Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 327

Cyclic Schemes for PDE-Based Image Analysis

Sven Grewenig, Joachim Weickert,
Christopher Schroers and Andrés Bruhn

Saarbrücken 2013

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 327
submitted: March 1, 2013

Cyclic Schemes for PDE-Based Image Analysis

Sven Grewenig
Mathematical Image Analysis Group, Dept. of Mathematics and Computer Science, Saarland University, Campus E1.7, 66123 Saarbrücken, Germany
grewenig@mia.uni-saarland.de

Joachim Weickert
Mathematical Image Analysis Group, Dept. of Mathematics and Computer Science, Saarland University, Campus E1.7, 66123 Saarbrücken, Germany
weickert@mia.uni-saarland.de

Christopher Schroers
Mathematical Image Analysis Group, Dept. of Mathematics and Computer Science, Saarland University, Campus E1.7, 66123 Saarbrücken, Germany
schroers@mia.uni-saarland.de

Andrés Bruhn
Intelligent Systems Group, Institute for Visualization and Interactive Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
bruhn@vis.uni-stuttgart.de

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax:	+ 49 681 302 4443
e-Mail:	preprint@math.uni-sb.de
WWW:	http://www.math.uni-sb.de/

Abstract

The simplest scheme for parabolic, diffusion-like partial differential equations (PDEs) in image analysis is given by the explicit finite difference discretisation. If a fixed time step size is used, this scheme is known to be inefficient due to a severe stability restriction. In this paper we show that a slight modication can boost the efficiency of the explicit scheme by several orders of magnitude. All one has to do in this novel Fast Explicit Diffusion (FED) scheme is to replace the originally fixed time step size by cycles of varying step sizes. We derive these cycles from a factorisation of symmetric smoothing filters, and we show that a box filter gives a favourable compromise between high efficiency and good smoothing properties.

However, such ideas are not restricted to parabolic problems only. They can also be adapted for solving elliptic PDEs that arise e.g. as Euler-Lagrange equations in variational image analysis. Also in this context, we propose a modification of the simplest iterative solver, namely the Jacobi method. This time, we introduce cycles of varying relaxation parameters that can be linked to our FED time step sizes. As a result, we obtain highly efficient methods for the elliptic case, so-called Fast-Jacobi (FJ) schemes.

Our novel FED and FJ schemes are applicable to a large range of PDE-based image analysis problems and can beat classical methods such as additive operator splittings and (semi-)implicit schemes. Applications include isotropic nonlinear diffusion filters with widely varying diffusivities as well as anisotropic diffusion methods for image filtering, inpainting and regularisation in computer vision. Moreover, they are equally suited for higher dimensional problems as well as higher order PDEs. Implementations are extremely simple, since the underlying explicit scheme or Jacobi method can be used as a black box solver. Last but not least, these cyclic schemes are perfectly suited for modern parallel architectures such as GPUs. We also provide a public domain code package that allows users to experiment with our cyclic methods.

1 Introduction

Solving image analysis problems with modern parabolic or elliptic partial differential equations (PDEs) can involve a number of numerical challenges, e.g.

- The diffusivity in diffusion filtering is a nonlinear function with a range over many orders of magnitude [1]. Thus, large stopping times are needed to achieve a desired degree of smoothing.

- Filters can involve very pronounced anisotropies [2]. This excludes some schemes that are efficient for isotropic problems (e.g. additive operator splittings (AOS) [3, 4]), and L^∞-stability may be violated.

- The data domain is not restricted to the 2-D scenario: 3-D filters are fairly common. Since they involve a huge amount of data, finding efficient algorithms is indispensable.

- Also higher order PDEs are used, in particular for inpainting problems. Explicit finite difference schemes require very small time step sizes, while implicit approaches are burdensome due to the higher stencil size of higher order PDE methods.

For each of these problem types, individual solutions have been developed. They may proceed in very different ways and involve different levels of implementation complexity (e.g. additive operator splittings [3, 4] and semi-implicit schemes [2]). Clearly, it would be desirable to have more generic tools that are simple to implement and broadly applicable. Moreover, the general availability of GPUs has shifted the focus of numerical methods from optimised sequential algorithms to easily implementable parallel methods.

Our Contribution. The goal of the present paper is to provide a framework for solving the above mentioned problems simultaneously. We introduce a class of numerical methods that are broadly applicable, easy to implement, and well-suited for parallel architectures. In the parabolic case, they are variants of the simplest numerical method: an explicit finite difference scheme. While the explicit scheme is usually applied with a fixed time step size that must satisfy a restrictive stability condition, we apply cycles of varying time step sizes where up to 50 % of the individual steps may violate this stability condition. Nevertheless, at the end of one cycle, one approximates a stable filter. We call these methods *Fast Explicit Diffusion (FED) schemes.* Due to the admissible violation of the step size limit, they can give speed-ups of several orders of magnitude compared to an explicit scheme with fixed time step size. Similar ideas can be applied in the elliptic setting where we modify the simple Jacobi over-relaxation (JOR) method such that the relaxation parameter is no longer fixed, but varied in a cyclic way, too. We will show that these so-called *Fast-Jacobi (FJ) methods* are much more efficient than their JOR predecessor. Both our FED and FJ schemes benefit from their intrinsic parallelism that makes them well-suited for modern parallel architectures such as GPUs. Moreover, they do not require specific implementation efforts: One can use an existent explicit scheme or JOR method as black box solver that is only modified by adapting its time step size or relaxation parameter in a cyclic way. These concepts have a much broader applicability than well-established numerical methods in the image analysis community such as (semi-)implicit methods and additive operator splittings: They are basically applicable in all parabolic or elliptic scenarios with symmetric system matrices.

Related Work. Our FED and FJ can be regarded as variants of the so-called *Super Time Stepping (STS)* schemes [5, 6, 7, 8] and *cyclic Richardson* methods [9, 10, 11]. While Super Time Stepping and cyclic Richardson methods are classical ideas in numerical analysis, they have not been applied in the image analysis community so far. Historically they were not highly popular in the numerical analysis community, since there were more efficient alternatives available when they have been discovered, or there were subtle problems to solve in order to guarantee numerical stability. With the wide availability of GPUs their intrinsic parallelism and their simplicity makes them methods of choice for modern image analysis problems. Moreover, we do not only introduce these methods to the image analysis community, but also derive them in a novel way via factorisations of box filters. This leads to parameter cycles that differ from those of classical cyclic Richardson or STS schemes. We will see that these new parameter cycles favour smoothing properties over rapid convergence. This makes them also attractive as basic solvers within a multigrid context. For example, the resulting *cascadic FED* allows to solve some elliptic problems with higher efficiency.
The present paper extends our conference article [12], in which we have introduced FED schemes, by the following substantial contributions:

- We present more theoretical insights explaining and illustrating the connection between linear filters and cyclic diffusion schemes.
- We introduce a novel efficient numerical method for the solution of elliptic problems: the Fast-Jacobi algorithm.
- We generalize the application of our novel numerical schemes to additional tasks, in particular to problems of higher order and higher dimensionality. Our applications also cover the implementation of such schemes on modern GPUs.

In this context, it should be mentioned that our conference paper [12] has lead to a constantly increasing number of applications using such schemes, e.g. for optic flow computation with a parallel GPU implementation [13], fast filtering methods on smartphones [14], medical image analysis [15, 16], variational depth-from-defocus [17], and a cyclic projected gradient method for convex optimisation [18].

Organisation of the Paper. Section 2 deals with the connection between linear symmetric filters and explicit diffusion schemes in one dimension. To this end, we present some important definitions, propositions and theorems. Moreover, we illustrate this connection by means of three examples for filters whose iterative application approximates Gaussian kernels. In Section 3 we present a more detailed description of our parabolic FED approach. Section 4 deals with the solution of elliptic problems and considers both the cascadic FED scheme and the so-called Fast-Jacobi solver. After this, we present five key applications in Section 5 and conclude the paper in Section 6. Proofs and additional mathematical details can be found in the Appendix.

2 Filter Factorisation

In this section, we derive and analyse the equivalence between symmetric 1-D filter kernels and explicit diffusion schemes with varying time step sizes. The derivation is based on a factorisation of the kernels.

2.1 Diffusion Interpretation of Smoothing Kernels

Let $\boldsymbol{f} = (f_i)_{i\in\mathbb{Z}}$ be a discrete 1-D signal given on a grid with mesh size $h > 0$. We define a discrete symmetric filter L^h_{2n+1} of finite length $(2n+1)h$, $n \in \mathbb{N}$, by

$$L^h_{2n+1}\, f_i \;:=\; \sum_{k=-n}^{n} w_k \cdot f_{i+k}\ , \tag{2.1}$$

where $w_k \in \mathbb{R}$ are the so-called weights of the filter. We should mention that the definition of the length comes from the support $\big[-(n+\frac{1}{2})h\,,\,(n+\frac{1}{2})h\big]$ of the continuous version $W(x)$ of the discrete kernel, which we get 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 $\big[(k-\frac{1}{2})h\,,\,(k+\frac{1}{2})h\big]$.

A special linear filter is the central finite difference approximation to the second order derivative:

$$\Delta_h f_i \;:=\; \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}\ . \tag{2.2}$$

It is important for the numerical solution of the linear heat equation

$$\partial_t u(x,t) \;=\; \partial_{xx} u(x,t)\ . \tag{2.3}$$

To see this, let us consider a grid point $x_j := \left(j - \frac{1}{2}\right) h$, a time step size $\tau > 0$ and a discrete point in time $t_k := k \cdot \tau$. Then an explicit discretisation of the linear heat equation in (x_j, t_k) is given by

$$u_j^{k+1} = (I + \tau \Delta_h) u_j^k , \tag{2.4}$$

where I is the identity operator and u_j^k approximates $u(x_j, t_k)$.
In the following, the operator Δ_h^m denotes the m times composition of Δ_h, i.e. it discretises the derivative of order $2m$.

The following main theorem establishes an interesting connection between L_{2n+1}^h and 1-D explicit diffusion schemes. We show that every discrete symmetric 1-D filter L_{2n+1}^h can be written as a weighted sum of discrete even-order derivative approximations:

$$L_{2n+1}^h = \sum_{m=0}^{n} \alpha_m^{(n)} \cdot \Delta_h^m . \tag{2.5}$$

Factorising this expansion gives the following result.

Theorem 1 (Diffusion Interpretation of Smoothing Kernels). *Let L_{2n+1}^h be an arbitrary discrete symmetric 1-D filter. Then the representation in Eq.* (2.5) *is unique. Its coefficients are given by*

$$\alpha_m^{(n)} = h^{2m} \cdot \sum_{k=m}^{n} \left(\binom{k+m}{2m} + \left(1 - \delta_{(k+m),0}\right) \binom{k+m-1}{2m} \right) w_k , \tag{2.6}$$

where $\delta_{i,j} := 1$ for $i = j$ and $\delta_{i,j} := 0$ else.
Moreover, if the weights w_k sum up to 1, *L_{2n+1}^h is equivalent to a cycle of n explicit 1-D linear diffusion steps, i.e.,*

$$L_{2n+1}^h = \prod_{i=0}^{n-1} (I + \tau_i \Delta_h) . \tag{2.7}$$

The time step sizes τ_i satisfy $\tau_i = z_i^{-1} \in \mathbb{C}$, where $z_i \in \mathbb{C} \setminus \{0\}$ are the roots of the polynomial

$$p_L(z) := \sum_{m=0}^{n} \alpha_m^{(n)} \cdot (-z)^m . \tag{2.8}$$

The (total) cycle time *$\theta_n := \sum_{i=0}^{n-1} \tau_i$ is given by*

$$\theta_n = h^2 \sum_{k=1}^{n} k^2 w_k . \tag{2.9}$$

For a detailed proof we refer to the Appendix A.1. Let us now illustrate this theorem by applying it to three different smoothing kernels.

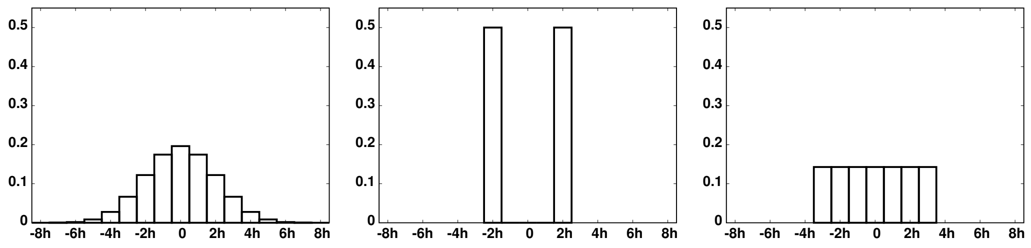

Figure 1: Illustration of three kernels that correspond to a diffusion time of $2h^2$. **(a) Left:** Binomial kernel with length $17h$. **(b) Middle:** MV kernel with length $5h$. **(c) Right:** Box kernel with length $7h$.

Table 1: Comparison of three filter kernels.

kernel	**binomial**	**MV**	**box**
kernel weights $\boldsymbol{w_k}$	$\frac{1}{4^n}\binom{2n}{n+k}$	$\frac{1}{2}\cdot\delta_{\lvert k\rvert,n}$	$\frac{1}{2n+1}$
time step sizes $\boldsymbol{\tau_i}$	$\frac{h^2}{4}$	$\frac{h^2}{2}\cdot\frac{1}{2\cos^2\left(\pi\frac{2i+1}{4n}\right)}$	$\frac{h^2}{2}\cdot\frac{1}{2\cos^2\left(\pi\frac{2i+1}{4n+2}\right)}$
cycle time $\boldsymbol{\theta_n}$	$\frac{h^2}{4}\cdot n$	$\frac{h^2}{2}\cdot n^2$	$\frac{h^2}{6}\cdot(n^2+n)$
cycle time order	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
approx. quality	very good	poor	good

2.2 Three Examples for Filter Factorisations

It is well known [19] that the analytic solution $u(x,t)$ of the 1-D linear heat equation with initial function $u(x,0)$ is given by the convolution of $u(x,0)$ with a Gaussian of standard deviation $\sigma = \sqrt{2t}$. In the discrete setting, the central limit theorem guarantees that a Gaussian convolution can be approximated by iterative applications of any nonnegative filter kernel whose weights sum up to 1.

Figure 1 illustrates three examples for such filter kernels: the binomial kernel, the so-called *maximum variance (MV) kernel*, and the box kernel [20]. By applying Theorem 1, we compute their corresponding time step sizes and their cycle times. The results are given in Table 1, and detailed derivations are described in the Appendix A.2, A.3, and A.4. Let us now compare the three cyclic schemes that correspond to factorisations of these filter kernels into explicit diffusion steps.

For our comparison, we approximate a 1-D Gaussian kernel with standard deviation $\sigma = \sqrt{12}h$. Convolution with such a Gaussian corresponds to a linear diffusion process with stopping time $T = \frac{1}{2}\sigma^2 = 6h^2$. We want to approximate this Gaussian by three iterations of our kernels. Thus, each kernel must correspond to a cycle time of $2h^2$.

Table 1 shows that a binomial filter factorisation leads to constant time step sizes $\tau_i = \frac{h^2}{4}$. Hence, we need 8 time steps to reach a cycle time of $2h^2$. In total, $3 \cdot 8 = 24$ steps are required for our Gaussian approximation. Due to the constant time step sizes, it is only possible to obtain a cycle time of order $\mathcal{O}(n)$ in n steps. This is rather inefficient. However, Figure 2(a) shows that a binomial kernel provides a very good approximation to a Gaussian.

The MV kernel corresponds to a cycle time of $\frac{h^2}{2} \cdot n^2$. Therefore, it requires a cycle with length $n = 2$ to yield a cycle time of $2h^2$. Consequently, $3 \cdot 2 = 6$ applications of the explicit scheme are sufficient to approximate our Gaussian with 3 MV iterations. This illustrates that the variable time steps derived from the MV filter give much more efficient schemes than the fixed time step scheme that results from the factorisation of the binomial filter. Indeed, Table 1 shows that a MV filter factorisation into n explicit diffusion steps allows a cycle time of order $\mathcal{O}(n^2)$. However, Figure 2(b) demonstrates that this high efficiency is achieved at the expense of a very poor approximation of the Gaussian.

In order to find a better compromise between efficiency and approximation quality, let us now have a look at a factorisation of the box filter. Since its cycle time is given by $\frac{h^2}{6} \cdot (n^2 + n)$, it follows that $n = 3$ diffusion steps are necessary to reach the time $2h^2$ in one cycle. Thus, for three cycles, $3 \cdot 3 = 9$ applications of the explicit scheme are needed. Although this is slightly less efficient than the MV filter factorisation, one can still obtain a cycle time of order $\mathcal{O}(n^2)$ within n steps. Moreover, Figure 2(c) illustrates that the approximation quality is almost as good as the binomial approximation. Thus, the box filter factorisation gives us the best of two worlds: high efficiency and good approximation quality.

3 Fast Explicit Diffusion (FED)

In the last section we have identified the box filter as a favourable kernel for factorisation into explicit diffusion steps. Let us now study the corresponding scheme with varying time step sizes in more detail. This will lead us to a versatile algorithm for all kinds of diffusion problems.

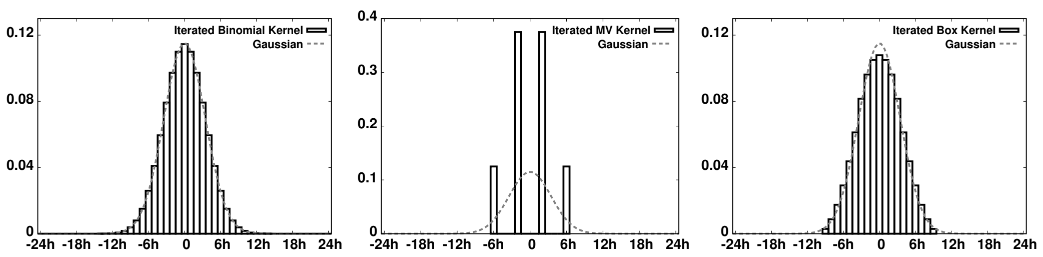

Figure 2: Comparison between the kernels in Fig. 1 after 3 iterations and their approximated Gaussian. **(a) Left:** An iterated binomial kernel approximates the Gaussian very well. **(b) Middle:** An iterated MV kernel yields a poor approximation of the Gaussian. Note that the scale in vertical direction differs from (a) and (c). **(c) Right:** An iterated box kernel achieves a good approximation quality.

3.1 Linear FED scheme

We reconsider the 1-D diffusion equation (2.3) and use the same space discretisation with grid size $h > 0$ and N grid points x_j, $j = 1, \ldots, N$. If we refrain from a time discretisation, the PDE becomes a time-continuous system of ordinary differential equations (ODEs):

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{A}\boldsymbol{u}\,, \tag{3.1}$$

where $\boldsymbol{u} = \boldsymbol{u}(t) \in \mathbb{R}^N$ is the vector with the entries $u_j(t) \approx u(x_j, t)$, and the matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ approximates the second order spatial derivative:

$$\boldsymbol{A} = \frac{1}{h^2} \cdot \begin{pmatrix} -1 & 1 & & & & \boldsymbol{0} \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ \boldsymbol{0} & & & & 1 & -1 \end{pmatrix}. \tag{3.2}$$

If the ODE system (3.1) is discretised in time using forward differences with time step size $\tau > 0$, and the right hand side is evaluated at the old time level, we obtain an explicit numerical scheme:

$$\boldsymbol{u}^{k+1} = (\boldsymbol{I} + \tau\,\boldsymbol{A})\,\boldsymbol{u}^k \qquad (k \geq 0). \tag{3.3}$$

Here we use the approximations $\boldsymbol{u}^k \in \mathbb{R}^N$ with the entries $u_j^k \approx u(x_j, t_k)$. The matrix $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ denotes the unit matrix.

To define a *cyclic* explicit scheme we assume $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$ and replace the constant time step size in Eq. (3.3) by a cycle of varying time step sizes τ_i:

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; (\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{A})\;\boldsymbol{u}^{k+1,\,i} \qquad (i=0,\dots,n-1). \tag{3.4}$$

After the complete cycle we get $\boldsymbol{u}^{k+1} := \boldsymbol{u}^{k+1,\,n}$. For our *Fast Explicit Diffusion (FED)* scheme, we use the varying time step sizes that originate from the factorisation of the box filter:

$$\tau_i \;=\; \frac{h^2}{2}\cdot\frac{1}{2\cos^2\left(\pi\cdot\frac{2i+1}{4n+2}\right)} \qquad (i=0,...,n-1). \tag{3.5}$$

Note that vector $\boldsymbol{u}^k = \left(u_j^k\right)_{j=1}^N$ approximates the values $u(x_j, k\cdot\theta_n)$, where

$$\theta_n = \frac{h^2}{6}\cdot\left(n^2+n\right) \tag{3.6}$$

is the time of one cycle with n time steps; cf. Table 1. Thus, we may interpret a full FED cycle with time θ_n as a single *super time step* in the sense of [7]. This interpretation will also be useful later on when we consider nonlinear problems.

Our FED scheme has a very interesting property: Some of the time step sizes τ_i violate the stability condition $\tau \le \frac{h^2}{2}$ for the explicit scheme (3.3) with constant time step size. This is caused by the factor $\frac{1}{2\cos^2\left(\pi\cdot\frac{2i+1}{4n+2}\right)}$ in Eq. (3.5), which can be significantly larger than 1. It is easy to see that up to 50 percent of the time step sizes can violate the stability constraint. Table 2 illustrates this: It shows both the smallest and largest three time step sizes for different n. Note that for $n = 1000$, the largest time step size is more than 200000 times larger than the stability limit. The total time after one cycle with 1000 iterations is more than 333 times larger than for an explicit scheme with constant step size given by the stability limit. This demonstrates the substantial speed-up that can be achieved with unstable time step sizes. However, at the end of a full cycle we obtain a stable scheme, since it is equivalent to a box filter.

So far, the times θ_n of the FED cycles cover only a discrete set of values. To allow arbitrary cycle times T, we simply compute the minimum cycle length n with $\theta_n \ge T$ and multiply the time step sizes by the factor $q := T/\theta_n \le 1$.

3.2 Extension to Arbitrary Diffusion Problems

Our FED scheme has been motivated in the 1-D setting with explicit linear diffusion filtering. This was for didactic reasons only: Since box filtering is

Table 2: The first three and last three step sizes of 1-D FED 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

already highly efficient, there is no practical use to factorise it into explicit linear diffusion steps. However, we have learned how to speed up an explicit scheme by replacing iterations with a constant time step size by cycles with varying time step sizes. Let us now show that this principle is very general and leads to highly efficient schemes in more challenging situations: We will see that it can be applied to one- and multidimensional diffusion-like processes, regardless if they are linear or nonlinear, isotropic or anisotropic, second order or higher order.

According to Gershgorin's theorem [21], the eigenvalues of the matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ from Eq. (3.2) lie in the interval $\left[-\frac{4}{h^2}\,,\,0\right]$. These eigenvalues determine the stability in the Euclidean norm: A stable explicit step requires a time step size τ such that all eigenvalues of the iteration matrix $\boldsymbol{I} + \tau\boldsymbol{A}$ are in the interval $[-1, 1]$. This is guaranteed if the time step size τ does not exceed a limit $\tau_{\max}$ that satisfies

$$\tau_{\max} \;=\; \frac{2}{\mu_{\max}(\boldsymbol{A})} \;=\; \frac{h^2}{2}\,, \tag{3.7}$$

where $\mu_{\max}(\boldsymbol{A})$ corresponds to the largest modulus of the eigenvalues of $\boldsymbol{A}$.

Keeping this in mind, it is straightforward to replace the matrix $\boldsymbol{A}$ by any symmetric, negative semidefinite matrix $\boldsymbol{P}$ that results from a space discretisation of a suitable parabolic PDE. Such symmetric, negative semidefinite matrices can originate from PDEs that are one- or multidimensional, of second or higher order, linear or nonlinear, isotropic or anisotropic. All one has to change in our setting is to adapt the time step sizes to the largest modulus of the eigenvalues $\mu_{\max}(\boldsymbol{P})$. Thus, we consider the generalisation of the FED time step sizes in Eq. (3.5), where we replace the 1-D explicit diffusion limit

$\tau_{\max} = \frac{h^2}{2}$ by the general limit $\tau_{\max} = \frac{2}{\mu_{\max}(\boldsymbol{P})}$ of the explicit scheme

$$\boldsymbol{u}^{k+1} \;=\; (\boldsymbol{I} \,+\, \tau\,\boldsymbol{P})\,\boldsymbol{u}^{k} \qquad (k \geq 0). \tag{3.8}$$

with constant time step size τ. Therefore, a general FED scheme uses the time step sizes

$$\tau_i \;=\; \frac{2}{\mu_{\max}(\boldsymbol{P})} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, \ldots, n{-}1). \tag{3.9}$$

One cycle with n such time steps can be seen as a super time step of size

$$\theta_n \;=\; \frac{2}{\mu_{\max}(\boldsymbol{P})} \cdot \frac{n^2+n}{3}\;, \tag{3.10}$$

which simply replaces $\frac{h^2}{2}$ in Eq. (3.6) by $\frac{2}{\mu_{\max}(\boldsymbol{P})}$. More details about the stability of the general FED scheme are given in the Appendix A.5.

Let us just explain how nonlinear problems can be handled that lead to a matrix $\boldsymbol{P}(\boldsymbol{u})$ which depends on the evolving data $\boldsymbol{u}(t)$. In this case, we can either use a worst case *a priori* estimate for the values $\mu_{\max}(\boldsymbol{P}(\cdot))$, or one can update it *a posteriori* after each cycle. For the experiments in Section 5, we use an a priori estimate. With $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^{k}$ our FED cycle for nonlinear problems is given by

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; \big(\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{P}(\boldsymbol{u}^{k})\big)\;\boldsymbol{u}^{k+1,\,i} \qquad (i = 0, ..., n{-}1). \tag{3.11}$$

Note that we keep the nonlinearities $\boldsymbol{P}(\boldsymbol{u}^k)$ constant during the whole cycle, i.e. we perform one super time step with $\boldsymbol{P}(\boldsymbol{u}^k)$ for the computation of $\boldsymbol{u}^{k+1} := \boldsymbol{u}^{k+1,\,n}$. As we will see later, this strategy makes also sense with respect to the numerical stability.

3.3 Connection to Super Time Stepping

Our FED scheme uses different time step sizes, where some of them may violate the stability limit. A similar method has been proposed by Yuan'Chzhao-Din and Saul'yev [5, 6]. Later, Gentzsch et al. [7, 8] as well as Alexiades et al. [22, 23] have used the same idea under the name *Super Time Stepping (STS)*. Contrary to our derivation, they have used a direct approach that is not based on a filter factorisation: They sought a set of different time step sizes that keeps stability after each cycle, and at the same time maximises the cycle time. In our filter factorisation framework, their method would factorise the MV kernel $\left(\frac{1}{2}, 0, \ldots, 0, \frac{1}{2}\right)$. Since this kernel is very sensitive with respect to high frequencies, they had to introduce an additional damping parameter $\nu > 0$ that ensures better attenuation properties for high frequencies. This regularisation can be seen as a trade-off between efficiency and damping quality, since larger values for ν scale down the cycle time. Hence, different damping parameters yield different results. In our FED framework, such an additional damping parameter is not necessary.

3.4 Numerical Stability

In the context of STS it is well-known that although the ordering of the explicit diffusion steps does not matter in exact arithmetic, it can influence the result in practice due to numerical rounding errors when n is large. Similar problems can also be observed for FED. To address this issue, we advocate two strategies that rearrange the sequence of the FED time step sizes within the cycles: κ-cycles and Leja ordering.

3.4.1 κ-Cycles

Gentzsch et al. [7] have proposed to rearrange the original sequence of the explicit time steps $\tau_0, \ldots, \tau_{n-1}$ within so-called κ-cycles.

To illustrate the principle by an example, let us assume that we have a cycle of length $n = 11$. Then the indices from 0 to 5 correspond to stable steps, while the indices from 6 to 10 represent unstable steps. To avoid an error accumulation towards the end of the cycle, we rearrange the indices in smaller subgroups that contain stable and unstable steps. For instance, we can choose the rearrangement $\boxed{0, 3, 6, 9}$, $\boxed{1, 4, 7, 10}$, $\boxed{2, 5, 8}$. Note that the indices within the groups differ by multiples of 3. Such a rearrangement represents a κ-cycle with $\kappa = 3$.

In general, a κ-cycle can be formulated as follows: Let p be the smallest prime number with $p \geq n$. For $m = 0, \ldots, p-1$, we compute the values

$$\Phi(m) \;=\; \big(m \cdot \kappa\big) \mod p\;, \tag{3.12}$$

where $\kappa \in \{2, \ldots, n-1\}$ steers the rearrangement. This yields a new sequence $\Phi(0)$, $\Phi(1), \ldots, \Phi(p-1)$. Since we want to cover only values from 0 to $n-1$, we drop all indices $\Phi(m)$ larger than $n-1$. So, we get a feasible rearrangement of the original sequence.

Unfortunately, there is no panacea for the choice of κ. However, it is possible to create a look-up table (using test problems) with suitable values $\kappa = \kappa(n)$ that ensure better robustness against numerical rounding errors. We have used κ-cycles in our conference paper [12].

3.4.2 Leja Ordering

Since suitable κ-cycles can only be found experimentally and therefore might depend on the setting of the test problems, we discuss an approach that is independent of such test settings: It is based on the so-called *Leja ordering* [24, 25] that has already been successfully applied to iterative solvers [26].

We sketch only the practical application of *Leja ordering* and refer e.g. to [26] for a theoretical justification. We consider a set S consisting of $\ell + 1$ real numbers $\{x_0, \ldots, x_\ell\}$. This set is *Leja ordered*, if the numbers are arranged such that

$$\prod_{k=0}^{j} |x_{j+1} - x_k| \;=\; \max_{x \in S} \prod_{k=0}^{j} |x - x_k| \;\; , \quad j = 0, \ldots, \ell - 1 \; , \tag{3.13}$$

where $|x_0| = \max\limits_{x \in S} |x|$. In some cases it might happen that at least two numbers fulfil the maximum condition. Then we just take the smallest value to have a unique rearrangement.

Applying the Leja algorithm on the set of the roots $\{z_i \,|\, i = 0, ..., n-1\}$ of the polynomial $p_L(z)$ from Theorem 1 yields a new sequence of the time step sizes $\tau_i = z_i^{-1}$. The main advantage compared to κ-cycles is that the arrangement given by the Leja algorithm only depends on the input points z_i. The Leja ordering can be computed conveniently in advance to create a look-up table with the Leja ordered sequences.

In the case of our previous example with cycle length $n = 11$, the Leja ordering yields the index sequence $0, 10, 5, 7, 3, 9, 2, 6, 1, 8, 4$. We observe that this rearrangement differs from the κ-cycle arrangement.

We use the Leja ordering within our experiments, since it gives an even higher numerical robustness than κ-cycles and thus allows larger cycle lengths. In realistic applications with $n \leq 1000$, however, both strategies are absolutely unproblematic.

3.4.3 Other Orderings

Besides the two presented rearrangements, there are further strategies such as the one by Lebedev and Finogenov [27]. It is based on a simple recursion relation. Unfortunately this recursion only works for cycle lengths $n = 2^k$, $k \in \mathbb{N}$. In the worst case, this strategy can double the cycle length and therefore the effort. However, for cycle lengths $n = 2^k$, it can be an elegant and powerful alternative to κ-cycles and Leja ordering.

1. **Input Data**:
   image $\boldsymbol{f}$, stopping time T, number M of FED cycles
2. **Initialisation**:
   - (a) Compute the smallest n such that the time θ_n of one FED cycle fulfils $\theta_n \geq T/M$, and define $q := T/(M \cdot \theta_n) \leq 1$.
   - (b) Compute the time step sizes $\tilde{\tau}_i := q \cdot \tau_i$ with τ_i according to (3.9).
   - (c) Choose a suitable ordering for the step sizes $\tilde{\tau}_i$ (e.g. κ-cycles or Leja ordering).
   - (d) If the problem is linear, compute the corresponding matrix $\boldsymbol{P}$.
3. **Filtering Loop**: $(k = 0, \ldots, M-1)$
   - (a) If the problem is nonlinear, compute the corresponding matrix $\boldsymbol{P}(\boldsymbol{u}^k)$.
   - (b) Perform one FED cycle with the above ordering of the n explicit time steps $\tilde{\tau}_i$, and increase k by 1.
   - (c) Go back to (a), if the stopping time T is not yet reached $(k < M-1)$.

Figure 3: General FED algorithm for diffusion filtering.

3.5 General FED Algorithm

At this point, we can give a summary of the general FED algorithm. It is shown in Fig. 3. Note that FED is essentially an explicit scheme with some overhead that is not time critical. Besides the rearrangement of the sequence, it is very important to update the nonlinearities only after one complete cycle. Updates within a cycle can be dangerous, because the stability of the intermediate results – and therefore a correct evaluation of the nonlinearities – can not be guaranteed for rearranged cycles.

Note that in the linear 1-D diffusion setting, one FED cycle represents a box filter. Since several iterations of a box filter are required to give a good approximation of Gaussian convolution (and thus of the correct linear diffusion result), one should also use more than one cycle to improve the accuracy of the FED scheme in other scenarios. For linear problems already $M = 3$ cycles can be sufficient, while nonlinear problems can require more cycles due to their nonlinear updates. Examples are given in Section 5.

4 FED-based Methods for Elliptic Problems

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. We shall discuss both options now.

4.1 Cascadic FED (CFED)

The first choice 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 [28], 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 [29, 30]. 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. To simplify matters, we always use the same parameter settings for the diffusion process on each level. We call this the *Cascadic Fast Explicit Diffusion (CFED)* approach.

4.2 Fast-Jacobi Solver

For the direct solution of elliptic equations, we now propose the so-called *Fast-Jacobi (FJ)* method. It combines the simple Jacobi over-relaxation (JOR) method [21] together with varying relaxation parameters that are based on the FED time step sizes. More precisely, we consider a linear system with N equations:

$$\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{c}\,, \tag{4.1}$$

where $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ is a symmetric, positive definite system matrix, $\boldsymbol{c} \in \mathbb{R}^N$ the given right hand side, and $\boldsymbol{x} \in \mathbb{R}^N$ the unknown solution.

We can solve this linear system by means of JOR iterations. With $\boldsymbol{D} :=$

diag($\boldsymbol{B}$), one JOR step with relaxation parameter $\omega_i > 0$ is given by

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \omega_i \boldsymbol{D}^{-1}\big(\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}^i\big) \tag{4.2}$$

where the upper index denotes the iteration level. The original JOR method uses a constant relaxation parameter, which means $\omega_i = \omega$ for all $i \geq 0$. If $\omega = 1$, it corresponds to the standard Jacobi method.

In contrast to these classical approaches, Fast-Jacobi with cycle length n uses the varying parameters

$$\omega_i = \omega_{\max} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, ..., n-1). \tag{4.3}$$

with

$$\omega_{\max} := \frac{2}{\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{B})}. \tag{4.4}$$

The value $\omega_{\max}$ plays a similar role as the time step size limit $\tau_{\max}$ in the case of the parabolic FED, and Eq. (4.3) is the elliptic analogue to (3.9).

To understand the stability of the Fast-Jacobi method and its relation to FED, it is instructive to investigate the error $\boldsymbol{e}^i := \boldsymbol{x}^i - \boldsymbol{x}$. It is easy to see that it satisfies

$$\boldsymbol{e}^{i+1} = \Big(\boldsymbol{I} - \omega_i \boldsymbol{D}^{-1}\boldsymbol{B}\Big)\boldsymbol{e}^i . \tag{4.5}$$

This is an FED scheme for the matrix $\boldsymbol{D}^{-1}\boldsymbol{B}$. The multiplication of $\boldsymbol{B}$ with $\boldsymbol{D}^{-1}$ can be interpreted as local adjustments of the relaxation parameters (or time step sizes). Such an adjustment can be very helpful as a preconditioner for matrices $\boldsymbol{B}$ whose diagonal entries strongly vary in their orders of magnitude. In terms of diffusion, this corresponds to a diffusivity function with a range over many orders of magnitude. Thus, one can expect that an elliptic problem with strongly varying diagonal entries can be solved more efficiently with Fast-Jacobi than with the FED approach.
With $\omega_{\max} = \frac{2}{\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{B})}$ one obtains stable cyclic schemes, since the eigenvalues of $\boldsymbol{I} - \omega_{\max}\boldsymbol{D}^{-1}\boldsymbol{B}$ are in the interval $[-1, 1]$. This avoids error explosion in (4.5).

A summary of the whole algorithm is depicted in Fig. 4. Besides the use of multiple cycles, one can further improve the convergence by embedding the Fast-Jacobi method into a coarse-to-fine approach similarly to CFED.

4.3 Connection to the Cyclic Richardson Method

The idea to use varying parameters for simple iterative algorithms has a long tradition. Already in 1910, Richardson [9] has introduced the following cyclic

```
1. **Input Data**:
   linear system $\boldsymbol{Bx} = \boldsymbol{c}$, cycle length $n$, number of cycles $M$, feasible $\omega_{\max} > 0$
2. **Initialisation**:
   (a) Compute the relaxation parameters $\omega_i$ according to Eq. (4.3).
   (b) Choose a suitable ordering for the 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 cycle ($i = 0,\ldots,n-1$) with the above ordering:
           $\boldsymbol{x}^{k,\,i+1} = \boldsymbol{x}^{k,\,i} + \tilde{\omega}_i\, \boldsymbol{D}^{-1}\big(\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}^{k,\,i}\big)$ .
   (c) $\boldsymbol{x}^k := \boldsymbol{x}^{k,\,n}$.
```

Figure 4: Fast-Jacobi method.

method:

$$\boldsymbol{x}^{i+1} \;=\; \boldsymbol{x}^{i} \;+\; \omega_i\,\big(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^{i}\big)\;. \tag{4.6}$$

Compared to Fast-Jacobi, there is no multiplication with a diagonal matrix. For a symmetric, positive definite matrix $\boldsymbol{B}$ with smallest eigenvalue $\lambda_{\min}$ and largest eigenvalue $\lambda_{\max}$, Young [10] has proposed the relaxation parameters

$$\omega_i \;=\; \frac{2}{\lambda_{\max}+\lambda_{\min} \;-\; (\lambda_{\max}-\lambda_{\min})\cdot\cos\left(\pi\cdot\frac{2(n-i)-1}{2n}\right)} \qquad (i=0,\ldots,n-1). \tag{4.7}$$

The method remains stable, if one replaces $\lambda_{\min}$ by 0. In this case, the factor $\frac{2}{\lambda_{\max}}$ can be interpreted as the maximum relaxation parameter $\omega_{\max}$ for the cyclic Richardson method (4.6). With the help of $2\cos^2 x = 1-\cos(\pi-2x)$ the resulting relaxation parameter cycles can be simplified to

$$\omega_i \;=\; \frac{2}{\lambda_{\max}}\cdot\frac{1}{2\,\cos^2\left(\pi\cdot\frac{2i+1}{4n}\right)} \qquad (i=0,\ldots,n-1). \tag{4.8}$$

This can be seen as the elliptic variant of the time step sizes that originate from a factorised MV kernel (cf. Table 1). Thus, the cyclic Richardson

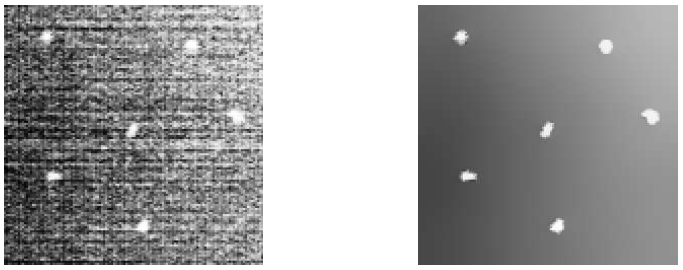
Figure 5: Test setting for nonlinear isotropic diffusion filtering. **(a) Left:** Mammogram (128×128 pixels). **(b) Right:** Filtered with the explicit scheme using $\lambda = 7.5$, $\sigma = 1$, $\tau = 10^{-2}$, and $T = 128$.

method is the elliptic analogue to the Super Time Stepping approach from Section 3.3. We can expect that it also suffers from an insufficient attenuation of high frequencies.

Since there can be very large relaxation parameters, both cyclic Richardson and Fast-Jacobi also require a rearrangement of the parameter sequence to improve the robustness against numerical rounding errors [11]. To this end, one can apply the strategies in Section 3.4 again.

5 Applications

Now we show that our proposed methods are well-suited to efficiently solve different parabolic or elliptic problems. We assume a uniform 2-D grid with the mesh sizes $h_x = h_y = 1$. All methods have been implemented in C and are executed on a standard desktop PC with a 3.2 GHz Intel Xeon processor. Our error measure is the relative mean absolute error (RMAE), $\sum_i \frac{|u_i - r_i|}{\|r\|_1}$ with $\|r\|_1 := \sum_i |r_i|$. The numerical result is denoted by u, and r is the corresponding reference solution.

5.1 FED for Isotropic Parabolic Problems

In our first experiment, we evaluate FED as a solver for isotropic parabolic problems. As a prototypical application we consider the nonlinear diffusion filter of Catté et al. [31]. It follows the evolution equation

$$\partial_t u \;=\; \operatorname{div}\left(g\left(|\boldsymbol{\nabla} u_\sigma|^2\right) \boldsymbol{\nabla} u\right) \;, \tag{5.1}$$

where u_σ denotes the function u convolved with a Gaussian of standard deviation $\sigma > 0$. The scalar-valued diffusivity function g is given by the

Table 3: Comparison of FED and AOS for nonlinear isotropic diffusion filtering with stopping time $T = 128$.

(super) time	**RMAE**	
step size	**AOS**	**FED**
32	0.0401	0.0069
16	0.0171	0.0034
8	0.0075	0.0021
4	0.0038	0.0013
2	0.0020	0.0006
1	0.0011	0.0003

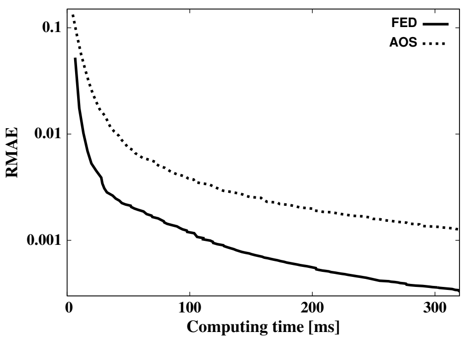

Figure 6: Computing time (millisec.) vs. RMAE (log-scaled) for the AOS and FED scheme.

diffusivity [32]

$$g(s^2) = \begin{cases} 1 & (s^2 = 0) \\ 1 - \exp\left(-\frac{3.315}{(s^2/\lambda^2)^4}\right) & (s^2 > 0). \end{cases} \tag{5.2}$$

For problems of this type, AOS schemes [3, 4] are regarded as efficient solvers. Hence, we compare our FED scheme to the AOS approach. It is easy to check that the explicit scheme on which FED is based has to satisfy the constant time step size limit $\tau_{\max} = 0.25$. Fig. 5 shows our test setting from [32] where we denoise a mammogram to improve the visibility of micro calcifications. For the stopping time $T = 128$ we compute a reference solution using the usual explicit scheme with a very small time step size $\tau = 10^{-2}$.

Table 3 compares the accuracy of FED and AOS for different time step sizes. In this context, we regard a full FED cycle as a super time step. We observe that both schemes are of first order in time: Reducing the time step size by a factor 2 decreases the error by a factor 2. However, for the same time step size, the FED error is about 4 times smaller than the AOS error, since FED does not suffer from splitting artifacts.

In practice one is of course interested in optimising the error w.r.t. the computing time. This relation is analysed in Fig. 6. We see that the FED scheme requires less computational effort than AOS to reach the same error. Thus, it is more efficient. If we consider for instance an error of 10^{-3}, FED is almost four times faster than AOS.

In conclusion, our experiment shows that for isotropic parabolic problems, FED is more accurate and more efficient than AOS.

5.2 FED for Anisotropic Parabolic Problems

A numerically more challenging scenario is given by anisotropic parabolic problems. This shall be illustrated by means of a coherence-enhancing anisotropic diffusion filter [33]. It is based on the PDE

$$\partial_t u \;=\; \operatorname{div}\left(\boldsymbol{D}\left(\boldsymbol{J}_\rho\left(\boldsymbol{\nabla} u_\sigma\right)\right)\boldsymbol{\nabla} u\right)\;, \tag{5.3}$$

with a symmetric, positive definite diffusion tensor $\boldsymbol{D} \in \mathbb{R}^{2\times 2}$. This diffusion tensor is a function of the structure tensor [34]

$$\boldsymbol{J}_\rho\left(\boldsymbol{\nabla} u_\sigma\right) \;:=\; G_\rho * \left(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top\right)\;, \tag{5.4}$$

where G_ρ is a 2-D Gaussian with standard deviation ρ. For more details we refer to [33].

The anisotropy of the diffusion tensor creates mixed derivative terms that can not be handled with typical AOS schemes anymore. Semi-implicit schemes that require the solution of linear systems are good alternatives. They are more efficient than explicit schemes with a constant time step size that is typically limited by $\tau_{\max} = 0.25$. Thus, we want to compare FED with such semi-implicit schemes. As space discretisation, we use the one from [35], since it hardly suffers from numerical diffusion artifacts. In the semi-implicit case, we solve the occurring linear systems of equations either with a successive over-relaxation (SOR) or a conjugate gradient (CG) algorithm [36]. Both solvers are stopped, if the residual of the current iteration $\boldsymbol{x}^k$ satisfies

$$\left\|\boldsymbol{B}\boldsymbol{x}^k - \boldsymbol{c}\right\|_2 \;<\; 10^{-3}\cdot\|\boldsymbol{c}\|_2\;, \tag{5.5}$$

where $\boldsymbol{B}$ denotes the system matrix and $\boldsymbol{c}$ the right hand side. The error tolerance $\varepsilon = 10^{-3}$ provides a good trade-off between the accurate solution of the linear system and the efficiency. While the CG method implicitly computes the residual for each iteration, the SOR solver needs an explicit evaluation that is done every 10 iterations. For SOR we optimise the relaxation parameter manually in order to obtain fast convergence ($\omega = 1.2$).

As a test scenario, we enhance a fingerprint image with this anisotropic diffusion process. First we compute a reference solution by applying a semi-implicit scheme with the small time step size $\tau = 10^{-2}$. The original image and the filtered result with stopping time $T = 256$ can be seen in Fig. 7.

Table 4 shows that FED and the semi-implicit method are both first order in time and yield comparable errors. With respect to computational efficiency illustrated in Fig. 8, however, FED outperforms the semi-implicit schemes,

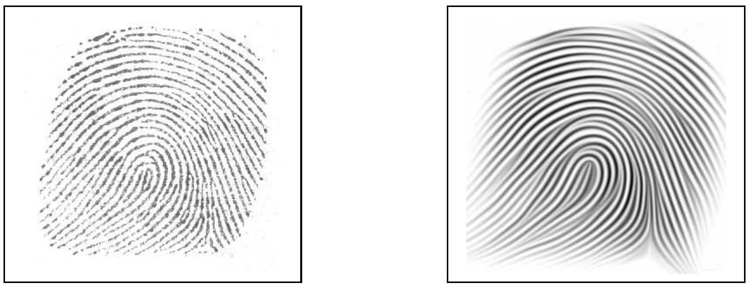

Figure 7: Test setting for nonlinear anisotropic coherence-enhancing diffusion filtering. **(a) Left:** Original *fingerprint* image (300×300 pixels). **(b) Right:** Filtered reference ($T = 256$, $\sigma = 0.5$, $\rho = 4$, $\tau = 10^{-2}$), rescaled to $[0, 255]$.

Table 4: Comparison of FED and the semi-implicit method for anisotropic diffusion.

(super) time	**RMAE**	
step size	**semi-impl.**	**FED**
64	0.0102	0.0112
32	0.0069	0.0075
16	0.0045	0.0049
8	0.0028	0.0028
4	0.0016	0.0015
2	0.0009	0.0008
1	0.0005	0.0004

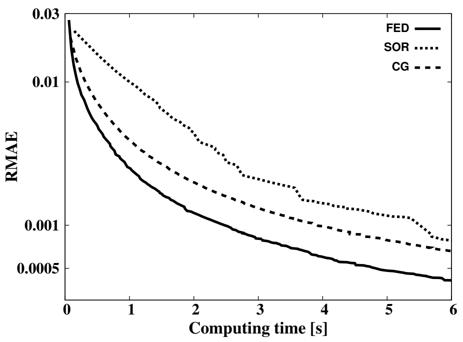

Figure 8: Computing time (sec.) vs. RMAE (log-scaled). The semi-implicit scheme uses SOR or CG solvers.

regardless whether they use SOR or CG as linear system solvers. Last but not least, it should be noted that FED is much simpler to implement than semi-implicit approaches and does not require to optimise additional parameters such as the error tolerance and the relaxation parameter.

5.3 CFED for Elliptic Problems with Constant Coefficients

So far, we have only performed experiments with parabolic PDEs. They describe evolution processes. Let us now analyse an elliptic problem that can be regarded as a steady state of a parabolic evolution.

As a prototype we consider an inpainting application that is inspired from PDE-based image compression (see e.g. [37]). It keeps a number of selected

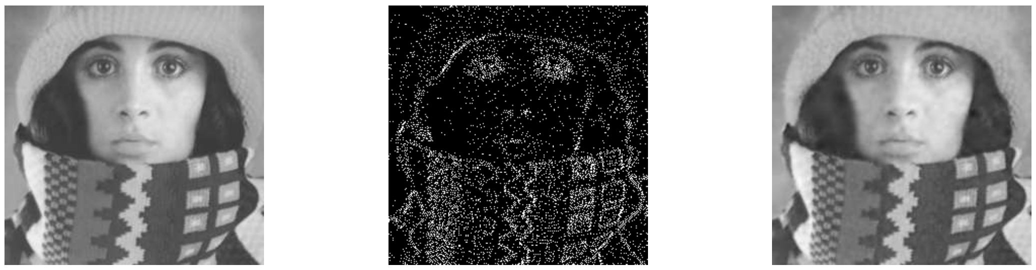
Figure 9: Biharmonic inpainting. **(a) Left:** Original *trui* image (256×256). **(b) Middle:** Inpainting mask. **(c) Right:** Reconstruction with biharmonic inpainting in the unspecified regions (stopping time $T = 10^6$).

pixels (given by the so-called inpainting mask), and interpolates the missing data by inpainting with the biharmonic equation

$$\Delta^2 u \;=\; 0. \tag{5.6}$$

Note that here we are dealing with a linear fourth order PDE with constant coefficients. To solve it numerically, we evolve its parabolic counterpart

$$\partial_t u \;=\; -\Delta^2 u \tag{5.7}$$

for $t \to \infty$ with two cyclic schemes for parabolic problems: FED and Super Time Stepping.

In order to apply cyclic schemes, we need an estimate of the stability limit $\tau_{\max} = \frac{2}{\mu_{\max}(\boldsymbol{P})}$ of the explicit scheme, where $\boldsymbol{P}$ is a discretisation of the biharmonic operator. By Gershgorin's theorem [21], one easily sees that for a discrete 4-point approximation $\boldsymbol{A}$ of the 2-D Laplacian, one has $\mu_{\max}(\boldsymbol{A}) = 8$. Thus, $\mu_{\max}(\boldsymbol{P}) = 8^2 = 64$, and we obtain $\tau_{\max} = \frac{1}{32}$. This very small step size limit makes an explicit scheme with constant time step size very inefficient. Thus, it is highly desirable to use cyclic schemes that allow steps beyond this restrictive limit. Moreover, since we are interested in the steady state, we use a cascadic embedding to speed up the evolution.

Our test setting is depicted in Fig. 9. We have computed a reference reconstruction for the stopping time $T = 10^6$ with the help of an explicit scheme ($\tau = 0.025$) on the original level, without any coarse-to-fine strategies.

For our experiments we compare CFED and a cascadic Super Time Stepping approach, where the coarse-to-fine strategy uses three levels: 256×256, 128×128, and 64×64 pixels. In Section 3.5 we have mentioned that already three cycles can provide good results for linear problems. Table 5 shows, in

Table 5: Comparison of CFED and cascadic Super Time Stepping (CSTS) with three cycles per level for biharmonic inpainting.

stopping time	**RMAE**	
per level	**CSTS**	**CFED**
50	0.06304	0.00225
100	0.06197	0.00134
200	0.06687	0.00068
400	0.06348	0.00032
800	0.07420	0.00015
1600	0.07725	0.00006

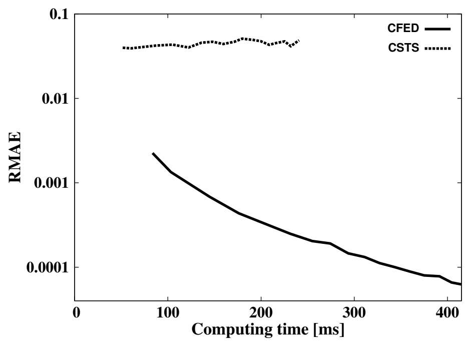

Figure 10: Computing time (millisec.) vs. RMAE (log-scaled) for CFED and CSTS.

fact, that CFED with three cycles per level yields very small errors which decrease when the stopping time grows. However, the errors of cascadic Super Time Stepping are up to about 1300 times larger. Interestingly, increasing the stopping time can not improve these errors, since cascadic Super Time Stepping suffers from bad attenuation properties of high frequencies. Therefore, CFED turns out to be much more efficient. This is shown in Fig. 10. Here, we have used stopping times from 50 to 1600 in order to illustrate the dependency between the errors and the computing times.

Overall, this example shows that CFED is well-suited for elliptic problems with constant coefficients, and that already three cycles can be sufficient for such linear problems.

5.4 Fast-Jacobi for Elliptic Problems with Strongly Varying Coefficients

The previous elliptic problem had constant coefficients, and it turned out that CFED is an appropriate solver for this purpose. In our next experiment, we consider an elliptic problem with strongly varying coefficients and show that in this case Fast-Jacobi is more efficient than FED.

Continuous Model Problem. Our prototypical scenario is given by an isotropic nonlinear image regularisation method. It computes a denoised version $u(\boldsymbol{x})$ of the image $f(\boldsymbol{x})$ by minimising an energy functional with a quadratic data term and with the subquadratic regulariser of Charbonnier et al. [38]:

$$E(u) \;=\; \int_\Omega \Big((u-f)^2 \,+\, \alpha \cdot 2\lambda^2 \sqrt{1 \,+\, |\boldsymbol{\nabla} u|^2/\lambda^2} \Big)\, d\boldsymbol{x} \;, \tag{5.8}$$

where Ω denotes the image domain, $\alpha > 0$ the regularisation weight, and $\lambda > 0$ is a contrast parameter. The corresponding Euler-Lagrange equation is given by

$$u - f \,-\, \alpha \,\mathrm{div}\, \big(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u \big) \;=\; 0 \tag{5.9}$$

with the diffusivity function

$$g(s^2) \;:=\; \frac{1}{\sqrt{1 \,+\, s^2/\lambda^2}} \;. \tag{5.10}$$

It is also possible to obtain a solution of the Euler-Lagrange equation (5.9) as the steady state solution of the parabolic gradient descent equation

$$\partial_t u \;=\; \mathrm{div}\, \big(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u \big) \;+\; \frac{f-u}{\alpha} \,. \tag{5.11}$$

FED Scheme. An explicit discretisation of Eq. (5.11) with time step size $\tau > 0$ and implicitly stabilised fidelity term yields

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} \;=\; \boldsymbol{A}(\boldsymbol{u}^k)\, \boldsymbol{u}^k \;+\; \frac{\boldsymbol{f} - \boldsymbol{u}^{k+1}}{\alpha} \;. \tag{5.12}$$

It can be rewritten as

$$\boldsymbol{u}^{k+1} \;=\; \frac{\alpha \left(I + \tau\, \boldsymbol{A}(\boldsymbol{u}^k) \right) \boldsymbol{u}^k \;+\; \tau \boldsymbol{f}}{\alpha + \tau} \;. \tag{5.13}$$

Note that this equation uses the expression $\boldsymbol{v}^{k+1} := (I + \tau\, \boldsymbol{A}(\boldsymbol{u}^k))\, \boldsymbol{u}^k$. It can be seen as an explicit scheme for a diffusion equation without data fidelity term. Since (5.13) 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 $\tau_{\max} = 0.25$. With $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$ an FED version of the explicit scheme (5.13) is given by

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; \frac{\alpha \left(I + \tau_i\, \boldsymbol{A}(\boldsymbol{u}^k) \right) \boldsymbol{u}^{k,\,i} \;+\; \tau_i\, \boldsymbol{f}}{\alpha + \tau_i} \qquad (i = 0, \ldots, n-1)\,, \tag{5.14}$$

where the time step sizes τ_i are chosen according to Eq. (3.9).

Fast-Jacobi Scheme. Instead of a parabolic evolution, we now want to solve the Euler-Lagrange equation (5.9) by means of the Fast-Jacobi method. The discretisation of Eq. (5.9) yields a nonlinear system of equations:

$$\big(\boldsymbol{I} \,-\, \alpha\, \boldsymbol{A}(\boldsymbol{u}) \big)\, \boldsymbol{u} \;=\; \boldsymbol{f} \;. \tag{5.15}$$

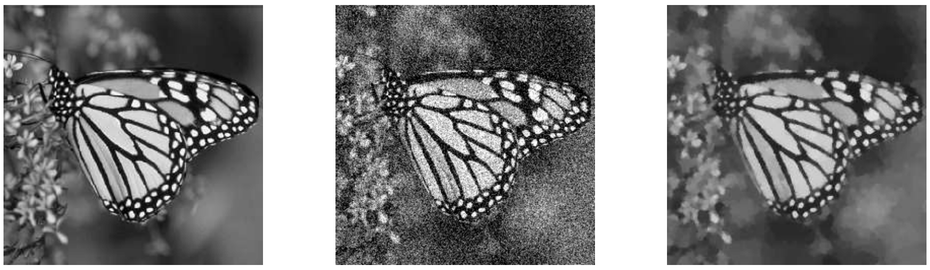

Figure 11: Test setting for Charbonnier regularisation. **(a) Left:** Original image (*monarch*, 256 × 256 pixels). **(b) Middle:** Noisy image (additive white Gaussian noise, $\sigma = 40$). **(c) Right:** Regularisation of the noisy image with $\lambda = 10^{-2}$ and $\alpha = 2500$.

It can be solved by the fixed point iteration

$$\underbrace{\big(\boldsymbol{I} - \alpha\, \boldsymbol{A}(\boldsymbol{u}^k)\big)}_{:=\,\boldsymbol{M}(\boldsymbol{u}^k)}\, \boldsymbol{u}^{k+1} \;=\; \boldsymbol{f} \qquad (k \geq 0). \tag{5.16}$$

Since the system matrix $\boldsymbol{M}(\boldsymbol{u}^k)$ is symmetric and positive definite, we can apply the Fast-Jacobi method with $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$:

$$\begin{aligned} \boldsymbol{u}^{k+1,\,i+1} &= \boldsymbol{u}^{k+1,\,i} + \omega_i\, \boldsymbol{D}^{-1}\Big(\boldsymbol{f} - \boldsymbol{M}(\boldsymbol{u}^k)\, \boldsymbol{u}^{k+1,\,i}\Big) \\ &= \Big(\boldsymbol{I} + \omega_i\, \alpha\, \boldsymbol{D}^{-1}\boldsymbol{A}(\boldsymbol{u}^k)\Big)\, \boldsymbol{u}^{k+1,\,i} + \omega_i\, \boldsymbol{D}^{-1}\big(\boldsymbol{f} - \boldsymbol{u}^{k+1,\,i}\big)\,. \end{aligned} \tag{5.17}$$

After the complete cycle with length n, we can set $\boldsymbol{u}^{k+1} := \boldsymbol{u}^{k+1,\,n}$ and update the nonlinearities $\boldsymbol{A}(\boldsymbol{u}^k)$ by $\boldsymbol{A}(\boldsymbol{u}^{k+1})$. With Gershgorin's theorem [21], one can safely estimate $\omega_{\max} = \frac{2}{\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{M}(\cdot))}$ by 1.

Experimental Evaluation. Our testbed is depicted in Fig. 11. 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 $\alpha = 2500$ and the contrast parameter $\lambda = 10^{-2}$. The corresponding reference solution in Fig. 11(c) has been computed by means of the Jacobi method with 100000 iterations and nonlinear updates after each iteration.

The first experiment in Fig. 12(a) deals with the comparison of FED and Fast-Jacobi. Both approaches use a cycle length of 25. Since $\tau_{\max} = 0.25$, this corresponds to the diffusion time $T = 0.25 \cdot \frac{25\cdot 26}{3} \approx 54.17$ per FED cycle. As one can see in Fig. 12(a), the speed of convergence is significantly

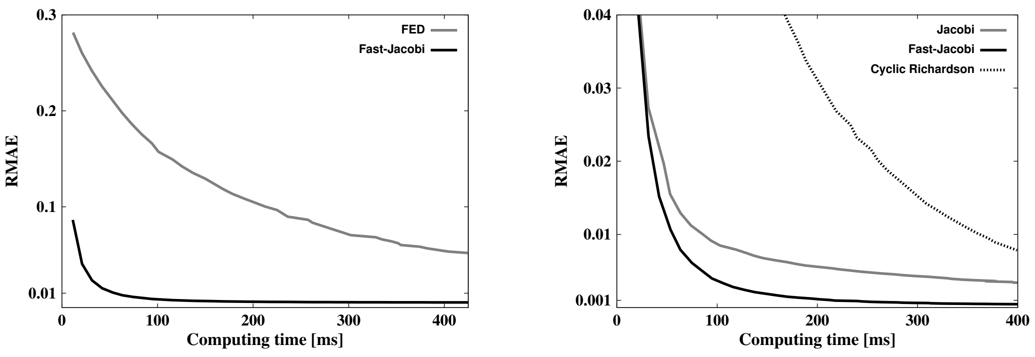

Figure 12: Computing time (milliseconds) vs. RMAE for Charbonnier regularisation. **(a) Left:** Comparison of FED and FJ with cycle length 25. **(b) Right:** Comparison of Jacobi, Fast-Jacobi and cyclic Richardson with cycle length 25.

higher for Fast-Jacobi than for FED. This shows that for elliptic problems with strongly varying coefficients, Fast-Jacobi should be preferred over FED.

In our second experiment, we compare Fast-Jacobi with its noncyclic JOR ancestor and the classical cyclic Richardson method. Since $\omega_{\max}$ was estimated by 1, the JOR method comes down to the Jacobi method. The cyclic Richardson method uses relaxation parameters from (4.8) where $\lambda_{\max}$ was obtained with the Gershgorin estimate $1 + 8\alpha$. The results are depicted in Fig. 12(b), where we again have used a cycle length of 25. We observe that Fast-Jacobi outperforms the Jacobi method which shows the usefulness of varying relaxation parameters. Interestingly, cyclic Richardson is inferior to both the Jacobi and the Fast-Jacobi method. For our problem with varying coefficients, it suffers from the lack of diagonal scaling that is inherent in Jacobi-type methods. Moreover, its use of MV relaxation parameters instead of box relaxation parameters gives worse attenuation of the noisy high frequencies.

In summary, we have demonstrated that Fast-Jacobi performs much better than a parabolic FED approach for elliptic problems with strongly varying coefficients. It is also a favourable choice among Jacobi-like solvers, since it combines fast convergence due to varying relaxation parameters with good attenuation properties of high frequencies.

It should be noted that Fast-Jacobi methods can cope well with strongly varying coefficients, but these coefficients must not become singular. This excludes e.g. total variation denoising [39] as long as no regularisation of the singularity is applied. To handle variational methods with singularities, spe-

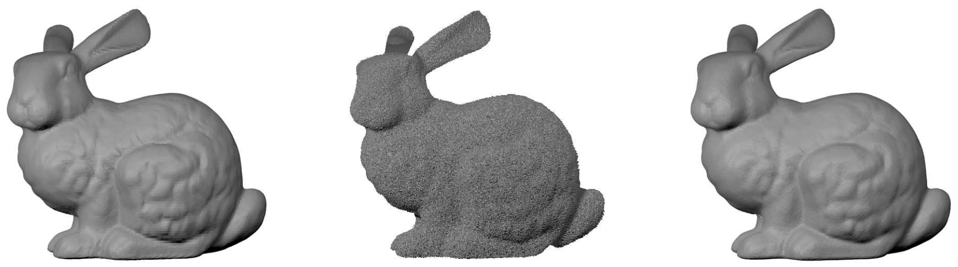

Figure 13: Test setting for anisotropic range image integration. **(a) Left:** One rendered image of *Stanford Bunny.* **(b) Middle:** Noisy range surface. **(c) Right:** Reconstruction computed with Fast-Jacobi ($\omega_{\max} = 0.3$, $n = 50$, 10 cycles).

cific nonsmooth optimisation algorithms based on primal-dual formulations have been developed; see e.g. [40, 41].

5.5 Higher Dimensional Problems and 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 [13]. In this subsection, we illustrate that this also holds for Fast-Jacobi and 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 [42, 43, 44]. Following the ideas in [42], the most important step of anisotropic range image integration is solving an elliptic PDE that can be written as

$$p(u)\, u - \alpha \, \mathrm{div}\left(\Psi'_S(\boldsymbol{J})\, \boldsymbol{\nabla} u\right) \;=\; q(u)\;, \tag{5.18}$$

where $u : \mathbb{R}^3 \to \mathbb{R}$ is the unknown solution. Note that $p(u)$ and $q(u)$ are suitably chosen, real-valued functions. Furthermore, $\boldsymbol{J} \in \mathbb{R}^{3\times 3}$ is the 3-D structure tensor, and the matrix valued function Ψ'_S yields the anisotropic behaviour. The parameter α denotes a positive smoothness weight. More details can be found in [42].

After the discretisation this corresponds to a nonlinear system with N equations

$$\left(\boldsymbol{P}(\boldsymbol{u}) - \alpha\, \boldsymbol{A}(\boldsymbol{u})\right) \boldsymbol{u} \;=\; \boldsymbol{q}(\boldsymbol{u})\;, \tag{5.19}$$

where N is the number of voxels. The vectors $\boldsymbol{u}, \boldsymbol{q}(\boldsymbol{u}) \in \mathbb{R}^N$ are obtained by a spatial discretisation of the functions u and $q(u)$, respectively. The matrix

Table 6: Computing times (sec.) for the sequential CPU and the parallel GPU implementation of the Fast-Jacobi algorithm for anisotropic range image integration.

data size	**CPU** [s]	**GPU** [s]	**speed up factor**
$\mathbf{64^3}$	30.03	0.31	96.9
$\mathbf{128^3}$	239.70	1.70	141.0
$\mathbf{256^3}$	2006.05	12.93	155.1

$\boldsymbol{A}(\boldsymbol{u}) \in \mathbb{R}^{N\times N}$ is the 3-D discrete divergence operator with a diffusion tensor $\Psi'_S(\boldsymbol{J})$. Moreover, $\boldsymbol{P}(\boldsymbol{u}) := \mathrm{diag}(\boldsymbol{p}(\boldsymbol{u})) \in \mathbb{R}^{N\times N}$ with the discrete version $\boldsymbol{p}(\boldsymbol{u}) \in \mathbb{R}^N$ of $p(u)$. This nonlinear system can be solved in a way that is analogous to Eq. (5.15).

In our experiment, we compare the running times of a sequential CPU and a parallel GPU implementation of the Fast-Jacobi applied to Eq. (5.19). This comparison is shown in Table 6, where we use the 3.2 GHz Intel Xeon processor and a single GPU of the NVIDIA GeForce GTX 690, respectively. The number of unknowns is identical to the voxel number. In our example with the *Stanford bunny*[1] shown in Fig. 13, we have tested reconstructions with up to $256^3 \approx 16\cdot 10^6$ unknowns. Our numerical experiments have shown that $\omega_{\max} = 0.3$ provides stable results, and convergence was achieved with 10 cycles of length 50. As we see in Table 6, the parallel implementation is up to 155 times faster than its sequential counterpart. In conclusion, our experiment illustrates that 3-D implementations of cyclic methods do not create additional challenges, and their parallelisation is straightforward and highly beneficial.

More generally, it should be mentioned that cyclic methods share the advantages of many iterative methods for solving linear and nonlinear systems of equations: Often a few iterations are sufficient to produce an approximation to the desired solutions with acceptable accuracy, while a user who can afford longer runtimes is rewarded by higher accuracy. In contrast to direct algorithms, cyclic methods perform only simple matrix-vector multiplications as well as additions, subtractions and scaling of vectors. Thus, they may fully exploit the sparsity of the matrices and do not require a huge memory overhead. This makes cyclic methods ideal algorithms for problems that require efficient parallel processing of huge datasets with increasing accuracy in time.

[1]taken from the Stanford 3-D scanning repository

6 Conclusions

We have shown that two of the simplest methods from numerical analysis of PDEs can lead to remarkably efficient algorithms when they are only slightly modified: This has led us to cyclic variants of the explicit scheme and the Jacobi method. By means of five prototypical scenarios, we have demonstrated that these cyclic schemes are widely applicable to all kinds of elliptic and parabolic problems in PDE-based image analysis that lead to symmetric matrices. We conjecture that they may also be applicable to some nonsymmetric problems; see e.g. [45, 46] for some related results. Since this requires a more complicated stability analysis, this is left for future research.

Although cyclic algorithms are around in the numerical analysis community for a long time, their use in image analysis is novel. However, this transfer of knowledge is not a one-way road: By considering a factorisation of general smoothing filters, we have also introduced novel, signal processing based ways of deriving cycle parameters to the numerical analysis community. They have led to hitherto unexplored methods with alternative parameter cycles. These methods have better smoothing properties than classical numerical concepts such as Super Time Stepping and the cyclic Richardson algorithm.

In the past, cyclic approaches have never been the most popular methods for the numerical solution of PDEs. With the widespread availability of low cost parallel computing hardware and the growing demand for simple algorithms that work for a broad class of problems, the situation has changed substantially. It seems that more than 100 years after Richardson's seminal work [9], cyclic methods are finally getting the merits they deserve.

Public Domain Code. Since we are convinced that the best way to experience the advantages of cyclic methods is to test them on own problems, we have developed a library that offers FED functionality for arbitrary diffusion processes. It is easy to embed into C and C++ programmes. This library is freely available from our website

`http://www.mia.uni-saarland.de/Research/SC_FED.shtml`.

Acknowledgements. We gratefully acknowledge funding by the German Research Foundation (DFG) through project We 2602/7-1 and the Gottfried Wilhelm Leibniz Prize We 2602/9-1.

References

[1] 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.

[2] J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner, Stuttgart, 1998.

[3] 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.

[4] 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.

[5] 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).

[6] V. K. Saul'yev. *Integration of Equations of Parabolic Type by the Method of Nets.* Pergamon, Oxford, 1964.

[7] W. Gentzsch and A. Schlüter. Über ein Einschrittverfahren mit zyklischer Schrittweitenänderung zur Lösung parabolischer Differentialgleichungen. *ZAMM, Zeitschrift für Angewandte Mathematik und Mechanik*, 58:T415–T416, 1978. (in German).

[8] 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, *Proceedings of the Third GAMM-Conference on Numerical Methods in Fluid Mechanics*, pages 109–117. Friedr. Vieweg & Sohn, 1979.

[9] 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.

[10] D. Young. On Richardson's method for solving linear systems with positive definite matrices. *Journal of Mathematics and Physics*, 32:243–255, 1954.

[11] 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.

[12] 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, Berlin, 2010. Springer.

[13] 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. Springer Berlin Heidelberg, 2012.

[14] 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.

[15] 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).

[16] 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.

[17] R. Ben-Ari and G. Raveh. Variational depth from defocus in real-time. In *Computer Vision Workshops, 2011 IEEE Int. Conf. on Computer Vision*, pages 522–529, 2011.

[18] S. Setzer, G. Steidl, and J. Morgenthaler. On cyclic gradient descent reprojection. Technical report, Department of Mathematics, Technical University of Kaiserslautern, 2011.

[19] G. Hellwig. *Partial Differential Equations.* Teubner, Stuttgart, 1977.

[20] W. M. Wells. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:234–239, 1986.

[21] R. S. Varga. *Matrix Iterative Analysis.* Prentice Hall, Englewood Cliffs, 1962.

[22] 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.

[23] 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.

[24] L. Reichel. Newton interpolation at Leja points. *BIT Numerical Mathematics*, 30(2):332–346, 1990.

[25] D. Calvetti and L. Reichel. On the evaluation of polynomial coefficients. *Numerical Algorithms*, 33(1–4):153–161, 2003.

[26] D. Calvetti and L. Reichel. Adaptive Richardson iteration based on Leja points. *Journal of Computational and Applied Mathematics*, 71:267–286, 1996.

[27] V. I. Lebedev and S. A. Finogenov. The order of choices of the iteration parameters in the cyclic Chebyshev iteration method. *Zhurnal Vychislitel'noy Matematiki i Matematicheskoy Fiziki*, 11(2):425–438, 1971. (in Russian).

[28] F. Bornemann and P. Deuflhard. The cascadic multigrid method for elliptic problems. *Numerische Mathematik*, 75:135–152, 1996.

[29] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[30] W. Hackbusch. *Multigrid Methods and Applications.* Springer, New York, 1985.

[31] 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.

[32] J. Weickert. Applications of nonlinear diffusion in image processing and computer vision. *Acta Mathematica Universitatis Comenianae*, 70(1):33–50, 2001.

[33] J. Weickert. Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 31(2/3):111–127, April 1999.

[34] 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, 1987.

[35] 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.

[36] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, second edition, 2003.

[37] 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.

[38] 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*, volume 2, pages 168–172, Austin, TX, November 1994. IEEE Computer Society Press.

[39] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[40] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1–2):89–97, 2004.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] G. Opfer and G. Schober. Richardson's iteration for nonsymmetric matrices. *Linear Algebra and its Applications*, 58:343–361, 1984.

[47] 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.

Appendix

A.1 Proof of Theorem 1 (Diffusion Interpretation of Smoothing Kernels)

At first, we prove by induction that

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

For $m = 0$, Eq. (A.1) obviously holds.

If we assume that Eq. (A.1) is valid for an arbitrary $m \geq 0$, this yields for $m{+}1$:

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

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

$$\binom{2m}{-1} \;=\; \binom{2m}{2m+1} \;=\; 0 \;. \tag{A.3}$$

This yields

$$\begin{aligned}
&\frac{1}{h^{2(m+1)}} \Biggl(f_{i+1+m} \,+\, \sum_{k=-m}^{m} (-1)^{m+k-1} \binom{2m}{m+k-1} f_{i+k} \\
&\qquad -\, 2 \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} f_{i+k} \\
&\qquad +\, \sum_{k=-m}^{m} (-1)^{m+k+1} \binom{2m}{m+k+1} f_{i+k} \,+\, f_{i-1-m} \Biggr) \\
&= \frac{1}{h^{2(m+1)}} \Biggl(f_{i+(m+1)} \,+\, f_{i-(m+1)} \,+\, \sum_{k=-m}^{m} (-1)^{m+k+1}\, f_{i+k} \,\cdot \\
&\qquad\qquad \cdot \left(\binom{2m}{m+k+1} \,+\, 2\binom{2m}{m+k} \,+\, \binom{2m}{m+k-1} \right) \Biggr) \,. \qquad \text{(A.4)}
\end{aligned}$$

By using the relation

$$\binom{2m}{m+k+1} \,+\, 2\binom{2m}{m+k} \,+\, \binom{2m}{m+k-1} \,=\, \binom{2m+2}{m+k+1} \,, \qquad \text{(A.5)}$$

we finally get

$$\begin{aligned}
&\frac{1}{h^{2(m+1)}} \left(f_{i+(m+1)} \,+\, f_{i-(m+1)} \,+\, \sum_{k=-m}^{m} (-1)^{(m+1)+k} \binom{2(m+1)}{(m+1)+k} f_{i+k} \right) \\
&= \frac{1}{h^{2(m+1)}} \left(\sum_{k=-(m+1)}^{m+1} (-1)^{(m+1)+k} \binom{2(m+1)}{(m+1)+k} f_{i+k} \right) \,. \qquad \text{(A.6)}
\end{aligned}$$

This concludes the proof of Eq. (A.1).

If we replace Δ_h^m by the left hand side of Eq. (A.1), then

$$\sum_{m=0}^{n} \alpha_m^{(n)} \cdot \Delta_h^m f_i \,=\, \sum_{m=0}^{n} \frac{\alpha_m^{(n)}}{h^{2m}} \cdot \sum_{k=-m}^{m} (-1)^{m+k} \binom{2m}{m+k} f_{i+k} \,. \qquad \text{(A.7)}$$

On the other hand we have the given (symmetric) filter

$$L_{2n+1}^h\, f_i \;=\; \sum_{k=-n}^{n} w_{|k|} \cdot f_{i+k} \qquad \text{(A.8)}$$

with the weights $w_0, \ldots, w_n \in \mathbb{R}$. Comparing the two equations (A.7) and (A.8) yields a system of $n+1$ linear equations with $n+1$ unknowns $\alpha_m^{(n)}$:

$$\sum_{m=k}^{n} (-1)^{m+k} \cdot \frac{1}{h^{2m}} \binom{2m}{m+k} \alpha_m^{(n)} \;=\; w_k \qquad \forall\, k \in \{0, ..., n\}\,. \tag{A.9}$$

The corresponding matrix-vector notation of Eq. (A.9) is given by

$$\boldsymbol{B}\, \boldsymbol{\alpha^{(n)}} \;=\; \boldsymbol{w}\ , \tag{A.10}$$

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

$$b_{k,m} \;=\; (-1)^{m+k} \cdot \frac{1}{h^{2m}} \binom{2m}{m+k} \qquad (k, m = 0, \ldots, n). \tag{A.11}$$

Since $b_{k,m} = 0$ for $k > m$ and $b_{k,k} = 1/h^{2k} \neq 0$, it follows that $\boldsymbol{B}$ is a regular upper triangular matrix. Therefore, Eq. (A.10) has the unique solution $\boldsymbol{\alpha^{(n)}} = \boldsymbol{B}^{-1}\boldsymbol{w}$. This shows that the coefficients $\alpha_m^{(n)}$ uniquely depend on the weights of the filter kernel.

To obtain a closed-form expression for the coefficients, we want to determine an explicit representation of the matrix $\boldsymbol{B}^{-1}$. To this end, we show that the entries of $\boldsymbol{B}^{-1} = \left(b_{k,m}^{-1}\right)$ are given by

$$b_{k,m}^{-1} \;=\; h^{2k} \left(\binom{m+k}{2k} + (1 - \delta_{m+k,0}) \cdot \binom{m+k-1}{2k} \right) \quad (k, m = 0, \ldots, n). \tag{A.12}$$

This means that we have to verify

$$\sum_{p=0}^{n} b_{k,p}^{-1} \cdot b_{p,m} \;=\; \delta_{k,m}\ . \tag{A.13}$$

Since both $\boldsymbol{B}$ and $\boldsymbol{B}^{-1}$ are upper triangular matrices, the summation is only necessary for $p \in \{k, ..., m\}$. Thus, the above equation can be simplified to

$$\sum_{p=k}^{m} b_{k,p}^{-1} \cdot b_{p,m} \;=\; \delta_{k,m}\ . \tag{A.14}$$

This equation obviously holds for $k > m$. If $k = m$, then it is also valid because of

$$b_{k,k}^{-1} \;=\; h^{2k} \;=\; \frac{1}{b_{k,k}}\,. \tag{A.15}$$

Let now $m > k \geq 0$. Then this yields

$$\sum_{p=k}^{m} b^{-1}_{k,p} \cdot b_{p,m} \;=\; h^{2(k-m)} \cdot \sum_{p=k}^{m} \left(\binom{p+k}{2k} \;+\; (1 \;-\; \delta_{p+k,\,0}) \binom{p+k-1}{2k} \right) \cdot$$
$$\cdot \; (-1)^{m+p} \binom{2m}{m+p} \;. \qquad \text{(A.16)}$$

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

$$s(p) \;:=\; \binom{p+k}{2k} \;+\; \binom{p+k-1}{2k} \;=\; \prod_{j=1}^{2k} \frac{p + (k+1-j)}{j} \;+\; \prod_{j=1}^{2k} \frac{p + (k-j)}{j} \qquad \text{(A.17)}$$

fulfils $s(p) = s(-p)$ and $s(p) = 0$ for $p \in \{-k{+}1, ..., k{-}1\}$.
With the help of this, we get

$$\begin{aligned} 0 \;&\overset{\text{(A.19)}}{=}\; \sum_{p=-m}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \\ &=\; \sum_{p=-m}^{-k} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} + \sum_{p=k}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \\ &=\; 2 \cdot \sum_{p=k}^{m} s(p) \cdot (-1)^{m+p} \binom{2m}{m+p} \;, \qquad \text{(A.18)} \end{aligned}$$

where we have used that

$$\sum_{j=0}^{r} (-1)^j \; P(j) \binom{r}{j} \;=\; 0 \qquad \text{(A.19)}$$

for any polynomial $P(j)$ with degree less than $r > 0$.
In the case of $k = 0$, we have $b^{-1}_{0,0} = 1$ and $b^{-1}_{0,p} = 2$ for $p \neq 0$. Hence,

$$\begin{aligned} \sum_{p=0}^{m} b^{-1}_{0,p} \cdot (-1)^{m+p} \binom{2m}{m+p} \;&=\; (-1)^m \binom{2m}{m} + 2 \cdot \sum_{p=1}^{m} (-1)^{m+p} \binom{2m}{m+p} \\ &=\; (-1)^m \binom{2m}{m} + \sum_{\substack{p=-m \\ p \neq 0}}^{m} (-1)^{m+p} \binom{2m}{m+p} \\ &=\; \sum_{p=-m}^{m} (-1)^{m+p} \binom{2m}{m+p} \;\overset{\text{(A.19)}}{=}\; 0 \;. \qquad \text{(A.20)} \end{aligned}$$

Considering the equations (A.18) and (A.20), it follows that

$$\sum_{p=k}^{m} b_{k,p}^{-1} \cdot b_{p,m} \;=\; 0 \tag{A.21}$$

for $m > k$. Thus, we get Eq. (2.6).

Now we assume that the weights w_k sum up to 1. According to the fundamental theorem of algebra, the polynomial $p_L(z)$ has n roots $z_0, ..., z_{n-1} \in \mathbb{C}$. Hence, it can be written as a product of n linear factors:

$$p_L(z) \;=\; c \cdot \prod_{m=0}^{n-1} (z_m - z) \;, \tag{A.22}$$

where $c \in \mathbb{R}$ is the normalisation factor

$$c \;=\; \alpha_0^{(n)} \cdot \left(\prod_{m=0}^{n-1} z_m \right)^{-1} \;. \tag{A.23}$$

Since the weights satisfy $w_k = w_{-k}$ and sum up to 1, we have

$$\alpha_0^{(n)} \;=\; \sum_{k=0}^{n} \left(\binom{k}{0} \;+\; (1 - \delta_{k,0}) \binom{k-1}{0} \right) w_k \;=\; w_0 \;+\; 2 \sum_{k=1}^{n} w_k \;=\; 1 \;. \tag{A.24}$$

This implies that $p_L(0) = \alpha_0^{(n)} = 1$ and the polynomial $p_L(z)$ can be rewritten as

$$p_L(z) \;=\; \prod_{m=0}^{n-1} \left(1 \;-\; \frac{z}{z_m} \right) \;. \tag{A.25}$$

Replacing $-z$ by the operator Δ_h and interpreting the corresponding product as a composition of operators finally shows that

$$L_{2n+1}^{h} \;=\; \prod_{i=0}^{n-1} \left(I \;+\; z_i^{-1} \, \Delta_h \right) \;, \tag{A.26}$$

where I denotes the identity operator. Obviously, the right hand side of Eq. (A.26) is a series of explicit diffusion steps.

Thus, Eq. (A.26) states that L_{2n+1}^{h} can be decomposed into n explicit linear diffusion steps with the time step sizes $\tau_i = z_i^{-1}$. Because of

$$\sum_{i=0}^{n-1} \tau_i \;=\; \sum_{i=0}^{n-1} z_i^{-1} \;=\; \alpha_1^{(n)} \;, \tag{A.27}$$

the cycle time of the scheme in Eq. (A.26) is equal to the coefficient

$$\alpha_1^{(n)} \;=\; h^2 \cdot \sum_{k=1}^{n} \left(\binom{k+1}{2} + \binom{k}{2} \right) w_k \;=\; h^2 \cdot \sum_{k=1}^{n} k^2 \cdot w_k \;, \qquad \text{(A.28)}$$

and the theorem is proven.

A.2 Factorisation of the Binomial Filter Kernel

A binomial kernel K^h_{2n+1} of length $(2n+1)h$ is defined by the weights

$$w_k \;=\; \frac{1}{4^n} \binom{2n}{n+k} \,. \qquad \text{(A.29)}$$

By construction, the nice property of binomial kernels is that the (discrete) convolution of two kernels is again a binomial kernel:

$$\left(K^h_{2n_1+1} * K^h_{2n_2+1}\right) \;=\; K^h_{2(n_1+n_2)+1} \qquad \forall n_1, n_2 \in \mathbb{N}. \qquad \text{(A.30)}$$

This means we can represent each binomial kernel by means of a convolution with kernels of length $3h$. More precisely, convolving K^h_3 with itself n–1 times yields K^h_{2n+1}. In terms of the polynomial, this corresponds to the n-th power of the polynomial belonging to the binomial kernel with length $3h$. Thus, to compute a filter factorisation, we simply have to consider a binomial kernel of length $3h$. Using Eq. (2.6) we get the two coefficients

$$\alpha_0^{(1)} \;=\; 1\,, \qquad \alpha_1^{(1)} \;=\; h^2 \cdot w_1^{(1)} \;=\; \frac{h^2}{4}\,. \qquad \text{(A.31)}$$

Hence, the polynomial of an arbitrary binomial kernel K^h_{2n+1} is given by

$$p_K(z) \;=\; \left(1 - \frac{h^2}{4} z\right)^n \;=\; \sum_{m=0}^{n} \binom{n}{m} \frac{h^{2m}}{4^m} (-z)^m \;. \qquad \text{(A.32)}$$

It has only a single root $z = \frac{4}{h^2}$ with multiplicity n. This implies a constant time step size $\tau = \frac{h^2}{4}$. Thus, the cycle time of K^h_{2n+1} is given by $\theta_n = \frac{h^2}{4} \cdot n$.

A.3 Factorisation of the Maximum Variance Filter Kernel

We consider the symmetric kernel M^h_{2n+1} that has positive weights only at the boundaries, i.e. $w_{-n} = w_n = \frac{1}{2}$ and $w_k = 0$ else. The coefficients $\alpha_m^{(n)}$

of p_M can be computed by means of Eq. (2.6):

$$\begin{aligned}
\alpha_m^{(n)} &= \frac{h^{2m}}{2} \cdot \left(\binom{n+m}{2m} + \binom{n+m-1}{2m} \right) \\
&= \frac{h^{2m}}{2} \cdot \binom{n+m}{2m} \left(1 + \frac{n-m}{n+m} \right) \\
&= h^{2m} \frac{n}{n+m} \binom{n+m}{2m} . \qquad \text{(A.33)}
\end{aligned}$$

Thus, the polynomial of M_{2n+1}^h is given by

$$p_M(z) = \sum_{m=0}^{n} h^{2m} \frac{n}{n+m} \binom{n+m}{2m} (-z)^m . \qquad \text{(A.34)}$$

The next step is to show that $p_M(z)$ is related to a Chebyshev polynomial of the first kind. They are defined by the recursion

$$\left\{ \begin{array}{rcl} T_0(x) &=& 1 \\ T_1(x) &=& x \\ T_{n+1}(x) &=& 2x \cdot T_n(x) - T_{n-1}(x) . \end{array} \right. \qquad \text{(A.35)}$$

and have the following closed-form representation [47]:

$$T_n(x) = \frac{n}{2} \cdot \sum_{m=0}^{\lfloor n/2 \rfloor} \frac{(-1)^m}{n-m} \binom{n-m}{m} (2x)^{n-2m} . \qquad \text{(A.36)}$$

Thus, we get

$$\begin{aligned}
p_M(z) &= n \sum_{m=0}^{n} \frac{(-1)^m}{n+m} \binom{n+m}{2m} (h^2 \cdot z)^m \\
&= n \sum_{m=0}^{n} \frac{(-1)^{n-m}}{2n-m} \binom{2n-m}{2(n-m)} \left(h^2 \cdot z\right)^{n-m} \\
&= (-1)^n \cdot \frac{2n}{2} \sum_{m=0}^{\lfloor 2n/2 \rfloor} \frac{(-1)^m}{2n-m} \binom{2n-m}{m} \left(h \cdot \sqrt{z}\right)^{2n-2m} \\
&= (-1)^n \cdot T_{2n} \left(\frac{h\sqrt{z}}{2} \right) . \qquad \text{(A.37)}
\end{aligned}$$

Note that we have changed the order of summation ($m \to n - m$) in the second step.

With this relation, the time step sizes are connected to the roots of the Chebyshev polynomials and can be computed as

$$\tau_i \;=\; \frac{h^2}{2} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n}\right)} \qquad (i = 0, ..., n-1). \tag{A.38}$$

According to Theorem 1, the cycle time is given by

$$\theta_n \;=\; h^2 \sum_{k=1}^{n} k^2 w_k \;=\; \frac{h^2}{2} \cdot n^2 \;. \tag{A.39}$$

A.4 Factorisation of the Box Filter Kernel

The weights w_k of a box filter with length $(2n+1)h$, B^h_{2n+1}, are uniform: $w_k = \frac{1}{2n+1}$. Its polynomial p_B is also related to Chebyshev polynomials.

According to Eq. (2.6) we have $\alpha_0^{(n)} = 1$, and for $m > 0$ we obtain

$$\begin{aligned}
\alpha_m^{(n)} \;&=\; \frac{h^{2m}}{2n+1} \sum_{k=m}^{n} \left(\binom{k+m}{2m} + \binom{k+m-1}{2m} \right) \\
&=\; \frac{h^{2m}}{2n+1} \left(\sum_{k=2m}^{n+m} \binom{k}{2m} + \sum_{k=2m}^{n+m-1} \binom{k}{2m} \right) \\
&=\; \frac{h^{2m}}{2n+1} \left(\binom{n+m+1}{2m+1} + \binom{n+m}{2m+1} \right) \\
&=\; \frac{h^{2m}}{2n+1} \left(\frac{n+m+1}{2m+1} \binom{n+m}{2m} + \frac{n-m}{2m+1} \binom{n+m}{2m} \right) \\
&=\; \frac{h^{2m}}{2n+1} \cdot \frac{2n+1}{2m+1} \binom{n+m}{2m} \;=\; \frac{h^{2m}}{2m+1} \binom{n+m}{2m} \;.
\end{aligned} \tag{A.40}$$

Thus, the polynomial $p_B(z)$ of the box filter B^h_{2n+1} is given by

$$p_B(z) \;=\; \sum_{m=0}^{n} \frac{h^{2m}}{2m+1} \binom{n+m}{2m} (-z)^m \;. \tag{A.41}$$

Furthermore, we can state that

$$\begin{aligned}
T_{2n+1}(x) \;&=\; \frac{2n+1}{2} \cdot \sum_{m=0}^{n} \frac{(-1)^m}{2n+1-m} \binom{2n+1-m}{2(n-m)+1} (2x)^{2(n-m)+1} \\
&=\; \frac{2n+1}{2} \cdot \sum_{m=0}^{n} \frac{(-1)^{n-m}}{n+m+1} \binom{n+m+1}{2m+1} (2x)^{2m+1} \\
&=\; (-1)^n (2n+1) x \cdot \sum_{m=0}^{n} \frac{(-1)^m}{2m+1} \binom{n+m}{2m} \left(4x^2\right)^m \;,
\end{aligned} \tag{A.42}$$

and hence for $z > 0$:

$$p_B(z) \;=\; (-1)^n \cdot \frac{2 \cdot T_{2n+1}\left(\frac{h\sqrt{z}}{2}\right)}{(2n+1)h\sqrt{z}} \;. \tag{A.43}$$

Note that this representation makes also sense for $z \to 0$, since

$$\lim_{z\to 0} \;(-1)^n \cdot \frac{2 \cdot T_{2n+1}\left(\frac{h\sqrt{z}}{2}\right)}{(2n+1)h\sqrt{z}} \;=\; 1 \;=\; p_B(0)\;. \tag{A.44}$$

Hence, the roots $z_0, \ldots, z_{n-1}$ of $p_B(z)$ are related to the n positive roots of T_{2n+1}. Since these roots are given by

$$x_i \;=\; \cos\left(\pi \cdot \frac{2i+1}{4n+2}\right) \qquad (0 \le i \le n{-}1), \tag{A.45}$$

the roots z_i of p_B fulfil

$$z_i \;=\; \frac{4}{h^2}\cdot x_i^2 \;=\; \frac{4}{h^2}\cdot \cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)\;. \tag{A.46}$$

This yields the following conclusion, which is a special case of Theorem 1: The 1-D discrete box filtering with B^h_{2n+1} is equivalent to a cycle of n explicit 1-D linear diffusion steps with the time step sizes

$$\tau_i \;=\; \frac{h^2}{2}\cdot\frac{1}{2\cos^2\left(\pi\cdot\frac{2i+1}{4n+2}\right)} \qquad (i = 0, ..., n{-}1). \tag{A.47}$$

With Theorem 1, the corresponding cycle time also grows quadratically in n:

$$\theta_n \;=\; h^2 \sum_{k=1}^{n} k^2 w_k \;=\; \frac{h^2}{2n+1}\sum_{k=1}^{n} k^2 \;=\; \frac{h^2}{6}\cdot\left(n^2+n\right)\;. \tag{A.48}$$

A.5 Stability Analysis of the FED Scheme

The result $\boldsymbol{u}^{k+1} \in \mathbb{R}^N$ with $k \ge 0$ after a complete cycle of the linear FED scheme (3.4) can be written as the matrix-vector product

$$\boldsymbol{u}^{k+1} \;=\; \underbrace{\left(\prod_{i=0}^{n-1}\left(\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{A}\right)\right)}_{=:\boldsymbol{Q_A}} \boldsymbol{u}^k\;. \tag{A.49}$$

Let us now analyse the eigenvalues $\lambda_1, \ldots, \lambda_N$ of the matrix $\boldsymbol{Q}_{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$. To this end, we consider the eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ of $\boldsymbol{A}$ with the corresponding eigenvalues $\mu_1(\boldsymbol{A}), \ldots, \mu_N(\boldsymbol{A}) \in \left[-\frac{4}{h^2}\,, 0\right]$. We have

$$\boldsymbol{Q}_{\boldsymbol{A}}\,\boldsymbol{v}_j \;=\; \Big(\prod_{i=0}^{n-1} (\boldsymbol{I} \,+\, \tau_i\,\boldsymbol{A})\Big)\,\boldsymbol{v}_j \;=\; \prod_{i=0}^{n-1} \big(1 \,+\, \tau_i\,\mu_j(\boldsymbol{A})\big)\boldsymbol{v}_j\;. \tag{A.50}$$

This means that the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ are also eigenvectors of $\boldsymbol{Q}_{\boldsymbol{A}}$ with eigenvalues

$$\lambda_j \;=\; \prod_{i=0}^{n-1} \big(1 \,+\, \tau_i\,\mu_j(\boldsymbol{A})\big) \qquad (j = 1, \ldots, N). \tag{A.51}$$

From the filter factorisation of the box filter, we know that

$$\prod_{i=0}^{n-1} \big(1 \,-\, \tau_i\,z\big) \;=\; p_B(z)\;, \tag{A.52}$$

which implies $\lambda_j \;=\; p_B(-\mu_j(\boldsymbol{A}))$ for $j = 1, \ldots, N$. Since the polynomial $p_B(z)$ satisfies $|p_B(z)| \leq 1$ for $z \in \left[0, \frac{4}{h^2}\right]$, we have $\lambda_j \in [-1, 1]$, i.e. the linear FED scheme is, as expected, in particular stable w.r.t. the Euclidean norm.

If we just replace the matrix $\boldsymbol{A}$ by an arbitrary symmetric and negative semidefinite matrix $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ whose eigenvalues $\mu_1(\boldsymbol{P}), \ldots, \mu_N(\boldsymbol{P})$ lie in $[-\mu_{\max}(\boldsymbol{P})\,, 0]$, an analogue computation for the eigenvalues $\lambda'_1, \ldots, \lambda'_N$ of $\boldsymbol{Q}_{\boldsymbol{P}}$ yields

$$\lambda'_j \;=\; \prod_{i=0}^{n-1} \big(1 \,+\, \tau_i\,\mu_j(\boldsymbol{P})\big) \;=\; p_B(-\mu_j(\boldsymbol{P})) \qquad (j = 1, \ldots, N). \tag{A.53}$$

Note that we still use the time step sizes τ_i of the linear FED scheme. Since it could happen that $\mu_{\max}(\boldsymbol{P}) > \frac{4}{h^2}$, we can not guarantee that the polynomial fulfils $|p_B(-\mu_j(\boldsymbol{B}))| \leq 1$ for all j. However, if we take more general time step sizes τ'_i as proposed in Eq. (3.9), i.e. replace the above time steps τ_i by $\tau'_i := \frac{2}{\mu_{\max}(\boldsymbol{P})} \cdot \frac{2}{h^2} \cdot \tau_i$, then we obtain the eigenvalues

$$\lambda'_j \;=\; \prod_{i=0}^{n-1} \big(1 \,+\, \tau'_i\,\mu_j(\boldsymbol{P})\big) \;=\; p_B\left(-\frac{4}{h^2}\cdot\frac{\mu_j(\boldsymbol{P})}{\mu_{\max}(\boldsymbol{P})}\right) \qquad (j = 1, \ldots, N). \tag{A.54}$$

Since $-\frac{\mu_j(\boldsymbol{P})}{\mu_{\max}(\boldsymbol{P})} \in [0, 1]$, we can now state that

$$\left|p_B\left(-\frac{4}{h^2}\cdot\frac{\mu_j(\boldsymbol{P})}{\mu_{\max}(\boldsymbol{P})}\right)\right| \;\leq\; 1 \qquad (j = 1, \ldots, N)\,, \tag{A.55}$$

and thus $\lambda'_j \in [-1, 1]$ for all j. This yields stability in the Euclidean norm for the cyclic scheme with the matrix $\boldsymbol{P}$ and the time step sizes from Eq. (3.9).