Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 327 (revised)

# Cyclic Schemes for PDE-Based Image Analysis

Joachim Weickert, Sven Grewenig,
Christopher Schroers and Andrés Bruhn

Saarbrücken 2015

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 327 (revised)
submitted: April 18, 2015

# Cyclic Schemes for PDE-Based Image Analysis

**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

**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

**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**

We investigate a class of efficient numerical algorithms for many partial differential equations (PDEs) in image analysis. They are applicable to parabolic or elliptic PDEs that have bounded coefficients and lead to space discretisations with symmetric matrices. Our schemes are easy to implement and well-suited for parallel implementations on GPUs, since they are based on the explicit diffusion scheme in the parabolic case, and the Jacobi method in the elliptic case. By supplementing these methods with cyclically varying time step sizes or relaxation parameters, we achieve efficiency gains of several orders of magnitude. We call the resulting algorithms Fast Explicit Diffusion (FED) and Fast Jacobi (FJ) methods. To achieve a good compromise between efficiency and accuracy, we show that one should use parameter cycles that result from factorisations of box filters. For these cycles we establish stability results in the Euclidean norm. Our schemes perform favourably in a number of applications, including 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, and they can also be interpreted as efficient first order methods for smooth optimisation problems.

# 1 Introduction

Solving image analysis problems with smooth parabolic or elliptic partial differential equations (PDEs) can involve a number of numerical challenges. Let us illustrate this by some examples.

- In nonlinear diffusion filtering, the diffusivity can be a function with a range over many orders of magnitude [46]. Thus, large stopping times are needed to achieve a desired degree of smoothing.
- Some filters are designed to allow strong anisotropies [61]. This excludes many schemes that are efficient for isotropic problems, and $L^\infty$-stability may be violated.
- The data domain is not restricted to the 2-D scenario: Three-dimensional problems are fairly common. Since they involve a huge amount of data, finding efficient algorithms is indispensable.
- Also higher order PDEs are used, e.g. for inpainting problems. In this case, explicit schemes require very small time step sizes, while implicit

approaches are burdensome due to the larger stencil size that is required for approximating higher order derivatives.

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. Examples include semi-implicit finite difference schemes [61], adaptive finite elements [6], adaptive finite volume methods [34], additive operator splittings [65], lattice Boltzmann techniques [33], locally analytic schemes [66], short time kernel approaches [59], vector extrapolation strategies [52], and multigrid methods [12].
Clearly, it would be desirable to have very 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 that can satisfy these requirements. We present 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 fairly 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 overrelaxation (JOR) method such that the relaxation parameter is no longer fixed, but varied in a cyclic way, too. We 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 a black box solver that is only modified by adapting its time step size or relaxation parameter in a cyclic way. These concepts have a 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 that have bounded coefficients and yield discretisations with symmetric matrices. These PDEs can be linear or nonlinear, isotropic

or anisotropic, of second or higher order, and two-dimensional or higher-dimensional. In their current formulation, our numerical methods are not designed for processes whose PDE formulation involves unbounded coefficients – such as total variation regularisation [53] – or whose discretisation leads to unsymmetric matrices – such as osmosis filtering [64].

**Related Work.** The present paper is the journal version of our conference article [27], in which we have introduced FED schemes. To our knowledge, this was the first time when cyclic schemes have been used in the image analysis community.
Our FED methods can be regarded as variants of the so-called *Super Time Stepping (STS)* schemes of [70], [55], and [25], while FJ techniques are related to *Cyclic Richardson* algorithms [51, 68, 5]. Although Super Time Stepping and Cyclic Richardson methods have been around in the numerical analysis community for many years, they have never become very popular: In the parabolic case, implicit methods [19, 35] and operator splitting techniques [45] have been favoured. In the elliptic setting, early cyclic methods have been suffering from numerical stability problems and had to compete with other efficient iterative solvers like SOR [22, 69]. Later on research on alternative methods such as preconditioned conjugate gradients [40] or multigrid techniques [11, 30] has dominated the field. We show that with the wide availability of GPUs, the intrinsic parallelism of cyclic algorithms and their simplicity makes them the methods of choice for many PDE-based image analysis problems.
It should be emphasised that we do not simply apply classical cyclic methods to these problems, 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 elliptic problems with higher efficiency.
The present paper extends our conference article [27] in a number of ways:

- We provide detailed proofs and 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 extend the application domain of our numerical schemes to additional tasks, in particular to problems of higher order and higher

dimensionality. These applications also cover implementations on modern GPUs.

It should be mentioned that our FED conference paper [27] has already found its way into a number of applications in image processing and computer vision. They include PDE-based compression of depth maps [32] and volumetric data sets [47], fast filtering methods on smartphones [38], multiscale feature detection [2], optic flow computation [29], variational depth-from-defocus [7], medical image registration [39, 56], and massively parallel analysis of functional data [49]. It has also inspired cyclic projected gradient methods for convex optimisation problems [58].

**Organisation of the Paper.** Section 2 establishes the connection between linear symmetric filters and explicit homogeneous diffusion schemes in one dimension. We illustrate this connection by means of three examples for filters whose iterative application approximates Gaussian kernels. In Section 3 we use these insights to construct our FED approach for parabolic problems. 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 six 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 homogeneous diffusion schemes with varying time step sizes. This derivation is based on a factorisation of the kernels.

## 2.1 Diffusion Interpretation of Symmetric 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 linear filter $L^h_{2n+1}$ of finite length $(2n+1)h$, $n \in \mathbb{N}_0$, by

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

where the weights $w_k \in \mathbb{R}$ of the filter kernel satisfy $w_k = w_{-k}$ for all $k \in \{1, ..., n\}$. Incorporating the pixel size $h$ in the notion of the filter length allows to interpret the discrete weights as the result of sampling a continuous weight function $w(y)$ at $2n+1$ equidistant grid points $y_{-n},...,y_0,...,y_n$. This

is necessary for a consistent approximation of the continuous PDEs that we will consider later on.

An interesting linear filter can be constructed from the central finite difference approximation to the second order derivative:

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

It can be used for the numerical solution of the one-dimensional homogeneous diffusion equation

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

To see this, let us consider some grid point $x_j$, a time step size $\tau > 0$ and a discrete point in time $t_k := k \cdot \tau$. Then an explicit discretisation of the diffusion equation in $(x_j, t_k)$ is given by the discrete symmetric filter

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

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

The theorem below allows to express any normalised discrete symmetric filter $L_{2n+1}^h$ in terms of $n$ explicit diffusion steps. It will be highly useful for our work. First we show that every discrete symmetric filter 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 leads to a representation by means of explicit diffusion steps with suitably chosen time step sizes.

**Theorem 1 (Diffusion Factorisation of Symmetric Filters).** *Let $L_{2n+1}^h$ be an arbitrary discrete symmetric linear 1-D filter. Then the representation* (2.5) *is unique. Its coefficients are given by*

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

*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 homogeneous diffusion steps:*

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

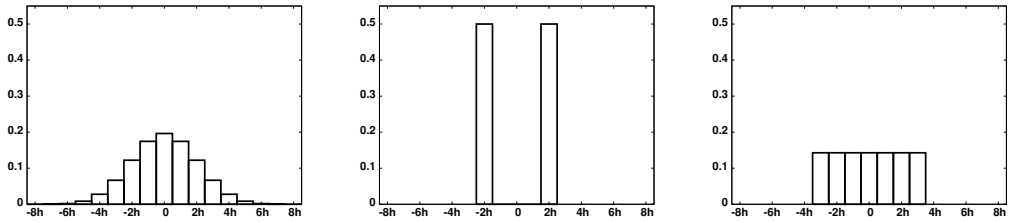


Figure 1: Illustration of three kernels that correspond to a diffusion time of $2h^2$. **(a) Left:** Binomial kernel with length $17h$. **(b) Middle:** MV kernel with length $5h$. **(c) Right:** Box kernel with length $7h$.

*The time step sizes* $\tau_i$ *satisfy* $\tau_i = \frac{1}{z_i}$, *where the* $z_i$ *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*

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

The closed-form expression (2.6) can be established by induction, comparison of coefficients, and explicit matrix inversion. The remaining part of the theorem is a consequence of the fundamental theorem of algebra. For a detailed proof we refer to the Appendix A.1.

## 2.2 Three Examples for Filter Factorisations

Let us now illustrate the preceding theorem by analysing three normalised symmetric filter kernels that are depicted in Figure 1: the binomial kernel, the so-called *maximum variance (MV) kernel*, and the box kernel [67]. By applying Theorem 1, we compute their corresponding time step sizes and their cycle times. These results are listed in Table 1. For detailed derivations we refer to the Appendices A.2, A.3, and A.4.

Now we evaluate these three kernels and their diffusion factorisations with respect to one goal: We want to approximate a Gaussian kernel of specified variance. This is possible due to the central limit theorem which guarantees that a Gaussian can be approximated by iterative application of any nonnegative filter kernel whose weights sum up to 1. The efficiency of our

Table 1: Comparison of three filter kernels.

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

approximation is measured by the number of explicit diffusion steps that we need to achieve the specified variance. The quality of the approximation becomes visible by comparing the resulting kernel shape to the shape of the Gaussian.

It is well known that a convolution of a signal $f$ with a Gaussian of variance $\sigma^2$ is equivalent to solving the homogeneous diffusion equation (2.3) with initial data $f$ and stopping time $T = \frac{1}{2}\sigma^2$; see e.g. [31]. In our evaluation we want to appoximate a Gaussian of variance $\sigma^2 = 12h^2$, which corresponds to a diffusion time of $T = 6h^2$. Moreover, we wish to approximate this Gaussian by three iterations of our kernels. Thus, each kernel must implement a diffusion time (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 which 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 particular, gaps between the individual peaks remain. Thus, one cannot expect that the MV kernel offers the desired attenuation of high frequencies that is characteristic for Gaussian convolution.

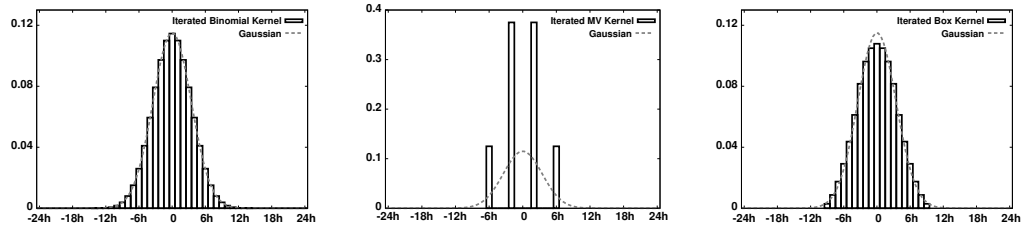


Figure 2: Comparison between the kernels in Fig. 1 after 3 iterations and their approximated Gaussian. **(a) Left:** An iterated binomial kernel approximates the Gaussian very well. **(b) Middle:** An iterated MV kernel yields a poor approximation of the Gaussian. Note that the scale in vertical direction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

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 box filter factorisation as an efficient and fairly accurate way to approximate Gaussian convolution in terms of explicit diffusion steps with varying time step sizes. On the other hand, Gaussian convolution is equivalent to homogeneous diffusion filtering, and the simplest way to perform homogeneous diffusion is to use an explicit scheme with constant time step size. Thus, let us now show that replacing the constant time step size in an explicit scheme by the nonconstant ones from box filter factorisation is a powerful general accelation strategy. First we revisit the homogeneous diffusion case in 1D, before we extend our findings to more general linear or nonlinear diffusion-like operators in any dimension.

## 3.1 FED Scheme for Homogeneous 1D Diffusion

We reconsider the 1-D diffusion equation (2.3) with homogeneous Neumann boundary conditions. Moreover, we perform a space discretisation with grid size $h > 0$ and $N$ grid points $x_j := \left(j - \frac{1}{2}\right) h$ with $j = 1, \ldots, N$. Then 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 $u_j(t) \approx u(x_j, t)$. The symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ approximates the second order spatial derivative operator and takes into account the homogeneous Neumann boundary conditions:

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

If the ODE system (3.1) is discretised in time with 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 in matrix–vector notation:

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

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.

It is instructive to analyse the stability of the explicit scheme (3.3). According to Gershgorin's theorem (see e.g. [60]), the eigenvalues of the matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ from (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 $\tau$ such that all eigenvalues of the iteration matrix $\boldsymbol{I} + \tau \boldsymbol{A}$ are in the interval $[-1, 1]$. This is guaranteed for

$$\tau \;\leq\; \frac{h^2}{2} \;=:\; \tau_{\max}. \tag{3.4}$$

We see that $n$ iterations with the explicit scheme with *fixed* time step size $\tau$ can only lead to a diffusion time of order $\mathcal{O}(n)$. On the other hand, in Section 2.2 we have learned that with the *varying* time step sizes $\tau_i$ from a box filter factorisation, $n$ explicit steps allow to reach a diffusion time of

order $\mathcal{O}(n^2)$. This motivates us to introduce the following cyclic acceleration strategy into our explicit scheme.

We denote $\boldsymbol{u}^{k+1,\,0} := \boldsymbol{u}^k$ and replace the constant time step size in Eq. (3.3) by a cycle of varying time step sizes $\tau_i$:

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; (\boldsymbol{I} \,+\, \tau_i\, \boldsymbol{A})\; \boldsymbol{u}^{k+1,\,i} \qquad (i = 0, \ldots, n-1). \tag{3.5}$$

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

$$\tau_i \;=\; \tau_{\max} \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, ..., n-1) \tag{3.6}$$

with $\tau_{\max} = \frac{h^2}{2}$. Then Table 1 also shows that the vector $\boldsymbol{u}^k = \left(u_j^k\right)_{j=1}^N$ approximates the values $u(x_j,\, k \cdot \theta_n)$, where

$$\theta_n \;=\; \frac{h^2}{6} \cdot \left(n^2 + n\right) \;=\; \tau_{\max} \cdot \frac{n^2+n}{3} \tag{3.7}$$

is the time of one cycle with $n$ time steps. Thus, we may interpret a full FED cycle with time $\theta_n$ as a single *super time step* in the sense of [25]. 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 $\tau_i$ violate the stability condition $\tau \le \frac{h^2}{2}$ for the explicit scheme 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. (3.6), 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 $\tau_{\max}$. This demonstrates the substantial speed-up that can be achieved with unstable time step sizes. However, at the end of a full cycle we can expect to obtain a stable scheme, since this corresponds to a box filter. A formal proof of the stability in the Euclidean norm can be found in Appendix A.5.

So far, the times $\theta_n$ of the FED cycles cover only a discrete set of values. Inspecting the proof in Appendix A.5 shows that the stability still holds if we replace $\tau_{\max}$ in (3.6) by any smaller but fixed time step size $\tau$ for which

the explicit scheme (3.3) is stable in the Euclidean norm. This allows us to adapt the scheme to our needs which we discuss next.

In a practical setting, we know the stability limit $\tau_{\max}$, and we wish to implement a diffusion time $T$ by a specified number $M$ of FED cycles that determines the quality of the approximation. Thus, we have to find an appropriate cycle length $n$ and the corresponding time step sizes $\tau_0,...,\tau_{n-1}$ of our FED scheme. This can be done as follows. First we compute $n$ as the smallest cycle length with $\theta_n \geq \frac{T}{M}$. Using (3.7) this yields

$$n \;=\; \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{12\,T}{M\tau_{\max}}} \right\rceil , \tag{3.8}$$

where the ceiling function $\lceil x \rceil$ denotes the smallest integer $\geq x$. Since one cycle with $n$ steps should implement the cycle time $\frac{T}{M}$, we can obtain $\tau$ from the ansatz

$$\frac{T}{M} \;=\; \tau \cdot \frac{n^2+n}{3}, \tag{3.9}$$

which yields

$$\tau \;=\; \frac{3\,T}{M(n^2+n)}\,. \tag{3.10}$$

This value for $\tau$ determines our cyclic step sizes:

$$\tau_i \;=\; \tau \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \qquad (i = 0, ..., n-1). \tag{3.11}$$

## 3.2 Extension to Arbitrary Diffusion Problems

Our FED scheme has been motivated in the 1-D setting with an explicit scheme for homogeneous diffusion filtering. This was for didactic reasons only: Since box filtering is already highly efficient, there is no practical advantage from factorising it into explicit diffusion steps. However, we have learned how to speed up an explicit scheme by replacing iterations with a constant time step size $\tau$ by cycles with varying time step sizes $\tau_0,..,\tau_{n-1}$. Let us now show that this principle is very general and leads to highly efficient schemes in more challenging situations.

Our goal is to accelerate a general linear explicit scheme

$$\boldsymbol{u}^{k+1} \;=\; (\boldsymbol{I} \,+\, \tau \boldsymbol{P})\,\boldsymbol{u}^k \qquad (k \geq 0), \tag{3.12}$$

Table 2: The first three and last three step sizes of FED for 1D homogeneous diffusion with different cycle lengths $n$. We have used $h = 1$ and $\tau = \tau_{\max} = 0.5$. The last two rows depict the cycle time $\theta_n$ and the speed-up of the FED scheme compared to the explicit scheme with constant time step size $\tau = 0.5$.

| $\boldsymbol{n}$ | **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 |
| **speed-up** | 17.00 | 33.67 | 83.67 | 167.00 | 333.67 |

where we have replaced the specific matrix $\boldsymbol{A}$ from (3.2) by an arbitrary symmetric, negative semidefinite matrix $\boldsymbol{P}$ that results from a space discretisation of a suitable linear parabolic PDE. This PDE can be one- or multidimensional, isotropic or anisotropic, of second or higher order.

The explicit scheme (3.12) is stable in the Euclidean norm if the eigenvalues of the iteration matrix $\boldsymbol{I} + \tau\boldsymbol{P}$ are in the interval $[-1, 1]$. Thus, the time step size $\tau > 0$ has to satisfy the restriction

$$\tau \;\leq\; \frac{2}{\rho(\boldsymbol{P})} \;=:\; \tau_{\max} \tag{3.13}$$

where $\rho(\boldsymbol{P})$ denotes the spectral radius of the symmetric matrix $\boldsymbol{P}$, i.e. the largest modulus of its eigenvalues. It practice one can often replace $\rho(\boldsymbol{P})$ by a simple worst case estimate with Gershgorin's theorem.

Interestingly our results on FED schemes for the 1D homogeneous diffusion scenario can be extended literally to this more general setting. The following theorem descibes that the stability of the explicit scheme (3.12) is inherited to its FED variant.

**Theorem 2 (FED Scheme for Linear Problems).** *Let $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ be symmetric and negative semidefinite, and let the time step size $\tau$ satisfy the*

*stability condition (3.13). Then the FED scheme*

$$\boldsymbol{u}^{k+1,\,0} \;=\; \boldsymbol{u}^{k}, \tag{3.14}$$

$$\boldsymbol{u}^{k+1,\,i+1} \;=\; (\boldsymbol{I} \,+\, \tau_i\, \boldsymbol{P})\; \boldsymbol{u}^{k+1,\,i} \qquad (i = 0, \ldots, n-1), \tag{3.15}$$

$$\boldsymbol{u}^{k+1} \;=\; \boldsymbol{u}^{k+1,\,n} \tag{3.16}$$

*with $k = 0, 1, ...$ and the cyclically varying time step sizes*

$$\tau_i \;=\; \tau \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \tag{3.17}$$

*is stable in the Euclidean norm, i.e.*

$$\|\boldsymbol{u}^{k+1}\|_2 \;\leq\; \|\boldsymbol{u}^{k}\|_2 \qquad (k \geq 0). \tag{3.18}$$

*Moreover, each FED cycle (3.14)–(3.16) with $n$ inner steps corresponds to one super time step of size*

$$\theta_n \;=\; \tau \cdot \frac{n^2 + n}{3}\,. \tag{3.19}$$

The proof is presented in Appendix A.6. One should note that this stability proof requires that $\boldsymbol{P}$ is symmetric and negative definite.

Let us now explain how nonlinear problems can be treated that lead to a symmetric, negative definite 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 function $\rho(\boldsymbol{P}(\boldsymbol{u}(t)))$ in the time step size restriction (3.13), or one can update it *a posteriori* at the beginning of 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

$$\boldsymbol{u}^{k+1,\,i+1} = \left(\boldsymbol{I} \,+\, \tau_i\, \boldsymbol{P}(\boldsymbol{u}^{k})\right)\; \boldsymbol{u}^{k+1,\,i} \qquad (i = 0, ..., n-1). \tag{3.20}$$

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 $\boldsymbol{P}(\boldsymbol{u}^{k})$ to obtain $\boldsymbol{u}^{k+1} := \boldsymbol{u}^{k+1,\,n}$. As we will see later, this strategy is also reasonable to ensure numerical stability.

One may argue that refraining from adapting the nonlinearity in the interior of each cycle compromises the accuracy for large cycle times and prohibits too large cycle times. This observation is not uncommon: Similar tradeoffs can also be experienced for implicit schemes. While very large time steps may be allowed from a stability viewpoint, in practice there will be a tradeoff if one wants to keep also the accuracy high. Thus, for nonlinear PDEs we recommend to use more but smaller FED cycles than for linear PDEs.

## 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 [70] and [55]. Later, the same idea has been used under the name *Super Time Stepping (STS)* [25, 24, 3, 4]. Contrary to our derivation, they have chosen 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 $\left(\frac{1}{2}, 0, \dots, 0, \frac{1}{2}\right)$. Since the MV kernel is very sensitive with respect to high frequencies, it has also been suggested to introduce an additional damping with a parameter $\nu > 0$ that ensures better attenuation properties for high frequencies [4]. This regularisation can be seen as a trade-off between efficiency and damping quality, since larger values for $\nu$ scale down the cycle time. Hence, different damping parameters yield different results. In our FED framework, such an additional damping parameter is not necessary since we restrict ourselves to box filter factorisations. They possess reasonable attenuation properties of high frequencies.

## 3.4 Stability with Respect to Rounding Errors

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.

Let us explain this behaviour by an example where we choose an ordering from small to large time step sizes. In the beginning the stable small time step sizes lead to a rapid decay of high frequent components such that they approach the machine precision. However, this also means that the relative perturbations by rounding effects become large. In the subsequent large time step sizes these rounding errors are amplified substantially such that instabilities can arise.

To address this issue, we advocate two strategies that obtain a better error balancing within each cycle by rearranging the sequence of the FED time step sizes: $\kappa$-cycles and Leja ordering.

### 3.4.1 $\kappa$-Cycles

Gentzsch and Schlüter [25] have proposed to rearrange the original sequence of the explicit time steps $\tau_0, \dots, \tau_{n-1}$ within so-called $\kappa$-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 in Equation (3.17) 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 $\kappa$-cycle with $\kappa = 3$.

In general, a $\kappa$-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

$$\Phi(m) \;=\; \big(m \cdot \kappa\big) \mod p\,, \tag{3.21}$$

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 $n-1$. This gives a feasible rearrangement of the original sequence.

Unfortunately, there is no panacea for the choice of $\kappa$. 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 $\kappa$-cycles in [27].

### 3.4.2 Leja Ordering

Since suitable $\kappa$-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* [50, 14] that has already been successfully applied to iterative solvers [13].

We sketch only the practical application of *Leja ordering* and refer e.g. to [13] 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

$$\prod_{k=0}^{j} |x_{j+1} - x_k| \;=\; \max_{x \in S} \prod_{k=0}^{j} |x - x_k| \qquad (j = 0, \ldots, \ell-1), \tag{3.22}$$

where $|x_0| = \max\limits_{x \in S} |x|$. In some cases it might happen that more than one number fulfils the maximum condition. Then we just take the smallest value to have a unique rearrangement.

The Leja ordering provides a numerically stable order of interpolation points of a polynomial. In our case these points are the roots $\{z_i \,|\, i = 0, ..., n-1\}$ of

the polynomial $p_L(z)$ from Theorem 1. Since $\tau_i = 1/z_i$, we have to apply the Leja algorithm to the inverse time steps $1/\tau_i$ of our FED cycle. Compared to $\kappa$-cycles, the Leja rearrangement offers the advantage that it only depends on the time step sizes $\tau_i$: It does not require to select additional parameters such as the $\kappa$ value. 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 it differs from the $\kappa$-cycle arrangement.

We use the Leja ordering within our experiments, since it gives an even higher numerical robustness than $\kappa$-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 proposed by Lebedev and Finogenov [36]. 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 $\kappa$-cycles and Leja ordering.

## 3.5 General FED Algorithm

At this point, we can give a summary of the general FED algorithm. It is shown in Fig. 3, where we have assumed that we know an *a priori estimate* of the stability bound $\tau_{\max}$ of the underlying explicit scheme. We see 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 are not recommended, because the stability of the intermediate results – and therefore a correct evaluation of the nonlinearities – cannot be guaranteed for rearranged cycles.

Note that in the 1D homogeneous 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 homogeneous 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

1. **Input Data**:
   image $\boldsymbol{f}$, stopping time $T$, number $M$ of FED cycles, stability limit $\tau_{\max}$ for the explicit scheme

2. **Initialisation**:

   (a) Set $\boldsymbol{u}^0 := \boldsymbol{f}$.

   (b) Compute the cycle length $n$ according to (3.8).

   (c) Compute $\tau$ according to (3.10).

   (d) Compute $\tau_0,..,\tau_{n-1}$ according to (3.11).

   (e) Choose a suitable ordering for the step sizes $\tau_i$ (e.g. with $\kappa$-cycles or Leja ordering).

   (f) If the problem is linear, compute the corresponding matrix $\boldsymbol{P}$.

3. **Filtering Loop**:
   For $k = 0, \ldots, M-1$ do:

   (a) If the problem is nonlinear, compute the corresponding matrix $\boldsymbol{P}(\boldsymbol{u}^k)$.

   (b) Perform one linear FED cycle (3.14)–(3.16) or one nonlinear FED cycle (3.14), (3.20), (3.16), with the above ordering of the $n$ explicit time steps $\tau_i$.

Figure 3: General FED algorithm for diffusion filtering.

cycles due to the need for more 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. Two algorithmic strategies are popular: 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 the steady state as quickly as possible, we embed our FED method into a coarse-to-fine strategy [8], 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 [10, 21, 11, 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 dyadically by pixel averaging to a certain coarse level and apply the FED scheme on this image. Afterwards we prolongate the corresponding solution to the next finer level by pixel doubling in each direction, which is the adjoint operator to our restriction operator. Then we apply FED again. 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.

Of coarse, it is also possible to extend this idea to non-dyadic down- and upsampling strategies, which becomes necessary if the image size is not a power of 2. In this case one can simply assume that the greyvalues are constant within each pixel and perform interpolation by the integral mean over the new pixel area.

## 4.2 Fast Jacobi (FJ) Solver

For the direct solution of elliptic problems, we now propose the so-called *Fast Jacobi (FJ)* method. It supplements the simple Jacobi over-relaxation (JOR) algorithm [60] 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. Such systems frequently arise from applying finite difference discretisations or the finite element method to linear elliptic problems. Also for nonlinear elliptic problems, they can appear after a suitable linerisation.

We can solve such a linear system by means of JOR iterations. With $\boldsymbol{D} := \operatorname{diag}(\boldsymbol{B})$, one JOR step with relaxation parameter $\omega > 0$ is given by

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

where the upper index denotes the iteration level. For $\omega = 1$, one obtains the standard Jacobi method.

To understand the stability of the JOR method, we investigate its error vector $\boldsymbol{e}^i := \boldsymbol{x}^i - \boldsymbol{x}$. It is easy to see that it satisfies

$$\boldsymbol{e}^{i+1} \;=\; \Big(\boldsymbol{I} \,-\, \omega\, \boldsymbol{D}^{-1}\boldsymbol{B}\Big)\, \boldsymbol{e}^i \;. \tag{4.3}$$

Since $\boldsymbol{B}$ is positive definite, the Euclidean norm of the error vector converges to zero if

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

where $\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{B})$ denotes the largest eigenvalue of $\boldsymbol{D}^{-1}\boldsymbol{B}$.

The error iteration (4.3) resembles the explicit scheme (3.12), where $\omega$ plays the role of the time step size $\tau$ and $-\boldsymbol{D}^{-1}\boldsymbol{B}$ replaces the negative definite matrix $\boldsymbol{P}$. Moreover, the stability condition (4.4) is the elliptic counterpart to (3.13). This analogy motivates us to introduce a cyclic variant of the JOR algorithm which we call *Fast Jacobi* method: For passing from $\boldsymbol{x}^k$ to $\boldsymbol{x}^{k+1}$, we define $\boldsymbol{x}^{k+1,0} := \boldsymbol{x}^k$ and compute

$$\boldsymbol{x}^{k+1,\,i+1} \;=\; \boldsymbol{x}^{k+1,\,i} \;+\; \omega_i\, \boldsymbol{D}^{-1}\big(\boldsymbol{c} \,-\, \boldsymbol{B}\boldsymbol{x}^{k+1,\,i}\big) \qquad (i = 0, ..., n-1) \tag{4.5}$$

with the cyclically varying relaxation parameters

$$\omega_i \;=\; \omega \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \,. \tag{4.6}$$

Afterwards we set $\boldsymbol{x}^{k+1} := \boldsymbol{x}^{k+1,n}$.

By construction, the Fast Jacobi method inherits the stability properties of the FED scheme. Moreover, in order to avoid the accumulation of rounding errors for large cycles, one should use the same reordering strategies as in Section 3.4.

For nonlinear problems of type

$$\boldsymbol{B}(\boldsymbol{x})\, \boldsymbol{x} \;=\; \boldsymbol{c} \,, \tag{4.7}$$

where $\boldsymbol{B} : \mathbb{R}^N \to \mathbb{R}^{N\times N}$ is a symmetric positive definite matrix-valued function of $\boldsymbol{x}$, we can modify the iterative scheme (4.5) in a similar way as in Section 3.2: We evaluate the nonlinearity in an outer loop and keep it constant within the inner loop. The inner loop consists of a linear Fast Jacobi cycle:

$$\begin{aligned} &\boldsymbol{x}^{k+1,\,i+1} \;=\; \boldsymbol{x}^{k+1,\,i} + \omega_i\, \boldsymbol{D}^{-1}(\boldsymbol{x}^k)\Big(\boldsymbol{c} \,-\, \boldsymbol{B}(\boldsymbol{x}^k)\, \boldsymbol{x}^{k+1,i}\Big) \\ &(i = 0, ..., n-1), \end{aligned} \tag{4.8}$$

where the parameters $\omega_i$ are determined using (4.4) and (4.6). However, since $\boldsymbol{B}$ and $\boldsymbol{D}$ are now nonlinear functions of $\boldsymbol{x}$, also our value for $\omega_{\max}$ may depend on $\boldsymbol{x}$. We can use either an *a priori* estimate for $\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{B}(.))$ or estimate $\mu_{\max}(\boldsymbol{D}^{-1}\boldsymbol{B}(\boldsymbol{x}^k))$ before each inner cycle.

A summary of the whole algorithm is depicted in Fig. 4, where we assume to have an *a priori* estimate for the nonlinearity. Similarly to FED, one observes that multiple outer cycles are beneficial to improve the convergence of the Fast Jacobi method.

In spite of these structural analogies, one should not forget that there is one essential difference between FED and FJ: the multiplication of $-\boldsymbol{B}$ with $\boldsymbol{D}^{-1}$. It can be interpreted as local adjustment 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 vary substantially 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 whose discretisation leads to strongly varying diagonal entries can be solved more efficiently with Fast Jacobi than with the FED approach.

## 4.3 Interpretation as Optimisation Method

For the sake of completeness, we also give an interpretation of the FJ method as an algorithm for a finite dimensional quadratic optimisation problem. Again we consider the linear system $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$ with a symmetric positive definite matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$. Then there exists a symmetric positive definite matrix $\tilde{\boldsymbol{B}} \in \mathbb{R}^{N\times N}$ with $\boldsymbol{B} := \tilde{\boldsymbol{B}}^\top\tilde{\boldsymbol{B}}$. Now we define $\tilde{\boldsymbol{c}} := (\tilde{\boldsymbol{B}}^\top)^{-1}\boldsymbol{c}$, and we consider the strictly convex energy

$$E(\boldsymbol{x}) \;=\; \tfrac{1}{2}\,\|\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{B}}\boldsymbol{x}\|_2^2. \tag{4.9}$$

Minimising $E(\boldsymbol{x})$ with the gradient descent method with step size $\omega$ gives the iterative scheme

$$\begin{aligned}
\boldsymbol{x}^{i+1} &= \boldsymbol{x}^i - \omega\,\boldsymbol{\nabla}E \\
&= \boldsymbol{x}^i - \omega\,\left(-\tilde{\boldsymbol{B}}^\top\tilde{\boldsymbol{c}} + \tilde{\boldsymbol{B}}^\top\tilde{\boldsymbol{B}}\boldsymbol{x}\right) \\
&= \boldsymbol{x}^i + \omega\,(\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}^i),
\end{aligned} \tag{4.10}$$

which is also known as the Richardson method for solving $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$. It is stable for $\omega \le \frac{2}{\rho(\boldsymbol{B})}$.

1. **Input Data**:
   linear system (4.1) or its nonlinear variant (4.7), cycle length $n$, stable parameter $\omega \in (0, \omega_{\max})$ of the JOR scheme, accuracy parameter $\epsilon > 0$.

2. **Initialisation**:

   (a) Set $k := 0$ and choose an initial vector $\boldsymbol{x}^0$.

   (b) Compute $\omega_0$,...,$\omega_{n-1}$ according to Eq. (4.6).

   (c) Perform a suitable ordering of these parameters (e.g. with $\kappa$-cycles or Leja ordering).

3. **Outer Loop**:
   Repeat

   (a) If the problem is nonlinear, compute $\boldsymbol{B}(\boldsymbol{x}^k)$ and its diagonal $\boldsymbol{D}(\boldsymbol{x}^k)$.

   (b) $\boldsymbol{x}^{k+1,\,0} := \boldsymbol{x}^k$.

   (c) Inner Loop:
   Perform one FJ cycle (4.5) or (4.8) with the above ordering of the $n$ parameters $\omega_i$.

   (d) $\boldsymbol{x}^{k+1} := \boldsymbol{x}^{k+1,\,n}$.

   (e) Increment $k$ by 1.

   until $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|_2 < \epsilon$ .

Figure 4: Fast Jacobi method.

Thus, the JOR method (4.2) can be seen as a gradient descent with diagonal preconditioning. This shows that the FJ method is a gradient descent method with diagonal preconditioning and the cyclically varying relaxation parameters (4.6). Alternatively, the diagonal preconditioning can also be interpreted as a redefinition of the norm. Using

$$\|\boldsymbol{y}\|_{\boldsymbol{D}^{-1}} := \|\boldsymbol{D}^{-1/2}\boldsymbol{y}\|_2^2, \tag{4.11}$$

a minimiser of

$$E(\boldsymbol{x}) \;=\; \tfrac{1}{2}\,\|\boldsymbol{c} - \boldsymbol{B}\boldsymbol{x}\|_{\boldsymbol{D}^{-1}} \tag{4.12}$$

must satisfy the preconditioned system

$$\boldsymbol{D}^{-1}\boldsymbol{B}\boldsymbol{x} \;=\; \boldsymbol{D}^{-1}\boldsymbol{c}. \tag{4.13}$$

## 4.4 Connection to the Cyclic Richardson Method

The idea of using varying parameters for simple iterative algorithms has a long tradition. Already Richardson [51] has proposed to allow different relaxation parameters $\omega_i$ in his iterative scheme

$$\boldsymbol{x}^{i+1} \;=\; \boldsymbol{x}^i \;+\; \omega_i \left(\boldsymbol{c} \;-\; \boldsymbol{B}\boldsymbol{x}^i\right), \tag{4.14}$$

but he did not come up with highly efficient parameter settings. While (4.14) resembles the structure of the Fast Jacobi scheme, we have already seen that the Richardson method does not use a diagonal preconditioning.
For a symmetric, positive definite matrix $\boldsymbol{B}$ with smallest eigenvalue $\lambda_{\min}$ and largest eigenvalue $\lambda_{\max}$, Young [68] has proposed to supplement the Richardson iterations with the cyclically varying relaxation parameters

$$\omega_i = \frac{2}{\lambda_{\max}+\lambda_{\min} \;-\; (\lambda_{\max}-\lambda_{\min}) \cdot \cos\left(\pi \cdot \frac{2(n-i)-1}{2n}\right)}$$
$$(i = 0, \ldots, n-1). \tag{4.15}$$

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 (4.14): With the help of $2\cos^2 x = 1-\cos(\pi-2x)$ the resulting relaxation parameter cycles can be simplified to

$$\omega_i \;=\; \frac{2}{\lambda_{\max}} \cdot \frac{1}{2\,\cos^2\left(\pi \cdot \frac{2i+1}{4n}\right)} \qquad (i = 0, \ldots, n-1). \tag{4.16}$$

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 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, the Cyclic Richardson also requires a rearrangement of the parameter sequence to improve the robustness against numerical rounding errors; see e.g. [5]. To this end, one can apply the strategies from Section 3.4 again.

# 5 Applications

Now we show that our proposed methods are well-suited to solve different parabolic or elliptic problems in an efficient way. We assume a uniform 2-D

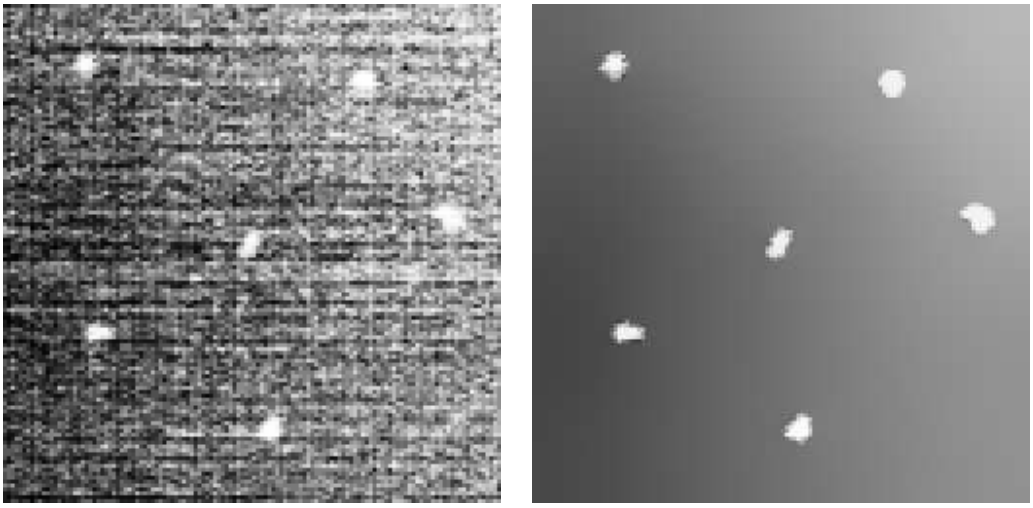

Figure 5: Test setting for nonlinear isotropic diffusion filtering. **(a) Left:** Mammogram ($128 \times 128$ pixels). **(b) Right:** Filtered with the explicit scheme using $\lambda = 7.5$, $\sigma = 1$, $\tau = 10^{-2}$, and $T = 128$.

grid with the mesh sizes $h_1 = h_2 = 1$. 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 between the numerical result $\boldsymbol{u}$ and the reference solution $\boldsymbol{r}$ is the *relative mean absolute error (RMAE)*. It is defined as $\sum_i \frac{|u_i - r_i|}{\|r\|_1}$ with $\|\boldsymbol{r}\|_1 := \sum_i |r_i|$.

## 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. [15]. 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 $u_\sigma$ denotes the function $u$ convolved with a Gaussian of standard deviation $\sigma > 0$. The scalar-valued diffusivity function $g$ is given by the diffusivity [63]

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

For problems of this type, AOS schemes [26, 37, 65] are regarded as efficient solvers. Hence, we compare our FED scheme to the AOS approach. It

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

| **(super) time step size** | **RMAE** | |
|---|---|---|
| | **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 |

is easy to check that the explicit scheme on which FED is based has to satisfy the constant time step size limit $\tau_{\max} = 0.25$. Figure 5 shows our test setting from [63] where we denoise a mammogram to improve the visibility of micro calcifications. For the stopping time $T = 128$ we compute a reference solution by applying the 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 indicates 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 [62]. It is based on a PDE with a symmetric, positive

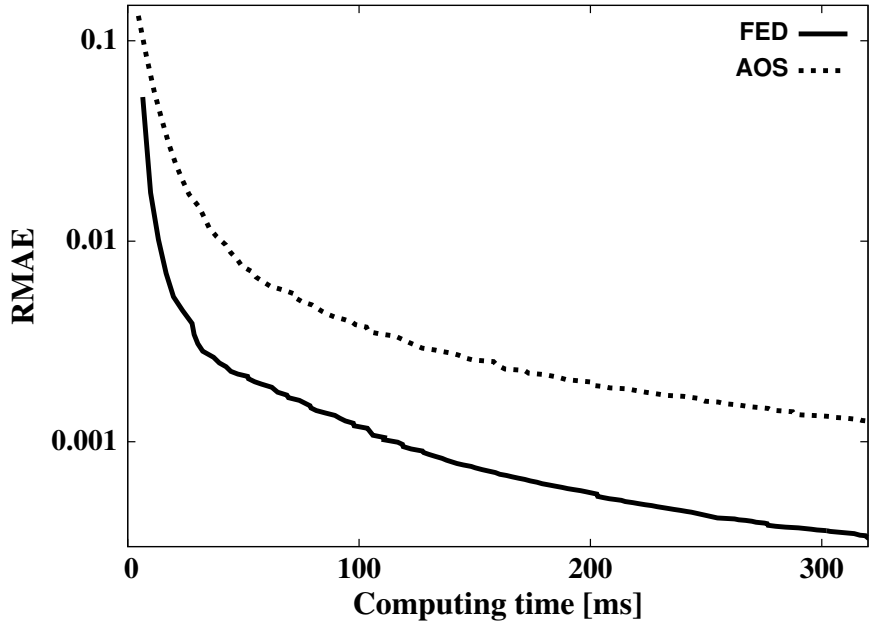


Figure 6: Computing time (milliseconds) vs. RMAE (log-scaled) for the AOS and FED scheme.

definite diffusion tensor $\boldsymbol{D} \in \mathbb{R}^{2\times 2}$:

$$\partial_t u \;=\; \operatorname{div}\left(\boldsymbol{D}\left(G_\rho * \left(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top\right)\right)\,\boldsymbol{\nabla} u\right), \tag{5.3}$$

where $G_\rho$ denotes a 2-D Gaussian with standard deviation $\rho$. This process performs basically one-dimensional diffusion along structures with pronounced local orientation. For more details we refer to [62].

The anisotropy of the diffusion tensor creates mixed derivative terms that cannot 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 [66], 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 [54]. Both solvers are stopped when the residual of the current iteration $\boldsymbol{x}^k$ satisfies

$$\left\|\boldsymbol{B}\boldsymbol{x}^k \,-\, \boldsymbol{c}\right\|_2 \;<\; 10^{-3}\cdot\|\boldsymbol{c}\|_2\;, \tag{5.4}$$

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

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

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 $\omega$ 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 reference 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, 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.

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

| (super) time step size | RMAE | |
|---|---|---|
| | 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 |

## 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. [23]). It keeps a number of selected pixels (given by the so-called inpainting mask), and interpolates the missing data by inpainting with the biharmonic equation

$$\Delta^2 u \;=\; 0. \tag{5.5}$$

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

$$\partial_t u \;=\; -\Delta^2 u \tag{5.6}$$

for $t \to \infty$ with three explicit schemes for parabolic problems: the standard explicit scheme, FED, and Super Time Stepping (STS).

In order to apply these schemes, we need an estimate of the stability limit $\tau_{\max} = \frac{2}{\rho(\boldsymbol{P})}$ of the explicit scheme, where $\boldsymbol{P}$ is a discretisation of the biharmonic operator. By Gershgorin's theorem [60], one easily sees that for a discrete 4-point approximation $\boldsymbol{A}$ of the 2-D Laplacian, one has $\rho(\boldsymbol{A}) = 8$. Thus, $\rho(\boldsymbol{P}) = 8^2 = 64$, and we obtain $\tau_{\max} = \frac{1}{32}$. This very small step size limit makes an explicit scheme with constant time step size fairly inefficient. Thus, it is highly desirable to use FED or STS that allow steps beyond this

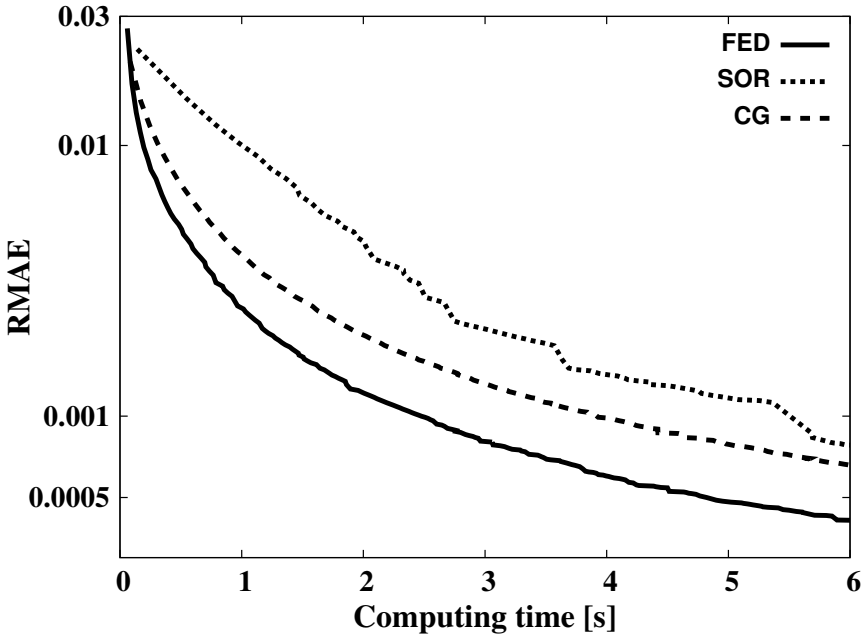


Figure 8: Computing time (seconds) vs. RMAE (log-scaled). The semi-implicit scheme uses SOR or CG solvers.

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.

In our experiments we compare CFED, cascadic Super Time Stepping, and a cascadic standard explicit scheme approach, where the coarse-to-fine strategy uses three levels: $256 \times 256$, $128 \times 128$, and $64 \times 64$ pixels. In Section 3.5 we have mentioned that already three cycles can provide good results for linear problems. In fact, Table 5 shows 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 does 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 also illustrated in Fig. 10. Here we have used stopping times from 50 to 1600 in order to show the dependency between the errors and the computing times. On the other hand, a comparison with the cascadic standard explicit scheme which does not suffer from poor attenuation of

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

Table 5: Comparison of CFED and cascadic Super Time Stepping (CSTS) with 3 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 |

high frequences, illustrates a much higher efficiency of CFED. Obviously the factorisation of the box filter leads to cycle coefficients that are responsible for an efficient damping of high frequencies. This makes them useful components within a cascadic multigrid setting.

Overall, this example illustrates 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 involved constant coefficients, where it turned out that CFED is an appropriate solver for this purpose. In our next exper-

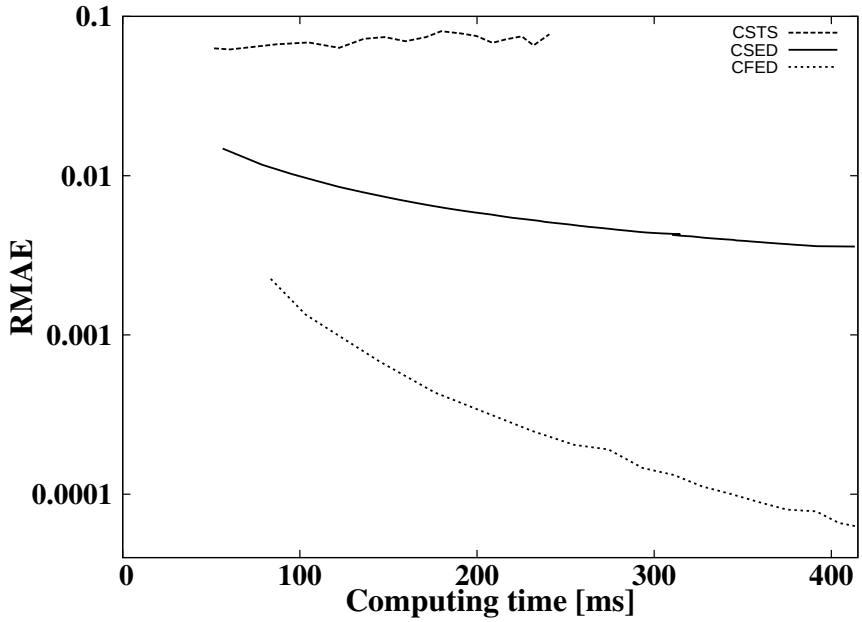


Figure 10: Computing time (milliseconds) vs. RMAE (log-scaled) for cascadic FED (CFED), cascadic super time stepping (CSTS), and a cascadic standard explicit diffusion scheme (CSED).

iment, we consider an elliptic problem with strongly varying coefficients and show that in this case Fast Jacobi is more efficient than a parabolic approach with 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. [18]:

$$E(u) \;=\; \int_{\Omega} \Big( (u-f)^2 \;+\; \alpha \cdot 2\lambda^2 \sqrt{1 + |\boldsymbol{\nabla} u|^2/\lambda^2} \Big)\; d\boldsymbol{x} \tag{5.7}$$

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

$$u - f \;-\; \alpha \operatorname{div}\big( g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u \big) \;=\; 0 \tag{5.8}$$

with the diffusivity function

$$g(s^2) \;:=\; \frac{1}{\sqrt{1 \;+\; s^2/\lambda^2}}\,. \tag{5.9}$$

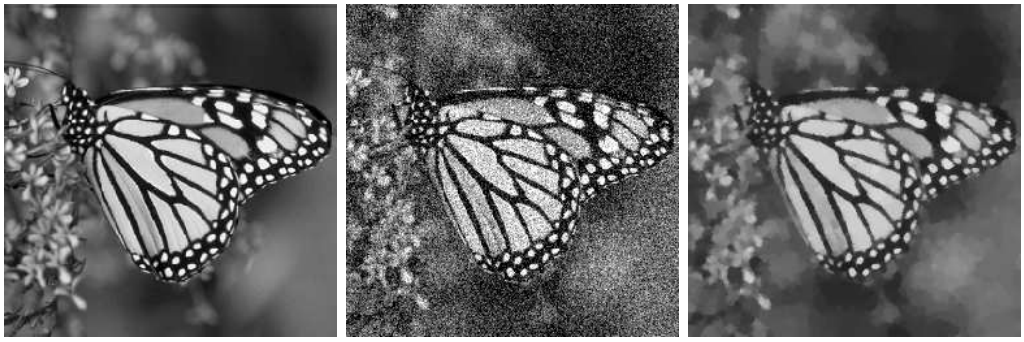

Figure 11: Test setting for Charbonnier regularisation. **(a) Left:** Original image (*monarch*, 256×256 pixels). **(b) Middle:** Noisy image (additive white Gaussian noise, $\sigma = 40$). **(c) Right:** Regularisation of the noisy image with $\lambda = 10^{-2}$ and $\alpha = 2500$.

It is also possible to obtain a solution of the Euler-Lagrange equation (5.8) as the steady state solution of the parabolic gradient descent equation

$$\partial_t u \;=\; \operatorname{div}\left(g(|\boldsymbol{\nabla} u|^2)\,\boldsymbol{\nabla} u\right) \;+\; \frac{f-u}{\alpha}\,. \tag{5.10}$$

**FED Scheme.** An explicit discretisation of Eq. (5.10) with time step size $\tau > 0$ and implicitly stabilised fidelity term yields

$$\frac{\boldsymbol{u}^{k+1}-\boldsymbol{u}^k}{\tau} \;=\; \boldsymbol{A}(\boldsymbol{u}^k)\,\boldsymbol{u}^k \;+\; \frac{\boldsymbol{f}-\boldsymbol{u}^{k+1}}{\alpha}\,. \tag{5.11}$$

Here the symmetric matrix $\boldsymbol{A}(\boldsymbol{u}^k)$ represents the usual central finite difference approximation to the divergence term $\operatorname{div}(g(|\boldsymbol{\nabla} u|^2)\boldsymbol{\nabla} u)$; see e.g. [61] for more details. Scheme (5.11) can be rewritten as

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

It should be noted that this equation involves the expression $(\boldsymbol{I}+\tau\,\boldsymbol{A}(\boldsymbol{u}^k))\,\boldsymbol{u}^k$. This is the solution of an explicit diffusion step without data fidelity term. Since (5.12) only performs a convex combination of this solution and $\boldsymbol{f}$, it has the same stability limit as this explicit diffusion scheme, namely $\tau_{\max} = 0.25$. In an optimisation context, Equation (5.12) can be interpreted as an application of a forward–backward splitting that computes first an explicit gradient descent step with respect to the nonlinear diffusion part followed by a proximal map with respect to the squared data term.

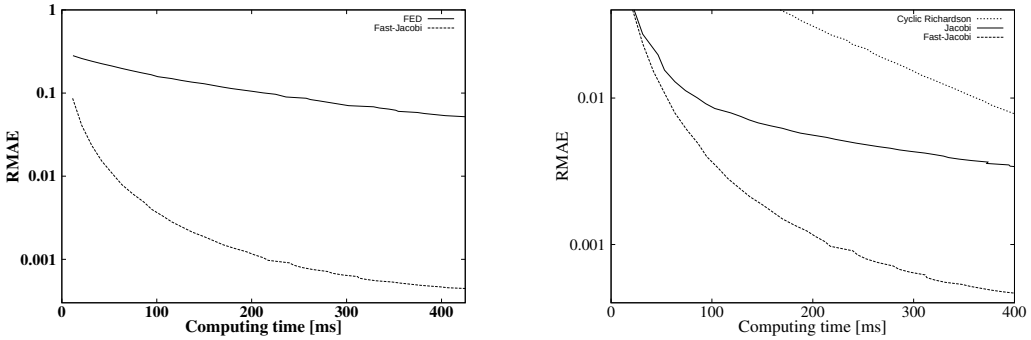


Figure 12: Computing time (milliseconds) vs. RMAE for Charbonnier regularisation. **(a) Left:** Comparison of FED and FJ with cycle length 25. **(b) Right:** Comparison of Jacobi, Fast Jacobi and Cyclic Richardson with cycle length 25.

One way to end up with an FED version of (5.12) is as follows. We replace the explicit step $(\boldsymbol{I} + \tau\, \boldsymbol{A}(\boldsymbol{u}^k))\, \boldsymbol{u}^k$ by a full FED cycle:

$$\boldsymbol{v}^{k+1} := (\boldsymbol{I} + \tau_{n-1}\, \boldsymbol{A}(\boldsymbol{u}^k)) \ldots (\boldsymbol{I} + \tau_0\, \boldsymbol{A}(\boldsymbol{u}^k))\, \boldsymbol{u}^k, \tag{5.13}$$

if necessary also with time step size permutations. Afterwards we obtain $\boldsymbol{u}^{k+1}$ from the convex combination

$$\boldsymbol{u}^{k+1} \;=\; \frac{\alpha \boldsymbol{v}^{k+1} \,+\, \theta_n \boldsymbol{f}}{\alpha + \theta_n}\,, \tag{5.14}$$

where $\theta_n$ denotes the cycle time step size. Since each full FED cycle is stable in the Euclidean norm, it follows that (5.14) with initialisation $\boldsymbol{u}^0 = \boldsymbol{f}$ inherits this stability:

$$\|\boldsymbol{u}^k\|_2 \leq \|\boldsymbol{f}\|_2 \qquad (k = 0, 1, ....). \tag{5.15}$$

**Fast Jacobi Scheme.** Instead of a parabolic evolution, we now want to solve the Euler-Lagrange equation (5.8) by means of the Fast Jacobi method. The discretisation of Eq. (5.8) yields a nonlinear system of equations:

$$\big(\boldsymbol{I} \,-\, \alpha\, \boldsymbol{A}(\boldsymbol{u})\big)\, \boldsymbol{u} \;=\; \boldsymbol{f}\,. \tag{5.16}$$

As is proposed in Section 4.2, we first replace this nonlinear problem by a sequence of linear systems of equations:

$$\underbrace{\big(\boldsymbol{I} \,-\, \alpha\, \boldsymbol{A}(\boldsymbol{u}^k)\big)}_{:=\, \boldsymbol{M}(\boldsymbol{u}^k)}\, \boldsymbol{u}^{k+1} \;=\; \boldsymbol{f} \qquad (k \geq 0). \tag{5.17}$$

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

$$\begin{aligned}\boldsymbol{u}^{k+1,\,i+1} &= \boldsymbol{u}^{k+1,\,i} \,+\, \omega_i\, \boldsymbol{D}^{-1}\Big(\boldsymbol{f} \,-\, \boldsymbol{M}(\boldsymbol{u}^k)\, \boldsymbol{u}^{k+1,\,i}\Big) \\ &= \Big(\boldsymbol{I} \,+\, \omega_i\, \alpha\, \boldsymbol{D}^{-1} \boldsymbol{A}(\boldsymbol{u}^k)\Big)\, \boldsymbol{u}^{k+1,\,i} \,+\, \omega_i\, \boldsymbol{D}^{-1}\big(\boldsymbol{f} \,-\, \boldsymbol{u}^{k+1,\,i}\big)\,. \quad (5.18)\end{aligned}$$

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 [60], 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 the 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) compares 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 higher for Fast Jacobi than for FED. This illustrates 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 counterpart and the Cyclic Richardson method. This allows us to investigate the usefulness of the specific parameter cycles of the Fast Jacobi method. Since $\omega_{\max}$ was estimated by 1, JOR comes down to the Jacobi method. Richardson's cyclic method uses the relaxation parameters from (4.16), and $\lambda_{\max}$ 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 sets it apart from 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 presented evidence that Fast Jacobi performs much better than a parabolic FED approach for elliptic problems with strongly

varying coefficients. It also appears to be 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 have to remain bounded. This excludes e.g. total variation denoising [53] as long as no regularisation is applied that avoids that the diffusivity becomes infinity. To handle variational methods with singularities, specific nonsmooth optimisation algorithms based on primal-dual formulations have been developed; see e.g. [16] or [17]. Interestingly, also in this scenario it is possible to design efficient cyclic alternatives, as was shown by Setzer et al. [58].

## 5.5 Fast Jacobi as an Optimisation Method

We have seen that the Fast Jacobi method can solve linear systems of equations with a symmetric system matrix that arise from discretisations of smooth elliptic PDEs. Thus, in all cases where this PDE can be interpreted as an Euler–Lagrange equation of an energy functional, one could also see the Fast Jacobi method as a numerical method for continuous optimisation problems. Numerical optimisation is a very broad field [9, 42], and in recent years many efforts have been pursued to devise efficient and well parallelisable first order methods; see e.g. [17, 48]. Therefore, let us now compare the Fast Jacobi method with other solvers for optimisation problems.

As a model problem, we follow the suggestion of one reviewer and consider a finite-dimensional variant of Nesterov's worst case function for smooth and strongly convex optimisation problems (see [41], Section 2.1.4):

$$f(\boldsymbol{x}) \;=\; \frac{\kappa-1}{8}\left(x_1^2+\sum_{i=1}^{N}(x_i-x_{i+1})^2-2x_1\right) \;+\; \frac{1}{2}\,\|\boldsymbol{x}\|_2^2, \tag{5.19}$$

with a parameter $\kappa > 1$, and a large integer number $N$. Its gradient is given by

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \frac{\kappa-1}{4}\,(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})+\boldsymbol{x} \tag{5.20}$$

with

$$\boldsymbol{A} = \begin{pmatrix} 2 & -1 & & & & \mathbf{0} \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ \mathbf{0} & & & & -1 & 1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5.21}$$

Setting the gradient to zero yields a linear system of equations. For $N \to \infty$, its analytical solution is known [41]:

$$x_k = \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \qquad (k = 1, 2, ...). \tag{5.22}$$

We approximate this optimisation problem numerically with the standard Jacobi, the Fast Jacobi, and the conjugate gradient method. Furthermore, we also consider a recent method for strongly convex optimisation named iPiasco [43]. This approach essentially consists of performing a usual gradient step with an additional inertial term, which is then evaluated using a proximal map.

Figure 13 displays the $\ell_2$ error for the different methods as a function of the CPU time. We have chosen the parameters $\kappa = 10$ and $N = 10^5$ and selected a cycle length of 4 for our Fast Jacobi approach. We observe that both iPiasco and Fast Jacobi outperform the conjugate gradient and standard Jacobi solvers. Evaluations on different hardware indicate that the runtimes of iPiasco and Fast Jacobi are comparable: While iPiasco is slighlty ahead on the architecture used for benchmarking in our experiments (3.2 GHz Intel Xeon Processor), tests on other hardware (2.6 GHz Intel Core i5) have shown that this can also be the other way around.

This experiment indicates that the Fast Jacobi method can be an alternative to modern first order optimisation algorithms such as iPiasco, for which optimality results in terms of linear convergence rates are known. In our future work we plan to investigate if also corresponding theoretical results can be established for Fast Jacobi techniques, using e.g. the recent framework of Drori and Teboulle [20].

## 5.6 Higher Dimensional Problems and GPU Implementations

By means of 2-D optic flow computations, it has already been demonstrated that FED is well-suited for parallelisation on GPUs [29]. In this subsection, we illustrate that this also holds for Fast Jacobi and in three dimensions.

As an example application we have chosen an anisotropic range image integration problem, which aims at acquiring a single 3-D model from multiple range images [57]. It requires to solve an elliptic PDE that can be written as

$$p(u)\, u - \alpha \operatorname{div} \left( \boldsymbol{D} \left( G_\rho * \left( \boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top \right) \right) \boldsymbol{\nabla} u \right) \;=\; q(u), \tag{5.23}$$

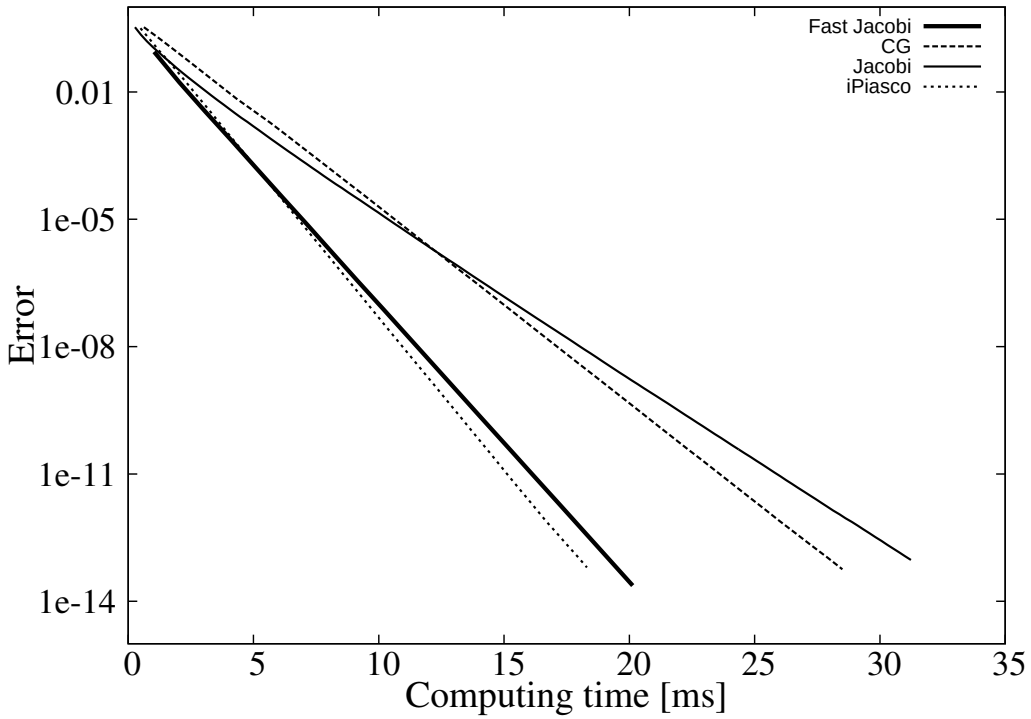


Figure 13: Comparison of the convergence rates on Nesterov's worst case function ((5.19)). The runtimes refer to a 3.2 GHz Intel Xeon Processor.

where $u : \mathbb{R}^3 \to \mathbb{R}$ is the unknown solution. Here $p(u)$ and $q(u)$ are suitably chosen, real-valued functions. The diffusion tensor $\boldsymbol{D}$, the Gaussian $G_\rho$, and the nabla operator $\boldsymbol{\nabla}$ are defined in a three-dimensional setting, and the parameter $\alpha$ denotes a positive smoothness weight. More details can be found in [57].

After a finite difference discretisation, our model becomes a nonlinear system with $N$ equations:

$$(\boldsymbol{P}(\boldsymbol{u}) - \alpha\, \boldsymbol{A}(\boldsymbol{u}))\, \boldsymbol{u} \;=\; \boldsymbol{q}(\boldsymbol{u})\;, \tag{5.24}$$

where $N$ is the number of voxels. The $N$-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{q}(\boldsymbol{u})$ are obtained by a spatial discretisation of the functions $u$ and $q(u)$, respectively. The matrix $\boldsymbol{A}(\boldsymbol{u}) \in \mathbb{R}^{N\times N}$ is the 3-D discrete divergence operator with a diffusion tensor $\boldsymbol{D}\left(G_\rho * \left(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top\right)\right)$. Moreover, $\boldsymbol{P}(\boldsymbol{u}) := \operatorname{diag}(\boldsymbol{p}(\boldsymbol{u})) \in \mathbb{R}^{N\times N}$ 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. (5.16).

In our experiment, we compare the running times of a sequential CPU and a parallel GPU implementation of the Fast Jacobi applied to Eq. (5.24). This comparison is shown in Table 6, where we use the 3.2 GHz Intel Xeon

Figure 14: 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).

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 |

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. 14, 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 within 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 the 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

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

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.

# 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 six prototypical scenarios, we have demonstrated that these cyclic schemes are widely applicable to all kinds of smooth 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. [28] or [44] for some related results. Since this requires a more complicated stability analysis, this is left for future research.

Although cyclic algorithms have been around in the numerical analysis community for a long time, their usefulness in image analysis has first been established in the conference version [27] of the present paper. However, this transfer of knowledge from numerical analysis to image analysis 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 previously 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 the seminal work of Richardson [51], 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 the FED and FJ functionalities for

a broad range of processes. It is easy to embed into C and C++ programs. 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 the project We 2602/7-1 as well as the Gottfried Wilhelm Leibniz Prize We 2602/9-1.

# References

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (9th printing)*, Dover, New York, 1972, ch. "Orthogonal Polynomials" (Ch. 22), pp. 771–802.

[2] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, *Fast explicit diffusion for accelerated features in nonlinear scale spaces*, in Proc. 2013 British Machine Vision Conference, T. Burghardt, D. Damen, W. Mayol-Cuevas, and M. Mirmehdi, eds., Bristol, England, Sept. 2013, BMVA Press, pp. 13.1–13.11.

[3] V. Alexiades, *Overcoming the stability restriction of explicit schemes via super-time-stepping*, in Proceedings of Dynamic Systems and Applications, vol. 2, Atlanta, Georgia, May 1995, pp. 39–44.

[4] V. Alexiades, G. Amiez, and P.-A. Gremaud, *Super-time-stepping acceleration of explicit schemes for parabolic problems*, Communications in Numerical Methods in Engineering, 12 (1996), pp. 31–42.

[5] R. S. Anderssen and G. H. Golub, *Richardson's non-stationary matrix iterative procedure*, Tech. Rep. STAN-CS-72-304, Computer Science Department, Stanford University, August 1972.

[6] E. Bänsch and K. Mikula, *A coarsening finite element strategy in image selective smoothing*, Computation and Visualization in Science, 1 (1997), pp. 53–61.

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

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

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[10] H. Brakhage, *Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode*, Numerische Mathematik, 2 (1960), pp. 183–196.

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

[12] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr, *A multigrid platform for real-time motion computation with discontinuity-preserving variational methods*, International Journal of Computer Vision, 70 (2006), pp. 257–277.

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

[14] ——, *On the evaluation of polynomial coefficients*, Numerical Algorithms, 33 (2003), pp. 153–161.

[15] 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 (1992), pp. 1895–1909.

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

[17] 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 (2011), pp. 120–145.

[18] 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, vol. 2, Austin, TX, Nov. 1994, IEEE Computer Society Press, pp. 168–172.

[19] J. Crank and P. Nicolson, *A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type*, Proceedings of the Cambridge Philosophical Society, 43 (1947), pp. 50–67.

[20] Y. Drori and M. Teboulle, *Performance of first-order methods for smooth convex minimization: a novel approach*, Mathematical Programming, 145 (2014), pp. 454–482.

[21] R. P. Fedorenko, *A relaxation method for solving elliptic difference equations*, USSR Computational Mathematics and Mathematical Physics, 1 (1962), pp. 1092–1096.

[22] S. Frankel, *Convergence rates of iterative treatments of partial differential equations*, Mathematical Tables and Other Aids to Computation, 4 (1950), pp. 65–75.

[23] 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 (2008), pp. 255–269.

[24] W. Gentzsch, *Numerical solution of linear and non-linear parabolic differential equations by a time discretisation of third order accuracy*, in Proceedings of the Third GAMM-Conference on Numerical Methods in Fluid Mechanics, E. H. Hirschel, ed., Friedr. Vieweg & Sohn, 1979, pp. 109–117.

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

[26] D. G. Gordeziani and G. V. Meladze, *Simulation of the third boundary value problem for multidimensional parabolic equations in an arbitrary domain by one-dimensional equations*, USSR Computational Mathematics and Mathematical Physics, 14 (1974), pp. 249–253.

[27] S. Grewenig, J. Weickert, and A. Bruhn, *From box filtering to fast explicit diffusion*, in Pattern Recognition, M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, eds., vol. 6376 of Lecture Notes in Computer Science, Berlin, 2010, Springer, pp. 543–552.

[28] 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, vol. 1281, Rhodes (Greece), 2010, pp. 761–764.

[29] 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 Trends and Topics in Computer Vision, K. N. Kutulakos, ed., vol. 6554 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 372–383.

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

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

[32] S. Hoffmann, M. Mainberger, J. Weickert, and M. Puhl, *Compression of depth maps with segment-based homogeneous diffusion*, in Scale Space and Variational Methods in Computer Vision, A. Kuijper, K. Bredies, T. Pock, and H. Bischof, eds., vol. 7893 of Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 319–330.

[33] B. Jawerth, P. Lin, and E. Sinzinger, *Lattice Boltzmann models for anisotropic diffusion of images*, Journal of Mathematical Imaging and Vision, 11 (1999), pp. 231–237.

[34] Z. Krivá and K. Mikula, *An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing*, Journal of Visual Communication and Image Representation, 13 (2002), pp. 22–35.

[35] P. Laasonen, *Über eine Methode zur Lösung der Wärmeleitungsgleichung*, Acta Mathematica, 81 (1949), pp. 309–317.

[36] V. I. Lebedev and V. N. Finogenov, *Ordering the iteration parameters in the cyclic Chebychev iterative method*, USSR Computational Mathematics and Mathematical Physics, 11 (1971), pp. 155–170.

[37] T. Lu, P. Neittaanmäki, and X.-C. Tai, *A parallel splitting up method and its application to Navier-Stokes equations*, Applied Mathematics Letters, 4 (1991), pp. 25–29.

[38] A. Luxenburger, H. Zimmer, P. Gwosdek, and J. Weickert, *Fast PDE-based image analysis in your pocket*, in Scale Space

and Variational Methods in Computer Vision, A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, eds., vol. 6667 of Lecture Notes in Computer Science, Springer, Berlin, Germany, 2011, pp. 544–555.

[39] A. Mang, T. A. Schuetz, S. Becker, A. Toma, and T. M. Buzug, *Cyclic numerical time integration in variational non-rigid image registration based on quadratic regularisation*, in Proc. Vision, Modeling, and Visualization 2012, Magdeburg, Germany, Nov. 2012, Eurographics Digital Library, pp. 143–150.

[40] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[41] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of Applied Optimization, Kluwer, Boston, 2004.

[42] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 2006.

[43] P. Ochs, T. Brox, and T. Pock, *iPiasco: Inertial proximal algorithm for strongly convex optimization*, Journal of Mathematical Imaging and Vision, (2015). Online First.

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

[45] D. W. Peaceman and H. H. Rachford Jr., *The numerical solution of parabolic and elliptic differential equations*, Journal of the Society for Industrial and Applied Mathematics, 3 (1955), pp. 28–41.

[46] P. Perona and J. Malik, *Scale space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 629–639.

[47] P. Peter, *Three-dimensional data compression with anisotropic diffusion*, in Pattern Recognition, J. Weickert, M. Hein, and B. Schiele, eds., vol. 8142 of Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 231–236.

[48] T. Pock and A. Chambolle, *Diagonal preconditioning for first order primal-dual algorithms in convex optimization*, in Proc. 13th International Conference on Computer Vision, Barcelona, Spain, Nov. 2011, pp. 1762–1769.

[49] L. L. Rakêt and B. Markussen, *Approximate inference for spatial functional data on massively parallel processors*, Computational Statistics and Data Analysis, 72 (2014), pp. 1723–1730.

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

[51] 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 (1910), pp. 307–357.

[52] G. Rosman, L. Dascal, A. Sidi, and R. Kimmel, *Efficient Beltrami image filtering via vector extrapolation methods*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 858–878.

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

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

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

[56] A. Schmidt-Richberg, J. Ehrhardt, R. Werner, and H. Handels, *Fast Explicit Diffusion for registration with direction-dependent regularization*, in Biomedial Image Registration, B. M. Dawant, G. E. Christensen, J. M. Fitzpatrick, and D. Rueckert, eds., vol. 7359 of Lecture Notes in Computer Science, Berlin Heidelberg, 2012, Springer, pp. 220–228.

[57] C. Schroers, H. Zimmer, L. Valgaerts, A. Bruhn, O. Demetz, and J. Weickert, *Anisotropic range image integration*, in Pattern Recognition, A. Prinz, T. Pock, H. Bischof, and F. Leberl, eds., vol. 7476 of Lecture Notes in Computer Science, Berlin, 2012, Springer, pp. 73–82.

[58] S. Setzer, G. Steidl, and J. Morgenthaler, *On cyclic gradient descent reprojection*, Computational Optimization and Applications, 54 (2013), pp. 417–440.

[59] A. Spira, R. Kimmel, and N. Sochen, *A short-time Beltrami kernel for smoothing images and manifolds*, IEEE Transactions on Image Processing, 16 (2007), pp. 1628–1636.

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

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

[62] ——, *Coherence-enhancing diffusion filtering*, International Journal of Computer Vision, 31 (1999), pp. 111–127.

[63] ——, *Applications of nonlinear diffusion in image processing and computer vision*, Acta Mathematica Universitatis Comenianae, 70 (2001), pp. 33–50.

[64] J. Weickert, K. Hagenburg, M. Breuss, and O. Vogel, *Linear osmosis models for visual computing*, in Energy Minimisation Methods in Computer Vision and Pattern Recognition, A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X.-C. Tai, eds., vol. 8081 of Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 26–39.

[65] 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 (1998), pp. 398–410.

[66] M. Welk, G. Steidl, and J. Weickert, *Locally analytic schemes: A link between diffusion filtering and wavelet shrinkage*, Applied and Computational Harmonic Analysis, 24 (2008), pp. 195–224.

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

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

[69] D. M. Young, *Iterative Methods for Solving Partial Difference Equations of Elliptic Type*, PhD thesis, Dept. of Mathematics, Harvard University, Cambridge, MA, May 1950.

[70] 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).

# Appendix

## A.1 Proof of Theorem 1 (Diffusion Factorisation of Symmetric Filters)

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 $m+1$:

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

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

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

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

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

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

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

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

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 $\alpha_m^{(n)}$:

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

for all $k \in \{0, ..., n\}$. The corresponding matrix-vector notation of Eq. (A.9) is given by

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

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 $\boldsymbol{\alpha}^{(\boldsymbol{n})} = \boldsymbol{B}^{-1}\boldsymbol{w}$. 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

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

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

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

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

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

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 $b_{0,0}^{-1} = 1$ and $b_{0,p}^{-1} = 2$ for $p \neq 0$. Hence,

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

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:

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

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

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

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

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

This implies that $p_L(0) = \alpha_0^{(n)} = 1$, which shows that $z = 0$ cannot appear as a root of the polynomial $p_L$. Morover, $p_L(z)$ can be rewritten as

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

Replacing $-z$ by the operator $\Delta_h$ 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 $L^h_{2n+1}$ can be decomposed into $n$ explicit homogeneous 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

$$\begin{aligned} \alpha_1^{(n)} \;&=\; h^2 \cdot \sum_{k=1}^{n} \left( \binom{k+1}{2} + \binom{k}{2} \right) w_k \\ &=\; h^2 \cdot \sum_{k=1}^{n} k^2 \cdot w_k, \end{aligned} \tag{A.28}$$

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

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

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

$$K^h_{2n_1+1} \,*\, K^h_{2n_2+1} \;=\; K^h_{2(n_1+n_2)+1}\,. \tag{A.30}$$

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

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

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

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

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 $\alpha^{(n)}_m$ of $p_M$ can be computed by means of Eq. (2.6):

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

Thus, the polynomial of $M^h_{2n+1}$ 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

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

Acoording to [1] they and have the following closed form:

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

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. They can be computed as

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

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 $\alpha_0^{(n)} = 1$. Moreover, 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

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

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 also makes 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 homogeneous diffusion steps with the time step sizes

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

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 (3.5)–(3.6)

The result $\boldsymbol{u}^{k+1} \in \mathbb{R}^N$ with $k \geq 0$ after a complete cycle of the linear FED scheme (3.5) can be written as the matrix-vector product

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

Let us now analyse the eigenvalues $\lambda_1, \ldots, \lambda_N$ of the matrix $\boldsymbol{Q_A} \in \mathbb{R}^{N \times N}$. To this end, we consider the orthonormal eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ of the symmetric matrix $\boldsymbol{A}$ with the corresponding real-valued eigenvalues

$$\mu_1(\boldsymbol{A}), \ldots, \mu_N(\boldsymbol{A}) \in \left[ -\tfrac{4}{h^2} \,,\, 0 \right] .$$

We have

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

This means that the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ are also eigenvectors of $\boldsymbol{Q_A}$ 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 $\lambda_j \in [-1, 1]$. Thus, the FED scheme (3.5)–(3.6) is stable in the Euclidean norm. This stability result remains also valid if we replace all $\tau_i$ by $q\tau_i$ with some fixed $q \in [0, 1)$, which is equivalent to replacing $\tau_{\max}$ in (3.6) by a constant $\tau \in [0, \tau_{\max})$.

## A.6 Proof of Theorem 2 (FED Scheme for Linear Problems)

If we replace the matrix $\boldsymbol{A}$ in Appendix A.5 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 $[-\rho(\boldsymbol{P}),\, 0]$, an analogue computation for the eigenvalues $\lambda'_1, \ldots, \lambda'_N$ of $\boldsymbol{Q_P}$ 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 $\tau_i$ of the linear FED scheme. Since it could happen that $\rho(\boldsymbol{P}) > \frac{4}{h^2}$, we cannot guarantee that the polynomial fulfils $|p_B(-\mu_j(\boldsymbol{B}))| \leq 1$ for all $j$. However, if we replace these time steps $\tau_i$ by $\tau'_i := \frac{2}{\rho(\boldsymbol{P})} \cdot \frac{2}{h^2} \cdot \tau_i$, then we obtain the eigenvalues

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

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

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

and thus $\lambda'_j \in [-1, 1]$ for all $j$. This shows stability in the Euclidean norm for the cyclic scheme with the matrix $\boldsymbol{P}$ and the time step sizes from Eq. (3.17) with $\tau = \tau_{\max} := \frac{2}{\rho(\boldsymbol{P})}$. Clearly, this stability is also preserved if the constant $\tau$ is chosen from the interval $(0, \tau_{\max})$.
To prove that the cycle time $\theta_n$ fulfils (3.19), note that (A.47) and (A.48) imply that

$$\sum_{i=0}^{n-1} \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \;=\; \frac{n^2+n}{3}\,. \tag{A.56}$$

Thus, we obtain

$$\theta_n \;=\; \sum_{i=0}^{n-1} \tau_i \;=\; \tau \sum_{i=0}^{n-1} \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)} \;=\; \tau \cdot \frac{n^2+n}{3}\,. \tag{A.57}$$