number $m$ of outer iterations, feasible $\omega_{\text{lim}} > 0$

2. Initialisation:
     – Define an initial vector $\boldsymbol{x}^0$.

3. Outer Iteration Loop: $(k = 1, \ldots, m)$
   (a) $\boldsymbol{x}^{k,-1} = \boldsymbol{x}^{k,0} := \boldsymbol{x}^{k-1}$.
   (b) Perform $n$ inner iterations $(j = 0, \ldots, n-1)$:
         $\boldsymbol{x}^{k,j+1} = \alpha_{j+1} \cdot \left(\boldsymbol{x}^{k,j} + \omega_{\text{lim}} \boldsymbol{D}_{\boldsymbol{B}}^{-1} \left(\boldsymbol{c} - \boldsymbol{Bx}^{k,j}\right)\right)$
                    $+ (1 - \alpha_{j+1}) \cdot \boldsymbol{x}^{k,j-1}$ .
   (c) $\boldsymbol{x}^k := \boldsymbol{x}^{k,n}$.
```

Figure 4.2: Semi-iterative Fast-Jacobi algorithm.

4.4.1 Parabolic Problems

The MSEs for the fingerprint enhancement with the coherence-enhancing diffusion filtering can be seen in Table 4.1. Besides the original FEDRK scheme that updates the nonlinearities after each inner step, we also show the results of a modified scheme that performs only a fixed number of updates within an outer FEDRK step. More precisely, given the number s of stages and the fixed number $\ell < s$, the inner updates are made after $\left\lfloor k \cdot \frac{s}{(\ell+1)} \right\rfloor$ steps, where $k = 1, \ldots, \ell$. This means that we keep the nonlinearities in the matrix $\boldsymbol{P}$ fixed between two updates. It is not surprising that the original FEDRK scheme is the best, because it uses the largest possible number of updates. However, even the modified schemes with only one or two inner updates per outer step yield significantly better results than the cyclic method. Regarding the MSEs in Table 3.8, all three FEDRK schemes outperform the semi-implicit method with respect to accuracy. This shows how useful these inner updates can be. In the context of efficiency, the original FEDRK method should not be used. On the one hand it is the most accurate method, but on the other hand it requires, due to many updates, much more computational effort. If we assume, for example, one outer

Table 4.1: MSE-Comparison between FED and FED Runge-Kutta schemes for the fingerprint enhancement with CED (Fig. 3.22).

cycles or			**MSE**	
outer steps	**FED**	**FEDRK**	**FEDRK (1)**	**FEDRK (2)**
1	88.382	17.683	54.576	37.578
2	39.429	3.146	22.701	15.438
3	27.448	1.995	15.812	9.934
4	21.758	1.579	12.987	8.176
5	18.239	1.146	10.541	6.421
10	11.093	0.536	4.855	2.330
25	3.841	0.197	1.038	0.588
50	0.943	0.089	0.315	0.239
100	0.260	0.030	0.083	0.062

FEDRK step, its running time is about one second, whereas the modified schemes with one and two intermediate updates require only around 0.08 and 0.1 seconds, respectively. To reach or fall below the corresponding MSE of the original FEDRK method with one outer step, the modified schemes require indeed two or three cycles, but are nevertheless still more than five times faster.

The efficiency of some methods is illustrated in Fig. 4.3. A comparison between FED and the equivalent FEDRK scheme without any inner update (i.e. only one update per complete outer step) is given in Fig. 4.3(a). As expected, this Runge-Kutta method is a bit more expensive, but the incorporation of inner updates improves the efficiency; see Fig. 4.3(b). If one wants to apply only a few outer steps or cycles, i.e. more inner steps, they even provide a better efficiency. However, for small numbers of inner steps, updates might be not so beneficial anymore, which deteriorates the efficiency.

Now we want to consider the FED Runge-Kutta scheme (4.58) combined with the above proposed predictor-corrector strategy. The accuracy and efficiency of this method are mainly influenced by the computation of the predictor. Its approximation quality depends on the number of outer Runge-Kutta steps and possible inner updates. If we assume e.g. three updates for the computation of the predictor, we could perform three Runge-Kutta steps without any inner updates or only one step including two inner up-

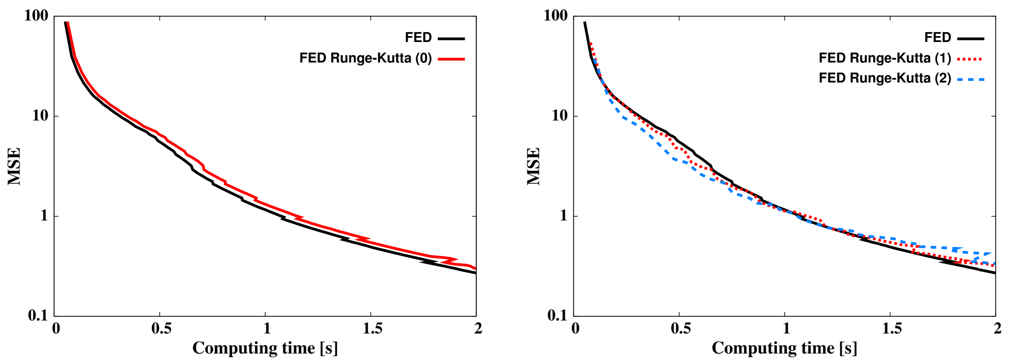


Figure 4.3: Computing time versus MSE for FED and FED Runge-Kutta (FEDRK) schemes with different numbers of updates. **(a) Left:** FED and FEDRK without inner updates. **(b) Right:** With inner updates.

dates. Note that the latter option needs less computational effort.

The results for different update strategies with an overall number of two or three updates per predictor step are shown in Table 4.2. As expected, increasing the number of updates improves the quality of both the predictor and the overall result. Moreover, the less expensive update strategy with one cycle yields smaller MSEs in almost all cases. Compared to the modified FED Runge-Kutta schemes in Table 4.1, we have much better results. However, the computation of the predictor requires additional effort. Thus, it is very interesting to look into the efficiency of these methods.

In Fig. 4.4 one can find some comparisons with respect to the efficiency. We have used the predictor-corrector scheme with one Runge-Kutta step for the computation of the predictor. The graph in Fig. 4.4(a) illustrates the benefit of additional inner updates. They clearly improve the efficiency of the scheme. A comparison with the usual FED scheme is shown in Fig. 4.4(b). Obviously, the predictor-corrector FED Runge-Kutta schemes can outperform the usual cyclic method with respect to the efficiency, i.e. it is up to two times faster. Note that the predictor-corrector method without any inner update could also be computed with the help of the usual FED scheme from Chapter 3, which would be also a reasonable extension of FED. Regarding the efficiency, we recommend to use not more than three inner updates per outer step.

4.4.2 Elliptic Problems

Our second experiment deals with the example for Charbonnier regularisation from Fig. 3.27. The main question is whether the semi-iterative

Table 4.2: MSE-Comparison between predictor-corrector FED Runge-Kutta schemes (PRK) with different update strategies (number of outer steps, inner updates) for the computation of the predictor.

outer steps	**MSE**			
	PRK (2,0)	**PRK (1,1)**	**PRK (3,0)**	**PRK (1,2)**
1	35.474	34.827	29.420	28.585
2	12.547	12.186	9.623	8.374
3	8.275	8.126	5.634	3.905
4	5.401	4.563	2.562	2.036
5	3.393	2.670	1.407	1.233
10	0.569	0.493	0.254	0.289
25	0.078	0.080	0.063	0.056

Fast-Jacobi solver also benefits from inner updates. More precisely, we can compare, for example, a solver with an even cycle length n and one inner update to another one that uses the cycle length $\frac{n}{2}$ without any inner update.

In Chapter 3 we have seen that the cycle length $n = 50$ yields an efficient Fast-Jacobi solver. Thus, we also use 50 inner iterations for our semi-iterative Fast-Jacobi method. If we do not use any inner update, the semi-iterative Fast-Jacobi method is equivalent to its cyclic counterpart. Unfortunately, the semi-iterative version requires about 20% more computational effort. However, we compare it to the solvers with one and two inner updates per outer iteration, respectively. This comparison is illustrated in Fig. 4.5(a). Since the methods have different computational efforts, due to the varying numbers of updates, we consider the computing time. The figure shows that the semi-iterative method can significantly benefit from inner updates, in particular for smaller numbers of outer iterations. More precisely, the semi-iterative methods can be more than two times faster.

Another comparison is shown in Fig. 4.5(b): Here we consider the semi-iterative Fast-Jacobi method with three different values for n, but use the same update strategy. This means we update the nonlinearities every 50 iterations. Thus, all methods have the same computational effort per iteration, and we can consider the number of iterations instead of the CPU time. The method with $n = 100$ inner iterations has the best performance. If n is further increased, the convergence can become worse, which is illustrated with $n = 200$. On the other hand, the method with only 50 inner

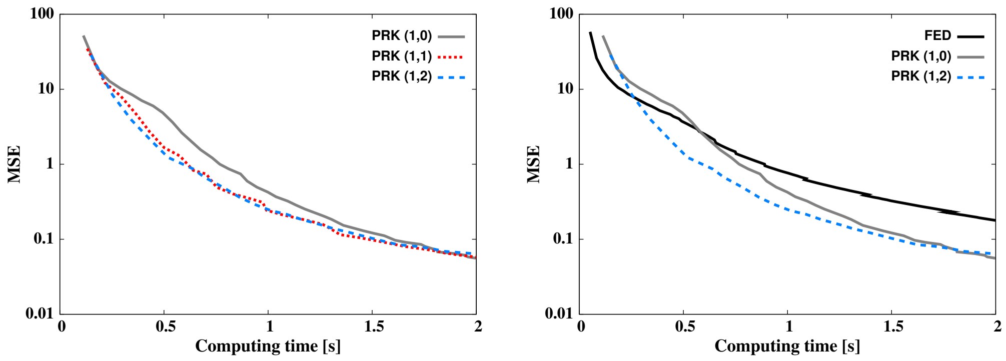


Figure 4.4: Computing time versus MSE for predictor-corrector FED Runge-Kutta schemes using one cycle for computing the predictor. **(a) Left:** With different update strategies. **(b) Right:** Comparison with the usual FED scheme.

iterations, i.e. inner updates are not necessary, also yields a suboptimal convergence speed. Hence, we have to optimise both n and the number of inner updates. However, the experiments suggest that up to three inner updates per outer iteration are sufficient. As before in the parabolic case, too many inner updates can deteriorate the efficiency.

4.5 Summary

This chapter has mainly dealt with a modified FED scheme. In contrast to Chapter 3, where we have decomposed the 1-D box filter into explicit diffusion steps, it is based on a recursion relation for box filters. On the one hand this implies more effort, because we have to go one step further in the past and consider the last two intermediate results instead of only one. However, on the other hand there is the major advantage that a rearrangement of the time step sizes is not necessary anymore. Moreover, this modified FED scheme is also well-suited for parallel computing.

In the context of inner stability we have shown that each inner time step is stable. To this end, it is possible to perform updates within an outer time step, which can improve the accuracy as well as the efficiency with respect to the solution of nonlinear problems. This modified FED scheme is related to the well-known class of explicit Runge-Kutta methods. In the corresponding literature one can also find Runge-Kutta methods that are related to Super Time Stepping. However, they still need an additional damping parameter.

Based on a Runge-Kutta scheme that uses the midpoint rule for numer-

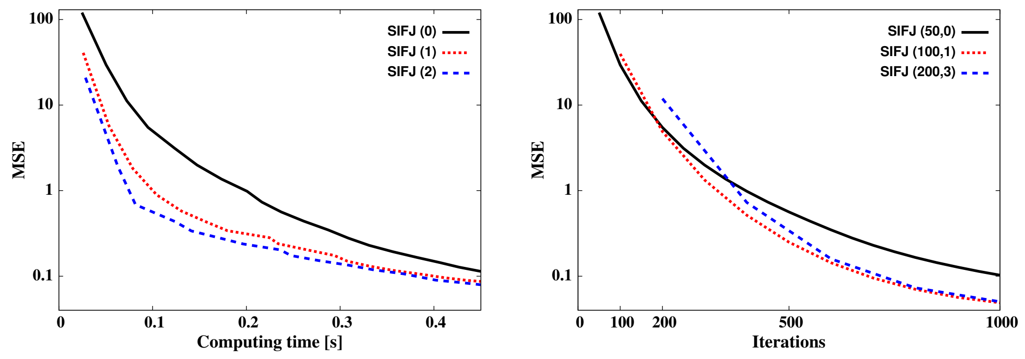


Figure 4.5: Computational effort versus MSE for the semi-iterative Fast-Jacobi (SIFJ) applied to Charbonnier regularisation. **(a) Left:** With $n = 100$ inner iterations and different numbers of inner updates. **(b) Right:** With different n and updates after every 50 iterations.

ical integration, we have proposed a predictor-corrector FED Runge-Kutta method that can further improve the accuracy.

Besides the parabolic FED scheme, we have also modified the Fast-Jacobi method such that there is no need for a rearrangement of varying relaxation parameters. Such methods are called semi-iterative, and also require the last two intermediate results, i.e. there is more computational effort than for usual iterative (cyclic) methods. However, similar to the parabolic case, this additional effort allows inner updates that can improve the convergence.

In the experimental section we have analysed the proposed methods with respect to their accuracy and efficiency.

For the Runge-Kutta methods we have seen that the most accurate approach is the original scheme which updates the nonlinearities after each inner time step. However, it is very expensive because of the large number of updates and therefore yields a poor efficiency. To this end, we have considered modified methods that only use a small, fixed number of inner updates per cycle. As demonstrated, even only one or two inner updates are sufficient to outperform the cyclic FED scheme from Chapter 3, provided that the cycle length or the number of inner time steps is not too small.

However, the efficiency of the FED Runge-Kutta schemes can be further improved by using a predictor-corrector strategy. It clearly outperforms the usual FED scheme, though the predictor-corrector strategy does not increase the first order consistency. We have seen that the accuracy and efficiency mainly depends on the predictor, whose quality can increase with the help of, for example, one or two inner updates per outer time step.

Since the approaches without any inner updates for the computation of the predictor could be also used in combination with the usual FED scheme, the predictor-corrector strategy is also a reasonable option for cyclic methods.

Moreover, inner updates are also very helpful in the context of the semi-iterative Fast-Jacobi method. They can improve the convergence and thus compensate the additional effort of semi-iterative methods, as we have shown in our example with Charbonnier regularisation. Similarly to the parabolic case, up to three inner updates are sufficient to obtain efficient algorithms.

Chapter 5

FED Extrapolation Methods

Fast is fine, but accuracy is everything.
Xenophon

In the last chapter, we have constructed a more accurate FED method with the help of the Runge-Kutta framework and a predictor-corrector strategy. Another way to create more accurate, or higher order, methods is the so-called *extrapolation of numerical schemes*. The idea is quite simple: Using a suitable linear combination of two or more low order numerical solutions, one can achieve a higher order approximation. The coefficients of this linear combination are usually determined by comparing the Taylor series of the low order numerical and the analytic solution. Since the extrapolation is based on the numerical scheme that provides the low order solutions, it is very easy to implement, provided that the code for the underlying numerical method already exists.

Ideas of this type have a long tradition and have already been mentioned by Huygens in 1654 [76], where he proposed an algorithm for the approximation of the number π. There are also works in the context of image processing like e.g. [8]. A detailed historical overview for extrapolation methods can be found in [15].

The goal of this chapter is to show that the combination of extrapolation methods and the FED scheme is possible and yields stable schemes. To this end, we consider an existing extrapolation framework that will serve as an example.

5.1 Richardson Extrapolation

Our example is the well-known Richardson extrapolation, which was introduced in 1927 by Richardson and Gaunt [112]. It is used e.g. in Romberg's method for numerical integration [113] and has also become popular for the solution of partial or ordinary differential equations [61, 62, 85, 88].

Let us again consider the linear 1-D diffusion equation and the corresponding time-continuous linear ODE system

$$\frac{d\boldsymbol{u}}{dt} \;=\; \boldsymbol{A_h}\boldsymbol{u}\;, \tag{5.1}$$

with the vector-valued function $\boldsymbol{u}$ and the matrix $\boldsymbol{A_h} \in \mathbb{R}^{N\times N}$ discretising the Laplacian operator. To solve this system of ODEs numerically, we discretise it in the temporal domain with a time step size $\tau > 0$ and use the scheme

$$\boldsymbol{u}^{k+1} \;=\; \boldsymbol{Q}_\tau\,\boldsymbol{u}^k\;, \tag{5.2}$$

where $\boldsymbol{u}^k \approx \boldsymbol{u}(k\cdot\tau)$ and $\boldsymbol{Q}_\tau \in \mathbb{R}^{N\times N}$ is a matrix that depends on the time step size τ as well as the matrix $\boldsymbol{A_h}$. The numerical scheme is for example explicit in the case of $\boldsymbol{Q}_\tau = \boldsymbol{I} + \tau\,\boldsymbol{A_h}$ and implicit for $\boldsymbol{Q}_\tau = (\boldsymbol{I} - \tau\,\boldsymbol{A_h})^{-1}$. Assuming that $\boldsymbol{Q}_\tau$ implies a scheme with first order accuracy in time, the Richardson extrapolation method works as follows: One has to compute the first order approximations

$$\boldsymbol{u}_1^{k+1} \;=\; \boldsymbol{Q}_\tau\,\boldsymbol{u}^k \tag{5.3}$$

with one time step of size τ,

$$\boldsymbol{u}_2^{k+1} \;=\; \boldsymbol{Q}_{\frac{\tau}{2}}\,\boldsymbol{Q}_{\frac{\tau}{2}}\,\boldsymbol{u}^k \tag{5.4}$$

with two steps using the time step size $\frac{\tau}{2}$ and combine them via

$$\boldsymbol{u}^{k+1} \;=\; 2\cdot\boldsymbol{u}_2^{k+1} \;-\; \boldsymbol{u}_1^{k+1}\;. \tag{5.5}$$

This extrapolated result has second order accuracy. More details including higher order schemes are given, for example, in [61, 62, 63, 88].

5.1.1 FED Extrapolation Scheme

Now we want to construct a second order FED scheme based on the above described extrapolation method. This means we have

$$\boldsymbol{Q}_\tau \;=\; \prod_{\ell=0}^{n-1}\left(\boldsymbol{I} \;+\; \tau_\ell\,\boldsymbol{A_h}\right) \tag{5.6}$$

with the time step sizes

$$\tau_\ell \;=\; \frac{h^2}{2} \cdot \frac{q}{2\,\cos^2\left(\pi \cdot \frac{2\ell+1}{4n+2}\right)} \qquad (\ell = 0, \ldots, n-1)\,. \tag{5.7}$$

The cycle length n and the factor $q \leq 1$ are chosen such that

$$\sum_{\ell=0}^{n-1} \tau_\ell \;=\; \tau\,, \tag{5.8}$$

which means that the cycle time is given by τ. Moreover, we define the matrix of the cycle with stopping time $\frac{\tau}{2}$:

$$\boldsymbol{Q}_{\frac{\tau}{2}} \;=\; \prod_{\ell=0}^{n-1} \left(\boldsymbol{I} \,+\, \tfrac{\tau_\ell}{2}\,\boldsymbol{A_h}\right)\,. \tag{5.9}$$

Thus, the Richardson extrapolation of the FED scheme reads

$$\left\{\begin{aligned} \boldsymbol{u}_1^{k+1} \;&=\; \left(\prod_{\ell=0}^{n-1} \left(\boldsymbol{I} \,+\, \tau_\ell\,\boldsymbol{A_h}\right)\right) \boldsymbol{u}^k \\ \boldsymbol{u}_2^{k+1} \;&=\; \left(\prod_{\ell=0}^{n-1} \left(\boldsymbol{I} \,+\, \tfrac{\tau_\ell}{2}\,\boldsymbol{A_h}\right)\right)^2 \boldsymbol{u}^k \\ \boldsymbol{u}^{k+1} \;&=\; 2 \cdot \boldsymbol{u}_2^{k+1} \,-\, \boldsymbol{u}_1^{k+1}\,. \end{aligned}\right. \tag{5.10}$$

The extrapolation framework states that this scheme has second order accuracy in time. However, the computation of $\boldsymbol{u}^{k+1}$ is based on a nonconvex linear combination of two numerical solutions. Hence, the stability can not be directly guaranteed. Therefore, our goal is to prove that the above extrapolation method is stable.

5.1.2 Stability Analysis

To show the stability of the *FED extrapolation scheme* (5.10), we consider the Euclidean norm. This means we want to prove that

$$\left\|\boldsymbol{u}^{k+1}\right\|_2 \;\leq\; \left\|\boldsymbol{u}^k\right\|_2 \tag{5.11}$$

for $k \geq 0$. Because of

$$\begin{aligned}\left\|\boldsymbol{u}^{k+1}\right\|_2 &= \left\|\left(2\cdot\left(\prod_{\ell=0}^{n-1}\left(\boldsymbol{I} + \tfrac{\tau_\ell}{2}\,\boldsymbol{A_h}\right)\right)^2 - \prod_{\ell=0}^{n-1}\left(\boldsymbol{I} + \tau_\ell\,\boldsymbol{A_h}\right)\right)\boldsymbol{u}^k\right\|_2 \qquad (5.12)\\ &\leq \left\|2\cdot\left(\prod_{\ell=0}^{n-1}\left(\boldsymbol{I} + \tfrac{\tau_\ell}{2}\,\boldsymbol{A_h}\right)\right)^2 - \prod_{\ell=0}^{n-1}\left(\boldsymbol{I} + \tau_\ell\,\boldsymbol{A_h}\right)\right\|_2 \cdot \left\|\boldsymbol{u}^k\right\|_2\,,\end{aligned}$$

we have to ensure that the real-valued eigenvalues of the symmetric matrix $2\cdot\left(\boldsymbol{Q}_{\frac{\tau}{2}}\right)^2 - \boldsymbol{Q}_\tau$ are bounded in absolute value by 1. This is equivalent to the boundedness of the corresponding amplification factor that is given by

$$2\cdot\left(\frac{1}{2n+1}\,U_{2n}\left(\sqrt{1-\tfrac{q\,h^2}{4}\cdot\tfrac{z}{2}}\right)\right)^2 - \frac{1}{2n+1}\,U_{2n}\left(\sqrt{1-\tfrac{q\,h^2}{4}\cdot z}\right) \qquad (5.13)$$

with $z \in [0, \frac{4}{h^2}]$. Thus, it remains to show that this polynomial takes values only in $[-1,1]$ for $z \in [0, \frac{4}{h^2}]$. Before we prove the corresponding theorem, we first show a lemma that will help us with the proof of the boundedness.

Lemma 5.1 (**Estimation for Polynomials**). *Let* $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ *be an arbitrary finite set of real-valued positive numbers, where the minimum value of the set is denoted by* $\alpha > 0$. *Then the* n*-degree polynomial*

$$q_n(x) := \prod_{i=0}^{n-1}\left(1 - \frac{x}{\alpha_i}\right) \qquad (5.14)$$

fulfils for $x \in [0, 2\alpha)$ *:*

$$-1 < 2\cdot q_n^2\left(\tfrac{x}{2}\right) - q_n(x) \leq 1\,. \qquad (5.15)$$

Proof. We prove Eq. (5.15) by induction over the degree $n \geq 0$. The case $n = 0$ is trivial. For the inductive step, we reconsider the above set and add a positive number $\alpha_n > 0$. A new possible minimum $\alpha' > 0$ satisfies $\alpha' \leq \alpha$. Thus, we can assume that Eq. (5.15) holds in particular for $x < 2\alpha'$.

We obtain for $n+1$:

$$\begin{aligned}
& 2 \cdot q_{n+1}^2\left(\tfrac{x}{2}\right) \;-\; q_{n+1}(x) \\
=\; & 2 \cdot \left(1 - \frac{x}{2\alpha_n}\right)^2 q_n^2\left(\tfrac{x}{2}\right) \;-\; \left(1 - \frac{x}{\alpha_n}\right) q_n(x) \\
=\; & 2 \cdot \left(1 - \frac{x}{2\alpha_n}\right)^2 q_n^2\left(\tfrac{x}{2}\right) \;-\; \left(\left(1 - \frac{x}{2\alpha_n}\right)^2 - \frac{x^2}{4\alpha_n^2}\right) q_n(x) \\
=\; & \left(1 - \frac{x}{2\alpha_n}\right)^2 \underbrace{\left(2 \cdot q_n^2\left(\tfrac{x}{2}\right) \;-\; q_n(x)\right)}_{\overset{(5.15)}{\leq}\; 1} + \frac{x^2}{4\alpha_n^2} \underbrace{q_n(x)}_{\leq 1} \\
\leq\; & \left(1 - \frac{x}{2\alpha_n}\right)^2 + \frac{x^2}{4\alpha_n^2} \\
=\; & 1 - \frac{x}{\alpha_n} + \frac{x^2}{2\alpha_n^2} \quad \leq \quad 1\,,
\end{aligned} \tag{5.16}$$

where we have used in the last step that $\frac{x}{\alpha_n} < \frac{2\alpha'}{\alpha_n} \leq 2$. Since $1 - \frac{x}{\alpha_n} > -1$ for $x < 2\alpha'$, the left inequality holds because of

$$\begin{aligned}
& \left(1 - \frac{x}{2\alpha_n}\right)^2 \left(2 \cdot q_n^2\left(\tfrac{x}{2}\right) \;-\; q_n(x)\right) \;+\; \frac{x^2}{4\alpha_n^2} q_n(x) \\
=\; & \underbrace{\left(1 - \frac{x}{\alpha_n}\right)\left(2 \cdot q_n^2\left(\tfrac{x}{2}\right) \;-\; q_n(x)\right)}_{> -1} + \frac{x^2}{2\alpha_n^2} q_n^2\left(\tfrac{x}{2}\right) \;>\; -1\,.
\end{aligned} \tag{5.17}$$

□

Equation (5.16) means for all $n \geq 1$ that the value 1 is reached only for $x = 0$. This avoids that some high frequency components are not damped. Now we are going to prove the theorem about the stability of the FED extrapolation scheme (5.10).

Theorem 5.2 (Stability of FED Extrapolation Scheme). *The FED extrapolation scheme* (5.10) *is stable in the sense of the Euclidean norm.*

Proof. Without loss of generality, we consider for $n \in \mathbb{N}$ and $z \in [0,1]$ the amplification factor

$$2 \cdot \left(\frac{1}{2n+1} U_{2n} \left(\sqrt{1 - \tfrac{z}{2}} \right) \right)^2 - \frac{1}{2n+1} U_{2n} \left(\sqrt{1 - z} \right) . \tag{5.18}$$

The polynomial

$$\frac{1}{2n+1} U_{2n} \left(\sqrt{1 - z} \right) = \prod_{i=0}^{n-1} \left(1 - \frac{z}{\alpha_i} \right) \tag{5.19}$$

has the roots

$$\alpha_i = \cos^2 \left(\pi \cdot \tfrac{2i+1}{4n+2} \right) > 0 . \tag{5.20}$$

The smallest value of these roots is α_{n-1} and, according to Lemma 5.1, it follows that the amplification factor is bounded in absolute value by 1 for $z < 2\alpha_{n-1}$. Let us now consider the case $z \geq 2\alpha_{n-1}$. By means of the estimation in Eq. (3.206) we have

$$\left| \frac{1}{2n+1} U_{2n} \left(\sqrt{1 - z} \right) \right| \leq \frac{1}{2n+1} \frac{1}{\sqrt{z}} \leq \frac{1}{(2n+1)\sqrt{2\, \alpha_{n-1}}} . \tag{5.21}$$

Furthermore, we get

$$\begin{aligned} \frac{1}{\sqrt{2\, \alpha_{n-1}}} &= \frac{1}{\sqrt{2}} \cdot \frac{1}{\cos \left(\frac{\pi}{2} - \pi \cdot \frac{2}{4n+2} \right)} \\ &= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sin \left(\pi \cdot \frac{1}{2n+1} \right)} \\ &\leq \frac{1}{\sqrt{2}} \cdot \frac{1}{3 \cdot \frac{1}{2n+1} \cdot \sin \left(\frac{\pi}{3} \right)} = \frac{2n+1}{3} \cdot \sqrt{\frac{2}{3}} , \end{aligned} \tag{5.22}$$

where we have used the inequality $\sin(\pi \cdot x) \geq 3x \cdot \sin(\frac{\pi}{3})$ for $x \in \left[0\, ,\, \frac{1}{3}\right]$, which follows from the concavity of the sine function in the interval $\left[0\, ,\, \frac{\pi}{3}\right]$. Putting these results together, we obtain the estimation

$$\left| \frac{1}{2n+1} U_{2n} \left(\sqrt{1 - z} \right) \right| \leq \sqrt{\frac{2}{27}} , \tag{5.23}$$

for $z \geq 2\alpha_{n-1}$ and analogously

$$\left| \frac{1}{2n+1} U_{2n} \left(\sqrt{1 - \tfrac{z}{2}} \right) \right| \leq \sqrt{\frac{4}{27}} . \tag{5.24}$$

Thus, the symbol fulfils

$$\left| 2 \cdot \left(\frac{1}{2n+1} U_{2n} \left(\sqrt{1 - \tfrac{z}{2}} \right) \right)^2 - \frac{1}{2n+1} U_{2n} \left(\sqrt{1 - z} \right) \right|$$

$$\leq 2 \cdot \left(\sqrt{\frac{4}{27}} \right)^2 + \sqrt{\frac{2}{27}} = \frac{8}{27} + \sqrt{\frac{2}{27}} \approx 0.5685 < 1 . \qquad (5.25)$$

Overall, it is bounded in absolute value by 1 for $z \in [0, 1]$, which yields the stability of the FED extrapolation scheme. $\square$

Interestingly, the stability of the Richardson extrapolation with the Super Time Stepping (STS) scheme depends on the damping factor ν. Although the STS method itself yields stable results with $\nu = 0$, the corresponding extrapolation is highly unstable. With $\nu = 0$, the symbol of an STS extrapolation scheme is given by

$$2 \cdot \big(T_n (1 - z) \big)^2 - T_n (1 - 2z) \qquad \big(z \in [0, 1] \big) , \qquad (5.26)$$

where we have just replaced the FED by the STS symbol. Figure 5.2 shows an example for $n = 10$. As one can see, this exemplar symbol takes values between -1 and 3. In order to enforce stability, one has to use a damping factor $\nu > 0$. Thus, the damping factor decides not only about the quality, but also now about the stability of the result. This is a clear disadvantage compared to the damping parameter-free FED extrapolation scheme.

5.1.3 Accuracy

Besides the stability, another important issue is the accuracy for large time step sizes $\tau \gg 0$, because a higher order scheme might be only better for very small step sizes $\tau \approx 0$. Since some applications might require large stopping times, it is useful to have schemes that provide a reasonable accuracy also for larger time step sizes. To this end, we consider the Taylor expansion of the FED extrapolation scheme (5.10). For $\boldsymbol{u}_1^{k+1}$ we have the usual Taylor expansion of an FED cycle,

$$\boldsymbol{u}_1^{k+1} = \left(\sum_{m=0}^{n} \frac{h^{2m}}{2m+1} \binom{n+m}{2m} \boldsymbol{A}_{\boldsymbol{h}}^m \right) \boldsymbol{u}^k , \qquad (5.27)$$

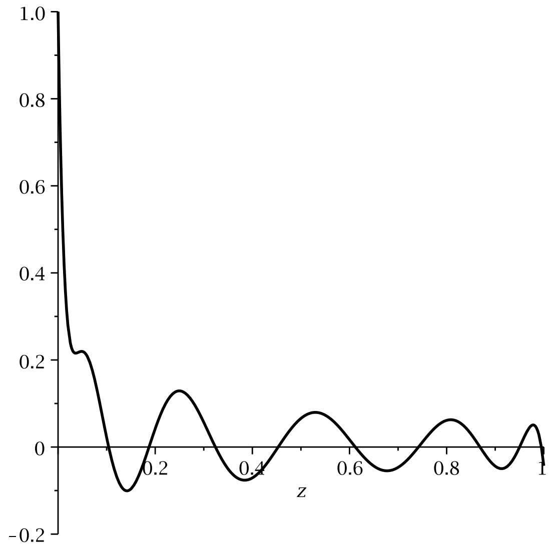


Figure 5.1: Symbol of Richardson extrapolation with FED for $n = 10$.

and $\boldsymbol{u}_2^{k+1}$ satisfies

$$\begin{aligned}\boldsymbol{u}_2^{k+1} &= \left(\sum_{m=0}^{n} \frac{h^{2m}}{2m+1} \binom{n+m}{2m} \frac{1}{2^m} \boldsymbol{A}_{\boldsymbol{h}}^m\right)^2 \boldsymbol{u}^k \\ &= \left(\sum_{m=0}^{2n} \frac{h^{2m}}{2^m} \left(\sum_{k=0}^{m} \frac{\binom{n+k}{2k}\binom{n+m-k}{2(m-k)}}{(2k+1)(2(m-k)+1)}\right) \boldsymbol{A}_{\boldsymbol{h}}^m\right) \boldsymbol{u}^k \,. \qquad (5.28)\end{aligned}$$

Overall, we obtain

$$\boldsymbol{u}^{k+1} = \left(\sum_{m=0}^{2n} c_m \cdot \boldsymbol{A}_{\boldsymbol{h}}^m\right) \boldsymbol{u}^k \,, \qquad (5.29)$$

with the coefficients

$$c_m := h^{2m} \left(\frac{1}{2^{m-1}} \sum_{k=0}^{m} \frac{\binom{n+k}{2k}\binom{n+m-k}{2(m-k)}}{(2k+1)(2(m-k)+1)} - \frac{\binom{n+m}{2m}}{2m+1} \right) . \qquad (5.30)$$

Because of the second order accuracy, the coefficients fulfil

$$c_1 = \frac{h^2}{3} \cdot \binom{n+1}{2} \qquad (5.31)$$

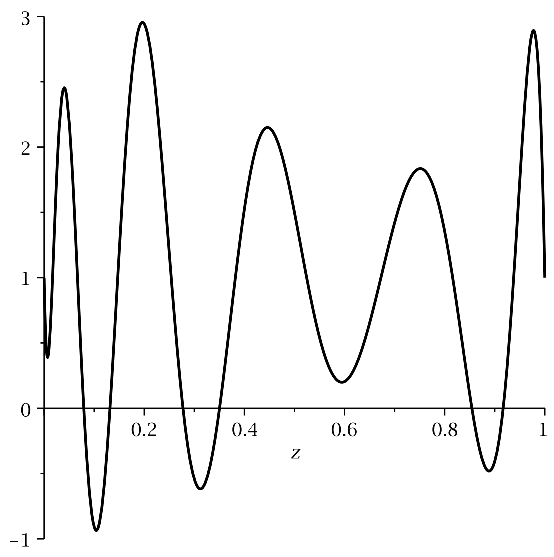


Figure 5.2: Symbol of Richardson extrapolation with STS for $n = 10$.

and

$$c_2 = \frac{h^4}{18} \cdot \binom{n+1}{2}^2 = \frac{1}{2} \cdot c_1^2 \,. \tag{5.32}$$

Note that the exact coefficients of the matrix exponential $\exp(c_1 \cdot \boldsymbol{A_h})$ are given by

$$\tilde{c}_m = \frac{1}{m!} \cdot c_1^m \,. \tag{5.33}$$

In order to analyse the accuracy, we compute the relative deviation

$$d_m = \frac{|\tilde{c}_m - c_m|}{\tilde{c}_m} \,. \tag{5.34}$$

Since $d_m = 0$ for $m \in \{0, 1, 2\}$, we consider the case $m > 2$. Some results are shown in Table 5.1. The behaviour of the higher order coefficients seems to be reasonable, in particular the deviation of the third order coefficient is only about 23%, which means that c_3 and $\tilde{c}_3$ are in the same order of magnitude. Thus, we can expect reasonable results even for larger time step sizes.

Table 5.1: Relative deviations d_m for $m \in \{3, 4, 5, 6\}$ and different n.

$\boldsymbol{n}$	$\boldsymbol{d_3}$	$\boldsymbol{d_4}$	$\boldsymbol{d_5}$	$\boldsymbol{d_6}$
5	0.2560	0.6016	0.8447	0.9555
10	0.2357	0.5542	0.7938	0.9218
50	0.2289	0.5379	0.7749	0.9074
100	0.2286	0.5373	0.7742	0.9069
500	0.2286	0.5372	0.7740	0.9068
1000	0.2286	0.5371	0.7740	0.9068

5.2 Nonlinear Problems

So far, we have considered extrapolation schemes for the linear case. For the numerical solution of the general problem

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{P}(\boldsymbol{u})\,\boldsymbol{u}\,, \tag{5.35}$$

we have to take care of the nonlinearities. To this end, we use again the predictor-corrector strategy similar to the Runge-Kutta scheme (4.27) that is based on the midpoint rule. This strategy has already been successfully used, e.g. in [63], for isotropic diffusion problems. According to the Runge-Kutta scheme (4.27), we obtain with the given data $\boldsymbol{u}^0$

$$\begin{aligned}
\boldsymbol{u}^1 &= \boldsymbol{u}^0 + \tau\,\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{u}^{1/2} \\
&= \boldsymbol{u}^0 + \tau\,\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{u}^0 + \frac{\tau^2}{2}\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{P}\left(\boldsymbol{u}^0\right)\boldsymbol{u}^0 \\
&= \boldsymbol{u}^0 + \tau\,\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{u}^0 + \frac{\tau^2}{2}\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\underbrace{\boldsymbol{P}\left(\boldsymbol{u}^{1/2}+\mathcal{O}(\tau)\right)}_{=\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)+\mathcal{O}(\tau)}\boldsymbol{u}^0 \\
&= \boldsymbol{u}^0 + \tau\,\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{u}^0 + \frac{\tau^2}{2}\boldsymbol{P}^2\left(\boldsymbol{u}^{1/2}\right)\boldsymbol{u}^0 + \mathcal{O}\left(\tau^3\right) \\
&= \left(\boldsymbol{I} + \tau\,\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right) + \frac{1}{2}\cdot\tau^2\,\boldsymbol{P}^2\left(\boldsymbol{u}^{1/2}\right)\right)\boldsymbol{u}^0 + \mathcal{O}\left(\tau^3\right)\,,
\end{aligned} \tag{5.36}$$

where we have used the predictor

$$\boldsymbol{u}^{1/2} \;=\; \boldsymbol{u}^0 \;+\; \frac{\tau}{2} \cdot \boldsymbol{P}\left(\boldsymbol{u}^0\right)\, \boldsymbol{u}^0 \,. \tag{5.37}$$

Note that we have assumed

$$\boldsymbol{P}\left(\boldsymbol{u}^{1/2} + \mathcal{O}(\tau)\right) \;=\; \boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right) \;+\; \mathcal{O}(\tau)\;, \tag{5.38}$$

which is equivalent to the differentiability of $\boldsymbol{P}(\boldsymbol{u})$ with respect to $\boldsymbol{u}$.

Thus, a second order approximation can be achieved, if we apply an extrapolation scheme with the matrix $\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)$. More precisely, based on the given data $\boldsymbol{u}^k$, we have to compute an intermediate solution $\boldsymbol{u}^{k+1/2}$, evaluate the corresponding matrix, and then use the scheme (5.10) with $\boldsymbol{P}\left(\boldsymbol{u}^{k+1/2}\right)$ instead of $\boldsymbol{A_h}$.

5.3 Implementation

The summary of the general FED extrapolation scheme is illustrated in Fig. 5.3. As mentioned above, it is based on a predictor-corrector method by means of the midpoint rule. To this end, we have to compute two different sequences of time step sizes. The first one is used in an FED cycle that yields the predictor solution. After the evaluation of the corresponding nonlinearities, we apply the FED extrapolation scheme with the second sequence of the time step sizes to get a more accurate solution. Since the scheme is completely based on FED, it is also suited for parallel computing. However, in this context, a problem is that the computation of $\boldsymbol{u}_2^{\cdot}$ requires more effort and one has to wait for the extrapolation step.

5.4 Numerical Experiments

We reconsider the fingerprint enhancement illustrated in Fig. 3.22 for the numerical experiments. Our goal is to find out whether the predictor-corrector FED extrapolation scheme performs better than the already presented numerical methods. On the one hand it should be more accurate because of the higher order approximation, but on the other hand it requires a higher computational effort, since we have to compute both a predictor and two further results for the linear combination. As we have already stated in the previous Chapter 4, the accuracy of the predictor is very important. To this end, it is likely that the whole extrapolation scheme can be improved if we replace the FED scheme computing the predictor by the

1. **Input Data**:
   - image f, stopping time T, number M_c of outer cycles, and explicit step size limit $\tau_{\lim}$
   - for the predictor steps: number M_p of outer cycles
2. **Initialisation**:
   (a) Compute the smallest integer n_c and define the time adjustment factor $q_c \leq 1$ such that the diffusion time of one cycle is equal to $\frac{T}{M_c}$.
   (b) For the predictor steps: Determine $n_p \in \mathbb{N}$ as well as q_p to reach the overall step size $\frac{T}{2\,M_c M_p}$ for one cycle.
   (c) Choose suitable orderings for the time step sizes.
   (d) Define the initial value $\boldsymbol{u}^0 := \boldsymbol{f}$.
3. **Filtering Loop** $(k = 0, ..., M_c - 1)$:
   (a) Compute the predictor $\boldsymbol{u}^{k+1/2}$ with an FED scheme using M_p cycles.
   (b) Evaluate the diffusion matrix $\boldsymbol{P}\left(\boldsymbol{u}^{k+1/2}\right)$.
   (c) Perform the corrector step by applying the FED extrapolation scheme (5.10) with $\boldsymbol{P}\left(\boldsymbol{u}^{k+1/2}\right)$ to obtain $\boldsymbol{u}^{k+1}$.

Figure 5.3: Predictor-corrector FED extrapolation scheme.

FED Runge-Kutta method that allows inner updates. Some results are given in Table 5.2. Note that we have used a strategy with only one outer step for the computation of the predictor. Indeed, the use of some inner updates for the predictor computation massively improves the accuracy of the extrapolation scheme, due to the improvement of the quality of the predictor. Table 5.2 shows that e.g. two inner updates can reduce the MSE by a factor of up to seven compared to the standard method without any inner update. In this context, Fig. 5.4(a) also illustrates the superior efficiency of the approaches that use inner updates.

Compared to the results of the predictor-corrector FED Runge-Kutta schemes in Table 4.2, the extrapolation schemes are mostly better. However, the difference is not so large despite the better approximation. This might be due to the numerical error that appears for the replacement of $\boldsymbol{P}(\boldsymbol{u}^0)$ by $\boldsymbol{P}\left(\boldsymbol{u}^{1/2}\right)$ in the Taylor expansion from Eq. (5.36). Figure 5.4(b)

Table 5.2: MSE-Comparison of predictor-corrector FED extrapolation (PCE) schemes with different numbers of updates for the predictor computation.

cycles	**MSE**		
	PCE (1,0)	**PCE (1,1)**	**PCE (1,2)**
1	29.950	19.679	15.319
2	14.756	9.330	5.705
3	10.735	6.137	2.885
4	8.698	3.362	1.705
5	7.313	2.162	1.099
10	2.040	0.542	0.309
25	0.214	0.070	0.044

shows a comparison with respect to the efficiency. On the one hand the extrapolation scheme with two inner updates for the predictor computation outperforms the usual FED scheme, but on the other hand it is less efficient than the predictor-corrector FED Runge-Kutta scheme with two inner updates. Thus, the additional effort for the extrapolation step unfortunately does not pay off.

5.5 Summary

In this chapter we have successfully embedded the FED scheme in an extrapolation framework based on the well-known Richardson extrapolation. It is easy to understand, and the additional implementation effort is only marginal, because one has to compute a linear combination of two first order solutions, i.e. existing code can be used.

However, this linear combination is nonconvex, which means that its stability can not derived from the stability of the first order results. To this end, we have shown that the FED extrapolation scheme is stable in the Euclidean norm. Moreover, we have presented an example demonstrating the difficulties regarding an STS extrapolation scheme, where the stability depends on the choice of the damping parameter.

For nonlinear problems we have considered the predictor-corrector strategy that we have also used in Chapter 4. In this context, it is interesting to compare such an extrapolation method to the predictor-corrector FED Runge-Kutta scheme.

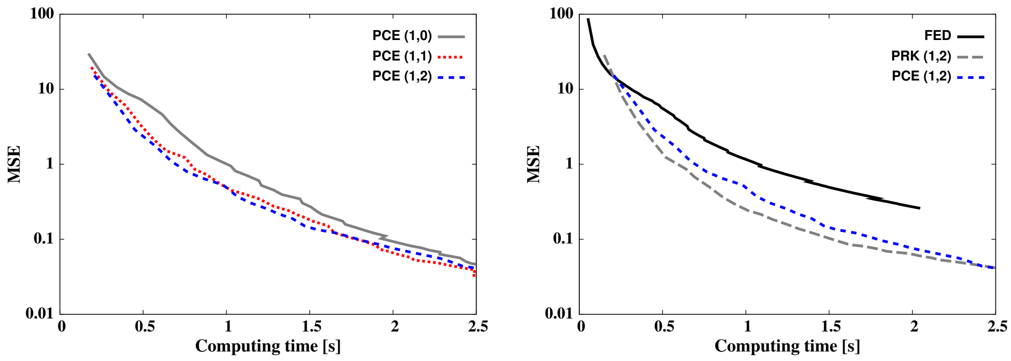


Figure 5.4: Computing time versus MSE for predictor-corrector FED extrapolation schemes. **(a) Left:** With different update strategies. **(b) Right:** Comparison with the usual FED and the predictor-corrector FED Runge-Kutta scheme.

This comparison has been done in the experimental section. Although the FED extrapolation scheme is more accurate, it can not beat the efficiency of the predictor-corrector method from Chapter 4. However, as before, it is possible to improve both the accuracy and efficiency, if we increase the number of inner updates for the predictor step. With the help of this, the extrapolation method becomes more efficient than the usual cyclic FED approach.

Chapter 6

Conclusions and Outlook

A conclusion is the place where you get tired of thinking.
Arthur Bloch

In this chapter, we conclude the thesis and discuss some aspects for future work.

6.1 Conclusions

This thesis has dealt with efficient explicit methods for solving parabolic and elliptic problems in PDE-based image analysis. They have been derived in the context of an interesting connection between linear, symmetric filters and explicit diffusion schemes with varying time step sizes. In this context, we have applied our theory by means of some examples like e.g. the binomial, the maximum variance or the box filter kernel. While 1-D binomial filters can be interpreted as an explicit linear diffusion scheme with a specific, constant time step size, the maximum variance kernel implies a method with very efficient cycles of varying time step sizes. They are related to the roots of Chebyshev polynomials, and some of them significantly violate the stability restriction for 1-D explicit linear diffusion schemes. This allows diffusion times that depend quadratically on the cycle length. However, the high efficiency is achieved at the cost of a poor approximation for Gaussian kernels. The cycles of the box filter also consist of unstable time step sizes that are related to roots of Chebyshev polynomials. Although this box filter factorisation is less efficient, its diffusion time depends also quadratically on the length of the cycle, and the approximation quality with respect to Gaussian kernels is much better. Thus, the box filter factorisation provides a good trade-off between efficiency and approximation quality.

To this end, we have decided to consider the Fast Explicit Diffusion (FED) scheme that corresponds to the box filter kernel. It can be transferred to arbitrary parabolic scenarios, and provides schemes that are stable in the Euclidean norm. Furthermore, FED is very easy to implement, i.e. existing explicit schemes have to be modified with only few additional code lines.

By means of our filter factorisation, we have found out that the already existing Super Time Stepping (STS) method is related to the maximum variance filter kernel. Because of its sensitivity concerning high frequency components such as noise, STS requires the optimisation of an additional damping parameter, whereas FED does not need such a parameter. This means that we have eliminated the damping parameter by means of our signal processing approach.

Moreover, the idea of varying time step sizes has led us to a Jacobi over-relaxation (JOR) method with varying relaxation parameters that are based on the FED time step sizes: Fast-Jacobi. It is related to Richardson's cyclic method and can provide a much better convergence than the usual JOR with a constant relaxation parameter. Similarly to FED, the implementation is easy, since the underlying JOR method can be used as a black box solver.

However, due to the unstable time step sizes or large relaxation parameters, the methods require rearrangements of the step size or relaxation parameter sequences. Such strategies have already been proposed in the context of STS and Richardson's method. We have chosen three approaches that also work well with both FED and Fast-Jacobi: κ-cycles, Leja, and Lebedev-Finogenov ordering. Since these rearrangements yield unstable intermediate results, the solution of nonlinear problems is only reasonable if the nonlinearities are not updated during a cycle.

Our numerical experiments have demonstrated that FED is very efficient and outperforms additive operator splitting (AOS) as well as semi-implicit schemes for the solution of nonlinear parabolic problems. To efficiently solve elliptic problems like e.g. image inpainting by means of a parabolic evolution, we have used FED in combination with a cascadic coarse-to-fine strategy (CFED). This works well for problems with constant coefficients such as biharmonic inpainting. However, for elliptic problems with (strongly) varying coeffcients, the Fast-Jacobi method is more efficient than a parabolic approach with $t \to \infty$. To show the benefit of parallel implementations, we have presented a GPU implementation for 3-D anisotropic range image integration. In our case, it is up to 140 times faster than a sequential CPU implementation of the Fast-Jacobi method.

Besides the filter factorisation, we have derived a recursion relation for box filters. This allows us to rewrite both the cyclic FED scheme and the Fast-Jacobi method such that a rearrangement of time steps or relaxation parameters is not necessary. In this context, the new FED Runge-Kutta scheme and the semi-iterative Fast-Jacobi method additionally achieve stable intermediate results that guarantee reasonable inner updates for nonlinear problems.

Our numerical experiments have demonstrated the possible advantages of these updates with respect to accuracy. Together with a predictor-corrector strategy for nonlinear parabolic problems, they can improve the efficiency in a way such that the usual cyclic FED scheme is inferior. We should mention that the predictor-corrector strategy could also be used in combination with cyclic FED schemes to improve the efficiency, which means that they are not limited to FED Runge-Kutta schemes. Moreover, the semi-iterative Fast-Jacobi solver can also benefit from inner updates during an outer iteration step. However, for both parabolic and elliptic problems, up to three updates are already sufficient, and further updates might deteriorate the efficiency.

To improve the first order consistency of FED schemes, one can combine them with the well-known Richardson extrapolation that yields second order consistency. We have proven that the resulting FED extrapolation scheme is stable in the Euclidean norm. In this context, we have found out that the stability of a corresponding STS extrapolation scheme depends on the damping parameter. This is a clear disadvantage compared to FED.

Although the consistency order of the FED extrapolation scheme is higher, the experiments have demonstrated the inferior efficiency with respect to the first order predictor-corrector methods. This is mainly due to the additional computational effort coming from the extrapolation framework.

6.2 Outlook

In this thesis, we have applied the cyclic methods FED as well as Fast-Jacobi to solve some parabolic and elliptic problems. However, FED or Fast-Jacobi can also be used to solve other problems that are beyond the scope of this work. Meanwhile, they have been used to speed-up an optic flow method with the help of a GPU-based implementation [168]: Although a full multigrid solver is about 20% faster than FED on the CPU, a parallel FED algorithm on the GPU can be more than 10 times faster and is additionally much less complicated to implement. Other efficient parallel FED schemes have been introduced in the context of medical applications, i.e. the modelling of tumour growth [96], biomedical image registration [122], or for variational depth-from-defocus [10]. Because of their simplicity, cyclic methods can be easily implemented as well as optimised on mobile platforms, as it has been done in [94]. Moreover, the idea of FED-based varying parameters has been employed for gradient descent reprojection in convex optimisation [124]. These publications indicate that the proposed cyclic methods already cover a variety of applications, and it is likely that they will be used in the context of further applications, due to their simplicity as well as suitability for parallel processing.

Regarding the dimension of the data for image enhancement, we have restricted ourselves to grey value images. However, it is possible to apply FED to colour images or matrix-valued data sets coming from e.g. diffusion tensor magnetic resonance imaging that is important for medical applications. Since semi-implicit schemes or, in general, the solution of equation systems can become cumbersome for higher-dimensional tasks due to the possible large neighbourhood structure, the benefit of explicit approaches such as FED is expected to increase even further. Moreover, instead of using only finite differences, it would be also interesting to see how FED works in combination with finite element methods (cf. e.g. [107]).

In the numerical experiments, we have observed that the rearrangement proposed by Lebedev and Finogenov [89] provides the largest robustness with respect to numerical rounding errors. However, it is tailored to cycle lengths $n = 2^p$. To avoid that the computational effort is possibly doubled, one could think about strategies with varying cycle lengths n_i that satisfy $n_i = 2^{p_i}$. More precisely, if a certain stopping time requires e.g. the minimum cycle length $n = 65$, one could apply two cycles with $n_1 = 64$ and $n_2 = 16$, respectively, instead of one cycle with length 128. In this context, it would be also interesting to compare the accuracy of cyclic methods with and without varying cycle lengths.

Besides these practical aspects, we would like to mention some interesting theoretical issues for future work: In Chapter 2 we have considered only filters whose amplification factors are related to Chebyshev polynomials. Recently, the so-called Legendre polynomials have been used to construct novel explicit methods [99]. Moreover, there is also the possibility to avoid orthogonal polynomials [46]. With the help of our proposed framework, one could derive the corresponding linear filters and analyse them with respect to e.g. approximation quality. For our cyclic scheme, we have shown that the natural sequence of FED time steps yields intermediate results being stable in the sense of the Euclidean norm. However, for the usual 1-D linear case, we have the conjecture that they are even stable with respect to a stronger stability criterion, namely the maximum-minimum principle which requires non-negative filter weights. For more general problems, we have already seen by means of a counter example for the L^∞-stability criterion in Chapter 3 that FED can violate the positivity of the filter weights. In this context, it would be interesting to analyse whether there are diffusivity functions that guarantee L^∞-stable isotropic diffusion processes. Another issue is the application of FED to problems that are related to non-symmetric matrices like for instance osmosis processes [69]. In this case, it is necessary to modify the methods such that they guarantee stability and reasonable results. Such approaches have been developed for STS in e.g. [66, 67]. Thus, one could try to transfer these ideas to the FED scheme.

Overall, we are optimistic that our proposed explicit methods become widely accepted in the field of PDE-based image analysis. In the past, these approaches have never been the most popular methods for the numerical solution of PDEs. With the advent of low cost parallel computing hardware and the growing demand for simple and widely applicable algorithms, the situation has changed substantially. It seems that explicit methods inspired by Richardson's seminal work [111] finally get the merits they deserve.

List of Abbreviations

AOS	additive operator splitting
CED	coherence-enhancing diffusion
CFED	cascadic Fast Explicit Diffusion
CFJ	cascadic Fast-Jacobi
CG	conjugate gradient
CMR	cascadic modified Richardson
CPU	central processing unit
DFT	discrete Fourier transform
DTFT	discrete time Fourier transform
EBF	extended box filter
EED	edge-enhancing diffusion
FED	Fast Explicit Diffusion
FJ	Fast-Jacobi
FEDRK	FED Runge-Kutta
GPU	graphics processing unit
JOR	Jacobi over-relaxation
MSE	mean squared error
ODE	ordinary differential equation
PCE	predictor-corrector FED extrapolation
PCG	preconditioned conjugate gradient
PDE	partial differential equation
PRK	predictor-corrector FED Runge-Kutta

SIFJ	semi-iterative Fast-Jacobi
SOR	successive over-relaxation
STS	Super Time Stepping
TV	total variation

Bibliography

[1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (9th printing)*, chapter "Orthogonal Polynomials" (Ch. 22), pages 771–802. Dover, New York, 1972.

[2] P. Albrecht. The Runge-Kutta theory in a nutshell. *SIAM Journal on Numerical Analysis*, 33(5):1712–1735, 1996.

[3] V. Alexiades. Overcoming the stability restriction of explicit schemes via super-time-stepping. In *Proceedings of Dynamic Systems and Applications*, volume 2, pages 39–44, Atlanta, Georgia, May 1995.

[4] V. Alexiades, G. Amiez, and P.-A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Communications in Numerical Methods in Engineering*, 12:31–42, 1996.

[5] R. S. Anderssen and G. H. Golub. Richardson's non-stationary matrix iterative procedure. Technical Report STAN-CS-72-304, Computer Science Department, Stanford University, August 1972.

[6] F. Andreu, C. Ballester, V. Caselles, and J. M. Mazón. Minimizing total variation flow. *Differential and Integral Equations*, 14(3):321–360, March 2001.

[7] F. Andreu, V. Caselles, J. I. Diaz, and J. M. Mazón. Qualitative properties of the total variation flow. *Journal of Functional Analysis*, 188(2):516–547, February 2002.

[8] D. Barash, M. Israeli, and R. Kimmel. An accurate operator splitting scheme for nonlinear diffusion filtering. In M. Kerckhove, editor, *Scale-Space and Morphology in Computer Vision*, volume 2106 of

Lecture Notes in Computer Science, pages 281–289. Springer, Berlin, 2001.

[9] A. Beck and M. Teboulle. Fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[10] R. Ben-Ari and G. Raveh. Variation depth from defocus in real-time. In *Computer Vision Workshops, 2011 IEEE International Conference on Computer Vision*, pages 522–529, Barcelona, Spain, 2011.

[11] L. Blanc-Féraud, P. Charbonnier, G. Aubert, and M. Barlaud. Nonlinear image processing: modelling and fast algorithm for regularization with edge detection. In *Proc. IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 474–477, 1995.

[12] F. Bornemann and P. Deuflhard. The cascadic multigrid method for elliptic problems. *Numerische Mathematik*, 75:135–152, 1996.

[13] R. N. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, New York, 1999.

[14] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[15] C. Brezinski. Extrapolation algorithms and Padé approximations: A historical survey. *Applied Numerical Mathematics*, 20:299–318, 1996.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial.* SIAM, Philadelphia, second edition, 2000.

[17] A. Bruhn, T. Jacob, M. Fischer, T. Kohlberger, J. Weickert, U. Brüning, and C. Schnörr. High performance cluster computing with 3-D nonlinear diffusion filters. *Real-Time Imaging*, 10(1):41–51, February 2004.

[18] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optic flow computation in real-time. *IEEE Transactions on Image Processing*, 14(5):608–615, 2005.

[19] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* Wiley-Interscience, New York, 1987.

[20] T. Butz. *Fourier Transformation for Pedestrians.* Springer, Heidelberg, 2006.

[21] D. Calvetti and L. Reichel. Adaptive Richardson iteration based on Leja points. *Journal of Computational and Applied Mathematics*, 71:267–286, 1996.

[22] D. Calvetti and L. Reichel. An adaptive Richardson iteration method for indefinite linear systems. *Numerical Algorithms*, 12:125–149, 1996.

[23] D. Calvetti and L. Reichel. On the evaluation of polynomial coefficients. *Numerical Algorithms*, 33(1–4):153–161, 2003.

[24] K. R. Castleman. *Digital Image Processing.* Prentice Hall, Englewood Cliffs, 1996.

[25] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 32:1895–1909, 1992.

[26] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1–2):89–97, 2004.

[27] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[28] T. F. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in Total Variation image restoration. *SIAM Journal on Numerical Analysis*, 36(2):354–367, 1999.

[29] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proc. 1994 IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 168–172, Austin, TX, November 1994. IEEE Computer Society Press.

[30] P. L. Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mémoires des Savants étrangers présentés à l'Acadéemie de Saint-Pétersbourg*, 7:539–586, 1854. (in French).

[31] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic geometric diffusion in surface processing. In *Proc. 2000 IEEE Conference on Visualization*, volume 1, pages 397–405, Salt Lake City, UT, October 2000.

[32] D. Colton. *Partial Differential Equations*. Random House, New York, 1998.

[33] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[34] G.-H. Cottet and L. Germain. Image processing through reaction combined with nonlinear diffusion. *Mathematics of Computation*, 61:659–673, 1993.

[35] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928. (in German).

[36] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 96*, volume 3, pages 303–312, 1996.

[37] P. Deuflhard. Cascadic conjugate gradient methods for elliptic partial differential equations: Algorithm and numerical results. In D. Keyes and J. Xu, editors, *Proceedings of the 7th International Conference on Domain Decomposition Methods*, volume 180 of *Contemporary Mathematics*, pages 29–42. AMS, 1994.

[38] S. Didas and J. Weickert. Integrodifferential equations for continuous multiscale wavelet shrinkage. *Inverse Problems and Imaging*, 1(1):47–62, 2007.

[39] P. G. L. Dirichlet. Sur la convergence des séries trigonométriques qui servent à represénter une fonction arbitraire entre des limites données. *Journal für die reine und angewandte Mathematik*, 4:157–169, 1829. (in French).

[40] J. Douglas and B. F. Jones. On predictor-corrector methods for nonlinear parabolic differential equations. *Journal of the Society of Industrial and Applied Mathematics*, 11(1):195–204, 1963.

[41] O. Drblíková and K. Mikula. Convergence analysis of finite volume scheme for nonlinear tensor anisotropic diffusion in image processing. *SIAM Journal on Numerical Analysis*, 46(1):37–60, 2007.

[42] J. J. Droux. Three-dimensional numerical simulation of solidification by an improved explicit scheme. *Computer Methods in Applied Science and Engineering*, 85:57–74, 1991.

[43] A. Edrei. Sur les déterminants récurrents et les singularités d′une fonction donnée par son developpement de Taylor. *Compositio Mathematica*, 7:20–88, 1939. (in French).

[44] M. Eiermann and W. Niethammer. On the construction of semiiterative methods. *SIAM Journal on Numerical Analysis*, 20(6):1153–1160, 1983.

[45] M. Eiermann, W. Niethammer, and R. S. Varga. A study of semiiterative methods for nonsymmetric systems of linear equations. *Numerische Mathematik*, 47:505–533, 1985.

[46] K. Eriksson, C. Johnson, and A. Logg. Explicit time-stepping for stiff ODEs. *SIAM Journal on Scientific Computing*, 25(4):1142–1157, 2004.

[47] E. Esser, X. Zhang, and T. F. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences*, 3(4):1015–1046, 2010.

[48] B. Fischer and L. Reichel. A stable Richardson iteration method for complex linear systems. *Numerische Mathematik*, 54:225–242, 1988.

[49] H. Fischer. *A History of the Central Limit Theorem: From Classical to Modern Probability Theory*. Springer, 2010.

[50] D. A. Flanders and G. Shortley. Numerical determination of fundamental modes. *Journal of Applied Physics*, 21:1326–1332, 1950.

[51] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, Interlaken, Switzerland, June 1987.

[52] W. L. Frank. Solution of linear systems by Richardson's method. *Journal of the ACM*, 7(3):274–286, 1960.

[53] J. N. Franklin. Numerical stability in digital and analogue computation for diffusion problems. *Journal of Mathematical Physics*, 37(4):305–315, 1959.

[54] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2–3):255–269, July 2008.

[55] W. B. Gearhart and H. S. Shultz. The function sin(x)/x. *The College Mathematics Journal*, 21(2):90–99, 1990.

[56] W. Gentzsch. Numerical solution of linear and non-linear parabolic differential equations by a time discretisation of third order accuracy. In E. H. Hirschel, editor, *Proc. Third GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 109–117. Vieweg, Braunschweig, 1979.

[57] W. Gentzsch and A. Schlüter. Über ein Einschrittverfahren mit zyklischer Schrittweitenänderung zur Lösung parabolischer Differentialgleichungen. *ZAMM, Zeitschrift für Angewandte Mathematik und Mechanik*, 58(7):T415–T416, 1978. (in German).

[58] S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izvestiya Akademii Nauk SSSR*, 7:749–754, 1931. (in German).

[59] T. Goldstein and S. Osher. The Split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[60] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods, Part I. *Numerische Mathematik*, 3(1):147–156, 1961.

[61] A. R. Gourlay and J. Morris. The extrapolation of first order methods for parabolic partial differential equations II. *SIAM Journal on Numerical Analysis*, 17(5):641–655, 1980.

[62] A. R. Gourlay and J. Morris. Linear combinations of generalized Crank–Nicolson schemes. *IMA Journal of Numerical Analysis*, 1:347–357, 1981.

[63] S. Grewenig. Extrapolationsverfahren für nichtlineare zweidimensionale Diffusionsprobleme. Diploma Thesis, Department of Mathematics, Saarland University, 2008. (in German).

[64] A. Guillou and B. Lago. Domaine de stabilité associé aux formules d′intégration numérique d′équations différentielles, à pas séparés et à

pas liés. Recherche de formules à grand rayon de stabilité. In *Premier congrès de ĺ association francaise de calcul (AFCAL)*, pages 43–56, Grenoble, 1961. (in French).

[65] K. F. Gurski. A stability and cost study of explicit super and dyadic time stepping for stiff nonsymmetric problems. In *AIP Conference Proceedings: Advances in Mathematical and Computational Methods*, volume 1368, pages 239–242, Waterloo (Canada), 2011.

[66] K. F. Gurski and S. O'Sullivan. An explicit super-time-stepping scheme for non-symmetric parabolic problems. In *AIP Conference Proceedings: International Conference of Numerical Analysis and Applied Mathematics*, volume 1281, pages 761–764, Rhodes (Greece), 2010.

[67] K. F. Gurski and S. O'Sullivan. A stability study of a new explicit numerical scheme for a system of differential equations with a large skew-symmetric component. *SIAM Journal on Numerical Analysis*, 49(1):368–386, 2011.

[68] W. Hackbusch. *Multigrid Methods and Applications*. Springer, New York, 1985.

[69] K. Hagenburg, M. Breuß, J. Weickert, and O. Vogel. Novel schemes for hyperbolic PDEs using osmosis filters from visual computing. In A. M. Bruckstein and B. ter Haar Romeny, editors, *Scale Space and Variational Methods*, volume 6667 of *Lecture Notes in Computer Science*, pages 532–543, Ein-Gedi, Israel, May 2011. Springer, Berlin, Germany.

[70] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, second edition, 2000.

[71] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, Berlin, second edition, 1996.

[72] M. Hanke. Accelerated Landweber iterations for the solution of ill-posed equations. *Numerische Mathematik*, 60(1):341–373, 1991.

[73] G. Hellwig. *Partial Differential Equations*. Teubner, Stuttgart, 1977.

[74] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[75] K. Höllig. Existence of infinitely many solutions for a forward-backward heat equation. *Transactions of the American Mathematical Society*, 278:299–316, 1983.

[76] C. Huygens. De circuli magnitudine inventa, 1654.

[77] W. M. Wells III. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:234–239, 1986.

[78] T. Iijima. Basic theory of pattern observation. In *Papers of Technical Group on Automata and Automatic Control.* IECE, Japan, December 1959. (in Japanese).

[79] T. Iijima. Basic theory on normalization of pattern (in case of typical one-dimensional pattern). *Bulletin of the Electrotechnical Laboratory*, 26:368–388, 1962. (in Japanese).

[80] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, 1989.

[81] S. Kichenassamy. The Perona–Malik paradox. *SIAM Journal on Applied Mathematics*, 57(5):1328–1342, 1997.

[82] I. P. E. Kinnmark and W. G. Gray. One-step integration methods of third-fourth order accuracy with large hyperbolic stability limits. *Mathematics and Computers in Simulation*, 16(3):181–184, 1984.

[83] I. P. E. Kinnmark and W. G. Gray. One-step integration methods with maximum stability regions. *Mathematics and Computers in Simulation*, 16(2):87–92, 1984.

[84] Z. V. Kovarik. Spectrum localization in Banach spaces I. *Linear Algebra and its Applications*, 8:225–236, 1974.

[85] J. D. Lambert. *Computational Methods in Ordinary Differential Equations.* Wiley, New York, 1973.

[86] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49(1):33–53, 1952.

[87] L. Landweber. An iteration formula for Fredholm integral equations of the first kind. *American Journal of Mathematics*, 73(3):615–624, 1951.

[88] J. D. Lawson and J. Morris. The extrapolation of first order methods for parabolic partial differential equations I. *SIAM Journal on Numerical Analysis*, 15(6):1212–1224, 1978.

[89] V. I. Lebedev and S. A. Finogenov. Ordering of the iterative parameters in the cyclical Chebyshev iterative method. *USSR Computational Mathematics and Mathematical Physics*, 11(2):155–170, 1971.

[90] F. Leja. Sur certaines suits liées aux ensembles plans et leur application à la representation conforme. *Annales Polonici Mathematici*, 4:8–13, 1957. (in French).

[91] R. J. LeVeque. *Numerical Methods for Conservation Laws.* Birkhäuser, Basel, 2nd edition, 2005.

[92] R. W. Lewis, I. Masters, and J. T. Cross. Automatic timestep selection for the super-time-stepping acceleration on unstructured grids using object-oriented programming. *Communications in Numerical Methods in Engineering*, 13:249–260, 1997.

[93] T. Lu, P. Neittaanmäki, and X.-C. Tai. A parallel splitting up method and its application to Navier-Stokes equations. *Applied Mathematics Letters*, 4(2):25–29, 1991.

[94] A. Luxenburger, H. Zimmer, P. Gwosdek, and J. Weickert. Fast PDE-based image analysis in your pocket. In A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 544–555. Springer, Berlin, Germany, 2011.

[95] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. Edge-based image compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition*, 44(9):1859–1873, September 2011.

[96] A. Mang, A. Toma, T. A. Schütz, S. Becker, and T. M. Buzug. Eine effiziente Parallel-Implementierung eines stabilen Euler-Cauchy-Verfahrens für die Modellierung von Tumorwachstum. In T. Tolxdorff, T. M. Deserno, H. Handels, and H.-P. Meinzer, editors, *Bildverarbeitung für die Medizin 2012*, Informatik aktuell, pages 63–68. Springer, Berlin, Germany, March 2012. (in German).

[97] W. Markoff. Über Polynome, die in einem gegebenen Intervall möglichst wenig von Null abweichen. *Mathematische Annalen*, 77:213–258, 1916. (in German, original 1892 in Russian).

[98] A. Meister. *Numerik linearer Gleichungssysteme*. Vieweg, Braunschweig, third edition, 2008.

[99] C. D. Meyer, D. S. Balsara, and T. Aslam. A second-order accurate super timestepping formulation for anisotropic thermal conduction. *Monthly Notices of the Royal Astronomical Society*, 422(3):2102–2115, 2012.

[100] C. Moler and C. van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[101] Y. E. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[102] G. Opfer and G. Schober. Richardson's iteration for nonsymmetric matrices. *Linear Algebra and its Applications*, 58:343–361, 1984.

[103] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, Englewood Cliffs, second edition, 1999.

[104] A. Ostrowski. Über Eigenwerte von Produkten Hermitescher Matrizen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 23:60–68, 1959. (in German).

[105] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *Proc. IEEE Computer Society Workshop on Computer Vision*, pages 16–22, Miami Beach, FL, November 1987. IEEE Computer Society Press.

[106] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[107] T. Preußer and M. Rumpf. An adaptive finite element method for large scale image processing. *Journal of Visual Communication and Image Representation*, 11(2):183–195, June 2000.

[108] T. Preußer and M. Rumpf. A level set method for anisotropic geometric diffusion in 3D image processing. *SIAM Journal on Applied Mathematics*, 62(5):1772–1793, 2002.

[109] L. Reichel. Newton interpolation at Leja points. *BIT Numerical Mathematics*, 30(2):332–346, 1990.

[110] L. Reichel. The application of Leja points to Richardson iteration and polynomial preconditioning. *Linear Algebra and its Applications*, 154–156:389–414, 1991.

[111] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equation, with an application to the stresses in a masonry dam. *Transactions of the Royal Society of London*, Ser. A(210):307–357, 1910.

[112] L. F. Richardson and J. A. Gaunt. The deferred approach to the limit. *Transactions of the Royal Society of London*, Ser. A(226):299–361, 1927.

[113] W. Romberg. Vereinfachte numerische integration. *The Royal Norwegian Society of Sciences and Letters*, 28(7):30–36, 1955. (in German).

[114] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[115] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, second edition, 2003.

[116] È. N. Sarmin and L. A. Chudov. On the stability of the numerical integration of systems of ordinary differential equations arising in the use of the straight line method. *USSR Computational Mathematics and Mathematical Physics*, 3(6):1537–1543, 1963.

[117] V. K. Saul'yev. *Integration of Equations of Parabolic Type by the Method of Nets.* Pergamon, Oxford, 1964. (original 1960 in Russian).

[118] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[119] O. Scherzer and J. Weickert. Relations between regularization and diffusion filtering. *Journal of Mathematical Imaging and Vision*, 12(1):43–63, February 2000.

[120] W. E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations.* Academic Press, San Diego, CA, 1991.

[121] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 452–461, Berlin, 2009. Springer.

[122] A. Schmidt-Richberg, J. Ehrhardt, R. Werner, and H. Handels. Fast Explicit Diffusion for registration with direction-dependent regularization. In B. M. Dawant, G. E. Christensen, J. M. Fitzpatrick, and D. Rueckert, editors, *Biomedial Image Registration*, volume 7359 of *Lecture Notes in Computer Science*, pages 220–228, Berlin Heidelberg, 2012. Springer.

[123] C. Schroers, H. Zimmer, L. Valgaerts, A. Bruhn, O. Demetz, and J. Weickert. Anisotropic range image integration. In A. Prinz, T. Pock, H. Bischof, and F. Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 73–82, Berlin, 2012. Springer.

[124] S. Setzer, G. Steidl, and J. Morgenthaler. A cyclic projected gradient method. *Computational Optimization and Applications*, 54:417–440, 2013.

[125] G. Shortley. Use of Tschebyscheff-polynomial operators in the numerical solution of boundary-value problems. *Journal of Applied Physics*, 24(4):392–396, 1953.

[126] G. D. Smith. *Numerical Solution of Partial Differential Equations.* Oxford University Press, 1985.

[127] B. P. Sommeijer. Increasing the real stability boundary of explicit methods. *Computers and Mathematics with Applications*, 19(6):37–149, 1990.

[128] G. Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, 1999.

[129] M. E. Taylor. *Pseudodifferential Operators.* Princeton University Press, Princeton, New Jersey, 1981.

[130] M. E. Taylor. *Partial Differential Equations I - Basic Theory.* Springer, New York, 1996.

[131] L. H. Thomas. Elliptic problems in linear difference equations over a network. Technical report, Watson Scientific Computing Laboratory, Columbia University, New York, NJ, 1949.

[132] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[133] L. Valgaerts, A. Bruhn, and J. Weickert. A variational model for the joint recovery of the fundamental matrix and the optical flow. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 314–324. Springer, Berlin, 2008.

[134] P. J. van der Houwen. Explicit Runge-Kutta formulas with increased stability boundaries. *Numerische Mathematik*, 20:149–164, 1972.

[135] P. J. van der Houwen. On the time-integration of parabolic differential equations. In G. A. Watson, editor, *Numerical Analysis*, volume 912 of *Lecture Notes in Mathematics*, pages 157–168. Springer, 1982.

[136] P. J. van der Houwen. The development of Runge-Kutta methods for partial differential equations. *Applied Numerical Mathematics*, 20:261–272, 1996.

[137] P. J. van der Houwen and B. P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *ZAMM, Zeitschrift für Angewandte Mathematik und Mechanik*, 60:479–485, 1980.

[138] R. S. Varga. A comparison of the successive overrelaxation method and semi-iterative methods using Chebyshev polynomials. *Journal of the Society for Industrial and Applied Mathematics*, 5:39–46, 1957.

[139] R. S. Varga. *Matrix Iterative Analysis.* Prentice Hall, Englewood Cliffs, 1962.

[140] J. G. Verwer, W. H. Hundsdorfer, and B. P. Sommeijer. Convergence properties of the Runge-Kutta-Chebyshev method. *Numerische Mathematik*, pages 157–178, 1990.

[141] J. G. Verwer, B. P. Sommeijer, and W. H. Hundsdorfer. RKC time-stepping for advection-diffusion-reaction problems. *Journal of Computational Physics*, 201:61–79, 2004.

[142] R. Vichnevetsky. New stability theorems concerning one-step numerical methods for ordinary differential equations. *Mathematics and Computers in Simulation*, 25:199–205, 1983.

[143] C. R. Vogel and M. E. Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17:227–238, 1996.

[144] H. Voss and U. Eckhardt. Linear convergence of generalized Weiszfeld's method. *Computing*, 25(3):243–251, 1980.

[145] J. Weickert. Anisotropic diffusion filters for image processing based quality control. In A. Fasano and M. Primicerio, editors, *Proc. Seventh European Conference on Mathematics in Industry*, pages 355–362. Teubner, Stuttgart, 1994.

[146] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing Supplement*, 11:221–236, 1996.

[147] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart, 1998.

[148] J. Weickert. Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 31(2/3):111–127, April 1999.

[149] J. Weickert. A real-time algorithm for assessing inhomogeneities in fabrics. *Real-Time Imaging*, 5(1):15–22, February 1999.

[150] J. Weickert. Private communication, 2010.

[151] J. Weickert and B. Benhamouda. A semidiscrete nonlinear scale-space theory and its relation to the Perona–Malik paradox. In F. Solina, W. G. Kropatsch, R. Klette, and R. Bajcsy, editors, *Advances in Computer Vision*, pages 1–10. Springer, Wien, 1997.

[152] J. Weickert, S. Ishikawa, and A. Imiya. On the history of Gaussian scale-space axiomatics. In J. Sporring, M. Nielsen, L. Florack, and P. Johansen, editors, *Gaussian Scale-Space Theory*, volume 8 of *Computational Imaging and Vision*, pages 45–59. Kluwer, Dordrecht, 1997.

[153] J. Weickert, S. Ishikawa, and A. Imiya. Linear scale-space has first been proposed in Japan. *Journal of Mathematical Imaging and Vision*, 10(3):237–252, May 1999.

[154] J. Weickert and H. Scharr. A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance. *Journal of Visual Communication and Image Representation*, 13(1/2):103–118, 2002.

[155] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, March 1998.

[156] J. Weickert, K. J. Zuiderveld, B. M. ter Haar Romeny, and W. J. Niessen. Parallel implementations of AOS schemes: A fast way of nonlinear diffusion filtering. In *Proc. 1997 IEEE International Conference on Image Processing*, volume 3, pages 396–399, Santa Barbara, CA, October 1997.

[157] M. Welk, G. Steidl, and J. Weickert. Locally analytic schemes: A link between diffusion filtering and wavelet shrinkage. *Applied and Computational Harmonic Analysis*, 24:195–224, 2008.

[158] A. P. Witkin. Scale-space filtering. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, volume 2, pages 945–951, Karlsruhe, Germany, August 1983.

[159] K. Yosida. *Functional Analysis.* Springer, Berlin, sixth edition, 1995.

[160] D. M. Young. On Richardson's method for solving linear systems with positive definite matrices. *Journal of Mathematics and Physics*, 32:243–255, 1954.

[161] D. M. Young. A historical overview of iterative methods. *Computer Physics Communications*, 53:1–17, 1989.

[162] Yuan'Chzhao-Din. *Some difference schemes for the solution of the first boundary value problem for linear differential equations with partial derivatives.* PhD thesis, Moscow State University, 1958. (in Russian).

[163] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L^1 range image integration. In *Proc. Ninth International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007. IEEE Computer Society Press.

Own Publications

[164] S. Grewenig, J. Weickert, and A. Bruhn. From box filtering to Fast Explicit Diffusion. In M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, editors, *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 543–552. Springer, Berlin, Germany, 2010.

[165] S. Grewenig, J. Weickert, C. Schroers, and A. Bruhn. Cyclic schemes for PDE-based image analysis. Technical Report 327, Department of Mathematics and Computer Science, Saarland University, Saarbrücken, March 2013.

[166] S. Grewenig, S. Zimmer, and J. Weickert. Rotationally invariant similarity measures for nonlocal image denoising. *Journal of Visual Communication and Image Representation*, 22(2):117–130, February 2011.

[167] P. Gwosdek, S. Grewenig, A. Bruhn, and J. Weickert. Theoretical foundations of Gaussian convolution by extended box filtering. In A.M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 447–458, Ein-Gedi, Israel, 2011. Springer, Berlin, Germany.

[168] P. Gwosdek, H. Zimmer, S. Grewenig, A. Bruhn, and J. Weickert. A highly efficient GPU implementation for variational optic flow based on the Euler-Lagrange framework. In K. N. Kutulakos, editor, *Trends and Topics in Computer Vision*, volume 6554 of *Lecture Notes in Computer Science*, pages 372–383, Heraklion, Greece, 2012. Springer Berlin Heidelberg.

[169] N. Persch, A. Elhayek, M. Welk, S. Grewenig, K. Narr, A. Kraegeloh, and J. Weickert. Enhancing 3-D cell structures in confocal and STED microscopy: A joint model for interpolation, deblurring and anisotropic smoothing. Technical Report 321, Department of Mathematics and Computer Science, Saarland University, Saarbrücken, January 2013.

[170] L. Pizarro, P. Mrázek, S. Didas, S. Grewenig, and J. Weickert. Generalised nonlocal image smoothing. *International Journal of Computer Vision*, 90(1):62–87, October 2010.