Università degli Studi dell'Insubria

DIPARTIMENTO DI SCIENZE E ALTA TECNOLOGIA
Corso di Dottorato in Informatica e Matematica del Calcolo

Ph.D. Thesis

Preconditioned fast solvers for large linear systems with specific sparse and/or Toeplitz-like structures and applications

Candidato:
Fabio Durastante

Relatori:
Prof. Daniele Bertaccini
Prof. Stefano Serra Capizzano

XXX Ciclo

“Lo scopo della Matematica è di determinare il valore numerico delle incognite che si presentano nei problemi pratici. Newton, Euler, Lagrange, Cauchy, Gauss, e tutti i grandi matematici sviluppano le loro mirabili teorie fino al calcolo delle cifre decimali necessarie”

Giuseppe Peano

“Non esistono problemi dai quali si può prescindere. Non c’è niente di più penoso di coloro i quali suddividono il pensiero dell’uomo in un pensiero da cui non si può prescindere e in uno da cui si può prescindere. Fra costoro si celano i nostri futuri carnefici.”

Una partita a scacchi con Albert Einstein,
Friedrich Dürrenmatt

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

"Ogni problema della matematica applicata e computazionale si riconduce alla fine a risolvere un sistema di equazioni *lineari*"

P. Zellini, La matematica degli dei e gli algoritmi degli uomini

The innermost computational kernel of many large–scale scientific applications and industrial numerical simulations, in particular where systems of partial differential equations or optimization problems are involved in the models, is often a sequence of large linear systems that can be either *sparse* or can show some kind of *structure*. This thesis deals with specific methods for addressing their solution, while keeping in mind and exploiting the originating setting.

Computing the solution of these linear systems typically consumes a significant portion of the overall computational time required by the simulation hence we focus our attention on the *efficiency* of the proposed techniques.

Recent advances in technology have led to a dramatic growth in the size of the matrices to be handled, and iterative techniques are often used in such circumstances, especially when *decompositional* and *direct* approaches require prohibitive storage requirements. Often, an iterative solver with a suitable preconditioner is more appropriate. Nevertheless, a preconditioner that is good for every type of linear system, i.e., for every problem, does not exist.

Let us consider the linear system

$$A\,\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where A is a real $n \times n$ matrix, $\mathbf{b}$ is the known vector and $\mathbf{x}$ is the vector of the unknowns. In the following, we concentrate on iterative solvers as methods for computing an approximate solution of the algebraic linear system (1.1). Recall that an iterative solver is a strategy that generates a

sequence of candidate approximations $\mathbf{x}^{(k)}$, $k = 0, 1, \ldots$, for the solution starting from a given initial guess $\mathbf{x}^{(0)}$. The iterative methods we consider involve the matrix A *only* in the context of matrix–vector multiplications. Thus, the *structure* and/or the *sparsity* of the underlying matrices is used to design an efficient implementation.

Lack of robustness and sometimes erratic convergence behavior are recognized potential weakness of iterative solvers. These issues hamper the acceptance of iterative methods despite their intrinsic appeal for large linear systems. Both the efficiency and robustness of iterative techniques can be substantially improved by using *preconditioning*. Preconditioning transforms the original linear system into a mathematically equivalent one, i.e., having the same solution and, under appropriate conditions, an iterative solver can converge faster. In this context it is appropriate to quote Saad in [244],

> "In general, the reliability of iterative techniques, when dealing with various applications, depends much more on the quality of the preconditioner than on the particular Krylov subspace accelerators used."

The main original contribution of this thesis can be summarized as follows: new *preconditioning techniques*, the related spectral analysis of the preconditioned matrices, the analysis of the computational cost, and few numerical experiments, confirming that the proposed techniques are effective and competitive with respect to the existing ones.

Consider the linear algebraic system

$$A\mathbf{x} = \mathbf{b}, \qquad \mathbf{x}, \mathbf{b} \in \mathbb{R}^n,\ A \in \mathbb{R}^{n\times n}. \tag{1.2}$$

Often we look for a mathematically (but not computationally!) equivalent linear system with more favorable spectral properties for its matrix $\tilde{A}$

$$\tilde{A}\mathbf{x} = \tilde{\mathbf{b}}, \qquad \mathbf{x}, \tilde{\mathbf{b}} \in \mathbb{R}^n,\ \tilde{A} \in \mathbb{R}^{n\times n}, \tag{1.3}$$

in order to speed up the convergence of the iterations of a given iterative solver. The linear system (1.3) is the preconditioned version of (1.2) and the matrix (or the matrices) that transform (1.2) in (1.3), usually implicitly, is (are) called *preconditioner* (*preconditioners*).

This thesis opens with Chapter 2 that represents an introduction to both classes of iterative solvers considered here: Krylov subspace methods and multigrid methods (MGM). Then, by taking into account the structures introduced in Section 2.2, we focus on some of the standard preconditioners from which our new contributions spring forth, see Sections 2.3 and 2.4.

In Part I we introduce our proposal for *sparse approximate inverse* preconditioners for either the solution of time–dependent Partial Differential Equations (PDEs), Chapter 3, and Fractional Differential Equations, containing both classical and fractional terms, Chapter 5. More precisely, we propose a new technique for updating preconditioners for dealing with sequences (see Section 1.1) of linear systems for PDEs and FDEs, that can be used also to compute matrix functions of large matrices via quadrature formula in Chapter 4 and for optimal control of FDEs in Chapter 6. At last, in Part II, we consider structured preconditioners for quasi–Toeplitz systems. The focus is towards the numerical treatment of discretized convection–diffusion equations in Chapter 7 and on the solution of FDEs with linear multistep formula in boundary value form in Chapter 8.

Chapter 3 contains excerpts from Bertaccini and Durastante [33] and Durastante [106]. My main contribution in this works lies in the extension with higher order interpolation formulas of the original techniques from Benzi and Bertaccini [20] and Bertaccini [31]. I have worked both on the theoretical statements and on the writing of the codes.

Chapter 4 contains excerpts from Bertaccini, Popolizio, and Durastante [41]. For this paper my main contribution has been the coding and the development of the strategy for selecting the reference preconditioner(s), in order to build the sequence of updated preconditioners.

Chapter 5 contains excerpts from Bertaccini and Durastante [35]. My main contribution in this work has been the idea of extending the use of approximate inverse preconditioners to treat sequences of matrices with structural decay of the entries, coming from the discretization of Fractional Differential Equations on Cartesian meshes. Operatively, I have worked on both the building of the theory and the writing of codes.

Chapter 6 contains excerpts from Cipolla and Durastante [81], the theoretical analysis needed to construct the sequence of discrete problems is the product of both authors and has been done in an equally distributed way. My main contribution concerns the application to this case of both the techniques and the preconditioners developed in Chapters 3 and 5.

Chapter 7 contains excerpts from Bertaccini, Donatelli, Durastante, and Serra-Capizzano [32]. For this work my main contribution was the construction of the optimized Runge–Kutta smoother, based

on the spectral analysis made with the tools in [126]. I have also worked on the implementation of the prototype of a multigrid V–cycle preconditioner.

Chapter 8 contains excerpts from Bertaccini and Durastante [34]. My main contribution in this work regards the theoretical framework, namely the application of the GLT Theory to the classical preconditioners in Bertaccini [27–30] and Bertaccini and Ng [38], and to the extension of the considered techniques for treating the numerical solution of discretized space–fractional differential equations.

1.1 *Sparsity, Structure and Sequence of Matrices*

We focus our attention on iterative methods based on the matrix–vector product kernel. One of our main goals is exploiting cases in which this operation can be performed efficiently and at a cost that is below $O(n^2)$ for an $n \times n$ matrix. Both *sparsity* and *structure* or *sparsity*, whenever present, can help for this; see Section 2.2.

Several preconditioning strategies have been developed during the years, see, e.g., the survey [19] and the book [211]. In the following Parts I and II, both the approaches have been used and, whenever possible, ideas and instruments coming from one setting have been applied to the other.

Another important issue is that whenever we face a problem that comes from the discretization of, e.g., a differential model or from the solution of an optimization problem, we do not have just one linear system, but we should think about a *sequence* of linear systems (and then sequence of matrices) parametrized by the underlying discretization step(s). Some important properties for our computational tasks can be interpreted only in term of *matrix sequences*. All over the thesis we will distinguish mainly between two kinds of sequences:

1. $\{A_n\}_{n\geq 1}$ with $A_n \in \mathbb{R}^{d_n\times d_n}$ or $A_n \in \mathbb{C}^{d_n\times d_n}$, that are sequences of matrices of growing dimension $d_n \overset{n\to\infty}{\longrightarrow} \infty$,
2. $\{A^{(k)}\}_k$ with $A^{(k)} \in \mathbb{R}^{d_n\times d_n}$ or $A^{(k)} \in \mathbb{C}^{d_n\times d_n}$ with a fixed value of d_n.

Generally speaking, we focus on the first kind of sequences, i.e., sequences of matrices of growing dimension, for dealing with *spectral properties* and discussing features that the discrete problem inherits from the continuous, through classical asymptotic arguments; see, e.g., Section 2.2.1 and Part II.

1.2 *Applications and Computational Framework*

In this thesis, the design of the preconditioners we propose starts from applications instead of treating the problem in a completely general way. The reason is that not all types of linear systems can be addressed with the same tools. In this sense, the techniques for designing efficient iterative solvers depends mostly on properties inherited from the continuous problem, that has originated the discretized sequence of matrices. Classical examples are *locality*, *isotropy* in the PDE context, whose discrete counterparts are *sparsity* and matrices *constant along the diagonals*, respectively; see Section 2.2. Therefore, it is often important to take into account the properties of the originating continuous model for obtaining better performances and for providing an accurate convergence analysis. In Parts I and II, we consider linear systems that arise in the solution of both linear and nonlinear partial differential equation of both *integer* and *fractional* type. For the latter case, an introduction to both the theory and the numerical treatment is given in Appendix A.

The approximation of functions of large matrices is treated in Chapter 4. Functions of matrices are ubiquitous in the solution of ordinary, partial and fractional differential equations, systems of coupled differential equations, hybrid differential–algebraic problems, equilibrium problems, measures of complex networks, and in many more contexts; see again Chapter 4.

All the algorithms and the strategies presented in this thesis are developed having in mind their parallel implementation. In particular, we consider the processor–co–processor framework, in which the main part of the computation is performed on a *Graphics Processing Unit* (GPU) accelerator. We recall that GPU–accelerated computing works by offloading the compute–intensive portions of our numerical linear algebra codes to the GPU, while the remainder of the code still runs on the CPU. Particularly, it is possible to implement efficiently matrix–vector multiplications, and these are the numerical kernel of both the iterative methods the preconditioners considered in Chapter 2.

2

Some Iterative Methods and Preconditioners

> "Reason cannot permit our knowledge to remain in an unconnected and rhapsodistic state, but requires that the sum of our cognitions should constitute a system. It is thus alone that they can advance the ends of reason."
>
> I. Kant, The Critique of Pure Reason.

In the thesis, we concentrate on the solution of linear system $A\,\mathbf{x} = \mathbf{b}$, where A is (if not otherwise stated) a real nonsingular square matrix, $\mathbf{b}$ is the known vector, and $\mathbf{x}$ is the vector of the unknowns.

In order to find an approximation for $\mathbf{x}$, we could use *direct methods*. However, often numerical approximation of many problems, in particular of partial differential equations, produces sparse and/or structured matrices A. Direct methods may not be the best choice when A is large and sparse and/or structured because

- they can destroy the underlying sparsity/structure during the resolution process.
- We could be interested in approximations with a lower accuracy than the one provided by, e.g., Gaussian Elimination with pivoting. This could be due to the low precision of the data or to the truncation error introduced by the approximation process generating the linear system(s).
- They cannot use the information given by the initial guess for the solution $\mathbf{x}$. The former is often available (and precious) in many classes of problems.

All these issues can be fulfilled by *iterative methods*.

Observe that differently from direct methods, iterative methods do not terminate after a finite number of steps. They require some *stopping criteria* in order to become *algorithms*. We terminate the underlying

iterations when an estimate for the $\|\cdot\|$ norm of the (unknown!) relative error

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{x}\|}$$

is less than a user-prescribed quantity ε, i.e.,

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}\|}{\|\mathbf{x}\|} < \varepsilon$$

and the iteration count k is less than a maximum allowed N. Very often, the estimate of the error is based on the *residual* $\mathbf{r}^{(k)}$:

$$\mathbf{r}^{(k)} = \mathbf{b} - A\,\mathbf{x}^{(k)},$$

a quantity that is easy to compute. Unfortunately, by using the following straightforward identities

$$A\,\mathbf{e}^{(k)} = -\mathbf{r}^{(k)}, \quad \|b\| \leq \|A\|\,\|x\|,$$

we derive the following upper bound for the norm of the relative error

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{x}\|} \leq k(A)\,\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|},$$

where $\kappa(A) = \|A\|\,\|A^{-1}\|$ is the *condition number* of the matrix A. Therefore, if at step k we experience that

$$\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|} < \varepsilon$$

and stop the iterative method, then our computed approximation $\mathbf{x}^{(k)}$ *can* be quite far from $\mathbf{x}$ if $k(A)$ is large, i.e., the relative error can be $\kappa(A)$ times greater than the desired accuracy, ε.

For what concerns the use of preconditioning, there are three basic equivalent systems we can consider. Thus let us allow for an invertible matrix M, whose properties will be defined later on in specific problem context. With the following transformation, we have a *left preconditioning* (with M):

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}, \quad \tilde{A} = M^{-1}A, \tag{2.1}$$

a *right preconditioning*:

$$\begin{cases} AM^{-1}\mathbf{u} = \mathbf{b}, & \tilde{A} = AM^{-1}, \\ \mathbf{x} = M^{-1}\mathbf{u}, \end{cases} \tag{2.2}$$

or a *split preconditioning* scheme, respectively, for which we use the matrix M in the factored form $M = M_1 M_2$, supposed well defined:

$$\begin{cases} M_1^{-1} A M_2^{-1} \mathbf{u} = M_1^{-1} \mathbf{b}, \quad \tilde{A} = M_1^{-1} A M_2^{-1}, \\ \mathbf{x} = M_2^{-1} \mathbf{u}. \end{cases} \tag{2.3}$$

What kind of transformation are we looking for? Potentially, M should be invertible, approximating A in some way, if A is sparse, then it would be nice to have a sparse M as well in order to reduce the computational cost of the *matrix-vector* products that may be needed for computing $M^{-1}\mathbf{v}$ with an auxiliary iterative method or a suitable direct solver. Our selection would be fine if we achieve a faster and cheaper convergence to the desired accuracy of our iterative method with respect to what happens when applying it directly to (1.2). The first problem is that some of the requirements above are mutually contradictory and a reasonable compromise is mandatory.

The first thought could be trying $M = A$. In this way, the product $M^{-1}A$ is (A and then M are supposed invertible) $A^{-1}A = I$ and then we could obtain the solution without iterating. This trivial approach has two flaws: (a) the time needed for computing the inverse of A is usually (much) higher than solving (1.2) with, say, the popular sparse Gaussian elimination (see, e.g., [134]) and, (b) the inverse of a sparse matrix is dense, in general.

Theorem 2.1 (Gilbert [129])**.** *The inverse of a sparse matrix can be dense.*

We have to account the time for computing our transformation and the time needed for the application to the iterative solver of our preconditioner M^{-1}. We stress that applying the preconditioner does not require computing the matrix–matrix product, a procedure that is too expensive to be taken into account. In all the considered cases, it is enough to compute matrix–vector products. Summarizing the above, we can express the time required to calculate the solution T_{slv} of the linear system $A\mathbf{x} = \mathbf{b}$ with a preconditioned iterative method as:

$$T_{\text{slv}} = T_{\text{setup}} + N_{\text{it}} \times T_{\text{it}}, \tag{2.4}$$

where T_{setup} is the time for computing our transformation, N_{it} is the number of iteration of the iterative solver needed to obtain the solution within the required tolerance and T_{it} is the time needed for each iteration, supposed constant for simplicity.

The main issue is that we need to find a balance between having M as a good approximation for A, i.e., in principle, minimizing $\|M^{-1}A - I\|$

or $\|M - A\|$ in some norm, and a reasonable setup time (T_{setup}) for building (possibly implicitly) the preconditioner M and the T_{it}[1], time needed for the iterations, having in mind to reduce T_{slv} with respect to the other approaches. Otherwise we are wasting computational resources.

In the sequel we recall some notions concerning Krylov subspace methods, Section 2.1, we define the aspects of *sparsity* and *structure* we are interested in, Section 2.2, and we introduce the two general classes of preconditioners, Sections 2.3 and 2.4, of which our proposals are special instances.

2.1 *Krylov Iterative Methods*

The idea of *projection techniques* is based on the extraction of an approximate solution for

$$A\mathbf{x} = \mathbf{b},$$

$A \in \mathbb{R}^{n\times n}$, through the projection of the approximations in a specific subspace of $\mathbb{R}^n$. If $\mathcal{K}$ is the search subspace or the subspace of *candidate approximants* and is of dimension m, then in general m constraints should be imposed in order to have a hope to extract a unique solution. Usually, these constraints are imposed by m independent orthogonality conditions. This requires defining another subspace of $\mathbb{R}^n$, $\mathcal{L}$, the *subspace of constraints*. This construction is shared by other mathematical frameworks and it is called the *Petrov-Galerkin* condition.

We say that a projection technique onto the subspace $\mathcal{K}$ is *orthogonal* to $\mathcal{L}$ when the candidate approximate solution ($\tilde{\mathbf{x}}$) for the underlying linear system is determined by imposing

$$\tilde{\mathbf{x}} \in \mathcal{K}, \text{ and } \mathbf{b} - A\tilde{\mathbf{x}} \perp \mathcal{L},$$

where $\perp$ means "orthogonal to", i.e., when the scalar product of the vector $\mathbf{b} - A\tilde{\mathbf{x}}$ against any vector in $\mathcal{L}$ is zero.

Specifically, following the nomenclature in [244], we call here a projection method *orthogonal* if $\mathcal{K} = \mathcal{L}$, and then *oblique* if it is otherwise.

In order to include the information on an initial guess $\mathbf{x}^{(0)}$, we should extract $\tilde{\mathbf{x}}$ in the affine subspace $\mathbf{x}^{(0)} + \mathcal{K}$ and then the problem reformulates as finding $\tilde{\mathbf{x}}$ such that

$$\tilde{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}, \quad \mathbf{b} - A\tilde{\mathbf{x}} \perp \mathcal{L}. \tag{2.5}$$

Most standard techniques use a sequence of projections. A new projection uses a new pair of subspaces $\mathcal{K}$ and $\mathcal{L}$ at each iteration and an

1 The T_{it} increases as a consequence of the loss of sparsity of the fictitious product $M^{-1}A$.

initial guess $\mathbf{x}^{(0)}$ as the most recent approximation obtained from the previous approximation step. Note that projection forms a unifying framework for many iterative solvers, including many of the classical stationary methods, e.g., if we take $\mathcal{K} = \mathcal{L} = \text{Span}\{\mathbf{e}_i\}$, where $\mathbf{e}_i$ is the ith unitary vector of $\mathbb{R}^n$, then we obtain the Gauss-Seidel method. The projections are cycled for $i = 1, \ldots, n$ until convergence.

When $\mathcal{K} = \mathcal{L}$ the Petrov-Galerkin conditions (2.5) are called the Galerkin conditions.

Now, let us consider a very illustrative matrix representation of the above projection techniques.

Let $V = [\mathbf{v_1}, \ldots, \mathbf{v_m}]$ an $n \times m$ matrix whose columns are the vectors $\mathbf{v}_i$, $i = 1, \ldots, m$ which form a basis of $\mathcal{K}$ and $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ an $n \times m$ matrix whose columns are the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ which form a basis of $\mathcal{L}$. If the candidate approximation for the solution of the underlying linear system is

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + V\,\mathbf{y}, \tag{2.6}$$

then the orthogonality conditions for the residual vector $\mathbf{r}$ with respect to vectors $\mathbf{v}_i$

$$\mathbf{r} \perp \mathcal{L} \quad \Rightarrow \quad W^T(\mathbf{b} - A\tilde{\mathbf{x}}) = 0$$

lead to the following system of equations:

$$W^T A\, V\, \mathbf{y} = W^T\, \mathbf{r}^{(0)}, \tag{2.7}$$

where $\mathbf{r}^{(0)} = \mathbf{b} - A\,\mathbf{x}^{(0)}$ is the initial residual. If the matrix $W^T A\, V$ is nonsingular, then we can use the matrix-vector notation for writing the approximate solution $\tilde{\mathbf{x}}$ as

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + V\, \left(W^T A\, V\right)^{-1} W^T \mathbf{r}^{(0)}.$$

Usually, the matrix $M = W^T A\, V$ is not formed explicitly but algorithms can compute the product $\mathbf{w} = M\mathbf{v}$ for any vector $\mathbf{v}$ and this is enough for all iterative methods considered here.

We note that if m is small compared to n (supposed always large here) and $M = W^T A\, V$ is nonsingular, then we could compute $\tilde{\mathbf{x}}$ by solving the $m \times m$ linear system $M\mathbf{y} = W^T\mathbf{r}^{(0)}$ by, e.g., a direct method. However, M, the "projected version" of A can be singular even if A is not. An example of this issue is given by the nonsingular A defined as

$$A = \begin{pmatrix} 0 & I \\ I & I \end{pmatrix}. \tag{2.8}$$

By taking $V = W = [\mathbf{e}_1, \dots, \mathbf{e}_m]$, where the $\mathbf{e}_i$s are the canonical basis vectors of $\mathbb{R}^m$, we end with a null $m \times m$ matrix M because $W^T A V$ is made of all zeros.

There are at least two very important cases where M is nonsingular.

Proposition 2.1 (Saad [244]). *If A, $\mathcal{K}$ and $\mathcal{L}$ satisfy one of the two conditions*

(a) A is symmetric and definite and $\mathcal{K} = \mathcal{L}$, or
(b) A is nonsingular and $A\mathcal{K} = \mathcal{L}$,

then $M = W^T A V$ is nonsingular for any choice of the bases of $\mathcal{K}$ e $\mathcal{L}$.

A *Krylov subspace* $\mathcal{T}_m$ for the matrix M, with $m \in \mathbb{N}$, related to a non null vector $\mathbf{v}$ is defined as

$$\mathcal{T}_m(M, \mathbf{v}) = \text{Span}\{\mathbf{v}, M\mathbf{v}, M^2\mathbf{v}, \dots, M^{m-1}\mathbf{v}\}. \tag{2.9}$$

We note two important properties that characterize the Krylov subspace methods.

Remark 2.1.

- *The Krylov subspaces are nested, i.e., $\mathcal{T}_m(M, \mathbf{v}) \subseteq \mathcal{T}_{m+1}(M, \mathbf{v})$. A* Krylov subspace method *is an algorithm that at step $m \geq 1$ uses Krylov subspaces for $\mathcal{K}_m$ and for $\mathcal{L}_m$.*
- *The property $\mathcal{T}_m(M, \mathbf{v}) \subseteq \mathcal{T}_{m+1}(M, \mathbf{v})$, $m = 1, 2, \dots$ (and $\mathcal{T}_m(M, \mathbf{v}) = \mathbb{R}^n$ for $m \geq n$ because $\mathcal{T}_m$ is a subspace) of the Krylov subspaces implies that any method for which a Petrov-Galerkin condition holds, in exact arithmetic, terminates in at most n steps. In practice one wants the methods to produce the desired approximation to the solution of the underlying linear system within a number of iteration much fewer than n.*

By choosing $\mathcal{K}_m$ and $\mathcal{L}_m$ as different Krylov subspaces, we can have different projection methods. Here we mention only the ones that will be used in the following chapters.

In particular, we consider Krylov subspace iterative algorithms derived from the *Lanczos* and *Arnoldi* famous algorithms.

The first one was introduced in 1950 by Lanczos [178] for estimating eigenvalues of sparse symmetric matrices. It generates a sequence of symmetric tridiagonal matrices whose eigenvalues, under suitable hypotheses, converge to those of M.

The second is due to Arnoldi [6], and was published in 1951 with the idea of extending the above Lanczos strategy to nonsymmetric matrices. It is based on the Hessenberg reduction of the matrix M.

2.1.1 The Conjugate Gradient: CG

The *Conjugate Gradient* (CG) algorithm is one of the most used, well known, and effective iterative techniques for solving linear systems with Hermitian and (positive or negative) definite matrices. CG is an orthogonal projection technique on the Krylov subspace $\mathcal{T}_m(M, \mathbf{r}^{(0)})$ (2.9), where $\mathbf{r}^{(0)} = \mathbf{b} - M\mathbf{x}^{(0)}$ is the initial residual. In theory, the underlying algorithm can be derived in at least two different ways:

1. by the minimization of a quadratic form, see, e.g., the seminal paper by Hestenes and Stiefel [157] and Golub and Van Loan [134];
2. by the tridiagonal reduction of the original matrix generated by the Lanczos algorithm; see Saad [244].

Here we will derive the Conjugate Gradient algorithm as a Krylov subspace method, i.e., by imposing that the approximate solution $\mathbf{x}^{(m)}$ belongs to the affine subspace $\mathbf{x}^{(0)} + \mathcal{K}_m$ where $\mathcal{K}_m$ is chosen to be a Krylov subspace, while the residual $\mathbf{r}^{(m)} = \mathbf{b} - A\,\mathbf{x}^{(m)}$ is orthogonal to another Krylov subspace, the *subspace of constraints*; see [244]. We proceed through the Lanczos Algorithm 2.1, that, under our hypotheses, transforms the underlying linear system generating a sequence of linear systems of increasing size, whose matrices are tridiagonal, i.e., the only nonzero entries are on the main, sub and super diagonals.

Algorithm 2.1: Lanczos algorithm
Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{v}_1 \in \mathbb{R}^n$ such that $\|\mathbf{v}_1\|_2 = 1$
Output: $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, $T_m = \text{trid}(\beta, \alpha, \beta) \in \mathbb{R}^{m\times m}$

```
for j = 1, ..., m do
    w_j = A v_j − β_j v_{j−1} ;
    α_j =< w_j, v_j > ;
    w_j = w_j − α_j v_j ;
    β_{j+1} = ||w_j||_2 ;
    if β_{j+1} = 0 then
        return;
    end
    v_{j+1} = w_j/||w_j||_2 ;
end
```

A sequence $\{\mathbf{x}^{(j)}\}$, converging to the solution of the linear system $A\,\mathbf{x} = \mathbf{b}$, can be generated by a sequence of m orthonormal vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$, $m \le n$, such that

$$\mathbf{x}^{(j)} - \mathbf{x}^{(0)} \in \text{Span}\{\mathbf{v}_1, \ldots, \mathbf{v}_j\},\ j \le n,$$

where $\mathbf{x}^{(0)}$ is an initial guess for $\mathbf{x}$ and $\mathbf{x}^{(j)}$ is such that the residual $\mathbf{r}^{(j)} = \mathbf{b} - A\,\mathbf{x}^{(j)}$ is minimum in some norm to be specified later. The Lanczos algorithm, introduced in 1950 for computing eigenvalues of symmetric matrices, see [178], neglecting rounding errors, generates a sequence of orthonormal vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_j\}$ by the three term relation

$$\beta_{j+1}\mathbf{v}_{j+1} = A\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}. \tag{2.10}$$

After j steps we obtain

$$\text{Span}\{\mathbf{v}_1, \ldots, \mathbf{v}_j\} = \text{Span}\{\mathbf{v}_1, A\mathbf{v}_1, \ldots, A^{j-1}\mathbf{v}_1\} = \mathcal{K}_j(A, \mathbf{v}_1).$$

If V_j is the $n \times j$ matrix whose columns are $\mathbf{v}_1, \ldots, \mathbf{v}_j$, then we obtain:

$$\mathbf{x}^{(j)} - \mathbf{x}^{(0)} = V_j\mathbf{y}^{(j)},\ \mathbf{y}^{(j)} \in \mathbb{R}^j,$$
$$V_j^T V_j = I_j,\ AV_j = V_jT_j + \tilde{\mathbf{r}}_j\mathbf{e}_j^T,\ \mathbf{e}_j^T = (0\ \cdots\ 0\ 1)_j.$$

where

$$\tilde{\mathbf{r}}_j = (A - \alpha_j I)\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}.$$

The matrix T_j is symmetric tridiagonal

$$T_j = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_j \\ & & & \beta_j & \alpha_j \end{bmatrix}. \tag{2.11}$$

If, at the mth iteration, $m \leq n$, we have $\|\tilde{\mathbf{r}}_m\| = 0$, then we find that

$$AV_m = V_mT_m \ \Rightarrow\ T_m = V_m^T A V_m,$$

i.e., A can be considered reduced to the tridiagonal T_m. By the previous steps and from (10) we find

$$A\mathbf{x} = \mathbf{b} \Rightarrow A(\mathbf{x} - \mathbf{x}^{(0)}) = \mathbf{b} - A\mathbf{x}^{(0)} = \mathbf{r}^{(0)},$$
$$\Rightarrow V_m^T A V_m\mathbf{y}_m = V_m^T\mathbf{r}^{(0)} \Rightarrow T_m\mathbf{y}_m = V_m^T\mathbf{r}^{(0)},$$

and therefore

$$\mathbf{y}_m = T_m^{-1}V_m^T\mathbf{r}^{(0)} \Rightarrow \mathbf{x} = \mathbf{x}^{(0)} + V_mT_m^{-1}(V_m^T\mathbf{r}^{(0)}). \tag{2.12}$$

From (2.12), we build the sequence $\{\mathbf{x}^{(j)}\}$ approximating $\mathbf{x}$:

$$\mathbf{x}^{(j)} = \mathbf{x}^{(0)} + V_jT_j^{-1}(V_j^T\mathbf{r}^{(0)}). \tag{2.13}$$

We do not need to store vectors $\mathbf{v}_j$, $j < i-1$. Indeed, at step m, a factorization $L_m U_m$ for T_m can be generated dynamically:

$$L_m = \begin{bmatrix} 1 & & & \\ \lambda_2 & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_m & 1 \end{bmatrix}, \; U_m = \begin{bmatrix} \eta_1 & \beta_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \beta_m \\ & & & \eta_m \end{bmatrix},$$

From (2.13) and $T_m^{-1} = U_m^{-1} L_m^{-1}$

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + V_m U_m^{-1} L_m^{-1} (V_m^T \mathbf{r}^{(0)}),$$

by posing

$$P_m = V_m U_m^{-1}, \quad \mathbf{z}_m = L_m^{-1} (V_m^T \mathbf{r}^{(0)}),$$

we express

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + P_m \mathbf{z}_m.$$

Thus $\mathbf{p}_m$, the last column of P_m, can be computed from the previous one:

$$\mathbf{p}_m = \frac{\mathbf{v}_m - \beta_m \mathbf{p}_{m-1}}{\eta_m}, \quad \lambda_m = \frac{\beta_m}{\eta_{m-1}}, \; \eta_m = \alpha_m - \lambda_m \beta_m,$$

by observing that $\mathbf{z}_m = (\mathbf{z}_{m-1} \; \zeta_m)^T$, where ζ_m is a scalar, and

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \zeta_m \mathbf{p}_m$$

that is the candidate approximation generated by updating the one of the previous step. Note that the residual vector $\mathbf{r}^{(m)} = \mathbf{b} - A\,\mathbf{x}^{(m)}$ is in the direction of $\mathbf{v}_{m+1}$, because (see [244])

$$\mathbf{r}^{(m)} = \mathbf{b} - M\mathbf{x}^{(m)} = -\beta_{m+1} (\mathbf{e}_m^T T_m^{-1} V_m^T \mathbf{r}^{(0)}) \mathbf{v}_{m+1}. \tag{2.14}$$

Moreover, the following result holds.

Proposition 2.2. *The vectors* $\mathbf{p}_1, \ldots, \mathbf{p}_m$, *where* $P_m = [\mathbf{p}_1, \ldots, \mathbf{p}_m]$, *are "A–orthogonal" or* conjugate, *i.e.,* $< A\,\mathbf{p}_i, \mathbf{p}_j >= 0$, $i \neq j$.

As a consequence of (2.14), we provide a version of CG from Lanczos Algorithm 2.1.

Let us express the solution and residual vectors at step jth as

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + \alpha_j \mathbf{p}_j, \Leftarrow \mathbf{r}^{(j+1)} = \mathbf{r}^{(j)} - \alpha_j A \mathbf{p}_j.$$

The mutual orthogonality of $\mathbf{r}^{(j)}$s give us the α_js:

$$0 =< \mathbf{r}^{(j+1)}, \mathbf{r}^{(j)} >=< \mathbf{r}^{(j)} - \alpha_j M\mathbf{p}_j, \mathbf{r}^{(j)} >,$$

$$\alpha_j = \frac{< \mathbf{r}^{(j)}, \mathbf{r}^{(j)} >}{< M\mathbf{p}_j, \mathbf{r}^{(j)} >}. \tag{2.15}$$

The next search direction, $\mathbf{p}_{j+1}$, is a linear combination of $\mathbf{r}^{(j+1)}$ and $\mathbf{p}_j$, i.e., $\mathbf{p}_{j+1} = \mathbf{r}^{(j+1)} + \beta_j \mathbf{p}_j$. Therefore, $< A\mathbf{p}_j, \mathbf{r}^{(j)} >=< A\mathbf{p}_j, \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1} >=< A\mathbf{p}_j, \mathbf{p}_j >$ that can be substituted in (2.15). Moreover, from the previous relations

$$\begin{aligned} \beta_j =& -\frac{< \mathbf{r}^{(j+1)}, M\mathbf{p}_j >}{< \mathbf{p}_j, M\mathbf{p}_j >} = \frac{< \mathbf{r}^{(j+1)}, (\mathbf{r}^{(j+1)} - \mathbf{r}^{(j)}) >}{\alpha_j < \mathbf{p}_j, M\mathbf{p}_j >} \\ =& \frac{< \mathbf{r}^{(j+1)}, \mathbf{r}^{(j+1)} >}{< \mathbf{r}^{(j)}, \mathbf{r}^{(j)} >}, \end{aligned} \tag{2.16}$$

we obtain the *Conjugate Gradient* method (Algorithm 2.2).

The CG algorithm, like the other Krylov subspace methods, has the nice property that the matrix A itself need not be formed or stored, only a routine for matrix-vector products is required in order to use the algorithm. This is the reason why Krylov subspace methods are often called *matrix-free*.

We need store only four vectors $\mathbf{x}$, $\mathbf{w}$, $\mathbf{p}$, and $\mathbf{r}$. Each iteration requires a single matrix–vector product to compute $\mathbf{w} = A\mathbf{p}$, two scalar products (one for $\mathbf{p}^T\mathbf{w}$ and one to compute $\|\mathbf{r}\|^2$), and three operations of the form $\alpha\,\mathbf{x} + \mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are vectors and α is a scalar. It is remarkable that the iteration can progress without storing a basis for the entire Krylov subspace thanks to the existence of a three-term relation for the symmetric and tridiagonal matrix in the Lanczos process. In particular, the symmetric Lanczos algorithm can be viewed as a simplification of Arnoldi's algorithm from the following section, when the matrix is symmetric.

It is clear that the CG-type algorithms, i.e., algorithms defined through short-term recurrences, are more desirable than those algorithms which require storing the entire sequences of vectors as is going to happen with the *GMRES* algorithm in the following section. The former algorithms require less memory and operations per step. An optimal Krylov subspace projection means a technique which minimizes a certain norm of the error, or residual, on the Krylov subspace independently from the starting vector. Such methods can be defined

Algorithm 2.2: Conjugate Gradient method

Input: $A \in \mathbb{R}^{n\times n}$ SPD, Maximum number of iterations N_{max}, Initial Guess $\mathbf{x}^{(0)}$

Output: $\tilde{\mathbf{x}}$, candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \|\mathbf{b} - A\mathbf{x}^{(0)}\|_2$, $\mathbf{r} = \mathbf{r}^{(0)}$, $\mathbf{p} \leftarrow \mathbf{r}$;
$\rho_0 \leftarrow \|\mathbf{r}^{(0)}\|^2$;
for $k = 1, \ldots, N_{max}$ **do**
  if $k = 1$ **then**
    $\mathbf{p} \leftarrow \mathbf{r}$;
  end
  else
    $\beta \leftarrow \rho_1/\rho_0$;
    $\mathbf{p} \leftarrow \mathbf{r} + \beta\,\mathbf{p}$;
  end
  $\mathbf{w} \leftarrow A\,\mathbf{p}$;
  $\alpha \leftarrow \rho_1/\mathbf{p}^T\mathbf{w}$;
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$;
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{w}$;
  $\rho_1 \leftarrow \|\mathbf{r}\|_2^2$;
  if <*Stopping criteria satisfied*> **then**
    Return: $\tilde{\mathbf{x}} = \mathbf{x}$;
  end
end

from the Arnoldi process. However, it was shown by Faber and Manteuffel that the latter cannot happen if A is non-Hermitian; see, e.g., [138, Chapter 6].

Now, let us consider what CG does by analyzing its convergence.

Theorem 2.2. *Let $\mathbf{x}^*$ be such that $A\,\mathbf{x}^* = \mathbf{b}$, A be symmetric and positive definite. If $\mathbb{P}_m$ is the set of polynomials of degree at most m, then the mth iteration of Conjugate Gradient produces an approximation $\mathbf{x}^{(m)}$ such that*

$$\begin{aligned}\|\mathbf{x}^* - \mathbf{x}^{(m)}\|_A &= \min_{\substack{p\in\mathbb{P}_m\\ p(0)=1}} \|p(A)\,(\mathbf{x}^* - \mathbf{x}^{(0)})\|_A \\ &\leq \|\mathbf{x}^* - \mathbf{x}^{(0)}\|_A \left| \min_{\substack{p\in\mathbb{P}_m\\ p(0)=1}} \max_{\lambda\in\lambda(A)} p(z) \right|,\end{aligned} \tag{2.17}$$

where the A–norm $\|\cdot\|_A$ is defined as $\|\cdot\|_A =< A\cdot, \cdot >^{1/2}$.

Corollary 2.1. *If A is symmetric and positive definite and the eigenvalues of A are such that $0 < \lambda_1 \leq \ldots \leq \lambda_n$, we have*

$$\frac{\|\mathbf{x}^* - \mathbf{x}^{(m)}\|_A}{\|\mathbf{x}^* - \mathbf{x}^{(0)}\|_A} \leq 2\left(\frac{\sqrt{k_2(A)} - 1}{\sqrt{k_2(A)} + 1}\right)^m, \tag{2.18}$$

where $k_2(A) = \lambda_n/\lambda_1$ is the 2–norm condition number of A.

A detailed proof for Corollary 2.1 can be found in [244, Sections 6.10 and 6.11].

Corollary 2.2. *Under the same hypotheses of Corollary 2.1, the Conjugate Gradient terminates after n iterations in exact arithmetic.*

Theorem 2.3. *Let A be Hermitian and positive definite. Let m an integer, $1 < m < n$ and $c > 0$ a constant such that for the eigenvalues of A we have*

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_{n-m+1} \leq c < \ldots \leq \lambda_n.$$

Let us suppose that the Conjugate Gradient method (Algorithm 2.2) in exact arithmetic is used. Then, fixed a $\varepsilon > 0$ we have

$$k^* = \operatorname*{argmin}_k \left\{ k \ : \ \frac{\|\mathbf{x}^* - \mathbf{x}^{(k)}\|_A}{\|\mathbf{x}^*\|_A} \leq \varepsilon \right\}$$

is bounded from above by

$$\min\left\{ \left\lceil \frac{1}{2}\sqrt{c/\lambda_1}\log\left(\frac{2}{\varepsilon}\right) + m + 1 \right\rceil, n \right\}. \tag{2.19}$$

For the matrices that naturally occurs in applications, this is a situation that is usually far from being common or realistic. Thus, to put ourselves in the hypothesis of Theorem 2.3, we can consider the *preconditioned Conjugate Gradient method (*PCG*)*. Since we are dealing with a matrix A that is symmetric and definite, we should preserve this for the preconditioned system (1.3). Therefore, we may search for a preconditioner M that is still symmetric and positive definite. With this choice, we can form the *Cholesky factorization* (see, e.g.,[133]) for $M = LL^T$, with L a lower triangular factor, and apply the *split* preconditioning from (2.3). However, this is not the only possible choice to preserve symmetry. Even in the case of left preconditioning (2.1), in which $M^{-1}A$ is not symmetric, this can be achieved. For obtaining this observe that $M^{-1}A$ is self–adjoint with respect to both the inner

products $< \cdot, \cdot >_M$ and $< \cdot, \cdot >_A$:

$$\begin{aligned} < M^{-1}A\mathbf{x}, \mathbf{y} >_M &= < A\mathbf{x}, \mathbf{y} > = < \mathbf{x}, A\mathbf{y} > = < \mathbf{x}, MM^{-1}A\mathbf{y} > \\ &= < \mathbf{x}, M^{-1}A\mathbf{y} >_M, \\ < M^{-1}A\mathbf{x}, \mathbf{y} >_A &= < AM^{-1}A\mathbf{x}, \mathbf{y} > = < \mathbf{x}, AM^{-1}A\mathbf{y} > \\ &= < \mathbf{x}, M^{-1}A\mathbf{y} >_A \ . \end{aligned} \tag{2.20}$$

Therefore, we can modify the original *CG* from Algorithm 2.2, by substituting the usual Euclidean product with the M–inner product and observing that this has not to be computed explicitly, since

$$\begin{aligned} < \mathbf{z}^{(j)}, \mathbf{z}^{(j)} >_M &= < M^{-1}\mathbf{r}^{(j)}, M^{-1}\mathbf{r}^{(j)} >_M = < \mathbf{r}^{(j)}, \mathbf{z}^{(j)} >, \\ < M^{-1}A\mathbf{p}^{(j)}, \mathbf{p}^{(j)} >_M &= < A\mathbf{p}^{(j)}, \mathbf{p}^{(j)} > . \end{aligned}$$

Therefore the substitution of the inner product can be implemented as summarized in Algorithm 2.3. Similarly one can observe that for the

Algorithm 2.3: Preconditioned Conjugate Gradient method
Input: $A \in \mathbb{R}^{n\times n}$ SPD, Maximum number of iterations N_{max}, Initial Guess $\mathbf{x}^{(0)}$, $M \in \mathbb{R}^{n\times n}$ SPD preconditioner
Output: $\tilde{\mathbf{x}}$, candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{x}^{(0)}$, $\mathbf{z}^{(0)} \leftarrow M^{-1}\mathbf{r}^{(0)}$, $\mathbf{p}^{(0)} \leftarrow \mathbf{z}^{(0)}$;
for $j = 0, \ldots, N_{max}$ **do**
  $\alpha_j \leftarrow {<\mathbf{r}^{(j)}, \mathbf{z}^{(j)}>}/{A\mathbf{p}^{(j)}, \mathbf{p}^{(j)}>}$;
  $\mathbf{x}^{(j+1)} \leftarrow \mathbf{x}^{(j)} + \alpha_j \mathbf{p}^{(j)}$;
  $\mathbf{r}^{(j+1)} \leftarrow \mathbf{r}^{(j)} - \alpha_j A\mathbf{p}^{(j)}$;
  if <*Stopping criteria satisfied*> **then**
    Return: $\tilde{\mathbf{x}} = \mathbf{x}^{(j+1)}$;
  end
  $\mathbf{z}^{(j+1)} \leftarrow M^{-1}\mathbf{r}^{(j+1)}$;
  $\beta_j \leftarrow {<\mathbf{r}^{(j+1)}, \mathbf{z}^{(j+1)}>}/{<\mathbf{r}^{(j)}, \mathbf{z}^{(j)}>}$;
  $\mathbf{p}^{(j+1)} \leftarrow \mathbf{z}^{(j+1)} + \beta_j \mathbf{p}^{(j)}$;
end

right preconditioning (2.2) the matrix AM^{-1} is not Hermitian with either the Euclidean or the M–inner product. On the other hand, we can retain symmetry (Hermitianity if A and M have complex entries) with respect to the M^{-1}–inner product. With these observations Algorithm 2.3 can be restated consequently. This reformulation is mathematically equivalent to Algorithm 2.3; see [244] for the details and the construction.

The condition under which Theorem 2.3 holds can be extended from a single matrix with a given spectrum, to the case of a sequence of matrices, in the sense of Section 1.1. As we hinted in there, when we deal with linear systems that come from real problems we never work with a single matrix A, but with a sequence $\{A_n\}_n \in \mathbb{R}^{n\times n}$ (or, possibly, $\mathbb{C}^{n\times n}$), whose dimension grows with respect to some discretization parameter. To encompass these general cases under the same theory and recognize the same *convergence behavior* given in Theorem 2.3, we need to introduce the concept of *proper cluster*.

Definition 2.1 ([132, 279] – Cluster (proper))**.** *A sequence of matrices $\{A_n\}_{n\geq 0}$, $A_n \in \mathbb{C}^{n\times n}$, has a **proper cluster** of eigenvalues at $p \in \mathbb{C}$ if, $\forall \varepsilon > 0$, the number of eigenvalues of A_n **not in** the ε–neighborhood $D(p,\varepsilon) = \{z \in \mathbb{C} \mid |z-p| < \varepsilon\}$ of p is bounded by a constant r that does not depend on n. Eigenvalues not in the* proper cluster *are called **outliers**. Furthermore, $\{A_n\}_n$ is **properly clustered** at a nonempty closed set $S \subseteq \mathbb{C}$ if for any $\varepsilon > 0$*

$$q_\varepsilon(n,S) = \sharp\left\{\lambda_j(A_n) \,:\, \lambda_j \notin D(S,\varepsilon) = \bigcup_{p\in S} D(p,\varepsilon)\right\} = O(1),\ n \to +\infty,$$

in which $D(S,\varepsilon)$ is now the ε–neighborhood of the set S, and we have denoted by $q_\varepsilon(n,S)$ the cardinality of the set of the outliers.

Thus, under the presence of a *properly clustered* spectrum, we can restate the convergence result Theorem 2.3 with a constant ratio λ_1/c that is *independent* from the size of the matrix, and thus a number of iterations needed for achieving a tolerance ε that is independent from the size of the matrix. Observe that since every A_n has only real eigenvalues, the set S of Definition 2.1 is a nonempty closed subset of $\mathbb{R}$ separated from zero. We have obtained what is usually called an *optimal rate of convergence*.

Remark 2.2. *Some requirements in Definition 2.1 can be relaxed. To ensure the optimal rate of convergence of a Krylov method for symmetric matrices the cluster should be proper. However, a somewhat "fast convergence" can be obtained also if the number of eigenvalues not in the cluster is bounded by a function of the size $o(n)$ for $n \to +\infty$. Another limiting case of this approach, as we will see in the following sections, is represented by non–Hermitian matrix sequences. Generalizations for dealing with these cases are also available, even if the role of the eigenvalues, in this case, is not anymore so crucial. We will come back on these convergence properties in Sections 2.1.2 to 2.1.4.*

2.1.2 Generalized Minimal Residual: GMRES

Generalized Minimum Residual, or *GMRES* is a projection method approximating the solution of linear system $A\mathbf{x} = \mathbf{b}$ on the affine subspace $\mathbf{x}^{(0)} + \mathcal{T}_m(A, \mathbf{v}_1)$, being $\mathbf{x}^{(0)}$ the starting guess for the solution. The Petrov–Galerkin condition for *GMRES* can be written as $\mathcal{L} = A\mathcal{T}_m(A, \mathbf{r}^{(0)})$, with $\mathbf{r}^{(0)} = \mathbf{b} - A\,\mathbf{x}^{(0)}$. *GMRES*, at each iteration, determines $\mathbf{x}^{(j+1)}$ such that $\|\mathbf{r}^{(j+1)}\|_2$ is minimum. To build it we need to introduce the *Arnoldi algorithm*: an orthogonal projection method on $\mathcal{K}_m(A, \mathbf{v}_1) = \mathcal{T}_m(A, \mathbf{v}_1)$ for general non–Hermitian matrices. It was introduced in 1951 as a way to reduce a matrix in Hessenberg form and this is the main use of it here. Moreover, Arnoldi suggested in [6] that the eigenvalues of the Hessenberg matrix generated from a $n \times n$ matrix A, after a number of steps less than n, can give approximations for the extremal eigenvalues of A. Later, the underlying algorithm was discovered to be an effective (cheap under appropriate assumptions) and powerful tool for approximating eigenvalues of large and sparse matrices. A shift–and–invert strategy is needed if one is searching other eigenvalues; see, e.g., [248] and references therein.

Algorithm 2.4: Arnoldi
Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{v}_1 \in \mathbb{R}^n$ such that $\|\mathbf{v}_1\|_2 = 1$
Output: $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, $H_m \in \mathbb{R}^{(m+1)\times m}$

1 **for** $j = 0, \ldots, m$ **do**
2 $\quad h_{i,j} \leftarrow < A\,\mathbf{v}_j, \mathbf{v}_i >,\ i = 1, \ldots, j;$
3 $\quad \mathbf{w}_j \leftarrow A\,\mathbf{v}_j - \sum_{i=1}^{j} h_{i,j}\,\mathbf{v}_i;$
4 $\quad h_{j+1,j} \leftarrow \|\mathbf{w}_j\|_2;$
5 $\quad$ **if** $h_{j+1,j} = 0$ **then**
6 $\quad\quad$ **exit**;
7 $\quad$ **end**
8 $\quad \mathbf{v}_j \leftarrow {}^{\mathbf{w}_j}/_{h_{j+1,j}};$
9 **end**

At each step in Algorithm 2.4 A is multiplied by $\mathbf{v}_j$ and the resulting vector $\mathbf{w}_j$ is orthonormalized against all previous vectors $\mathbf{v}_j$ by a Gram-Schmidt-like process. The process stops if $\mathbf{w}_j$ computed in Line 3 is the zero vector.

Note that we need to store the (usually dense) matrix $n \times m$ V_m, whose columns are given by $\mathbf{v}_1, \ldots, \mathbf{v}_m$ at each step m, and the $(m+1)\times m$ upper Hessenberg matrix $H_m = (h_{i,j})$ with $m^2 + 1$ nonzero entries.

Theorem 2.4. *If Algorithm 2.4 does not terminate before step m, then vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ form an orthonormal basis for the Krylov subspace*

$$\mathcal{K}_m(A, \mathbf{v}_1) = \text{Span}\{\mathbf{v}_1, A\,\mathbf{v}_1, \ldots, A^{m-1}\,\mathbf{v}_1\}.$$

Theorem 2.5. *Let V_m an $n \times m$ with columns $\mathbf{v}_1, \ldots, \mathbf{v}_m$, $\overline{H}_m = (h_{i,j})$ the $(m+1) \times m$ upper Hessenberg matrix whose entries are computed in the Arnoldi Algorithm 2.4 and H_m the $m \times m$ submatrix extracted from $\overline{H}_m$ by deleting its last line. Then*

$$A\,V_m = V_m\,\overline{H}_m + \mathbf{w}_m\,\mathbf{e}_m^T = V_{m+1}\,H_m, \tag{2.21}$$

where $\mathbf{e}_m$ is $[0, \ldots, 0, 1]^T \in \mathbb{R}^m$ and

$$V_m^H\,A\,Vm = H_m. \tag{2.22}$$

Theorem 2.6. *The Arnoldi Algorithm 2.4 stops at step $j < n$ if and only if the minimal polynomial of $\mathbf{v}_1$ has degree j. In this case the subspace $\mathcal{K}_j$ is invariant for A.*

Let us write the *GMRES* algorithm starting from its properties:

$$A\,V_m = V_m H_m + \mathbf{w}_m \mathbf{e}_m^T = V_{m+1}\overline{H}_m, \tag{2.23}$$

$V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ is the orthonormal basis of $\mathcal{K}_m(M, \mathbf{v}_1)$ and H_m is the $m \times m$ Hessenberg submatrix extracted from $\overline{H}_m$ by deleting the $(m+1)$th line. At step m, the candidate solution $\mathbf{x}^{(m)}$ will be the vector minimizing the residual in the 2–norm:

$$\|\mathbf{r}^{(m)}\|_2 = \|\mathbf{b} - A\,\mathbf{x}^{(m)}\|_2. \tag{2.24}$$

From the previous relations, we find

$$\begin{aligned}
\mathbf{b} - A\,\mathbf{x}^{(m)} &= \mathbf{b} - A\,(\mathbf{x}^{(0)} + V_m\mathbf{y}) \\
&= \mathbf{r}^{(0)} - A\,V_m\mathbf{y} \\
&= V_{m+1}\left(V_{m+1}^T\mathbf{r}^{(0)} - \overline{H}_m\mathbf{y}\right) \\
&= V_{m+1}(\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y})
\end{aligned}$$

where $\beta = \|\mathbf{r}^{(0)}\|_2$, $\mathbf{e}_1 = [1, 0, \ldots, 0]_{m+1}^T$. To minimize the expression

$$\|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2, \quad \mathbf{y} \in \mathbb{R}^m \tag{2.25}$$

we need to solve a linear least squares problem $(m+1)\times m$. Its approximate solution at step m is given by

$$\mathbf{x}^{(m)} = \mathbf{x}^{(0)} + V_m \mathbf{y}^{(m)},$$

where $\mathbf{y} = \mathbf{y}^{(m)} \in \mathbb{R}^m$ minimizes (2.25). We use modified Gram–Schmidt orthogonalization in *GMRES* Algorithm 2.5.

To solve the least squares problem (2.25), it is useful to transform the upper Hessenberg matrix into an upper triangular one by Givens plane rotations Q_j (see, e.g., [134]) that is

$$Q = Q_{m-1} \cdot \ldots \cdot Q_1 \in \mathbb{R}^{(m+1)\times(m+1)}$$

$$\begin{aligned} \min_{\mathbf{y}} \|\beta \mathbf{e}_1 - \overline{H}_m \mathbf{y}\|_2 &= \min_{\mathbf{y}} \|Q\beta \mathbf{e}_1 - Q\overline{H}_m \mathbf{y}\|_2 \\ &= \min_{\mathbf{y}} \|\mathbf{g}^{(m)} - \overline{R}_m \mathbf{y}\|_2 \end{aligned}$$

where $\mathbf{g}^{(m)} \in \mathbb{R}^{m+1}$, $\overline{R}_m \in \mathbb{R}^{(m+1)\times m}$, $\mathbf{y} \in \mathbb{R}^m$. It can be easily checked that the last component of $\mathbf{g}^{(m)}$ is the norm of the residual $\mathbf{r}^{(m)}$; see Saad and Schultz in [249]. By observing that

$$\begin{aligned} \|\beta \mathbf{e}_1 - \overline{H}_m \mathbf{y}\|_2^2 &= \|Q\beta \mathbf{e}_1 - Q\overline{H}_m \mathbf{y}\|_2^2 \\ &= \|\mathbf{g}^{(m)} - \overline{R}_m \mathbf{y}\|_2^2 \\ &= |\mathbf{e}_{m+1}^T \mathbf{g}^{(m)}|^2 + \|\hat{\mathbf{g}}^{(m)} - R_m \mathbf{y}\|_2^2, \end{aligned}$$

we retrieve

$$\mathbf{y}^{(m)} = R_m^{-1} \hat{\mathbf{g}}^{(m)},$$

where $R_m \in \mathbb{R}^{m\times m}$ is the upper triangular matrix extracted by deleting the last line from $\overline{R}_m$ and $\hat{\mathbf{g}}^{(m)}$ is the vector whose entries are the first m of $\mathbf{g}^{(m)}$. The plane rotations Q_j, $j = 1, \ldots, m-1$, can be applied, for each iteration of *GMRES*, at the mth matrix $\overline{H}_m$. This gives also the residual norm without computing explicitly $\mathbf{r}^{(m)} = \mathbf{b} - A\,\mathbf{x}^{(m)}$.

GMRES stops at step jth if and only if $h_{j+1,j} = 0$. In this case (and in exact arithmetic), the computed solution $\mathbf{x}^{(j)}$ can be considered exact because $\mathbf{b} - A\,\mathbf{x}^{(j)} = 0$ and $\mathbf{x} = \mathbf{x}^{(j)} \in \mathcal{K}_j(A, \mathbf{v}_1)$. It is the only possible forced stop for *GMRES*, and is a *Lucky Breakdown* differently to what we will see for bi–Lanczos, *BiCG*, *BiCGstab*,*etc.*. Note that, for the same reasons of *CG*, *GMRES* terminates in at most n iterations in exact arithmetic. Nevertheless, *GMRES* can stagnate until the last step, i.e.,

Algorithm 2.5: GMRES
Input: $A \in \mathbb{R}^{n\times n}$,$\mathbf{b} \in \mathbb{R}^n$, Maximum number of iteration m, Initial Guess $\mathbf{x}^{(0)}$
Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\,\mathbf{x}^{(0)}$;
$\beta \leftarrow \|\mathbf{r}^{(0)}\|_2$;
$\mathbf{v}_1 \leftarrow {}^{\mathbf{r}^{(0)}}/_{\beta}$;
for $j = 1, \ldots, m$ **do**
  $\mathbf{w}_j \leftarrow A\,\mathbf{v}_j$;
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} \leftarrow < \mathbf{w}_j, \mathbf{v}_i >$;
    $\mathbf{w}_j \leftarrow \mathbf{w}_j - h_{i,j}\,\mathbf{v}_i$;
  end
  $h_{j+1,j} \leftarrow \|\mathbf{w}_j\|_2$;
  if $h_{j+1,j} = 0$ *or* <*Stopping criteria satisfied*> **then**
    $m = j$;
    break;
  end
  $\mathbf{v}_{j+1} = {}^{\mathbf{w}_j}/_{\|\mathbf{w}_j\|_2}$;
end
Compute $\mathbf{y}^{(m)}$ such that
$\|\mathbf{r}^{(m)}\|_2 = \|\mathbf{b} - A\,\mathbf{x}^{(m)}\|_2 = \|\beta\mathbf{e}_1 - \underline{H}_m\mathbf{y}\|_2 = \min_{\mathbf{y}\in\mathbb{R}^m}$;
Build candidate approximation $\tilde{\mathbf{x}}$;

there can be no significant residual error reduction until the nth step; see [138] for more details and considerations on convergence issues.

There are various possibilities to modify *GMRES* for overcoming the memory issues we mentioned.
Among the most common *GMRES* variants we recall (see, e.g., Saad [244, Section 6.5]):

- Quasi-*GMRES*: instead of the orthogonalization procedure Arnoldi, we keep in memory at most a fixed number k of vectors $\mathbf{v}_{m-k+1}, \ldots, \mathbf{v}_m$. In this way, H_m becomes a banded Hessenberg matrix.
- Restarted *GMRES* (see Algorithm 2.6): after a maximum number of iterations k, usually from 10 up to 50, if *GMRES* has not reached convergence, then we set $\mathbf{r}^{(0)} = \mathbf{r}^{(m)}$, $\mathbf{x}^{(0)} = \mathbf{x}^{(m)}$, and we return to the beginning of the cycle.

The limit of these strategies is that we must pay something for the

loss of information due to partial orthogonalization, in the case of the Quasi-*GMRES*, or to restarting in the case of Restarted-*GMRES*. The price we pay can be summarized as follows: the possibility of stagnation of the algorithms when the matrix of the underlying linear system is not positive definite, less robustness, and the absence of the nice and complete convergence theory of *GMRES*.

Proposition 2.3. *If A is positive definite, i.e., $\mathbf{x}^T A\,\mathbf{x} > 0$, $\|\mathbf{x}\| > 0$, then* GMRES*(m) (and then* GMRES *that can be considered* GMRES*(m) with $m = \infty$) converges for all $m \geq 1$.*

If $\mathbf{r}^{(j)} = \mathbf{b} - A\,\mathbf{x}^{(j)}$, then at step jth of *GMRES*, we have

$$\begin{aligned} \|\mathbf{r}^{(j)}\|_2 &= \|(I - A\,\mathrm{q}_{j-1}(A))\mathbf{r}^{(0)}\|_2 \\ &= \min_{\mathrm{q}(z)\in\mathbb{P}_{j-1}} \|(I - A\mathrm{q}(A))\mathbf{r}^{(0)}\|_2, \end{aligned}$$

where $\mathrm{q}_j(z)$ is a polynomial of degree at most $j-1$ in z.

Theorem 2.7. *If A can be diagonalized, i.e., if we can find $X \in \mathbb{R}^{n\times n}$ non singular and such that*

$$A = X\,\Lambda\,X^{-1},\ \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n),\ \kappa_2(X) = \|X\|_2\,\|X^{-1}\|_2,$$

$\kappa_2(X) = \|X\|_2\,\|X^{-1}\|_2$ *condition number of X, then at step m, we have*

$$\|r\|_2 \leq \kappa_2(X)\|\mathbf{r}^{(0)}\|_2 \min_{\mathrm{p}(z)\in\mathbb{P}_m, \mathrm{p}(0)=1} \max_{i=1,\ldots,n} |\mathrm{p}(\lambda_i)|, \tag{2.26}$$

where $\mathrm{p}(z)$ *is the polynomial of degree less or equal to m such that* $\mathrm{p}(0) = 1$ *and the expression in the right–hand side in* (2.26) *is minimum.*

Let us focus now again on the natural case of a sequence of matrices of growing size $\{A_n\}_n$, with $A_n \in \mathbb{R}^{n\times n}$.

Thus, we can consider a linear system with a matrix A_n having a spectrum $\sigma(A_n)$ clustered around a certain point (Definition 2.1) in the complex plane far from the origin[2]. It is natural to partition $\sigma(A_n)$ as follows

$$\sigma(A_n) = \sigma_c(A_n) \cup \sigma_0(A_n) \cup \sigma_1(A_n),$$

where $\sigma_c(A_n)$ denotes the clustered set of eigenvalues of A_n and $\sigma_0(A_n) \cup \sigma_1(A_n)$ denotes the set of the outliers. Here we assume that the clustered set $\sigma_c(A_n)$ of eigenvalues is contained in a convex set Ω.

2 If the spectrum of A_n is not clustered and we consider the use of a *preconditioner* with a given matrix sequence P_n, then the following analysis will focus on, e.g., $K_n = P_n^{-1}A_n$ or $K_n = A_n P_n^{-1}$.

Algorithm 2.6: Restarted GMRES or GMRES(m)
Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, Maximum number of iteration m, Initial Guess $\mathbf{x}^{(0)}$
Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\,\mathbf{x}^{(0)}$;
$\beta \leftarrow \|\mathbf{r}^{(0)}\|_2$;
$\mathbf{v}_1 \leftarrow {}^{\mathbf{r}^{(0)}}/_{\beta}$;
for $j = 1, \ldots, m$ **do**
  $\mathbf{w}_j \leftarrow A\,\mathbf{v}_j$;
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} \leftarrow < \mathbf{w}_j, \mathbf{v}_i >$;
    $\mathbf{w}_j \leftarrow \mathbf{w}_j - h_{i,j}\,\mathbf{v}_i$;
  end
  $h_{j+1,j} \leftarrow \|\mathbf{w}_j\|_2$;
  if $h_{j+1,j} = 0$ *or* <*Stopping criteria satisfied*> **then**
    $m = j$;
    exit;
  end
  $\mathbf{v}_{j+1} = {}^{\mathbf{w}_j}/_{\|\mathbf{w}_j\|_2}$;
end
Compute $\mathbf{y}^{(m)}$ such that
$\|\mathbf{r}^{(m)}\|_2 = \|\mathbf{b} - A\,\mathbf{x}^{(m)}\|_2 = \|\beta\mathbf{e}_1 - \underline{H}_m\mathbf{y}\|_2 = \min_{\mathbf{y}\in\mathbb{R}^m}$;
if <*Stopping criteria* not *satisfied*> **then**
  $\mathbf{r}^{(0)} \leftarrow \mathbf{r}^{(m)}$;
  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(m)}$;
  Goto 2;
end
Build candidate approximation $\tilde{\mathbf{x}}$;

Now, let us consider in more detail the sets

$$\sigma_0(A_n) = \{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_{j_0}\} \quad \text{and} \quad \sigma_1(A_n) = \{\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{j_1}\}$$

denoting two sets of j_0 and j_1 outliers, respectively. The sets σ_0 and σ_1 are defined such that, if $\hat{\lambda}_j \in \sigma_0(A_n)$, then we have

$$1 < \left|1 - \frac{z}{\hat{\lambda}_j}\right| \leq c_j, \quad \forall z \in \Omega,$$

while, for $\tilde{\lambda}_j \in \sigma_1(A_n)$,

$$0 < \left|1 - \frac{z}{\tilde{\lambda}_j}\right| < 1, \quad \forall z \in \Omega,$$

respectively.

Under the above assumptions, we can state the following bound.

Theorem 2.8 (Bertaccini and Ng [39]). *The number of full* GMRES *iterations j needed to attain a tolerance ε on the relative residual in the 2-norm $\|\mathbf{r}^{(j)}\|_2/\|\mathbf{r}^{(0)}\|_2$ for the linear system $A_n\mathbf{x} = \mathbf{b}$, where A_n is diagonalizable, is bounded above by*

$$\min\left\{ j_0 + j_1 + \left\lceil \frac{\log(\varepsilon) - \log(\kappa_2(X_n))}{\log(\rho)} - \sum_{\ell=1}^{j_0} \frac{\log(c_\ell)}{\log(\rho)} \right\rceil, n \right\}, \tag{2.27}$$

where

$$\rho^k = \frac{\left(a/d + \sqrt{(a/d)^2 - 1}\right)^k + \left(a/d + \sqrt{(a/d)^2 - 1}\right)^{-k}}{\left(c/d + \sqrt{(c/d)^2 - 1}\right)^k + \left(c/d + \sqrt{(c/d)^2 - 1}\right)^{-k}}, \tag{2.28}$$

and the set $\Omega \in \mathbb{C}^+$ is the ellipse with center c, focal distance d and major semi axis a.

We stress that

$$\rho \simeq \tilde{\rho} = \frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}}.$$

In particular, when the major axis is parallel to the imaginary axis, is centered in $(c, 0)$, $c > 0$, and has length $2a$, while the minor axis $2b$, respectively, we have $a \geq b$ and

$$\tilde{\rho} = \frac{a + \sqrt{a^2 - |a^2 - b^2|}}{c + \sqrt{c^2 + |a^2 - b^2|}} = \frac{a + b}{c + \sqrt{c^2 + a^2 - b^2}}.$$

In practice for our model problems, we use this expression to approximate ρ in the bound (2.27).

According to Theorem 2.8, the outliers do not affect the asymptotic convergence rate of the *GMRES* method, but rather they introduce a latency effect of $j_0 + j_1$ iterations plus the term $\sum_{l=1}^{j_0} \log(c_l)/\log(\rho)$; see (2.27).

We stress that the condition number of X_n, the matrix that diagonalizes A_n, cannot be neglected in the above bound, otherwise the eigenvalues alone can give highly misleading information on the convergence process, see [140]. On the other hand, if X_n has a huge condition number (e.g., growing exponentially with the size of the matrix), then the underlying bound is useless.

Remark 2.3. *Following [138], let us consider some* GMRES *issues for A highly nonnormal and/or A non definite. In these two cases, the results based on the Chebyshev polynomials cannot be used any more and we have to reconsider the bound* (2.26) *on the residual and its sharpness.*

- **Nondefinite matrices.** *If the eigenvalues are clustered around the origin, then finding a minimizing polynomial that has value* 1 *on the origin and is less than* 1 *everywhere on some closed curve around the origin containing the eigenvalues, is impossible by the maximum principle. Therefore, the convergence analysis becomes less useful and the bound is not sharp at all. Similarly, it is impossible to build a polynomial having value* 1 *on the origin and having small absolute value in all the scattered eigenvalues, unless we let the polynomial degree grow. But in this way we retrieve a slow convergence*[3]*.*
- **Nonnormal matrices.** *the bound depends on the condition number of the eigenvector matrix X, if it is large, then the convergence analysis can produce again a nonsharp bound.*

We can say more: any convergence curve is possible; see the next theorem proposed in Greenbaum, Pták, and Strakoš [139] and Greenbaum and Strakoš [140].

Theorem 2.9 (Greenbaum, Pták, and Strakoš [139])**.** *Given a nonincreasing positive sequence $\{f_k\}_{k=0,\dots,n-1}$ with $f_{n-1} > 0$ and a set of nonzero complex numbers $\{\lambda_i\}_{i=1,2,\dots,n} \subset \mathbb{C}$, there exists a matrix A with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ and a right–hand side $\boldsymbol{b}$ with $\|\boldsymbol{b}\| = f_0$ such that the residual vectors $\boldsymbol{r}^{(k)}$ at each step of the* GMRES *algorithm applied to solve $A\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{x}^{(0)} = \boldsymbol{0}$, satisfy $\|\boldsymbol{r}^{(k)}\| = f_k$, $\forall\, k = 1, 2, \dots, n-1$.*

3 A high polynomial degree implies that the underlying Krylov subspace has large dimension, possibly approaching the degree of freedom of the problem or even more and therefore more iterations are required by *GMRES*.

Remark 2.4. *Observe that the assumption* $f_{n-1} > 0$ *in the above theorem means that* GMRES *reaches the solution exactly at iteration* n*, and both the dimension of the Krylov space and the degree of the minimizing polynomial is* n*. The result can be modified to obtain every iteration/residual graph.*

In case of *GMRES* or *GMRES*(m), or any other solver for non–Hermitian linear systems, the same three preconditioning options (left–, right–, split–) are available, even though we will see that the mathematical equivalence between them is lost. Moreover, if the preconditioner is ill–conditioned, then these differences become substantial.

Let us build a *GMRES* algorithm for the solution of (2.1), i.e., *GMRES* using the Krylov subspace $\mathcal{K}_m(M^{-1}A, \mathbf{r}^{(0)})$ or the *left–preconditioned Krylov subspace*, i.e.,

$$\mathcal{K}_m(M^{-1}A, \mathbf{r}^{(0)}) = \text{Span}\left\{\mathbf{r}^{(0)}, M^{-1}A\mathbf{r}^{(0)}, \dots \left(M^{-1}A\right)^{m-1} \mathbf{r}^{(0)}\right\}.$$

This is obtained simply by using the *Arnoldi* Algorithm 2.4 with the modified Gram–Schmidt process and is summarized in Algorithm 2.7. Observe that in this algorithm all the residual vectors and their norms are computed with respect to the preconditioned residuals. Therefore, to enforce a stopping criterion based on the (unpreconditioned) residual, in theory we need to multiply the preconditioned residuals by M and store them separately from the preconditioned ones. Alternatively, at each step, compute and use the true residual $\mathbf{r}^{(j)}_{\text{true}} = \mathbf{b} - A\mathbf{x}^{(j)}$, which has the same cost as the previous approach (one more matrix-vector product per iteration), and compare it in norm with $\varepsilon \|\mathbf{r}^{(0)}_{\text{true}}\|$.

Let us build *GMRES* for the solution of (2.2), i.e., *GMRES* using the Krylov subspace $\mathcal{K}_m(AM^{-1}, \mathbf{r}^{(0)})$, the *right–preconditioned Krylov subspace*:

$$\mathcal{K}_m(AM^{-1}, \mathbf{r}^{(0)}) = \text{Span}\left\{\mathbf{r}^{(0)}, AM^{-1}\mathbf{r}^{(0)}, \dots \left(AM^{-1}\right)^{m-1} \mathbf{r}^{(0)}\right\}.$$

Similarly to *GMRES* Algorithm 2.5, we derive Algorithm 2.8.

Line 16 of Algorithm 2.8 forms the approximate solution of the linear system as a linear combination of the preconditioned vectors $\mathbf{z}^{(i)}$, for $i = 1, \dots, m$. Since these vectors $\mathbf{v}^{(i)}$ are obtained by using the same preconditioner M^{-1}, i.e., $M^{-1}V_m\mathbf{y}^{(m)}$, we do not need to store them. On the other hand, if we use a nonconstant preconditioner, as is the case of performing few iterations of another method instead of a fixed approximation M for A, then we need to store all the $\mathbf{z}^{(i)}$s explicitly for providing the candidate approximation. In this way we derive *Flexible GMRES/FGMRES*; see Section 2.1.3.

Algorithm 2.7: GMRES with left preconditioning
Input: $A \in \mathbb{R}^{n\times n}$, Maximum number of iterations m, Initial Guess $\mathbf{x}^{(0)}$, $M \in \mathbb{R}^{n\times n}$ preconditioner
Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow M^{-1}(\mathbf{b} - A\mathbf{x}^{(0)})$; `/* Arnoldi process */`
$\beta \leftarrow \|\mathbf{r}^{(0)}\|_2$;
$\mathbf{v}^{(1)} = {}^{\mathbf{r}^{(0)}}/_{\beta}$;
for $j = 1, \ldots, m$ **do**
  $\mathbf{w} \leftarrow M^{-1}A\mathbf{z}^{(j)}$;
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} \leftarrow < \mathbf{w}, \mathbf{v}^{(i)} >$;
    $\mathbf{w} \leftarrow \mathbf{w} - h_{i,j}\mathbf{v}^{(i)}$;
  end
  $h_{j+1,j} \leftarrow \|\mathbf{w}\|_2$;
  $\mathbf{v}^{(j+1)} \leftarrow {}^{\mathbf{w}}/_{h_{j+1,j}}$;
end
$V_m \leftarrow [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}]$; `/* Build the Krylov subspace basis */`
$\mathbf{y}^{(m)} \leftarrow \arg\min_{\mathbf{y}} \|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$;
$\mathbf{x}^{(m)} \leftarrow \mathbf{x}^{(0)} + V_m\mathbf{y}^{(m)}$;
`// Convergence check and possibly a restart`
if <*Stopping criteria satisfied*> **then**
  Return: $\tilde{\mathbf{x}} = \mathbf{x}^{(m)}$;
else
  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(m)}$; `/* Restart */`
  goto 1;
end

If our preconditioner M is a positive definite matrix, or if M can be factored as $M = LU$, then we can use split-preconditioning. In this case we need to work with the following system equivalent to $A\mathbf{x} = \mathbf{b}$:

$$L^{-1}AU^{-1}\mathbf{u} = L^{-1}\mathbf{b}, \quad \mathbf{x} = U^{-1}\mathbf{u}.$$

We initialize the algorithm with the residual $\mathbf{r}^{(0)} = L^{-1}(\mathbf{b} - A\mathbf{x}^{(0)})$ and, to assemble the candidate approximate solution, we multiply the linear combination $V_m\mathbf{y}^{(m)}$ by U^{-1}. Note that, also in this case, the residual vectors and their norms are computed with respect to the preconditioned residuals. Therefore, to enforce a stopping criterion based on the (unpreconditioned) residual, we need to multiply the

Algorithm 2.8: GMRES with right preconditioning

Input: $A \in \mathbb{R}^{n\times n}$, Maximum number of iterations m, Initial Guess $\mathbf{x}^{(0)}$, $M \in \mathbb{R}^{n\times n}$ preconditioner

Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow b - A\mathbf{x}^{(0)}$; /* Arnoldi process */

$\beta \leftarrow \|\mathbf{r}^{(0)}\|_2$;

$\mathbf{v}^{(1)} = {}^{\mathbf{r}^{(0)}}/_{\beta}$;

for $j = 1, \ldots, m$ **do**

  $\mathbf{z}^{(j)} \leftarrow M^{-1}\mathbf{v}^{(j)}$;

  $\mathbf{w} \leftarrow A\mathbf{z}^{(j)}$;

  for $i = 1, \ldots, j$ **do**

    $h_{i,j} \leftarrow < \mathbf{w}, \mathbf{v}^{(i)} >$;

    $\mathbf{w} \leftarrow \mathbf{w} - h_{i,j}\mathbf{v}^{(i)}$;

  end

  $h_{j+1,j} \leftarrow \|\mathbf{w}\|_2$;

  $\mathbf{v}^{(j+1)} \leftarrow {}^{\mathbf{w}}/_{h_{j+1,j}}$;

end

$V_m \leftarrow [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}]$; /* Build the Krylov subspace basis */

$\mathbf{y}^{(m)} \leftarrow \arg\min_{\mathbf{y}} \|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$;

$\mathbf{x}^{(m)} \leftarrow \mathbf{x}^{(0)} + M^{-1}V_m\mathbf{y}^{(m)}$;

// Convergence check and possibly a restart

if <*Stopping criteria satisfied*> **then**

  Return: $\tilde{\mathbf{x}} = \mathbf{x}^{(m)}$;

else

  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(m)}$; /* Restart */

  goto 1;

end

preconditioned residuals by L (instead of M for the left preconditioned *GMRES*) and store them separately from the preconditioned ones. Alternatively and more computationally reliable, at each step one can compute and use the true residual $\mathbf{r}_{\text{true}}^{(j)} = \mathbf{b} - A\mathbf{x}^{(j)}$, which has the same cost as the previous approach (one more matrix-vector product per iteration), and compare it in norm with $\varepsilon\|\mathbf{r}_{\text{true}}^{(0)}\|$. Observe that if there is no concern about preserving the SPD property, or starting the theoretical analysis from a single preconditioner M used in a factored form, then we can use any two invertible matrices M_1 and M_2 instead of L and U in the above discussion.

So what kind of preconditioning (left/right/split) approach is appropriate? In principle, for a fixed preconditioner M, the spectra of the eigenvalues of the three associated linear systems (2.1), (2.2) and (2.3) are exactly the same (but the eigenvectors are different). One should expect them to behave at least similarly, even if a convergence analysis based only on the eigenvalues can be misleading for non–Hermitian problems.

Proposition 2.4 (Saad [244])**.** *The approximate solution obtained by left or right preconditioned* GMRES *is of the form*

$$\begin{aligned}\mathbf{x}^{(m)} =& \mathbf{x}^{(0)} + s_{m-1}(M^{-1}A)M^{-1}\mathbf{r}^{(0)} \\ =& \mathbf{x}^{(0)} + M^{-1}s_{m-1}(M^{-1}A)\mathbf{r}^{(0)}\end{aligned}$$

where s_{m-1} is a polynomial of degree $m-1$ that minimizes the residual norm $\|\mathbf{b} - A\mathbf{x}^{(m)}\|_2$ in the right preconditioning case (Algorithm 2.8) and the preconditioned residual norm $\|M^1(\mathbf{b} - A\mathbf{x}^m)\|_2$ in the left preconditioning case (Algorithm 2.7).

Therefore the two preconditioning approaches differ for only a multiplication by M^{-1} in the optimized quantity, while the residuals are taken in the same affine space. In many practical situations the difference in the convergence behavior of the two approaches is often not substantial if M is not ill–conditioned.

2.1.3 Flexible GMRES (FGMRES): GMRES With Variable Preconditioning

Saad [245] introduced in 1993 the *Flexible GMRES* (*FGMRES*), a generalization of *GMRES* that allows changing the preconditioner at each step. This is exploited to use few iterations of another iterative method as a preconditioner. The first paper describing a Krylov subspace method

with a variable preconditioning strategy was the one by Axelsson and Vassilevski [10] that introduced the *Generalized Conjugate Gradient method* (*GCG*). *FGMRES* can be expressed as a particular case of the GCG, but with an implementation producing a more efficient computational scheme, e.g., see [284]. In this framework we will see that iterative solvers such as *SOR*, *SSOR*, see, e.g., [87, 223], and *multigrid* or *domain decomposition methods*, e.g., [136], can be used as a preconditioner with a remarkable parallel potential.

It is natural to consider preconditioners based on iterations of other iterative methods, possibly of another Krylov subspace method. The latter case provides an *inner–outer Krylov method*, that can be viewed as having a polynomial preconditioner with a polynomial that changes from one step to the next and is defined implicitly by the polynomial generated by the (inner) Krylov subspace method.

What we need to do is change the preconditioner at every step of the standard *GMRES* algorithm with right preconditioning, i.e., in Line 5 of Algorithm 2.9 we compute

$$\mathbf{z}^{(j)} \leftarrow M_j^{-1}\mathbf{v}^j,$$

and store them for updating the $\mathbf{x}^{(m)}$ vector in Line 16. This is the simple modification producing the *Flexibile GMRES* (*FGMRES*) described in Algorithm 2.9. As remarked in [261, 262], the *FGMRES* is defined not only by the fact that we use a sequence of preconditioners $\{M_j^{-1}\}_j$, i.e., we change it from one iteration to the next, but also because the solution is obtained directly from the new preconditioned basis Z_m. We neglect the basis V_m, that we need to store anyway to perform the orthogonalization steps inside the Arnoldi process. Thus, the difference between *FGMRES* and the usual *GMRES* algorithm is that the action of AM_j^{-1} on a vector $\mathbf{v}$ of the Krylov subspace is no longer in the space generated by the columns of V_{m+1}. Therefore, the subspace Span(Z_m) is not necessarily a Krylov subspace. Nevertheless, the basis Z_m and V_{m+1} can be related by an expression similar to the one in equation (2.23):

$$AZ_m = V_{m+1}\overline{H}_m. \tag{2.29}$$

Moreover, if we denote, as usual, by H_m the $m \times m$ matrix obtained by $\overline{H}_m$ by deleting its last row and by $\tilde{\mathbf{v}}_{j+1}$ the $(j+1)$th $\mathbf{w}$ vector before the normalization, then we find (see also the similarities with (2.23))

$$AZ_m = V_m H_m + \tilde{\mathbf{v}}_{j+1}\mathbf{e}_m^T. \tag{2.30}$$

We now have the tools to state and prove an optimality property of *FGMRES* as well.

Algorithm 2.9: FGMRES: GMRES with variable preconditioning

Input: $A \in \mathbb{R}^{n\times n}$, Maximum number of iterations m, Initial Guess $\mathbf{x}^{(0)}$, $\{M_j \in \mathbb{R}^{n\times n}\}_j$ preconditioners

Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow b - A\mathbf{x}^{(0)}$; /* Arnoldi process */

$\beta \leftarrow \|\mathbf{r}^{(0)}\|_2$;

$\mathbf{v}^{(1)} = {}^{\mathbf{r}^{(0)}}/_{\beta}$;

for $j = 1, \ldots, m$ **do**

  $\mathbf{z}^{(j)} \leftarrow M_j^{-1}\mathbf{v}^{(j)}$;

  $\mathbf{w} \leftarrow A\mathbf{z}^{(j)}$;

  for $i = 1, \ldots, j$ **do**

    $h_{i,j} \leftarrow < \mathbf{w}, \mathbf{v}^{(i)} >$;

    $\mathbf{w} \leftarrow \mathbf{w} - h_{i,j}\mathbf{v}^{(i)}$;

  end

  $h_{j+1,j} \leftarrow \|\mathbf{w}\|_2$;

  $\mathbf{v}^{(j+1)} \leftarrow {}^{\mathbf{w}}/_{h_{j+1,j}}$;

end

$Z_m \leftarrow [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}]$; /* Build the Preconditioned subspace basis */

$\mathbf{y}^{(m)} \leftarrow \arg\min_{\mathbf{y}} \|\beta\mathbf{e}_1 - \overline{H}_m\mathbf{y}\|_2$;

$\mathbf{x}^{(m)} \leftarrow \mathbf{x}^{(0)} + Z_m\mathbf{y}^{(m)}$;

// Convergence check and possibly a restart

if <*Stopping criteria satisfied*> **then**

  Return: $\tilde{\mathbf{x}} = \mathbf{x}^{(m)}$;

else

  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(m)}$; /* Restart */

  goto 1;

end

Proposition 2.5 (Saad [245]). *The candidate approximate solution* $\mathbf{x}^{(m)}$ *obtained at step m of Algorithm 2.9 minimizes the residual norm* $\|\mathbf{b} - A\mathbf{x}^{(m)}\|_2$ *over the affine subspace* $\mathbf{x}^{(0)} + \text{Span}(Z_m)$.

Note that, differently from the *GMRES* and right–preconditioned *GMRES*, the nonsingularity of A no longer implies the nonsingularity of the H_j matrices.

Proposition 2.6 (Saad [245]). *Assume that* $\beta = \|\mathbf{r}^{(0)}\|_2 \neq 0$ *and that* $j-1$ *steps of* FGMRES *(Algorithm 2.9) have been successfully performed. In addition, let* H_j *be nonsingular. Then,* $\mathbf{x}^{(m)}$ *is exact if and only if* $h_{j+1,j} = 0$.

Details on *FGMRES* using as a variable preconditioner a second Krylov subspace method can be found in [260].

2.1.4 The Bi–Lanczos Algorithm: BiCG, BiCGstab and BiCGstab(l)

In addition to his famous algorithm, generalized in the algorithm called *bi–Lanczos* [179], Lanczos suggested a Krylov projection algorithm for linear systems $A\mathbf{x} = \mathbf{b}$ for nonsymmetric real or non-Hermitian matrices; see Algorithm 2.10. It produces two sequences of vectors that are bi-orthogonal, i.e.,

$$\{\mathbf{v}_j\},\ \{\mathbf{w}_j\},\ \mathbf{v}_j, \mathbf{w}_j \in \mathbb{R}^n,$$

such that $< \mathbf{v}_i, \mathbf{w}_j >= 0$, $i \neq j$, $< \mathbf{v}_i, \mathbf{w}_i >\neq 0$. $\mathbf{v}_j$, $\mathbf{w}_j$ are determined to satisfy the three-term recurrence

$$\begin{aligned} \delta_{j+1}\mathbf{v}_{j+1} &= A\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}, \\ \beta_{j+1}\mathbf{w}_{j+1} &= A^T\mathbf{w}_j - \alpha_j\mathbf{w}_j - \delta_j\mathbf{w}_{j-1}, \end{aligned} \tag{2.31}$$

and β_j, δ_j should be such that $< \mathbf{v}_j, \mathbf{w}_j >= 1$ for all j; see, e.g., [244] and Algorithm 2.10. At step j under appropriate conditions, $\mathbf{v}_1, \ldots, \mathbf{v}_m$ and $\mathbf{w}_1, \ldots, \mathbf{w}_m$ form two orthonormal bases, one for each of the Krylov subspaces

$$\mathcal{K}_m(A, \mathbf{v}_1) = \text{Span}\{\mathbf{v}_1, A\,\mathbf{v}_1, \ldots, A^{m-1}\mathbf{v}_1\},$$

and

$$\mathcal{K}_m(A^T, \mathbf{w}_1) = \text{Span}\{\mathbf{w}_1, A^T\mathbf{w}_1, \ldots, (A^T)^{m-1}\mathbf{w}_1\}.$$

Let us summarize some properties that will be used in the sequel; see [179, 244, 289] for more details.

Theorem 2.10. *If* $\delta_{m+1} > 0$, *then at step m for bi–Lanczos, we have:*

Algorithm 2.10: Lanczos Biorthogonalization or Bi–Lanczos

Input: $\mathbf{v}_1, \mathbf{w}_1 \in \mathbb{R}^n$ such that $< \mathbf{v}_1, \mathbf{w}_1 >= 1$, $A \in \mathbb{R}^{n\times n}$, $m \in \mathbb{N}$ such that $m \leq n$

Output: $V = [\mathbf{v}_1, \ldots, \mathbf{v}_j], W = [\mathbf{w}_1, \ldots, \mathbf{w}_j] \in \mathbb{R}^{n\times j}$, $j \leq m$, $T_j \in \mathbb{R}^{j\times j}$

$\beta_1 \leftarrow 0, \delta 1 \leftarrow 0$;
$\mathbf{w}_0 \leftarrow (0, \ldots, 0)^T, \mathbf{v}_0 \leftarrow (0, \ldots, 0)^T$;
for $j = 1, \ldots, m$ **do**
  $\alpha_j \leftarrow < A\mathbf{v}_j, \mathbf{w}_j >$;
  $\hat{\mathbf{v}}_{j+1} \leftarrow A\mathbf{v}_j - \alpha_j \mathbf{w}_j - \beta_j \mathbf{v}_{j-1}$;
  $\hat{\mathbf{w}}_{j+1} \leftarrow A^T\mathbf{w}_j - \alpha_j \mathbf{w}_j - \delta_j \mathbf{w}_{j-1}$;
  $\delta_{j+1} \leftarrow \left| < \hat{\mathbf{v}}_{j+1}, \hat{\mathbf{w}}_{j+1} > \right|^{1/2}$. ;
  if $\delta_{j+1} = 0$ **then**
    return
  end
  $\beta_{j+1} \leftarrow {<\hat{\mathbf{v}}_{j+1}, \hat{\mathbf{w}}_{j+1})>}/{\delta_{j+1}}$;
  $\mathbf{v}_{j+1} \leftarrow {\hat{\mathbf{v}}_{j+1}}/{\delta_{j+1}}$;
  $\mathbf{w}_{j+1} \leftarrow {\hat{\mathbf{w}}_{j+1}}/{\beta_{j+1}}$;
end

i. $< \mathbf{v}_j, \mathbf{w}_i >= 0$, $i \neq j$, *i.e.,* $\mathbf{v}_1, \ldots, \mathbf{v}_j, \mathbf{w}_1, \ldots, \mathbf{w}_j$ *are a bi-orthogonal set of vectors;*

ii. $\mathbf{v}_1, \ldots, \mathbf{v}_m$ *is a basis for* $\mathcal{K}_m(A, \mathbf{v}_1)$ *and* $\mathbf{w}_1, \ldots, \mathbf{w}_m$ *for* $\mathcal{K}_m(A^T, \mathbf{w}_1)$*;*

iii. If $V_m = [\mathbf{v}_1 \ \ldots \ \mathbf{v}_m]$, $W_m = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_m]$*, then we have:*

$$W_m^T A V_m = T_m,$$

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix}. \tag{2.32}$$

If at step m we find $\delta_{m+1} = 0$, then we have the desired basis for the subspace $\mathcal{K}_m(A, \mathbf{v}_1)$. On the other hand, if $\beta_{m+1} = 0$ with $\delta_{m+1} \neq 0$, then the basis of $\mathcal{K}_m(A^T, \mathbf{w}_1)$ is determined, *bi–Lanczos* stops, but we do not have enough info for $\mathcal{K}_m(A, \mathbf{v}_1)$. This event is called *serious breakdown* (Wilkinson [289]) and it is a pathology shared by all algorithms derived from *bi–Lanczos*. The reader is referred to [57, 58] for algorithmic

Algorithm 2.11: Bi–Conjugate Gradients, or BiCG
Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, Maximum number of iterations $N_{\max}$
Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{x}^{(0)}$;
choose $\mathbf{r}^{(0)*}$ such that $< \mathbf{r}^{(0)}, \mathbf{r}^{(0)*} > \neq 0$;
$\mathbf{p}^{(0)} \leftarrow \mathbf{r}^{(0)}$, $\mathbf{p}^{(0)*} \leftarrow \mathbf{r}^{(0)*}$;
for $j = 1, \ldots, N_{max}$ **do**
  $\alpha_j \leftarrow {<\mathbf{r}^{(j)}, \mathbf{r}^{(j)*}>}/{<A\mathbf{p}^{(j)}, \mathbf{p}^{(j)*}>}$;
  $\mathbf{x}^{(j+1)} \leftarrow \mathbf{x}^{(j)} + \alpha_j \mathbf{p}^{(j)}$;
  $\mathbf{r}^{(j+1)} \leftarrow \mathbf{r}^{(j)} - \alpha_j A\mathbf{p}^{(j)}$;
  if <*Stopping criteria satisfied*> **then**
    return $\tilde{\mathbf{x}} = \mathbf{x}^{(j+1)}$
  end
  $\mathbf{r}^{(j+1)*} \leftarrow \mathbf{r}^{(j)*} - \alpha_j A^T \mathbf{p}^{(j)*}$;
  $\beta_j \leftarrow {<\mathbf{r}^{(j+1)}, \mathbf{r}^{(j+1)*}>}/{<\mathbf{r}^{(j)}, \mathbf{r}^{(j)*}>}$;
  $\mathbf{p}^{(j+1)} \leftarrow \mathbf{r}^{(j+1)} + \beta_j \mathbf{p}^{(j)}$;
  $\mathbf{p}^{(j+1)*} \leftarrow \mathbf{r}^{(j+1)*} + \beta_j \mathbf{p}^{(j)*}$;
end

way to avoid breakdowns by jumping over the singular denominators exploiting the block bordering method for orthogonal polynomials.

From the previous proposition we obtain that the operator T_m is the oblique projection of A on the Krylov subspace $\mathcal{K}_m(A, \mathbf{v}_1)$ orthogonal to $\mathcal{K}_m(A^T, \mathbf{w}_1)$. Similarly, T_m^T represents the projection of A^T on $\mathcal{K}_m(A^T, \mathbf{w}_1)$ orthogonal to $\mathcal{K}_m(A, \mathbf{v}_1)$. We stress that *bi–Lanczos* explicitly needs the transpose matrix A^T.

The vector $\mathbf{x}^{(m)}$ generated by *bi–Lanczos* is searched in the affine subspace $\mathbf{x}^{(0)} + \mathcal{K}_m(A, \mathbf{v}_1)$ and its residual is parallel to $\mathbf{v}_{m+1}$ and orthogonal to $\mathrm{Span}\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$. From Theorem 2.10, we have:

$$\begin{gathered}
\mathbf{x}^{(j)} - \mathbf{x}^{(0)} = V_j \mathbf{y}^{(j)},\ \mathbf{y}^{(j)} \in \mathbb{R}^j \\
\Rightarrow AV_m \mathbf{y}^{(m)} = \mathbf{b} - A\mathbf{x}^{(0)} = \mathbf{r}^{(0)} \Rightarrow W_m^T A V_m \mathbf{y}^{(m)} = W_m^T \mathbf{r}^{(0)} \\
\Rightarrow T_m \mathbf{y}^{(m)} = W_m^T \mathbf{r}^{(0)} \Rightarrow \mathbf{y}^{(m)} = T_m^{-1} \left(W_m^T \mathbf{r}^{(0)} \right) \\
\Rightarrow \mathbf{x}^{(m)} = \mathbf{x}^{(0)} + V_m T_m^{-1} \left(W_m^T \mathbf{r}^{(0)} \right).
\end{gathered}$$

Let $L_m\, U_m$ be the LU factorization for T_m. By defining $P_m = V_m U_m^{-1}$ and $P_m^* = W_m L_m^{-1}$, $\mathbf{p}_1, \ldots, \mathbf{p}_m$, $\mathbf{p}_1^*, \ldots, \mathbf{p}_m^*$ column vectors of P_m and of

P_m^*, respectively, we can express $\mathbf{x}^{(m)}$ as follows:

$$\begin{aligned}\mathbf{x}^{(m)} &= \mathbf{x}^{(0)} + V_m T_m^{-1} V_m^T \mathbf{r}^{(0)} \\ &= \mathbf{x}^{(0)} + V_m U_m^{-1} L_m^{-1} V_m^T \mathbf{r}^{(0)} \\ &= \mathbf{x}^{(0)} + P_m L_m^{-1} V_m^T \mathbf{r}^{(0)} \\ &= \mathbf{x}^{(m-1)} + \alpha_{m-1} \mathbf{p}_{m-1}.\end{aligned}$$

The sequences $\mathbf{p}_\mathbf{j}^*$ and $\mathbf{p}_i$ are A-orthogonal, i.e., $< \mathbf{p}_i^*, A\,\mathbf{p}_j >= 0$ for $i \neq j$. Therefore, we derive the *Bi-Conjugate Gradient* Algorithm 2.11, or *BiCG* for short, by the bi–Lanczos simply generalizing the steps we did for the Lanczos' algorithm to get *CG* two times: one for each directions $\mathbf{p}_j$, $\mathbf{p}_j^*$. Note also that the residual vectors $\mathbf{r}^{(j)}$, $\mathbf{r}_*^{(j)}$ are parallel to $\mathbf{v}_{j+1}$, $\mathbf{w}_{j+1}$, respectively, and $\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(m)}$, $\mathbf{r}_*^{(1)}, \ldots, \mathbf{r}_*^{(m)}$ are such that $< \mathbf{r}^{(i)}, \mathbf{r}_*^{(j)} >= \delta_{i,j}$, $i \neq j$, $i, j \leq m$, where $\delta_{j,j} = 1$, $\delta_{i,j} = 0$, $i \neq j$.

BiCG can become unstable for two reasons:

1. possible breakdown of its *bi–Lanczos* core;
2. breakdown of the LU factorization (without pivoting) for T_m.

The first issue can occur for a breakdown of *bi–Lanczos* because $\mathbf{r}_*^{(m+1)} = 0$ but $\mathbf{r}^{(m+1)} \neq 0$ or $\|\mathbf{w}_{m+1}\| = 0$ with $\|\mathbf{v}_{m+1}\| \neq 0$. In practice, $\text{Span}\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ is invariant for A^T, but $\text{Span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ it is not for A.

There exist *look-ahead* techniques to overcome this issue; see [56, 59, 120], but they are computationally expensive and are more useful when *bi–Lanczos* is used for computing (some) eigenvalues of a given matrix. The *look-ahead* techniques destroy the tridiagonal structure of T_m. Therefore, often it is simpler to choose a different $\mathbf{r}_*^{(0)}$ and start again.

The computationally most expensive operations of *bi–Lanczos*–based algorithms are the two matrix-vector product per each iteration and some vector sums and scalar products. For *BiCG* we need approximately

$$m(2\,\text{nnz}(M) + 8n)$$

flop for m iterations, where $\text{nnz}(M)$ are the nonzero entries of A. Note that, as for all methods in this section and differently from *GMRES*, the vectors that must be kept in memory do not depend on the iteration number m.

BiCGStab and *BiCGStab*(l), introduced by Van der Vorst [284, 285], can be considered as transpose-free variants of *BiCG*. However, *BiCGStab* is so popular that it is better to describe it in some detail.

In order to use it as an iterative solver for the linear system $A\mathbf{x} = \mathbf{b}$, the key idea of *BiCGStab* is expressing the residual and direction vectors at step m directly as polynomials in the matrix A

$$\mathbf{r}^{(m)} = p_m(A)\mathbf{r}^{(0)}, \quad \mathbf{p}_m = q_m(A)\mathbf{r}^{(0)}, \tag{2.33}$$

where $p_m(z)$, $q_m(z)$ are polynomials of degree m in z that assume value 1 for $z = 0$. Similarly,

$$\mathbf{r}_*^{(m)} = p_m(A^T)\mathbf{r}_*^{(0)}, \quad \mathbf{p}_m^* = q_m(A^T)\mathbf{r}_*^{(0)}. \tag{2.34}$$

From *BiCG* Algorithm 2.11, by using (2.33), (2.34) we find

$$\begin{aligned} \alpha_m &= \frac{< p_m(A)\mathbf{r}^{(0)}, p_m(A^T)\mathbf{r}_*^{(0)} >}{< Aq_m(A)\mathbf{r}^{(0)}, q_m(A^T)\mathbf{r}_*^{(0)} >} \\ &= \frac{< p_m^2(A)\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} >}{< Aq_m^2(A)\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} >}. \end{aligned} \tag{2.35}$$

The parameter β_m is derived similarly. Sonneveld [266] observed that, by using (2.35), we can directly build, at each step of *BiCG*, the vectors

$$\mathbf{r}^{(m)} = p_m^2(A)\mathbf{r}^{(0)}, \quad \mathbf{p}_m = Aq_m^2(A)\mathbf{r}^{(0)}. \tag{2.36}$$

In this way, we can skip the construction of $\mathbf{r}_*^{(m)}$, $\mathbf{p}_m^*$, where $p_m(z)$, $q_m(z)$ are the same polynomials used for (2.33) and (2.34). The recurrence relations for the polynomials (2.36) are

$$\begin{aligned} p_{m+1}(z) &= p_m(z) - \alpha_m z q_m(z), \\ q_{m+1}(z) &= p_m(z) - \beta_m q_m(z). \end{aligned} \tag{2.37}$$

If we write the vectors $\mathbf{r}^{(m)}$ and $\mathbf{p}_m$, $\mathbf{q}_m$ as

$$\mathbf{r}^{(m)} = p_m^2(A)\mathbf{r}^{(0)}, \quad \mathbf{p}_m = q_m^2(A)\mathbf{r}^{(0)}, \quad \mathbf{q}_m = p_{m+1}(A)q_m(A)\mathbf{r}^{(0)},$$

then we can adapt the recurrence relations (2.37) in order to get $\mathbf{r}^{(m)}$ and $\mathbf{x}^{(m)}$ as in *BiCG*'s algorithm and thus we obtain the so–called *Conjugate Gradient Squared*, or *CGS*; see Algorithm 2.12 ([55, 244, 266] for more details). Here we will not spend much time on *CGS* because it is less robust than *BiCGstab*.

Van der Vorst in [285] observed that vector $\mathbf{r}^{(m)}$ can be expressed not only by squaring $p_m(z)$, but also with the product of $p_m(z)$ and $s_m(z)$,

Algorithm 2.12: Conjugate Gradient Squared Method (CGS)
Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, Maximum number of iterations $N_{\max}$, Initial Guess $\mathbf{x}^{(0)}$.
Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{x}^{(0)}$;
Choose an arbitrary $\mathbf{r}_*^{(0)}$ vector;
$\mathbf{p}_0 \leftarrow \mathbf{u}_0 \leftarrow \mathbf{r}^{(0)}$;
for $j = 1, \ldots, N_{max}$ **do**
  $\alpha_j \leftarrow <\mathbf{r}^{(j)}, \mathbf{r}_*^{(0)}> / <A\mathbf{p}^{(j)}, \mathbf{r}_*^{(0)}>$;
  $\mathbf{q}^{(j)} \leftarrow \mathbf{u}^{(j)} - \alpha_j A\mathbf{p}^{(j)}$;
  $\mathbf{x}^{(j+1)} \leftarrow \mathbf{x}^{(j)} + \alpha_j(\mathbf{u}^{(j)} + \mathbf{q}^{(j)})$;
  $\mathbf{r}^{(j+1)} \leftarrow \mathbf{r}^{(j)} - \alpha_j(\mathbf{u}^{(j)} + \mathbf{q}^{(j)})$;
  if <*Stopping criteria satisfied*> **then**
    return $\tilde{\mathbf{x}} = \mathbf{x}^{(j+1)}$
  end
  $\beta_j \leftarrow <\mathbf{r}^{(j+1)}, \mathbf{r}_*^{(0)}> / <\mathbf{r}^{(j)}, \mathbf{r}_*^{(0)}>$;
  $\mathbf{u}^{(j+1)} \leftarrow \mathbf{r}^{(j+1)} + \beta_j \mathbf{q}^{(j)}$;
  $\mathbf{p}_{j+1} \leftarrow \mathbf{u}^{(j+1)} + \beta_j(\mathbf{q}^{(j)} + \beta_j \mathbf{p}^{(j)})$;
end

where the latter is a polynomial of degree m, called *stabilizing polynomial*, written as m linear factors in z, $g_{m,1}(z) := (1 - \omega_m z)$. Therefore,

$$\mathbf{r}^{(m)} = s_m(A)p_m(A)\mathbf{r}^{(0)},$$

and $s_m(z)$ can be defined recursively as

$$s_{m+1}(z) = (1 - \omega_m z)s_m(z) = g_{m,1}(z)s_m(z), \tag{2.38}$$

where ω_m is a scalar, determined at each iteration to minimize $\|r_{m+1}\|_2$.

Therefore

$$\begin{aligned} s_{m+1}(z)p_{m+1}(z) &= (1 - \omega_m z)s_m(z)p_{m+1}(z) \\ &= (1 - \omega_m z)(s_m(z)p_m(z) - \alpha_m z s_m(z)q_m(z)), \\ s_m(z)q_m(z) &= s_m(z)(p_m(z) + \beta_{m-1}q_{m-1}(z)) \\ &= s_m(z)p_m(z) + \beta_{m-1}(1 - \omega_{m-1}z)s_{m-1}(z)q_{m-1}(z), \end{aligned}$$

with $p_m(z)$ being the polynomial of the residual of *BiCG* algorithm. Then, from (2.37), we find $p_{m+1}(z) = p_m(z) - \alpha_m z q_m(z)$. By writing the

Algorithm 2.13: BiCGstab method

Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, Maximum number of iterations $N_{\max}$, Initial Guess $\mathbf{x}^{(0)}$.

Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{x}^{(0)}$;
choose $\hat{\mathbf{r}}^{(0)}$ such that $< \mathbf{r}^{(0)}, \hat{\mathbf{r}}^{(0)} > \neq 0$;
$\mathbf{p}_0 \leftarrow \mathbf{r}^{(0)}$;
for $j = 1, \ldots, N_{max}$ **do**
  $\alpha_j \leftarrow {<\mathbf{r}^{(j)}, \hat{\mathbf{r}}^{(0)}>}/{<A\mathbf{p}_j, \hat{\mathbf{r}}^{(0)}>}$;
  $\mathbf{s}_j \leftarrow \mathbf{r}^{(j)} - \alpha_j A\mathbf{p}_j$;
  if <*Stopping criteria satisfied*> **then**
    $\mathbf{x}^{(j+1)} \leftarrow \mathbf{x}^{(j)} + \alpha_j \mathbf{p}_j$;
    return $\tilde{\mathbf{x}} = \mathbf{x}^{(j+1)}$
  end
  $\omega_j \leftarrow {<A\mathbf{s}_j, \mathbf{s}_j>}/{<A\mathbf{s}_j, A\mathbf{s}_j>}$;
  $\mathbf{x}^{(j+1)} \leftarrow \mathbf{x}^{(j)} + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j$;
  $\mathbf{r}^{(j+1)} \leftarrow \mathbf{s}_j - \omega_j A\mathbf{s}_j$;
  if <*Stopping criteria satisfied*> **then**
    return $\tilde{\mathbf{x}} = \mathbf{x}^{(j+1)}$
  end
  $\beta_j \leftarrow {\alpha_j <\mathbf{r}^{(j+1)}, \hat{\mathbf{r}}^{(0)}>}/{\omega_j <\mathbf{r}^{(j)}, \hat{\mathbf{r}}^{(0)}>}$;
  $\mathbf{p}_{j+1} \leftarrow \mathbf{r}^{(j+1)} + \beta_j(\mathbf{p}_j - \omega_j A\mathbf{p}_j)$;
end

direction vector $\mathbf{p}_m$ as

$$\mathbf{p}_m = q_m(A)s_m(A)\mathbf{r}^{(0)},$$

we obtain the recurrence relations for $\mathbf{r}^{(m+1)}$ and $\mathbf{p}_{m+1}$:

$$\begin{aligned} \mathbf{r}^{(m+1)} &= (I - \omega_j A)(\mathbf{r}^{(m)} - \alpha_m A\mathbf{p}_m), \\ \mathbf{p}_{m+1} &= \mathbf{r}^{(m+1)} + \beta_m(I - \omega_m A)\mathbf{p}_m. \end{aligned}$$

The computation of the scalar parameter β_m in *BiCG* requires $\mathbf{r}_*^{(m)}$:

$$\beta_m = \frac{< \mathbf{r}^{(m+1)}, \mathbf{r}_*^{(m+1)} >}{< \mathbf{r}^{(m)}, \mathbf{r}_*^{(m)} >},$$

while in *BiCGstab* the previous expression is obtained by using

$$\rho_m =< \mathbf{r}^{(m)}, \mathbf{r}_*^{(0)} >=< p_m(A)\mathbf{r}^{(0)}, s_m(A^T)\mathbf{r}_*^{(0)} >=< s_m(A)p_m(A)\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} >,$$

$$\frac{\rho_{m+1}}{\rho_m} = \frac{\omega_m < \mathbf{r}^{(m+1)}, \mathbf{r}_*^{(m+1)} >}{\alpha_m < \mathbf{r}^{(m)}, \mathbf{r}_*^{(m)} >} \Rightarrow \beta_m = \frac{\alpha_m}{\omega_m}\frac{\rho_{m+1}}{\rho_m}. \tag{2.39}$$

The (2.39) is obtained by considering that $\mathbf{r}^{(m)} = p_m(A)\mathbf{r}^{(0)}$ is orthogonal to all vectors of the form $(A^T)^i\mathbf{r}_*^{(0)}$ for $i < m$. From the previous relations we find:

$$\begin{aligned} < \mathbf{r}^{(m)}, \mathbf{r}_*^{(m)} > &= < p_m(A)\mathbf{r}^{(0)}, p_m(A^T)\mathbf{r}_*^{(0)} >= \\ &= < p_m(A)\mathbf{r}^{(0)}, p_m(A^T)\mathbf{r}_*^{(0)} >= \\ &= < p_m(A)\mathbf{r}^{(0)}, \frac{\delta_m^0}{\tau_j^0}s_m(A^T)\mathbf{r}_*^{(0)} >= \\ &= \frac{\delta_m^0}{\tau_m^0}\rho_m, \end{aligned}$$

where

$$p_m(z) = \sum_{i=0}^{m} \tau_{m-i}^{(m)} z^i, \quad s_m(z) = \sum_{i=0}^{m} \delta_{m-i}^{(m)} z^i.$$

From (2.38) we compute

$$\delta_{m+1}^0 = -\omega_j \delta_m^0, \quad \tau_{m+1}^0 = \alpha_j \tau_0^m.$$

Finally,

$$\beta_m = \frac{\alpha_m}{\omega_m}\frac{\rho_{m+1}}{\rho_m} = \frac{\alpha_m}{\omega_m}\frac{(\mathbf{r}^{(m+1)}, \mathbf{r}_*^{(0)})}{(\mathbf{r}^{(m)}, \mathbf{r}_*^{(0)})}.$$

From *BiCG* algorithm we have

$$\begin{aligned} \alpha_m &= \frac{< \mathbf{r}^{(m)}, \mathbf{r}_*^{(m)} >}{< A\mathbf{p}_m, \mathbf{p}_m^* >} = \frac{< p_m(A)\mathbf{r}^{(0)}, p_m(A^T)\mathbf{r}_*^{(0)} >}{< Aq_m(A)\mathbf{r}^{(0)}, q_m(A^T)\mathbf{r}_*^{(0)} >} \\ &= \frac{< p_m(A)\mathbf{r}^{(0)}, s_m(A^T)\mathbf{r}_*^{(0)} >}{< Aq_m(A)\mathbf{r}^{(0)}, s_m(A^T)\mathbf{r}_*^{(0)} >} \\ &= \frac{< s_m(A)p_m(A)\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} >}{< As_m(A)q_m(A)\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} >} \\ &= \frac{< \mathbf{r}^{(m)}, \mathbf{r}_*^{(0)} >}{< A\,\mathbf{p}_m, \mathbf{r}_*^{(0)} >} = \frac{\rho_m}{< A\mathbf{p}_m, \mathbf{r}_*^{(0)} >}. \end{aligned} \tag{2.40}$$

By setting

$$q_m(z) = \sum_{i=0}^{m} \gamma_{m-i}^{(m)} z^i,$$

we obtain the second line in (2.40) by observing that $\tau_m^0 \equiv \gamma_m^0$ and considering the orthogonality of $p_m(A)\mathbf{r}^{(0)}$ and $(A^T)^i \mathbf{r}_*^{(0)}$, $i < m$. The stabilization scalar ω_m is chosen for minimizing the residual norm of

$$\mathbf{r}^{(m+1)} = (I - \omega_m A) s_m(A) p_{m+1}(A) \mathbf{r}^{(0)}.$$

By writing

$$\mathbf{s}_m = \mathbf{r}^{(m)} - \alpha_m A \mathbf{p}_m,$$

we have

$$\mathbf{r}^{(m+1)} = (I - \omega_m A)\mathbf{s}_m. \tag{2.41}$$

Therefore, minimizing the norm of (2.41), we find

$$\omega_m = \frac{< A\mathbf{s}_m, \mathbf{s}_m >}{< A\mathbf{s}_m, A\mathbf{s}_m >}.$$

Moreover, the update of the candidate approximation $\mathbf{x}^{(m+1)}$ can be computed from $\mathbf{r}^{(m+1)}$ as

$$\begin{aligned} \mathbf{r}^{(m+1)} &= \mathbf{s}_m - \omega_m A \mathbf{s}_m = \mathbf{r}^{(m)} - \alpha_m A \mathbf{p}_m - \omega_m A \mathbf{s}_m \\ &= \mathbf{r}^{(m)} - \alpha_m A \mathbf{p}_m - \omega_m A \mathbf{s}_m. \end{aligned}$$

By substituting $\mathbf{r}^{(m)} = \mathbf{b} - A\mathbf{x}^{(m)}$ in the previous expression, we finally obtain

$$\begin{aligned} -A\left(\mathbf{x}^{(m+1)} - \mathbf{x}^{(m)}\right) = \mathbf{r}^{(m+1)} - \mathbf{r}^{(m)} = -\alpha_m A \mathbf{p}_m - \omega_m A \mathbf{s}_m, \\ \Rightarrow \quad \mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \alpha_m \mathbf{p}_m + \omega_m \mathbf{s}_m. \end{aligned}$$

Note that, if ω_m is relatively small or zero, then we can have instability. In this case, it is better to stop the algorithm because the update of the vectors is not reliable. Otherwise, if we continue, then we would get a stagnating convergence; see [285]. A similar behavior can be observed if $| < \mathbf{r}^{(m)}, \mathbf{r}_*^{(0)} > |$ is very small or zero. In this case it is much better to start again by providing a new $\mathbf{r}_*^{(0)}$ such that $< \mathbf{r}^{(0)}, \mathbf{r}_*^{(0)} > \neq 0$.

We have seen that *BiCGstab* can break down also for nonsingular matrices. Fortunately, there are some variants that can help.

- *BiCGstab*(l): choose a polynomial of degree l greater than one, $g_{m,l}(z)$, in (2.38);

- *BiCGStab*(l)^{OR}: choose a polynomial $g_{m,l}(z)$ of degree $l \geq 1$ in order to determine implicitly a more stable basis for $\mathcal{K}_m(A^T, \mathbf{r}_*^{(0)})$.

Here we discuss briefly only the first attempt. The second often brings less impressive improvements to the convergence; see [264, 265]. Moreover, in our experience, a more effective convergence improvement in difficult problems is much better faced with *preconditioning*.

Let us recall the update of the residual vectors in *BiCGstab*: at step m, we have $\mathbf{r}^{(m+1)}$ from $\mathbf{r}^{(m)}$ by the linear term $(I - \omega_m A)$:

$$\mathbf{r}^{(m+1)} = (I - \omega_m A)s_m(A)p_{m+1}(A)\mathbf{r}^{(0)}. \tag{2.42}$$

The matrices we are interested in here are not Hermitian, thus their eigenvalues can be complex. The polynomials that minimize the 2 norm of (2.42) can have complex roots; see [284]. Unfortunately, the polynomial in (2.42) can only have real roots because it is a product of linear (real) factors. We can overcome this issue by increasing the degree of the polynomial in the right–hand side of (2.42). By providing the polynomial in A for $\mathbf{r}^{(m+1)}$ (2.42) as a product of quadratic factors, we obtain *BiCGstab*(2), Algorithm 2.14. At each iteration, it provides two *BiCG* steps instead of one. As in [264, 284], in order to simplify the notation, the indices of the vectors are not displayed in *BiCGstab(2)*'s Algorithm 2.14.

Note that one of the two *BiCG*' steps is not used to build the approximations at the current step, but only to build the second order polynomial $g_{m,2}(z)$ in the update for $\mathbf{r}^{(m+1)}$. Therefore, in *BiCGstab(2)*, $\mathbf{r}^{(m+1)}$ is updated from $\mathbf{r}^{(m)}$ as follows:

$$\mathbf{r}^{(m+1)} = s_{m,2}(M)p_{m,2}(M)\mathbf{r}^{(0)},$$

where $s_{m,2}(z)$, $p_{m,2}(z)$ are polynomials in z given in the form of the product of quadratic factors.

Similarly to *BiCGstab*, *BiCGstab*(2) becomes unstable when one of the scalars ω_1, ω_2 or $|(\mathbf{r}^{(m)}, \mathbf{r}_*^{(0)})|$ is numerically zero.

The use of polynomials made of factors of degree 2 can be easily generalized giving *BiCGstab*(l), where, at each iteration, one performs l steps of *BiCG*; see [263, 264] for more details. However, we experienced that $l = 2$ in *BiCGstab*(l) usually is enough.

For one iteration of *BiCGstab* we need 2 matrix-vector products, 4 scalar products and 6 updates of vectors. On the other hand, one iteration of *BiCGstab*(2) requires slightly less than approximately twice the above. On the other hand, often *BiCGstab* and *BiCGstab*(2) require

Algorithm 2.14: BiCGStab(2)

Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, Maximum number of iterations $N_{\max}$, Initial Guess $\mathbf{x}^{(0)}$.

Output: $\tilde{\mathbf{x}}$ candidate approximation.

$\mathbf{r}^{(0)} = \mathbf{b} - A, \mathbf{x}^{(0)}$
choose $\mathbf{r}_*^{(0)}$ such that $(\mathbf{r}^{(0)}, \mathbf{r}_*^{(0)}) \neq 0$
$\mathbf{u}_0 = 0$, $\rho_0 = 1$, $\alpha = 0$, $\omega_2 = 1$
for $j = 1 : N_{max}$ **do**
  $\rho_1 = (\mathbf{r}, \mathbf{r}_*^{(0)})$, $\beta = \frac{\alpha\rho_1}{-\omega_2\rho_0}$, $\rho_0 = \rho_1$;
  $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$;
  $\mathbf{v} = A\mathbf{u}$;
  $\gamma = (\mathbf{v}, \mathbf{r}_*^{(0)})$, $\alpha = \rho_0/\gamma$;
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{v}$;
  $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$;
  if <*Stopping criteria satisfied*> **then**
    $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$;
    return $\tilde{\mathbf{x}} = \mathbf{x}$
  end
  $\mathbf{s} = A\mathbf{r}$;
  $\rho_1 = (\mathbf{s}, \mathbf{r}_*^{(0)})$, $\beta = \alpha\rho_1/\rho_0$, $\rho_0 = \rho_1$;
  $\mathbf{v} = \mathbf{s} - \beta\mathbf{v}$;
  $\mathbf{w} = A\mathbf{v}$;
  $\gamma = (\mathbf{w}, \mathbf{r}_*^{(0)})$, $\alpha = \rho_0/\gamma$;
  $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$;
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{v}$;
  $\mathbf{s} = \mathbf{s} - \alpha\mathbf{w}$;
  $\mathbf{t} = A\mathbf{s}$;
  $\omega_1 = (\mathbf{r}, \mathbf{s})$, $\mu = (\mathbf{s}, \mathbf{s})$, $\nu = (\mathbf{s}, \mathbf{t})$;
  $\omega_2 = (\mathbf{r}, \mathbf{t})$, $\tau = (\mathbf{t}, \mathbf{t})$;
  $\tau = \tau - \nu^2/\mu$, $\omega_2 = (\omega_2 - \nu\omega_1/\mu)/\tau$;
  $\omega_1 = (\omega_1 - \nu\omega_2)/\mu$;
  $\mathbf{x} = \mathbf{x} + \omega_1\mathbf{r} + \omega_2\mathbf{s} + \alpha\mathbf{u}$;
  $\mathbf{r} = \mathbf{r} - \omega_1\mathbf{s} - \omega_2\mathbf{t}$;
  if <*Stopping criteria satisfied*> **then**
    return $\tilde{\mathbf{x}} = \mathbf{x}$
  end
  $\mathbf{u} = \mathbf{u} - \omega_1\mathbf{v} - \omega_2\mathbf{w}$;
end

more or less the same number of matrix–vector products to converge to a prescribed tolerance.

We cannot give a convergence analysis for *BiCG*, *CGS*, *BiCGstab* and *BiCGstab*(l). Nevertheless, some hints can be given. In particular, if:

- the matrix A of the underlying linear system is not very ill-conditioned;
- A can be diagonalized with a matrix of eigenvectors not too ill-conditioned and the eigenvalues are clustered far away from the origin,

then *BiCGstab* often requires approximately the same order of matrix–vector products as *BiCG* and *BiCGStab*(2), to converge to a prescribed tolerance. Otherwise, the convergence behavior of the iterative methods mentioned can be very different. It is interesting to note that *BiCGstab*(l), for $l > 1$, is often more robust than *BiCGstab*.

2.1.5 Which Krylov Subspace Method Should We Use?

After this brief review of the main Krylov iterative methods for the solution of linear systems, we can give some clarifications and comments on the various methods we decided to use in the subsequent chapters.

For generic real and symmetric or Hermitian definite (positive or negative) problems, we use PCG (Algorithm 2.3) with suitable preconditioning, see, e.g., Chapter 4 in which it is employed in the context of evaluating the matrix–vector product of a matrix function against a generic vector.

For nonsymmetric–real and non-Hermitian linear systems the choice is more delicate and problem dependent. If the sequence of matrices has no special properties to enable fast matrix–vector multiplication, then GMRES/GMRES(m) (Algorithms 2.5, 2.6, and 2.8) is an appropriate choice if it converges in few iterations, as is the case of sequences with clustered spectrum (Definition 2.1). On the other hand, we exploit it in Chapters 3, 5, 7 and 8 whenever we need to test theorems and results on the convergence behavior of the preconditioned systems, since for these methods we possess a clear and profound convergence analysis; see again Section 2.1.2. On the other hand, when what we need is a fast and reliable solution, i.e., after we have consolidated the theoretical results regarding convergence issues, we move towards the use of the BiCGstab (Algorithm 2.13) for obtaining better timings, and this happens in all the subsequent chapters. As we have discussed in Section 2.1.4, there are cases in which the standard BiCGstab can break down also for nonsingular matrices, under such conditions we resort to the BiCGstab(2) (Algorithm 2.14); see, e.g., Chapters 5 and 8. In conclusion, we use FGMRES (Algorithm 2.9) when a preconditioner is

available which needs an auxiliary iterative method for its application, as is the case in Chapter 8, in which we apply it as an inner–outer Krylov method, in conjunction with GMRES(m) as inner method. In conclusion, only the methods BiCG (Algorithm 2.11) and CGS (Algorithm 2.12) are not explicitly employed, since they have been introduced only for deriving the BiCGstab and to describe its main features.

2.2 *Sparsity and Structure*

Usually, we will talk about and distinguish between *dense* and *sparse* matrices, since in numerical linear algebra the computations made with them need to be faced in different ways and present indeed different costs. A first *operative notion* of sparse matrix, independent of the kind of problem, is the one given by Wilkinson by negation in [290]:

> "The matrix may be sparse, either with the nonzero elements concentrated on a narrow band centered on the diagonal or alternatively they may be distributed in a less systematic manner. We shall refer to a matrix as dense if the percentage of zero elements or its distribution is such as to make it uneconomic to take advantage of their presence."

The above is a heuristic starting point that gives the idea: a *sparse* matrix is not *dense*. If we think about a sequence of matrices $A_n \in \mathbb{R}^{d_n \times d_n}$, where d_n is, e.g., a function of a discretization parameter from a PDE model, then we can give a rigorous definition.

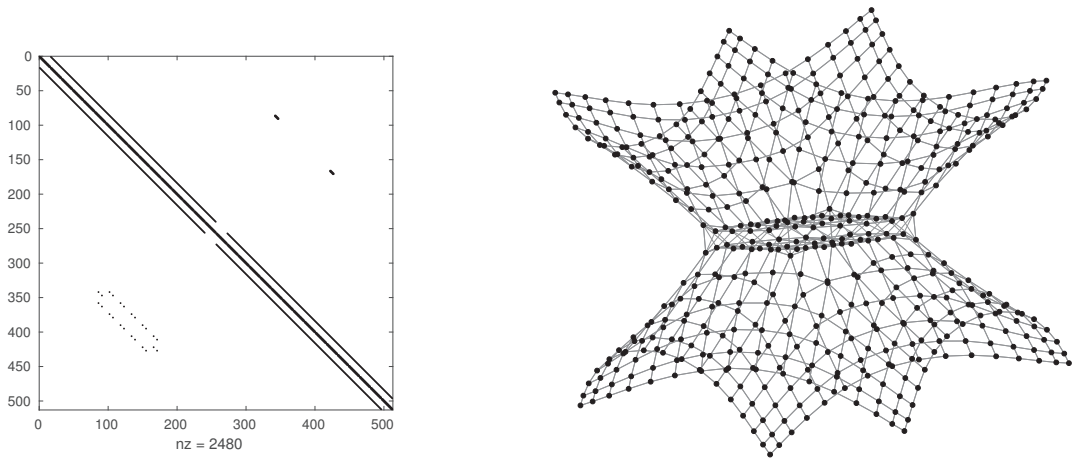
Definition 2.2 (Sparse matrix)**.** *A matrix A_n of a given sequence of matrix in $\mathbb{R}^{d_n \times d_n}$ is sparse if the number of nonzero entries of A,* $\mathrm{nnz}(A)$*, is $O(d_n)$.*

A useful tool in sparse matrix computation is *graph theory*.

Definition 2.3. *(*$\mathrm{struct}(A_n)$*) Given a sparse matrix $A_n \subset \mathbb{R}^{n \times n}$, consider the graph $G(A_n)$, called the structure of A_n, or $G(A_n) = \mathrm{struct}(A_n)$, defined by the vertex set V and edge set E:*

$$\begin{aligned} V &= \{i : 1 = 1, \ldots, n\}, \\ E &= \{(i, j) : i \neq j \text{ and } (A_n)_{i,j} \neq 0\}. \end{aligned}$$

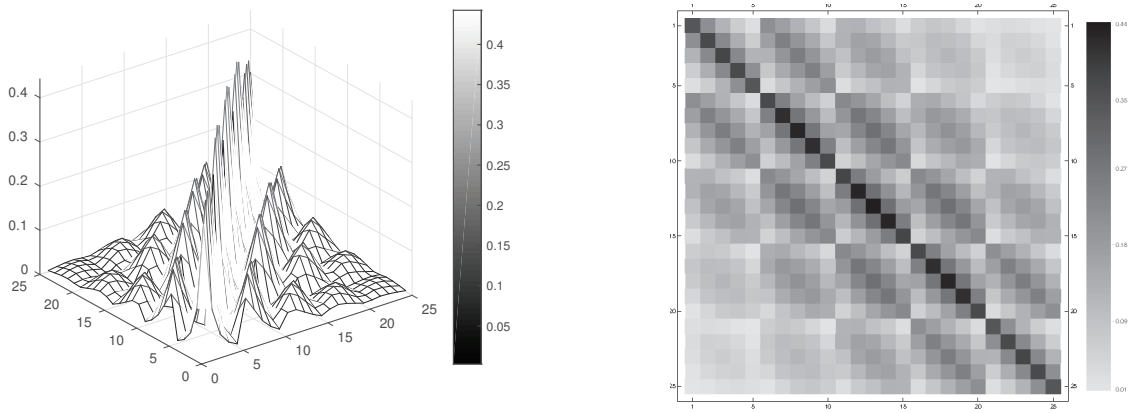
A graphical representation of both $G(A_n)$ and the sparsity pattern of a matrix A_n, i.e., a 2D plot in which the axes represent the rows and columns of A and for each nonzero entry of A_n a point is plotted, is given in Figure 2.1. Observe that in this discussion the *size* of the nonzeros entries of the matrix A_n does not play a role. This can be a limitation, as we will see in the following, see, e.g., Section 2.4.1, because in many cases we need to handle matrices that are indeed dense, but in



(a) Sparsity Pattern (b) Matrix Graph

Figure 2.1. Representing sparse matrix: pattern and matrix graph.

which many of the entries are of negligible magnitude. Sometimes these matrices are called *numerically sparse* or *localized* by *thresholding*, because by dropping the entries which fall below a threshold magnitude, they can be approximated by a sparse matrix. Another useful graphical tool is the *city plot*, based on adding a 3rd dimension to the sparsity pattern, either with a color scale or a true zeta axis, on the basis of the magnitude of the elements, see Figure 2.2. Observe also that both *sparse*



(a) City Plot (b) City Plot Pattern

Figure 2.2. Representing sparse matrix: city plot.

and *localized* matrices sometimes exhibit a distribution of their entries in some particular form or have particular properties. We will come back to this point by considering specialized preconditioners for certain *structured matrices*; see Section 2.2.1 and Part II.

2.2.1 Toeplitz and Generalized Locally Toeplitz Matrix Sequences

Toeplitz matrices arise in a wide set of contexts. They appear in the *finite difference discretization* (with uniform mesh) of differential operators [186]; in *integral equations*; in the treatment of *queue* and related problems [80], *image deconvolution*, *deblurring* and *filtering* [152]; in the *numerical computation of Padé coefficients* [54]; *time series* treatment [109] and many other fields.

This topic has attracted great interest in recent years. Analysis of this *linear space* of matrices can be performed with tools from functional analysis and the classical tools of numerical linear algebra [48–50, 141]. A large number of results are available regarding this topic.

References for the related topics in Fourier series can be found in [107, 173, 295] and will not be covered here.

Definition 2.4 (Toeplitz Matrix)**.** *A Toeplitz matrix is a matrix of the form*

$$T_n = \begin{bmatrix} t_0 & t_{-1} & \dots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & \dots & t_{2-n} \\ \vdots & t_1 & t_0 & \ddots & \vdots \\ t_{n-2} & \dots & \ddots & \ddots & t_{-1} \\ t_{n-1} & t_{n-2} & \dots & t_1 & t_0 \end{bmatrix}, \tag{2.43}$$

i.e., its entries are constant along the diagonals. A subset of this linear space of matrices is given by the matrices for which exists an $f \in \mathbb{L}^1([-\pi, \pi])$, such that

$$t_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\theta) e^{-ik\theta} d\theta, \ k = 0, \pm 1, \pm 2, \dots,$$

the t_k are the Fourier coefficients of f. In this case we write $T_n = T_n(f)$ where f is the generating function of the matrix $T_n(f)$.

This class of matrices takes their name from *Otto Toeplitz* (1881–1940) in light of his early work on bilinear forms related to Laurent series [276]. Here we focus only on the fundamental properties needed for the iterative solution of linear systems

$$T_n(f)\mathbf{x} = \mathbf{b}, \ \mathbf{x}, \mathbf{b} \in \mathbf{R}^n. \tag{2.44}$$

As one may suspect, many properties of a Toeplitz matrix generated by a function f are connected with the properties of f itself.

Proposition 2.7.

1. *The operator $T_n : \mathbb{L}^1[-\pi,\pi] \to \mathbb{C}^{n\times n}$ defined by equation (2.43) is linear and positive, i.e., if $f \geq 0$ then $T_n(f) = T_n(f)^H \; \forall\, n$ and $\mathbf{x}^H T_n(f)\mathbf{x} \geq 0 \; \forall\, \mathbf{x} \in \mathbb{C}^n$.*
2. *Let $f \in \mathbb{L}^1[-\pi,\pi]$ be such that f is real valued, $m_f = \operatorname{ess\,inf}(f)$ and $M_f = \operatorname{ess\,sup}(f)$. If $m_f > -\infty$ then $m_f \leq \lambda_j(T_n(f)) \; \forall j = 1,\dots,n$ and if $M_f < \infty$ then $M_f \geq \lambda_j(T_n(f)) \; \forall j = 1,\dots,n$. Moreover, if f is not a real constant and both the strict inequalities hold, then*

$$m_f < \lambda_j(T_n(f)) < M_f \quad \forall j = 1,\dots,n.$$

Let us recall another tool, useful to study the spectrum of the underlying matrices: the *asymptotic distribution* of the eigenvalues and of the singular values of the matrices $\{T_n(f)\}_n$.

Definition 2.5 (Asymptotic eigenvalue distribution)**.** *Given a sequence of matrices $\{X_n\}_n \in \mathbb{C}^{d_n\times d_n}$ with $d_n = \dim X_n \overset{n\to+\infty}{\longrightarrow} \infty$ monotonically and a μ-measurable function $f : D \to \mathbb{R}$, with $\mu(D) \in (0,\infty)$, we say that the sequence $\{X\}_n$ is distributed in the sense of the eigenvalues as the function f and write $\{X_n\}_n \sim_\lambda f$ if and only if, for any F continuous with bounded support, we have*

$$\lim_{n\to\infty} \frac{1}{d_n} \sum_{j=0}^{d_n} F(\lambda_j(X_n)) = \frac{1}{\mu(D)} \int_D F(f(t))dt,$$

where $\lambda_j(\cdot)$ indicates the jth eigenvalue.

Definition 2.6 (Asymptotic singular values distribution)**.** *Given a sequence of matrices $\{X_n\}_n \in \mathbb{C}^{d_n\times d_n}$ with $d_n = \dim X_n \overset{n\to+\infty}{\longrightarrow} \infty$ monotonically and a μ-measurable function $f : D \to \mathbb{R}$, with $\mu(D) \in (0,\infty)$, we say that the sequence $\{X\}_n$ is distributed in the sense of the singular values as the function f and write $\{X_n\}_n \sim_\sigma f$ if and only if, for any F continuous with bounded support, we have*

$$\lim_{n\to\infty} \frac{1}{d_n} \sum_{j=0}^{d_n} F(\sigma_j(X_n)) = \frac{1}{\mu(D)} \int_D F(|f(t)|)dt,$$

where $\sigma_j(\cdot)$ indicates the jth singular value.

The core result of this class is the Grenander and Szegö Theorem [141], with the related results in [8, 222], that prove the relations in Definitions 2.5 and 2.6 for the matrix sequence $\{T_n(f)\}$ and f the

generating function (the symbol) with restriction on the boundedness of f. A stronger result is the one obtained by Tyrtyshnikov [278] and Tyrtyshnikov and Zamarashkin [280].

Theorem 2.11 (Eigenvalue and singular value distribution)**.** *Given the generating function f, $\{T_n(f)\}$ is distributed in the sense of the eigenvalues (Definition 2.5) as f, written also as $T_n(f) \sim_\lambda f$, if one of the following conditions hold:*

1. *Grenander and Szegö [141]: f is real valued and $f \in \mathbb{L}^\infty$,*
2. *Tyrtyshnikov [278] and Tyrtyshnikov and Zamarashkin [280]: f is real valued and $f \in \mathbb{L}^1$.*

Moreover, $T_n(f)$ is distributed in the sense of the singular values (Definition 2.6) as f, written also as $T_n(f) \sim_\sigma f$, if one of the following conditions hold:

1. *Avram [8] and Parter [222]: $f \in \mathbb{L}^\infty$,*
2. *Tyrtyshnikov [278] and Tyrtyshnikov and Zamarashkin [280]: $f \in \mathbb{L}^1$.*

We now discuss briefly the computational complexity of the operation $T_n(f)\mathbf{v}$ for a generic vector $\mathbf{v} \in \mathbb{C}^n$. In principle, a matrix–vector product with the matrix $T_n(f)$ that is dense in general, costs $O(n^2)$ flops. However, $T_n(f)$ depends only on $2n-1$ parameters. Therefore, we expect that the cost of the operation can be sensibly reduced. In the sequel, we briefly recall that the Toeplitz structure of T allows reducing the cost of this operation up to $O(n \log(n))$ flops. To obtain this, we need to introduce another algebra of matrices that we will use also for preconditioning, the *circulant* matrices.

Definition 2.7 (Circulant Matrix)**.** *A **circulant matrix** $C_n \in \mathbb{C}^{n\times n}$ is a Toeplitz matrix in which each row is a cyclic shift of the row above that is $(C_n)_{i,j} = c_{(j-i) \bmod n}$:*

$$C_n = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & \dots & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \ddots & & \vdots \\ c_{n-2} & c_{n-1} & c_0 & c_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & c_2 \\ \vdots & & \ddots & \ddots & c_0 & c_1 \\ c_1 & \dots & \dots & c_{n-2} & c_{n-1} & c_0 \end{bmatrix}.$$

Proposition 2.8 (Characterization of Circulant matrices)**.** *The matrices in of the algebra of circulant matrices $\mathscr{C}$ are characterized by being simultaneously diagonalized by the unitary matrix F_n*

$$(F_n)_{j,k} = \frac{1}{\sqrt{n}} e^{2\pi i\, j\, k/n}, \quad j,k = 1,\dots,n.$$

Therefore,

$$\mathscr{C} = \left\{ C_n \in \mathbb{C}^{n\times n} \mid C_n = F_n^H D F_n \ : \ D = \operatorname{diag}(d_0, d_1, \dots, d_{n-1}) \right\}. \quad (2.45)$$

The unitary matrix F_n in Proposition 2.8 is indeed the Fourier matrix F_n. Therefore, the product $C_n\mathbf{y}$ can be formed with Algorithm 2.15. The cost of computing the matrix–vector product is reduced to the cost

Algorithm 2.15: Circulant matrix–vector product
Input: First row of the circulant matrix $\mathbf{c} = [c_0, c_1, \dots, c_{n-1}]$, $\mathbf{y}$
Output: $\mathbf{x} = C\mathbf{y}$
1 $\mathbf{f} \leftarrow F_n\mathbf{y}$;
2 $\boldsymbol{\lambda} \leftarrow \sqrt{n} F_n \mathbf{c}$; `// We are computing the eigenvalues of` C_n
3 $\mathbf{z}^T \leftarrow [f_1\lambda_1, f_2\lambda_2, \dots, f_n\lambda_n]$;
4 $\mathbf{x} \leftarrow F_n^H \mathbf{z}$;

of computing matrix–vector products with the Fourier matrix. This can be achieved by using the implementation of the DFT (Discrete Fourier Transform) algorithm[4] by Cooley and Tukey [83], also known as Fast Fourier Transform, or *FFT*, that performs the computation in $O(n\log(n))$ operations with an accuracy of $O(\varepsilon\log(n))$.

We recall that:

Theorem 2.12. *Let $\omega = \exp(\mathbf{i}\theta)$, $-\pi < \theta \leq \pi$ and W be an $n\times n$ $\{\omega\}$–circulant matrix. Then, the following Schur decomposition for W holds true:*

$$W = \Omega^H F_n^H \Lambda F_n \Omega, \quad (2.46)$$

where $\Omega = \operatorname{diag}(1, \omega^{-1/n}, \dots, \omega^{-(n-1)/n})$, Λ is a diagonal matrix containing the eigenvalues of W and F_n is the Fourier matrix.

4 There exist many low–cost implementation of the DFT algorithms based on different approaches, like the one based on *divide et impera* like the Cooley and Tukey [83], Bruun [67] and Guo, Sitton, and Burrus [144], the one by Winograd [291] based on the factorization of the polynomial $x^n - 1$ or the Good–Thomas [135, 274] working on the Chinese remainder Theorem. For an up–to–date implementation of the DFT refer to the FFTW library [121] at http://www.fftw.org/.

Note that circulant matrices are simply the {1}-circulant matrices and thus Theorem 2.12 gives also the Schur decomposition in Proposition 2.8.

We can compute the product $T_n\mathbf{v}$ in $O(n\log(n))$ operations simply by embedding the T_n matrix in a circulant matrix of size $2n$ in the following way:

$$C_{2n}\begin{bmatrix}\mathbf{v}\\ \mathbf{0}_n\end{bmatrix}=\begin{bmatrix}T_n & E_n\\ E_n & T_n\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ \mathbf{0}_n\end{bmatrix}=\begin{bmatrix}T_n\mathbf{v}\\ E_n\mathbf{v}\end{bmatrix} \tag{2.47}$$

where E_n is defined to make C_{2n} circulant, and therefore is

$$E_n=\begin{bmatrix}0 & t_{n-1} & \dots & t_2 & t_1\\ t_{1-n} & 0 & t_{n-1} & \dots & t_2\\ \vdots & t_{1-n} & 0 & \ddots & \vdots\\ t_{-2} & \dots & \ddots & \ddots & t_{n-1}\\ t_{-1} & t_{-2} & \dots & t_{1-n} & 0\end{bmatrix}.$$

Applying Algorithm 2.15 to C_{2n}, and therefore computing the matrix-vector product $T_n\mathbf{v}$ in this way, requires $O(n\log(n))$ operations.

Remark 2.5. *Observe that Algorithm 2.15 is efficient only if we have to compute a single product with T_n or C_n. If the product has to be computed repeatedly, as is exactly our case, then we should modify it by using as input directly the eigenvalues of C_{2n} or C_n and gain an FFT application for each iteration.*

Multilevel operators of this kind appear naturally from multidimensional problems discretized with uniform meshes. Among the most popular examples we can find discretizations of system of ODEs, see, e.g., [28, 40], of the system of PDEs, see e.g., [69, 76, 190]; regularization problems, see, e.g., [44, 151, 167].

Let us start with two particular examples of a *multilevel operator*:

BTTB the block Toeplitz matrix of size nm with Toeplitz blocks of size m, i.e., $T_{n,m}(f)\in\mathbb{C}^{nm\times nm}$,

$$T_{n,m}(f)=\begin{bmatrix}T_m^{(0)} & T_m^{(-1)} & \dots & T_m^{(2-n)} & T_m^{(1-n)}\\ T_m^{(1)} & T_m^{(0)} & \ddots & \dots & \vdots\\ \vdots & \ddots & T_m^{(0)} & \ddots & \vdots\\ \vdots & \dots & \ddots & \ddots & T_m^{(-1)}\\ T_m^{(n-1)} & T_m^{(n-2)} & \dots & T_m^{(1)} & T_m^{(0)}\end{bmatrix}, \tag{2.48}$$

that has an overall Toeplitz structure and in which each block $T_m^{(j)}$ for $j = 1-n, \dots, 0, \dots, n-1$ is a Toeplitz matrix of size m itself,

BCCB the block Circulant matrix of size nm with Circulant blocks of size m, i.e., $C_{n,m} \in \mathbb{C}^{nm \times nm}$,

$$C_{n,m} = \begin{bmatrix} C_m^{(0)} & C_m^{(1)} & C_m^{(2)} & \dots & \dots & C_m^{(n-1)} \\ C_m^{(n-1)} & C_m^{(0)} & C_m^{(1)} & C_m^{(2)} & & \vdots \\ C_m^{(n-2)} & C_m^{(n-1)} & C_m^{(0)} & C_m^{(1)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & C_m^{(2)} \\ \vdots & & \ddots & \ddots & C_m^{(0)} & C_m^{(1)} \\ C_m^{(1)} & \dots & \dots & C_m^{(n-2)} & C_m^{(n-1)} & C_m^{(0)} \end{bmatrix}, \quad (2.49)$$

that has an overall Circulant structure and in which each block $C_m^{(j)}$ for $j = 0, \dots, n-1$ is a Circulant matrix of size m itself.

In the general case we can give the following definition,

Definition 2.8 (Multilevel Toeplitz and Circulant Matrix)**.** *Given a number of levels $m \in \mathbb{N}$ we define the set of n–admissible multiindices $k = (k_1, \dots, k_m)$ and $l = (l_1, \dots, l_m)$ where $n = (n_1, \dots, n_m)$ and the following inequalities hold*

$$0 \leq k_j,\, l_j \leq n_j - 1, \quad j = 1, 2, \dots, m.$$

The m–level Toeplitz matrix $T_n \in \mathbb{C}^{N(n) \times N(n)}$, with $N(n) = \prod_{j=1}^{m} n_j$ is therefore defined as the matrix

$$T_N = [t_{k-l}],$$

while the m–level Circulant matrix C_N is obtained whenever

$$t_{k-l} = t_{(k-l) \mod n}$$

where, by definition, we have

$$k \mod n = (k_1 \mod n_1, \dots, k_m \mod n_m).$$

We can again associate the m–level Toeplitz matrix with a **generating** complex–valued **function** f of m real variables which is 2π–periodic in each of them.

Definition 2.9 (m–level generating function)**.** *Given $f : Q_m \to \mathbb{C}$ that is 2π–periodic in each variable on $Q_m = [-\pi, \pi]^m$ and $f \in \mathbb{L}^1$, we can consider the multiindices $k = (k_1, \ldots, k_m)$ and associate it to its Fourier series*

$$f(\mathbf{x}) \approxeq \sum_{k \in \mathbb{Z}^m} t_k e^{i \cdot k_1 x_1} \cdot \ldots \cdot e^{i \cdot k_m x_m}.$$

In this way we have the coefficients t_k needed for the construction of the multilevel matrices of Definition 2.8.

We can now formulate the same question about *asymptotic distribution* of m–level Toeplitz matrices we answered for the 1–level case with Theorem 2.11. The case with $f \in \mathbb{L}^\infty$ can be recovered from the original work of Grenander and Szegö [141], for which a detailed account is in the work of Sakrison [250]. The case of $\mathbb{L}^2$ and $\mathbb{L}^1$ generating functions is obtained in Tyrtyshnikov [278] and Tyrtyshnikov and Zamarashkin [280].

For the multilevel case no structured decomposition of the inverse is known, i.e., we do not have any generalization of the Gohberg and Semencul [130] formula and, in general, we need to resort to iterative methods. Therefore, from here on we concentrate on some examples of efficient preconditioners for the iterative methods in Section 2.1 to solve Toeplitz, multilevel Toeplitz and Toeplitz-like systems.

Note that it is even more important to reduce the computational cost of matrix-vector products with multilevel Toeplitz matrices. For the BTTB case we can apply the same embedding strategies we have used for the $T_n(f)$ matrix, i.e., we embed the $T_{n,m}(f)$ into a BCCB matrix $C_{2n,2m}$ and extend the vector $\mathbf{v}$ we want to multiply to a vector of size $4nm$. We can diagonalize the BCCB matrix $C_{2n,2m}$ with the multilevel Fourier matrix $F_{2n,2m} = F_{2n} \otimes F_{2m}$ and compute the product. This procedure has the cost of $O(nm \log(nm))$.

We can now devote our attention to the theory of *Generalized Locally Toeplitz* (GLT) sequences. This is an apparatus devised by Garoni and Serra-Capizzano [124, 125] and Serra-Capizzano [254, 255], stemming from the seminal work by Tilli [275], for computing the asymptotic spectral distribution, in the sense of Definitions 2.5 and 2.6, of the discretization matrices arising from the numerical discretization of integro–differential continuous problems; e.g., in our case Partial Differential Equations (Chapter 7) and Fractional Partial Differential Equations (Chapter 8).

We start by recalling some preliminaries needed to build the definition of GLT matrix sequences [254], and we will introduce them, as we have anticipated, as a generalization/modification of the no-

tion of Locally Toeplitz matrix sequences taken from [275]; see [254, Remark 1.1].

Let us start from the definition of approximating class of sequences, that will be the tool needed for the computation of the asymptotic spectral/singular value distribution of our matrix sequences.

Definition 2.10 (a.c.s.). *Given a sequence of matrices $\{X_n\}_n \in \mathbb{C}^{d_n \times d_n}$ we say that the sequence of matrix sequences $\{\{B_{n,m}\}_n : m \in \mathbb{N}\}$ is an approximating class of sequences for $\{X_n\}_n$, and write $\{\{B_{n,m}\}_n\} \xrightarrow{a.c.s.} \{X_n\}_n$ for $m \to +\infty$, if and only if there exist functions $\omega(m)$, $c(m)$, and n_m, independent of n, such that*

$$X_n = B_{n,m} + N_{n,m} + R_{n,m}, \tag{2.50}$$

with $\|N_{n,m}\|_2 \leq \omega(m)$ and $\operatorname{rank}(R_{n,m}) \leq c(m)n$, $\forall n \geq n_m$, with $c(m)$, $\omega(m) \overset{m\to+\infty}{\longrightarrow} 0$.

The notion of *approximating classes of sequences* (a.c.s.) was first introduced in [253]. This notion lays the foundations for all the spectral approximation results for matrix-sequences we will use in the following. Definition 2.10 states, roughly speaking, that $\{\{B_{n,m}\}_n : m \in \mathbb{N}\}$ is an a.c.s. for $\{X_n\}_n$ if, for large values of m, the sequence $\{B_{n,m}\}_n$ approximates $\{X_n\}_n$ in the sense that X_n is eventually equal to $B_{n,m}$ plus a *small–rank matrix*, where "small" is weighted with respect to the matrix size d_n, plus a *small–norm* matrix. It turns out that the notion of a.c.s. can be interpreted as a *notion of convergence* in the space of matrix–sequences $\mathfrak{E} = \{\{A_n\}_n : \{A_n\}_n \text{ is a matrix-sequence}\}$. Therefore, one can define a topology $\tau_{\text{a.c.s.}}$ on the space $\mathfrak{E}$ such that $\{\{B_{n,m}\}_n\}_m$ is an a.c.s. for $\{A_n\}_n$ if and only if $\{\{B_{n,m}\}_n\}_m$ converges to $\{A_n\}_n$ in $(\mathfrak{E}, \tau_{\text{a.c.s.}})$ in topological sense; see [122, 126].

We have already recalled two out of the three important classes of matrices, namely *circulant matrices* (Definition 2.7), and *Toeplitz matrices* 2.4, which are used as building blocks for LT and GLT matrix–sequences. What is left are the *diagonal sampling matrices*. Given a function a defined on $[0,1]$, the diagonal sampling matrix of size n generated by a is defined as

$$D_n(a) = \operatorname{diag}\left(a\left(\frac{j}{n}\right)\right)_{j=1}^{n}.$$

Definition 2.11 (LT Sequences). *A sequence of matrices $\{A_n\}_n$, where $A_n \in \mathbb{C}^{d_n \times d_n}$, is said to be* Locally Toeplitz (LT) *with respect to a symbol $k(x,\theta) = a(x)f(\theta)$, with $a : [0,1] \to \mathbb{C}$ and $f : [-\pi,\pi] \to \mathbb{C}$, if f is*

Lebesgue-measurable and the class $\{\{LT_n^m(a,f)\}_n : m \in \mathbb{N}\}$, *given by*

$$LT_n^m(a,f) = D_m(a) \otimes T_{\lfloor n/m \rfloor}(f) \oplus O_{n \bmod m},$$

with $D_m(a)$ *defined above, is an a.c.s. for* $\{A_n\}_n$. *In this case we write* $\{A_n\}_n \sim_{LT} k = a \otimes f$.

It is understood that $LT_n^m(a,f) = O_n$, being O_n the $n \times n$ zero matrix, when $n < m$ and that the term $O_{n \bmod m}$ is not present when n is an integer multiple of m. Moreover, the tensor (Kronecker) product, "$\otimes$", is always applied before the direct sum, "$\oplus$".

We now introduce the formal definition of GLT matrix sequences in the unilevel setting and we recall their most important features.

Definition 2.12 (GLT sequence)**.** *Let* $\{X_n\}_n \in \mathbb{C}^{d_n \times d_n}$ *be a matrix sequence and* $k : [0,1] \times [-\pi,\pi] \to \mathbb{C}$ *a measurable function. We say that* $\{X_n\}_n$ *is a* Generalized Locally Toeplitz (GLT) *sequence with symbol* k *if the following condition is met:*

$$\forall\, \varepsilon > 0\ \exists \{X_n^{(i,\varepsilon)}\}_n \sim_{LT} a_{i,\varepsilon} \otimes f_{i,\varepsilon},\ i = 1, \ldots, N_\varepsilon, \tag{2.51}$$

such that

- $\sum_{i=1}^{N_\varepsilon} a_{i,\varepsilon} \otimes f_{i,\varepsilon} \to k$ *in measure over* $[0,1] \times [-\pi,\pi]$ *when* $\varepsilon \to 0$,
- $\left\{ \left\{ \sum_{i=1}^{N_\varepsilon} X_n^{(i,\varepsilon)} \right\}_n \right\}_{\varepsilon > 0}$ *is an a.c.s. of* $\{X_n\}_n$ *for* $\varepsilon \to 0$,

and write $\{X_n\}_n \sim_{GLT} k$ *where we have adapted the definition of a.c.s. by using a real parameter* ε *instead of the integer* m.

We recall some properties of the GLT sequences in the following proposition; see [126] for more details. These properties represent a formulation equivalent to the original definition.

Proposition 2.9 (GLT sequences properties)**.**

GLT1 $\{A_n\}_n \sim_{GLT} \chi \Rightarrow \{A_n\}_n \sim_\sigma \chi$. *Moreover, if* $\{A_n\}_n$ *is a sequence of Hermitian matrices* $\Rightarrow \{A_n\}_n \sim_\lambda \chi$;

GLT2 $\{A_n\}_n \sim_{GLT} \chi$ *and* $A_n = X_n + Y_n$ *with each* X_n *Hermitian, norm bounded* $\|X_n\|_2 = O(1)$, *and* $\|Y_n\|_2 \to 0 \Rightarrow \{A_n\}_n \sim_\lambda \chi$;

GLT3 $\{T_n(f)\}_n \sim_{GLT} f$ *if* $f \in \mathbb{L}^1[-\pi,\pi]$, $\{D_n(a)\}_n \sim_{GLT} a$ *if* $a : [0,1] \to \mathbb{C}$ *is continuous (also Riemann-integrable) and* $\{Z_n\}_n \sim_{GLT} 0$ *if and only if* $\{Z_n\} \sim_\sigma 0$;

GLT4 The set of GLT matrices is a $*$*-algebra:*
If $\{A_n\}_n \sim_{\text{GLT}} \kappa$ *and* $\{B_n\}_n \sim_{\text{GLT}} \xi$ *then*

- $\{A_n^*\}_n \sim_{\text{GLT}} \overline{\kappa}$,
- $\{\alpha A_n + \beta B_n\}_n \sim_{\text{GLT}} \alpha\kappa + \beta\xi$ *for all* $\alpha, \beta \in \mathbb{C}$,
- $\{A_n B_n\}_n \sim_{\text{GLT}} \kappa\xi$.

GLT5 $\{B_{n,m}\}_m \sim_{GLT} \chi_m$, $\{\{B_{n,m}\}_n\}$ *is an a.c.s. for* $\{A_n\}_n$, $\chi_m \to \chi$ *in measure* $\Rightarrow \{A_n\}_n \sim_{GLT} \chi$.

GLT6 *If* $\{A_n\}_n \sim_{\text{GLT}} \kappa$ *and* $\kappa \neq 0$ *a.e. then* $\{A_n^\dagger\}_n \sim_{\text{GLT}} \kappa^{-1}$.

GLT7 *If* $\{A_n\}_n \sim_{\text{GLT}} \kappa$ *and each* A_n *is Hermitian, then* $\{f(A_n)\}_n \sim_{\text{GLT}}$ $f(\kappa)$ *for every continuous function* $f : \mathbb{C} \to \mathbb{C}$.

2.3 *Multigrid Preconditioners*

Multigrid methods (MGM) are classes of algorithms for solving linear systems, using a hierarchy of smaller linear systems. They are an example of a class of techniques called multiresolution methods, very useful in problems exhibiting multiple scales of behavior. For example, many basic relaxation methods exhibit different rates of convergence for short– and long–wavelength components of the errors, suggesting these different scales be treated differently, as in a Fourier analysis approach to multigrid. MGM can be used as solvers as well as preconditioners.

The main idea of multigrid is to accelerate the convergence of a basic iterative method, which generally reduces short–wavelength error when used for differential operators, by a correction of the fine grid solution approximation, accomplished by solving a coarse problem. The coarse problem, while cheaper to solve, is similar to the fine grid problem in that it also has short– and long–wavelength errors. It can also be solved by a combination of relaxation and coarser grids. This recursive process is repeated until a grid is reached where the cost of a direct solution there is negligible compared to the cost of one relaxation sweep on the fine grid. This multigrid cycle typically reduces all error components by a fixed amount bounded well below one, independent of the fine grid mesh size. The typical application for multigrid is in the numerical solution of some special classes of PDEs. In particular, the former has been shown to be effective for elliptic (linear) partial differential equations in multiple dimensions. Multigrid preconditioners can be effective and cheap for some combinations of specific problems and computing architectures.

Here we focus on the analysis of the so–called *algebraic multigrid* (AMG) giving the priority first to the algebraic properties of the linear systems and then to the underlying geometry.

For an extensive treatment of the multigrid solvers we suggest the books by Briggs, Henson, and McCormick [63], Ruge and Stüben

[242], and Trottenberg, Oosterlee, and Schuller [277] and the references therein.

While in the *geometric* context grids are easily interpreted in terms of the discretization of the underlying PDEs, for the AMG *grids* can be intended simply as an index set that cannot be explicitly corresponding to any geometrical grid. Therefore, let us suppose having a sequence of coarser and coarser grids $\{\Omega_k\}_k$

$$\Omega_l \supset \ldots \supset \Omega_k \supset \Omega_{k-1} \supset \ldots \supset \Omega_0, \tag{2.52}$$

where the coarsest grid is characterized by the index 0 while l represents the finest grid, with the agreement that to a coarser grid corresponds a smaller index set. For each level of the grid, i.e., for each index k, we have the associated linear operators

1. $S_k^{(1)}, S_k^{(2)} : \mathbb{R}^{n_k \times n_k} \to \mathbb{R}^{n_k}$ iteration matrices of the smoothers. They can be either equal or different (one for the restriction phase and one for the prolongation).
2. $A_k : \mathbb{R}^{n_k \times n_k} \to \mathbb{R}^{n_k}$ restriction of the matrix of the linear system.
3. $I_k^{k-1} : \Omega_k \to \Omega_{k-1}$, restriction operator.
4. $I_{k-1}^k : \Omega_{k-1} \to \Omega_k$, prolongation operator.

n_k is the number of unknowns of the system at level k and the original linear system we want to solve $A\mathbf{x} = \mathbf{b}$ reads as $A_l \mathbf{x}_l = \mathbf{f}_l$. Moreover, we can introduce a parameter γ indicating what type of cycle has to be employed at the coarser level and the number of cycles that needs to be carried out at the current level. The underlying idea of this parameter is making the cycle performing more work on the coarser level, thus exploiting the possibilities of doing more work in reduced dimension. When $\gamma = 1$ the cycle is usually called a V–cycle, while for $\gamma > 1$ a W–cycle. With this notation, we can express the outline on one cycle of the multigrid Algorithm 2.16.

While in geometric multigrid the restricted matrix could be the discretization of the underlying differential problem on the coarser grids, in a general algebraic context this is no more possible, i.e., an automatic way to restrict the operator is needed. Usually, the *Galerkin condition* is selected. This corresponds in defining the sequences of restricted matrices as

$$I_{k-1}^k = (I_k^{k-1})^T, \quad A_{k-1} = I_k^{k-1} A_k I_{k-1}^k, \ \forall\, k = l-1, \ldots, 0, \tag{2.53}$$

assuming to have *full rank* restriction and prolongation operators. These choices are convenient in many ways, e.g., if the starting matrix A is

Algorithm 2.16: Multigrid cycle (MGM)

Data: Multigrid structure $\{A_k\}_{k=l}^{0}$, l, $\{S_k^{(1)}\}_{k=l}^{0}$, $\{S_k^{(2)}\}_{k=l}^{0}$, $\{I_k^{k-1}\}_{k=l}^{0}$ and $\{I_{k-1}^{k}\}_{k=l}^{0}$, $\mathbf{b}_k{}_{k=l}^{0}$, initial guess $\mathbf{u}^{(j)}$.

Output: Approximation $\mathbf{u}^{(j+1)}$ to the solution of $\mathbf{x}_l$.

Input: $\mathbf{u}_k^{(j+1)} = \text{MGM}(A_k, \mathbf{b}_k, \mathbf{x}_k^{(j)}, k, \nu_1, \nu_2, \gamma)$

`// Presmoothing`

ν_1 steps of *presmoother* $S_k^{(1)}$ applied to $A_k\tilde{x}_k^{(j)} = \mathbf{b}_k$;

`// Coarse Grid Correction`

Compute the residual $\mathbf{r}_k^{(j)} = \mathbf{b}_k - A_k\tilde{x}_k^{(j)}$;

Restrict the residual $\mathbf{r}_{k-1}^{(j)} = I_k^{k-1}\mathbf{r}_k^{(j)}$;

if $k = 1$ **then**

  Direct solver for $A_{k-1}\mathbf{e}_{k-1}^{(j)}$;

else

  for $i = 1, \ldots, \gamma$ **do**

    $\mathbf{e}_{k-1}^{(j)} = \text{MGM}(A_{k-1}, \mathbf{r}_{k-1}, 0, k-1, \nu_1, \nu_2, \gamma)$

  end

end

Prolong the error $\mathbf{e}_k^{(j)} = I_{k-1}^{k}\mathbf{e}_{k-1}^{(j)}$;

Update the approximation $\mathbf{x}_k^{(j)} = \tilde{\mathbf{x}}_k^{(j)} + \mathbf{e}_k^{(j)}$;

`// Postsmoothing`

ν_2 steps of *postsmoother* $S_k^{(2)}$ applied to $A_k\tilde{x}_k^{(j+1)} = \mathbf{b}_k$ with initial guess $\mathbf{x}_k^{(j)}$;

symmetric and positive definite then all corresponding restrictions are. Moreover all intermediate coarse grid correction operators,

$$\Pi_k = I - I_{k+1}^{k} A_{k+1}^{-1} I_k^{k+1} A_k, \tag{2.54}$$

become automatically *orthogonal projectors*.

As it stands, we can express one application of the multigrid cycle as an iteration with a classical stationary scheme, i.e., we can express an iteration matrix M_l that fully describes one sweep of multigrid recursively as

$$\begin{cases} M_0 = 0, & k = 0, \\ M_k = (S_k^{(1)})^{\nu_1}\left(I_k - I_{k-1}^{k}(I_{k-1} - M_{k-1}^{\gamma})A_{k-1}^{-1}I_k^{k-1}A_k\right)(S_k^{(2)})^{\nu_2} & k \geq 1. \end{cases}$$

A *multigrid method* is simply the fixed point iteration with matrix M_l. Thus, we can characterize the convergence by imposing $\rho(M_l) < 1$. We can start investigating the convergence of this method by selecting an appropriate *smoothing* and *coarsening* strategy.

Theorem 2.13 (Mandel [198] and McCormick [200]). *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive definite. Assume that the prolongation operators I_{k-1}^k have full rank and that the Galerkin condition* (2.53) *holds. Furthermore, given the orthogonal projector Π_k* (2.54)*, if*

$$\forall\, \mathbf{e}_k \;\exists\, \delta_1 > 0 \;:\; \|S_k^{(2)}\mathbf{e}_k\|_A^2 \le \|\mathbf{e}_k\|_A^2 - \delta\|\Pi_k\mathbf{e}_k\|_A^2, \tag{2.55}$$

independently of $\mathbf{e}_k$ and k, then the multigrid method based on the cycle in Algorithm 2.16, with $\gamma = 1$ (V–cycle), $\nu_1 = 0$ and $\nu_2 \ge 1$ (no presmoother*), has a converge factor bounded above by $\sqrt{1-\delta_1}$ with $\delta_1 \le 1$. Otherwise, if the following condition holds*

$$\forall\, \mathbf{e}_k \;\exists\, \delta_2 > 0 \;:\; \|S_k^{(1)}\mathbf{e}_k\|_A^2 \le \|\mathbf{e}_k\|_A^2 - \delta\|\Pi_k S_k^{(1)}\mathbf{e}_k\|_A^2, \tag{2.56}$$

independently of $\mathbf{e}_k$ and k, then the multigrid method based on the cycle in Algorithm 2.16, with $\gamma = 1$ (V–cycle), $\nu_1 \ge 1$ and $\nu_2 = 0$ (no postsmoother*), has a converge factor bounded above by $1/\sqrt{1-\delta_2}$ with $\delta_2 \le 1$.*

Remark 2.6. *If we have a multigrid cycle (Algorithm 2.16) with both a pre– and post–smoother satisfying relations* (2.55) *and* (2.56)*, then we can give an estimate of the convergence factor as $\sqrt{1-\delta_1/1-\delta_2}$.*

Theorem 2.13 states that it is the balance between the *smoothing* and the *coarsening* that makes the convergence possible. In the case of PDEs, the interplay between these two components can be interpreted in terms of the frequencies of the error, i.e., condition (2.55) means that the error $\mathbf{e}_k$ that cannot be efficiently reduced by the orthogonal projector Π_k needs to be uniformly reduced by $S_k^{(1)}$. On the other hand, on the components on which Π_k is effective, i.e., that are in the range of I_{k-1}^k, the smoothing operator is allowed to be ineffective. Therefore, while the *geometric multigrid* usually relies more on the availability of a smoother that is tuned on the problem, *algebraic multigrid* is usually more focused on finding a suitable matrix–dependent coarsening strategy. More specifically, in geometric solvers, all coarser levels are predefined and we select the smoothing so that this coupling is effective. On the other side, for the AMGs, we fix a simple smoothing procedure and explicitly construct an operator–dependent coarsening strategy. In real applications the line of demarcation between these two approaches

is indeed blurred, often the two kinds of information tend to be used jointly. Let us stress that the hypotheses (2.55) and (2.56) are not very easy to verify, since they mix together properties of the coarsening strategy, through the orthogonal projector Π_k, and of the smoother. We would like to have some easier to verify *sufficient conditions*, to imply hypothesis (2.55). To obtain them we need to define the following scalar product

$$< \mathbf{x}, \mathbf{y} >_{A^2} = < D^{-1}A\mathbf{x}, \mathbf{y} >, \text{ with } A, D \text{ SPD},$$

where we will usually assume that $D = \text{diag}(A)$. Then, the following two conditions can be used

$$\begin{aligned} &\exists\, \alpha > 0: \quad \|S_k^{(2)}\mathbf{e}_k\|_A^2 \leq \|\mathbf{e}_k\|_A^2 - \alpha\|\mathbf{e}_k\|_{A^2}^2, && \text{(smoothing condition)} \\ &\exists\, \beta > 0: \quad \|\Pi_k\mathbf{e}_k\|_A^2 \leq \beta\|\mathbf{e}_k\|_{A^2}^2, && \text{(approximation property)} \end{aligned}$$

having $\delta_1 = \alpha/\beta$. Similarly for hypothesis (2.56) we have, again with $\delta_2 = \alpha/\beta$, the conditions

$$\begin{aligned} &\exists\, \alpha > 0: \quad \|S_k^{(1)}\mathbf{e}_k\|_A^2 \leq \|S_k^{(1)}\mathbf{e}_k\|_A^2 - \alpha\|\mathbf{e}_k\|_{A^2}^2, && \text{(smoothing condition)} \\ &\exists\, \beta > 0: \quad \|\Pi_k\mathbf{e}_k\|_A^2 \leq \beta\|\mathbf{e}_k\|_{A^2}^2. && \text{(approximation property)} \end{aligned}$$

Smoothing conditions are usually the easiest part to be obtained. On the other hand, the approximation property can range from trickier to very hard or impossible. A way to further relax the *approximation property* is reducing the requirements of Theorem 2.13 for a proof only for the convergence of the two–grid method, i.e., the case with $l = 2, \gamma = 1$. Moreover, since the two relations can be obtained in the same way, we restrict our statement to the case in which smoothing is performed only after each coarse grid correction step, i.e., we will consider having only the smoothing matrix $S_k^{(2)}$.

Theorem 2.14 (Brandt [52]). *Let A_k be a positive definite matrix and let us consider a multigrid cycle (Algorithm 2.16) with $\gamma = 1$ and $l = 2$ in which we identify the two levels with k and K, respectively and $S_k^{(1)} = 0$, i.e., no pre–smoother is used. If $S_k^{(2)}$ satisfies the smoothing property, then the interpolation operator I_K^k has full rank and for each $\mathbf{e}_k$ we have*

$$\exists\, \beta > 0: \ \min_{\mathbf{e}_k} \|\mathbf{e}_k - I_K^k\mathbf{e}_K\|_D^2 \leq \beta\|\mathbf{e}_k\|_A^2,$$

with β independent from $\mathbf{e}_k$, then $\beta \geq \alpha$. Moreover, the two–grid method converges with a convergence factor $\sqrt{1 - \alpha/\beta}$.

Extending these convergence properties to the full V–cycle case is in practice impossible by using merely algebraic arguments, i.e., without recurring when possible to the underlying geometric properties. Nevertheless, it often turns out that two level convergence can be extended, at least approximatively, to the V–Cycle in a way independent of the problem size. Since we are mostly interested in using AMG as a preconditioner, for us this is enough, even if a further analysis in this direction would be very desirable, see [242, 269, 277] for discussions on these issues. There exist several classical proofs of convergence for multigrid methods applied to a wide number of elliptic partial differential equations, see, e.g., [7, 11, 51, 102, 147]. We focus the remaining part of this section on properties for some classes of matrices working in the widest possible generality. For proving the convergence of the two–grid methods in the symmetric positive definite case other ways are indeed possible. One can consider, e.g., [137], in which a general class of stationary iterative methods is introduced for solving SPD systems, of which the two–grid is an example. In this respect, we just mention that for structured matrices, some simplifications of the theory can be obtained using a function approach, both in a Toeplitz setting [78, 114, 115] and in a GLT setting [97, 99, 256].

Multigrid preconditioners have been applied mostly in *elliptic* PDE settings, with both structured and unstructured grids. We will not focus explicitly on any of these applications, considering instead the common ground and making explicit the main points that are used in the various applications.

From another perspective, the multilevel method we have described in Algorithm 2.16, provides an *approximate inverse* of A, by suitably combining some *approximate inverses* of the hierarchy of matrices that represent A in increasingly coarser spaces from (2.52). Therefore, we can devise two distinct phases of our preconditioning routine, based on the general multigrid algorithm. They are a *setup phase*, summarized in Algorithm 2.17, and an *application phase* in which the preconditioner is applied to the residual vectors of our Krylov subspace method as $\mathbf{z} = M_l^{-1}\mathbf{r}$, see Sections 2.1.1 to 2.1.4.

For this second phase we can identify two basic approaches, i.e., the *additive* one and the *multiplicative* one:

additive case: at each level, the smoother and the coarse–space correction are applied to the restriction of the vector $\mathbf{r}$ to that level, and the resulting vectors are added;

multiplicative case: at each level, the smoother and the coarse–space correction are applied in turn, the former on a vector resulting

Algorithm 2.17: Setup phase of MGM preconditioner
Input: $A \in \mathbb{R}^{n\times n}$, Set of indices/grid Ω
1 $A_l \leftarrow A$, $\Omega_l \leftarrow \Omega$;
2 Set up $S_k^{(1)}$ and $S_k^{(2)}$;
3 **for** $k = l, l-1, \ldots, 1$ **do**
4 Generate Ω_{k-1} from Ω_k;
5 Define I_k^{k-1} and put $I_{k-1}^k = (I_k^{k-1})^T$;
6 Compute $A_{k-1} = I_k^{k-1} A_k I_{k-1}^k$;
7 Set up $S_{k-1}^{(1)}$ and $S_{k-1}^{(2)}$;
8 **end**

from the application of the latter and/or vice versa.

An example of multiplicative preconditioner, where the smoother (or possibly the smoothers) is applied before and after the coarse–space correction, is the symmetrized multiplicative preconditioner or, more generally, the V–cycle preconditioner summarized in Algorithm 2.18.

Application of this kind of preconditioners to the PDE setting are, e.g., in [79, 95, 216, 224, 234, 283] and many others, we refer again to the review [269] for some other cases where multigrid is a competitive option.

2.4 *Approximate Inverse Preconditioners*

The so–called *Approximate inverses* or *Approximate Inverse Preconditioners* are preconditioners approximating directly A^{-1} and not A, as usual. They have been intensively studied recently, in particular in the past twenty years; see the review [19] and, e.g., the more recent [36] and references therein. They do not require solving a linear system to be applied and often have very interesting parallel potentialities. On the other hand, they usually face a computational cost sensibly higher with respect to, e.g., incomplete LU techniques.

We recall that *incomplete LU* factorizations (ILU) are derived by performing Gaussian elimination incompletely on the matrix A of the system. A dropping strategy for specific entries in predetermined non-diagonal positions and/or whose moduli are smaller than a prescribed quantity is implemented; see [244, Sections 10.3 and 10.4] for further details on the construction, also in Section 2.4.2 few details on how this dropping strategies work are given in the context of inversion algorithms for the incomplete LU factors. Moreover, *existence* and *stability*

Algorithm 2.18: V–cycle preconditioner

Input: Vector $\mathbf{v} \in \mathbb{R}^{n\times n}$, MGM preconditioner from Algorithm 2.17

Output: Vector $\mathbf{w} = M_l^{-1}\mathbf{v}$

$\mathbf{v}_l \leftarrow \mathbf{v}$;

/* Recursively traverse levels downward: smoothing then residual restriction */

for $k = l, l-1, \ldots, 2$ **do**

  $\mathbf{y}_k \leftarrow S_k^{(1)}\mathbf{v}_k$, ($\nu_1$ times);

  $\mathbf{r}_k \leftarrow \mathbf{v}_k - A_k\mathbf{y}_k$;

  $\mathbf{v}_{k-1} \leftarrow I_k^{k-1}\mathbf{r}_k$;

end

$\mathbf{y}_0 \leftarrow S_0^{(1)}\mathbf{v}_0$, ($\nu_1$ times);

/* Recursively traverse levels upward: residual prolongation then smoothing */

for $k = 1, \ldots, l$ **do**

  $\mathbf{y}_k \leftarrow \mathbf{y}_k + I_{k-1}^{k}\mathbf{y}_{k-1}$;

  $\mathbf{r}_k \leftarrow \mathbf{v}_k - A_k\mathbf{y}_k$;

  $\mathbf{r}_k \leftarrow S_k^{(2)}\mathbf{r}_k$, ($\nu_2$ times);

  $\mathbf{y}_k \leftarrow \mathbf{y}_k + \mathbf{r}_k$;

end

$\mathbf{w} \leftarrow \mathbf{y}_l$;

of these algorithms is mostly based on having a matrix A of the linear system that is either an M–matrix or an H–matrix.

Definition 2.13 (M–matrix)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is called a non–singular M–matrix if*

1. $a_{i,i} > 0 \; \forall \, i = 1, 2, \ldots, n$;
2. $a_{i,j} \leq 0 \; \forall \, i \neq j, i, j = 1, 2, \ldots, n$;
3. $\det(A) \neq 0$;
4. *the entries of the inverse of A are positive.*

Definition 2.14 (H–matrix)**.** *Let $A = (a_{i,j}) \in \mathbb{R}^{n\times n}$, B such that $B = (b_{i,j}) \in \mathbb{R}^{n\times n}$, $b_{i,i} = a_{i,i} \; \forall \, i = 1, \ldots, n$ and $b_{i,j} = -|a_{i,j}|$ for $i \neq j$. A is an H–matrix if B is an M–matrix.*

However, ILU can have very poor performances for important applications, where A is not an M– or an H–matrix and/or that require

a parallel implementation, since they require, at each step, the solution of two triangular systems, that is a recognized serial bottleneck.

One possible remedy is to try to find a preconditioner that does not require solving a linear system. For example, the original system can be preconditioned by a matrix M which is a direct approximation to the inverse of A, thus requiring just matrix-vector multiplications for its application.

There exist several completely different algorithms for computing a sparse approximate inverse, with each approach having its own strengths and limitations.

Usually, two main basic types of approximate inverses exist, depending on whether the preconditioner M, approximating A^{-1}, is expressed as a single matrix or as a product of two or more matrices. The latter type of preconditioners are known as *factorized sparse approximate inverses* and they are of the form $M = W D^{-1} Z^T$ or $M = W Z^T$, where Z and W are lower triangular matrices and D is diagonal, assuming that M admits a LU factorization. Within each class, there are several different techniques, depending on the algorithm used to compute the approximate inverse or approximate inverse factors. At present, there are three approaches that are more used:

- *Residual norm minimization* [163–165],
- *inversion and sparsification of an ILU* [104], and
- *incomplete bi-conjugation* [21, 22, 24].

However, before discussing the last two in the way they have been modified in [36], some results regarding the entries of the inverse of a given matrix are essential.

2.4.1 On the Decay of the Entries of A^{-1}

At the beginning of the discussion on preconditioners, in Theorem 2.1 it was observed that the inverse A^{-1} of a sparse matrix A has often significantly more entries than A. Nevertheless, under suitable conditions, the magnitude of the elements of the inverse can play a fundamental role. Prior to analyzing this problem, consider an example that guides us.

Example 2.1. *Consider the tridiagonal symmetric positive definite matrix*

$$A = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}_{n\times n}.$$

Instead of representing A^{-1} with its pattern, we observe its cityplot; see Figure 2.3. Even if the elements of the inverse are all different from zero, note

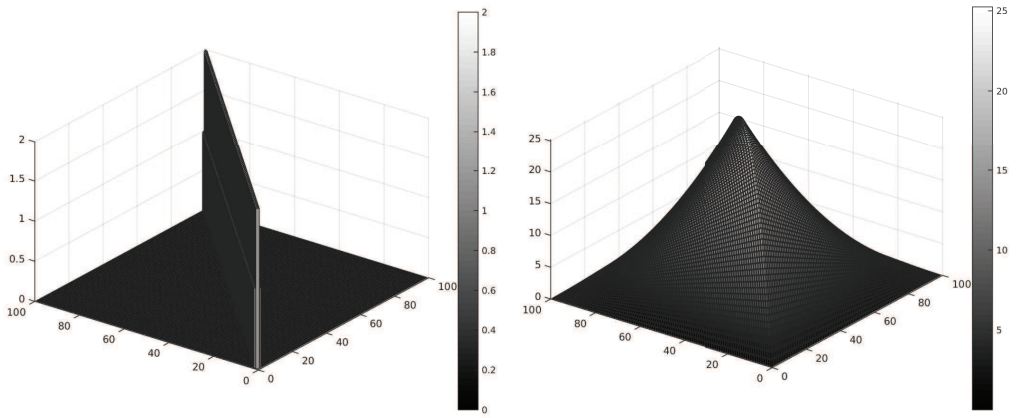


Figure 2.3. Example 2.1. Cityplots of the A matrix (on the left) and of A^{-1} on the right.

the presence of a decay, i.e., a decreasing of their absolute value away from the main diagonal of the matrix.

Let us start investigating this decay in the inverse matrices in order to use it, whenever it exists, for preconditioning strategies.

The first result that can help is due to Demko, Moss and Smith.

Theorem 2.15 (Demko, Moss, and Smith [89])**.** *Let A and A^{-1} be in $\mathcal{B}(l^2(S))$ (that is defined just after the present statement). If A is positive definite and m-banded then we have:*

$$(|A^{-1}|)_{i,j=1}^{n} = |b_{i,j}| \le C\lambda^{|i-j|},$$

where:

$$\lambda = \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^{2/m},$$

and

$$C = \|A^{-1}\| \max\left\{1, \frac{\left(1+\sqrt{\kappa(A)}\right)^2}{2\kappa(A)}\right\}.$$

If A fails to be positive definite but is still m-banded, quasi-centered, bounded, and boundedly invertible then:

$$(|A^{-1}|)_{i,j=1}^{n} \leq C_1 \lambda_1^{|i-j|},$$

where

$$\lambda_1 = \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^{\frac{1}{m}},$$

and

$$C_1 = (m+1)\lambda_1^{-m}\|A^{-1}\|\kappa(A)\max\left\{1, \frac{1}{2}\left[\frac{1+\kappa(A)}{\kappa(A)}\right]^2\right\}.$$

To prove this statement some preliminary work is needed. Let us start with a general complex, separable, Hilbert space H, and let $\mathscr{B}(H)$ denote the Banach algebra of all linear operators on H that are also bounded. Now if $A \in \mathscr{B}(H)$ then A can be represented as a matrix with respect to any complete orthonormal set. In this way, A can be regarded as an element of $B(l^2(S))$, where $S = \{1, 2, \ldots, N\}$. In this space, the usual matrix product defines the action A over the space. A can be considered as a matrix representing a bounded operator in $\mathscr{B}(l^2(S))$. Recall that A is *m-banded* if there is an index l such that

$$a_{i,j} = 0, \quad \text{if } j \notin [i-l, i-l+m].$$

Then A is said to be *centered* and *m-banded* if m is even and the l above can be chosen to be $m/2$. In this case the zero elements of the *centered* and *m-banded* are:

$$a_{i,j} = 0, \quad \text{if } |i-j| > \frac{m}{2}.$$

Remark 2.7. *Selfadjoint matrices are naturally centered, i.e., a tridiagonal selfadjoint matrix is centered and 2-banded.*

Now let $\mathbb{R}_n$ denote, as usual, the polynomial of degree less than or equal to n. If $K \subseteq \mathbb{C}$ and f is a fixed complex–valued bounded function on K, then we define the norm

$$\|f\|_K = \sup_{z \in K} |f(z)|$$

and the relative approximation error for the set of polynomial $\mathbb{R}_n$ to an f over the set K as

$$e_n(K) = \inf_{p \in \mathbb{R}_n} \|f - p\|_K.$$

To proceed, a result due to Tchebychev [271], and Bernstein [26] is needed; see [202] for a modern presentation.

Lemma 2.1. *Let $f(x) = 1/x$ and let $0 < a < b$. Set $r = b/a$ and:*

$$q = q(r) = \frac{\sqrt{r} - 1}{\sqrt{r} + 1}$$

then:

$$e_n([a, b]) = \frac{(1 + \sqrt{r})^2}{2ar} q^{n+1}$$

Proposition 2.10 (Demko, Moss, and Smith [89])**.** *Let A be a positive definite, m-banded, bounded and boundedly invertible matrix in $l^2(S)$. Let $[a, b]$ be the smallest interval containing $\sigma(A)$. Set $r = b/a$, $q = q(r)$ as in the Lemma 2.1, and set $C_0 = (1 + \sqrt{r})^2/(2ar)$ and $\lambda = q^{2/m}$. Then,*

$$|A^{-1}| = (|b_{i,j}|)_{i,j=1^n} \leq C\lambda^{|i-j|}$$

where:

$$C = C(a, r) = \max\{a^{-1}, C_0\}.$$

Now following the authors of the result above, we report an extension of the latter for a more generic type of matrix A. Before doing this, the *quasi–centered* matrix definition is needed. A is said to be *quasi–centered* if the central diagonal is contained within the nonzero bands of the matrix, i.e., $A \in \mathcal{B}(l^2(S))$ is invertible only if A is quasi–centered. Note also that this is not true for $A \in l^2(Z)$.

Proposition 2.11 (Demko, Moss, and Smith [89])**.** *Let A be m-banded, bounded and boundedly invertible on $l^2(S)$. Let $[a, b]$ be the smallest interval containing $\sigma(AA^H)$. Then, setting $r = b/a$, $q = q(r)$ as in Lemma 2.1, and $\lambda_1 = q^{1/m}$, there is a constant C_1 depending on A so that*

$$(|A^{-1}|)_{i,j=1}^{n} = |b_{i,j}| \leq C_1 \lambda_1^{|i-j|}.$$

If A is quasi–centered, then C_1 can be chosen as $C_1 = (m+1)\|A\|\lambda_1^{-m} C(a, r)$.

The proof of Theorem 2.15 is given by two previous propositions. Observe that Theorem 2.15 does not apply to the sequence of matrices from Example 2.1, since their condition number grows like $O(n^2)$, i.e.,

the spectral condition number is unbounded. Being familiar with the physics of the problem, this has to be expected, since we are trying to approximate the Green's function for d^2/dx^2. Therefore, even if the matrix satisfies a bound of the kind of Theorem 2.15, it deteriorates as $n \to +\infty$.

In many cases understanding the behavior of the decay is important not only from the main diagonal but also when more complicated patterns appear. A useful result in this sense is given in Canuto, Simoncini, and Verani [71] for Kronecker sums of symmetric and positive definite banded matrices, i.e.,

$$S = I_n \otimes M + M \otimes I_n,$$

where I_n is the identity matrix in $\mathbb{R}^{n\times n}$, a structure very frequently encountered when dealing with the discretizations of PDEs on a Cartesian grids, see, e.g., [186], or other tensor product structures.

Theorem 2.16 (Canuto, Simoncini, and Verani [71]). *Let $M \in \mathbb{R}^{n\times n}$ be a symmetric and positive definite matrix of bandwidth b. For $k, t \in \{1, 2, \dots, n^2\}$, let*

$$j = \lfloor {}^{t-1}/_{n} \rfloor + 1, \quad i = t - n\lfloor {}^{t-1}/_{n} \rfloor,$$

and l, m such that

$$m = \lfloor {}^{k-1}/_{n} \rfloor + 1, \quad l = k - n\lfloor {}^{k-1}/_{n} \rfloor.$$

If λ_{min} and λ_{max} are the extreme eigenvalues of M, and $\lambda_1 = \lambda_{min} + \imath\omega$, $\lambda_2 = \lambda_{max} + \imath\omega$,$R = \alpha + \sqrt{\alpha^2 - 1}$ with $\alpha = {}^{|\lambda_1|+|\lambda_2|}/_{\lambda_2-\lambda_1}$ and $\beta = |\lambda_{max} - \lambda_{min}|$, then the following results hold

1. *If $i \neq l$ and $j \neq m$, then*

$$\left|\left(S^{-1}\right)_{k,t}\right| \leq \frac{1}{2\pi}\frac{64}{\beta^2}\int_{-\infty}^{+\infty}\left(\frac{R^2}{(R^2-1)^2}\right)^2\left(\frac{1}{R}\right)^{\frac{|i-l|}{b}+\frac{|j-m|}{b}-2} d\omega;$$

2. *If either $i = l$ or $j = m$, then*

$$\left|\left(S^{-1}\right)_{k,t}\right| \leq \frac{1}{2\pi}\frac{8}{\beta}\int_{-\infty}^{+\infty}\frac{R^2}{(R^2-1)^2\sqrt{\lambda_{min}^2+\omega^2}}\left(\frac{1}{R}\right)^{\frac{|i-l|}{b}+\frac{|j-m|}{b}-1} d\omega;$$

3. *If both $i = l$ and $j = m$, then*

$$\left|\left(S^{-1}\right)_{k,t}\right| \leq \frac{1}{2\pi}\int_{-\infty}^{+\infty}\frac{1}{\lambda_{min}^2+\omega^2} d\omega = \frac{1}{2\lambda_{min}}.$$

Another result that is very useful and that is pivotal for several results in both Chapters 5 and 8 is the following.

Theorem 2.17 (Jaffard [166])**.** *The sets of invertible matrix* $(A)_{h,k} \in \mathcal{B}(l^2(\mathbb{K}))$, $\mathbb{K} = \mathbb{Z}, \mathbb{N}$, *such that either*

$$|a_{h,k}| \leq C(1 + |h - k|)^{-s}, \tag{2.57}$$

or

$$|a_{h,k}| \leq C \exp(-\gamma|h - k|) \tag{2.58}$$

are algebras, denoted by Q_s *and* $\mathcal{E}_\gamma$*, respectively.*

Observe that this result does not require any particular sparsity pattern for the underlying matrix, and is based only on the values assumed by the entries.

Remark 2.8. *In Theorem 2.17 we used the hypothesis* $A \in \mathcal{B}(l^2(\mathbb{K}))$*. We stress that the fact that the single matrix A of finite dimension is invertible does not ensure that the operator over* $l^2(\mathbb{K})$ *is invertible with a bounded inverse. This kind of feature is not easily proven in general and require the use of tools from C*–algebras.*

Decay conditions for general matrices remain an open problem.

Not all of the entries of such inverses, though nonzero, have the same weight. The same principle adopted for the ILU factorizations in [244] can be applied when investigating the strategies that generate the approximate inverses with some degree of dropping.

2.4.2 Inversion and Sparsification or INVS

The inversion and sparsification of an existing *ILU* decomposition is a strategy used often to get approximate inverse preconditioners. Three immediate positive aspects are that the latter is produced in factorized form: the availability of reliable packages producing ILU factorizations, and the inherent parallel potentialities after the *ILU* decomposition; see, e.g., van Duin [104] and [36].

Let us focus on the approach by van Duin [104] as reconsidered in [36] and here called *INVS* (from INVersion and Sparsification) for brevity. The strategy is based on performing a sparse inversion technique on the triangular factors of an existing incomplete factorization in the form $M = LDU$, where D is a diagonal matrix and L and U are lower and upper triangular with ones on the main diagonal, respectively. The latter factorization can be obtained with a slight modification of the classical *ILU* techniques, see, e.g., [108, 189, 244, 246]: let $M = L\,U_1$

be a *ILU* preconditioner and let D be the main diagonal of U_1. D is nonsingular otherwise M is a singular preconditioner. Then, by posing $U = D^{-1}U_1$, we deduce the following

$$M = L\,D\,U \rightarrow M^{-1} = U^{-1}D^{-1}L^{-1} = WD^{-1}Z^T$$

are the factorizations for M and for its inverse. However, as proved in Theorem 2.1, Z and W can be dense lower triangular matrices. In order to get a sparse incomplete factorization for A^{-1} and therefore of M^{-1}, a sparsification process for Z and W can be based on threshold and position or both, similar to what is seen for *ILU* decompositions, generating the sparse lower triangular matrices $\tilde{Z}$ and $\tilde{W}$. After having obtained sparse approximations $\tilde{Z}$, $\tilde{W}^T$ for the matrices L^{-1} and U^{-1}, use them to get the *explicit* preconditioner[5] for A^{-1} of the form

$$\tilde{M}^{-1} = \tilde{U}^{-1}D^{-1}\tilde{L}^{-1} = \tilde{W}D^{-1}\tilde{Z}^T.$$

We call it the *INVS*.

To produce the underlying inversion, start writing U as[6]

$$U = I + \sum_{i=1}^{n-1} \mathbf{e}_i \mathbf{u}_i^T.$$

By observing that $\forall\, j \leq k$, we find $\mathbf{e}_k \mathbf{u}_k^T \mathbf{e}_j \mathbf{u}_j^T = 0$, since the jth entry of $\mathbf{u}_k$ is zero $\forall\, j \leq k$, rewrite U as

$$U = \prod_{i=n-1}^{1} \left(I + \mathbf{e}_i \mathbf{u}_i^T\right). \tag{2.59}$$

The inverse of the factors in (2.59) is straightforward[7]:

$$\left(I + \mathbf{e}_i \mathbf{u}_i^T\right)^{-1} = I - \mathbf{e}_i \mathbf{u}_i^T,$$

5 Explicit preconditioner: no linear systems should be solved to apply it but, if used with a Krylov subspace method, then just matrix-vector multiplications with $\tilde{Z}$, $\tilde{W}^T$ and D^{-1} are required.

6 As usual we are using $\mathbf{e}_i$ notation for the vectors of the canonical basis, while $\mathbf{u}_i$ is the ith row of the matrix U with the element $u_i(j) = 0$ for $j \leq i$.

7 The inverse of the factors in (2.59) can be computed directly by the fact that U is upper triangular, and using again that $\forall\, j \leq k$, we find $\mathbf{e}_k \mathbf{u}_k^T \mathbf{e}_j \mathbf{u}_j^T = 0$, which gives the needed result: $(I - \mathbf{e}_i \mathbf{u}_i^T)(I + \mathbf{e}_i \mathbf{u}_i^T) = I$. Otherwise, it can be computed by a straightforward application of the Sherman–Morrison formula for the inversion of $(A + \mathbf{u}\mathbf{v}^T)$; see Sherman and Morrison in [259] for details.

and then

$$U^{-1} = \prod_{i=1}^{n-1} \left(I - \mathbf{e}_i \mathbf{u}_i^T\right). \tag{2.60}$$

Now, since U^{-1} is also an upper triangular matrix, the above expression can be rewritten as a sum

$$U^{-1} = I + \sum_{i=1}^{n-1} \mathbf{e}_i \hat{\mathbf{u}}_i^T, \tag{2.61}$$

where $\hat{\mathbf{u}}_i^T$, the strict upper triangular part of the ith row of U^{-1}, is obtained as

$$\hat{\mathbf{u}}_i^T = -\mathbf{u}_i^T \prod_{j=i+1}^{n-1} \left(I - \mathbf{e}_i \mathbf{u}_i^T\right). \tag{2.62}$$

The expression for L^{-1} can be obtained similarly.

Remark 2.9. *From* (2.62) *we observe that no* $\hat{\mathbf{u}}_j$ *is needed for the calculation of* $\hat{\mathbf{u}}_i$ *for* $i \neq j$*, so the whole inversion process can be executed in parallel on a distributed memory machine, i.e., computer systems with many processors in which each processor has its own private memory.*

Algorithm 2.19: Sparse product algorithm.
Input: $U \in \mathbb{R}^{n\times n}$ strict upper triangular matrix

```
for i = 1, ..., n − 1 do
    û_i^T ← −u_i^T;
    j ← first nonzero position in û_i^T;
    while j < n do
        α ← −û_i^T e_j;
        û_i^T = û_i^T + α u_j^T ;          // As a sparse operation.
        j ← next nonzero position in û_i^T;
    end
end
```

A straightforward implementation of the formula (2.62) is in the Algorithm 2.19, that has the flaw of generating dense matrices, recall Theorem 2.1. To sparsify using some dropping strategy, similarly to what is done usually for the ILU factorization, see the modifications needed in Algorithm 2.19.

Pattern drop. A fixed pattern S for the matrix is given, so $\hat{\mathbf{u}}_i^T(k)$ is only calculated when $(i,k) \in S$.

Neumann drop. [8] We start from rewriting formula (2.60), namely the Neumann series expansion for the formula (2.59):

$$\begin{aligned} U^{-1} =& I - \sum_{j_1=1}^{n-1} \mathbf{e}_{j_i} \mathbf{u}_{j_1}^T + \sum_{j_2=1}^{n-2} \left(\mathbf{e}_{j_2} \mathbf{u}_{j_2}^T \sum_{j_1=j_2+1}^{n-1} \mathbf{e}_{j_1} \mathbf{u}_{j,1}^T \right) + \\ & - \sum_{j_3=1}^{n-3} \left(\mathbf{e}_{j_3} \mathbf{u}_{j_3}^T \sum_{j_2=j_3+1}^{n-2} \left(\mathbf{e}_{j_2} \mathbf{u}_{j_2}^T \sum_{j_1=j_2+1}^{n-1} \mathbf{e}_{j_1} \mathbf{u}_{j,1}^T \right) \right) + \dots \end{aligned} \tag{2.63}$$

by truncating this expression at a number of extra term m we obtain the dropping $\hat{U}_m$. The main issue of this approach is that the update $\mathbf{u}_k^T$ can be computed m times in the worst case for $\hat{\mathbf{u}}_i^T$.

Positional fill level. Similarly to $ILU(P)$, define a level of fill initialized for U as:

$$\text{lev}_{i,j} = \begin{cases} 0 & \text{if } \mathbf{u}_i^T(j) \neq 0, \\ +\infty & \text{if } \mathbf{u}_i^T(j) = 0, \end{cases}$$

and the function to update the levels of fill is

$$\text{lev}_{i,k} = \min(\text{lev}_{i,j} + 1, \text{lev}_{i,k}).$$

In this way, Algorithm 2.19 becomes Algorithm 2.20.

Positional fill level II. Instead of using the level of fill of the approximate inverse matrix, we can choose the level of fill of the original sparse triangular factor. This choice changes only the initialization step in Algorithm 2.20:

$$\text{lev}_{i,j} = \begin{cases} \text{lev}_{i,j}^U & \text{if } \mathbf{u}_i^T(j) \neq 0, \\ +\infty & \text{if } \mathbf{u}_i^T(j) = 0. \end{cases}$$

Threshold drop. Algorithm 2.19 can be implemented with the same elementary operators as the incomplete LU factorization with thresholding from [246], and quoting from [36]:

- in the copy–in phase, Step 2, we initialize the set of nonzero entries for the current row $\hat{\mathbf{u}}_i$;
- in the update phase in Step 7 we also insert the relevant indices into the set to ensure that the retrieval of the next nonzero at Step 11 is performed efficiently;

8 Note that the first two terms are available without cost.

Algorithm 2.20: Positional fill level inversion of a sparse triangular matrix or *INVK*

Input: $U \in \mathbb{R}^{n\times n}$ strict upper triangular matrix, initial pattern of the matrix $\text{lev}_{i,j}$.

```
for j = 1, ..., n − 1 do
    û_i^T ← −u_i^T;
    j ← first nonzero position in û_i^T;
    while j < n do
        if lev_{i,j} ≤ p then
            α ← −û_i^T e_j;
            û_i^T ← û_i^T + α û_j^T;
            lev_{i,k} = min(lev_{i,j} +1, lev_{i,k});
        else
            û_i^T(j) ← 0;
        end
        j ← next nonzero position in û_i^T;
    end
end
```

- at the end of the inner loop, we perform a copy–out operation bringing the row $\hat{\mathbf{u}}_i$ into its desired final state, copying the largest entries up to the maximum allowed number of nonzeros.

From the above discussion, in Algorithm 2.21 we are looking again to implement efficiently the following two operations:

1. select and remove the lowest ranked element from a set;
2. add an element to the set.

As highlighted in [36] this can be achieved efficiently on a set with an order relation. This item can be implemented by means of a *Partially Ordered Set Abstract Data Type S*; see [162]. With this kind of data structure both the insertion of a new element and the deletion of the lowest ranked element (where the rank is given within respect to the chosen order) can be performed with a cost of $O(\log(|S|))$, where $|S|$ is, as usual, the cardinality of the set S. Observe that the copy–out operations in both factorization and inversion can be implemented by making again use of a partially ordered set, or by keeping the p largest entries in the current row.

A parallel implementation, exploiting the GPU architecture of this algorithm, is proposed in [36].

Algorithm 2.21: Inversion of triangular matrices with numerical drop or *INVT*

Input: $U \in \mathbb{R}^{n\times n}$ strict upper triangular matrix

1 **for** $j = 1, \ldots, n-1$ **do**
2 $\quad \hat{\mathbf{u}}_i^T \leftarrow -u_i^T$;
3 $\quad j$ location of first nonzero in $\hat{\mathbf{u}}_i^T$;
4 $\quad$ **while** $j < n$ **do**
5 $\quad\quad \alpha \leftarrow -\hat{\mathbf{u}}_i^T e_j = -\hat{\mathbf{u}}_i^T(j)$;
6 $\quad\quad$ **if** $|\alpha| > \epsilon$ **then**
7 $\quad\quad\quad \hat{\mathbf{u}}_i^T \leftarrow \hat{\mathbf{u}}_i^T + \alpha u_j^T$;
8 $\quad\quad$ **else**
9 $\quad\quad\quad \hat{\mathbf{u}}_i^T(j) \leftarrow 0$;
10 $\quad\quad$ **end**
11 $\quad\quad j$ location of next nonzero in $\hat{\mathbf{u}}_i^T$;
12 $\quad$ **end**
13 $\quad$ Drop elements in $\hat{\mathbf{u}}_i$ as necessary to achieve the desired number of nonzeros.;
14 **end**

With the appropriate implementation for this data structure, we are now in a position to report the estimate of the cost of building an approximate inverse as in Algorithm 2.21 that was obtained in [36].

Theorem 2.18 (Bertaccini and Filippone [36])**.** *Let* $\text{nnz}_{\mathbf{u}}$ *be the average number of nonzeros per row for* $\mathbf{u}$, $\text{nnz}_{\hat{\mathbf{u}}}$ *for* $\hat{\mathbf{u}}$ *and that the bounds*

$$|S| \leq \gamma\, \text{nnz}_{\mathbf{u}}, \tag{2.64}$$

$$\text{nnz}_{\hat{\mathbf{u}}} \leq \beta\, \text{nnz}_{\mathbf{u}}, \tag{2.65}$$

hold true, where $|S|$ *is the maximum size of the set of the nonzero entries in any of the* $\hat{\mathbf{u}}_i$ *before the application of the drop rule at Step 13. Then, the computational cost of Algorithm 2.21 is given by*

$$O(\gamma\beta n \cdot \text{nnz}_{\mathbf{u}}^2(1 + \log(\gamma\, \text{nnz}_{\mathbf{u}}))). \tag{2.66}$$

The above result relies on two crucial assumptions about the size of both the constant β and γ: we need small constants to achieve a good asymptotical cost. Assumption (2.65) is the easiest to justify: from the discussion at the beginning of Section 2.4, we know that we desire preconditioners with a number of nonzeros of the same order as

the coefficient matrix A, hence $\beta \approx 1$. Observe that we can enforce the number of nonzeros at Step 13, thus the assumption above is reasonable.

On the other hand, assumption (2.64) is a bit more complex: it relies on the behavior of the profile of $\mathbf{u}$ and $\hat{\mathbf{u}}$. We can consider hypothesis (2.64) plausible whenever we are in presence of the conditions discussed in Section 2.4.1. Since the latter is indeed equivalent to an exponential decaying argument for the entries of the inverse of the Cholesky factor; see also the applications in Sections 4.2 and 5.2.1. We stress that also enforcing the dropping rules at 6 and 13 in Algorithm 2.21 helps in keeping $|S|$ under control.

The cost of INVK and INVT has also been analyzed in the original paper [104], obtaining for approximate inversion of the upper factor the estimate

$$C_{\text{invrt}} = O\left(\text{nnz}_{\hat{U}} \, \frac{\text{nnz}_U}{n}\right),$$

where nnz_U is the number of nonzeros above the main diagonal in U and likewise $\text{nnz}_{\hat{U}}$ for the sparsified version $\hat{U}$.

In [36], was noted that the upper bound for the first term $\text{nnz}_{\hat{U}}$ is given by the product $n\beta\text{nnz}_{\mathbf{u}}$ while the second term is $\text{nnz}_{\mathbf{u}}$. This estimate is then equivalent to the reported one in (2.66), under the mild assumption that $\log(\gamma \, \text{nnz}_{\mathbf{u}})$ is bounded by a small constant.

2.4.3 Incomplete Biconjugation: AINV

Let us focus on another approach for preconditioning, based on efficient approximations to the inverse of the underlying matrix. In particular, approaches that do not necessitate apriori information on the sparsity pattern of A^{-1} are considered here. This kind of procedure have been developed in different forms in many papers; see, e.g., [22, 60, 61, 170].

Factorized Approximate Inverse preconditioners for general sparse matrices can be efficiently constructed by means of a generalization of the Gram–Schmidt process known as *biconjugation*.

An *approximate inverse preconditioner in factorized form* (*AINV*), was proposed by Benzi et al. in 1996, see [22] and later extended in [24] and in [21]. The main idea comes from an algorithm first proposed in 1948 in [119], a variation of the root–free Cholesky decomposition of A.

AINV strategy is based on the observation that if a matrix $A \in \mathbb{R}^{n\times n}$ is nonsingular, and if two vector sequences $\{\mathbf{z}_i, i = 1 \ldots n\}$ and $\{\mathbf{w}_i, i = 1 \ldots n\}$ A–biconjugate are given, i.e., $\mathbf{z}_i^T A \mathbf{w}_j = 0$ if and only if $i \neq j$,

then we can express a biconjugation relation as follows:

$$Z^T A W = D = \begin{pmatrix} p_1 & 0 & \dots & 0 \\ 0 & p_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & p_n \end{pmatrix} \tag{2.67}$$

where $p_i = \mathbf{z}_i^T A \mathbf{w}_i \neq 0$. Thus, W and Z must be nonsingular, since D is nonsingular. Therefore, in matrix form,

$$A = Z^{-T} D W^{-1}$$

from which it readily follows that

$$A^{-1} = W D^{-1} Z^T. \tag{2.68}$$

If W and Z are triangular, then they are the inverses of the triangular factors in the familiar LDU decomposition of A (see, e.g., [134]), as can be easily seen by comparing the two expressions

$$A = LDU, \text{ and } A = Z^{-T} D W^{-1}.$$

Algorithm 2.22: Biconjugation

$w_i^{(0)} \leftarrow z_i^{(0)} \leftarrow e_i \qquad 1 \leq i \leq n;$
for $i = 1, \dots, n$ **do**
  for $j = i, i+1, \dots, n$ **do**
    $p_j^{(i-1)} \leftarrow \mathbf{a}_{i,:}^T \mathbf{w}_j^{(i-1)}; \qquad q_j^{(i-1)} \leftarrow \mathbf{a}_{:,i}^T \mathbf{z}_j^{(i-1)};$
  end
  for $j = i+1, \dots, n$ **do**
    $\mathbf{w}_j^{(i)} \leftarrow \mathbf{z}_j^{(i-1)} - \left(\dfrac{p_j^{(i-1)}}{p_i^{(i-1)}} \right) \mathbf{w}_i^{(i-1)};$
    $\mathbf{z}_j^{(i)} \leftarrow \mathbf{z}_j^{(i-1)} - \left(\dfrac{q_j^{(i-1)}}{q_i^{(i-1)}} \right) \mathbf{z}_i^{(i-1)};$
  end
end
$\mathbf{z}_i \leftarrow \mathbf{z}_i^{(i-1)}, \ \mathbf{w}_i \leftarrow \mathbf{w}_i^{(i-1)}, \ \mathbf{p}_i \leftarrow \mathbf{p}_i^{(i-1)}, \ 1 \leq i \leq n;$

Observe that there are infinitely many biconjugate sequences $\{\mathbf{w}\}$ and $\{\mathbf{z}\}$ satisfying the above relations. To find one of them, it is enough

to apply a biconjugation procedure to an appropriate pair of nonsingular matrices $W^{(0)}, Z^{(0)} \in \mathbb{R}^{n\times n}$. From a computational point of view, one can start with $W^{(0)} = Z^{(0)} = I$, thus obtaining Algorithm 2.22, where $\mathbf{a}_{i,:}^T$ is the ith row of A and $\mathbf{a}_{:,i}^T$ is the ith column of A, i.e., the ith row of A^T. If the procedure reaches completion without breakdowns, i.e., all the diagonal entries are nonzero, then the resulting matrices W and Z will be triangular. Thus, for symmetric positive definite matrices, the process does not break down. Another interesting feature of Algorithm 2.22 is that the process for building W can proceed *independently* of Z.

To turn Algorithm 2.22 into a practical approximation procedure, and therefore for a possible preconditioner, we need to "sparsify" the resulting W and Z by dropping elements in the vectors $\mathbf{w}_i$ and $\mathbf{z}_i$. In principle this could be done at the end of Algorithm 2.22, but this would mean storing the (dense) matrices W and Z until the end. In practice, the sparsification is done at each update for the vectors $\mathbf{w}$ and $\mathbf{z}$.

Similar to the case of the incomplete factorization of ILU type, it is possible to prove that the incomplete inverse factorization exists (in exact arithmetic) when A is an H–matrix; see [22].

Proposition 2.12 (Benzi, Meyer, and Tüma [22])**.** *Let A be an H–matrix and let $\hat{A}$ be the associated M–matrix. If p_i and $\hat{p}_i$ denote the pivots computed by the inverse factorization Algorithm 2.22 applied to A and to $\hat{A}$, respectively, then $p_i \geq \hat{p}_i$. Furthermore, if $\hat{p}_i$ denote the pivots computed by the incomplete inverse factorization algorithm applied to A, then $\overline{p}_i \geq \hat{p}_i$.*

We stress that, despite the many similarities, there is a noticeable difference with the case of incomplete factorizations. It is well known that if A is an M-matrix, then the incomplete factorization induces a regular splitting $A = \hat{L}\hat{U} - R$, i.e., $\rho(I - \hat{U}^{-1}\hat{L}^{-1}A) < 1$, while this is not necessarily true for the incomplete inverse factors produced by biconjugation; see [24].

Example 2.2. *Consider the symmetric matrix HB/bcsstk07 from the Harwell–Boeing collection [103]. Producing $M^{-1} = ZD^{-1}Z^T$ with a drop tolerance of $\varepsilon = 0.1$. The estimated spectral radius for this splitting is $\rho(I - M^{-1}A) = 1.44 > 1$, and so the splitting is* not *convergent.*

In theory, AINV can suffer of breakdown when the coefficient matrix is not an H–matrix. But the process as modified in [21] will not break down for symmetric and positive definite matrices. The modified method was called *Stabilized AINV*, or *SAINV*.

The procedure in Algorithm 2.22 is a *right looking* variant, i.e., when a vector $\mathbf{z}_i$ is finalized, it is used to update all the vectors $\mathbf{z}_j$, $j > i$.

An alternative formulation is the *left looking* variant as suggested in [36], i.e., all the updates to $\mathbf{z}_i$ involving $\mathbf{z}_j$, $j < i$, are performed in a single iteration of the outer loop. We show the procedure for Z in Algorithm 2.23, W can be handled in the same way. As usual, in exact arithmetic, the numerical behavior of the two algorithms is the same. Nevertheless, the distribution of the computational work in the two variants is indeed different. We observe that the left–looking variant groups together all the updates to a given column. We perform more (sparse) dot products, using the "true" $\mathbf{z}_i$, i.e., before sparsification.

As observed in [36], these features can be beneficial from a numerical point of view, since:

1. The dot products at 5 and 8 in Algorithm 2.23 are computed with the full vector $\mathbf{z}_i$, before the application of the drop tolerance.
2. The dropping rule on $\mathbf{z}_i$ entries is applied only at the end of the update loop, whereas in the right-looking version is applied at each update.

From the experiments in [36], the left–looking variant seems to suffer less from pivot breakdown.

Algorithm 2.23: Left Looking Biconjugation for Z

1 $\mathbf{z}_1^{(0)} \leftarrow \mathbf{e}_1$; $\quad p_1^{(0)} \leftarrow a_{1,1}$;
2 **for** $i = 2, \ldots, n$ **do**
3 $\quad \mathbf{z}_i^{(0)} \leftarrow \mathbf{e}_i$;
4 $\quad$ **for** $j = 1, \ldots, i-1$ **do**
5 $\quad\quad p_i^{(j-1)} \leftarrow \mathbf{a}_{j,:}^T \mathbf{z}_i^{(j-1)}$;
6 $\quad\quad \mathbf{z}_i^{(j)} \leftarrow \mathbf{z}_i^{(j-1)} - \left(\dfrac{p_i^{(j-1)}}{p_j^{(j-1)}}\right) \mathbf{z}_j^{(j-1)}$;
7 $\quad$ **end**
8 $\quad p_i^{(i-1)} \leftarrow \mathbf{a}_{i,:}^T \mathbf{z}_i^{(i-1)}$;
9 **end**

Recall that even if A is sparse, then there is no guarantee that Z is sparse too; see again the discussions made in Section 2.4.1 about the decay of the entries of the inverse of a matrix.

Example 2.3. *As an example of the fill–in for the Z matrix in the AINV Algorithm, we consider the application to the HB/sherman1*[9] *matrix* (2.4) *of the Harwell-Boeing Collection for various drop tolerances.*

9 Information concerning this matrix can be found in [86].

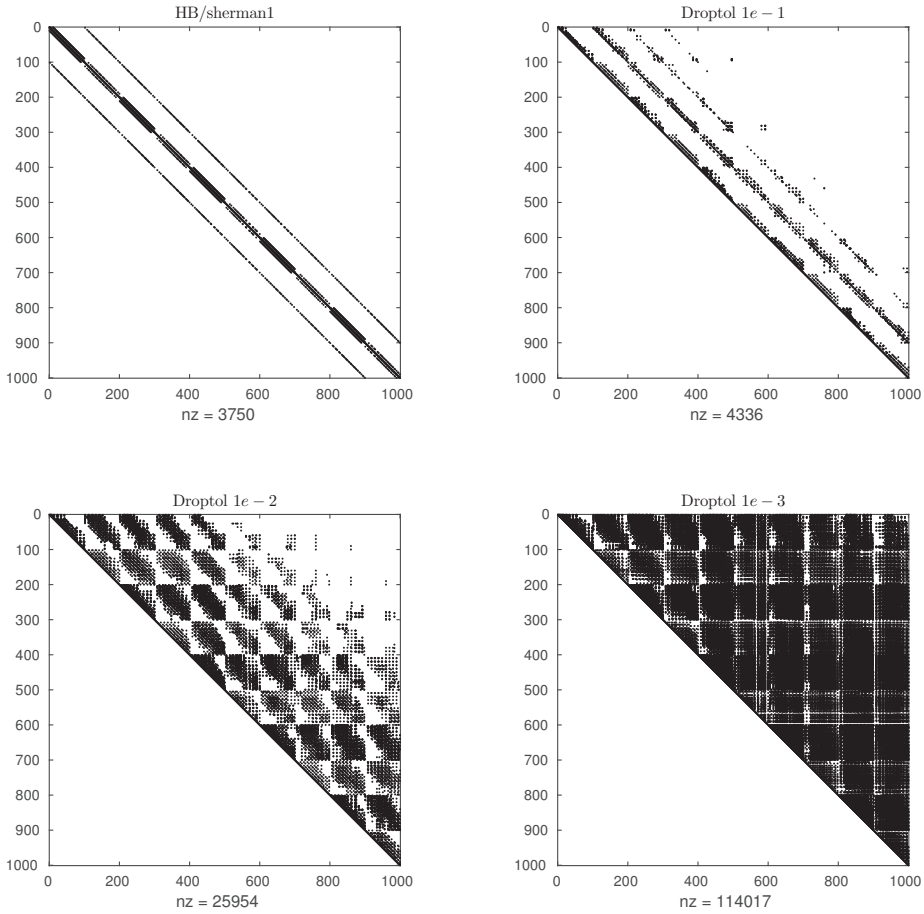


Figure 2.4. $AINV(A, \varepsilon)$ for the HB/sherman1 matrix at various ε.

Consider again Algorithm 2.23. In an actual implementation, the vector $\mathbf{z}_i$ could be stored in full format during the execution of Loop 4, and there can be two different ways to apply the dropping rule:

1. at Statement 6 the update of $\mathbf{z}_i$ is only performed for a sufficiently large value of p_i/p_j;
2. after the Statement 8 a dropping rule is applied to $\mathbf{z}_i$ thereby sparsifying it for final storage.

Observe that the application of the first dropping rule based on p_i/p_j was discouraged in the paper that introduces right–looking AINV for symmetric systems [22], while in the experiments in [36] with the left–looking approach, the dropping rule was applied, without adverse numerical effects. Moreover, it seems to provide a performance advantage. Observe that in the dropping rule applied to $\mathbf{z}_i$, both the usual threshold comparisons and the limitation on the maximum

number of nonzeros where allowed in [36].

A key observation made in [36] is that the execution of Statement 5 in Algorithm 2.23 computes the dot product between $\mathbf{a}_{j,:}$ and $\mathbf{z}_i$ even if in most cases this product can be exactly zero because of the (mis)match between the position of nonzeros in $\mathbf{a}_{j,:}$ and $\mathbf{z}_i$. Thus, we are performing quadratic cost operations without any contribution to the result. To avoid this, in [36], they propose executing Loop 4 only when is necessary. This is equivalent to letting each step j be the lowest index among those not processed yet such that row $\mathbf{a}_{j,:}$ has at least one nonzero element in a column corresponding to a nonzero entry in $\mathbf{z}_i^{(j-1)}$.

To achieve this goal, an extra copy of the pattern of $\mathbf{a}$ in a column–oriented format is retained, and, quoting again [36], we do the following:

1. at the beginning of the loop 4, $\mathbf{z}_i \leftarrow \mathbf{e}_i$. Therefore, the set of indices $\{j\}$ is initialized with $R_{\mathbf{a}_{:,j}} = \{i : \mathbf{a}_{i,j} \neq 0\}$, the set of row indices of nonzero entries in column i of matrix A;
2. at each iteration in Loop 4, choose j to be the smallest index in the set that is greater than the last one visited;
3. at Step 6, whenever an entry $\mathbf{z}_i(k)$ becomes nonzero, add the row indices $R_{\mathbf{a}_{:,k}}$ corresponding to the nonzeros of column k in matrix A to the set of indices to be visited.

Moreover, for easing the implementation, we can keep copies of the input matrix A both in a row–oriented and column–oriented storage format. This allows building Z and W at the same time, and within the same outer loop: we access the rows and columns of A^T by accessing the columns and rows (respectively) of A. On the other hand, the inner loop results to be separate between Z and W in any case. It runs on a subset of indices specific to each of the triangular factor. The result is Algorithm 2.24; see again [36]. The implementation makes use of a dense work vector $\tilde{\mathbf{z}}$ to compute z_i and w_i. The indices of the nonzero entries are kept in a heap hp. Another heap rhp is used to hold the indices of the rows with at least one nonzero in a column matching a nonzero entry in $\tilde{\mathbf{z}}$, thus giving the set of rows for which we have to compute the scalar products.

The computational cost is estimated in the following result.

Theorem 2.19 (Bertaccini and Filippone [36])**.** *Let* nz_a *be the average number of nonzeros per row in* A *and* nz_z *for* z*; let the bounds*

$$|S| \leq \gamma\, \mathrm{nz}_a, \tag{2.69}$$

$$\mathrm{nz}_z \leq \beta\, \mathrm{nz}_a, \tag{2.70}$$

Algorithm 2.24: Practical left-looking biconjugation

/* For a sparse matrix A let $C_{\mathbf{a}_{i,:}} = \{j : a_{i,j} \neq 0\}$ the set of column indices in row i, and similarly let $R_{\mathbf{a}_{:,j}} = \{i : a_{i,j} \neq 0\}$ */

/* For a set S with an order relation $\leq$, let first(S) be the operator returning the smallest element in S */

$\mathbf{z}_1^{(0)} \leftarrow \mathbf{e}_1$; $p_1^{(0)} \leftarrow a_{1,1}$;

for $i = 2, \ldots, n$ **do**

  /* Inner loop over Z_j */

  $\tilde{\mathbf{z}} \leftarrow \mathbf{e}_i$; $S \leftarrow R_{\mathbf{a}_{:,i}}$;

  while $S \neq \emptyset$ **do**

    $j \leftarrow \text{first}(S)$; $S \leftarrow S - \{j\}$;

    $p(i) \leftarrow \mathbf{a}_{j,:}\tilde{\mathbf{z}}$; $\alpha \leftarrow \left(p(i)/p(j)\right)$;

    if $|\alpha| > \epsilon$ *(drop rule)* **then**

      $\tilde{\mathbf{z}} \leftarrow \tilde{\mathbf{z}} - \alpha \mathbf{z}_{:,j}$;

      for $k \in R_{\mathbf{z}_{:,j}}$ **do**

        $S \leftarrow S \cup \{l \in R_{\mathbf{a}_{:,k}} : j < l < i\}$;

      end

    end

  end

  $p(i) \leftarrow \mathbf{a}_{i,:}\tilde{\mathbf{z}}$;

  Apply a drop rule to $\tilde{\mathbf{z}}$;

  $\mathbf{z}_{:,i} \leftarrow \tilde{\mathbf{z}}$;

  /* Inner loop over W_j */

  $\tilde{\mathbf{z}} \leftarrow \mathbf{e}_i$; $S \leftarrow C_{\mathbf{a}_{i,:}}$;

  while $S \neq \emptyset$ **do**

    $j \leftarrow \text{first}(S)$; $S \leftarrow S - \{j\}$;

    $q(i) \leftarrow \mathbf{a}_{:,j}^T\tilde{\mathbf{z}}$; $\alpha \leftarrow \left(q(i)/q(j)\right)$;

    if $|\alpha| > \epsilon$ *(drop rule)* **then**

      $\tilde{\mathbf{z}} \leftarrow \tilde{\mathbf{z}} - \alpha \mathbf{w}_{:,j}$;

      for $k \in R_{\mathbf{w}_{:,j}}$ **do**

        $S \leftarrow S \cup \{l \in C_{\mathbf{a}_{k,:}} : j < l < i\}$;

      end

    end

  end

  $q(i) \leftarrow (\mathbf{a}_{:,i})^T\tilde{\mathbf{z}}$;

  Apply a drop rule to $\tilde{\mathbf{z}}$;

  $\mathbf{w}_{:,i} \leftarrow \tilde{\mathbf{z}}$;

end

hold true, where $|S|$ is the maximum cardinality of the sets of entries in any of the z_i before the application of the drop rule at Line 15. Then the computational cost of Algorithm 2.24 is

$$O(\gamma n \, \mathrm{nz}_a^2(1 + \beta(1 + \log(\gamma \, \mathrm{nz}_a)))). \tag{2.71}$$

The situation is thus analogous to that of Theorem 2.18. To be more precise, we observe that the bound given by the constant β in (2.70) refers to the ratio between the size of the rows of Z and A. When we enforce that the number of nonzeros in the preconditioner is comparable to that in the matrix A, we are enforcing for β a value that is approximately one half of (2.65). We are comparing, in this case, the upper triangle of the inverse to the upper triangle of the incomplete factorization. Then we obtain that the ratio of number of nonzeros in the preconditioner with respect to the number of nonzeros in A is again β, just as it was in the case of $INVT$. This is due to the fact that we have applied the biconjugation process twice, one time for Z and one for W.

Similarly to what we have seen for the INVS algorithms, the application of the dropping rules in Statements 7, 15 21 and 29 of Algorithm 2.24 have the effect of enforcing a control over the size of the set S. Again, we have that improving the factor γ in this way improves the overall timing needed for the construction of the preconditioner. The key element here is that with dropping Rules 15 and 29 we limit the number of accepted nonzeros.

Since original AINV, that was proposed in [22], can suffer from pivot breakdown when applied to matrices that are not H–matrices, also the more robust version (SAINV) from [21] should be considered. It computes the pivots p_i as

$$p_i \leftarrow \mathbf{z}_i^T A \mathbf{z}_i,$$

instead of the simplified formula

$$p_i \leftarrow \mathbf{a}_{i,:} \mathbf{z}_i.$$

Therefore, if A is symmetric and positive definite, then the pivots cannot be zero since this define a scalar product. For nonsymmetric matrices, even if we do not necessarily define a scalar product, we can apply a similar procedure for avoiding breakdowns. Indeed, there are important cases where pivot breakdown cannot occur for SAINV also in the nonsymmetric case, consider, in particular, the cases in which the symmetric part of the matrix is positive definite; see [233].

If we apply the full formula to the left–looking algorithm then we obtain Algorithm 2.25 from [36]. The product with A is applied at

Algorithm 2.25: Practical left-looking biconjugation stabilized

/* For a sparse matrix A let $C_{\mathbf{a}_{i,:}} = \{j : a_{ij} \neq 0\}$ the set of column indices in row i, and similarly let $R_{\mathbf{a}_{:,j}} = \{i : a_{ij} \neq 0\}$ */

/* For a set S with an order relation $\leq$, let first(S) be the operator returning the smallest element in S */

$\mathbf{z}_1^{(0)} \leftarrow \mathbf{e}_1; \quad p_1^{(0)} \leftarrow a_{1,1};$

for $i = 2, \ldots, n$ **do**

  /* Inner loop over Z_j */

  $\tilde{\mathbf{z}} \leftarrow \mathbf{e}_i; \quad S \leftarrow R_{\mathbf{a}_{:,i}};$

  while $S \neq \emptyset$ **do**

    $j \leftarrow \text{first}(S); S \leftarrow S - \{j\};$

    $p(i) \leftarrow ((\mathbf{w}_{:,j})^T A)\tilde{\mathbf{z}}; \quad \alpha \leftarrow \left(p(i)/p(j)\right);$

    if $|\alpha| > \epsilon$ *(drop rule)* **then**

      $\tilde{\mathbf{z}} \leftarrow \tilde{\mathbf{z}} - \alpha \mathbf{z}_{:,j}$;

      for $k \in R_{\mathbf{z}_{:,j}}$ **do**

        $S \leftarrow S \cup \{l \in R_{\mathbf{a}_{:,k}} : j < l < i\}$;

      end

    end

  end

  Apply a drop rule to $\tilde{\mathbf{z}}$;

  $\mathbf{z}_{:,i} \leftarrow \tilde{\mathbf{z}};$

  /* Inner loop over W_j */

  $\tilde{\mathbf{z}} \leftarrow \mathbf{e}_i; \quad S \leftarrow C_{\mathbf{a}_{i,:}};$

  while $S \neq \emptyset$ **do**

    $j \leftarrow \text{first}(S); \quad S \leftarrow S - \{j\};;$

    $q(i) \leftarrow (AZ_j)^T \tilde{\mathbf{z}}; \quad \alpha \leftarrow \left(q(i)/q(j)\right);;$

    if $|\alpha| > \epsilon$ *(drop rule)* **then**

      $\tilde{\mathbf{z}} \leftarrow \tilde{\mathbf{z}} - \alpha \mathbf{w}_{:,j}$;

      for $k \in R_{\mathbf{w}_{:,j}}$ **do**

        $S \leftarrow S \cup \{l \in C_{\mathbf{a}_{k,:}} : j < l < i\}$;

      end

    end

  end

  Apply a drop rule to $\tilde{\mathbf{z}}$;

  $\mathbf{w}_{:,i} \leftarrow \tilde{\mathbf{z}};$

  $p(i) \leftarrow q(i) \leftarrow (\mathbf{w}_{:,i})^T A \mathbf{z}_{:,i};$

end

Steps 6, 19 and 29. Note that the two triangular matrices W and Z are no longer independent of each other. Indeed, the computation of the p_i and q_i must be performed at Step 29 where the relevant elements of both W and Z are available. On the other hand, stabilization is not always beneficial. As an example, in all the tests performed in [36], there is no significant advantage in using Algorithm 2.25 over Algorithm 2.24.

I

Sparse Structure and Preconditioners

"An "A^{-1}" in a formula almost always means "solve a linear system" and almost never means "compute A^{-1}."""

Golub and Van Loan, Matrix Computations

3

Interpolatory Updates of Approximate Inverse Preconditioners

The solution of sequences of linear systems as

$$A^{(k)}\mathbf{x}^{(k)} = \mathbf{b}^{(k)},\ k \geq 0, \{A^{(k)}\}_k \subset \mathbb{R}^{n\times n},\ \{\mathbf{x}^{(k)}\}_k, \{\mathbf{b}^{(k)}\}_k \subset \mathbb{R}^n, \quad (3.1)$$

is an important problem that arises in several contexts in scientific computing. Note that we use the term *sequences of linear systems* whenever all the matrices $A^{(k)}$ share something and/or do not differ much in some sense. Moreover, we assume always that the sequences $\{A^{(k)}\}_k$ are also sequences of matrices of growing dimension, i.e., $\{A^{(k)}\}_k \equiv \{\{A_n^{(k)}\}_n\}_k$ with $A_n^{(k)} \in \mathbb{R}^{n\times n}$. With an abuse of notation, we do not report explicitly the dependence on the index n, nonetheless, when we talk about *clustering properties* (Definition 2.1) we are always referring to sequences indexed by n; refer also to Section 1.1.

The most frequent and natural example of this setting is the solution of time–dependent partial differential equations by implicit methods.

Example 3.1. T*ime-dependent PDEs. Consider for instance a partial differential equation of the form*

$$\frac{\partial u}{\partial t} = \mathcal{L}u + g, \quad (3.2)$$

on a region with Dirichlet boundary conditions and an initial condition $u(x,0) = u_0(x)$. An implicit time discretization with time step τ_m and a finite difference/element approach in space for $\mathcal{L}$ with stepsize h gives a sequence of linear systems if $\mathcal{L} = A$ is, e.g., linear

$$(I - \frac{\tau_m}{h^2}A)\mathbf{u}^{(m+1)} = u^m + \tau_m\Phi(h, \mathbf{g}^{(m+1)}, \mathbf{u}^{(m)}, \mathbf{u}^{(m-1)}, \ldots). \quad m \geq 0. \quad (3.3)$$

If $\mathcal{L}$ is non linear then, by using a quasi-Newton step, we can get again linear systems as in (3.3)*. Typically the time step τ_m will not be constant, but will change adaptively from time to time.*

Example 3.2. U*nconstrained optimization.*[1]*. Consider the Newton method for finding a zero* $\mathbf{x}^*$ *of the function* $F(\mathbf{x}) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$ *at least of class* $\mathscr{C}^1(\mathbb{R}^n)$*. Let us apply a Krylov subspace method for solving the linear systems*

$$J\left(\mathbf{x}^{(k)}\right)\mathbf{s}^{(k)} = -F\left(\mathbf{x}^{(k)}\right), \quad k = 1, 2, \ldots, \tag{3.4}$$

where $\mathbf{x}^{(k)}$ *is the current candidate solution,* J *is the Jacobian matrix, or at least an approximated version of the operator providing the action of the Jacobian operator on a vector. Then, a candidate approximation for* $\mathbf{x}^*$ *can be found in the sequence* $\{\mathbf{x}^{(k)}\}$ *generated starting from an initial guess* $\mathbf{x}^{(0)}$ *as usual:*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}.$$

Therefore also in this case (3.4) *is a sequence of linear systems. If a Krylov solver is used to face with these linear systems, then those methods are usually called Newton–Krylov methods.*

More generally, sequences of systems of the form (3.1) occur in the numerical solution of discretized nonlinear systems of ordinary and partial differential equations with implicit methods; see, e.g., [28, 186, 282] and the references therein. Sequences of linear systems also occur in other contexts, such as regularization of ill-posed least squares problems, see, e.g., [42], trust region methods in nonlinear optimization, and elsewhere.

The linear systems of the underlying sequences could need a suitable preconditioner. Reusing the same preconditioner for all linear systems in the sequence each time often leads to slow convergence, whereas recomputing a preconditioner each time is both costly and wasteful. Clearly, there is a broad range of possibilities within these two extremes. It should be possible to modify an existing preconditioner at a cost much lower than recomputing a preconditioner from scratch; even if the resulting preconditioner can be expected to be less effective than a brand new one in terms of iteration count, the overall cost should be reduced.

Due to these motivations, there is an intense activity research on the update of preconditioners, starting from the seminal paper by Meurant [205]. In the sequel, we will present our approach based on some fairly general strategies for updating sequences of generic preconditioners based on incomplete factorizations. Further interesting approaches can be found in [16–18, 46, 70, 272, 273]. Other strategies,

1 An account for Newton and inexact Newton method can be found in [91], specific information on the Newton–Krylov methods can be found in [172].

mainly based on block Krylov methods and not considered here, can be devised if the linear systems (3.1) share the same matrix.

In [33, 106] we proposed the update of few preconditioners in factorized inverse form by interpolating the inverse factors of few decompositions. We concentrate on quadratic matrix interpolation and therefore we start building our preconditioner from three preconditioners computed for three appropriately chosen different matrices in the sequence (3.1) given by

$$A^{(i)}\,\mathbf{x}^{(i)} = \mathbf{b}^{(i)}, \quad i = 0, 1, \ldots, s. \tag{3.5}$$

However, this paradigm is fairly general and can be applied to higher degree or spline-like interpolation.

We stress that our approach requires that the matrices used for interpolation should be known in advance while this is not required in the other above mentioned "non interpolation-based" updating paradigms. On the other hand, the strategies we proposed in [33, 106] can use the p matrices of the sequence $\{A^{(i)}\}_{i=0}^{s}$ to build the needed preconditioners and then use interpolation for the others.

In order to build up the underlying interpolated preconditioners, we need to provide preconditioners $P^{(i_j)}$ for (a few) matrices $A^{(i_j)}$, $j = 0, 1, 2, \ldots, p$, in approximate inverse form, as in Sections 2.4.2 and 2.4.3. Given $A^{(i_j)}$, $j = 0, 1, 2, \ldots, p$ ($p = 2$ in our tests in Section 3.2), we need to generate $p + 1$ well defined and sparse approximations in factorized form

$$P^{(i_j)} = W^{(i_j)}\,D^{(i_j)^{-1}}\,Z^{(i_j)^T} \tag{3.6}$$

for $A^{(i_j)^{-1}}$. Here we concentrate on incomplete factorization algorithms from Section 2.4. Given ϵ the drop tolerance of the algorithm generating the incomplete factorizations, i.e., the threshold below which the extra–diagonals elements are set to zero, it is intended that

$$\lim_{\epsilon \to 0} \| A^{(i_j)^{-1}} - W^{(i_j)}\,D^{(i_j)^{-1}}\,Z^{(i_j)^T} \| = 0, \quad j = 0, 1, \ldots, p$$

for any matrix norm $\|\cdot\|$, i.e., that for $\epsilon = 0$ the factorizations for the inverse matrices are *exact*.

The algorithms for approximate inverse factorization considered in the following belong to two classes: *inverse ILU* from Section 2.4.2 and *approximate inverse preconditioners* or *AINV* from Section 2.4.3.

3.1 *Interpolated Preconditioners*

Given the underlying sequence of $n \times n$ matrices $\{A^{(i)}\}_{i=0}^{p}$, let us begin with the computation of three reference approximate inverse precondi-

tioners for the matrices $A^{(i_0)}$, $A^{(i_1)}$ and $A^{(i_2)}$ chosen appropriately from the sequence $\{A^{(i)}\}_i$, i.e.,

$$\left\{\text{approx. inverse factor. } {A^{(i)}}^{-1}\right\} = W^{(i)}\, {D^{(i)}}^{-1}\, {Z^{(i)}}^{T}, \quad i = i_0, i_1, i_2. \tag{3.7}$$

The choice of $A^{(i_0)}$, $A^{(i_1)}$ and $A^{(i_2)}$ is problem-dependent and it is made in order to maximize the probabilities to get a reasonable approximation for all the interpolated preconditioners. We will not focus on this aspect here, leaving more details for some specific case study, see, e.g., Section 4.3.4 and Section 5.2.2, related to specific applications.

As we have discussed in Section 2.4 the factorizations (3.7) can be produced by various algorithms. Here we focus on inversion and sparsification algorithms and AINV as revisited in [36] and described in Section 2.4, particularly, see Algorithm 2.25. We build the two preconditioner factors by mean of two separate quadratic interpolation of the points $(\alpha_i, {Z^{(i)}}^T)$, and $(\alpha_i, W^{(i)})$, for $i = i_0, i_1, i_2$. Observe that the coefficients $\{\alpha_i\}_{i=0}^{s}$ are the discretization of a parameter α linked to the problem and to the i indices, e.g., if we are dealing with variable time step integrator then it is the variable time step, otherwise it could be an interpolation or a specific parameter linked to the variable in time coefficients. The functions for the interpolation are given by the quadratic polynomial

$$p_2(\alpha;\cdot) = a + b\alpha + c\alpha^2, \tag{3.8}$$

where we obtain the expression for the coefficients a, b, c for a generic triplet of matrices $M^{(1)}, M^{(2)}, M^{(3)}$ as

$$\begin{aligned}
a = &\quad \frac{\alpha_0\,(\alpha_0-\alpha_1)\,\alpha_1 M^{(3)} + \alpha_2\left(\alpha_1\,(\alpha_1-\alpha_2)\,M^{(1)} + \alpha_0\,(\alpha_2-\alpha_0)\,M^{(2)}\right)}{(\alpha_0-\alpha_1)\,(\alpha_0-\alpha_2)\,(\alpha_1-\alpha_2)},\\
b = &\quad \frac{\left(\alpha_1^2-\alpha_0^2\right)M^{(3)} + \left(\alpha_0^2-\alpha_2^2\right)M^{(2)} + \left(\alpha_2^2-\alpha_1^2\right)M^{(1)}}{(\alpha_0-\alpha_1)\,(\alpha_0-\alpha_2)\,(\alpha_1-\alpha_2)},\\
c = &\quad \frac{(\alpha_0-\alpha_1)\,M^{(3)} + (\alpha_1-\alpha_2)\,M^{(1)} + (\alpha_2-\alpha_0)\,M^{(2)}}{(\alpha_0-\alpha_1)\,(\alpha_0-\alpha_2)\,(\alpha_1-\alpha_2)}.
\end{aligned}$$

Observe that they can be computed by formally solving the classical 3×3 linear system for quadratic interpolation, given by

$$\begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 \\ 1 & \alpha_1 & \alpha_1^2 \\ 1 & \alpha_2 & \alpha_2^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} M_0 \\ M_1 \\ M_2 \end{bmatrix}.$$

We build the approximations for the Z_α and W_α matrices as functions of the parameter α by using equations (3.8) with the coefficients a, b, c computed for the $\{Z^{(i_j)}\}_{j=0}^2$ and $\{W^{(i_j)}\}_{j=0}^2$ matrices coming from the factorized approximate inverses of the reference matrices. In this way we have

$$Z_\alpha = p_2(\alpha; \{Z^{(i_j)}\}_{j=0}^2) \quad \text{and} \quad W_\alpha = p_2(\alpha; \{W^{(i_j)}\}_{j=0}^2). \tag{3.9}$$

We can take as a preconditioner the approximate inverse of the generic matrix $A^{(i)}$ of the sequence given by

$$M_i^{-1} = W_\alpha (D^{(i_j)} + [Z_\alpha^T \Delta W_\alpha]_k)^{-1} Z_\alpha^T,$$

where $D^{(i_j)}$ is the diagonal matrix coming from one of the reference preconditioner (3.6). The matrix Δ is given by

$$\Delta = A^{(i)} - A^{(i_j)}, \tag{3.10}$$

and the operator $[\cdot]_k$ extracts the k upper and lower diagonals of a matrix ($[\cdot]_0$ gives the main diagonal).

By exploiting the formalism introduced in [15], instead of the simple band–extraction function $[\cdot]_k$, in general we could use the function

$$\tilde{E} = g(Z_\alpha^T \Delta W_\alpha). \tag{3.11}$$

Similarly, the function g serves to generate a sparse matrix from a dense one. If X is a dense matrix, $g(X) = \tilde{X}$, where $\tilde{X}$ is a sparse matrix made by selecting only certain entries of X.

Therefore, what we do from the computational point of view is working with $g = [\cdot]_k$ and the k banded approximation of the correction matrix E. In this way, the matrix $\tilde{E}$ is not completely computed, in this regard we can consider also the approximation from Theorem 4.4 in which under the hypothesis of fast decay of the elements for the $\{{A^{(i)}}^{-1}\}_i$ a cheaper approximation can be obtained considering instead the k banded approximation of Z_α and W_α before computing their product. Then, to choose between the different reference matrices, i.e., between the different $D^{(i_j)}$, taking into account the value assumed by the α parameter for each single linear system, we take the index $i^* = i_j$ as the one that realizes

$$i^* = \arg \min_{i_j = i_0, i_1, i_2} \|A^{(i)} - A^{(i_j)}\|_F, \quad \Delta = A^{(i)} - A^{(i^*)}. \tag{3.12}$$

Observe that this depends on the selection of the original reference matrices $\{A^{(i_j)}\}_{j=0}^{2}$, that, as we have already discussed at the beginning, is a problem dependent choice, and on the current matrix $A^{(i)}$ to be preconditioned; see Section 3.2 for an example of this choice and the effect on the selection of the i^* index. When this selection has been made, the computation of the i^* amounts only in computing few Frobenius matrix norm.

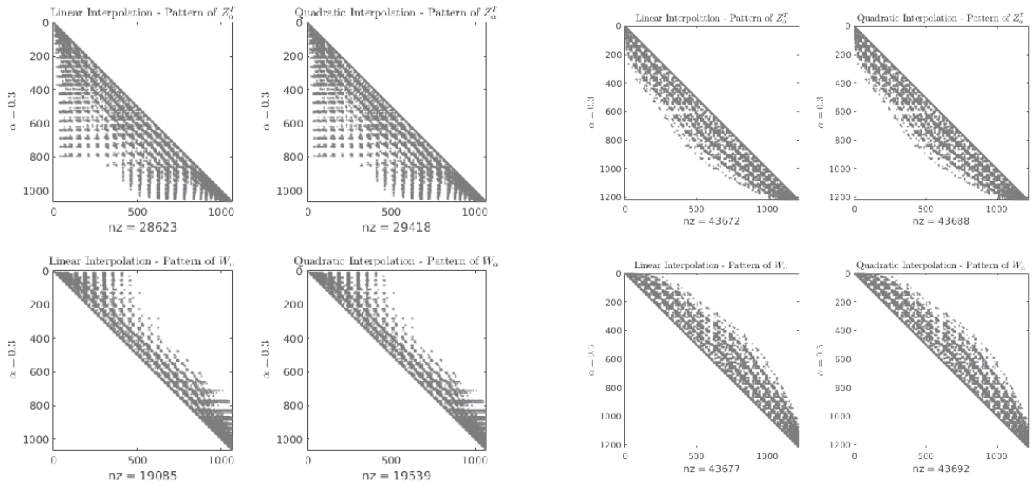
Therefore we build a preconditioner update in factorized form

$$M_{i,k}^{-1} = W_\alpha (D^{(i^*)} + E_k)^{-1} Z_\alpha^T, \text{ with } \begin{cases} Z_\alpha = p_2(\alpha; \{Z^{(i_j)}\}_{j=0}^{2}), \\ W_\alpha = p_2(\alpha; \{W^{(i_j)}\}_{j=0}^{2}), \\ E_k = \left[Z_\alpha^T \Delta W_\alpha\right]_k . \end{cases} \tag{3.13}$$

Regarding the *memory occupation*, we observe that it amounts, at most, in three times the sum of the nonzero elements of the reference matrices, assuming a reasonable worst case with a non cancellation rule:

$$\text{nnz}(p_2(\alpha, M^{(1)}, M^{(2)}, M^{(3)})) \leq 3 \sum_i \text{nnz}(M^{(i)}),$$

see, e.g., Figure 3.1, referring to the three experiments in the next section.



(a) Experiment 2 (b) Experiment 4

Figure 3.1. Memory occupation for the Z_α^T matrices.

.

Moreover, the asymptotic computational cost for the construction of the preconditioner $M_{i,k}^{-1}$ and its application is the same as linear interpolation and as the updates in [15, 31, 106].

Using the upper bound for condition numbers of $n \times n$ regular triangular matrices from [183], we can also prove a bound for the condition number of both the matrices Z_α and W_α obtained by the quadratic interpolation formula (3.8).

Corollary 3.1 (Bertaccini and Durastante [33]; Bound for $\kappa_2(\cdot)$)**.** *Let us consider the unit upper triangular matrices of order n given by*

$$Z_\alpha = p_2(\alpha, Z^{(i_0)}, Z^{(i_1)}, Z^{(i_2)}), \quad W_\alpha = p_2(\alpha, W^{(i_0)}, W^{(i_1)}, W^{(i_2)}). \tag{3.14}$$

If the non–diagonal elements of Z_α have absolute values not larger than a_Z, then we can bound $\kappa_2(Z)$ as

$$\begin{aligned} \|Z_\alpha^{-1}\|_2 \leq & \|Z_\alpha^{-1}\|_F \leq \sqrt{n\left(\frac{2}{a_Z+2}\right) + \frac{(a_Z+1)^{2n}-1}{(a_Z+2)^2}}, \\ \|Z_\alpha\|_2 \leq & \|Z_\alpha\|_F \leq \sqrt{n + \frac{n(n-1)}{2}a_Z^2}. \end{aligned} \tag{3.15}$$

If the non–diagonal elements of W_α have absolute values not larger than a_W, then we can bound $\kappa_2(W)$ as

$$\begin{aligned} \|W_\alpha^{-1}\|_2 \leq & \|W_\alpha^{-1}\|_F \leq \sqrt{n\left(\frac{2}{a_W+2}\right) + \frac{(a_W+1)^{2n}-1}{(a_W+2)^2}}, \\ \|W_\alpha\|_2 \leq & \|W_\alpha\|_F \leq \sqrt{n + \frac{n(n-1)}{2}a_W^2}. \end{aligned} \tag{3.16}$$

Moreover, we can express the values of a_Z and a_W as

$$\begin{aligned} a_Z = & p_2\left(\alpha; \max_{i,j}|z_{i,j}^{(i_0)}|, \max_{i,j}|z_{i,j}^{(i_1)}|, \max_{i,j}|z_{i,j}^{(i_2)}|\right), \\ a_W = & p_2\left(\alpha; \max_{i,j}|w_{i,j}^{(i_0)}|, \max_{i,j}|w_{i,j}^{(i_1)}|, \max_{i,j}|w_{i,j}^{(i_2)}|\right). \end{aligned} \tag{3.17}$$

The condition number of the interpolated matrices is bounded by the condition number of the reference matrices. Therefore, reference preconditioners with a reasonable condition number will potentially generate interpolated preconditioners with a reasonable condition number too.

Let us write the difference between the interpolated preconditioner (computed on the basis of the *exact* factorizations for the inverses of $A^{(i)}$) and its target matrix.

Lemma 3.1 (Bertaccini and Durastante [33]). *Let $L^{(i)}D^{(i)}U^{(i)}$ be the exact factorization of $A^{(i)} \in \mathbb{R}^{n\times n}$ and consider the quadratic interpolated matrix $M_{i,k} = L_\alpha D_\alpha U_\alpha$, where*

$$L_\alpha = p_2(\alpha, \{L^{(i_j)}\}_{j=0}^2),\ U_\alpha = p_2(\alpha, \{U^{(i_j)}\}_{j=0}^2),\ D_\alpha = D^{(i^*)} + E_k.$$

We have

$$M_{i,k} - A^{(i)} = (L_\alpha D^{(i^*)} U_\alpha - L^{(i)}D^{(i)}U^{(i)}) + (Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta),$$

where $\Delta = A^{(i)} - A^{(i^*)}$, as in equation (3.12) *and*

$$M_{i,k} - A^{(i)} = 0, \qquad i = i_0, i_1, i_2, \tag{3.18}$$

i.e., the interpolated matrix $M_{i,k}$[2] (computed on the basis of the exact *factorization for the inverses) is exact for interpolation points corresponding to $\alpha = \alpha_i$, $i = 0, 1, 2$.*

Proof.

$$\begin{aligned} M_{i,k} - A^{(i)} =& L_\alpha(D^{(i^*)} + E_k)U_\alpha - (A_i - A^{(i^*)} + A^{(i^*)}) \\ =& L_\alpha D^{(i^*)} U_\alpha + L_\alpha E_k U_\alpha - \Delta - A^{(i^*)} \\ =& \left(L_\alpha D^{(i^*)} U_\alpha - A^{(i^*)}\right) + (L_\alpha E_k U_\alpha - \Delta) \\ =& \left(L_\alpha D^{(i^*)} U_\alpha - L^{(i)}D^{(i)}U^{(i)}\right) + (Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta). \end{aligned}$$ □

We define for later use the matrix

$$C_\alpha = L_\alpha D^{(i^*)} U_\alpha - L^{(i)}D^{(i)}U^{(i)}. \tag{3.19}$$

In the style of [15, 31] we prove that the updated preconditioner $M_{i,k}^{-1}$ in (3.13) applied to the generic ith matrix of the sequence $\{A_n^{(i)}\}_n$ clusters eigenvalues of the preconditioned matrix sequence $\{M_{i,k}^{-1}A_n^{(i)}\}_n$ for any $i = 0, \ldots, s$.

Theorem 3.1 (Bertaccini and Durastante [33]). *Given the sequence of linear systems $A^{(i)}\mathbf{x} = \mathbf{b}^{(i)}$ for $i = 0, 1, \ldots, s$, let us consider the preconditioner defined in equation* (3.13). *If there exists $\delta \geq 0$ and $t \ll n$ such that*

$$Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta = U\Sigma V^T, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),$$

2 The matrix $M_{i,k}$ is called *interpolated preconditioner* if the interpolation is performed on approximate inverse decompositions for $A^{(i)}$.

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_t \geq \delta > \sigma_{t+1} \geq \ldots \geq \sigma_n, \tag{3.20}$$

and

$$\max_{\alpha\in(\alpha_1,\alpha_2)} \|D^{(i^*)^{-1}} E_k\| \leq \frac{1}{2}, \tag{3.21}$$

then there exist matrices C_α, Δ, F *and a scalar constant* c_α *such that*

$$M_{i,k}^{-1} A^{(i)} = I + M_{\alpha,k}^{-1} C_\alpha + \Delta + F, \tag{3.22}$$

with $\mathrm{Rank}(\Delta) = t \ll n$, *independent from* α *and*

$$\|F\|_2 \leq \frac{2\delta c_\alpha}{\beta} \frac{\sqrt{n\left((n-1)a_W^2 + 2\right)}\sqrt{n\left((n-1)a_Z^2 + 2\right)}}{2\,n\,b_W b_Z}, \tag{3.23}$$

where

$$\beta = \min\{\max_{r=1,\ldots,n} |d_r|^{-1}, c\}, \quad d_r = \left(D^{(i^*)}\right)_{r,r}, \tag{3.24}$$

with c *constant used to avoid zero pivots in the matrix* D_{α_1},

$$b_W = \min_{i,j=1,2,\ldots,n} |w_{i,j}^{(i_0)}|^{1/2}, \quad b_Z = \min_{i,j=1,2,\ldots,n} |z_{i,j}^{(i_0)}|^{1/2},$$

and

$$\|M_{i,k}^{-1} C_\alpha\|_2 \leq 2c_\alpha \kappa_2(L_\alpha)\kappa_2(U_\alpha)\frac{\max_r |d_r|}{\min_r |d_r|} + \|M_{i,k}^{-1} A_{\alpha_1}\|_2.$$

Proof. By using the decomposition in Lemma 3.1 we have

$$M_{i,k}^{-1} A_\alpha = I + M_{i,k}^{-1} C_\alpha + M_{i,k}^{-1}(Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta).$$

We can use the hypothesis (3.20) on the singular value decomposition of

$$Z_\alpha^{-T} E_k W_\alpha^{-1} - \Delta = \Delta_1 + F_1,$$

$$\begin{aligned} \Delta_1 &= U\,\mathrm{diag}(\sigma_1, \ldots, \sigma_t, 0, \ldots, 0)V^H, \\ F_1 &= U\,\mathrm{diag}(0, \ldots, 0, \sigma_{t+1}, \ldots, \sigma_n)V^H. \end{aligned} \tag{3.25}$$

Therefore we obtain

$$M_{i,k}^{-1} A^{(i)} = I + M_{i,k}^{-1} C_\alpha + M_{i,k}^{-1}\Delta_1 + M_{i,k}^{-1} F_1.$$

To proceed we now need to get an estimate of the norm $\|M_{\alpha,k}^{-1}\|_2$:

$$\begin{aligned}
\|M_{i,k}^{-1}\|_2 =& \|W_\alpha (D^{(i^*)} + E_k)^{-1} Z_\alpha\|_2 \leq \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \|(D^{(i^*)} + E_k)^{-1}\|_2 \\
\leq& \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \|{D^{(i^*)}}^{-1} (I + {D^{(i^*)}}^{-1} E_k)^{-1}\|_2 \\
\overset{\text{by (3.21)}}{\leq}& \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \|{D^{(i^*)}}^{-1}\|_2 \left\| \sum_{n\geq 0} (-1)^n ({D^{(i^*)}}^{-1} E_k)^n \right\|_2 \\
\overset{\text{by (3.21)}}{\leq}& \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \max_{r=1,2,\ldots,n} (|d_r|^{-1}) c_\alpha \left(1 - \|{D^{(i^*)}}^{-1} E_k\|_2\right) \\
\leq& 2 c_\alpha \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \max_{r=1,2,\ldots,n} (|d_r|^{-1}).
\end{aligned}$$

We now define the matrix $F = M_{i,k}^{-1} F_1$ and, by using the notation and results of Corollary 3.1,

$$\begin{aligned}
\|F\|_2 \leq& \|M_{i,k}^{-1}\|_2 \|F_1\|_2 \overset{\text{by (3.20)}}{\leq} \delta \|M_{i,k}^{-1}\|_2 \\
\leq& 2\delta c_\alpha \|W_\alpha\|_2 \, \|Z_\alpha\|_2 \max_{r=1,2,\ldots,n} (|d_r|^{-1}) \\
\leq& \frac{2\delta c_\alpha}{\beta} \frac{\|W_\alpha\|_2 \, \|Z_\alpha\|_2}{\min\limits_{r=1,2,\ldots,n} \|w_{:,r}^{(\alpha_1)}\| \min\limits_{r=1,2,\ldots,n} \|z_{:,r}^{(\alpha_1)}\|} \\
\leq& \frac{2\delta c_\alpha}{\beta} \frac{\sqrt{n\left((n-1)a_W^2 + 2\right)} \sqrt{n\left((n-1)a_Z^2 + 2\right)}}{2\, n\, b_W b_Z}.
\end{aligned}$$

Let us work on the term $M_{i,k}^{-1} C_\alpha$, for which we observe that is 0 for $\alpha = \alpha_i$, $i = i_0, i_1, i_2$.

$$\begin{aligned}
\|M_{i,k}^{-1} C_\alpha\|_2 \leq& \|M_{i,k}^{-1} (L_\alpha D^{(i^*)} U_\alpha - L_1 D^{(i^*)} U_1)\|_2 \\
\leq& \|D^{(i^*)}\|_2 \|M_{i,k}^{-1}\|_2 \|L_\alpha\|_2 \|U_\alpha\|_2 + \|M_{i,k}^{-1} A^{(i_1)}\|_2 \\
\leq& 2 c_\alpha \kappa_2(L_\alpha) \kappa_2(U_\alpha) \frac{\max_r |d_r|}{\min_r |d_r|} + \|M_{i,k}^{-1} A^{(i_1)}\|_2.
\end{aligned}$$

By introducing the low rank matrix $\Delta = M_{i,k}^{-1} \Delta_1$ we complete the proof. □

We just proved that our interpolated preconditioners can show a clustering of the spectra of the underlying matrices under appropriate conditions. This is a behavior that has to be expected in this kind of framework, where the updated preconditioner greatly relies on the reference ones and is consistent with the behavior of the condition number of the updates in Corollary 3.1.

3.2 *Numerical Examples*

We resume here some of the tests performed on the proposed interpolated preconditioners. We compare the *fixed AINV*, the *updated AINV*, the *interpolated AINV*, and the interpolated 2 AINV, where the meaning of these abbreviations is explained below.

- **fixed AINV**: compute just an approximate inverse preconditioner and use it for all the other linear systems in the sequence.
- **updated AINV**: compute the approximate inverse preconditioner for the first matrix of the sequence and update it for all the others as in [15].
- **interpolated AINV**: compute two approximate inverse preconditioners, one for the first matrix of the sequence and another for the last and use them in the linear matrix interpolation strategy proposed in [106].
- **interpolated 2 AINV**: the quadratic polynomial matrix interpolated preconditioner introduced in Section 3.1.

We also tried ILU(0) and ILUT(ϵ) preconditioners with $\epsilon = 10^{-1}$ and 10^{-2}; see [247] for details. However, the results are not reported because the computed factors are too ill–conditioned for all experiments below.

Note that details of tests using *recomputed preconditioners*, i.e., iterative solvers using approximate inverse preconditioners computed from scratch for each linear system, are omitted. Indeed, we experienced that their timings were always greater than all the others in the tables, regardless of the implementations we tried.

The codes are in a prototype stage using Matlab R2016b in order to simplify changes and porting to more powerful platforms. Our machine is a laptop running Linux with 8 Gb memory and CPU Intel® Core™i7-4710HQ CPU with clock 2.50 GHz. *GMRES* and *BiCGstab* are considered with a relative residual stopping criterium of $\epsilon = 10^{-9}$ in order to test the performances of the preconditioners. A similar behavior is observed also with $\epsilon = 10^{-6}$ and is not reported here.

The timings do not include the cost of generating the approximate inverse factorizations, because these are not computed in Matlab but use the implementation proposed in [36].

We stress again that for computing one *interpolation preconditioner* the triplet of matrices described in Section 3.1 should be known in advance.

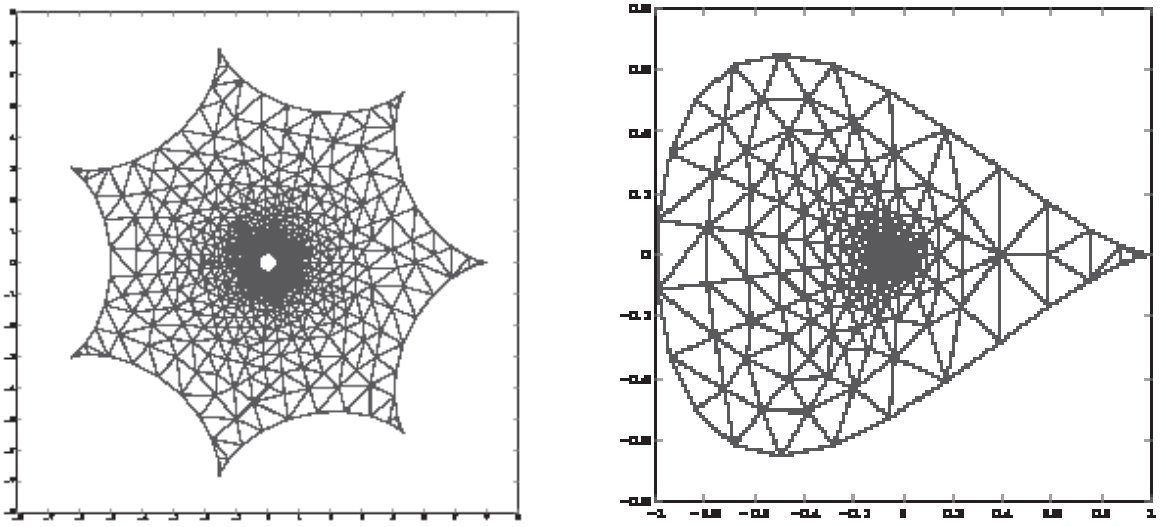
Unsteady State Case

We consider the finite element approximation of the convex combination ($\alpha \in [0,1]$) of the following equations for $(x,y) \in \Omega$,

$$\begin{cases} u_t + a_1(x,y)u_x + b_1(x,y)u_y - \nabla \cdot (k_1(x,y)\nabla u) = f_1(x,y), \\ u_t + a_2(x,y)u_x + b_2(x,y)u_y - \nabla \cdot (k_2(x,y)\nabla u) = f_2(x,y). \end{cases},$$

and $k_i(x,y), a_i(x,y)b_i(x,y) > 0 \ \forall (x,y) \in \Omega \ i = 1,2$, with the same boundary conditions on the borders of the domain.

Experiment 1. We start considering the boundary given in figure 3.2(a) parametrized by the equations $\mathscr{C}_1 \setminus \mathscr{C}_2$, where



(a) Mesh for the Experiment 1. (b) Mesh for the steady state experiments.

Figure 3.2. Finite element mesh for the experiments.

$$\mathscr{C}_1 = \begin{cases} x = b\cos\left(t\left(\frac{a}{b} - 1\right)\right) + t\cos(a-b), & a = 7, b = 1, \\ y = t\sin(a-b) - b\sin\left(t\left(\frac{a}{b} - 1\right)\right), & t \in [0, 2\pi] \end{cases}$$

$$\mathscr{C}_2 = \begin{cases} x = 0.3\cos(t), \\ y = 0.3\sin(t). \end{cases}$$

The discretization is obtained by applying the backward Euler method for the time derivative and taking the test functions in the space of the piecewise linear continuous polynomials

$$P1_h = \left\{ v \in H^1(\Omega) \,|\, \forall\, K \in \mathscr{T}_h, \ \ v|_K \in \mathbb{R}_1[x] \right\}.$$

The coefficient function are chosen as

$$a_1(x) = 50(1 + 0.9\sin(100\pi x)), \quad b_1(y) = 50(1 + 0.3\sin(100\pi y)),$$
$$k_1(x, y) = 1 + x^2y^2\exp(-x^2 - y^2),$$
$$a_2(x, y) = \cos^2(2x + y), \quad b_2(x, y) = \cos^2(x + 2y), \quad k_2(x, y) = 1 + x^4 + y^4,$$
$$f_1(x, y) = f_2(x, y) = \pi^2(\sin(x) + \cos(y))$$

and boundary condition as

$$u(x, y, t) = x, \ (x, y) \in \mathscr{C}_1, \quad u(x, y, t) = y, \ (x, y) \in \mathscr{C}_2, \ \forall t \geq 0.$$

The initial condition is the zero function over the entire domain. FreeFem++ software [156] is used.

The results of the preconditioning strategies for this problem are reported in Table 3.1 with *GMRES*. The reference AINV preconditioners are computed with a drop tolerance $\delta = 10^{-2}$, and only the main diagonal of matrix Δ is used, i.e., we are using $[\Delta]_k$ with $k = 1$. The values of α used for reference are $\alpha = 0, 0.5, 1$, respectively.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
24	0.069517	24	0.045601	24	0.060492	24	0.048347
28	0.057633	27	0.050633	27	**0.055109**	25	0.062979
32	0.074485	31	0.058652	31	**0.060355**	26	0.060886
37	0.078756	35	0.067780	35	0.068347	28	**0.064882**
42	0.096048	40	0.079990	40	0.082364	29	**0.067872**
47	0.118547	44	0.095973	44	0.094332	29	**0.062821**
51	0.122968	48	0.104468	48	0.107474	29	**0.067773**
55	0.127817	53	0.118813	53	0.117902	29	**0.067542**
59	0.142804	58	0.136582	58	0.140420	29	**0.063895**
63	0.156484	63	0.155178	63	0.150704	29	**0.063404**
66	0.166710	69	0.188811	69	0.179450	30	**0.062864**

Table 3.1. Experiment 1, $\alpha = 0 : 1e - 1 : 1$, size of the matrix $n = 10765$, *GMRES* algorithm. The columns IT and T(s) contains respectively the average number of iterations and the average time needed to perform them for solving the αth linear system.

From these experiments the performance of the fixed ILU preconditioner are omitted because, even if convergent, the related matrices are close to singular or badly scaled. Also the results with the unpreconditioned *GMRES* are omitted because they never reach convergence, due to stagnation or to the fact that they reach the maximum number of

iteration. These results are obtained by choosing the following indices i^* in equation (3.12), $i^* = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 \end{pmatrix}$, that fit well with the expected behavior of the interpolation, and that will be the same for all the other experiments.

Experiment 2. We consider the same settings as the previous experiment, just changing the coefficients functions as

$$a_1(x) = 50(1 + 0.9\sin(100\pi x)), \quad b_1(y) = 50(1 + 0.3\sin(15\pi y)),$$
$$k_1(x, y) = x^2 y^2 \exp(-x^2 - y^2),$$
$$a_2(x) = 1 + 0.6\sin(100\pi x), \quad b_2(y) = 1 + 0.6\sin(100\pi y),$$
$$k_2(x, y) = x^2 + y^2,$$
$$f_1(x, y) = f_2(x, y) = \pi^2(\sin(x) + \cos(y))$$

and the boundary conditions as

$$u(x, y, t) = 0, \ (x, y) \in \mathscr{C}_1 \cup \mathscr{C}_2, \quad \forall t \geq 0.$$

The result of the experiments are collected in Table 3.2. Again we

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
14	0.021906	14	0.022726	14	0.021684	14	0.023445
22	0.035538	22	0.037535	22	0.035833	15	**0.033629**
32	0.055325	33	0.064074	33	0.060134	17	**0.037763**
42	0.081190	43	0.093787	43	0.084896	18	**0.041768**
52	0.110713	53	0.119737	53	0.121316	19	**0.044107**
61	0.130961	64	0.152890	64	0.172717	20	**0.037859**
71	0.165355	75	0.213161	75	0.222384	20	**0.046074**
80	0.204361	85	0.243111	85	0.264776	20	**0.047538**
89	0.253234	95	0.291630	95	0.308552	21	**0.048328**
97	0.313632	104	0.341767	104	0.349776	21	**0.048535**
105	0.359167	114	0.401225	114	0.439643	21	**0.051557**

Table 3.2. Experiment 2, $\alpha = 0 : 1e - 1 : 1$, size of the matrix $n = 10765$, *GMRES* algorithm. The columns IT and T(s) contains respectively the average number of iterations and the average time needed to perform them for solving the αth linear system.

exclude the results with the unpreconditioned *GMRES* in Table 3.2 because it never reached convergence due to stagnation.

For this case we also consider the solution with restarted *GMRES*, i.e., *GMRES*(50), with the same settings used for the reference AINV preconditioners. Result for this case are collected in Table 3.3.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
1 14	0.026153	1 14	0.023149	1 14	0.022158	1 14	0.021390
1 22	0.035022	1 22	0.039135	1 22	0.034276	1 15	**0.031460**
1 32	0.055942	1 33	0.056661	1 33	0.056447	1 17	**0.038575**
1 42	0.080134	1 43	0.081664	1 43	0.081475	1 18	**0.039210**
2 2	0.109333	2 4	0.108720	2 4	0.107498	1 19	**0.048113**
2 12	0.114464	2 15	0.122823	2 15	0.121662	1 20	**0.037171**
2 22	0.127012	2 27	0.150389	2 27	0.142716	1 20	**0.044717**
2 32	0.146543	2 39	0.211334	2 39	0.170653	1 20	**0.043358**
2 42	0.170200	2 50	0.236887	2 50	0.200712	1 21	**0.045948**
2 50	0.191966	3 11	0.230883	3 11	0.214339	1 21	**0.046141**
3 9	0.212033	3 22	0.232350	3 22	0.253504	1 21	**0.046807**

Table 3.3. Experiment 2, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10765$, *GMRES*(50). The columns IT and T(s) contains respectively the average number of iterations and the average time needed to perform them for solving the αth linear system.

Again, the results with the unpreconditioned algorithm are omitted, because it never reaches convergence within the maximum number of allowed iterations.

As a last test for this set of parameters we consider same settings but *BiCGstab* instead of *GMRES;* see Table 3.4. The use of *BiCGstab* without

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
11.5	0.117478	11.5	0.027163	11.5	0.027324	11.5	**0.025539**
20.0	0.052809	20.5	0.045016	20.5	**0.043589**	12.5	0.048411
32.5	0.068980	36.5	0.078104	37.0	0.077960	15.0	**0.056692**
67.5	0.138580	71.5	0.148651	70.5	0.141385	16.0	**0.061397**
103.0	0.215669	113.0	0.235227	119.5	0.235002	17.5	**0.065243**
154.0	0.311725	185.5	0.387039	185.5	0.373679	17.5	**0.051079**
231.5	0.474620	288.5	0.587745	291.0	0.594474	18.5	**0.069581**
115.0	0.229996	357.5	0.731463	369.5	0.753368	18.5	**0.069533**
363.5	0.756622	486.0	0.988916	478.5	0.971904	18.5	**0.070102**
218.0	0.574992	308.0	0.630262	660.0	1.302695	19.5	**0.072888**
152.0	0.312507	250.0	0.492896	250.0	0.501580	18.5	**0.064389**

Table 3.4. Experiment 2, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10765$, *BiCGstab*. The columns IT and T(s) contains respectively the average number of iterations and the average time needed to perform them for solving the αth linear system.

preconditioners is not reported because it never converges.The fixed ILU preconditioners are again numerically singular.

Steady State Case

We consider now finite element approximation for the steady state equation

$$\begin{cases} -\nabla \cdot (a(x,y)\nabla u) + \mathbf{b}(x,y) \cdot \nabla u + c(x,y)u = f(x,y), & (x,y) \in \Omega, \\ u = 0, & (x,y) \in \partial\Omega. \end{cases}$$

where Ω is the domain whose boundary is parametrized by the curve

$$\begin{cases} x = \cos(t), \\ y = \sin(t)\sin^m(t/2). \end{cases} \cup \begin{cases} x = 0.01\cos(t), \\ y = 0.01\sin(t). \end{cases} \quad t \in [0, 2\pi].$$

An example of the mesh is reported in figure 3.2(b).

As a test functions for the FEM method we use the elements in

$$P2_h = \left\{ v \in H^1(\Omega) \,|\, \forall\, K \in \mathcal{T}_h,\ \ v|_K \in \mathbb{R}_2[x] \right\}.$$

We generate a couple of problems $\{A^{(0)}, \mathbf{b}^{(0)}\}$ and $\{A^{(1)}, \mathbf{b}^{(1)}\}$ for different coefficients functions. The generic matrix of the sequence $\{A^{(\alpha)}\}_{\alpha=0}^{1}$ is given by the convex combination of parameter $\alpha \in [0,1]$ of the $\{A^{(0)}, A^{(1)}\}$ matrices. The right hand sides are obtained in the same way. As for the previous set of experiments, the matrices are generated with the FreeFem++ software [156].

Experiment 3. We consider as a first experiment for the steady state case the following pair of coefficient functions

$$\begin{gathered} a_1(x,y) = x^2 + y^2, \quad \mathbf{b}_1(x,y) = (1/2x, -1/2\sin(2\pi y)), \\ c_1(x,y) = x + y, \quad f_1(x) = \cos(x); \\ a_2(x,y) = x^4 + y^4, \quad \mathbf{b}_2(x,y) = (1/2x^2\sin(4\pi x), -1/2y^2), \\ c_2(x,y) = cos(x) + sin(y), \quad f_2(x,y) = \exp(-x-y). \end{gathered}$$

We use *GMRES* without restart. The reference AINV preconditioners are computed with a drop tolerance of $\delta = 1e-2$. For what concerns the correction matrix Δ, we stress that again only the main diagonal is considered. The result of the experiment are reported in Table 3.5. Similarly to the other experiments, the behavior of the fixed ILU preconditioner is not reported due to factors numerically singular. Moreover, the *GMRES* algorithm without preconditioning stagnates in all cases.

As for the PDE experiment we test the strategy also with *GMRES*(50). The results for this experiment are collected in Table 3.6. As expected, also unpreconditioned *GMRES*(50) does not converge and ILU preconditioners do not work.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
30	0.060449	30	0.065464	30	0.064820	30	0.099999
631	14.214685	146	0.737450	146	0.716860	84	**0.383758**
796	20.459523†	166	0.891988	166	0.926938	70	**0.292455**
900	28.008585†	176	0.992743	176	1.092765	64	**0.193377**
965	31.440721†	184	1.074857	184	1.209516	57	**0.150901**
1000	34.477231†	190	1.063854	190	1.616214	56	**0.152377**
1000	34.813366†	193	1.188407	193	2.748226	58	**0.162440**
1000	33.785674†	195	1.316548	195	2.296641	59	**0.173655**
1000	32.593510†	197	2.153401	197	1.846188	59	**0.169026**
1000	36.562952†	199	1.725839	199	1.411787	59	**0.170680**
1000	32.182831†	200	1.333837	200	1.387623	59	**0.167210**

Table 3.5. Experiment 3, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *GMRES* algorithm. The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
1 30	0.080122	1 30	0.078660	1 30	0.070380	1 30	0.064500
24 47	2.803143	4 14	0.399624	4 14	0.605011	2 39	**0.243955**
37 38	6.054049	4 47	0.407366	4 47	0.882234	2 26	**0.185375**
46 35	7.011219	5 10	0.423589	5 10	1.007688	2 18	**0.177025**
60 26	6.506396	5 16	0.459732	5 16	0.940276	2 9	**0.131443**
58 10	10.064129	5 22	0.504838	5 22	0.816400	2 9	**0.137760**
59 13	6.901644	5 25	0.487150	5 25	0.815360	2 12	**0.136431**
63 18	8.833342	5 28	0.486202	5 28	0.865197	2 14	**0.148239**
83 3	10.995113	5 31	0.492352	5 31	0.688509	2 15	**0.244746**
70 15	10.177334	5 35	0.819401	5 35	0.573916	2 14	**0.167503**
100 16	14.269126	5 37	0.905483	5 37	0.568182	2 12	**0.184751**

Table 3.6. Experiment 3, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *GMRES*(50). The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
19.5	0.067411	19.5	0.051467	19.5	0.040739	19.5	0.038550
493.5	0.891831	74.5	0.184125	75.5	0.143949	43.5	0.104094
641.0	$1.155428^{\dagger}$	83.5	0.144690	83.5	0.158401	38.5	**0.089504**
845.5	$1.491812^{\dagger}$	98.0	0.181133	97.5	0.184459	35.0	**0.092599**
992.5	$1.752002^{\dagger}$	101.5	0.182306	97.5	0.173160	30.5	**0.070884**
891.5	$1.566276^{\dagger}$	107.0	0.193929	107.0	0.180993	30.5	**0.073284**
979.5	$1.639698^{\dagger}$	113.5	0.203736	114.0	0.215719	31.0	**0.080443**
999.0	$1.730491^{\dagger}$	103.5	0.172221	100.5	0.182073	35.0	**0.085624**
969.0	$1.725403^{\dagger}$	100.5	0.177965	100.5	0.179372	35.0	**0.090680**
985.0	$1.728930^{\dagger}$	115.0	0.199124	103.5	0.188824	37.0	**0.097700**
956.0	$1.693190^{\dagger}$	100.0	0.219341	100.0	0.167194	32.5	**0.079203**

Table 3.7. Experiment 3, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *BiCGstab*. The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

Finally, we test our preconditioning strategy by using *BiCGstab*. The results are collected in Table 3.7. Also nonpreconditioned *BiCGstab* stagnates in all the instances. Moreover, ILU preconditioners are again numerically singular.

Experiment 4. We consider the following coefficients function for the generation of the A_0 and A_1 matrices

$$\begin{aligned} a_1(x,y) &= \exp(-x^2 - y^2), \quad \mathbf{b}_1(x,y) = (1/2x^2, 1/2y^2), \\ c_1(x,y) &= \exp(x+y)\sin(x+y), \quad f_1(x) = x^2 + y^2; \\ a_2(x,y) &= x^4 + y^4, \quad \mathbf{b}_2(x,y) = (1/2x^2, -y\sin(y)), \\ c_2(x,y) &= \exp(-x-y)\cos(x+y), \quad f_2(x,y) = \exp(-x-y). \end{aligned}$$

The results of this experiment, obtained with the same settings of the preconditioner as in the previous experiment, are collected in Table 3.8. Also in this case results relative to ILU preconditioning are omitted for the same reason as the other cases, the same is obtained with the unpreconditioned *GMRES* algorithm, i.e., stagnation.

We tested also the underlying preconditioners with the same settings with *GMRES*(50) and collected the results in Table 3.9. As for all the other cases, there is no convergence both with the fixed ILU preconditioner or *GMRES*(50) without preconditioning.

As a final test, we consider the application of interpolated preconditioners with *BiCGstab*. The results are reported in Table 3.10. Also in this case *BiCGstab* without preconditioning reaches the maximum

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
1 29	0.055336	1 29	0.055269	1 29	0.068000	1 29	0.054568
1 1000	33.387865†	1 265	2.114993	1 265	2.061204	1 160	0.878736
1 1000	35.783919†	1 269	2.333016	1 269	2.285620	1 117	**0.556560**
1 1000	34.168806†	1 271	4.119048	1 271	2.717876	1 94	**0.429240**
1 1000	34.281963†	1 271	3.353541	1 271	4.158225	1 80	**0.307191**
1 1000	32.349127†	1 270	2.317685	1 270	3.172648	1 78	**0.246143**
1 1000	34.388641†	1 269	2.571918	1 269	2.277063	1 79	**0.266362**
1 1000	34.698645†	1 268	3.364838	1 268	2.908988	1 79	**0.286741**
1 1000	34.584848†	1 267	2.268857	1 267	2.739444	1 79	**0.283751**
1 1000	33.514831†	1 266	2.838103	1 266	2.229427	1 78	**0.250284**
1 1000	33.350699†	1 263	2.787934	1 263	2.105237	1 77	**0.266518**

Table 3.8. Experiment 4, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *GMRES*. The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
1 29	0.060891	1 29	0.065797	1 29	0.055385	1 29	0.066218
1000 50†	140.582902	10 43	1.160647	10 43	1.072561	6 21	**0.495410**
1000 50†	139.655607	10 22	1.085158	10 22	1.031818	4 1	**0.716037**
1000 50†	144.163840	10 12	1.053174	10 12	1.083582	3 29	**0.355628**
1000 50†	141.416805	10 23	1.211638	10 23	1.155820	2 38	**0.252275**
1000 50†	142.241942	10 24	1.142104	10 24	1.132278	2 45	**0.241513**
1000 50†	140.596667	9 43	1.063141	9 43	1.566531	2 46	**0.242411**
1000 50†	142.573760	9 47	2.073712	9 47	2.220387	2 45	**0.244345**
1000 50†	147.190427	9 10	1.774650	9 10	1.661911	2 45	**0.252679**
1000 50†	147.823975	10 37	1.844827	10 37	1.847715	2 47	**0.281788**
1000 50†	142.734801	10 45	1.742876	10 45	1.338020	2 44	**0.239700**

Table 3.9. Experiment 4, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *GMRES*(50). The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

AINV fixed		AINV updated		AINV interpolated		AINV interpolated 2	
IT	T(s)	IT	T(s)	IT	T(s)	IT	T(s)
19.0	0.036232	19.0	0.045473	19.0	0.039100	19.0	0.038763
†	†	212.5	0.394025	205.0	0.371115	112.5	**0.313655**
†	†	206.5	0.396116	204.0	0.387600	74.0	**0.195428**
†	†	213.5	0.393649	200.0	0.376831	59.5	**0.173430**
†	†	193.0	0.356738	205.0	0.370740	50.0	0.131997
†	†	218.0	0.384271	218.0	0.397479	51.0	**0.147157**
†	†	195.5	0.373446	195.5	0.407083	52.0	**0.136462**
†	†	202.5	0.357391	200.5	0.640379	58.5	**0.160359**
†	†	200.0	0.358550	221.5	0.698809	53.0	**0.153582**
†	†	201.5	0.365729	202.5	0.650513	55.0	**0.144541**
†	†	207.5	0.403811	207.5	0.666290	52.5	**0.141122**

Table 3.10. Experiment 4, $\alpha = 0 : 1e-1 : 1$, size of the matrix $n = 10054$, *BiCGstab*. The columns IT and T(s) contains respectively the number of iterations and the time needed to perform them for solving the αth linear system.

number of iterations without converging, while the application of the fixed ILU preconditioners presents the same issues as the fixed AINV preconditioner.

4

Approximation of Functions of Large Matrices

Another task in which the use of both approximate inverse preconditioners and their update is effective is the numerical evaluation of a function $\Psi(A) \in \mathbb{C}^{n\times n}$ of a matrix $A \in \mathbb{C}^{n\times n}$: this is ubiquitous in models for applied sciences. Functions of matrices are involved in the solution of ordinary, partial and fractional differential equations, systems of coupled differential equations, hybrid differential–algebraic problems, equilibrium problems, measures of complex networks and several other applications. Motivated by the variety of the involved problems, important advances in the development of numerical algorithms for matrix function evaluations have been presented over the years and a rich literature is devoted to this subject; see, e.g., [149, 158, 206, 243].

In [41] we considered the case of large matrices A. This is a challenging situation and, for the computation of $\Psi(A)$, the available literature offers few strategies.

The existing numerical methods for computing matrix functions can be broadly divided into three classes: those employing approximations of Ψ, those based on similarity transformations of A and matrix iterations. When the size n of the matrix argument A is very large, as for example when it stems from a fine grid discretization of a differential operator, similarity transformations and matrix iterations can be not feasible since their computational cost can be of the order of n^3 flops in general. We focus on approximations of $\Psi(A)$ of the form

$$f(A) = \sum_{j=1}^{N} c_j (A + \xi_j I)^{-1} \tag{4.1}$$

where scalars c_j and ξ_j can be complex and I is the $n \times n$ identity matrix. The above approach has been proven to be effective for a wide set of functions Ψ.

In general, computing (4.1) requires inverting several complex–valued matrices and, with the exception of lucky or trivial cases, if n

is large, then this can be computationally expensive. We propose to overcome this difficulty by approximating directly each term $(A+\xi_j I)^{-1}$ by the efficient updates of an inexact sparse factorization. We take inspiration by the preconditioners update discussed in Chapter 3 and Section 3.1, and we specialize them also for the complex cases as proposed in [31, Section 3].

Moreover, such strategy can be extended to the computation of the action of the matrix function on vectors, that is, to compute $\Psi(A)\mathbf{v}$ for a given vector $\mathbf{v}$. Vectors of this form often represent the solution of important problems. The simplest example is the vector $\exp(t_1 A)\mathbf{y}_0$ which represents the solution at a time t_1 of the differential equation $y'(t) = A\mathbf{y}(t)$ subject to the initial condition $\mathbf{y}(t_0) = \mathbf{y}_0$.

If the interest is just in obtaining the vector $\Psi(A)\mathbf{v}$ and not $\Psi(A)$, then *ad hoc* strategies can be applied as, for example, well known Krylov subspace methods [3, 111, 128, 161, 171, 208–210, 231, 243].

4.1 *Computing Function of Matrices*

Many different definitions have been proposed over the years for matrix functions. We refer to the book by Higham [158] for an authoritative introduction and references.

Here we make use of a definition based on the *Cauchy integral*: given a closed contour Γ lying in the region of analyticity of Ψ and containing the spectrum of A, $\Psi(A)$ is defined as

$$\Psi(A) = \frac{1}{2\pi i}\int_{\Gamma} \Psi(z)(zI - A)^{-1} dz. \tag{4.2}$$

Thus, any analytic function Ψ admits an approximation of the form (4.1). Indeed, the application of any quadrature rule with N points on the contour Γ, or a suitable parametrization of it, leads to an approximation as in (4.1).

In [149] the authors address the choice of the conformal maps for dealing with the contour Γ in case of special functions like A^α and $\log(A)$. In this setting A is a real symmetric matrix, whose eigenvalues lie in the interval $[m, M] \subset (0, \infty)$. The basic idea therein is to approximate the integral in (4.2) by means of the trapezoidal rule applied to a circle in the right half–plane surrounding $[m, M]$. Thus,

$$\Psi(A) \approx f(A) = \gamma A \operatorname{Im} \sum_{j=1}^{N} c_j (\xi_j I - A)^{-1} \tag{4.3}$$

where γ depends on m, M and a complete elliptic integral, while the ξ_j and c_j involve Jacobi elliptic functions evaluated in N equally spaced quadrature nodes. We refer to [149] for the implementation details and we stress that we make use of their results for our numerical tests. In particular, an error analysis is presented, whose main results are reported below.

Theorem 4.1 (Hale, Higham, and Trefethen [149]). *Let A be a real matrix with eigenvalues in $[m, M], 0 < m < M$, let Ψ be a function analytic in $\mathbb{C}\backslash(-\infty, 0]$ and let $f(A)$ be the approximation of the form reported in* (4.3). *Then*

$$\|\Psi(A) - f(A)\| = O(e^{-\pi^2 N/(\log(M/m)+3)}).$$

The analysis in [149] also applies to matrices with complex eigenvalues, provided that they lie near to the real axis.

An approximation like (4.1) can also derive from a rational approximation R_N to Ψ, given by the ratio of two polynomials of degree N, with the denominator having simple poles. A popular example is the Chebyshev rational approximation for the exponential function on the real line. This has been largely used over the years and it is still a widely used approach, since it guarantees accurate results even for low degree N, say $N = 16$. Its poles and residues are listed in [82], while in [73] the approximation error is analyzed and the following useful estimate is given

$$\sup_{x \geq 0} |\exp(-x) - R_N(x)| \approx 10^{-N}.$$

Another example is the diagonal Padé approximation to the logarithm, namely

$$\log(I + A) \approx f(A) = A \sum_{j=1}^{N} \alpha_j (I + \beta_j A)^{-1}. \tag{4.4}$$

This is the core of the `logm_pade_pf` code in the package by Higham [158] and we will use it in our numerical tests in Section 4.3. Unfortunately, as for every Padé approximant, formula (4.4) works accurately only when $\|A\|$ is relatively small, otherwise scaling-and-squaring techniques or similar need to be applied. The error analysis for the matrix case reduces to the scalar one, according to the following result.

Theorem 4.2 (Kenney and Laub [169]). *If $\|A\| < 1$ and $f(A)$ is defined as* (4.4) *then*

$$\|\log(I + A) - f(A)\| \leq |f(-\|A\|) - \log(1 - \|A\|)|.$$

In some important applications, the approximation of the matrix $\Psi(A)$ is not required and it is enough to compute the vector $\Psi(A)\mathbf{v}$ for a given vector $\mathbf{v}$. In this case, by using (4.1), we formally obtain the approximation

$$f(A)\mathbf{v} = \sum_{j=1}^{N} c_j (A + \xi_j I)^{-1}\mathbf{v}, \tag{4.5}$$

which requires to evaluate $(A + \xi_j I)^{-1}$ or $(A + \xi_j I)^{-1}\mathbf{v}$ for several values of ξ_j, $j = 1, \ldots, N$. Usually, if A is large and sparse or structured, then the matrix inversions in (4.5) should be avoided since each term $\mathbf{w}_j \equiv (A + \xi_j I)^{-1}\mathbf{v}$ is mathematically equivalent to the solution of the algebraic linear system

$$(A + \xi_j I)\, \mathbf{w}_j = \mathbf{v}. \tag{4.6}$$

4.2 *The Updating Technique*

Following the approach in Chapter 3, we consider here an incomplete factorization for A^{-1}, also called *seed preconditioner* P_0, in order to build up an approximate factorization (or, better saying, to approximate an incomplete factorization) for each factor $(A + \xi I)^{-1}$, as ξ varies. Thus, given

$$P_0 = \tilde{W}\, \tilde{D}^{-1}\, \tilde{Z}^H, \tag{4.7}$$

we use the information included in the component matrices of P_0, $\tilde{W}$, $\tilde{D}$, $\tilde{Z}$. Choosing to use a single reference preconditioner (4.7) we build a sequence of approximate factorization candidates with $\tilde{E}$, a sparsification of the nonsymmetric *real valued* matrix E given by $E = Z^H W$, as in (3.11). Observe that in this case Δ in (3.10) is given simply by $\Delta = I$, since we are dealing with *shifted* linear systems. Then the approximation of A_ξ^{-1} given by P_ξ is defined as

$$P_\xi = \tilde{W}\, \left(\tilde{D} + \xi\tilde{E}\right)^{-1} \tilde{Z}^H, \tag{4.8}$$

where we use the approximate inverse (4.8) both as a preconditioner for Krylov solvers and for approximating directly $(A + \xi_j\, I)^{-1}$. In particular, $f(A)$ is approximated by

$$\tilde{f}(A) = \sum_{j=1}^{N} c_j P_{\xi_j}. \tag{4.9}$$

On the other hand, if the entries of A^{-1} decay fast away from the main diagonal, then we can consider again the matrix function $g = [\cdot]_m$,

$$[\cdot]_m : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n},$$

extracting m upper and lower bands (with respect to the main diagonal, which is the 0-diagonal) of its matrix argument generating an (m, m)–*banded matrix*. As we have hinted in Section 3.1, a substantial saving can be made by approximating

$$[\tilde{Z}^H \ \tilde{W}]_m \text{ with } [\tilde{Z}^H]_m[\tilde{W}]_m, \quad m > 0,$$

with a reasonable quality of the approximation indeed, under suitable conditions and provided $m > 0$, the relative error

$$\frac{\|[\tilde{Z}^H \ \tilde{W}]_m - [\tilde{Z}^H]_m[\tilde{W}]_m\|}{\|\tilde{Z}^H \ \tilde{W}\|}$$

can be moderate in a way that will be detailed in the result below.

Let us state a pair of results that can be derived as corollaries of Theorem 2.15 from [89],

Theorem 4.3 (Bertaccini, Popolizio, and Durastante [41])**.** *Let $A \in \mathbb{C}^{n\times n}$ be a nonsingular (m, m)–banded matrix, with $A \in \mathscr{B}(l^2(S))$ and condition number $\kappa_2(A) \geq 2$. Then, by denoting with $b_{i,j}$ the i, j–entry of A^{-1} and with*

$$\beta = \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}\right)^{1/2m},$$

for all $\tilde{\beta} > \beta$, $\tilde{\beta} < 1$, there exists a constant $c = c(\tilde{\beta}, A) > 0$ such that

$$|b_{i,j}| \leq c\, \tilde{\beta}^{|i-j|},$$

with

$$c \leq (2m + 1)\frac{\kappa_2(A) + 1}{\kappa_2(A) - 1}\|A^{-1}\|\kappa_2(A) \leq 3(2m + 1)\|A^{-1}\|\kappa_2(A).$$

Proof. The desired result follows by Theorem 2.15 [89, Theorem 2.4] for a finite dimensional Hilbert space, using a (m, m)-banded matrix $A \in \mathscr{B}(l^2(S))$, i.e., a $2m$–banded matrix, and $\kappa_2(A) \geq 2$. Indeed, by direct application of the second part of Theorem 2.15 we obtain:

$$\begin{aligned} |b_{i,j}| \leq & (2m + 1)\lambda_1^{-2m}\|A^{-1}\|\kappa_2(A)\cdot \\ & \cdot \max\left\{1, \frac{1}{2}\left[\frac{1 + \kappa_2(A)}{\kappa_2(A)}\right]^2\right\}\left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}\right)^{\frac{|i-j|}{2m}}. \end{aligned}$$

Since $\lambda_1 = \left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^{\frac{1}{2m}}$, and $\kappa_2(A) \geq 2$, the maximum assumes the value 1, i.e., $\max\left\{1, \frac{1}{2}\left[\frac{1+\kappa_2(A)}{\kappa_2(A)}\right]^2\right\} = 1$, and therefore:

$$\begin{aligned}
|b_{i,j}| \leq & (2m+1)\frac{\kappa_2(A)+1}{\kappa_2(A)-1}\|A^{-1}\|\kappa_2(A)\left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^{\frac{|i-j|}{2m}} \\
\leq & 3(2m+1)\|A^{-1}\|\kappa_2(A)\left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^{\frac{|i-j|}{2m}} \\
\leq & c\tilde{\beta}^{|i-j|}.
\end{aligned}$$

□

The assumption on the condition number is very reasonable because there is no need at all to use preconditioning for the approximation of $f(A)\mathbf{v}$ for matrices with condition numbers below 2 and similar reasonings can be used for the approximation of $f(A)$.

We can note immediately that the results in Theorem 4.3, without suitable further assumptions, can be of very limited use because:

- the decay of the extradiagonal entries can be very slow, in principle arbitrarily slow;
- the constant c in front of the bound depends on the *condition number* of A and we are usually interested in approximations of A^{-1} such that their condition numbers can range from moderate to high;
- the bound is far to be tight in general. A trivial example is given by a diagonal matrix with entries $a_{j,j} = j$, $j = 1, \ldots, n$. We have $b_{i,j} = 0$, $i \neq j$ but of course $\kappa_2(A) = {}^{a_{n,n}}/_{a_{1,1}} = n$.
- If we take $m = n$ and n is very large, then $\tilde{\beta}$ must be chosen very near 1 and it is very likely that no decay can be perceptible with the bound in Theorem 4.3.

However, the issues presented here are more properly connected with the decay properties of the matrices Z, W (and therefore $\tilde{Z}$, $\tilde{W}$); see again the general discussion in Section 2.4.1. Using similar arguments as in Theorem 4.1 in [25], it is possible to state the following result.

Corollary 4.1 (Bertaccini, Popolizio, and Durastante [41])**.** *Let $A \in \mathbb{C}^{n\times n}$ be invertible, $A \in \mathcal{B}(l^2(S))$, and with its symmetric part positive definite. Then for all i, j with $j > i$, the entries $z_{i,j}$ in $Z = L^{-H}$ and $w_{i,j}$ in $W = U^{-1}$ satisfy the following upper bound:*

$$|z_{i,j}| \leq c_1\,\tilde{\beta}_1^{j-i}, \quad |w_{i,j}| \leq c_2\,\tilde{\beta}_2^{j-i}, \quad j > i$$

(note that $z_{i,j}$, $w_{i,j} = 0$ *for* $j \leq i$*), where*

$$0 < \tilde{\beta}_1,\ \tilde{\beta}_2 \leq \tilde{\beta} < 1$$

and c_1, c_2 *are positive constants,* $c_1, c_2 \leq c_3 \cdot \kappa_2(A)$.

Proof. The desired result follows by Theorem 4.3 and [25, Theorem 4.1]. Since $ZD^{-1/2} = L^{-T} = A^{-1}L$ and the fact that $l_{i,j} = 0$ for $i < j$ and $i - j > m$ we find that $z_{i,j} = \sum_{k=j}^{j+m-1} b_{i,k} l_{k,j}$ for each $i \leq j$. By using Theorem 2.15 we easily get that:

$$|z_{i,j}| \leq \sum_{k=1}^{j+m-1} |b_{i,k}||l_{k,j}| \leq c \sum_{k=1}^{j+m-1} \beta^{i-j} |l_{k,j}|.$$

Without loss of generality, we can assume that $\max_{i=1,\ldots,n} a_{i,i} = 1$ and thus $|l_{i,j}| \leq 1$ (otherwise, to enforce such condition, it is sufficient to divide A by its largest diagonal entry: β remains unchanged and c is replaced by another constant $\tilde{c}$). Therefore, we obtain, by a direct application of Theorem 4.3,

$$|z_{i,j}| \leq \tilde{c} \sum_{k=0}^{m} \tilde{\beta}^{j-i+k} = \tilde{c}\tilde{\beta}_1^{j-i} \sum_{k=0}^{m} \tilde{\beta}_1^k \leq c_1 \tilde{\beta}_1^{j-i}.$$

In conclusion the existence of c_3 and the bound for the $\tilde{\beta}_2$ follows by applying the same procedure to the matrix W. □

If the seed matrix A is, e.g., diagonally dominant, then the decay of the entries of A^{-1} and therefore of W, Z ($\tilde{W}$, $\tilde{Z}$) is faster and more evident. This can be very useful for at least two reasons:

- the factors $\tilde{W}$, $\tilde{Z}$ of the underlying approximate inverse in factorized form can have a narrow band for drop tolerances even just slightly more than zero;
- banded approximations can be used not only for post–sparsifying $\tilde{W}$, $\tilde{Z}$ in order to get more sparse factors, but also the update process can benefit from the fast decay.

We use the above properties in the following result in order to obtain an *a–priori* estimate of the error produced using a cheap estimate of the correction factors. Clearly, this is very important for the implementation of our strategy.

Theorem 4.4 (Bertaccini, Popolizio, and Durastante [41]). *Let $A \in \mathbb{C}^{n\times n}$ be invertible, $A \in \mathcal{B}(l^2(S))$, and with its symmetric part positive definite. Let $g_m = [\cdot]_m$ be a matrix function extracting the m upper and lower bands of its argument. Then, given the matrices from Corollary 4.1, we have*

$$[\tilde{Z}^H \tilde{W}]_m = [\tilde{Z}^H]_m[\tilde{W}]_m + E(A,m), \quad |(E(A,m))_{i,j}| \leq c_4\tilde{\beta}^{|i-j|}$$

where $c_4 = c_1c_2$.

Proof. For a fixed value of the size n, let m be the selected bandwidth, then we have:

$$([\tilde{Z}^H]_m[\tilde{W}]_m)_{i,j} = \sum_{k=\max\{1,\max\{i-m,j-m\}\}}^{\min\{n,\max\{i+m,j+m\}\}} (\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j}$$

$$([\tilde{Z}^H\tilde{W}]_m)_{i,j} = \begin{cases} \sum_{k=1}^{n}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j}, & |i-j| \leq k, \\ 0, & \text{otherwise.} \end{cases}$$

Let us assume that $m = 1$ (tridiagonal matrix), in this case the difference $E(A,m)$ can be expressed, by direct inspection, as:

$$\begin{aligned}([\tilde{Z}^H\tilde{W}]_1 - [\tilde{Z}^H]_1[\tilde{W}]_1)_{i,j} = &- \chi_{\max(1,i-1)\leq 1\leq j+1}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &- \chi_{\max(1,i-1)\leq 2\leq j+1}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &+ n\chi_{i\leq j+1}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j},\end{aligned}$$

where χ is a function on the i, j–indices that takes value 1 when i and j satisfy the condition written at its subscript and 0 otherwise. In the same way, if $m = 2$ (pentadiagonal matrix), then we find:

$$\begin{aligned}([\tilde{Z}^H\tilde{W}]_2 - [\tilde{Z}^H]_2[\tilde{W}]_2)_{i,j} = &- \chi_{\max(1,i-2)\leq 1\leq j+2}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &- \chi_{\max(1,i-2)\leq 2\leq j+2}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &- \chi_{\max(1,i-2)\leq 3\leq j+2}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &+ n\chi_{i\leq j+2}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j}.\end{aligned}$$

Thus we assume, as inductive hypothesis, that for $m < n/2$ we have:

$$\begin{aligned}(E(A,m))_{i,j} = &- \sum_{l=1}^{m+1} \chi_{\max(1,i-m)\leq l\leq j+m}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j} \\ &+ n\chi_{i\leq j+m}(\tilde{Z}^H)_{i,k}(\tilde{W})_{k,j}.\end{aligned} \tag{4.10}$$

The conclusion follows by induction over n (since the admissible values of m are bounded by $n/2$) and again direct inspection. In conclusion, the bound on the correction term is given by the triangle inequality and an application of Corollary 4.1 to (4.10). □

As a matter of fact, we see that a fast decay of entries of A^{-1} guarantees that the essential component of the proposed update matrix, i.e., $\tilde{E} = \tilde{Z}^H \cdot \tilde{W}$, can be cheaply, easily and accurately approximated by the product $[\tilde{Z}^H]_m[\tilde{W}]_m$, without performing the possibly time and memory consuming matrix–matrix product $\tilde{Z}^H \tilde{W}$.

On the other hand, if the decay of the entries of A^{-1} is fast, then a diagonal approximation of $\tilde{Z}^H \tilde{W}$ can be accurate enough. In this case, there is no need of applying the approximation in Theorem 4.4. The update matrix $\tilde{E}$ can be produced explicitly by the exact expression of $\mathrm{diag}(\tilde{Z}^H \tilde{W})$, as given in the following corollary.

Corollary 4.2 (Bertaccini, Popolizio, and Durastante [41])**.** *Let $A \in \mathbb{C}^{n\times n}$ be invertible, $A \in \mathcal{B}(l^2(S))$, and with its symmetric part (m, m)–banded and positive definite, $1 \le m \le n$. Then, the diagonal approximation for $\tilde{E}$ generated by the main diagonal of $\tilde{Z}^H \tilde{W}$ is given by*

$$\tilde{E} = \mathrm{diag}(\tilde{Z}^H \tilde{W}) = (d_{i,i}),$$

where

$$d_{i,i} = 1 + \sum_{j=1,\ i-j\le m}^{i-1} z_{j,i} w_{j,i}, \quad 1 \le i \le n.$$

Proof. The claimed thesis follows by induction on i and by the definition of banded matrix, the same technique as the proof of Theorem 4.4 is used, while observing that both $\tilde{Z}^H$ and $\tilde{W}$ have ones on the main diagonals. □

In all our numerical experiments we use the approximations proposed in Theorem 4.4 and in Corollary 4.2 without perceptible loss of accuracy; see Section 4.3.

4.3 *Numerical Examples*

The codes are in a prototype stage using Matlab R2016a in order to simplify changes and portability to more powerful platforms, so timings of our strategies can be surely improved, in particular in a parallel environment. The machine used is a laptop running Linux with 8 Gb memory and CPU Intel® Core™ i7-4710HQ CPU with clock 2.50 GHz.

The sparse inversion algorithm selected for each numerical test (those used here are described in Section 4.2) takes into account the choice made for the computation of the reference (or *seed* for short) preconditioners. If the matrix used to compute the seed preconditioner

is real, then we use the AINV. Otherwise, the *inversion and sparsification of the ILUT algorithm* (INVT), requiring a dual threshold strategy, see [36] for details and a revisitation of AINV and INVT techniques. In Section 4.3.4 we give some details on the selection of the seed preconditioners. In the following, the symbols τ denotes drop tolerance for AINV while $\tau_{\tilde{L}}$, $\tau_{\tilde{Z}}$ the threshold parameter for ILU decomposition and for post–sparsification of the inverted factors of INVT; again see Section 2.4.2 and Section 2.4.3 for details and discussions about the implementation of these procedures [36]. At last, ε_{rel} denotes the standard relative (to a reference solution) error.

Other details on the parameters and strategies used are given in the description of each experiment.

4.3.1 Role of g and τ

For clarifying the role of the function g introduced in (3.11) and the drop tolerance τ for AINV, we compare the results of our *Update* approach to compute $\exp(A)$ with the built–in Matlab function `expm`. We use the expression in (4.1) for the Chebyshev rational approximation of degree $N = 16$ so that we can consider the approximation error negligible.

We consider the test matrix $A = (A)_{i,j}$ in [23] with entries

$$(A)_{i,j} = e^{-\alpha(i-j)}, \quad i \geq j, \quad (A)_{i,j} = e^{-\beta(j-i)} \quad i < j, \quad \alpha, \beta > 0. \tag{4.11}$$

This is an example of the so–called *pseudo-sparse* matrix, that is, a completely dense matrix with rapidly decaying entries. These matrices are usually replaced with banded matrices obtained by considering just few bands or by dropping entries which are smaller than a certain threshold. We sparsify this matrix by keeping only 15 off–diagonals on either side of its main diagonal. We consider $\alpha = \beta = 0.5$ and $\alpha = \beta = 1.2$ for a small example, namely 50×50, to make the presentation more clear. The approximation we refer to is (4.9), in which we let τ and g change, with effects on the factors $\tilde{Z}$, $\tilde{W}$ and $\tilde{E}$ in (3.11), respectively. The continuous curves in Figure 4.1 refer to the "exact" approach, that is, for $\tau = 0$ leading to dense factors W and Z. In the abscissa we report the number of extra-diagonals selected by g. Notice that both τ and $g = [\cdot]_m$ are important because even for $\tau = 0$ more extra-diagonals are necessary to reach a high accuracy. From the plots in Figure 4.1, we note that the loss of information in discarding entries smaller than τ cannot be recovered even if g extracts a dense matrix. In the left plot, for a moderate decay in the off-diagonals entries, a conservative τ is necessary to keep the most important information. On the other hand, when the decay is more evident, as in the right plot, a large τ is

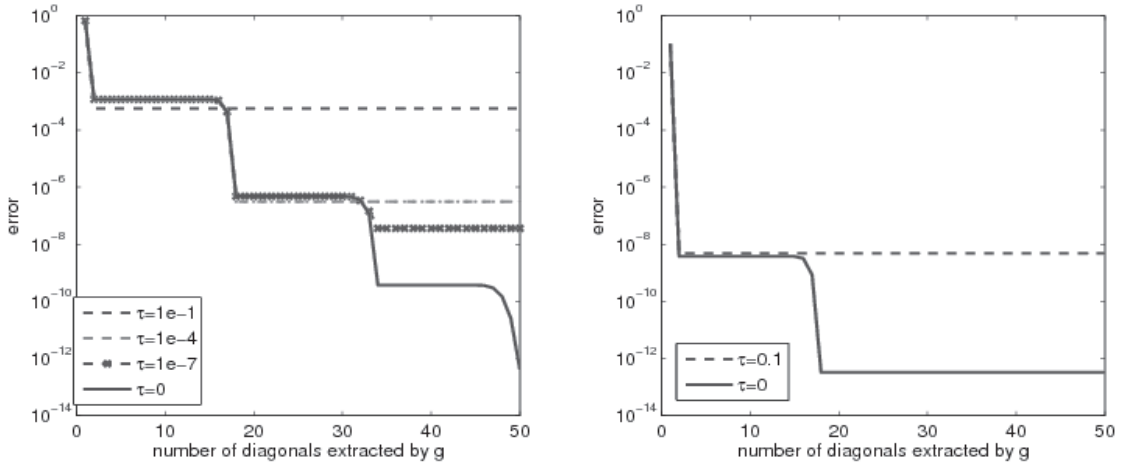


Figure 4.1. Behavior of the error for $\exp(A)$ as τ and g vary. The 50×50 matrix argument A has the expression in (4.11) with $\alpha = \beta = 0.5$ (left), $\alpha = \beta = 1.2$ (right). The x–axis reports the number of diagonals the function g selects while the y–axis reports the error with respect to the Matlab's `expm(A)`. AINV is used with the tolerance τ given in the legend.

enough, as a consequence, the choice of g keeping just two diagonals gives already a reasonable accuracy. We find similar results also for the logarithm, as well as for other input matrices.

4.3.2 Approximating $\Psi(A)$

Let us focus on the approximation of $\exp(A)$ and $\log(A)$. In the following tables, the columns *Update* refers to the approximation (4.9). Columns *Direct* are based on the direct inversion of the matrices $(A + \xi_j I)^{-1}$ in (4.1).

The fill–in for computing the incomplete factors for approximating the underlying matrices is computed as

$$\text{fill–in} = \frac{nnz(\tilde{Z}) + nnz(\tilde{W}) - n}{n^2}, \tag{4.12}$$

where n denotes the size of the underlying matrix and $nnz(\cdot)$ the number of its nonzero entries.

We consider the evaluation of $\log(A)$ where the entries of A are as in (4.11), with $\alpha = 0.2$ and $\beta = 0.5$ and n varies from 500 to 8000. For this matrix we use a drop tolerance $\tau = 0.1$ to compute a sparse approximate inverse factorization of A with AINV. The resulting factors $\tilde{Z}$ and $\tilde{W}$ are bidiagonal and thus we take $g(X) = X$. The inversion of the tridiagonal factors is the more demanding part of the *Update*

technique. For this test, we compare the *Update* and *Direct* methods, based on the approximation (4.3), with the Matlab function `logm` and the `logm_pade_pf` code in the package by N. Higham [158].

Numerical tests on scalar problems show that the degree $N = 5$ for the Padé approximation (4.4) and $N = 7$ for the approximant in (4.3) allow to reach a similar accuracy with respect to the reference solution. Thus, we use these values for N in our tests. Results in Table 4.1 show

n	Update	Direct	logm	logm_pade_pf	fill–in
500	1.53	1.33	13.05	**0.67**	6e-3
1000	4.90	4.69	44.40	**3.31**	3e-3
2000	**12.23**	13.86	407.28	38.67	1e-3
4000	**37.04**	56.23	6720.36	522.25	7e-4
8000	**168.41**	412.30	70244.41	6076.00	7e-4

Table 4.1. Execution time in seconds for $\log(A)$ for A as in (4.11) with $\alpha = 0.2$ and $\beta = 0.5$ as the dimension n varies, AINV with $\tau = 1e - 1$ is used.

that, for small examples, the *Update* and the `logm_pade_pf` approaches require a similar execution time, while the efficiency of the former becomes more striking with respect to all the others as the problem dimension increases.

We now consider the error for the matrix exponential. The test matrix is symmetric as in (4.11) for three choices of the parameter α. We analyze the error of the approximations provided by the *Update* and *Direct* methods, for the Chebychev rational approximation of degree $N = 8$, with respect to the results obtained by the `expm` Matlab command. We consider $\alpha = \beta = 1$, $\alpha = \beta = 1.5$, $\alpha = \beta = 6$. For the first two cases, the drop tolerance for the AINV is $\tau = 0.1$ and g extracts just the main diagonal and one superdiagonal. For the third case, AINV with $\tau = 10^{-3}$ is used and $\tilde{Z}$, $\tilde{W}$ are both diagonal. No matrix inversion is thus performed. Results from Table 4.2 show the good accuracy reached by using the *Update* approach. Indeed, although are present sparsification errors (see the action of τ and g), the error is comparable to the one of the *Direct* method, which does not suffer from truncation. For the case $\alpha = \beta = 6$, the difference between the two errors is more noticeable, but it has to be balanced with great savings in timings. Indeed, in this case the decay of the off–diagonal entries of the inverse of A is very fast and we exploit this feature by combining the effect of the small drop tolerance $\tau = 10^{-3}$ and a function g extracting just the main diagonal. Then, the computational cost is much smaller for the *Update* approach,

n	Update	Direct
500	1.1e-7	2.3e-8
1000	1.1e-7	2.3e-8
2000	1.1e-7	2.3e-8
4000	1.1e-7	2.3e-8

n	Update	Direct
500	2.3e-08	2.3e-08
1000	2.3e-08	2.3e-08
2000	2.3e-08	2.3e-08
4000	2.3e-08	2.3e-08

n	Update	Direct
500	4.5e-06	1.8e-08
1000	4.5e-06	1.8e-08
2000	4.5e-06	1.8e-08
4000	4.5e-06	1.8e-08

Table 4.2. Errors for the *Update* and *Direct* methods compared to the Matlab's `expm(A)`. The parameters are $\tau = 0.1$, $\tilde{Z}$ and $\tilde{W}$ bidiagonal, for $\alpha = \beta = 1$ (left) and $\alpha = \beta = 1.5$ (right); $\alpha = \beta = 6$, $\tau = 10^{-3}$ and $\tilde{Z}$ and $\tilde{W}$ are diagonal (bottom). AINV is used.

since no matrix inversion is explicitly performed and we experienced an overall linear cost in n, as in the other experiments. Thus, when a moderate accuracy is needed, the *Update* approach is preferable, since it is faster; see Table 4.3. We test our approach in the context of the

n	Update	Direct	`expm`	fill–in
500	**0.01**	0.06	1.97	2.0e-3
1000	**0.00**	0.01	6.19	1.0e-3
2000	**0.00**	0.01	30.52	5.0e-4
4000	**0.00**	0.06	172.89	2.5e-4
8000	**0.01**	0.10	910.16	1.3e-4

Table 4.3. Timings in seconds for $\exp(A)$ with A as in (4.11) with $\alpha = \beta = 6$, $\tau = 10^{-3}$ and g extracting just the main diagonal. The *fill–in* column refers to the fill–in occurred for computing the factors $\tilde{W}$ and $\tilde{Z}$ measured as in (4.12).

numerical solution of reaction–diffusion partial differential equations of the form

$$\partial_t u(x, y, z, t) = -k\nabla^2 u(x, y, z, t) + \gamma(x, y, z)u(x, y, z, t). \qquad (4.13)$$

Discretizing (4.13) with the space variables by second order centered differences, the reference solution can be computed by means of the matrix $\exp(A)$. We take $k = 1e - 8$, and the action of $\gamma(x, y, z)$ is given by the matrix–vector product on the semidiscrete equation between

G = sparsify(rand(n^3, n^3)), where n is the number of mesh points along one direction of the domain $\Omega = [0,1]^3$. Sparsify gives a sparse version of G with 0.1% of fill–in. The Laplacian is discretized with the standard 7-points stencil with homogeneous Dirichlet conditions, i.e., the semidiscrete equation reads as

$$\mathbf{u}_t(t) = (A + G)\mathbf{u}(t).$$

The results of this experiment are reported in Table 4.4. The reference matrix is computed by using the incomplete inverse LDU factorization (INVT) that needs two drop tolerances, $\tau_L = 1e-6$ and $\tau = \tau_Z = 1e-8$. The former is the drop tolerance for the ILU process and the latter for the post-sparsification of the inversion of LU factors respectively. A tridiagonal approximation of the correction matrix $E = Z^T W$ is used.

	Direct		Update		expm(A)	
n^3	T(s)	ε_{rel}	T(s)	ε_{rel}	T(s)	fill-in
512	0.15	2.85e-07	**0.07**	2.82e-07	0.92	100.00 %
1000	0.83	2.85e-07	**0.35**	2.83e-07	8.19	100.00 %
1728	4.28	2.85e-07	**0.94**	2.83e-07	46.23	92.40 %
4096	118.39	2.85e-07	**3.72**	2.84e-07	669.39	51.77 %
8000	834.15	2.85e-07	**9.69**	2.82e-07	4943.73	28.84 %

Table 4.4. Execution time in seconds for exp(A) and relative errors (ε_{rel}) with respect to `expm(A)` for A the discretization matrix of (4.13) (the time needed for building the reference matrix is not considered). INVT with $\tau_L = 1e-6$ and $\tau = \tau_Z = 1e-8$ is used.

4.3.3 Approximating $\Psi(A)\mathbf{v}$

To apply our approximation for $\Psi(A)\mathbf{v}$, where A is large and sparse (and/or possibly structured), we use a Krylov iterative solver for the systems $(A + \xi_j I)\mathbf{x} = \mathbf{v}$ in (4.5) with and without preconditioning (the corresponding columns will be labeled as *Prec* and *Not prec*). The iterative solvers considered are *BiCGstab* and CG (the latter for symmetric matrices). The preconditioner is based on the matrix $\tilde{W}(\tilde{D} + \xi_j \tilde{E})^{-1}\tilde{Z}^H$ as in (4.8). The entries of A are those reported in (4.11), while $\mathbf{v}$ is the normalized unit vector. The average of the iterates in Table 4.5 is much smaller when the preconditioner is used. Moreover, preconditioned iterations are independent on the size of the problem.

In Table 4.6 we report the error, with respect to the Matlab's `expm(A)`$\mathbf{v}$, of the approximations given by the *Prec* and *Not prec* options. The entries

	Prec		Not prec	
n	iters	T(s)	iters	T(s)
500	2	**0.05**	21	0.11
1000	2	**0.05**	19	0.18
2000	2	**0.08**	18	0.33
4000	2	**0.95**	17	2.96

Table 4.5. Iterates average and execution time in seconds for $\log(A)\mathbf{v}$ for A as in (4.11) with $\alpha = 0.2$, $\beta = 0.5$. The linear systems are solved with the Matlab's implementation of *BiCGstab* with and without preconditioning. INVT with $\tau = \tau_L = \tau_Z = 1e-1$ is used.

in the test matrix have so a fast decay, since $\alpha = \beta = 6$, that the term $\tilde{E}$ can be chosen diagonal. In this case we do not need to solve nontrivial linear systems. Interestingly, a good accuracy is reached with respect to the true solution. Moreover, the timings for the *Prec* approach is negligible with respect to that for the *Not prec*. Let us consider a series of tests

n	Prec	Not prec
	ε_{rel}	ε_{rel}
500	4.5e-06	1.8e-08
1000	4.5e-06	1.8e-08
2000	4.5e-06	1.8e-08
4000	4.5e-06	1.8e-08

Table 4.6. Error for $\exp(A)\mathbf{v}$ for A as in (4.11) with $\alpha = \beta = 6$. *Prec*: our technique with $\tilde{E}$ diagonal. *Not prec.*: (4.5) when the linear systems are solved with the Matlab's *PCG* used without preconditioning. INVT with $\tau = \tau_L = \tau_Z = 1e-1$ is used.

matrices of a different nature: infinitesimal generators, i.e., transition rate matrices from the *MARCA* package by Stewart [268]. They are based on large non–symmetric ill–conditioned matrices whose condition number ranges from 10^{17} to 10^{21} and their eigenvalues are in the square in the complex plane given by $[-90, 5.17e-15] \times i[-3.081, 3.081]$. As a first example, we consider the *NCD* model. It consists of a set of terminals from which the same number of users issue commands to a system made by a central processing unit, a secondary memory device and a filling device. In Table 4.7 we report results for various n, obtained by changing the number of terminals/users. The considered matrices A are used

to compute $\exp(A)\mathbf{v}$, $\mathbf{v} = [1, 2, \ldots, n]/n$. We compare the performance of *BiCGstab* for solving the linear systems in (4.5) without updated preconditioner and with our updating strategy, where g extracts only the main diagonal and the INVT algorithm with $\tau_Z = 1e-4$ and $\tau_L = 1e-2$ is used. The comparison is made in terms of the time needed for solving each linear system, i.e., the global time needed to compute $\exp(A)\mathbf{v}$. Both methods are set to achieve a relative residual of 10^{-9} and the degree of the Chebyshev rational approximation is $N = 9$. The column ε_{rel} contains the relative error between our approximation and $\text{expm}(A)\mathbf{v}$ over the norm of the value computed by Matlab's built-in routine. For the case with the highest dimension, expm gives out of memory error.

	Not prec		Update		
n	iters	T (s)	iters	T (s)	ε_{rel}
286	7.50	7.62e-03	7.50	**7.60e-03**	8.73e-09
1771	17.60	**2.95e-02**	17.60	3.00e-02	3.46e-07
5456	29.00	**1.15e-01**	29.00	1.18e-01	5.23e-06
8436	34.50	2.44e-01	28.00	**1.67e-01**	1.50e-05
12341	43.10	3.39e-01	33.20	**2.68e-01**	3.87e-05
23426	64.30	9.66e-01	42.30	**6.26e-01**	†

Table 4.7. Approximation of $\exp(A)\mathbf{v}$, A from NCD queuing network example. *BiCGstab*, $N = 9$, tol= $1e-9$, INVT algorithm with $\tau_Z = 1e-4$ and $\tau_L = 1e-2$ is used. A † is reported on the ε_{rel} when the expm gives out of memory error and no reference solution is available.

We consider computing $\exp(A)\mathbf{v}$ for the matrix TSOPF_FS_b9_c6 of dimension 14454 coming from [86], results are reported in Table 4.8 and Figure 4.2. For this case we do not have a reference for the error because Matlab's `expm` gives out of memory error. Instead, we consider the norm 2 of the difference of the solutions obtained for consecutive values of N for $N = 6, \ldots, 30$. The other settings for the solver remain unchanged in order to evaluate the efficiency of the algorithm for the same level of accuracy, i.e., we are using again the INVT algorithm with $\tau_Z = 1e-4$ and $\tau_L = 1e-2$. We observe two different effects for higher degree of approximations in Table 4.8. On one hand, from Figure 4.2, the relative error is reduced, as expected from the theoretical analysis, while, on the other, it makes the shifted linear system more well–conditioned. Note that the gain obtained using our preconditioning strategy is noticeable even for large matrices.

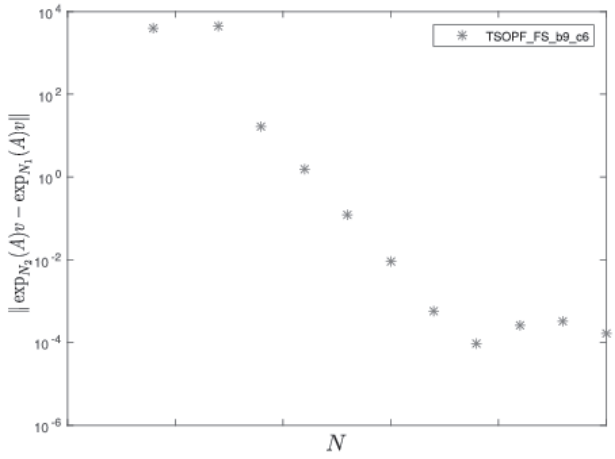


Figure 4.2. Accuracy for various values of N for the TSOPF_FS_b9_c6 matrix

Let us take the following A from [195]:

$$a_{i,j} = \frac{1}{2 + (i-j)^2}, \tag{4.14}$$

in order to approximate $\log(A)\mathbf{v}$, $\mathbf{v} = (1, 1, \ldots, 1)^T$. A is symmetric positive definite with a minimum eigenvalue of the order of 10^{-2} and its entries decay polynomially. We approximate $\log(A)\mathbf{v}$ with (4.5); *BiCGstab* is used with the update strategy and without it (Not prec), the seed is computed with INVT with $\tau_Z = 1e-1$ and $\tau_L = 1e-2$, we also include the results with Matlab's $\mathrm{logm}(A)\mathbf{v}$. In particular, we use $N = 30$ for the approximation of the logarithm function. Results are collected in Table 4.9.

Finally, we consider some matrices from *The University of Florida Sparse Matrix Collection* (see [86]), focusing on INVT with a seed preconditioner with $\tau_Z = 1e-1$ and $\tau_L = 1e-2$. The results are collected in Table 4.10, and they confirm what we observed in the other tests.

4.3.4 Choosing the Reference Preconditioner(s)

To generate a viable update (4.8), we need to compute an appropriate seed preconditioner (4.7). Note that the poles ξ_j in the partial fraction expansions (4.5) for the Chebyshev approximation of the exponential have a modulus that grows with the number of points; see, e.g., Figure 4.3. For this example the matrices from the *mutual exclusion model* in [268] are used. This is a model of M distinguishable processes or users that share a resource, but only M', with $1 \leq M' \leq M$ that could use it at the same time. The legend is thus reported as "mutex matx$M$

Matrix TSOPF_FS_b9_c6 Size: 14454, $\kappa_2(A)$ =3.1029e+12				
Not prec		Prec		
iters	T(s)	iters	T(s)	N
171.33	1.22e+00	15.00	**2.62e-01**	6
145.20	1.38e+00	36.50	**1.05e+00**	9
99.58	1.44e+00	8.75	**2.33e-01**	12
77.93	1.32e+00	7.64	**2.29e-01**	14
70.38	1.25e+00	7.06	**2.44e-01**	16
71.00	1.45e+00	6.61	**2.65e-01**	18
59.70	1.34e+00	6.15	**2.80e-01**	20
53.32	1.32e+00	5.95	**2.93e-01**	22
51.67	1.38e+00	5.75	**3.14e-01**	24
46.65	1.37e+00	5.58	**3.30e-01**	26
44.86	1.44e+00	5.39	**3.49e-01**	28
43.30	1.47e+00	5.20	**3.66e-01**	30

Table 4.8. Approximation of $\exp(A)\mathbf{v}$ as the degree N of the Chebyshev approximation varies. Timings and accuracy for TSOPF_FS_b9_c6 [86]. INVT algorithm with $\tau_Z = 1e-4$ and $\tau_L = 1e-2$ is used.

M'''. Therefore, we need to take into account the possibility that the resolvent matrices

$$(\xi_j I - A)^{-1}$$

become diagonally dominant or very close to the identity matrix, up to a scalar factor, or in general with a spectrum that is far from the one of $-A$. Sometimes the underlying matrices related to the resolvent above can be so well conditioned that the iterative solver does not need any preconditioner anymore. In this case, any choice of the seed preconditioner as an approximate inverse of the $-A$ matrix is almost always a poor choice, and thus also the quality of the updates; see [20, 31]. In Figure 4.3 (right) we report the mean iterations required when the P_{seed} corresponding to ξ_j is used for $j = 1, \ldots, N$, while $j = 0$ refers to the seed preconditioner for $-A$, all obtained with INVT for $\tau_L = 1e-5$ and $\tau_Z = 1e-2$. The plot clearly confirms that working with $-A$ is always the most expensive choice, while better results are obtained for whatever pole and sometimes the pole with the largest modulus is slightly better than the others.

Observe also that in this way complex arithmetic should be used to build the approximate inverse of the matrix $(\xi_1 I - A)$, whose main diagonal has complex–valued entries.

BiCGstab	Not prec		Update			logm$(A)v$	
n	iters	T(s)	iters	T(s)	fill–in	T (s)	ε_{rel}
1000	11.88	2.4621e+00	5.07	1.2596e+00	3.16 %	**1.6922e-01**	1.91e-06
4000	11.05	3.4215e+01	4.73	1.6885e+01	0.80 %	**1.4778e+01**	1.54e-06
8000	10.58	1.3378e+02	4.53	**6.5783e+01**	0.40 %	1.1698e+02	1.66e-06
12000	10.32	2.9698e+02	4.38	**1.4577e+02**	0.27 %	4.2827e+02	1.74e-06

Table 4.9. Computation of $\log(A)\mathbf{v}$ with A as in equation (4.14). Note the moderate decay and a spectrum that ranges in the interval $[6e-2, 3]$. For INVT $\tau_Z = 1e-1$ and $\tau_L = 1e-2$ are used.

BiCGstab		Not prec		Update			logm$(A)v$	
Name	n	iters	T(s)	iters	T(s)	fill–in	T(s)	ε_{rel}
1138_bus	1138	198.93	1.36e+00	31.18	4.01e-01	0.84 %	**2.27e-01**	4.41e-07
Chem97ZtZ	2541	27.98	3.44e-01	6.43	**1.23e-01**	0.10 %	3.54e+00	1.87e-07
bcsstk21	3600	157.85	4.76e+00	76.43	**3.16e+00**	1.36 %	1.00e+01	3.10e-07
t2dal_e	4257	232.00	4.21e+00	98.90	**1.78e+00**	0.02 %	2.58e+00	6.82e-04
crystm01	4875	23.35	1.03e+00	11.48	**5.64e-01**	0.17 %	2.53e+01	3.16e-07

Table 4.10. Approximation of $\log(A)\mathbf{v}$ with A SPD from The University of Florida Sparse Matrix Collection. The real parts of the eigenvalues are all in the interval $[2.324e-14, 1.273e+08]$. For INVT $\tau_Z = 1e-1$ and $\tau_L = 1e-2$ are used.

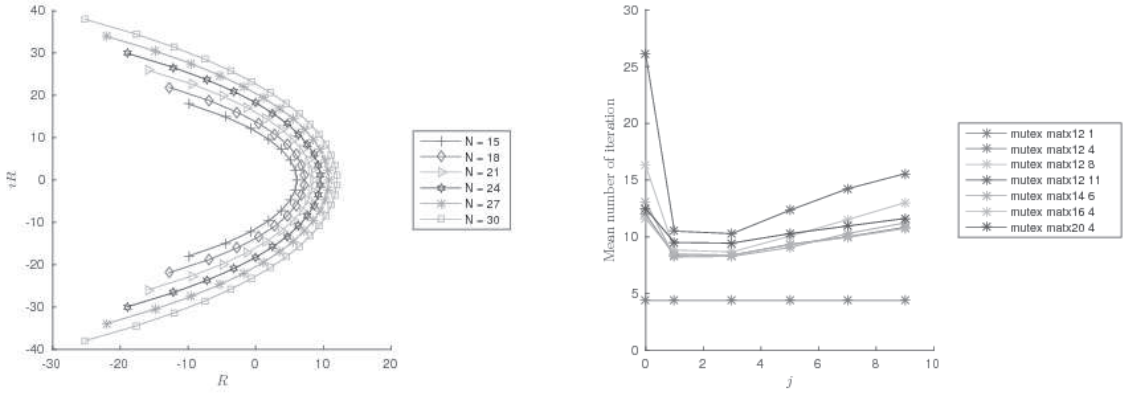


Figure 4.3. Position of the poles of Chebyshev approximation of exp (left) and a sample of mean number of iterations for different choice of P_{seed} for the *mutual exclusion model* from [268].

4.3.5 $\Psi(A)\mathbf{v}$ With Updates and With Krylov Subspace Methods

A popular class of effective algorithms for approximating $\Psi(A)\mathbf{v}$ for a given matrix A relies on Krylov subspace methods. The basic idea is to project the problem into a smaller space and then to make its solution potentially cheaper. The favorable computational and approximation properties have made the Krylov subspace methods extensively used; see, among the others, [161, 194, 208, 243].

Over the years some tricks have been added to these techniques to make them more effective, both in terms of computational cost and memory requirements, see, e.g., [3, 111, 171, 209, 210, 231]. In particular, as shown by Hochbruck and Lubich [161], the convergence depends on the spectrum of A. For our test matrices the spectrum has a moderate extension in the complex plane. Thus, the underlying Krylov subspace techniques for approximating $\Psi(A)\mathbf{v}$ can be appropriate.

The approximation spaces for these techniques are defined as in (2.9) as

$$\mathcal{K}_m(A, v) = \text{span}\{\mathbf{v}, A\mathbf{v}, \dots, A^{m-1}\mathbf{v}\}.$$

Since, as we have seen in Section 2.1, the basis given by the vectors $\mathbf{v}, A\mathbf{v}, \dots, A^{m-1}\mathbf{v}$ can be very ill–conditioned, one usually applies the modified Gram-Schmidt method (Algorithm 2.4) to compute an orthonormal basis with starting vector $\mathbf{v}_1 = \mathbf{v}/\|\mathbf{v}\|$. Thus, if these vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ are the columns of a matrix V_m and the upper Hessenberg matrix H_m collects the coefficients $h_{i,j}$ of the orthonormalization process, then the following Arnoldi formula holds

$$AV_m = V_m H_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^T,$$

with $\mathbf{e}_m$ denoting the mth column of the identity matrix. An approximation to $\Psi(A)\mathbf{v}$ can be obtained as

$$\mathbf{y}_m = \|\mathbf{v}\| V_m \Psi(H_m)\mathbf{e}_1.$$

The procedure reduces to the three-term Lanczos recurrence when A is symmetric, which results in a tridiagonal matrix H_m. One has still to face with the issue of evaluating a matrix function. If $m \ll n$, for the much smaller matrix argument size H_m, which is $m \times m$ then several approaches can be tried, for example the built–in function `funm` in Matlab, based on the Schur decomposition of the matrix argument, and the Schur–Parlett algorithm to evaluate the function of the triangular factor [158].

We consider the application of our strategy for the computation of $\exp(A)\mathbf{v}$, with a matrix A generated from the discretization of the following 2D advection-diffusion problem

$$\begin{cases} u_t = \dfrac{\partial}{\partial x} k_1 \dfrac{\partial u}{\partial x} + \dfrac{\partial}{\partial y} k_2(y) \dfrac{\partial u}{\partial y} + \dots \\ \qquad \dots + t_1(x) \frac{\partial u}{\partial x} + t_2(y), & x \in [0,1]^2, \\ \frac{\partial u}{\partial y} u(x,y,t) = 0, & x \in \partial[0,1]^2, \\ u(x,y,0) = u_0(x,y), \end{cases} \tag{4.15}$$

where the coefficients are $k_1 = 1e-2$, $k_2(x) = 2 + 1e-5\cos(5\pi x)$, $t_1(x) = 1 + 0.15\sin(10\pi x)$ and $t_2(x) = 1 + 0.45\sin(20\pi x)$, while second order centered differences and first order upwind are used to discretize the Laplacian and the convection, respectively. The purpose of this experiment, whose results are reported in Table 4.11, is comparing our updating approach, using INVT with $\tau_L = 1e-5$ and $\tau_Z = 1e-2$, with a Krylov subspace method. For the latter we use the classical stopping criterion based on monitoring

$$\gamma = h_{m+1,m} |e_m^T \exp(H_m) e_1|.$$

We stop the iteration when γ becomes smaller than 10^{-6}. The threshold γ was tuned to the accuracy expected by the *Update* approach.

From these experiences, we can conclude that our techniques, under appropriate hypotheses of sparsity or locality of the matrices, seem to be of comparable performances to solve the $\Psi(A)\mathbf{v}$ problem.

Moreover, we can expect even more interesting performances when simultaneous computations of vectors such as $\{\Psi(A)\mathbf{w}_j\}_{j=1}^K$ are required. The latter induces another level of parallelism beyond the one that can be exploited in the simultaneous computation of the term of equation (4.5).

n	Update		Arnoldi	
	ε_{rel}	T(s)	ε_{rel}	T(s)
100	2.64e-06	6.74e-02	3.51e-06	**2.91e-02**
196	3.81e-06	**7.80e-02**	1.20e-06	1.09e-01
484	1.22e-08	**2.97e-01**	1.60e-08	5.23e-01
961	7.68e-07	**1.27e+00**	1.68e-07	2.70e+00

Table 4.11. Errors and execution time for $\exp(A)\mathbf{v}$ for A obtained as the finite difference discretization of (4.15). INVT with $\tau_L = 1e-5$ and $\tau_Z = 1e-2$ is used.

In particular, this can be true when K is large and each vector $\mathbf{w}_j$ does depend on the previous values $\mathbf{w}_i$ and $\Psi(A)\mathbf{w}_i$. In the above setting we can construct the factors once in order to reduce the impact of the initial cost of computing the approximate inverse factors. A building cost that can be greatly reduced by using appropriate algorithms and architectures; see [36] and the discussion in Section 2.4.

5

Sparse Preconditioner for Mixed Classical and Fractional PDEs

Partial differential equations provide tools for modelling phenomena in many areas of science. Nonetheless there exist phenomena for which this kind of modelling is not as effective. For example, the processes of anomalous diffusion, the dynamics of viscoelastic and polymeric materials; see [225, 267] for details and a list of further applications. Indeed, most of the processes associated with these have non–local dynamics, for which the use of *fractional partial derivatives* seems to be much more effective. For physical interpretation see, e.g., [227] and the discussion in Chapter 6 and Appendix A.

To deal with the simulation of these models, we use the matrix approach framework as suggested in [226, 228, 229] and briefly recollected in Appendix A.3. The goal consists in transferring computational techniques developed for ordinary partial differential equations to differential equations with fractional partial derivative, called also *fractional differential equations* or *FDEs* for brevity. In recent years there have been contributions in this field; see, e.g., [75, 187, 188, 196, 201, 292].

In [35] we proposed the use of a structural property of the fractional derivative known as the *short–memory principle* and that describes the memory properties of the fractional derivatives, i.e., it measures their non–locality; see the discussion in Section 5.1.1. Specifically, we are dealing with non–local operators, but their structure permits to observe a decay of correlations towards the extremes of the interval of integration. As observed in [188, Chap. 2.6], "Up to now, the short memory principal has not been thoroughly studied so is seldom used in the real applications". As an example, we can consider the predictor–corrector approach in [90] and the work in [230]. For the solution with Krylov subspace methods, strategies with approximate inverse preconditioners have been developed. The latter builds structured approximations of the inverse of the discretization matrix in the fashion

of an inverse circulant–plus–diagonal preconditioner, see the work in [212, 221], or Toeplitz–plus–diagonal multigrid and preconditioning in [213, 214].

The novelty of our proposal is mainly in the use of *short–memory principle* as a mean to generate sequences of approximations for the inverse of the discretization matrix with a low computational effort. With the support of this precious property, we can solve the underlying discrete problems effectively by the preconditioned Krylov iterative methods of Section 2.1. In this way, there is no loss of accuracy in the discretization of the differential model, because the decay properties of the operators are used to approximate their inverses. Use of this solution framework also allows to exploit strategies for updating approximate inverses from Section 4.2 to treat problems with coefficients varying over time, or to apply methods of integration with variable time step; see also [15, 20, 31, 33, 106]. Finally, we note that from the computational point of view this also allows the use of GPUs, for which these techniques have been recently specialized; see again the introduction to this issues in Section 2.4.3 and [36, 84, 85, 113] for the implementation details. In Section 5.3 we will discuss how to specialize this for our case.

5.1 *Matrix approach*

We are going to briefly recall the approximation of the fractional integral and differential operator in matrix form. To focus our analysis on the numerical linear algebra issues we have collected all the results and the discussion relative to Fractional Calculus in Appendix A.

According to [225], we start recalling the notation for the fractional operators we are interested in. From now on with the notation $\Gamma(\cdot)$ we mean the *Euler gamma function*, the usual analytic continuation to all complex numbers (except the non–positive integers) of the convergent improper integral function

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx. \tag{5.1}$$

Definition 5.1 (Fractional Operators)**.** *Given a function $y(t)$ we define*

***Fractional Integral**. Given $\alpha > 0$ and $a < b \in \mathbb{R} \cup \{\pm\infty\}$,*

$$J^{\alpha}_{a,x} y(x) = \frac{1}{\Gamma(\alpha)} \int_a^x (x-\xi)^{\alpha-1} y(\xi) d\xi. \tag{5.2}$$

***Riemann–Liouville**. Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m - 1 < \alpha \leq m$*

the left-side Riemann–Liouville fractional derivative reads as

$$ {}_{RL}D^{\alpha}_{a,x}y(x) = \frac{1}{\Gamma(m-\alpha)}\left(\frac{d}{dx}\right)^m \int_a^x \frac{y(\xi)d\xi}{(x-\xi)^{\alpha-m+1}}, \tag{5.3} $$

while the right-side Riemann–Liouville fractional derivative

$$ {}_{RL}D^{\alpha}_{x,b}y(x) = \frac{1}{\Gamma(m-\alpha)}\left(-\frac{d}{dx}\right)^m \int_x^b \frac{y(\xi)d\xi}{(\xi-x)^{\alpha-m+1}}. \tag{5.4} $$

Symmetric Riesz. *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m-1 < \alpha \leq m$ the symmetric Riesz derivative reads as*

$$ \frac{d^{\alpha}y(x)}{d|x|^{\alpha}} = \frac{1}{2}\left({}_{RL}D^{\alpha}_{a,x} + {}_{RL}D^{\alpha}_{x,b}\right). \tag{5.5} $$

Caputo. *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m-1 < \alpha \leq m$ the left-side Caputo fractional derivative reads as*

$$ {}_{C}D^{\alpha}_{a,x}y(x) = \frac{1}{\Gamma(m-\alpha)} \int_a^x \frac{y^{(m)}(\xi)d\xi}{(x-\xi)^{\alpha-m+1}}, \tag{5.6} $$

while the right-side

$$ {}_{C}D^{\alpha}_{x,b}y(x) = \frac{(-1)^m}{\Gamma(m-\alpha)} \int_x^b \frac{y^{(m)}(\xi)d\xi}{(\xi-x)^{\alpha-m+1}}. \tag{5.7} $$

Grünwald–Letnikov. *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m-1 < \alpha \leq m$ the left-side Grünwald–Letnikov fractional derivative reads as*

$$ {}_{GL}D^{\alpha}_{a,x}y(x) = \lim_{\substack{h\to 0 \\ Nh=t-a}} \frac{1}{h^{\alpha}} \sum_{j=0}^{N} (-1)^j \binom{\alpha}{j} y(x-jh), \tag{5.8} $$

while the right-side

$$ {}_{GL}D^{\alpha}_{x,b}y(x) = \lim_{\substack{h\to 0 \\ Nh=b-t}} \frac{1}{h^{\alpha}} \sum_{j=0}^{N} (-1)^j \binom{\alpha}{j} y(x+jh). \tag{5.9} $$

Generally speaking, the definitions given above are equivalent only for functions that are suitably smooth. Nevertheless, in some cases, relationships can be established between the fractional derivatives written in the above forms, see Appendix A.1 and [225] for a complete account.

To treat fractional differential equations, i.e., dealing with equations written in terms of the operators in Definition 5.1, we recall some matrix-based approaches. We can consider the one introduced in [226] and further generalized in [228, 229]. This method is based on a suitable matrix representation of discretized fractional operators in a way that is alike to the numerical differentiation for standard integer order differential equations.

Let us fix an interval $[a, b] \subseteq \mathbb{R}$, an order of fractional derivative α and consider the equidistant nodes of step size $h = (b-a)/N$, $\{x_k = a + kh\}_{k=0}^{N}$, with $x_0 = a$ and $x_N = b$. Then, for functions $y(x) \in \mathscr{C}^r([a, b])$ with $r = \lceil \alpha \rceil$ and such that $y(x) \equiv 0$ for $x < a$, we have

$$ {}_{\mathrm{RL}}D^{\alpha}_{a,x}\, y(x) = {}_{\mathrm{GL}}D^{\alpha}_{a,x}\, y(x), \quad {}_{\mathrm{RL}}D^{\alpha}_{x,b}\, y(x) = {}_{\mathrm{GL}}D^{\alpha}_{x,b}\, y(x). \tag{5.10} $$

Therefore, we can approximate both the left and right sided Riemann–Liouville derivatives with the truncated Grünwald–Letnikov expansion

$$ {}_{\mathrm{RL}}D^{\alpha}_{a,x_k}\, y(x) \approx \frac{1}{h^{\alpha}} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} y((k-j)x), \quad k = 0, \ldots, N, \tag{5.11} $$

$$ {}_{\mathrm{RL}}D^{\alpha}_{x_k,b}\, y(x) \approx \frac{1}{h^{\alpha}} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} y((k+j)x), \quad k = 0, \ldots, N. \tag{5.12} $$

In this way we can define the lower and upper Toeplitz triangular matrices, respectively

$$ B_L^{(\alpha)} = \frac{1}{h^{\alpha}} \begin{bmatrix} \omega_0^{(\alpha)} & 0 & 0 & \ldots & 0 \\ \omega_1^{(\alpha)} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \omega_N^{(\alpha)} & \omega_{N-1}^{(\alpha)} & \cdots & \omega_1^{(\alpha)} & \omega_0^{(\alpha)} \end{bmatrix}, \; B_U^{(\alpha)} = \left(B_L^{(\alpha)} \right)^T, \tag{5.13} $$

where the coefficients $\omega_j^{(\alpha)}$ are defined as

$$ \omega_j^{(\alpha)} = (-1)^j \binom{\alpha}{j}, \; j = 0, 1, \ldots, N. \tag{5.14} $$

To satisfy the semi–group property of the left-right Riemann–Liouville fractional derivatives, it is also needed that $y^{(k)}(a) = 0$ for the left, respectively $y^{(k)}(b) = 0$ for the right, for each $k = 1, 2, \ldots, r-1$. It is crucial to observe that there is a decay of the coefficients along the diagonals of the discretization matrices.

Proposition 5.1. *Given the discretization formula in equation* (5.13)*, the following decay rate for the coefficients holds*

$$1 < \alpha < 2, \qquad |\omega_j^{(\alpha)}| = O(1/|j|^{\alpha+1}), \qquad j \to +\infty. \tag{5.15}$$

This is a well know consequence of the asymptotic relation for the Gamma function in [286]:

$$\lim_{x\to+\infty} \frac{\Gamma(x+\alpha)}{x^\alpha \Gamma(x)} = 1, \quad \forall \alpha \in \mathbb{R}. \tag{5.16}$$

If we are interested in obtaining an even sharper bound for the constant, then the estimate for the sequence of real binomial coefficients in [177, Theorem 4.2] can be applied. The strategy we have described is a numerical scheme with accuracy $O(h)$.

With the same strategy, a further discretization is obtained, showing the same decaying property for the Symmetric Riesz fractional derivative.

Schemes with higher accuracy can also be derived by observing that

$$(1-z)^\alpha = \sum_{j=0}^{+\infty} \omega_j^{(\alpha)} z^j, \ z \in \mathbb{C}. \tag{5.17}$$

Therefore, as have been done in [196], by substituting the generating function of the 1st order one-side differences with the one of the desired order and posing

$$z = \exp(-\imath\theta),$$

we obtain the coefficients for the scheme of higher accuracy. We stress that all the procedure can be done automatically by using Fornberg algorithm [117] and FFTs; see Appendix A.3.1. What we need to observe is that also in this case we can state the following proposition.

Proposition 5.2. *Given a one-side finite difference discretization formula represented by the polynomial $p_q(z)$ of degree q, with an associated Fourier symbol $f(\theta) \triangleq p_q(\exp(-i\theta))$, and $\alpha \in (0,1) \cup (1,2)$. The coefficients for the discretization formula of ${}_{RL}D^\alpha_{x,a}$ are given by the Fourier coefficients $\{\omega_j^{(\alpha,q)}\}_j$ of the function $f(\theta)^\alpha$ and*

$$\begin{aligned} &0 < \alpha < 1, \quad \omega_j^{(\alpha,q)} = O(1/|j|^\alpha), \quad j \to +\infty, \\ &1 < \alpha < 2, \quad \omega_j^{(\alpha,q)} = O(1/|j|^{1+\alpha}), \quad j \to +\infty. \end{aligned}$$

Proof. The results follow by standard relations between Hölder continuity, regularity and Fourier coefficients. See [100, 225, Section 7.6], and Appendix A.3.1 for details. □

For our purposes it is enough to observe that also in this case the matrix shows a polynomial decay of the coefficients, as can be seen in Figure 5.1.

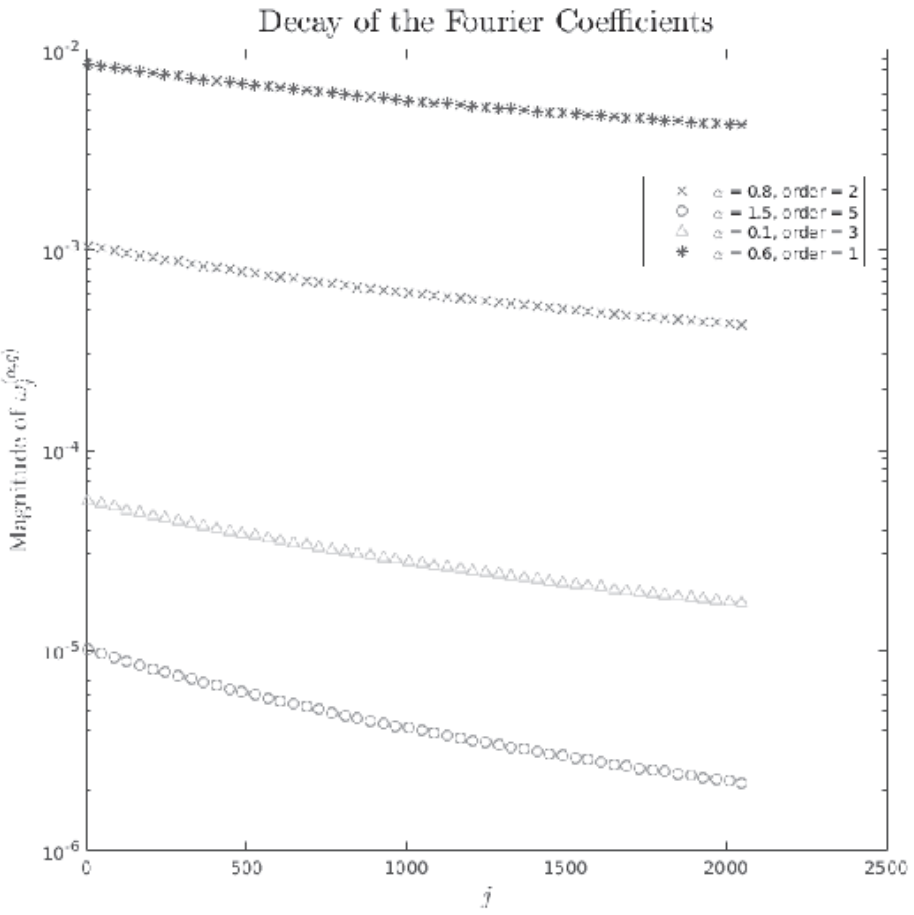


Figure 5.1. Decay of the Fourier coefficients as in Propositions 5.1 and 5.2

Another strategy to obtain methods with higher order of accuracy is using the *shifted Grünwald–Letnikov* approximation from [201]. For our purpose, it is enough to say that it consists in building matrices that are no more lower triangular, but with coefficients on the other diagonals obtained with the same approximations; see Definition A.13. Therefore, the decay of the entries is preserved with the same behavior displayed in Propositions 5.1 and 5.2.

To discretize symmetric Riesz fractional derivatives, other approaches can be also taken into account. We recall here only the so-called

central–fractional–difference approach from [220], with its further generalizations in [75]. By observing that for $\alpha \in (1,2]$ the Riesz fractional derivative operator of Definition 6.1 can be rewritten as

$$\frac{\partial^\alpha u(x)}{\partial |x|^\alpha} = -\frac{1}{2\cos(\alpha\frac{\pi}{2})\Gamma(2-\alpha)}\frac{d^2}{dx^2}\int_{\mathbb{R}}\frac{u(\xi)d\xi}{|x-\xi|^{\alpha-1}}, \tag{5.18}$$

the following $O(h^2)$ scheme, given by [220], can be obtained

$$\frac{\partial^\alpha u(x)}{\partial |x|^\alpha} = -\frac{1}{h^\alpha}\sum_{k=-\frac{b-x}{h}}^{\frac{x-a}{h}} g_k u(x-kh) + O(h^2),$$

with

$$g_k = \frac{(-1)^k\Gamma(\alpha+1)}{\Gamma(\alpha/2-k+1)\Gamma(\alpha/2+k+1)}. \tag{5.19}$$

The decay of the coefficients of the scheme from [220] have been proved with the same techniques as in Proposition 5.1; see also Proposition A.11.

Corollary 5.1. *For large values of k for the coefficients g_k of equation* (5.19) *we have*

$$g_k = O(1/|k|^{\alpha+1}), \qquad j \to +\infty.$$

Again, by Corollary 5.1 we infer the decay property we need for the following arguments. Indeed, discretizing the differential operator over the same uniform grid over $[a,b] \subseteq \mathbb{R}$ it is possible to restate the problem in symmetric Toeplitz matrix form:

$$O_N^{(\alpha)} = -\frac{1}{h^\alpha}\begin{bmatrix} \varsigma_0 & \varsigma_1 & \cdots & \varsigma_N \\ \varsigma_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \varsigma_1 \\ \varsigma_N & \cdots & \varsigma_1 & \varsigma_0 \end{bmatrix}, \quad \frac{\partial^\alpha u(x)}{\partial |x|^\alpha} = O_N^{(\alpha)}u(x_k) + O(h^2),$$

showing again the decay property we need.

We remark also that discretizations of higher order for the symmetric Riesz derivative were introduced and effectively applied in [94]. For our purposes, it is enough to note that the entries of the matrix form are weighted sums of the coefficients g_k of equation (5.19). Therefore, they show the same decay properties, although with coefficients of different magnitude.

The matrices generated by the Grünwald–Letnikov approximation and the central–fractional-differences share the same decay property along the diagonals. This feature depends on a structural property of the fractional derivatives. While the classical derivatives are local operators, the fractional derivatives and integral operator of Definition 5.1 are non–local. Again, as we have just observed, the role of the *history* of the behavior of the $y(x)$ function, when we go near to the starting or ending point has less importance: the *short–memory* principle.

5.1.1 The Short–Memory Principle

To introduce the *short–memory* principle we can follow the approach in [225, Section 7.3] defining a *memory length* L and then imposing the approximation

$$ {}_{\mathrm{RL}}D^{\alpha}_{a,x}y(x) \approx {}_{\mathrm{RL}}D^{\alpha}_{x-L,x}y(x), \quad x > a + L. $$

Therefore the error produced by zeroing out the entries of the matrix representing the operator is given by

$$ \begin{aligned} E(x) =& \left| {}_{\mathrm{RL}}D^{\alpha}_{a,x}y(x) - {}_{\mathrm{RL}}D^{\alpha}_{x-L,x}y(x) \right| \\ \leq& \frac{\sup_{x\in[a,b]} y(x)}{L^{\alpha}|\Gamma(1-\alpha)|}, \qquad\qquad a + L \leq x \leq b. \end{aligned} $$

Thus, fixed an admissible error ε, we obtain that

$$ L \geq \left(\frac{\sup_{x\in[a,b]} f(x)}{\varepsilon|\Gamma(1-\alpha)|} \right)^{\frac{1}{\alpha}} \quad \Rightarrow \quad \begin{aligned} &E(x) \leq \varepsilon, \\ &a + L \leq x \leq b. \end{aligned} $$

The underlying properties can be used for building a predictor–corrector approach for FDEs as in [90], or for applying truncation to reduce the computational cost for the exponential of the discretization matrix as in [230].

Differently from these approaches, we want to preserve all the information obtained by the discretization and use the short–memory principle, i.e., the decaying of the entries, for gaining information on the inverse of the discretization matrix.

As we have hinted in Section 2.4.1, classical results on the decays of the inverse of a matrix A, as in [89], have been proven to be useful in different frameworks. They have been generalized also to other matrix function than the simple $f(z) = z^{-1}$; see again Chapter 4 and references

therein. For our needs, we are going use the results in [166] that require that A is not sparse or banded, but having entries that show polynomial or exponential decay. This is exactly the case of our discretizations; see Figure 5.1 for the coefficients and Figure 5.2 for the decay of the inverse.

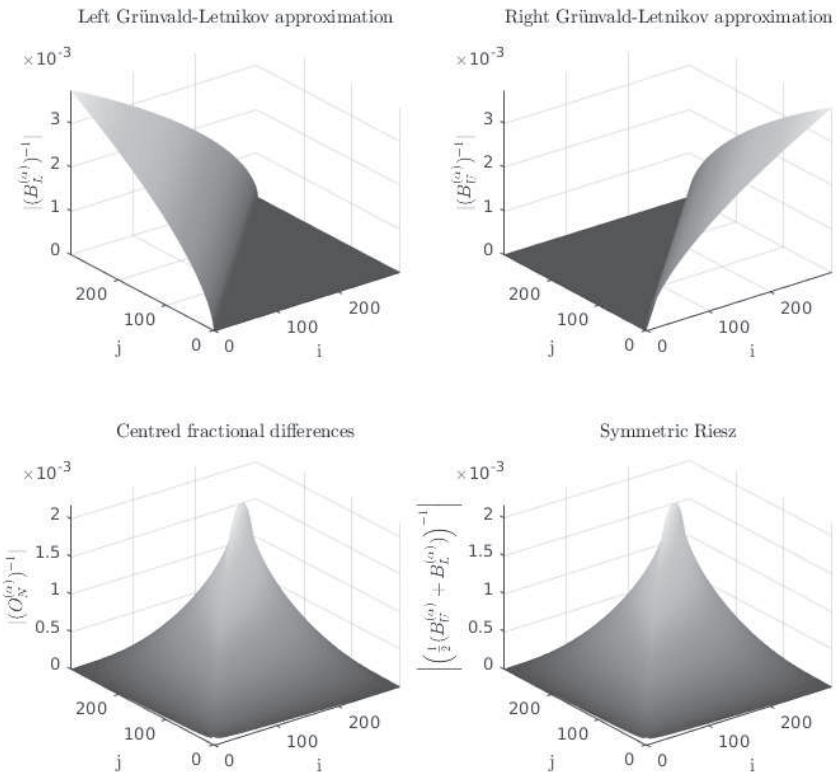


Figure 5.2. Decay of the inverse matrix relative to the various discretizations, $n = 300$ and $\alpha = 1.7$

We can now recall the result we have given in Theorem 2.17 from [166, 174] to check that the decay really exists.

Proposition 5.3. *Given the discretizations in Proposition 5.1, 5.2 or in Corollary 5.1, if* $\alpha \in (1, 2)$ *then the following relation holds*

$$\exists\, C > 0 \;:\; |(A^{-1})_{h,k}| = |\theta_{h,k}| \leq C(1 + |h - k|)^{-\alpha-1}, \tag{5.20}$$

while for $\alpha \in (0, 1)$ *we deduce that*

$$\exists\, C > 0 \;:\; |(A^{-1})_{h,k}| = |\theta_{h,k}| \leq C(1 + |h - k|)^{-\alpha}. \tag{5.21}$$

Proof. To prove the results, it is enough to observe that by equation (5.15) of Proposition (5.1), we infer that

$$\exists\, C > 0 \;:\; |(A)_{h,k}| = |a_{h,k}| = |\omega^{(\alpha)}_{|h-k|}| \leq C(1 + |h - k|)^{-\alpha-1}. \tag{5.22}$$

Therefore, by Theorem 2.17, the results hold. The bounds for the other discretizations are obtained in the same way. □

Observe that this result applies with a constant C that is *not independent* from h in the cases in which combinations of both left– and right–derivatives are used or in the Riesz setting. As Figure 5.2 clearly shows, where there is only one–sided derivative this is not the case. The crucial point is the invertibility in l^2 of the underlying infinite dimensional operator; see also Remark 2.8.

5.1.2 Multidimensional FPDEs

The other case we consider is the one of the multidimensional FPDEs. In the linear constant coefficients case over a Cartesian mesh the matrices of discretization can be written as a sum of Kronecker products of matrices that discretize the equation in one dimension.

Given two matrices $A, B \in \mathbb{R}^{n\times n}$, their Kronecker product $(C_{\alpha,\beta})_{\alpha,\beta} \triangleq (A_{i,j})_{i,j} \otimes (B_{k,l})_{k,l}$ is defined elementwise as

$$c_{\alpha,\beta} = a_{i,j}b_{k,l}, \quad \alpha = n(i-1)+k,\ \beta = n(j-1)+l, \quad 1 \le i,j,k,l \le n. \tag{5.23}$$

As a corollary of the previous, we can state the following result.

Proposition 5.4. *Given $A, B \in \mathbb{R}^{n\times n}$, $A = (a_{i,j})$, $B = (b_{i,j})$,*

$$|a_{i,j}| \le C_1(1+|i-j|)^{-s_1},\ |b_{i,j}| \le C_2(1+|i-j|)^{-s_2}, \tag{5.24}$$

and I the identity matrix of order n, we have

$$A \oplus B \triangleq A \otimes I + I \otimes B, \tag{5.25}$$

and there exist $C > 0$ such $s = \min\{s_1, s_2\} > 0$ and

$$|(A \oplus B)_{\alpha,\beta}| \le C(1+|\alpha-\beta|)^{-s}. \tag{5.26}$$

Proof. By Theorem 2.17 we know that the set of matrices whose entries show a polynomial decay is an algebra. Therefore, we only need to prove that $A \otimes I$ and $I \otimes B$ show the decaying property too. For $(I \otimes B)_{\alpha,\beta}$ we find simply that $(I \otimes B)_{\alpha,\beta} = \delta_{i,j}b_{k,l}$, where $\delta_{i,j}$ is the usual Kronecker delta. Then, we obtain the block matrix with n copies of the B matrix on the main diagonal. This implies that we have preserved the same decay property as the B matrix having added only zeros entries. On the other hand, for $(A \otimes I)$ we have

$$(A \otimes I)_{\alpha,\beta} = a_{i,j}\delta_{k,l},$$

that is the following block matrix

$$A \otimes I = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{bmatrix}, \quad A_{i,j} = a_{i,j} I. \tag{5.27}$$

Again, we can use the same decay bound for the matrix A, even if it is no more sharp because the values are now interspersed by diagonals of zero. □

Sharper bounds for this kind of structures have been obtained in [71]. Nevertheless, they refer to the case of banded matrices A, B and can become of interest when we consider equations with both fractional and classical derivatives.

Remark 5.1. *The decay of the entries and the strategy of dropping entries of prescribed small modulus in the inverse matrices can be applied also when specific forms of the short–memory principle have been used for approximating directly the system matrix.*

5.2 *Solution Strategies*

The observed decay of the entries for the inverse of the matrices that discretize the FPDEs allows us to devise an effective algorithm. The whole strategy is based on the possibility of discarding elements of prescribed small modulus in the calculation of an approximate inverse of the matrix of interest as discussed in Section 2.4.

On this basis, our proposal is to solve the discretized differential equations in matrix form, written in terms of the operators we introduced in Section 5.1. To this end, we use an appropriate Krylov subspace method, see, e.g., Section 2.1, with an approximate inverse preconditioner in factorized form discussed in Section 5.2.1. *Experiments 1,2* and *3* in Section 5.3 illustrate the application of our approach to few test problems.

On the other hand, if we have a good approximation of the inverse of the discretization matrix, then we can use it as a direct method for the solution of the given FPDE. In this way the solution procedure is reduced to compute an appropriate approximate inverse and to perform matrix–vector products.

For the second approach, we consider the solution of a pure fractional partial differential equation, i.e., without derivatives of integer order. Having discretized it in terms of the formulas in Section 5.1,

we find sequences of matrices, which, together with their inverses, share the decay property called *short–memory principle*. In practice, we approximate the underlying inverses with the approximate inverses in Section 5.2.1.

Similarly to the approach in [230], we are going to trade off accuracy and performance. Nevertheless, instead of discarding elements of the discretized operator and then solving the associated linear systems, we are going to act directly on the inverse of the operator, building a sort of direct method, see *Experiment 4* in Section 5.3.

Before trying both strategies on our test problems, we stress that we will focus only on the class of algorithms from [61] to compute the approximate inverse through the use of a *biorthogonalization* procedure (*conjugation* for the Hermitian case); see Section 2.4.3. Moreover, we will consider again the strategy from Section 4.2 for updating these approximations, in the case where the discretization of the equation depends on the time step, giving rise to a sequence of algebraic linear systems with variable coefficient matrices.

We stress that also other strategies, based on the use of approximate inverses architecture, have been used in literature. The one considering the solutions of Hermitian positive definite Toeplitz-plus-diagonal systems from [212], which is based on the exponential decay of the entries of the matrix combined with the use of approximate inverse circulant-plus-diagonal preconditioners. A similar idea, although specialized for the case of discretization of fractional diffusion equations, has been developed in [221]. In the latter the construction of an approximate inverse preconditioner is obtained by the circulant approximation of the inverses of a scaled Toeplitz matrix that is then combined together in a row-by-row fashion.

5.2.1 Approximate Inverse Preconditioners

Given a matrix A symmetric and positive definite, we are interested in the appropriate sparse approximation of the inverse of A in factored form as introduced in Section 2.4, thus:

$$A^{-1} \approx ZD^{-1}Z^T, \tag{5.28}$$

where the matrix Z is lower triangular. out without dropping and in exact arithmetic, we find $Z = L^{-T}$, where L the unit Cholesky factor of A. In general, we consider the sparse approximations for unsymmetric matrices as in (2.68).

We recall that there exist stabilized algorithms for computing approximate inverses for nonsymmetric matrices that are less prone to

breakdowns than others; see the discussion in Section 2.4 and [19] for a broad discussion on applications. Indeed, we recall that in general incomplete factorizations (ILUs; see [247]) are not guaranteed to be nonsingular, also for positive definite matrices. This issue holds also for efficient inverse ILU techniques from Section 2.4.2. In particular, as we hinted in Section 5.2, we will not use inverse ILU techniques for our test problems in Section 5.3, because of the frequent breakdowns and poor performances. In fact ILU factorizations do not preserve, in general, the decay observed in the discretization matrices.

To show that the decay allows to build approximate inverses significantly more sparse, we report in Table 5.1 the fill–in percentage of the approximate factorization for the test problem

$$\begin{cases} u_t = d_+(x)\,{}_{\mathrm{RL}}D^{\alpha}_{a,x}u + d_-(x)\,{}_{\mathrm{RL}}D^{\alpha}_{x,b}u + f. & x \in (a,b), \\ u(a,t) = u(b,t) = 0, & t \in [0,T], \\ u(x,0) = u_0(x), & x \in [a,b], \end{cases} \tag{5.29}$$

where the fractional derivative order is $\alpha \in (1,2)$, $f \equiv f(x,t)$ is the forcing term and $d_{\pm}(x)$ are two non–negative functions representing the variable diffusion coefficients.

The convergence properties of this approach will be discussed in the numerical experiment section.

Tolerance	nnz(W)	nnz(Z)	fill–in
5.00e-01	4094	4095	0.20%
1.00e-01	52253	51867	2.48%
1.00e-02	117294	116453	5.57%
1.00e-03	214242	208970	10.09%
1.00e-04	332837	319870	15.56%
1.00e-05	466732	443718	21.71%
1.00e-06	630214	594081	29.19%

Table 5.1. Ratio of fill–in for various drop tolerance for the Problem (5.29) with matrix size $n = 2048$ and $\alpha = 1.8$

5.2.2 Updating Factorizations for the Approximate Inverses

The discretization of time–dependent FPDEs produces a sequence of linear systems whose matrices are dependent on the time step as in Example 3.1. As we have discussed in Section 4.2, usually facing a sequence of linear systems with different matrices makes expensive to

rebuild a new preconditioner each time. On the other hand, reusing the same preconditioner can be not appropriate.

The update strategy we can consider in here is either the one developed in [20, 31] and further in [15] or the one we introduced in Section 4.2.

In this case we have a sequence of nonsymmetric matrices $\{A^{(k)}\}_{k=0}^{t}$, and we use $A^{(0)}$ as the *reference matrix*; see again Sections 3.1 and 4.2. Therefore, by mimicking the construction in Section 4.2, we consider again an initial approximation of the form given in (2.68) for the inverse of $A^{(0)}$ in factored form:

$$P^{(0)^{-1}} = WD^{-1}Z^T.$$

By writing each $A^{(k)}$, for $k \geq 1$, as

$$A^{(k)} = A^{(k)} - A^{(0)} + A^{(0)} = A^{(0)} + \Delta_k,\ \Delta_k \triangleq A^{(k)} - A^{(0)},$$

we build the updated preconditioner

$$A^{(k)^{-1}} \approx P^{(k)^{-1}} = W(D+E)^{-1}Z^T,\ E \triangleq g(Z^T\Delta_k W), \tag{5.30}$$

where g is the generic sparsification function from Section 3.1. We chose again the function $g = [\cdot]_m$ extracting a m–banded submatrix of its argument. With this choice, the formula (5.30) becomes

$$A^{(k)^{-1}} \approx P^{(k)^{-1}} = W(D+E)^{-1}Z^T,\ E \triangleq [Z^T\Delta_k W]_m. \tag{5.31}$$

In order to use the approximate inverses in a direct manner, rather than as a preconditioner for an iterative method, we consider exploiting them as the *true* inverses of the matrices of the previous steps, i.e., we approximate the solution of the linear system with just a matrix–vector product with them.

5.3 *Numerical Examples*

The numerical experiments are performed on a laptop running Linux with 8 Gb memory and CPU Intel® Core™ i7-4710HQ CPU with clock 2.50 GHz, while the GPU is an NVIDIA GeForce GTX 860M. The scalar code is written and executed in Matlab R2015a, while for the GPU we use C++ with Cuda compilation tools, release 6.5, V6.5.12 and the CUSP library [84].

We build our approximate inverses with the CUSP library [84] that implements the standard scaled Bridson algorithm for approximate

inverses in factorized form. This choice was made to exploit highly parallel computer architectures such as GPUs. We recall that in this kind of setting it is possible to implement efficiently matrix–vector multiplications, that are the numerical kernel of both iterative Krylov subspace and approximate inverse preconditioners considered here. Moreover, the implementation of the underlying approximate inverses relies on a left–looking variant of the *biorthogonalization* procedure in Algorithm 2.25 that, as observed in [36] and in Section 2.4.3, sometimes suffers less from pivot breakdown. In the proposed experiments, we use GPU only to build the preconditioners in order to emphasize the related performances. However, it is possible to implement also the solving phase on GPUs, i.e., the iterative methods for the underlying algebraic linear systems.

The proposed strategies are scalable also on architectures that use more than one degree of parallelism, i.e., using more than one GPU to further improve the speedups.

We start considering the following problem containing both integer and fractional derivatives

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} = & \nabla \cdot (< \mathbf{t}, \nabla u >) \quad (x,y) \in \Omega, t > 0, \\ & +\mathscr{L}^{(\alpha,\beta)} u, \\ u(x,y,0) = & u_0(x,y), \\ u(x,y,t) = & 0, \qquad\qquad (x,y) \in \partial\Omega, t > 0, \end{cases} \tag{5.32}$$

where the fractional operator is given by

$$\begin{aligned} \mathscr{L}^{(\alpha,\beta)}. \triangleq\ & d_+^{(\alpha)}(x,y)\, {}_{\mathrm{RL}}D_{a,x}^{\alpha} \cdot + d_-^{(\alpha)}(x,y)\, {}_{\mathrm{RL}}D_{x,b}^{\alpha} \cdot \\ & + d_+^{(\beta)}(x,y)\, {}_{\mathrm{RL}}D_{a,y}^{\beta} \cdot + d_-^{(\beta)}(x,y)\, {}_{\mathrm{RL}}D_{y,b}^{\beta} \cdot . \end{aligned} \tag{5.33}$$

and $\mathbf{t} = ((a(x), b(y))$. To discretize the problem, we consider the five points discretization of the Laplace operator combined with the shifted Grünwald–Letnikov approximation for the fractional one. As a time integrator, we choose the Backward–Euler method, by taking into account the behavior of the error of approximation and the overall stability of the method; see [201].

Experiment 1. As a first choice for the solution of problem (5.32), we consider the coefficients

$$\begin{aligned} a(x) =& 1 + 0.5\sin(3\pi x),\ b(y) = 1 + 0.7\sin(4\pi y) \\ d_+^\alpha(x,y,t) =& d_-^\alpha(x,y,t) = e^{4t}x^{4\alpha}y^{4\beta}, \\ d_+^\beta(x,y,t) =& d_-^\beta(x,y,t) = e^{4t}(2-x)^{4\alpha}(2-y)^{4\beta} \\ u_0(x,y) =& x^2y^2(2-x)^2(2-y)^2, \end{aligned} \tag{5.34}$$

over the domain $[0,2]^2$ and the time interval $t \in [0,2]$. In Table 5.2 we consider the use of the *GMRES*(50) algorithm with a tolerance of $\varepsilon = 1e-6$ and a maximum number of admissible iterations `MAXIT` = 1000. The approximate inverses are computed with a drop tolerance $\delta = 0.1$. The update formula (5.31) for the preconditioner is used with only a diagonal update, i.e., the function $g(\cdot) = [\cdot]_0$ extracts only the main diagonal and only one reference preconditioner is used; see Section 3.1. In Table 5.2 we report the average timings and the number of external/internal iterations. When a "†" is reported, at least one of the iterative solvers does not reach the prescribed tolerance in `MAXIT` iterations.

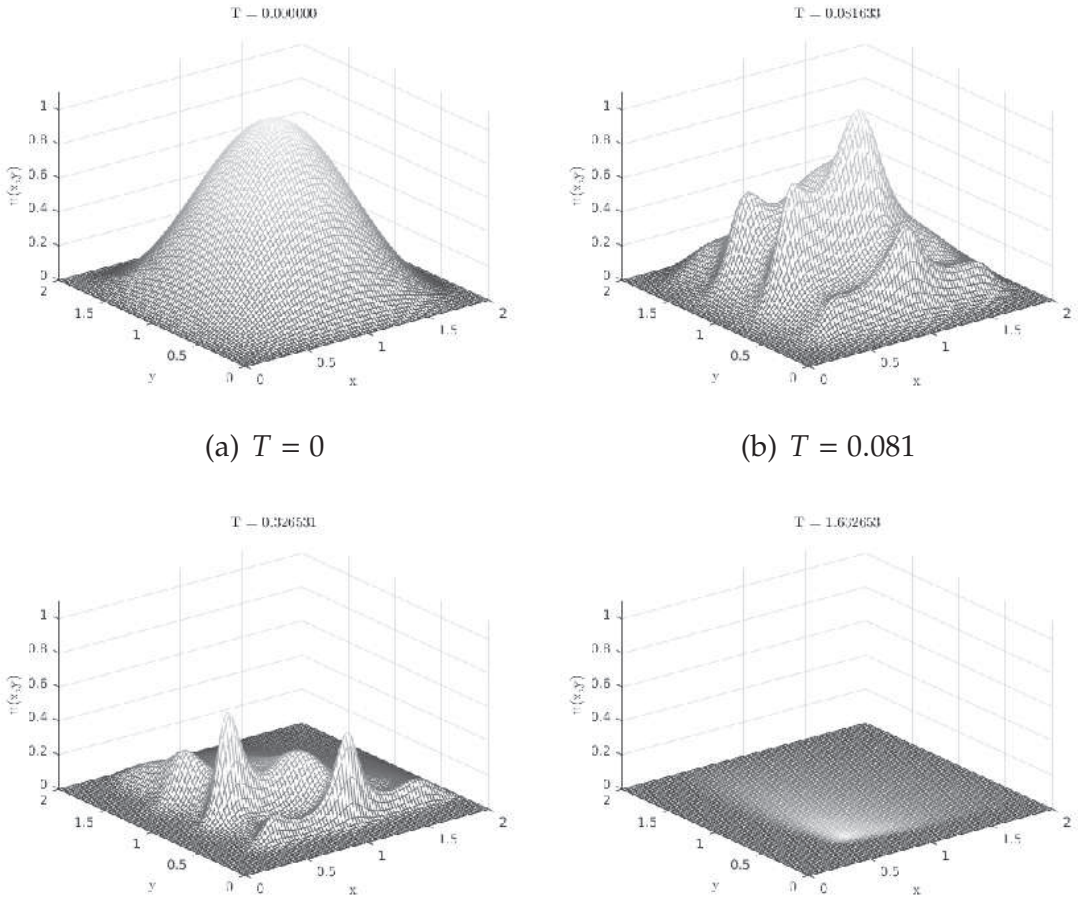
			GMRES(50)		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.6	1.2	5.38 %	†	†	4.45 26.24	7.68e-01	2.04 26.08	**2.95e-01**
1.3	1.8	6.51 %	†	†	4.55 27.57	7.79e-01	2.20 23.71	**3.09e-01**
1.5	1.5	4.41 %	†	†	4.43 28.04	7.61e-01	2.04 26.08	**2.93e-01**

Table 5.2. Test problem in (5.32). $Nx = Ny = 80$, $Nt = 50$

We report in Figure 5.3 the solution obtained at different time steps. We consider the solution of this problem also with *BiCGstab* and *GMRES* algorithms with the same settings. The results for this case are reported in Table 5.3 and Table 5.4, respectively.

			BiCGstab		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.6	1.2	5.38 %	†	†	134.78	5.74e-01	69.08	**2.94e-01**
1.3	1.8	6.51 %	†	†	127.18	5.38e-01	75.05	**3.21e-01**
1.5	1.5	4.41 %	†	†	123.46	5.08e-01	74.43	**3.09e-01**

Table 5.3. Test problem in (5.32). $Nx = Ny = 80$, $Nt = 50$



(a) $T = 0$ (b) $T = 0.081$

(c) $T = 0.327$ (d) $T = 1.632$

Figure 5.3. Plot of the solution of the problem (5.32) for $\alpha = 1.6$, $\beta = 1.2$. For a physical interpretation of this kind of problem refer to Appendix A.1.1.

Experiment 2. We now consider a slightly different model in which the coefficient function is used in divergence form

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} = & \nabla \cdot \big(a(x,y)\nabla u\big) \quad (x,y) \in \Omega, t > 0, \\ & +\mathscr{L}^{(\alpha,\beta)} u, \\ u(x,y,0) = & u_0(x,y), \\ u(x,y,t) = & 0, \qquad\qquad (x,y) \in \partial\Omega, t > 0. \end{cases} \tag{5.35}$$

The fractional operator $\mathscr{L}^{(\alpha,\beta)}$ is defined as in equation (5.33). We set $N_x = N_y = 80$ and $N_t = 70$ over the domain $\Omega = [0,2]^2$ and $T = [0,2]$. The same choice of the previous example have been done for the fractional diffusion coefficients, while various function have been tested for variable diffusion. We start with the *GMRES*(50) algorithm with a tolerance of $\varepsilon = 1e-6$. The drop tolerance for the approximate inverse preconditioners is set again to $\delta = 0.1$. Results for this case are collected

			GMRES		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.6	1.2	5.38 %	†	†	159.55	1.89e+00	70.12	**3.30e-01**
1.3	1.8	6.51 %	†	†	159.86	1.85e+00	71.12	**3.37e-01**
1.5	1.5	4.41 %	†	†	161.12	1.91e+00	70.08	**3.28e-01**

Table 5.4. Test problem in (5.32). $Nx = Ny = 80$, $Nt = 50$

in Table 5.5.

			GMRES(50)		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.2	1.8	5.96 %	†	†	2.93 31.77	4.89e-01	1.52 27.48	**1.94e-01**
1.5	1.5	2.74 %	†	†	2.90 30.36	4.55e-01	1.55 24.33	**1.88e-01**
1.8	1.8	4.27 %	†	†	3.65 23.93	5.82e-01	2.19 20.54	**2.90e-01**
1.8	1.3	5.21 %	†	†	3.10 30.87	5.05e-01	1.67 28.26	**2.25e-01**

Table 5.5. Test problem in (5.35). $Nx = Ny = 80$, $Nt = 70$ and $a(x, y) = \exp(-x^3 - y^3)$

			GMRES		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.8	1.8	4.27 %	†	†	130.94	1.18e+00	69.14	**3.16e-01**
1.8	1.3	5.21 %	†	†	114.45	9.56e-01	59.19	**2.55e-01**
1.5	1.5	2.74 %	†	†	107.43	9.30e-01	51.35	**2.24e-01**
1.2	1.8	5.96 %	†	†	111.30	9.95e-01	52.01	**2.22e-01**

Table 5.6. Test problem in (5.35). $Nx = Ny = 80$, $Nt = 70$ and $a(x, y) = \exp(-x^3 - y^3)$

Experiment 3. We now consider a different but related model problem: a time–dependent 2D mixed fractional convection–diffusion equation, where fractional diffusion is combined with classical transport. By using the same notation as the previous cases we write

$$\begin{cases} u_t(x, y, t) = \mathscr{L}^{(\alpha,\beta)}u + < \mathbf{t}, \nabla u >, & (x, y) \in [0, 2]^2,\ t \geq 0, \\ u(x, y, 0) = u_0(x, y), & \\ u(x, y, t) = 0, & (x, y) \in \partial[0, 2]^2\ \forall t \geq 0, \end{cases} \tag{5.36}$$

where $\mathbf{t}(x, y) = (t_1(x, y), t_2(x, y))$. Regarding the space variables, we discretize the fractional operator with shifted Grünwald–Letnikov

			BiCGstab)		FIXED PREC.		UPDATED PREC.	
α	β	fill-in	IT	T(s)	IT	T(s)	IT	T(s)
1.8	1.8	4.27 %	†	†	†	†	77.67	**3.34e-01**
1.8	1.3	5.21 %	†	†	116.08	4.97e-01	60.93	**2.63e-01**
1.5	1.5	2.74 %	†	†	117.43	4.96e-01	55.17	**2.32e-01**
1.2	1.8	5.96 %	†	†	107.55	4.65e-01	54.17	**2.36e-01**

Table 5.7. Test problem in (5.35). $Nx = Ny = 80$, $Nt = 70$ and $a(x, y) = \exp(-x^3 - y^3)$.

approximation and the transport term with standard first order centered finite differences. For the time approximation we use backward Euler method to be consistent with the approximation error and for the dominant diffusion behavior of the equation.

For this experiment, we chose the following variable (in space) coefficients for the fractional operator $\mathscr{L}^{(\alpha,\beta)}$

$$\begin{aligned} d_+^{\alpha}(x, y, t) =& d_-^{\alpha}(x, y, t) = x^{4\alpha} y^{4\beta}, \\ d_+^{\beta}(x, y, t) =& d_-^{\beta}(x, y, t) = (2 - x)^{4\alpha} (2 - y)^{4\beta}, \end{aligned} \tag{5.37}$$

while for the transport part we choose

$$t_1(x, y) = \frac{y + \beta}{x + y + \alpha}, \quad t_2(x, y) = \frac{x + \alpha}{x + y + \beta}, \tag{5.38}$$

with the usual initial condition $u_0(x, y) = x^2 y^2 (2 - x)^2 (2 - y)^2$. The *GMRES*(50) algorithm is used with a fixed approximate inverse preconditioner with drop tolerance $\delta = 0.1$. The results are reported in Table 5.8.

A similar behavior is obtained also with the *GMRES* algorithm, with the same overall settings and the same drop tolerance for the approximate inverse preconditioner. Results are reported in Table 5.9.

Finally, we consider the solution with the *BiCGstab*(2) algorithm [146], instead of the classical *BiCGstab*, to deal with the possibility of having a discretization matrix with non–real eigenvalues, eigenvalues that are not approximated well by the first order factors of the polynomials built by the standard *BiCGstab*. Tolerance for the method and drop-tolerance for the approximate inverse preconditioner are set to be the same as those for the other algorithms. Results are collected in Table 5.10.

			GMRES(50)		PRECONDITIONED	
α	β	fill-in	IT	T(s)	IT	T(s)
1.2	1.2	5.54 %	18.39 25.53	3.24e+00	2.37 24.39	**3.59e-01**
1.2	1.3	5.89 %	26.34 27.49	4.70e+00	2.34 25.95	**3.47e-01**
1.2	1.5	6.38 %	47.61 28.58	8.54e+00	2.34 22.10	**3.44e-01**
1.2	1.8	9.33 %	256.00 25.78	4.68e+01	2.00 18.20	**2.60e-01**
1.3	1.2	6.07 %	25.93 29.05	4.69e+00	2.00 22.27	**2.79e-01**
1.3	1.3	5.51 %	31.63 28.86	5.62e+00	2.47 27.49	**3.86e-01**
1.3	1.5	6.14 %	56.44 24.20	1.04e+01	2.39 31.37	**3.95e-01**
1.3	1.8	8.53 %	387.71 23.53	7.04e+01	2.22 21.69	**3.15e-01**
1.8	1.2	8.05 %	294.10 29.90	5.34e+01	1.56 26.95	**2.14e-01**
1.8	1.3	7.39 %	477.75 28.00	8.80e+01	1.61 29.17	**2.32e-01**
1.8	1.5	7.05 %	†	†	2.25 21.75	**3.20e-01**
1.8	1.8	7.18 %	†	†	3.71 28.61	**6.33e-01**

Table 5.8. Test problem in (5.36). $Nx = Ny = 80$, $Nt = 60$.

Experiment 4. We consider the following constant coefficients fractional diffusion equation from [292]

$$\begin{cases} u_t(x,t) = K\frac{\partial^\alpha}{\partial|x|^\alpha}u(x,t), & t \in [0,T],\ x \in [0,\pi], \\ u(x,0) = x^2(\pi - x), & \alpha \in (1,2), \\ u(0,t) = u(\pi,t) = 0, \end{cases} \tag{5.39}$$

whose analytical solution, on an infinite domain, is given by

$$u(x,t) = \sum_{n=1}^{+\infty}\left(\frac{8}{n^3}(-1)^{n+1} - \frac{4}{n^3}\right)\sin(nx)\exp\left(-n^\alpha Kt\right). \tag{5.40}$$

To solve numerically problem (5.39) we consider both a direct application of the short–memory principle, i.e., we extract a banded approximation of the discretization matrix, and the use of approximate inverses instead of the true inverses. Both the results will be compared with the solution(s) obtained by solving the underlying sequence of linear systems.

As a first test, we consider the discretization of the symmetric Riesz FDE as a half-sum of left and right-sided Caputo derivatives, using the backward Euler scheme for advancing in time. We set the number of diagonals to be extracted as $d = 150$, and, to obtain a similar bandwidth in the inverse, drop-tolerances of $\delta = 5e-7$ and $\delta = 1e-7$. The averages of 2-norm of the errors are reported in Table 5.11. For the choice of

			GMRES		PRECONDITIONED	
α	β	fill-in	IT	T(s)	IT	T(s)
1.2	1.2	5.54 %	1.00 365.86	5.51e+00	1.00 79.22	**4.00e-01**
1.2	1.3	5.89 %	1.00 443.95	7.97e+00	1.00 79.92	**4.07e-01**
1.2	1.5	6.38 %	1.00 660.56	1.69e+01	1.00 78.15	**4.06e-01**
1.2	1.8	9.33 %	†	†	1.00 64.12	**2.87e-01**
1.3	1.2	6.07 %	1.00 436.73	7.72e+00	1.00 66.49	**3.08e-01**
1.3	1.3	5.51 %	1.00 514.36	1.06e+01	1.00 83.31	**4.37e-01**
1.3	1.5	6.14 %	†	†	1.00 82.80	**4.28e-01**
1.3	1.8	8.53 %	†	†	1.00 74.12	**3.64e-01**
1.5	1.2	6.06 %	1.00 637.86	1.58e+01	1.00 59.08	**2.59e-01**
1.5	1.3	6.16 %	†	†	1.00 66.36	**3.03e-01**
1.5	1.5	5.78 %	†	†	1.00 95.56	**5.45e-01**
1.5	1.8	7.35 %	†	†	1.00 92.97	**5.06e-01**
1.8	1.2	8.05 %	†	†	1.00 53.88	**2.34e-01**
1.8	1.3	7.39 %	†	†	1.00 57.36	**2.52e-01**
1.8	1.5	7.05 %	†	†	1.00 73.69	**3.67e-01**
1.8	1.8	7.18 %	†	†	1.00 115.46	**7.25e-01**

Table 5.9. Test problem in (5.36). $Nx = Ny = 80$, $Nt = 60$.

$\delta = 5e-7$ the bandwidth of the approximate inverse is 334 (instead of 300 given by the direct application of the short–memory principle) and an error that is of comparable modulus. The time needed for the solution is $T = 0.93s$ with the band-approximation, while we observe $T = 0.08s$ by using the approximate inverses. On the other hand, if we decrease the drop tolerances, then we obtain a solution with also a smaller error than the one obtained by solving the sequence of linear system with Gaussian elimination implemented in Matlab, i.e. the well known \, because the good information is already completely included in the underlying reduced model.

We also consider discretizing the symmetric Riesz derivatives with Ortigueira's centered fractional differences scheme, again by using backward Euler for advancing in time. The averages of the 2-norm of the errors are reported in Table 5.11. In this case, we obtain a bandwidth of 330 for the approximate inverses that can be compared with 300 of the direct approximation. The timings are $T = 0.77s$ with the direct application of the short–memory principle and $T = 0.09s$, by using the approximate inverses. The profile of the relative error has the same behavior as the former discretization.

			BiCGstab(2)		PRECONDITIONED	
α	β	fill–in	IT	T(s)	IT	T(s)
1.2	1.2	5.54 %	220.75	1.43e+00	44.88	**3.45e-01**
1.2	1.3	5.89 %	303.59	1.95e+00	44.69	**3.43e-01**
1.2	1.5	6.38 %	†	†	46.27	**3.48e-01**
1.2	1.8	9.33 %	†	†	39.61	**3.07e-01**
1.3	1.2	6.07 %	297.31	1.94e+00	36.27	**2.73e-01**
1.3	1.3	5.51 %	384.61	2.49e+00	49.42	**3.70e-01**
1.3	1.5	6.14 %	†	†	52.27	**3.93e-01**
1.3	1.8	8.53 %	†	†	46.22	**3.67e-01**
1.5	1.2	6.06 %	†	†	32.20	**2.42e-01**
1.5	1.3	6.16 %	†	†	37.92	**2.86e-01**
1.5	1.5	5.78 %	†	†	60.53	**4.54e-01**
1.5	1.8	7.35 %	†	†	64.41	**4.91e-01**
1.8	1.2	8.05 %	†	†	31.63	**2.42e-01**
1.8	1.3	7.39 %	†	†	34.68	**2.67e-01**
1.8	1.5	7.05 %	†	†	46.03	**3.50e-01**
1.8	1.8	7.18 %	†	†	80.80	**6.14e-01**

Table 5.10. Test problem in (5.36). $Nx = Ny = 80$, $Nt = 60$.

We experimented also the possibility outlined in Remark 5.1, i.e., the use of a banded approximation of the discretization matrix and of the approximate inverse as the true inverses. We observe that in these cases the effect of the terms of small norm is adding noise, i.e., ill-conditioning of the matrices, that reduces the overall accuracy obtained by the discretization methods.

			Complete		Banded		Direct	
Type	α	K	$T(s)$	err.	$T(s)$	err.	$T(s)$	err.
C	1.8	0.25	9.39	2.42e-02	0.93	4.29e-02	**0.08**	4.24e-02
							0.16	1.64e-02
O	1.8	0.25	9.27	3.03e-02	0.77	4.29e-02	**0.09**	4.43e-02
							0.11	3.33e-02
O*	1.5	0.75	8.64	7.43e-02	0.58	2.62e-01	**0.10**	7.66e-02
O*	1.2	1.25	9.43	5.27e-01	0.63	2.78e-01	**0.06**	2.73e-01
C*	1.8	1.50	8.66	1.15e-01	1.05	1.53e-01	**0.16**	1.83e-01

Table 5.11. Test problem in (5.39). Timings and averages of the 2-norm of the errors over all time steps. *C*: discretization using the half-sum of Caputo derivatives, *O*: Ortigueira discretization, while C^* and O^*: our method applied to the banded approximation of the discretization matrix, see Remark 5.1. Column *Complete*: the reference solution is used with the standard discretization matrix. Column *Banded*: the *Short–memory principle* is applied to the discretization of the operator. Column *Direct*: our approach using the approximate inverses. All the discretizations use $N = 2^{10}$.

6

Fractional PDEs Constrained Optimization

Among the key ingredients for the success of PDE constrained optimization, there is surely the wide set of possible applications, from fluid dynamic to control of heat–transfer processes, e.g, see [160, 185, 237]. To further extend the set of possible applications, in [81] we considered the numerical treatment of a nonstandard class of PDE constrained problems. By taking into account the use of quadratic objective functionals in the so–called class of *tracking type* costs functionals (see equation (6.1) below), we used as constraint a *fractional partial differential equation* (FPDE). In this framework, the appealing purpose of extending the existing strategies for classical PDE constrained optimization to the fractional case was already explored in a number of directions, see, e.g., [5], where the authors analyze a *matrix transfer* approach for fractional powers of elliptic operators or [96] and the use of the *discretize–then–optimize* approach is considered for the same class of problems. In this chapter, the FPDE constrained optimization problem is tackled instead by using the *optimize–then–discretize* technique, that has also been applied for the control of the fractional Fokker–Planck equation; see [4]. We start dealing with quadratic functionals with linear FDEs as constraints – for which the KKT formulation is also possible (see [96]) – in order to perform the analysis in a clearer way. Then, we move to the setting of semilinear FDEs for which only the *optimize–then–discretize* approach is viable; see Section 6.1.3.

The general formulation of the problem we analyze can be stated as follows, given a desired state z_d

$$\text{find } y, u \text{ such that } \begin{cases} \min J(y,u) = \dfrac{1}{2}\|y - z_d\|_2^2 + \dfrac{\lambda}{2}\|u\|_2^2, \\ \text{subject to } e(y,u) = 0, \end{cases} \tag{6.1}$$

where J and e are two continuously Fréchet differentiable functionals such that

$$J : Y \times U \to \mathbb{R}, \quad e : Y \times U \to W,$$

with Y, U and W reflexive Banach spaces. If we suppose that $e_y(\overline{y}, \overline{u}) \in \mathcal{B}(Y, W)$ is a bijection (where $\mathcal{B}(Y, W)$ is the set of bounded linear operators), then using the Implicit Function Theorem, we can deduce the existence of a (locally) unique solution $y(u)$ to the state equation $e(y, u) = 0$. We can then reformulate the problem in the form

$$\min_{u \in U} f(u) = \min_{u \in U} J(y(u), u), \tag{6.2}$$

where $J(y(u), u)$ is the reduced cost functional. Moreover, because $f(u)$ is continuous, convex and radially unbounded (see [237]), Problem (6.1) admits an optimal solution. We report here Theorem 3.3 in [237] which characterizes the optimal solution of (6.2) through necessary first order conditions:

Theorem 6.1. *Let $\overline{u} \in U$ be a local solution of* (6.2) *and $\overline{y}(\overline{u})$ its associated state, then there exists $p \in W'$ such that the following system of equations is satisfied:*

$$\begin{cases} e(y(\overline{u}), \overline{u}) = 0, \\ e_y(y(\overline{u}), \overline{u})^* p = J_y(y(\overline{u}), \overline{u}), \\ e_u(y(\overline{u}), \overline{u})^* p = J_u(y(\overline{u}), \overline{u}), \end{cases} \tag{6.3}$$

being W' the topological dual.

If $p \in W'$ satisfies the second equation in (6.3) then we call it *adjoint state*.

Observe that using the Lagrangian formulation of the problem

$$\mathcal{L}(y, u, p) = J(y, u) - < p, e(y, u) >_{W', W},$$

where $\mathcal{L} : Y \times U \times W' \to \mathbb{R}$, the optimality conditions in (6.3) are equivalent to the following conditions on $\mathcal{L}$ (see [237]):

$$\begin{cases} e(y(\overline{u}), \overline{u}) = 0, \\ \mathcal{L}_y(y, \overline{u}, p) = 0, \\ \mathcal{L}_u(y, \overline{u}, p) = 0. \end{cases} \tag{6.4}$$

The discussion is divided in this chapter as follows: in Section 6.1 we discuss the construction of the *optimize–then–discretize* approach for our problem from equation (6.1). In Section 6.2 we then discuss both the optimization routine and the application of the preconditioners from Section 6.2.2 for accelerating the computation of both the function $f(u)$ and the gradient $\nabla f(u)$ that are needed inside the optimization algorithm. As it will be clearer later, such computations represent the main computationally expensive part of the underlying strategy. At last,

in Section 6.3 we perform few numerical tests showing and comparing the computational performances of the proposed strategy.

The numerical tests contained in Section 6.3 highlight how the use of the preconditioning techniques we proposed in Section 6.2.2 for the acceleration in the computations turn out to be crucial in order to make the optimize-then-discretize technique effective.

We would like to stress that one of the objectives we followed in [81] was to obtain a fully reproducible and portable approach. Therefore, we illustrate the work by using preexisting and benchmarked software. In particular, we are going to use the Poblano Optimization Toolbox [105] and the CUSP library v0.5.1 [84] for the computation of the approximate inverse preconditioners; see also Section 5.3. Moreover, we point out that the technique presented here can be applied to general radially unbounded functionals $J(y,u)$ with nonlinear FDE constraints $e(y,u)$.

6.1 *Theoretical Results*

This section is devoted to the analysis of the Lagrangian conditions (6.4) for two FPDE constrained optimization problems of the form (6.1). We will obtain them for both the *Fractional Advection Dispersion Equation* (FADE) from [110, 201] and the two–dimensional *Riesz Space Fractional Diffusion equation*, e.g., see [225]. To the best of the author knowledge, the results from [81], even if formally similar to the elliptic case, have not been previously stated for the FPDEs in the existing literature.

6.1.1 The 1D Case

Let us consider the following Fractional Advection Dispersion Equation (FADE) [201]; see Appendix A.2 and Definition 5.1.

$$\begin{cases} -a\left(l\,{}_{\mathrm{RL}}D^{2\alpha}_{0,x} + r\,{}_{\mathrm{RL}}D^{2\alpha}_{x,1}\right)y(x) + \ldots & x \in \Omega = (0,1), \\ \qquad \ldots + b(x)y'(x) + c(x)y(x) = u(x), & \\ y(0) = y(1) = 0, & \end{cases} \tag{6.5}$$

for $\alpha \in (1/2, 1)$, $a, l, r > 0$ and $l + r = 1$, $b(x) \in \mathscr{C}^1(\overline{\Omega})$, $c(x) \in \mathscr{C}(\overline{\Omega})$ and such that $c(x) - 1/2 b'(x) \geq 0$.

Problem (6.1) rewrites for the FADE (6.5) as

$$\begin{cases} \min J(y,u) = & \frac{1}{2}\|y - z_d\|_2^2 + \frac{\lambda}{2}\|u\|_2^2, \\ \quad \text{subject to} & -a\left(l\,{}_{\mathrm{RL}}D^{2\alpha}_{0,x} + r\,{}_{\mathrm{RL}}D^{2\alpha}_{x,1}\right)y(x) + \\ & +b(x)y'(x) + c(x)y(x) - u(x) = 0, \\ & y(0) = y(1) = 0. \end{cases} \tag{6.6}$$

Let us define $H_0^{\mu}(\Omega)$ for $\mu > 0$ the closure of $\mathbb{C}_0^{\infty}(\Omega)$ in the norm,

$$\|u\|_{\mu} = \left(\|u\|_2^2 + \|\,{}_{\mathrm{RL}}D_{0,x}^{\mu}u\|_2^2\right)^{1/2}. \tag{6.7}$$

Remark 6.1. *We could have defined the norm $\|\cdot\|_{\mu}$ equivalently for the right–side Riemann–Liouville fractional derivatives or the Symmetric Riesz derivative, because for every $\mu > 0$ all the various definitions of these spaces are equal, moreover for $\mu \neq n - 1/2$ also all their norm are equivalent, see [110, Section 2]. This is indeed an example of a fractional Sobolev space, either of proper Besov space, see [92] for more details. Finally observe that, using standard arguments, it is easy to prove that the Fractional Sobolev Spaces employed in this section are reflexive; for further details on these issues see Appendix A.2.1 and references therein.*

In the following discussion $< \cdot, \cdot >$ will indicate the standard $\mathbb{L}^2$ inner product. Duality pairing, when needed, is explicitly indicated.

Proposition 6.1 (Cipolla and Durastante [81]; Lagrange Conditions for the FADE)**.** *Using the notation of Chapter 6 let us define $U = \mathbb{L}^2(\Omega)$, $Y = H_0^{\alpha}(\Omega)$ and $W = Y'$. The Lagrangian conditions* (6.4) *for Problem* (6.6) *are expressed in weak form as*

$$\begin{cases} B(y, v) = F(v), & \forall\, v \in Y, \\ \tilde{B}(p, w) =< y - z_d, w >, & \forall w \in Y, \\ < p, h >= -\lambda < u, h >, & \forall\, h \in Y, \end{cases} \tag{6.8}$$

where $u \in U$, $y \in Y$ and

$$\begin{aligned} B(y, v) = &- al < {}_{RL}D_{0,x}^{\alpha}y(x), {}_{RL}D_{x,1}^{\alpha}v(x) > + \\ &- ar < {}_{RL}D_{x,1}^{\alpha}y(x), {}_{RL}D_{0,x}^{\alpha}v(x) > + \\ &+ < b(x)y'(x) + c(x)y(x), v(x) >, \\ \tilde{B}(p, w) = &- al < {}_{RL}D_{0,x}^{\alpha}w(x), {}_{RL}D_{x,1}^{\alpha}p(x) > + \\ &- ar < {}_{RL}D_{x,1}^{\alpha}w(x), {}_{RL}D_{0,x}^{\alpha}p(x) > + \\ &- < (b(x)p(x))', w > + < c\,p, w >, \\ F(v) = &< u, v > . \end{aligned} \tag{6.9}$$

Proof. The variational formulation of (6.5) can be easily obtained from [293, Theorem 1]. Taking an arbitrary $v \in Y$, the weak formulation of $e(y, u) = 0$ can be interpreted as

$$e(y, u)v \triangleq B(y, v) - F(v) \;:\; Y \to Y'. \tag{6.10}$$

The FPDE admits a weak solution through the Lax–Milgram Theorem because B is bilinear, continuous and coercive and F is continuous (see [293, Theorem 1]). Therefore, we can express the Lagrangian for this problem as

$$\mathscr{L}(y,u,p) = J(y,u) - < p, e(y,u) >_{W',W} : \; Y \times \mathbb{L}^2(\Omega) \times W' \to \mathbb{R}.$$

We can now compute the Gâteaux derivative of $\mathscr{L}$ for any function $w \in Y$ (see [160, 237])

$$\begin{aligned}
\mathscr{L}_y(y,u,p)w =& J_y(y,u)w - < p, e_y(y,u)w >_{W',W} \\
=& < y - z_d, w > + al < {}_{\mathrm{RL}}D^{\alpha}_{x,1} p, {}_{\mathrm{RL}}D^{\alpha}_{0,x} w >_{Y',Y} + \\
& + ar < {}_{\mathrm{RL}}D^{\alpha}_{0,x} p, {}_{\mathrm{RL}}D^{\alpha}_{x,1} w >_{Y',Y} + < (b(x)p(x))', w >_{Y',Y} + \\
& - < c\, p, w > = < y - z_d, w > - \tilde{B}(p,w) = 0,
\end{aligned}$$

and thus we obtain,

$$\mathscr{L}_y(y,u,p)w = 0 \;\Leftrightarrow\; \tilde{B}(p,w) = < y - z_d, w > \quad \forall w \in Y,$$

that again has a unique solution by the Lax–Milgram Theorem and [110, Theorem 3.5]. At last we have to compute the Gâteaux derivative of $\mathscr{L}$ respect to the variable u

$$\mathscr{L}_u(y,u,p)h = \lambda < u, h > + < p, h >, \; \forall\, h \in Y.$$

This completes the proof. □

Corollary 6.1 (Cipolla and Durastante [81])**.** *The gradient for the objective function $f(u)$ from equation* (6.2) *for the Problem* (6.6) *reads as*

$$\nabla f(u) = \mathscr{L}_u(y(u), u, p(u)) = p + \lambda u, \tag{6.11}$$

where p is the solution of the adjoint equation:

$$\begin{cases}
-a \left(r \, {}_{RL}D^{2\alpha}_{0,x} + l \, {}_{RL}D^{2\alpha}_{x,1} \right) p(x) + \ldots \\
\qquad \ldots - \frac{d}{dx}(b(x)p(x)) + c(x)p(x) = y(x) - z_d(x), \\
p(0) = p(1) = 0.
\end{cases} \tag{6.12}$$

Proof. The characterization of the gradient of $f(u)$ is straightforward, see [160, p. 63]. Furthermore we have exchanged the fractional operators in the bilinear part of (6.12), since the adjoint property for fractional operators in $\mathbb{L}^2$ holds, see [251]. □

6.1.2 The 2D Case: Riesz Space Fractional Diffusion

We treat now, in a similar way, the case of the two–dimensional Riesz space fractional diffusion equation [225], that reads as

$$\begin{cases} -K_{x_1}\frac{\partial^{2\alpha}y}{\partial|x_1|^{2\alpha}} - K_{x_2}\frac{\partial^{2\beta}y}{\partial|x_1|^{2\beta}} + \mathbf{b}\cdot\nabla y + cy = u, & (x_1,x_2)\in\Omega, \\ y \equiv 0, & (x_1,x_2)\in\partial\Omega, \end{cases} \tag{6.13}$$

where $\mathbf{b} \in \mathscr{C}^1(\Omega,\mathbb{R}^2)$, $c \in \mathscr{C}(\Omega)$, $u \in \mathbb{L}^2(\Omega)$, $K_{x_1}, K_{x_2} \geq 0$ and $K_{x_1}+K_{x_2} > 0$, $\alpha,\beta \in (1/2,1)$, $\Omega = [a,b]\times[c,d]$. Here the symmetric Riesz fractional operator is defined as follows.

Definition 6.1. *Given a function $u(x_1,x_2)$ and given $1/2 < \mu \leq 1$ and $n-1 < 2\mu \leq n$, we define the* symmetric Riesz derivative *as,*

$$\frac{\partial^{2\mu}u(x_1,x_2)}{\partial|x_1|^{2\mu}} = -c_{2\mu}\left({}_{RL}D^{2\mu}_{a,x_1} + {}_{RL}D^{2\mu}_{x_1,b}\right)u(x_1,x_2), \quad c_{2\mu} = \frac{1}{2\cos(\mu\pi)}$$

and

$$\begin{aligned} {}_{RL}D^{2\mu}_{a,x}u(x_1,x_2) =& \frac{1}{\Gamma(n-2\mu)}\left(\frac{\partial}{\partial x_1}\right)^n \int_a^{x_1} \frac{u(\xi,x_2)d\xi}{(x_1-\xi)^{2\mu-n+1}}, \\ {}_{RL}D^{\mu}_{x,b}u(x_1,x_2) =& \frac{1}{\Gamma(n-2\mu)}\left(-\frac{\partial}{\partial x_1}\right)^n \int_{x_1}^{b} \frac{u(\xi,x_2)d\xi}{(\xi-x_1)^{2\mu-n+1}}. \end{aligned}$$

And analogously we can operate in the x_2 direction.

Thus Problem (6.1) rewrites in this case as

$$\begin{cases} \min J(y,u) = \frac{1}{2}\|y-z_d\|_2^2 + \frac{\lambda}{2}\|u\|_2^2, \\ \quad \text{subject to } -K_{x_1}\frac{\partial^{2\alpha}y}{\partial|x_1|^{2\alpha}} - K_{x_2}\frac{\partial^{2\beta}y}{\partial|x_2|^{2\beta}} + \mathbf{b}\cdot\nabla y + cy = u, \\ \quad\quad y \equiv 0, (x_1,x_2)\in\partial\Omega. \end{cases} \tag{6.14}$$

Proposition 6.2 (Cipolla and Durastante [81]; Lagrange Conditions for the Riesz space fractional diffusion)**.** *Using the notation of Chapter 6 let us define $U = \mathbb{L}^2(\Omega)$, $Y = H_0^{\alpha}(\Omega)\cap H_0^{\beta}(\Omega)$ and $W = Y'$. The Lagrangian conditions* (6.4) *for Problem* (6.14) *are expressed in weak form as*

$$\begin{cases} B(y,v) = F(v), & \forall\, v\in Y, \\ \tilde{B}(p,w) =< y-z_d, w >, & \forall w\in Y, \\ < p,h >= -\lambda < u,h >, & \forall\, h\in Y, \end{cases} \tag{6.15}$$

where $u \in U$, $y \in Y$ and setting $C_x = K_{x_1} c_{2\alpha}$ and $C_{x_2} = K_{x_2} c_{2\beta}$,

$$\begin{aligned}
B(y,v) =& C_{x_1}\left(< {}_{RL}D^{\alpha}_{a,x_1} y, {}_{RL}D^{\alpha}_{x_1,b} v > + < {}_{RL}D^{\alpha}_{x_1,b} y, {}_{RL}D^{\alpha}_{a,x_1} v >\right) \\
&+ C_{x_2}\left(< {}_{RL}D^{\beta}_{c,x_2} y, {}_{RL}D^{\beta}_{x_2,d} v > + < {}_{RL}D^{\beta}_{x_2,d} y, {}_{RL}D^{\beta}_{c,x_2} v >\right) \\
&+ < \mathbf{b}\cdot\nabla y, v > + < cy, v >, \\
\tilde{B}(p,w) =& C_{x_1}\left(< {}_{RL}D^{\alpha}_{a,x_1} p, {}_{RL}D^{\alpha}_{x_1,b} w > + < {}_{RL}D^{\alpha}_{x_1,b} p, {}_{RL}D^{\alpha}_{a,x_1} w >\right) \\
&+ C_{x_2}\left(< {}_{RL}D^{\beta}_{c,x_2} p, {}_{RL}D^{\beta}_{x_2,d} w > + < {}_{RL}D^{\beta}_{x_2,d} p, {}_{RL}D^{\beta}_{c,x_2} w >\right) \\
&- < \nabla\cdot(\mathbf{b}p), w > + < cp, w >, \\
F(v) =& < u, v > .
\end{aligned}$$

Proof. Taking an arbitrary $v \in Y$ the weak formulation [68] of (6.13) can be interpreted as

$$e(y,u)v \triangleq B(y,v) - F(v) : Y \to Y'.$$

The FPDE admits a weak solution through the Lax–Milgram Theorem because B is bilinear, continuous and coercive (whenever $c - 1/2\nabla\cdot\mathbf{b} \geq 0$ a.e. on Ω, see, e.g., [232]) and F is continuous in Ω, see [68, Theorem 2]. The Lagrangian for this problem is

$$\mathcal{L}(y,u,p) = J(y,u) - < p, e(y,u) >_{W',W} : Y \times \mathbb{L}^2(\Omega) \times Y' \to \mathbb{R}.$$

We can compute conditions (6.15) directly, using the fact that left and right Riemann–Liouville fractional derivatives are one the adjoint of the other ([68, Lemma 5]). More specifically we have

$$\begin{aligned}
\mathcal{L}_y(y,u,p)w =& J_y(y,u)w - < p, e_y(y,u)w >_{W',W} \\
=& < y - z_d, w > - C_{x_1}\left(< {}_{\mathrm{RL}}D^{\alpha}_{a,x_1} p, {}_{\mathrm{RL}}D^{\alpha}_{x_1,b} w >_{Y',Y} + \right. \\
&\left. + < {}_{\mathrm{RL}}D^{\alpha}_{x_1,b} p, {}_{\mathrm{RL}}D^{\alpha}_{a,x_1} w >_{Y',Y}\right) + \\
&- C_{x_2}\left(< {}_{\mathrm{RL}}D^{\beta}_{c,x_2} p, {}_{\mathrm{RL}}D^{\beta}_{x_2,d} w >_{Y',Y} + \right. \\
&\left. + < {}_{\mathrm{RL}}D^{\beta}_{x_2,d} p, {}_{\mathrm{RL}}D^{\beta}_{c,x_2} w >_{Y',Y}\right) + \\
&+ < \nabla\cdot(\mathbf{b}p), w >_{Y',Y} - < cp, w > \\
=& < y - z_d, w > - \tilde{B}(p,w) = 0,
\end{aligned}$$

and thus we obtain,

$$\mathcal{L}_y(y,u,p)w = 0 \;\Leftrightarrow\; \tilde{B}(p,w) = < y - z_d, w > \quad \forall w \in Y,$$

that again has a unique solution by the Lax–Milgram Theorem and [68, Theorem 2] with $c - 1/2\nabla \cdot \mathbf{b} \geq 0$ a.e. on Ω. We obtain again the same result as for the FADE case by considering the Gâteaux derivative of $\mathcal{L}$ with respect to the variable u

$$\mathcal{L}_u(y, u, p)h = \lambda < u, h > + < p, h >, \ \forall h \in Y. \qquad \square$$

Corollary 6.2 (Cipolla and Durastante [81])**.** *The gradient for the objective function $f(u)$ from equation* (6.2) *for the Problem* (6.14) *reads as*

$$\nabla f(u) = \mathcal{L}_u(y(u), u, p(u)) = p + \lambda u, \tag{6.16}$$

where p is the solution of the adjoint equation for $\mathbf{x} \in \Omega$

$$\begin{cases} -K_{x_1} \frac{\partial^{2\alpha} p}{\partial |x_1|^{2\alpha}} - K_{x_2} \frac{\partial^{2\beta} p}{\partial |x_2|^{2\beta}} + \ldots & (x_1, x_2) \in \Omega \\ \qquad \ldots - \nabla \cdot (p\mathbf{b}) + cp = y - z_d, & \\ p \equiv 0, & (x_1, x_2) \in \partial\Omega. \end{cases} \tag{6.17}$$

Remark 6.2. *Proposition 6.2 could have also been written for the nonsymmetric case, i.e., for terms that are convex combination of Riemann–Liouville fractional derivatives; analogous results to those in Proposition 6.1 could be stated in this case.*

6.1.3 The Semilinear Case

As a simple extension of the considered framework, we can focus on the semilinear generalization of Problem (6.14)

$$\begin{cases} \min J(y, u) = \dfrac{1}{2}\|y - z_d\|_2^2 + \dfrac{\lambda}{2}\|u\|_2^2, \\ \text{subject to } -K_{x_1} \dfrac{\partial^{2\alpha} y}{\partial |x_1|^{2\alpha}} - K_{x_2} \dfrac{\partial^{2\beta} y}{\partial |x_2|^{2\beta}} + \mathbf{b} \cdot \nabla y + c y^{\zeta} = u, \\ \quad y \equiv 0, (x_1, x_2) \in \partial\Omega. \end{cases} \tag{6.18}$$

where $\zeta \in \mathbb{N}$, $\mathbf{b} \in \mathscr{C}^1(\Omega, \mathbb{R}^2)$, $c \in \mathscr{C}(\Omega)$, $u \in \mathbb{L}^2(\Omega)$, $K_{x_1}, K_{x_2} \geq 0$ and $K_{x_1} + K_{x_2} > 0$, $\alpha, \beta \in (1/2, 1)$, $\Omega = [a, b] \times [c, d]$. Observe that also the semilinear generalization of the 1D case (6.6) could be recovered easily by employing the subsequent results.

Proposition 6.3 (Cipolla and Durastante [81]; Lagrange Conditions for the Riesz space semilinear fractional diffusion)**.** *Under the notation in Chapter 6, let us define $U = \mathbb{L}^2(\Omega)$, $Y = H_0^{\alpha}(\Omega) \cap H_0^{\beta}(\Omega)$ and $W = Y'$. The Lagrangian conditions* (6.4) *for Problem* (6.14) *are expressed in weak form as*

$$\begin{cases} B(y, v) = F(v), & \forall v \in Y, \\ \tilde{B}(p, w) = < y - z_d, w >, & \forall w \in Y, \\ < p, h > = -\lambda < u, h >, & \forall h \in Y, \end{cases} \tag{6.19}$$

where $u \in U$, $y \in Y$ and setting $C_x = K_{x_1} c_{2\alpha}$ and $C_{x_2} = K_{x_2} c_{2\beta}$,

$$\begin{aligned}
B(y,v) =& C_{x_1}\left(< {}_{RL}D^{\alpha}_{a,x_1} y, {}_{RL}D^{\alpha}_{x_1,b} v > + < {}_{RL}D^{\alpha}_{x_1,b} y, {}_{RL}D^{\alpha}_{a,x_1} v >\right) \\
&+ C_{x_2}\left(< {}_{RL}D^{\beta}_{c,x_2} y, {}_{RL}D^{\beta}_{x_2,d} v > + < {}_{RL}D^{\beta}_{x_2,d} y, {}_{RL}D^{\beta}_{c,x_2} v >\right) \\
&+ < \mathbf{b}\cdot\nabla y, v > + < c y^{\zeta}, v >, \\
\tilde{B}(p,w) =& C_{x_1}\left(< {}_{RL}D^{\alpha}_{a,x_1} p, {}_{RL}D^{\alpha}_{x_1,b} w > + < {}_{RL}D^{\alpha}_{x_1,b} p, {}_{RL}D^{\alpha}_{a,x_1} w >\right) \\
&+ C_{x_2}\left(< {}_{RL}D^{\beta}_{c,x_2} p, {}_{RL}D^{\beta}_{x_2,d} w > + < {}_{RL}D^{\beta}_{x_2,d} p, {}_{RL}D^{\beta}_{c,x_2} w >\right) \\
&- < \nabla\cdot(\mathbf{b}p), w > + < \zeta c y^{\zeta-1} p, w >, \\
F(v) =& < u, v > .
\end{aligned}$$

Proof. The proof is obtained using the same techniques as in Proposition 6.2. For proving the existence of the solution of (6.18), it suffices to consider the Browder–Minty Theorem [64, 237] instead of the Lax–Milgram Theorem; moreover, we used the fact that fractional Sobolev spaces are continuously embedded into $\mathbb{L}^{2\zeta}(\Omega)$ spaces; see [92]. □

Corollary 6.3. *The gradient for the objective function $f(u)$ from equation* (6.2) *for the Problem* (6.18) *reads as*

$$\nabla f(u) = \mathcal{L}_u(y(u), u, p(u)) = p + \lambda u, \tag{6.20}$$

where p is the solution of the adjoint equation for $\mathbf{x} \in \Omega$

$$\begin{cases} -K_{x_1} \dfrac{\partial^{2\alpha} p}{\partial |x_1|^{2\alpha}} - K_{x_2} \dfrac{\partial^{2\beta} p}{\partial |x_2|^{2\beta}} - \nabla\cdot(p\mathbf{b}) + \zeta c y^{\zeta-1} p = y - z_d, \\ p \equiv 0. \end{cases} \tag{6.21}$$

Therefore, in Problem (6.19) we need to face a nonlinear state equation while, by Corollary 6.3, we still need the solution of a linear fractional differential adjoint equation, that now depends on the current value of y.

6.2 *Algorithms*

In this section we discuss the *finite difference* discretization of both the FADE (6.5) and the Riesz Fractional Diffusion equation (6.13) in conjunction with the optimization algorithms for the solution of the constrained problem of equation (6.1). We will also treat the acceleration of the solution routines of the FPDE inside the optimization algorithm, since this determines substantially the cost of one step of the optimization algorithm and of one step of the Line–Search routine. Indeed, this

corresponds in reducing the cost needed for computing both $f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_k)$, which represents the main cost per iteration of the considered optimization algorithm; see Section 6.2.3. We stress that we are going to use the finite difference discretization to use the standard Euclidean scalar product in the optimization routine; see [237] for more details.

6.2.1 Discretization of the FPDEs

We are dealing with the finite difference discretization of fractional Riemann–Liouville operators in space, therefore we are going to use again the the *right p–shifted Grünwald–Letnikov formula* Definitions 5.11 and 5.12; see Appendix A.3.1 and [201].

Therefore, whenever a grid $\{x_k\}_k = \{a + kh\}_k$ is set on the domain $\Omega = [a, b]$ with stepsize $h = {}^{b-a}/_{n}$, we can approximate any Riemann–Liouville derivative of a suitably smooth function $y(x)$. The parameter p can be chosen again as $p = 1$ to optimize the performance as the minimizer of $|\alpha - {}^{p}/_{2}|$ for $1 < \alpha \leq 2$. Combining formulas (5.12)–(5.11) with the standard finite difference schemes for ordinary equations, we obtain the discretization of the full FADE (equation (6.5)). Let us focus on the discretization of the fractional operator with homogeneous Dirichlet boundary conditions, i.e., $y(a) = y(b) = 0$,

$$-\frac{1}{h^\alpha}\begin{bmatrix} \omega_1 & \omega_0 & 0 & \dots & 0 \\ \omega_2 & \omega_1 & \omega_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \omega_1 & \omega_0 \\ \omega_{n-1} & \dots & \dots & \omega_2 & \omega_1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-1} \end{bmatrix},$$

which can be expressed in compact form as the Toeplitz matrix (Definition 2.4),

$$T_{(\alpha)}\mathbf{y} = \mathbf{u}. \tag{6.22}$$

While for the right Riemann–Liouville derivative we have

$$-\frac{1}{h^\alpha}\begin{bmatrix} \omega_1 & \omega_2 & \dots & \dots & \omega_{n-1} \\ \omega_0 & \omega_1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \omega_0 & \omega_1 & \omega_2 \\ 0 & \dots & 0 & \omega_0 & \omega_1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{n-1} \end{bmatrix}.$$

The latter can be expressed in compact form, using the same convention as equation (6.22), as the Toeplitz matrix

$$T_{(\alpha)}^{T}\mathbf{y} = \mathbf{u}. \tag{6.23}$$

For the ordinary part we obtain

$$\begin{bmatrix} c_1 & \frac{b_1}{2h} & & \\ -\frac{b_2}{2h} & \ddots & \ddots & \\ & \ddots & c_3 & \frac{b_{n-2}}{2h} \\ & & -\frac{b_{n-1}}{2h} & c_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix},$$

that in matrix form reads as,

$$O_n\mathbf{y} = \mathbf{u}. \tag{6.24}$$

We can now assemble the full discretization for the FADE equation assembling the relationships (6.22), (6.23) and (6.24) as,

$$\begin{aligned} A\mathbf{y} &= \mathbf{f}, \text{ with} \\ A &= a(lT_{(\alpha)} + rT_{(\alpha)}^{T}) + O_n, \\ \mathbf{f} &= \mathbf{u}. \end{aligned}$$

Observe that from equations (6.22)–(6.23), we can also obtain the building blocks for the discretization of the *adjoint equation* (6.12), we only need to modify the part relative to the ordinary derivatives as,

$$\begin{bmatrix} c_1 & -\frac{b_2}{2h} & & \\ \frac{b_1}{2h} & c_2 & \ddots & \\ & \ddots & \ddots & -\frac{b_{n-1}}{2h} \\ & & \frac{b_{n-2}}{2h} & c_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix},$$

that in matrix form can be expressed as

$$O_n'\mathbf{y} = \mathbf{u}. \tag{6.25}$$

As a last step, putting together relationships (6.22),(6.23) and (6.25), we obtain the following expression for Problem (6.12) as

$$\begin{aligned} A'\mathbf{p} &= \mathbf{f}', \text{ with} \\ A' &= a(rT_{(\alpha)} + lT_{(\alpha)}^{T}) + O_n', \\ \mathbf{f}' &= \mathbf{y} - \mathbf{z}_d. \end{aligned}$$

It is possible to treat the discretization of the Riesz space–fractional differential equation (6.13) in complete analogy to the one dimensional case, that is using the discretization of the Riemann–Liouville derivatives and their relation with the Riesz derivative (see Definition 6.1 and the discussion in Appendix A.3.2). Otherwise, it is possible to use the second order accurate *fractional centered derivative scheme* for Riesz derivative from [220]; see again Appendix A.3.2 for the details. Coupling the selected method with the usual finite difference scheme for the convective and reactive terms, we obtain in both cases the discretization of the *state* and *adjoint* equation (respectively equation (6.13) and (6.17)) for the two–dimensional *Riesz Space Fractional Diffusion equation*. In more details, such discretization is linked to the one dimensional discretization through Kronecker sums, i.e.,

$$A = K_{x_1}(R_{x_1}^{(\alpha)} \otimes I_{x_2}) + K_{x_2}(I_{x_1} \otimes R_{x_2}^{(\beta)}) + B_{x_1}(T_{x_1} \otimes I_{x_2}) + B_{x_2}(I_{x_1} \otimes T_{x_2}) + C,$$

where I_{x_i}, for $i = 1, 2$, are the identity matrix relative to the grid on the x_i direction, $R_{x_1}^{(\alpha)}$ and $R_{x_2}^{(\beta)}$ are the dense Toeplitz matrices associated with the one dimensional fractional order derivatives in the two directions, the T_{x_i} $i = 1, 2$, are the finite difference matrices for the convective terms obtained with centered differences, $\{B_{x_i}\}_{i=1,2}$ and C are respectively the evaluation of the functions $\mathbf{b} = (b^{(1)}, b^{(2)})$ and c on the relative nodes of the (i, j)–finite difference grid with N_{x_1}, N_{x_2} nodes and amplitude h, k respectively. The discretization matrix A' of the *adjoint* equation is then obtained in this case simply by substituting the matrix related to the convective terms $B = B_{x_1}(T_{x_1} \otimes I_{x_2}) + B_{x_2}(I_{x_1} \otimes T_{x_2})$, with the following block matrix

$$B' = \begin{bmatrix} B_1 & J_2 & & \\ -J_1 & B_2 & \ddots & \\ & \ddots & \ddots & J_{N_{x_1}} \\ & & -J_{N_{x_1}-1} & B_{N_{x_1}} \end{bmatrix},$$

where

$$B_j = \begin{bmatrix} 0 & \frac{b_{j,2}^{(2)}}{2k} & & \\ -\frac{b_{j,1}^{(2)}}{2k} & \ddots & \ddots & \\ & \ddots & \ddots & \frac{b_{j,N_{x_2}}^{(2)}}{2k} \\ & & -\frac{b_{j,N_{x_2}-1}^{(2)}}{2k} & 0 \end{bmatrix}, \quad J_j = \begin{bmatrix} \frac{b_{j,1}^{(1)}}{2h} & 0 & 0 & 0 \\ 0 & \frac{b_{j,2}^{(1)}}{2h} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{b_{j,N_{x_2}}^{(1)}}{2h} \end{bmatrix}.$$

Lastly, for facing the solution of the state equation in the semilinear case, see Proposition 6.3, we need to solve a nonlinear equation each time.

The Jacobian can be computed easily, since the problem is semilinear, and thus the Newton method appears to be a natural tool. Indeed, finding the solution of the discrete semilinear state equation is equivalent to solving

$$H(\mathbf{y}) \equiv \left[K_{x_1}(R_{x_1}^{(\alpha)} \otimes I_{x_2}) + K_{x_2}(I_{x_1} \otimes R_{x_2}^{(\beta)}) + B\right]\mathbf{y} + C\mathbf{y}^{\zeta} - \mathbf{u} = 0,$$

where the power of vectors is computed elementwise. Then, the Jacobian is given by:

$$J_H(\mathbf{y}) = K_{x_1}(R_{x_1}^{(\alpha)} \otimes I_{x_2}) + K_{x_2}(I_{x_1} \otimes R_{x_2}^{(\beta)}) + B + \zeta C\,\mathrm{diag}(\mathbf{y}^{\zeta-1}). \quad (6.26)$$

Finally, we also observe that the discretization matrix of the adjoint equation varies at each step of the minimization process since it depends on the current values of $\mathbf{y}$. Thus, we have the following parametric family of matrices

$$A'(\mathbf{y}) = K_{x_1}(R_{x_1}^{(\alpha)} \otimes I_{x_2}) + K_{x_2}(I_{x_1} \otimes R_{x_2}^{(\beta)}) + B' + \zeta C\,\mathrm{diag}(\mathbf{y}^{\zeta-1}) \quad (6.27)$$

discretizing the adjoint equation.

6.2.2 Preconditioners for FPDEs

As we have discussed in Sections 5.1 and 5.1.1 the matrices generated by the *shifted Grünwald–Letnikov* discretization and the *fractional centered derivative scheme* for Riesz derivative share the same decay property along the diagonals; see [35, 225].

Therefore, we are going to use the preconditioners from Section 5.2.1 which exploit the *short–memory principle*, i.e., the decaying of the entries, in order to gain information on the inverse of the discretization matrix. In this case, the procedure is used for discarding elements of prescribed *small modulus* in the calculation of an approximate inverse of the matrices A and A'. This technique will produce explicit preconditioners for Krylov subspace methods called the approximate inverse preconditioners; see, e.g., [15, 20, 31, 36].

Also in this case, we have set the focus on the incomplete biconjugation process from [36] since its left–looking/outer product formulation permits to obtain sparser factors under suitable conditions. For algorithmic issues regarding these computations we refer again to [19, 36, 61, 62] and to the discussion in Section 2.4, since we use again the implementation in [84] based on Bridson's *outer product* formulation.

Clearly, there exist many preconditioners developed for the solution of linear FPDEs. Hence, in general, different valid choices could be used

for different formulation of the FPDE constraints, e.g., in the case of constant coefficient linear FPDEs with only fractional derivatives, one can consider the proposals in [182], while for more general variable coefficients linear FPDEs with only fractional derivatives one can consider the proposals in [100, 221].

Observe that in our case we need to face both linear and semilinear FPDEs with mixed type of derivatives, both fractional and classic so we will use again the *approximate inverse preconditioners* (2.68) for the solution of the linear systems with matrix A and A' from Section 6.2.1, i.e., for the numerical solution of the *state* and *adjoint* equations for our Problem (6.1).

In the semilinear case instead, as seen at the end of Section 6.2.1, we need to face two sequences of linear systems with varying matrices, i.e., we have a sequence of matrices $\{J_H(\mathbf{y}^{(k)})\}_{k\geq 0}$ from (6.26). The latter sequence is generated for the solution of the state equation (6.18) using the Newton method, and the sequence of matrices $\{A'(\mathbf{y}^{(k)})\}_{k\geq 1}$ from (6.27), i.e., we are in the case depicted in Example 3.2. It is usually expensive to rebuild a new preconditioner for each new matrix, while on the other hand, reusing the same preconditioner cannot be appropriate if the matrices in the sequences are very different. The update techniques from Section 4.2 with its specialization from Section 5.2.2 (for updating sequences of matrices coming from discretization of FPDEs) can be applied also in this case; see also [15, 20, 31, 33]. In addition, the techniques introduced in [15] can be applied in the case at hand for adapting the updates for Newton–Krylov methods.

6.2.3 Optimization Routine: the L–BFGS Algorithm

The L–BFGS method, originally introduced in [193], continuously updates the quasi–Newton matrix, approximation of the Hessian of Problem (6.2), using a fixed limited amount of storage. At every step, the oldest information contained in the matrix is discarded and replaced by a new one with the aim of producing an up to date model of the objective function, see [105, 193, 215] for further details.

It can be easily observed that the computational cost of the L–BFGS algorithm can be divided into two parts, the $O(Mn)$ flops of the optimization procedure plus the cost required for the evaluation of the function and the gradient. It is worth noticing that the Line-Search step, performed with the More-Thuente cubic interpolation ([207]) and satisfying the strong Wolfe conditions, could require more than one evaluation of the function and the gradient. As it is clear from (6.11) or (6.16), each of such evaluations requires the solution of two FPDEs,

and hence this turns out to be the dominating computational cost per step for the L–BFGS. In our case this is done through the definition of the following Matlab routine (in the non preconditioned case) for the evaluation of $f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_k)$:

```
function [f,g] = F(A,Aadj,u,zd,lambda)
y = A\u;          % Solution of the state equation
p = Aadj\(y-zd) % Solution of the adjoint equation
f = 0.5*(y-zd)'*(y-zd) + lambda/2*(u'*u);
g = lambda*u+p;
end
```

where for the variables we have used the same name employed in the theoretical analysis.

The above observation fully justifies the approach adopted in [81]: thus we are going to consider the speedup obtained by exploiting the particular structure of systems (6.22) and (6.23). More specifically we introduce a preconditioned Krylov iterative method instead of the direct solver for the linear system, as discussed in Sections 6.2.2 and 6.3.2 and Chapter 5.

6.3 *Numerical examples*

We will divide this section into two different parts. The first one is devoted to showing that the strategy described in Section 6.2 produces a feasible solution for Problem (6.1), while the other will be focused on showing the acceleration obtained with the proposed preconditioning strategy for the linear systems, see Section 6.2.2.

The results have been obtained on a laptop running Linux with 8 Gb memory and CPU Intel® Core™ i7-4710HQ CPU with clock 2.50 GHz, while the GPU is an NVIDIA GeForce GTX 860M. The scalar code is written and executed in Matlab R2016b, while for the GPU we use C++ with Cuda compilation tools, release 7.5, V7.5.17 and the CUSP library v0.5.1 [84].

6.3.1 Constrained Optimization Results

Throughout this set of experiments, the discretization parameter N will be set to $N = 10^3$. The L–BFGS algorithm options are set as in the default Poblano settings, see [105, Table 1,2 and 4], and the limited memory parameter is either $M = 5$, for the one dimensional case, either $M = 10$, for the two dimensional one, see also the discussion on L–BFGS algorithm from Section 6.2.3.

Let us start from the solution of the Problem (6.6) with the following choice of coefficients and desired state

$$\begin{aligned} a = 3,\ l = 0.8,\ r = 0.2,\ b(x) = 0,\ c(x) = 0,\ 2\alpha = 1.3, \\ z_d(x) = -x^2(x-1)^2 \sin(20\pi x). \end{aligned} \tag{6.28}$$

In Figure 6.1 we compare the solution obtained with different values of the regularization parameter λ. The error decreases sensibly with the three values of $\lambda = 10^{-3}, 10^{-5}, 10^{-9}$ and the retrieved solution becomes closer to the desired state $z_d(x)$.

As second test, we consider the following choice of coefficients and a non continuous desired state

$$\begin{aligned} a = 1,\ l = 0.2,\ r = 0.8,\ b(x) = 1 + 0.3\sin(20\pi x), \\ c(x) = 35\exp(-x^2/2),\ 2\alpha = 1.1, \\ z_d(x) = H(x - 1/3) - H(x - 2/3), \end{aligned} \tag{6.29}$$

where $H(x)$ is the Heaviside step function [2]. In Figure 6.1 we compare the solution obtained with different values of the regularization parameter λ. The decrease in the error is smaller than the one of the previous experiment with the three values of $\lambda = 10^{-3}, 10^{-5}, 10^{-9}$, but the latter has to be expected because the desired state $z_d(x)$ is indeed discontinuous being the sum of the two Heaviside functions.

Finally, we consider a constrained problem with the Riesz space–fractional diffusion equation (6.14), i.e., the following choice of coefficients for the equation (6.13)

$$\begin{aligned} K_{x_1} = K_{x_2} = 3, c(x_1, x_2) = 0.5, \\ \mathbf{b} = (2 + \cos(\pi/3)\cos(x_1)\sin(x_2), 2 - \sin(\pi/3)\sin(x_1)\cos(x_2)), \end{aligned} \tag{6.30}$$

as values of the fractional order of differentiation $2\alpha = 2\beta = 1.8$ and regularization parameter $\lambda = 1e-6$ on the domain $\Omega = [0,1]^2$. The desired state is the two–dimensional version of the state (6.29), i.e.,

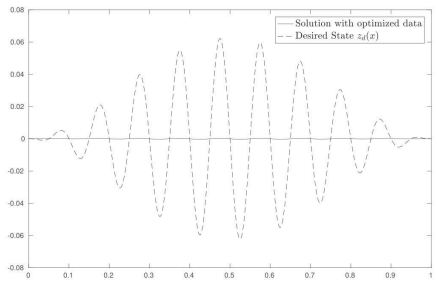
$$z_d(x, y) = (H(x - 1/3) - H(x - 2/3))(H(y - 1/3) - H(y - 2/3)). \tag{6.31}$$

A depiction of the obtained result is given in Figure 6.2.

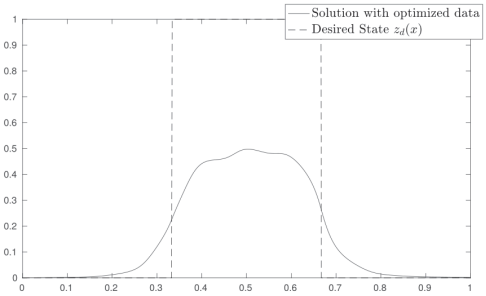
6.3.2 Accelerating Convergence

The 1D Case: FADE. Let us consider a problem similar to the one in equation (6.28), given the following choice of coefficients and desired state:

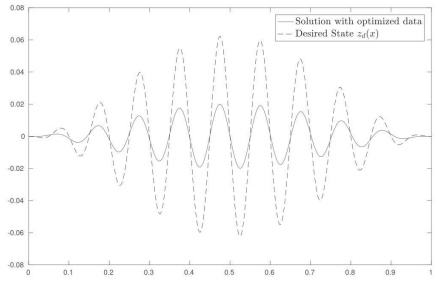
$$\begin{aligned} a = 3,\ l = 0.8,\ r = 0.2,\ b(x) = 1/2x^2 + \sin(x), \\ c(x) = 2 + \exp(-x),\ 2\alpha = 1.8,\ z_d(x) = \sin(10\pi x), \end{aligned} \tag{6.32}$$



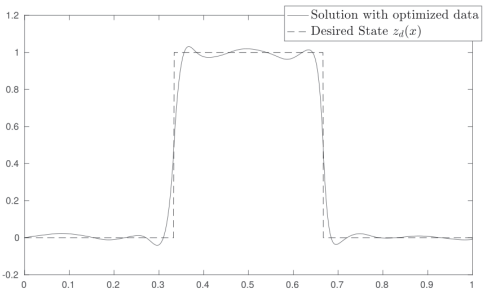
(a) $\lambda = 10^{-3}$, $\|y - z_d\|_2 = 8.87e-04$



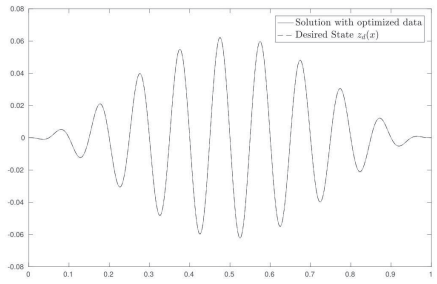
(b) $\lambda = 10^{-3}$, $\|y - z_d\|_2 = 1.05e-02$



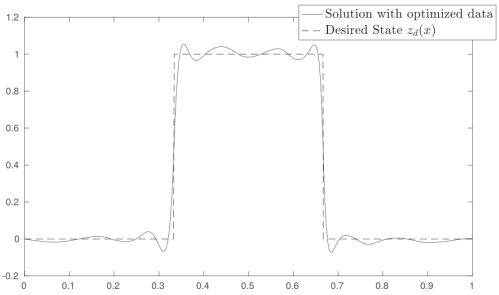
(c) $\lambda = 10^{-5}$, $\|y - z_d\|_2 = 6.07e-04$



(d) $\lambda = 10^{-5}$, $\|y - z_d\|_2 = 2.17e-03$



(e) $\lambda = 10^{-9}$, $\|y - z_d\|_2 = 4.97e-07$



(f) $\lambda = 10^{-9}$, $\|y - z_d\|_2 = 1.77e-03$

Figure 6.1. Left column: coefficients and desired state from equation (6.28). Right column: coefficients and desired state from equation (6.29).

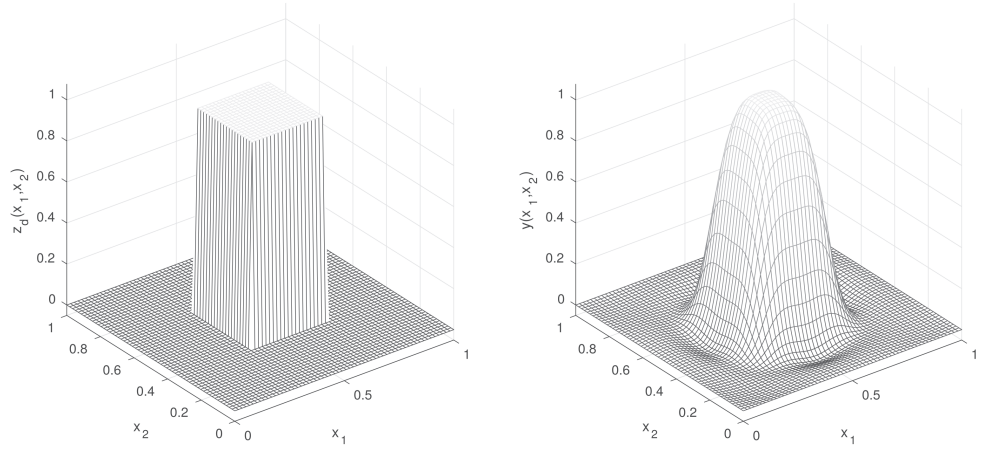


Figure 6.2. Desired state (on the left) and result of the optimization procedure (on the right) for Problem (6.14) with coefficients from equation (6.30), $2\alpha = 2\beta = 1.8$ and regularization parameter $\lambda = 1e-6$.

setting the regularization parameter λ to $\lambda = 1e-9$. As a solution method for the linear systems, we use the *BiCGstab* method with a tolerance on the residual of $\varepsilon = 1e-6$ and a maximum number of iteration MAXIT = 1000. We test both the solution with the unpreconditioned method and the one preconditioned with the strategy described in Section 6.2.2, that is the *approximate inverse* preconditioner with a drop–tolerance of $\delta = 1e-1$ and $\delta = 1e-2$. The obtained results are given in Table 6.1. We have reported both the mean number of *BiCGstab* iterations needed for the solution of the linear systems associated to the state and adjoint equation together with the total time needed for the complete optimization routine. Both the number of iterations and the execution time are sensibly reduced by this preconditioning approach. Moreover, when the size of the matrices increases the unpreconditioned *BiCGstab* does not reach convergence within the permitted maximum number of iteration, so the symbol † is reported. If we do the comparison with the solution with a direct method, namely the \ operator in Matlab, we observe that the iterative method becomes more convenient as the dimension increases.

The 2D Case: Riesz Space Fractional Diffusion. Let us start from the same choice of coefficients from equation (6.30), with $2\alpha = 2\beta = 1.8$ and regularization parameter $\lambda = 1e-6$, see also Figure 6.2. We test again the acceleration obtained by applying the AINV preconditioner from Section 6.2.2 in conjunction with the *BiCGstab* algorithm inside the routine for the computation of $f(u)$ and $\nabla f(u)$. In both cases the

	BiCGstab					
n	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)
100	10109	10770	59	9.25E-05	21	1.047
500	38997	40422	48	4.52E-04	16	9.0983
1000	†	†	†	†	†	†
2500	†	†	†	†	†	†
5000	†	†	†	†	†	†
	BiCGstab+ AINV(1e-1)					
n	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)
100	2220	2124	98	8.46E-05	34	0.2979
500	4221	4202	50	4.57E-04	16	1.2036
1000	4727	5047	34	9.15E-04	11	2.3365
2500	8676	10084	28	2.39E-03	8	6.4128
5000	†	†	†	†	†	†
	BiCGstab+ AINV(1e-2)					
n	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)
100	1161	1126	99	8.97E-05	35	0.2003
500	2339	2413	62	4.82E-04	22	0.8124
1000	2448	2442	35	9.30E-04	11	1.4517
2500	3699	4148	27	2.47E-03	8	**4.0436**
5000	7115	7678	27	3.434E-03	8	**31.4463**
	Direct					
n			Func. Eval.	$\|\nabla f\|_2$	IT	T(s)
100			89	9.40E-05	31	**0.0423**
500			60	2.30E-04	20	**0.2932**
1000			35	9.24E-04	11	**1.2968**
2500			27	2.41E-03	8	22.5293
5000			27	3.42E-03	8	62.6766

Table 6.1. Solution of test problem with coefficient from equation (6.32). A † is reported when *BiCGstab* iterated MAXIT times but did not converge. The number of matrix vector products $\sharp A\mathbf{v}$ and $\sharp A'\mathbf{v}$ needed for solving, respectively, the state and adjoint equations is given together with the overall solution time for the procedure. In boldface we have reported the best time.

method is set to achieve a tolerance on the residual of $\varepsilon = 1e-6$, while the threshold for the AINV is set either to $1e-1$ or to $1e-2$. Results are given in Table 6.2. Both the number of matrix–vector product and the time needed for the solving all the problems are reduced by the two preconditioning routines. Moreover, as expected, the iterative approach outperforms the direct solver.

As a last example, we consider the following choice of coefficients for the equation (6.13)

$$\begin{aligned} K_{x_1} = 2,\ K_{x_2} = 1.5,\ c(x,y) = 1 + 0.5\cos(x_1 x_2), \\ \mathbf{b} = (\beta + 0.5\sin(4\pi x_1)\cos(5\pi x_2), \dots \\ \dots \alpha + 0.7\sin(7\pi x_2)\cos(4\pi x_1)), \end{aligned} \tag{6.33}$$

again on the domain $\Omega = [0,1]^2$. The results are obtained for the desired state

$$z_d(x_1, x_2) = \sin(5\pi x_1)\sin(5\pi x_2). \tag{6.34}$$

The order of the fractional derivatives $2\alpha, 2\beta$, and the regularization parameter λ are given in Tables 6.3 and 6.4, together with the performances of the method. We can observe that the problem becomes more difficult, both for the solution of the FPDEs and from the optimization point of view, for the lower values of the order α and β, in accordance with what observed in [35], since for lower orders decay in the matrices becomes weaker. Nevertheless, the effect of the preconditioner still gives good performances in reducing the number of matrix vector products and thus the time needed for the solution of the problem.

The Semilinear Case. We consider the same choice of coefficients of the linear case in (6.33), while selecting $\zeta = 3$ for the semilinear exponent in (6.18). We consider again the oscillating desired state from Equation (6.34). The results are collected in Table 6.5 and are indeed consistent with the ones from the other cases. The solution of the semilinear equation is computed with the classic Newton–Krylov method [168] with no line–search and preconditioned *BiCGstab* as solver. We observe that the convergence of the Newton iteration does not need any particular care for this case: it reaches a norm accuracy of the residual of $1e-6$ with a number of iterations, which ranges between 2 and 6, being initialized with the constant function $y^{(0)} \equiv 1$. For this reason we ruled out both the necessity of using an update of the preconditioners, and the necessity of using a line–search.

Clearly, in the case of more complex nonlinearities, for which the Newton method could require more than the few iterations needed in this case, the possibility of updating the preconditioners gives a

Preconditioner		*BiCGstab*, $2\alpha = 2\beta = 1.8$, $\lambda = 1e-6$						Direct	
	I	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	5001	5064	94	2.10e-04	24	0.4636	24	0.8023
	40	8191	8740	76	1.27e-03	19	2.1135	19	10.62
	60	11770	12626	78	2.15e-03	19	11.1975	19	96.41
	80	14424	15449	74	4.91e-03	18	48.7579	18	409.24
	AINV(1e-1)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	T(s)	T(s)		
N	20	2625	2792	94	2.04e-04	24	0.3013	24	0.8023
	40	4185	4554	76	1.26e-03	19	1.2978	19	10.62
	60	6825	6747	78	2.14e-03	19	7.9346	19	96.41
	80	8222	7954	74	4.915e-03	18	28.6939	18	409.24
	AINV(1e-2)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	T(s)	T(s)		
N	20	1138	1198	94	2.14e-04	24	**0.1931**	24	0.8023
	40	1753	1849	76	1.27e-03	19	**0.7884**	19	10.62
	60	2617	2701	78	2.15e-03	19	**5.6228**	19	96.41
	80	3201	3292	74	4.91e-03	18	**14.9991**	18	409.24

Table 6.2. Results for the solution of Problem (6.14) with coefficients (6.30) and desired state (6.31). The number of matrix vector products $\sharp A\mathbf{v}$ and $\sharp A'\mathbf{v}$ needed for solving, respectively, the state and adjoint equations is given together with the overall solution time for the procedure. In boldface we have reported the best time.

Preconditioner		*BiCGstab*, $2\alpha = 1.3, 2\beta = 1.2, \lambda = 1e-6$						Direct	
	I	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	46476	73524	976	2.71e-04	26	4.9912	28	6.23
	40	80834	130061	899	1.15e-03	27	25.4829	27	146.40
	60	92609	158198	745	3.19e-03	24	116.5496	24	1669.18
	80	108270	197247	710	5.49e-03	21	488.6336	21	6029.44
	AINV(1e-1)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	20832	41173	1155	2.99e-04	28	3.2712	28	6.23
	40	36156	73400	976	1.21-03	27	16.1750	27	146.40
	60	51793	109713	958	3.35e-03	24	89.8328	24	1669.18
	80	66275	140296	933	5.70e-03	21	385.6470	21	6029.44
	AINV(1e-2)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	10216	12361	947	3.22e-04	28	**1.7162**	28	6.23
	40	24771	38206	1497	1.40e-03	27	**14.2141**	27	146.40
	60	20346	34563	836	2.99e-03	24	**46.0631**	24	1669.18
	80	33924	59922	1079	4.37e-03	21	**207.9970**	21	6029.44

Table 6.3. Results for the solution of Problem (6.14) with coefficients (6.33) and desired state (6.34). The number of matrix vector products $\sharp A\mathbf{v}$ and $\sharp A'\mathbf{v}$ needed for solving, respectively, the state and adjoint equations is given together with the overall solution time for the procedure; IT denotes the number of iterations needed by the L–BFGS algorithm. In boldface we have reported the best time.

	Preconditioner	*BiCGstab*, $2\alpha = 1.1, 2\beta = 1.8, \lambda = 1e-9$						Direct	
	I	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	34008	40346	380	3.02e-04	50	3.1414	52	2.67
	40	49603	58500	245	1.22e-03	42	13.0047	44	32.82
	60	81056	94216	237	3.02e-03	37	81.4658	36	335.74
	80	121244	140609	233	5.21e-03	32	421.9529	32	1432.13
	AINV(1e-1)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	13098	12349	388	3.11e-04	52	1.3566	52	2.67
	40	15854	14694	218	1.20e-03	44	4.8374	44	32.82
	60	23496	24406	237	3.15e-03	37	30.9931	36	335.74
	80	29293	29717	215	5.18e-03	32	112.3551	32	1432.13
	AINV(1e-2)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	5715	6036	382	3.03e-04	50	**0.8399**	52	2.67
	40	7682	7841	228	1.15e-03	39	**3.3400**	44	32.82
	60	13626	13914	238	3.18e-03	36	**25.2802**	36	335.74
	80	18825	20112	219	5.21e-03	32	**90.6548**	32	1432.13

Table 6.4. Results for the solution of Problem (6.14) with coefficients (6.33) and desired state (6.34). The number of matrix vector products $\sharp A\mathbf{v}$ and $\sharp A'\mathbf{v}$ needed for solving, respectively, the state and adjoint equations is given together with the overall solution time for the procedure; IT denotes the number of iterations needed by the L–BFGS algorithm. In boldface we have reported the best time.

	Preconditioner	*BiCGstab*, $2\alpha = 1.8, 2\beta = 1.1, \lambda = 1e-9$						Direct	
	I	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	104556	124313	850	3.01E-04	71	22.8574	72	16.5558
	40	311377	391300	1028	1.36E-03	91	208.1587	91	391.5316
	60	407443	614429	846	2.63E-03	69	1484.9491	69	2933.0407
	80	†	†	†	†	†	†	†	> 6 h
	AINV(1e-1)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	25332	35125	863	3.03E-04	71	10.4899	72	16.5558
	40	67634	131694	1021	1.36E-03	91	103.6186	91	391.5316
	60	96839	230940	866	2.63E-03	69	620.5231	69	2933.0407
	80	95508	247277	623	5.08E-03	58	1511.2287	†	> 6 h
	AINV(1e-2)	$\sharp A\mathbf{v}$	$\sharp A'\mathbf{v}$	Func. Eval.	$\|\nabla f\|_2$	IT	T(s)	IT	T(s)
N	20	9571	13331	870	3.04E-04	71	**6.1979**	72	16.5558
	40	23730	32888	1042	1.35E-03	91	**54.4773**	91	391.5316
	60	29274	52081	862	2.63E-03	69	**291.2442**	69	2933.0407
	80	30957	56806	623	5.08E-03	58	**633.7291**	†	> 6 h

Table 6.5. Results for the solution of Problem (6.18) with coefficients (6.33), with $\zeta = 3$ and desired state (6.34). The number of matrix vector products $\sharp A\mathbf{v}$ and $\sharp A'\mathbf{v}$ needed for solving, respectively, the state and adjoint equations is given together with the overall solution time for the procedure; IT denotes the number of iterations needed by the L–BFGS algorithm. In boldface we have reported the best time.

further acceleration to the overall solution time; see, e.g., the numerical examples in [15, 33, 35].

Remark 6.3. *We also observe that the number of L–BFGS iterations decreases within respect to the refinement of the grid, i.e., finer grids correspond to fewer iterations and fewer function evaluations, see Tables 6.1, 6.2, 6.3 and 6.5. This has to be expected, since to finer grids correspond higher accuracy in the computation of both f and ∇f. On the other hand, when λ decreases the number of L–BFGS iterations grows accordingly.*

II

Toeplitz–Like Structure and Preconditioners

"If there is one thing in mathematics that fascinates me more than anything else (and doubtless always has), it is neither number nor size, but always form. And among the thousand-and-one faces whereby form chooses to reveal itself to us, the one that fascinates me more than any other and continues to fascinate me, is the structure hidden in mathematical things"

A. Grothendieck, Recoltes et semailles, réflexions et témoignage sur un passé de mathématicien

7

Optimizing a Multigrid Runge–Kutta Smoother

The solution of unsteady viscous flow problems is of crucial importance in fluid dynamics. In [32] we studied the numerical solution of an unsteady flow problem, following the approach by Birken [45]. The latter uses a multigrid strategy with a special type of smoother of Runge–Kutta type, which is designed to be parallel and adapted to the problem. We recall that such problems have gained increasing attention in the last fifteen years due, probably, to achievements regarding the steady state case. We refer to the work by Caughey and Jameson [74], where it is shown that a steady two dimensional flow around airfoils can be treated in a few seconds on a PC, by using a very limited number (at most five) multigrid steps. The unsteady case is substantially more difficult and the multigrid strategy tuned on the steady state case deteriorates dramatically when moving to the unsteady state setting. To this end, see the papers [217, 218], in which a coarsening that "follows" the flow is used, thus producing flow related aggregates, or the proposals [43, 145, 294], in which multigrid solvers for the convection–diffusion equation are investigated, both on the smoothing and on the coarsening sides.

As already mentioned, in the constant coefficient case this issue is considered in [45], with the use of a specific class of smoothers based on explicit Runge-Kutta methods, which show a low storage requirement and scale well in parallel. The tuning of a small number of parameters makes the whole strategy very fast and efficient.

Inspired by the strategies in [45, 281], where the solution of PDEs with constant coefficients is considered, in this chapter, we propose a generalization useful for variable coefficient convection-diffusion linear PDEs. This generalization requires different strategies than in, e.g, [47], while we take inspiration from the approach in [37]. In particular, the theory of Generalized Locally Toeplitz matrix sequences (GLT) from Section 2.2.1 is used, which can be viewed as a generalization of the Local Fourier Analysis in which variable coefficients, irregular domains,

nonconstant time or space stepping can be taken into account [254, 255]. In particular, for matrix sequences coming from the approximation by local methods (finite differences, finite elements, discontinuous Galerkin, isogeometric analysis etc.), there exists a function, often called symbol, constructed by using the approximation method and the data of the problem (variable coefficients, domain, geometry), which describes the global behavior of the singular values of the resulting matrices and also the standard eigenvalues, under additional assumptions. We refer the reader to [14, 98, 101, 123, 254] and references therein for the application of the GLT theory to approximation methods for PDEs using finite differences, finite elements, discontinuous Galerkin and isogeometric analysis. We need such tools both for performing a spectral analysis of the matrices of the underlying problem and for building our acceleration technique, generalizing the results in [45]. In addition, as an expected but new byproduct, this contribution represents the first application, to the best of our knowledge, of the GLT theory to a discretization of PDEs by using finite volumes.

The remaining part of this chapter is organized as follows. In Section 7.1 a spectral analysis of the underlying sequence of matrices is performed, by using tools taken from the theory of GLT sequences; see Section 2.2.1. In Section 7.2 the proposed multigrid accelerator is introduced, by employing a Runge–Kutta smoother with coefficients optimized over the matrix spectrum, asymptotically described in Section 7.1. The setting of the classical geometric multigrid is used here. Nevertheless, a great part of the analysis can be extended directly to be applied to the algebraic multigrid case as well.

The optimized algorithms we propose are tested on a linear variable coefficient 2D convection–diffusion initial-boundary value problem. The aim is to show that the framework of GLT matrix sequences permits to extend the results obtained in [45, 148] for variable coefficients.

7.1 *The Model Problems and the GLT Spectral Analysis*

The section is divided into four interconnected parts stemming from Section 2.2.1 in which we defined Toeplitz and circulant matrices, and we introduced the GLT matrix sequences and their main features. In Section 7.1.1 we introduce the 1D model problem, its basic finite volume approximation, and we study the spectral properties of the resulting matrix sequence in the GLT setting. In Section 7.1.2 we consider 2D problems. In Section 7.1.3 we study other discretizations. Finally, in Section 7.1.4 we include a convection term in the convection–diffusion model considered in Section 7.1.1.

7.1.1 The 1D Model Problem

We start by considering the solution of the one dimensional in space linear transport equation, with variable coefficients and periodic boundary conditions, given by

$$\begin{cases} u_t + (q(x)u(x,t))' = 0, & x \in (\alpha,\beta),\ q(x) > 0\,\forall\, x \in [\alpha,\beta], \\ u(\alpha,t) = u(\beta,t), & \forall\, t \geq 0. \end{cases} \tag{7.1}$$

Since we want to make use of the GLT theory introduced in Section 2.2.1 and, in particular, of Definition 2.11, we define the following change of variables

$$a(x) = q(y(x)), \qquad y(x) = \alpha + (\beta - \alpha)x,$$

if we set the grid for the discretization over the domain $[\alpha,\beta]$ with n grid points, i.e., $\Delta x = (\beta-\alpha)/(n-1)$, then we obtain the following discretization matrix for the finite volume scheme

$$\begin{cases} u_1 - u_n = 0, \\ \dfrac{a_{j+1/2}u_{j+1/2} - a_{j-1/2}u_{j-1/2}}{\Delta x} = 0, \quad j = 2,\dots,n. \end{cases} \Rightarrow$$

$$\Rightarrow B_n = \begin{bmatrix} a_{1/2} & 0 & \dots & -a_{1/2} \\ -a_{3/2} & a_{5/2} & & \\ & \ddots & \ddots & \\ & & -a_{n-1/2} & a_{n+1/2} \end{bmatrix}.$$

Observe that in matrix form we obtain the entries related to the boundary values multiplied by $a_{1/2}$. In this way we have the same solution for the problem and the following semi–discrete formulation,

$$\mathbf{u}_t + \frac{1}{\Delta x} B_n \mathbf{u} = 0, \tag{7.2}$$

and, by discretizing it by *implicit Euler* method with time step Δt, we obtain the fully discretized system

$$A_n \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)}, \ \text{ where } \ A_n = I + \frac{\Delta t}{\Delta x} B_n,\ k = 0,1,\dots. \tag{7.3}$$

If $a(x) = 1\ \forall\, x \in [0,1]$, then the matrix B_n is the circulant matrix formed as $T_n(1-\exp(-i\theta)) + \mathbf{e}_1\mathbf{e}_n^T$, $\mathbf{e}_j$ being the jth vector of the canonical basis of $\mathbb{R}^n$, and with B_n having as eigenvalues the function $1-\exp(-i\theta)$ evaluated at

$$\frac{2\pi j}{n}, \quad j = 0,\dots,n-1.$$

As a consequence the eigenvalues of A_n are given by

$$\lambda_n\left(\frac{2\pi j}{n}\right),$$

with $\lambda_n(\theta) = 1 + \frac{\Delta t}{\Delta x}(1 - \exp(-i\theta))$.

In the variable coefficient case this is not true anymore, see for example Figure 7.1, and the theory of GLT sequences can help, at least for the singular values, since the matrices are highly non–Hermitian.

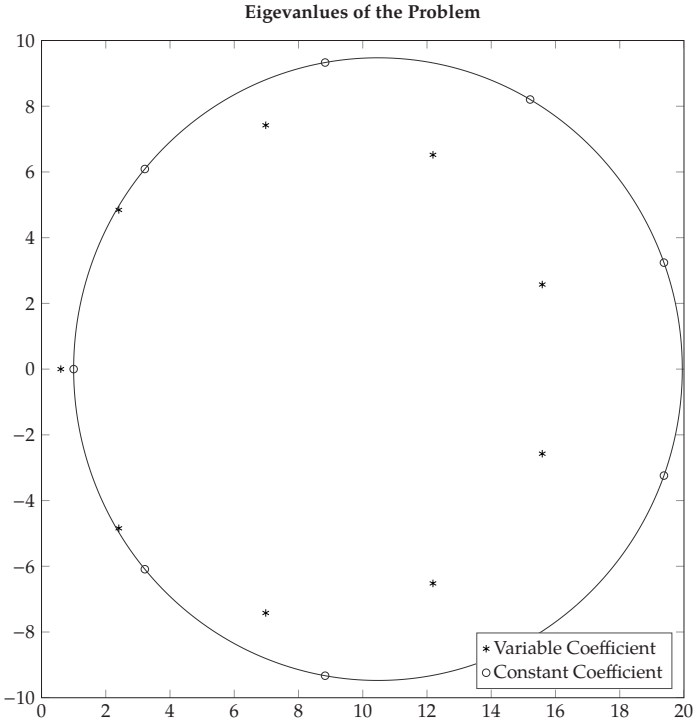


Figure 7.1. Distribution of the eigenvalues of A for the cases $a(x) \cong 1$ and a generic $a(x)$.

Our aim is to prove that the sequence of matrices $\{B_n\}_n$ obeys the diagram below, where $n-1$ is the number of discretization cells. Therefore, we set n grid points, and get the function g, the measurable function that represents the distribution of the singular values for the generic matrix of the sequence $\{B_n\}_n$. See Definition 2.6 and property **GLT5** from Proposition 2.9. Then, $\sim_\sigma$ means "having as asymptotic singular value distribution like"(Definition 2.6).

$$\begin{array}{ccc} \{\{B_{n,m}\}_n\}_m & \xrightarrow{m \to +\infty} & \{B_n\}_n \\ \wr & & \wr \\ g_m & \xrightarrow[m \to +\infty, \text{a.e.}]{} & g \end{array} \tag{7.4}$$

Proposition 7.1 (Bertaccini, Donatelli, Durastante, and Serra-Capizzano [32]). *$\{B_n\}_n$ is an LT sequence in the sense of Definition 2.11 for the kernel*

$$\kappa(x,\theta) = a(x)(1-\exp(-i\theta))$$

if $a(x)$ is continuous. Moreover the functions $c(m), n_m$, and $\omega(m)$ can be chosen as

$$c(m) = \frac{1}{m}, \quad n_m = m^2 + m, \quad \text{and} \quad \omega(m) = 2\omega_a(1/m),$$

where $\omega_a(\cdot)$ is the modulus of continuity of the function a.

Proof. We start building the matrices $R_{n,m}$ for our case. Without loss of generality we suppose that we are working with an m such that $m|n$. Then, we split the generic matrix $B_n \in \mathbb{R}^{n\times n}$ into m blocks of size $\frac{n}{m}\times\frac{n}{m}$,

$$B_n = \begin{bmatrix} S_1 & & & * \\ * & S_2 & & \\ & * & \ddots & \\ & & * & S_m \end{bmatrix} = B_{n,m} + R_{n,m} + N_{n,m}, \quad (7.5)$$

we set the matrix $R_{n,m}$ as the matrix built from the entries of the previous step of the decomposition, denoted by a "$*$" in (7.5). In this way we find

$$\text{Rank}(R_{n,m}) = m + 1 \leq \frac{n}{m} = c(m)n, \quad \forall n \geq n_m = m^2 + m, \quad (7.6)$$

and thus, $c(m) = \frac{1}{m} \to 0$.

Let us observe that the generic block S_j for $j \geq 1$ is made up from the evaluation of the function $a(x)$ over the points $\left\{z_i = \frac{i+1/2}{n} : \frac{j-1}{m} < z_i < \frac{j}{m}\right\}$. Thus, we can write

$$S_j = a(j/m)T_{\frac{n-1}{m}}(1-\exp(-i\theta)) + E^{(j)}_{n,m}, \quad \forall j \geq 1,$$

$$\text{where } E^{(j)}_{n,m} = \begin{bmatrix} \alpha^{(j)}_1 & & \\ \alpha^{(j)}_2 & \alpha^{(j)}_3 & \\ & \ddots & \ddots \end{bmatrix}. \quad (7.7)$$

By the continuity of the function $a(x)$, we can bound each entry of the block $E_{n,m}^{(j)}$ as

$$|\alpha_i^{(j)}| \leq \sup_{t \leq 1/m} |a(t) - a(i)| = \omega_a(1/m) \overset{m\to\infty}{\longrightarrow} 0.$$

In this way we write the two sequences of matrices $\{B_{n,m}\}$ and $\{N_{n,m}\}$ as

$$\begin{aligned} B_{n,m} =& \text{blockdiag}\left(a(1/m)T_{\frac{n-1}{m}}(1-\exp(-i\theta)), \dots \right. \\ & \left. \dots, a(1)T_{\frac{n-1}{m}}(1-\exp(-i\theta))\right), \\ N_{n,m} =& \text{blockdiag}(E_{n,m}^{(1)}, \dots, E_{n,m}^{(m)}). \end{aligned} \tag{7.8}$$

Moreover we have

$$\|N_{n,m}\| \leq \|N_{n,m}\|_1 \leq 2\omega_a(1/m) \overset{m\to\infty}{\longrightarrow} 0,$$

and the sequence of matrices $\{B_n\}_n$ is a LT sequence (Definition 2.11). □

Proposition 7.2 (Bertaccini, Donatelli, Durastante, and Serra-Capizzano [32])**.** *For any m, the matrix sequence $\{\{B_{n,m}\}_n\}_m$ in* (7.8) *is distributed in the sense of the singular values as the function*

$$\begin{aligned} g_m(x,\theta) =& a_m(x)\left(1-\exp(-i\theta)\right), \\ \text{where } a_m(x) =& \sum_{j=1}^{m} a(j/m)\chi_{[j/m,(j+1)/m]}(x), \end{aligned}$$

where $D = [0,1]\times[-\pi,\pi]$ (see Definition 2.6) and $\chi_{[j/m,(j+1)/m]}(x)$ is the characteristic function of the set $[j/m,(j+1)/m]$.

Proof. Let F be a generic $\mathcal{C}_c(\mathbb{R}_0^+)$. We have

$$\begin{aligned} \lim_{n\to+\infty} \sum_{j=1}^{m} \frac{F(\sigma_j(B_{n,m}))}{n} &= \lim_{n\to+\infty} \frac{1}{m} \sum_{j=1}^{m} \sum_{k=1}^{n/m} \frac{F\left(\left|a\left(\frac{j}{m}\right)\left(1-\exp\left(-i\frac{k\pi}{n/m+1}\right)\right)\right|\right)}{n/m} \\ \left(\nu = \frac{n}{m}\right) \dashrightarrow &= \lim_{\nu\to+\infty} \sum_{j=1}^{m} \frac{1}{m} \sum_{k=1}^{\nu} \frac{F\left(\left|a\left(\frac{j}{m}\right)\left(1-\exp\left(-i\frac{k\pi}{\nu+1}\right)\right)\right|\right)}{\nu}. \end{aligned}$$

Then, by using the singular values distribution results for the product of diagonal and Toeplitz matrices and the overall block-diagonal structure of the matrix, we obtain that the latter displayed quantity coincides with

$$\frac{1}{m}\sum_{j=1}^{m}\frac{1}{2\pi}\int_{-\pi}^{\pi}F\left(\left|a\left(\frac{j}{m}\right)(1-\exp(-i\theta))\right|\right)d\theta$$
$$=\frac{1}{2\pi}\int_{-\pi}^{\pi}\int_{0}^{1}F\left(\left|a_m(x)(1-\exp(-i\theta))\right|\right)d\theta\,dx. \qquad \square$$

Proposition 7.3 (Bertaccini, Donatelli, Durastante, and Serra-Capizzano [32]). *Given B_n as in Proposition 7.1, given $B_{n,m}$ as in* (7.8)*, considered*

$$g(x,\theta)=a_m(x)\big(1-\exp(-i\theta)\big)$$

as in Proposition 7.2 and $g(x,\theta)=a(x)\big(1-\exp(-i\theta)\big)$, the commuting diagram in (7.4) *holds true.*

Proof. The lower horizontal relation in (7.4) is clear since

$$\begin{aligned} g_m(x,\theta)=a_m(x)\big(1-\exp(-i\theta)\big) &\overset{m\to+\infty}{\longrightarrow}\\ \overset{m\to+\infty}{\longrightarrow} g(x,\theta)=a(x)\big(1-\exp(-i\theta)\big) &\quad \text{a.e.}. \end{aligned} \tag{7.9}$$

The upper horizontal relation in (7.4) amounts in claiming that $\{\{B_{n,m}\}_n\}_m$ is an approximating class of sequences for $\{B_n\}_n$, the latter being proved in Proposition 7.1. Finally, the left vertical relation is proven in Proposition 7.2, while the right one follows from Proposition 7.1, because, by definition, any LT matrix sequence is a GLT matrix sequence with the same symbol and, by item **GLT1**, the GLT symbol is also the singular value distribution function. $\square$

In the previous proposition we have proved the commuting diagram described in (7.4) and in particular the relation $\{B_n\}_n \sim_\sigma g$. Now we show as the machinery contained in items **GLT1-GLT5** can be used for computing the singular value distribution for $\{A_n\}_n$.

We first assume that the ratio between the step sizes $\Delta t, \Delta x$ coincides with the constant c, or it tends to c. Hence by item **GLT3**, $\{I_n\}_n$ and $\{\frac{\Delta t}{\Delta x}I_n\}_n$ are GLT sequences with constant symbols 1 and c, respectively. Then using item **GLT4** (that is the $*$-algebra structure of the GLT sequences), we deduce that $\{\frac{\Delta t}{\Delta x}B_n\}_n$ is a GLT sequence with symbol cg and finally, again by item **GLT4** and since $A_n = I_n + \frac{\Delta t}{\Delta x}B_n$, we infer that $\{A_n\}_n$ is a new GLT sequence with symbol

$$\lambda(x,\theta) = 1 + c\,g(x,\theta) = 1 + ca(x)\big(1 - \exp(-i\theta)\big). \tag{7.10}$$

In the style of [45], to proceed with the analysis of the model problem and define the smoother, we use a *dual time stepping approach* to solve the equation (7.3). In this way we obtain a hyperbolic equation with the addition of the *pseudo time* t^*

$$\begin{cases} u_{t^*} = u^{(k)} - A_n u(t^*), \\ u(t_0^*) = u^{(k)}. \end{cases} \tag{7.11}$$

Therefore, we have two different discretizations of the model problem. If we are interested in the *steady state solutions*, then we work with the discretization (7.3). On the other hand, for *unsteady problems*, we need to work with the dual time stepping approach in equation (7.11).

7.1.2 The 2D model problem

Let us consider the 2D version of equation (7.1) with periodic boundary conditions on $\Omega = Q \times \mathbb{T} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [t_{\min}, t_{\max}]$,

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} + \dfrac{\partial a(x)u(x,y,t)}{\partial x} + \ldots & (x,y) \in Q, \\ \qquad \ldots + \dfrac{\partial b(y)u(x,y,t)}{\partial y} = 0, & t \in \mathbb{T}, \\ u(x_{\min}, y, t) = u(x_{\max}, y, t), & y \in \partial\Omega, \\ u(x, y_{\min}, t) = u(x, y_{\max}, t), & x \in \partial\Omega, \\ u(x,y,0) = u_0(x,y). \end{cases} \tag{7.12}$$

The aim is to use it as a step to get the advection-diffusion equation. We discretize (7.12) over a grid of n_1 elements on the x direction and n_2 elements over the y direction, respectively, using *backward differences* of order 1. We prefer to introduce numerical diffusion errors instead of the numerical dispersion we would have with centered schemes[1]. As a consequence, in semi-discrete form for $i = 2, \ldots, n_1 - 1$, and $j = 1, \ldots, n_2 - 1$, we find

$$u_t + \frac{a_i u_{i,j} - a_{i-1} u_{i-1,j}}{\Delta x} + \frac{b_j u_{i,j} - b_{j-1} u_{i,j-1}}{\Delta y} = 0. \tag{7.13}$$

1 Note, however, that the spectral analysis, with appropriate modifications with respect to the Fourier symbol of the underlying schemes, remains valid even if we decide to use centered differences. See [240] for diffusive/dispersive behaviour of these schemes.

By using a lexicographic ordering, we take advantage of the Kronecker product representation to write the matrix of the semidiscretized problem, without boundary conditions for $\mathbf{n} = (n_1, n_2)$, that is

$$B_n = T^a_{n_1}(1 - \exp(-i\theta)) \otimes I_{n_2} + I_{n_1} \otimes T^b_{n_2}(1 - \exp(-i\theta)). \tag{7.14}$$

In this way, after we have imposed periodic boundary conditions, we get the semidiscrete equation

$$\mathbf{u}_t = -B_n \mathbf{u}(t). \tag{7.15}$$

To fully discretize this equation we consider the Crank–Nicolson method with time step Δt at time t_{k+1}:

$$(2I_{n_1 n_2} + \Delta t B_n)\mathbf{u}^{(k+1)} = (2I_{n_1 n_2} - \Delta t B_n)\mathbf{u}^{(k)}, \quad A_n = 2I_{n_1 n_2} + \Delta t B_n. \tag{7.16}$$

Note that in order to advance in time, we need to solve a sequence of linear systems with matrices A_n. By using the characterization for the eigenvalues of the Kronecker sum of two matrices (see, e.g., [181, Theorem 13.16]) and the results for GLT sequences of matrices, illustrated in (7.4), we derive the $\{A_n\}_n$ distribution in the sense of the singular values by writing the quantity to optimize as

$$\begin{aligned} \lambda(x, y, \theta_1, \theta_2) =& 2 + \frac{\Delta t}{\Delta x} a(x)(1 - \exp(-i\theta_1)) + \dots \\ & \dots + \frac{\Delta t}{\Delta y} b(y)(1 - \exp(-i\theta_2)). \end{aligned} \tag{7.17}$$

To proceed with the analysis of the 2D model problem, as for the 1D case, we use again *dual–time stepping* procedure for equation (7.16), with a fictitious time step Δt^*.

7.1.3 Further Discretizations

By means of Propositions 7.1 and 7.2 we can obtain information on the spectrum of the matrix $B_{n,m}$ for the various finite difference stencils. We use the finite difference scheme for

$$\frac{\partial a(x)u}{\partial x}$$

given by

$$\begin{aligned} & \frac{a_j u_j - a_{j-1} u_{j-1}}{\Delta x}, \\ \text{or } & \frac{3a_j u_j - 4a_{j-1} u_{j-1} + u_{j-2}}{2\Delta x}, \\ \text{or } & \frac{2a_{j+1} u_{j+1} + 3a_j u_j - 6a_{j-1} u_{j-1} + u_{j-2}}{6\Delta x}. \end{aligned} \tag{7.18}$$

By denoting with $f(\theta)$ their symbol we have

$$\begin{aligned} f(\theta) =& \frac{1-\exp(-i\theta)}{\Delta x}, \\ \text{or } f(\theta) =& \frac{3-4\exp(-i\theta)+\exp(-2i\theta)}{2\Delta x} \\ \text{or } f(\theta) =& \frac{3+2\exp(i\theta)-6\exp(-i\theta)+\exp(-2i\theta)}{6\Delta x}. \end{aligned}$$

We restate Proposition 7.3 as

Proposition 7.4. *The sequence of matrices $\{B_n\}_n$ related to one of the discretizations proposed in* (7.18) *is a GLT sequence in the sense of Definition 2.12 for the kernel $\kappa(x,\theta) = a(x)f(\theta)$, if $a(x)$ is continuous and $f(\theta)$ is the related Fourier symbol. Moreover, the function $c(m)$ and $\omega(m)$ are such that $c(m) = m+1$ and $\omega(m) = 2\omega_a(1/m)$, where $\omega_a(\cdot)$ is the modulus of continuity of the function a.*

By using the latter statements, we can give distribution results, also in dimensions greater than one as for the 2D case in the previous section.

7.1.4 The Convection-Diffusion Equation

Let us consider a second order centered finite difference discretization for the complete convection–diffusion equation model problem

$$\begin{aligned} &\frac{\partial u(x,y,t)}{\partial t} + \frac{\partial a(x)u(x,y,t)}{\partial x} + \frac{\partial b(y)u(x,y,t)}{\partial y} = \\ &\frac{\partial}{\partial x}\left(\kappa_1(x)\frac{\partial u(x,y,t)}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa_2(y)\frac{\partial u(x,y,t)}{\partial y}\right), \end{aligned} \tag{7.19}$$

where $(x,y) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ with initial and boundary conditions. By using Proposition 7.4 we deduce that $\{B_n\}_n$ is a GLT sequence. Since it shows a negligible non–Hermitian part, by using item **GLT2**, we infer that the GLT symbol represents not only the singular value distribution but also the eigenvalue distribution. Hence, for the sequence associated to the discretization matrix in equation (7.19), we have

$$\begin{aligned} \{B_n\}_n \sim_\lambda \;\; & k_2(x,y,\theta_1,\theta_2), \;\; \text{with} \\ k_2(x,y,\theta_1,\theta_2) =& \kappa_1(x)f_2(\theta_1) + \kappa_2(y)f_2(\theta_2), \end{aligned}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are the Fourier symbols of the scheme used for the discretization of the diffusive terms. By using the standard 5 point

stencil we have $f_1(\theta) = f_2(\theta) = 2 - 2\cos(\theta)$. We observe that we are showing the asymptotic behavior and hence the terms related to the first order derivatives are not present: this explains the reason why the symbol $k_2(x, y, \theta_1, \theta_2)$ does not contain the weight functions $a(x)$ and $b(y)$ (see also the following remark).

Remark 7.1. *We stress that here the asymptotic behavior of the spectra of the matrices is potentially less interesting for us than the spectral information computed for large or very large but viable values of the discretization parameters. In view of this, we prefer to avoid normalization of the symbol as in [132, Theorem 3.4] or in [127, Theorem 3.3]. Otherwise, important information about the advection term can be not visible anymore. Note that in the above mentioned references the focus was on the asymptotic properties of the spectra and in view of this, the symbol is observed to be asymptotically influenced only by the higher order derivative, i.e.,* $\kappa_1(x, y, \theta_1, \theta_2) \approx 0$, *or, more precisely,* $Z \sim_\lambda \kappa_1(x, \theta) = 0$ *because* $\|Z\|_2 \leq C/n$ *as* $n \to +\infty$. *See also [126].*

As a consequence of the remark above and of the use of a multigrid algorithm (and therefore using discretizations coarser than the starting one), we need to consider also lower order terms and step sizes in the sequel.

7.2 *The Optimized Multigrid Preconditioner*

Let us build a multigrid preconditioner to deal with the numerical solution of the linear systems generated by the discretizations described in previous sections.

In particular we focus on applying our multigrid preconditioners for Krylov subspace methods like *GMRES*(m) [249] and for *BiCGstab* [285], as described in Section 2.3.

To focus on the building of the various step of our multigrid preconditioner, we remind the construction of the standard multigrid in Algorithm 2.16, with V–cycle.

7.2.1 Runge-Kutta Smoothers

The main target of our new optimized multigrid algorithm, in the spirit of [45], is the construction of a smoother with an amplification factor that, at each level, is tuned on the spectral properties of the underlying matrices. To this end, we focus on the use of an explicit Runge–Kutta method. The final goal is the efficient solution of 2D variable coefficients convection-diffusion problems (7.19). Moreover, to make our results comparable with the one in [45], we also consider the 1D case.

There exist different formulations of the s-stage explicit Runge-Kutta methods that can be used as smoothers for multigrid algorithms.

As a first choice we start following [12, 45, 270, 281]. We select as a smoother an *explicit multistage Runge-Kutta method* written for the generic IVP in the form $\mathbf{u}_t = G(\mathbf{u})$, $\mathbf{u}_n = \mathbf{u}(t_n)$:

$$\begin{cases} \mathbf{u}^{(0)} = \mathbf{u}_n, \\ \mathbf{u}^{(j)} = \mathbf{u}_n + \alpha_j \Delta t^* G(\mathbf{u}^{(j-1)}), \quad j = 1, \ldots, s-1, \\ \mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t^* G(\mathbf{u}^{(s-1)}), \end{cases} \tag{7.20}$$

where $\Delta t^* \in \mathbb{R}$ is a free parameter, a sort of pseudo time step used to control the smoothing and $\alpha_j \in [0, 1]$. In our case we deal with a linear problem for which the dynamic $G(\cdot)$ of the IVP is given by $G(\mathbf{u}) = A\mathbf{u}$. The application of *one step* of the explicit s-stage Runge-Kutta method can be represented by using the following expression using the *stability polynomial* $P_s(z)$

$$\mathbf{u}_{n+1} = P_s(-\Delta t^* A)\mathbf{u}_n.$$

We use the Runge–Kutta scheme as a smoother. Thus we need an effective damping of the components of the error. Let us suppose that $\mathbf{u}_s$ is the solution to the IVP $\mathbf{u}_t = G(\mathbf{u})$ computed with the Runge–Kutta method with s-stages (7.20) after some step, and let us also suppose that $\mathbf{u}(t)$ is the true solution to $\mathbf{u}_t = G(\mathbf{u}) = A\mathbf{u} + \mathbf{b}$. We define the error at time t as $\mathbf{e}(t) = \mathbf{u}(t) - \mathbf{u}_s$. By subtracting the two equations $\mathbf{u}_{s,t} = G(\mathbf{u}_s)$ and $\mathbf{u}_t = G(\mathbf{u})$, we deducet that the error satisfies the *error equation*

$$\begin{aligned} \frac{d\mathbf{e}}{dt} = A\mathbf{e}, \quad &\Rightarrow \mathbf{e}^{(n+1)} = P\mathbf{e}^{(n)}, \\ &\Rightarrow P = (I + \alpha_1 \Delta t A(1 + \alpha_2 \Delta t A(1 + \ldots))). \end{aligned}$$

If we consider the spectral decomposition of A, or in the non symmetric case, its singular values, then we obtain that the amplification of the jth mode of the error is given by $|P_s(-\Delta t^* \lambda(x, \theta))|$. To have an effective damping of the error we need to optimize the quantity

$$\begin{aligned} &\left| P_s\left(-\Delta t^* - \frac{\Delta t^* \Delta t}{\Delta x} a(x)\left(1 - \exp(-i\theta)\right)\right)\right| = \\ &\left| P_s\left(-\Delta t^* - \frac{\Delta t^* \Delta t}{\Delta x} a(x) + \frac{\Delta t^* \Delta t}{\Delta x} a(x) \exp(-i\theta)\right)\right|. \end{aligned} \tag{7.21}$$

By defining

$$c = \frac{\Delta t^* \Delta t}{\Delta x}, \quad r = \frac{\Delta t}{\Delta x},$$

we obtain that (7.21) is given by

$$\left| P_s \left(-\frac{c}{r} - c\, a(x) + c\, a(x) \exp(-i\theta) \right) \right|.$$

In view of this, we define

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

where $|P_s(z)|$ is a function that can be optimized only for the values of θ, c and x, with $\theta \in [\pi/2, 3\pi/2]$. Given the standard interpolation rules from Section 7.2.2, since on the coarse grid we can represent functions with $\theta \in [-\pi/2, \pi/2]$, we optimize the smoother in the complementary set in the reference interval, $c \in [0, 1]$ and $x \in [x_{\min}, x_{\max}]$.

In general, for a generic symbol $f(\theta)$ we can write

$$z(\theta, c, x; r) = -\frac{c}{r} - c\, a(x) f(\theta),$$

that in 2D becomes

$$z(\theta_1, \theta_2, c, x, y; r) = -\frac{c}{r} - c\, a(x) f(\theta_1) - c\, b(y) f(\theta_2).$$

We can now consider the polynomials $P_2(z)$, $P_3(z)$ and $P_4(z)$ for the Runge–Kutta methods with stages 2,3 and 4, that are

$$\begin{aligned} P_2(z) &= 1 + z + \alpha_1 z^2, \\ P_3(z) &= 1 + z + \alpha_2 z^2 + \alpha_1 \alpha_2 z^3, \\ P_4(z) &= 1 + z + \alpha_3 z^2 + \alpha_3 \alpha_2 z^3 + \alpha_1 \alpha_2 \alpha_3 z^4, \end{aligned}$$

and minimize the amplification factor by working on

$$\min_{c, P_s} \max_{(|\theta|, x) \in [\pi/2, 3\pi/2] \times [x_{\min}, x_{\max}]} |P_s(z(\theta, c, x; r)))|^2 . \tag{7.22}$$

In our multigrid framework, the quantity $a(x)/\Delta x$ changes at each coarser discretization level, while Δt is fixed. Then, our optimization procedure for the pseudo time step Δt^* on each grid level depends both on the ratio $c = \frac{\Delta t^* \Delta t}{\Delta x}$ and on the values of the function a on its grid. In this way, the link between the pseudo time step and the CFL condition is exploited at each discretization level of the multigrid algorithm, i.e., we are optimizing the smoother at *each* level of the algorithm.

Remark 7.2. *We are using a small number of steps for the Runge–Kutta algorithm, thus we do not expect a zero coefficient from the optimization*

procedure. Nevertheless, this negative event is possible as it can be seen from the definition of the stability polynomial $P_s(z)$

$$P_s(z) = 1 + \sum_{l=1}^{s} \left(\prod_{i=s-l+1}^{s} \alpha_i \right) z^l = 1 + \sum_{l=1}^{s} \beta_l z^l. \tag{7.23}$$

If $\exists l < s$ *such that* $\beta_l = 0$ *then* $\forall\, k \in \{l+1, \dots, m\}$ *we would find* $\beta_k = 0$.

In light of the previous Remark 7.2, we want to stress that this kind of procedure is feasible also for other formulations of the Runge–Kutta algorithms.

Among them, following the proposal in [148], we consider the *periodic and non stationary formulation* and the *split formulation*.

We begin with the **periodic and non stationary** case. Let us consider the following s-stage algorithm

$$\begin{cases} \mathbf{u}^{(0)} = \mathbf{u}_n, \\ \mathbf{v}^{(0)} = \Delta t^* \mathbf{r}_n = \Delta t^* (A\mathbf{u}_n - \mathbf{b}), \\ \mathbf{u}^{(1)} = \mathbf{u}^{(0)} - \alpha_0 \mathbf{v}^{(0)}, \\ \mathbf{v}^{(l)} = -\Delta t^* A \mathbf{v}^{(l-1)}, \\ \mathbf{u}^{(l+1)} = \mathbf{u}^{(l)} - \alpha_{l+1} \mathbf{v}^{(l)}, \qquad l = 1, \dots, s-1, \\ \mathbf{u}_{n+1} = \mathbf{u}^{(s)}, \end{cases} \tag{7.24}$$

for $\Delta t^* \in \mathbb{R}$, from which we obtain the stability polynomial given by

$$P'_s(z) = 1 + \sum_{l=1}^{s} \alpha_l z^l. \tag{7.25}$$

Observe that if we work with the same auxiliary time step as the other formulation, then the α_is in (7.25) have the same values as the β_is in equation (7.23).

Then, we consider the **formulation** based on a **splitting of the matrix** of the linear system. We consider the splitting of the matrix $A_n = B + C$ imposing that B and C have the same eigenvectors basis as A. In this way, we can split the eigenvalues λ_i of the matrix A in the sum of $\lambda_{i,B}$ and $\lambda_{i,C}$, being, respectively, the eigenvalues of the matrices B and C, for $i = 1, \dots, n$.

Assuming this, we can formulate the algorithm as

$$\begin{cases} \mathbf{u}^{(0)} = \mathbf{u}_n, \\ \mathbf{v}^{(0)} = \Delta t^* \mathbf{r}_n = \Delta t^* (A\mathbf{u}_n - \mathbf{b}), \\ \mathbf{u}^{(1)} = \mathbf{u}^{(0)} - \alpha_1 \mathbf{v}^{(0)}, \\ \mathbf{v}^{(B,l)} = -\Delta t^* B \mathbf{v}^{(0)}, \\ \mathbf{v}^{(C,l)} = -\Delta t^* C \mathbf{v}^{(0)}, \\ \mathbf{u}^{(l+1)} = \mathbf{u}^{(l)} - \alpha_{B,l+1} \mathbf{v}^{(B,l)} - \alpha_{C,l+1} \mathbf{v}^{(C,l)}, \quad l = 1, \dots, s-1, \\ \mathbf{u}_{n+1} = \mathbf{u}^{(s)}, \end{cases} \tag{7.26}$$

for $\Delta t^* \in \mathbb{R}$, which leads to a stability polynomial of the form

$$P_s''(z_1, z_2) = 1 + \left(\alpha_1 + \sum_{l=2}^{s} \left(\alpha_{B,l} z_1^{l-1} + \alpha_{C,l} z_2^{l-1} \right) \right) (z_1 + z_2). \tag{7.27}$$

The damping relation on the error is given by

$$\mathbf{e}_{n+1} = P_s''(-\Delta t^* B, -\Delta t^* C)\mathbf{e}_n, \ \Rightarrow \ \|\mathbf{e}_{n+1}\| \leq |P_s''(-\Delta t^* \lambda_{B,i}, -\Delta t^* \lambda_{C,i})| \|\mathbf{e}_n\|.$$

For the splitting we consider

$$A_n = \frac{1}{2}(A_n + A_n^T) + \frac{1}{2}(A_n - A_n^T) = B + C,$$

which satisfies our assumptions on the eigenvectors and permits to split the real and imaginary part of the spectrum of the matrix A.

Let us now focus on the stability region

$$\mathcal{R}_s = \{ |P_s(z)| \leq 1, \ z \in \mathbb{C} \}$$

of the method obtained by optimizing the coefficients; see Section 7.2.3. In Figure 7.2 we report the stability region that has been computed for $s = 2, 3, 4$ related to the three formulations of the algorithm. The figures are plotted with the same scale, so we can observe immediately that, at least for this choice of coefficients function, the standard formulation achieves the largest stability regions. Moreover, when we use the splitted formulation, we do not improve the information gained through the optimization algorithm, even if we have added another coefficient; see Table 7.1. From what we see from this example and from the results in Section 7.3, we tend to recommend the standard formulation that, moreover, has also the lowest number of parameters to optimize for. Before taking into account the effect of the *prolongation* and *restriction* steps, we report also explicitly the way to apply the same strategy in

(a) 2 Stages

α			Δt^*	$\lvert P_s(z)\rvert$
$\alpha_1 = 0.262114$			0.520098	0.135796
$\alpha_1 = 0.886808$	$\alpha_2 = 0.206134$		0.586482	0.135796
$\alpha_1 = 1$	$\alpha_{B,1} = 0.185471$	$\alpha_{C,1} = 0.210629$	0.666667	0.167326

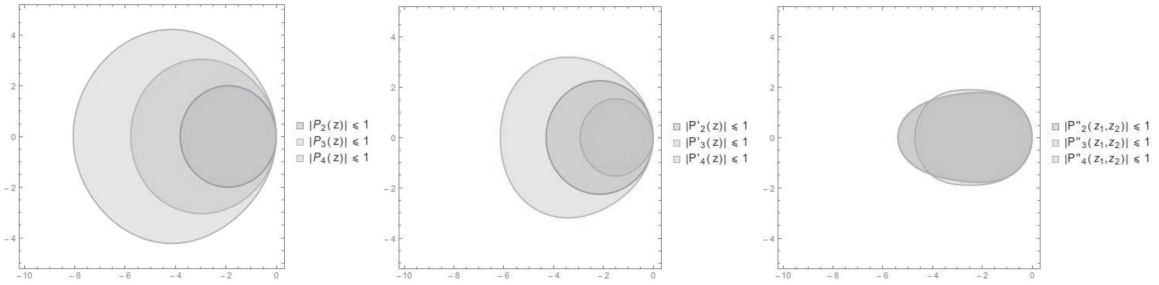
(b) 3 Stages

α			Δt^*	$\lvert P_s'(z)\rvert$
$\alpha_1 = 0.114598$	$\alpha_2 = 0.334990$		0.779855	0.049302
$\alpha_1 = 1$	$\alpha_2 = 0.310642$	$\alpha_3 = 0.032709$	0.666667	0.077337
$\alpha_1 = 1$	$\alpha_{B,1} = 0.185471$ $\alpha_{C,1} = 0.210629$	$\alpha_{B,2} = 0$ $\alpha_{C,2} = 0$	0.666667	0.167326

(c) 4 Stages

α				Δt^*	$\lvert P_s''(z)\rvert$
$\alpha_1 = 0.061469$	$\alpha_2 = 0.161492$	$\alpha_3 = 0.361230$		1.075903	0.015710
$\alpha_1 = 2.743347$	$\alpha_2 = 2.718669$	$\alpha_3 = 1.204485$	$\alpha_4 = 0.203123$	0.392181	0.015710
$\alpha_1 = 1$	$\alpha_{B,1} = 0.163546$ $\alpha_{C,1} = 0.207943$	$\alpha_{B,2} = 0$ $\alpha_{C,2} = 0.014026$	$\alpha_{B,3} = 0.002193$ $\alpha_{C,3} = 0.001084$	0.666667	0.155560

Table 7.1. Example of optimized parameters referring to the stability regions in Figure 7.2, i.e., to the (1D) convection problem with $a(x) = 1 + 0.6\sin(40\pi x)$. Each rows correspond to the standard, periodic and splitted formulation respectively



(a) Standard Formulation (b) Periodic Non Stationary Formulation (c) Splitted Formulation

Figure 7.2. Stability region for the three formulation of the Runge-Kutta algorithm for the (1D) model problem with coefficient function $a(x) = 1 + 0.6\sin(40\pi x)$.

the 2D case. We need to restart from the evaluation of the stability polynomial $P_s(z)$ over the matrix $-\Delta t^* A_{n,m}$. Suppose that $\Delta x = \Delta y$, i.e., the same discretization step in both the directions. Then define the function

$$z_2(\theta_1, \theta_2, c, x, y; r) = -2\frac{c}{r} - c(a(x) + b(y)) + \ldots \\ \ldots + c(a(x)\exp(-i\theta_1) + b(y)\exp(-i\theta_2)),$$

which can be optimized only for the values of θ, c, x and y, with $\theta_{1,2} \in [-\pi/2, \pi/2]$, $c \in [0, 1]$ and $(x, y) \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.

To optimize the amplification factor $|P_s(z_2)|$ we need to solve the following min–max problem

$$(\alpha_{\text{opt}}, c_{\text{opt}}) = \arg\min_{\alpha,c} \max_{(x,y),(\theta_1,\theta_2)} |P_s(z_2(\theta_1, \theta_2, c, x, y; r))|, \qquad (7.28)$$

where P_s can be the polynomial associated with one of the three formulations of the considered Runge–Kutta smoothers.

7.2.2 Projection and Restriction Operators

Having decided the smoother of the multigrid algorithm we can now focus on the grid-transfer operators. We are working with both 1D and 2D model problems, so we are going to choose two slightly different strategies for the two cases.

In the case of the 1D equation, as a restriction operator, we can choose to join two contiguous discretization cells into one, using the fact that we have discretized the equation with the values in the middle of the cell, and apply full–weighting, as in [45].

This is equivalent to set the restriction operation R and the projection operation P, respectively as

$$R = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & 1 \end{bmatrix}, \quad P = 2R^T = 2\begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{bmatrix}. \tag{7.29}$$

Then, instead of using Galerkin conditions, we rebuild the operator at each coarser level of the V-cycle.

In the case of the 2D equations, we choose the full–weighting transfer grid operator with mask

$$p = \begin{vmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{vmatrix}. \tag{7.30}$$

Therefore, the projector and the restriction operator with full–weighting are selected as

$$P = 2T_n(p)K_n, \quad R = \frac{1}{2}P^T.$$

As observed in [148], the choice of the objective function using only the spectral information of Section 7.1 for the optimization process does not take into account the effects of *aliasing* and *attenuation* phenomena caused by the choice of the coarsening strategy, i.e., we are not optimizing the full multigrid algorithm. From the GLT perspective, the latter can be seen as an effect of property **GLT4**, that is from the structure of $*$–algebra of the GLT sequences. In view of this, we can try to take into account the strategy in [148] for considering the *attenuation* by modifying the objective functions as explained below. Define the transmittance function $\mu_r(\theta)$ and $\mu_p(\theta)$ for the restriction and projection operations respectively, as the Fourier symbol of the Toeplitz matrix representing the operator, before the cut via the matrix K_n. For our 1D case this amounts to set

$$\mu_r = \frac{1}{2}(1 + \exp(i\theta)), \quad \mu_p = (1 + \exp(i\theta)),$$

which leads to the optimization problem

$$(\alpha_{\text{opt}}, c_{\text{opt}}) = \arg\min_{\alpha,c} \max_{(x,\theta)} \left|(1 - \mu_p(\theta)\mu_r(\theta)P_s(z(\theta, c, x, y; r))\right|. \tag{7.31}$$

By this modification, we can now expect to have always a stable smoother, and also optimized performances all over the V-cycle iteration of the algorithm. A plot of the correction factor for the linear interpolation and cell joining is reported in Figure 7.3.

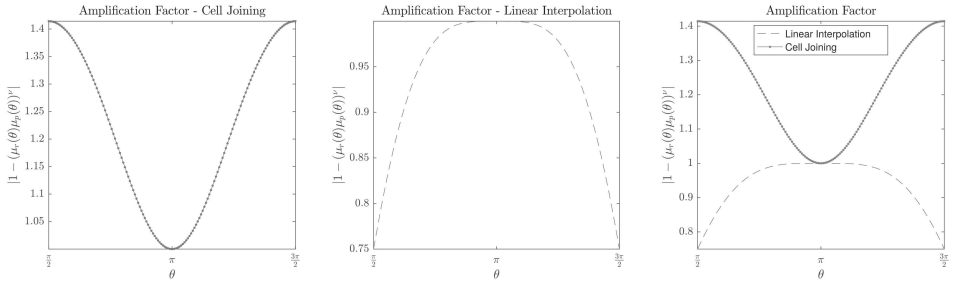


Figure 7.3. Amplification factor for different coarsening strategies.

By considering the standard formulation, we report the results in Table 7.2 from the optimization process. If we compare the results in

Order	α_1	α_2	α_3	Δt^*	$\lvert(1-\mu_p\mu_r)P_s(z)\rvert$
2	0.266857			0.526085	0.188208
3	0.114987	0.333856		0.787074	0.067405
4	0.061619	0.161517	0.360965	1.084765	0.021554

Table 7.2. Optimization parameters for the standard Runge-Kutta formulation with correction given by taking into account the coarsening strategy. This refers to the same (1D) case treated in Table 7.1.

Table 7.2 with those where the correction suggested by the coarsening strategy is not considered (Table 7.1), then we see a limited impact in the variable coefficient case.

7.2.3 Few Observations on the Optimization Procedure

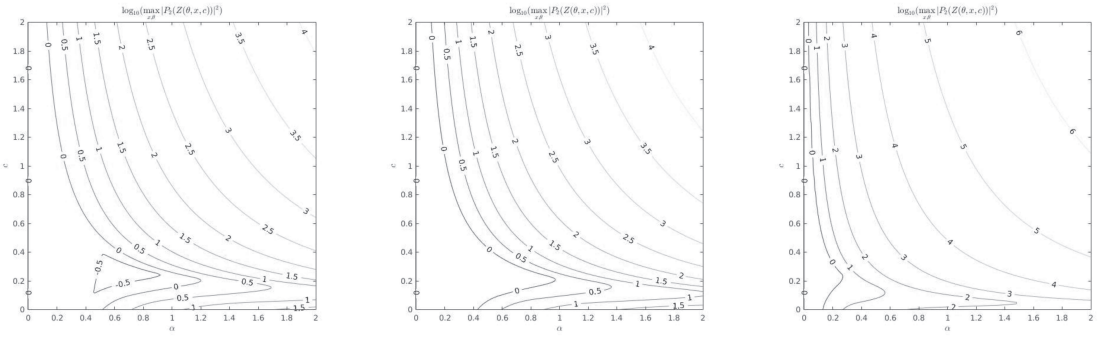
Let us focus on the optimization procedure needed to obtain the coefficients for the Runge-Kutta smoothers.

For solving the nonlinear minmax problem we use the *sequential quadratic programming method* (SQP) from [53]. This approach requires a continuous objective function, and the function (7.28) satisfies this requirement.

Let us focus on this issues looking at the plot of the level sets of the function

$$f(\alpha, c) = \log_{10}\left(\max_{(|\theta|,x)\in[\pi/2,3\pi/2]\times[x_{\min},x_{\max}]} |P_2(z(\theta, c, x; r))|^2\right),$$

for various choices of the coefficient function; see Figure 7.4. We consider only the two–stage method in standard formulation. Trying to locate



(a) $a(x) = 1 + 0.6\sin(40\pi x)$ (b) $a(x) = \frac{1}{2}x^2$ (c) $a(x) = 20x^2\exp(-x^2)$

Figure 7.4. Isoline of the function $f(\alpha, c) = \log_{10}(\max_{(\theta,x)} |P_2(z(\theta, c, x; r))|^2)$ for various $a(x)$.

the true optimum values is a problem that can be addressed in many ways. For instance we can replace the deterministic optimization with a fully probabilistic optimization or an hybrid genetic algorithm. This comes at a price: the time needed to locate the absolute minimum, if is it possible, is usually higher. To satisfy our needs we can avoid of finding the absolute minimum by accepting the first local minimum, that satisfies the stability constraint

$$|P_s(z(\theta^*, c^*, x^*; r))| < 1.$$

In this way we accept the possibility of having not the true optimal smoothing parameters, but others which still guarantee the stability of the smoothing method.

Nevertheless, if more than one dimension is considered, then the possibility of having multiple local minima increases. To face this problem, we can put together some mixed optimization strategies, i.e., use both a genetic algorithm or a particle swarm optimization, together with a strategy for finding local minima of a function. By using *Matlab's optimization toolbox*, we can try the *GA* solver or the *particleswarm* solver together with the *fmincon* algorithm. To make the optimization procedure compatible with the order of time needed for the solution of the linear systems, we should turn to more efficient and specialized implementations of these algorithms.

We conclude this subsection by comparing the optimization procedure from [45], for which the computation is performed, with the

code attached to the same article, and the procedure with the *fminimax* algorithm from *Matlab's optimization toolbox*. We consider the (1D) pure transport case with periodic boundary conditions over the space domain $\Omega = [0, 2]$ and $t \in [0, 3]$. The grid points are, respectively, $n = 2^{12} + 1$ and $2^6 + 1$. For the CFL, needed for the procedure in [45] to work, we consider using $CFL = \hat{a}\Delta t/\Delta x$ where $\hat{a}$ is either

$$\hat{a} = \max_{\Omega} a(x), \quad \hat{a} = \min_{\Omega} a(x), \quad \hat{a} = \frac{\max_{\Omega} a(x) + \min_{\Omega} a(x)}{2}. \tag{7.32}$$

The percentage ratio between our running time, and that obtained for the fixed CFL is shown in Table 7.3.

	RK3			RK4		
$a(x)$	Max	Min	Med	Max	Min	Med
$1 + 0.6\sin(4\pi x)$	1.12%	1.13%	1.12%	0.11 %	0.12%	0.12 %
$0.1 + x^2/2$	1.12%	1.22%	1.21%	0.10%	0.11%	0.11%

Table 7.3. Comparison of the optimization procedure with the case of variable and constant coefficients.

We observe that the performances of the optimization algorithm are not affected by the variation of the value of the CFL condition. This is due to the way in which the optimization in [45] is done, i.e., the values are affected only by the number of parameters. Therefore we observed that the performances of this phase do depend only on the choice of an effective optimization strategy. The convergence performance for the various constant approximations of $a(x)$ are considered in detail in the numerical experiments section.

7.2.4 Optimizing the Spectral Radius

Since the multigrid algorithm is a stationary method, its convergence rate is determined by the spectral radius of its iteration matrix. Therefore, as in [45], we may try to determine the coefficients of the Runge-Kutta smoothers to optimize this quantity. Clearly, this is a procedure that is more computationally expensive than the one needed for optimizing the smoothing of the error. In particular, we need to compute an expression for the spectral radius of the iteration matrix M, and then optimize the quantity $\rho(M)$ as a function of the parameters $\{\alpha, c\}$. Moreover, with a growing dimension of the problem at hand, i.e., for growing values of n, the number of levels of the multigrid algorithm will grow accordingly.

The optimization procedure changes also with the dimension of the system as well as for the variation of the coefficient function.

As a strategy to keep low the computational effort we can think, as in [45], to stop the refining of the matrix M at a fixed number of levels, i.e., 2 or 3, without considering that the number of actual levels is greater.

As this procedure seems less effective from the point of view of the computing times and of the requirement of a closed formula for the spectral distribution of the iteration matrix, we choose not to follow this path.

7.3 *Numerical Examples*

Here we test the methods with optimized parameters for solving the linear systems arising from the model problems in Section 7.1. To compare the results with those in [45], that treat only the 1D cases with constant coefficients using the multigrid as a solver, we focus on the same kind of framework while introducing our optimization technique on the spectral symbol for the variable coefficient case. Therefore, we check the performances of the multigrid algorithm with optimized Runge-Kutta smoother used as a solver and not as a preconditioner in the 1D case, with the various possibilities described in Section 7.2. Moreover, we test the underlying multigrid algorithm used as a preconditioner for Krylov method in the 2D formulations.

We build the multigrid algorithm with:

Pre-smoother ν sweeps of one of the Runge-Kutta algorithms;

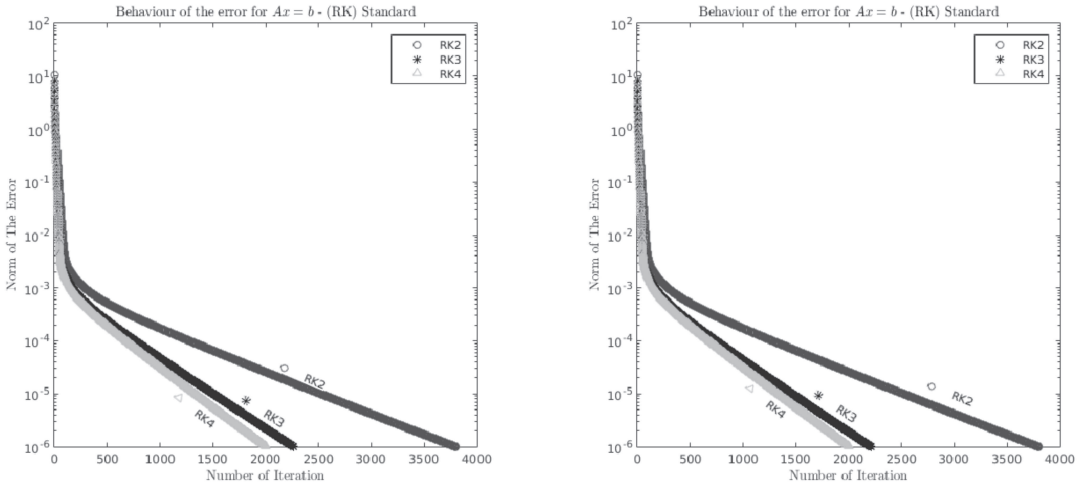
Grid-transfer operator join and split of the cells in equation (7.29). The matrix of the problem is recomputed at each level,

Post-Smoother ν sweeps with a Runge-Kutta algorithm (same as pre-smoother),

Low level solver one sweep with a Runge-Kutta algorithm (same as pre-smoother).

We start with the pure transport problem in equation (7.3), for which we set $\nu = 2$, a tolerance of $\varepsilon = 1e-6$, the spatial domain is $\Omega = [0, 2]$ with $n = 2^{11} + 1$ space steps and $m = 2^6 + 1$ time steps over the time interval $T = [0, 3]$. As a coefficient function we choose $a(x) = x^2 \exp(-x^2)$.

In Figure 7.5 we report the experiments for these settings. Note that using the optimization problem (7.31) does not alter the results with respect to the optimization problem in the form (7.28). We also see that we have a very rapid decay of the error within the first few iterations.



(a) Standard objective function (b) Weighted objective function

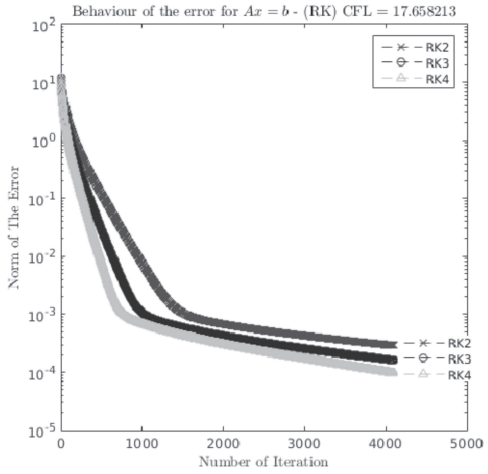
Figure 7.5. Convergence for (1D) transport with standard Runge-Kutta formulation using the standard and the weighted objective function.

This suggests trying few iterations of this algorithm as a preconditioner for a Krylov subspace method. Obviously, this can be useful only for problems in more than one dimension, because in the 1D case it is well known that banded Gaussian elimination is optimal.

Before moving further from this case we compare it with the algorithm by Birken [45]. In view of this, we need to select the optimized coefficients from that settings. We are in presence of variable coefficients and then we can choose to approximate the function $a(x)$ as in equation (7.32). We can refer either to the tabulated values of CFL in [45] or to

$$\mathrm{CFL} = \frac{\hat{a}\Delta t}{\Delta x}$$

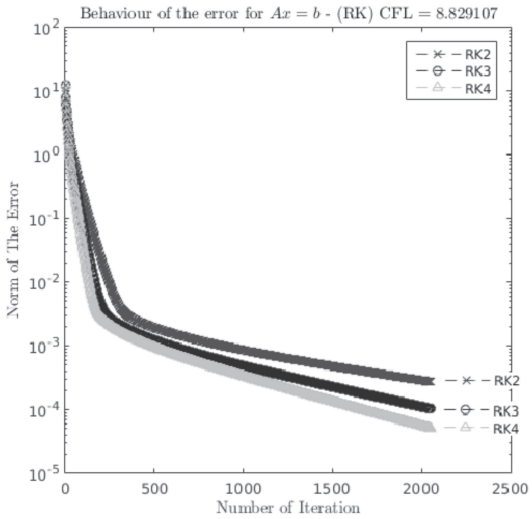
and then using the optimization algorithm from [45], i.e., for the optimized coefficients obtained for the constant $a(x) \equiv \hat{a}$. Let us consider as a first example the same settings as the previous experiment, for which we obtain a maximum CFL value of $\mathrm{CFL}_{\max} = 17.658213$, and an average CFL value of $\mathrm{CFL}_{\mathrm{avg}} = 8.829107$. We observe that for this choice of the coefficient taking the minimum has no sense. Then, by applying the optimization procedure for the constant coefficient case, we find the set of optimized coefficients reported in Table 7.6(b) alongside Figure 7.6(a). From the comparison of the Figure 7.5 with the figures 7.6(a), 7.6(c), we can observe that the performances of the algorithm optimized taking into account only the maximum of the function $a(x)$ are far worse than those which take into account the



(a) Convergence history,CFL_{max}

Order	α_1	α_2	α_3	Δt^*	$\lvert P_s(z)\rvert$
2	0.325966			0.056054	0.099252
3	0.145038	0.394377		0.083891	0.067405
4	0.080000	0.200000	0.420000	0.111600	0.004485

(b) Optimized Coefficients, CFL_{max}



(c) Convergence history, CFL_{med}

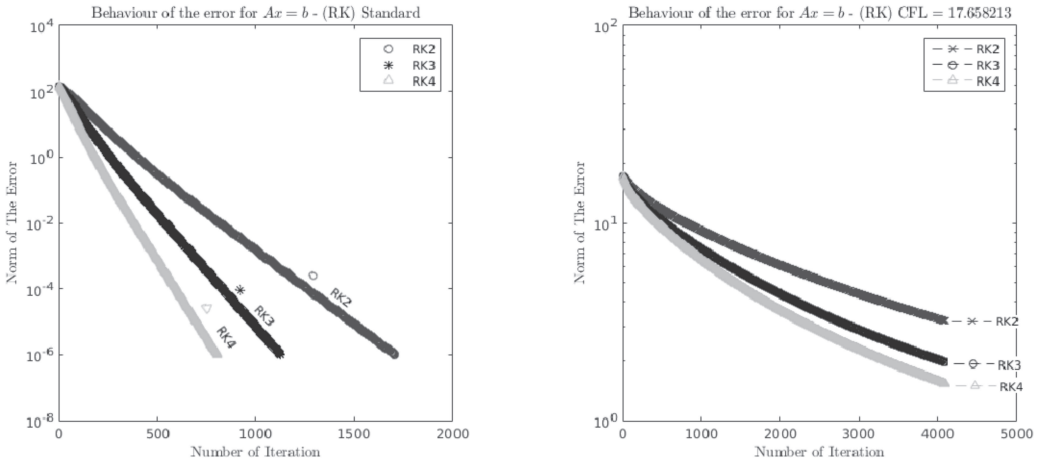
Order	α_1	α_2	α_3	Δt^*	$\lvert P_s(z)\rvert$
2	0.319437			0.110824	0.088783
3	0.145	0.39		0.1648	0.0146485
4	0.08	0.2	0.415	0.2061	0.00269064

(d) Optimized Coefficients, CFL_{med}

Figure 7.6. Multigrid algorithm in the form illustrated in [45] for the $\text{CFL}_{\text{max}} = 17.658213$ and $\text{CFL}_{\text{med}} = 8.829107$, applied to a nonconstant coefficient problem like the one in equation (7.3).

spectral information added by using GLT tools. Moreover, regarding the time needed for the optimization procedure, the strategy illustrated in Section 7.2.3, instead of the one in [45] is such that for the 3-stage scheme the time needed for our strategy is the 1.69%, while for the 4-stage the time needed is the 0.17% of the time required by the code in [45].

We can also check the application of the two strategies for a different coefficient function. To this end, we use the oscillating function $a(x) = 10(1 + 0.6\sin(50\pi x))$ keeping all the other discretization parameters the same. We find $\hat{a} = 16$ and fixed CFL condition of $CFL = 1536$.



(a) Using GLT information (b) Fixed CFL

Figure 7.7. Convergence for (1D) transport with standard Runge–Kutta formulation, objective functions for both variable and fixed coefficients.

A comparison of the two solution strategies is reported in Figure 7.7. We have already seen a gain in the speed of convergence obtained by using all the spectral information from Section 7.1, instead of using some simple estimates to drive us back to the constant coefficient case.

7.3.1 The 2D Case

We now devote our analysis to the solution of equations in 2D (7.19). By choosing $(x, y) \in [0, 2] \times [0, 2]$ and the coefficients

$$a(x) = 5(1 + 0.6\sin(10\pi x)),\ b(y) = 5(1 + 0.6\sin(10\pi y)),$$
$$\kappa_1(x) = \frac{x^2}{2},\ \kappa_2(y) = \frac{y^2}{2},$$

considering the time interval $[0, 5]$, we generate the discrete problem using a grid of 127×127 elements in space and 80 elements in time. For

advancing in time we choose the Crank-Nicolson scheme. The multigrid algorithm is as follows:

Pre-smoother $\nu = 5$ sweeps of Runge-Kutta methods with optimized coefficients;

Grid-transfer operator the operator is generated from the mask (7.30) with full–weighting, while the matrix is the discretization at each coarser grid of the operator;

Low level solver $\nu = 5$ sweeps of Runge-Kutta methods with optimized coefficients.

In this way, the algorithm mimics the flow of the solution, i.e., without a further smoothing phase after the performance of the new time step. We start testing the multigrid accelerator with the *GMRES* Algorithm stopped when the norm of the relative residual is less than $\varepsilon = 1e-6$. We use 6 multigrid iterations for each preconditioning step. The number of Runge-Kutta iterations within the multigrid accelerator is set to $\nu = 5$. We compare results with the other two formulations of the Runge-Kutta algorithm, the periodic and the nonstationary, which is used to perform only 6 iterations of the preconditioner (see (7.24)) and the formulation with matrix splitting, which is used to do only 1 iteration of the preconditioner (see (7.26)). We label the three smoothers, i.e., standard, periodic and split formulation, respectively with the labels *std*, *per* and *spl*. The results are reported in Table 7.4.

GMRES	MGM,RK2 IT	MGM,RK IT	MGM,RK4 IT	I IT
std.	39	32	31	186
per.	41	40	53	
spl.	73	77	88	

Table 7.4. Multigrid preconditioner coupled with the *GMRES* algorithm. The row labels indicate the different formulation of the Runge–Kutta smoother as in Section 7.2.1. Therefore *std.*, *per.* and *spl.* stands, respectively, for the *standard*, *periodic* and *split* formulation, while *I* stands for the unpreconditioned *GMRES*

From these results we observe that usually a lower number of parameters is preferable for the timings of both the optimization and the solution of the linear systems.

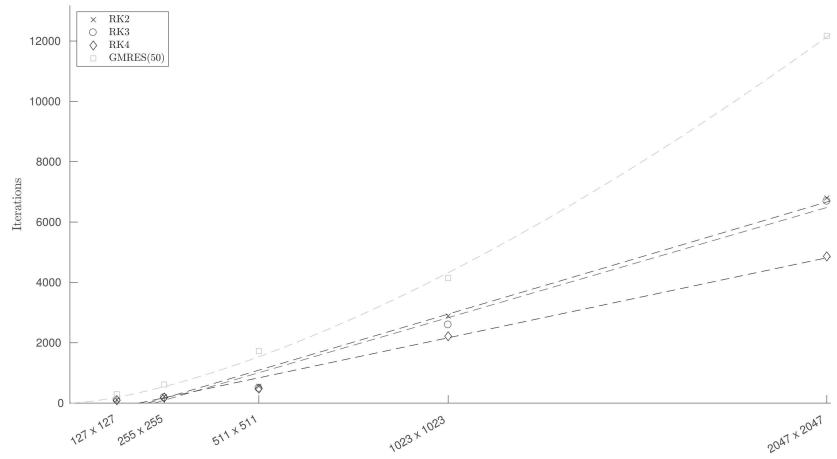
We can now consider the results of this preconditioner with a set of 2D grids of increasing size. We consider again a mixed case, i.e.,

equation (7.19) with coefficients

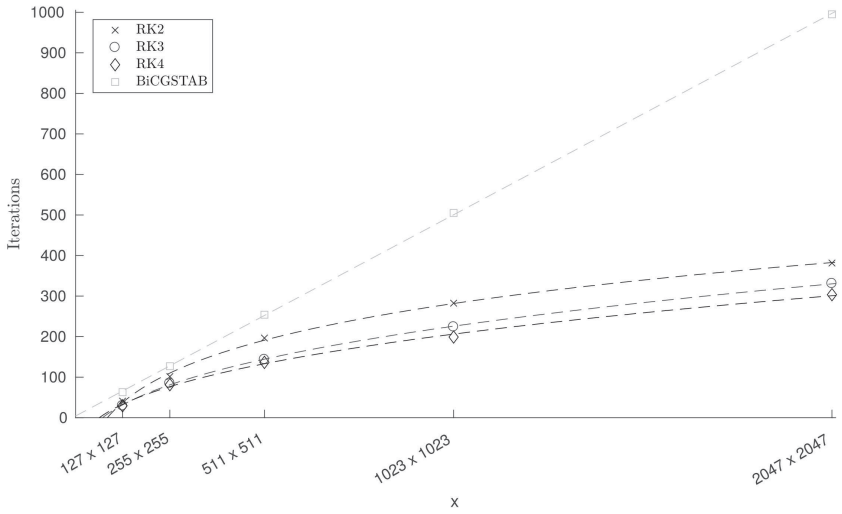
$$\begin{gathered} a(x) = \frac{1}{2}x^2,\ b(y) = \frac{1}{2}y^2, \\ \kappa_1(x) = \exp(-(x-1)^2/2),\ \kappa_2(y) = \exp(-(y-1)^2/2). \end{gathered} \tag{7.33}$$

The discretization is obtained by using the 9 point stencil for the diffusion term and the four point scheme for the transport part; see equation (7.18). We choose as Krylov accelerators the Matlab built-in implementations of *GMRES*(50) and *BiCGstab* with a tolerance of $\varepsilon = 1e-6$ and again 3 iterations of our multigrid algorithm as preconditioner with a maximum number of allowed iterations equal to the number of grid points in one direction, i.e., $MAXIT = 2^k - 1$ for the various k. The choice of the *GMRES* restarted version and of the *BiCGstab* algorithm has been made to have a feasible usage of the memory. The smoother for the preconditioner is used only in the standard formulation in order to reduce the time needed for the optimization step; see Section 7.2.3. To illustrate better the results, we collect them in Figure 7.8(a) for the *GMRES*(50) and in Figure 7.8(b) for the *BiCGstab* algorithm.

Experiments in Figures 7.8(a) and 7.8(b) confirm the analysis made in Section 7.2. The number of iterations is always less than for the unpreconditioned methods. Even if optimality is not reached, i.e., the number of iterations for convergence is not independent of the grid size, it increases slowly with respect to the schemes without preconditioning.



(a) *GMRES*(50)



(b) *BiCGstab*

Figure 7.8. Behaviour with finer and finer grids for the *GMRES*(50) and *BiCGstab* algorithms. Coefficients in equation (7.33). The size of the discretization grid is given by $(2^k - 1) \times (2^k - 1)$ over the $[0, 2]^2 \times [0, 5]$ with 80 time steps. The unpreconditioned version is used as comparison.

8

Structured Preconditioners for Fast Solution of FDEs

In [34] we considered few equations from the two classes of initial value problems (IVPs) with fractional derivatives *in space*, discussed both in Chapter 5 and Appendix A. Here we resume the underlying main results. In particular, we focus on the *fractional diffusion equation*

$$\begin{cases} \frac{\partial}{\partial t} y(x,t) = d_+(x,t)\, {}_{\mathrm{RL}}D^{\alpha}_{x_L,x}\, y(x,t) + \dots \\ \quad \dots + d_-(x,t)\, {}_{\mathrm{RL}}D^{\alpha}_{x,x_R}\, y(x,t) + g(x,t), & (x,t) \in Q, \\ y(x_L,t) = y(x_R,t) = 0, & 0 \le t \le T, \\ y(x,t_0) = y_0(x), & x \in [x_L, x_R], \end{cases} \tag{8.1}$$

for $Q = (x_L, x_R) \times (t_0, T]$, $\alpha \in (1,2)$, $f(x,t)$ the source (or sink) term and the diffusion coefficients $d_{\pm}(x,t) \ge 0$ with $d_+(x,t) + d_-(x,t) > 0$ $\forall\, x, t$, and more in general on the *fractional advection dispersion equation*

$$\begin{cases} \frac{\partial}{\partial t} y(x,t) = & d_+(x,t)\, {}_{\mathrm{RL}}D^{\alpha}_{x_L,x}\, y(x,t) + \dots \\ & \dots + d_-(x,t)\, {}_{\mathrm{RL}}D^{\alpha}_{x,x_R}\, y(x,t) + \dots \\ & \dots + b(x) y_x(x,t) + c(x) y(x,t) + g(x,t), \\ & x \in (x_L, x_R),\, t \in (t_0, T], \\ y(x_L,t) = & y(x_R,t) = 0, \quad 0 \le t \le T, \\ y(x,t_0) = & y_0(x), \quad x \in [x_L, x_R], \end{cases} \tag{8.2}$$

where $b(x) \ge 0 \in \mathscr{C}^1$ and $c(x) \ge 0 \in \mathscr{C}^0$. Similarly, one can take into account the 2D symmetric (Riesz) version of the *fractional diffusion equation*, given by

$$\begin{cases} \dfrac{\partial u}{\partial t} - K_x \dfrac{\partial^{2\alpha} u}{\partial |x|^{2\alpha}} - K_y \dfrac{\partial^{2\beta} u}{\partial |x|^{2\beta}} + \dots & (x,y) \in Q, \\ \dots + \mathbf{b} \cdot \nabla u + c u = g, & \\ u(x,y,t) = 0, & (x,y) \in \partial Q, \\ u(x,y,0) = u_0(x,y), & (x,y) \in \Omega, \end{cases} \tag{8.3}$$

where $Q = \Omega \times [0,T]$, $\mathbf{b} \in \mathscr{C}^1(\Omega, \mathbb{R}^2)$, $c \in \mathscr{C}(\Omega)$, $u \in \mathbb{L}^2(\Omega)$, $K_x, K_y \ge 0$ and $K_x + K_y > 0$, $\alpha, \beta \in (1/2, 1)$.

We consider again the Definition 5.1 for Riemann–Liouville fractional derivatives. Then to discretize equations (8.1) and (8.2), we use the p–shifted Grünwald–Letnikov discretization for the fractional Riemann–Liouville operators; see Chapter 5 and Appendix A.3.1. In a completely analogous way, we can obtain a discretization of the equation (8.3) by means of the fractional centered discretization from Appendix A.3.2 [220], which shares a similar decay property for the coefficients of the p–shifted Grünwald–Letnikov discretization, as observed in Section 5.1. By means of the above discretization, together with the centered finite difference scheme for the $b(x)u_x(x,t)$ and for the $\mathbf{b}\cdot\nabla u$, we find a semidiscretization for both equations (8.1) and (8.2) that is

$$\frac{d}{dt}\mathbf{y}(t) = J_m\mathbf{y}(t) + \mathbf{g}(t),\ t\in(t_0,T],\ \mathbf{y}(t) = [y^{(1)}(t),\dots,y^{(m)}(t)]^T. \quad (8.4)$$

The initial condition is

$$\mathbf{y}(t_0) = [y_0(x_1),\dots,y_0(x_m)]^T = \mathbf{y}_0,$$

the Jacobian matrix and forcing term are $J_m \in \mathbb{R}^{m\times m}$, $g(t)\in\mathbb{R}^m$, respectively.

Two properties of J_m are crucial for us:

- the decay along the diagonal in absolute values of its coefficients we used in Sections 5.1 and 5.1.1;
- the behavior of the eigenvalues of the matrices $\{J_m\}$, i.e., their spectral distribution.

Another ingredient we use here is ω–circulant matrices. In Section 8.2 we propose an hybrid preconditioner based on block ω–circulant matrices and on the latter property of the Jacobian matrix.

In the following we use the GLT theory (briefly described in Section 2.2.1) to determine the distribution of singular values and of the eigenvalues of the matrix sequences related to $\{J_m\}$, in a way that is similar to what we have done in Chapter 7.

Proposition 8.1 (Donatelli, Mazza, and Serra-Capizzano [100])**.** *Let us fix a time t_m and assume that the functions $d_+(x) = d_+(x,t_m)$ and $d_-(x) = d_-(x,t_m)$ are both Riemann integrable over $[x_L,x_R]$. Then, the matrix sequence $\{h^\alpha J_m\}_m$ is a GLT sequence with symbol*

$$\hat{f}(\hat{x},\theta) = f(x_L + (x_R - x_L)\hat{x},\theta),$$

where

$$f(x,\theta) = -d_+(x)e^{-\mathbf{i}\theta}(1-e^{\mathbf{i}\theta})^\alpha - d_-(x)e^{\mathbf{i}\theta}(1-e^{-\mathbf{i}\theta})^\alpha,$$

$(\hat{x},\theta) \in [0,1]\times[-\pi,\pi]$, $(x,\theta) \in [x_L, x_R]\times[-\pi,\pi]$.

In particular, since $\{h^\alpha J_m\}_m$ is a GLT sequence for f, with f from $[0,1]\times[-\pi,\pi]$, then $\{J_m\} \sim_\sigma f$, with f from $[0,1]\times[-\pi,\pi]$. Moreover, if J_m is Hermitian, then this holds also in the sense of the eigenvalues. This gives that the eigenvalues of J_m have negative real part, i.e., $\Re(\lambda_i) < 0$. Note that there is a zero of order α in 0 for the GLT symbol of $\{J_m\}$, see [100, Proposition 6]. Therefore, *any* circulant preconditioner that produces a clustering at the unity in the case of constant d_+ and d_- coefficients, is no more effective in the general variable coefficient case, for the underlying theory regarding multilevel circulant preconditioning see [219, 257, 258].

In the following Section 8.1 we recall some notions on linear multistep formulas in boundary value form. Section 8.2 includes our proposals from [34] to precondition the large, sparse and structured linear systems generated by the discretization of linear multistep formulas in boundary value form. Convergence and spectral results are provided. Finally, in Section 8.3 some numerical examples and comparisons with some of the most recent solving strategies are proposed.

8.1 *Linear Multistep Formulas in Boundary Value Form*

After discretization with respect to space variables, FDEs (8.2) and (8.1), but also a time-dependent PDE, can be reduced to the solution of the IVP

$$\begin{cases} \mathbf{y}'(t) = J_m\mathbf{y}(t) + \mathbf{g}(t), \quad t \in (t_0, T], \\ \mathbf{y}(t_0) = \mathbf{y}_0, \end{cases} \tag{8.5}$$

where $\mathbf{y}(t)$, $\mathbf{g}(t) : \mathbb{R} \to \mathbb{R}^m$, $\mathbf{y_0} \in \mathbb{R}^m$, and $J_m \in \mathbb{R}^{m\times m}$. We apply to (8.5) fully implicit methods for differential equations based on *Linear Multistep Formulas* (LMFs) in boundary value form; see [9, 66] and references therein. These methods approximate the solution of the IVP (8.5) by means of a discrete boundary value problem. Consider the application to (8.5) of the following μ-step LMF over a uniform mesh $t_j = t_0 + jh$, for $j = 0,\dots,s$, $h = (T-t_0)/s$, namely

$$\sum_{i=-\nu}^{\mu-\nu} \alpha_{i+\nu}\mathbf{y}_{n+i} = h\sum_{i=-\nu}^{\mu-\nu} \beta_{i+\nu}\mathbf{f}_{n+i}, \quad n = \nu,\dots,s-\mu+\nu. \tag{8.6}$$

Here, $\mathbf{y}_n$ is the discrete approximation to $\mathbf{y}(t_n)$, $\mathbf{f}_n = J_m\mathbf{y}_n + \mathbf{g}_n$ and $\mathbf{g}_n = \mathbf{g}(t_n)$. The method in (8.6) should be used with ν initial conditions and $\mu - \nu$ final conditions. Therefore, we need the values $\mathbf{y}_0, \ldots, \mathbf{y}_{\nu-1}$ and the values $\mathbf{y}_{s-\mu+\nu+1}, \ldots, \mathbf{y}_s$. The initial condition in (8.5) provides only one value, i.e., $\mathbf{y}_0$. In order to get the other initial and final values, we have to provide additional $(\mu-1)$ equations. The coefficients $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ of these equations can be chosen such that the truncation errors for these initial and final conditions are of the same order as the one in (8.6); see [66, p. 132] for details. We stress that all the methods considered here are consistent, i.e., their characteristic polynomials

$$\rho(z) = z^\nu \sum_{j-\nu}^{k-\nu} \alpha_{j+\nu} z^j, \qquad \sigma(z) = z^\nu \sum_{j=-\nu}^{k-\nu} \beta_{j+\nu} z^j,$$

are such that

$$\rho(1) = 0, \qquad \rho'(1) = \sigma(1).$$

By combining (8.6) with the above mentioned additional methods, we obtain a discrete *Boundary Value Problem* (BVM); see [66]. These equations can be restated to give the following linear system of algebraic equations

$$M\mathbf{y} \equiv (A \otimes I_m - hB \otimes J_m)\mathbf{y} = \mathbf{e}_1 \otimes \mathbf{y}_0 + h(B \otimes I_m)\mathbf{g} \equiv \mathbf{b}, \tag{8.7}$$

where

$$\mathbf{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{s+1}, \ \mathbf{y} = [\mathbf{y}_0^T, \ldots, \mathbf{y}_s^T]^T \in \mathbb{R}^{(s+1)m},$$
$$\mathbf{g} = [\mathbf{g}_0^T, \ldots, \mathbf{g}_s^T]^T \in \mathbb{R}^{(s+1)m}, \ A, B \in \mathbb{R}^{(s+1)\times(s+1)}.$$

The matrices A and B are obtained from the coefficients of the formula (8.6) and from the coefficients of the auxiliary linear multistep formulas as

$$A = \begin{bmatrix}
1 & \ldots & 0 & & & & \\
\alpha_0^{(1)} & \ldots & \alpha_k^{(1)} & & & & \\
\vdots & & \vdots & & & & \\
\alpha_0^{(\nu-1)} & \ldots & \alpha_k^{(\nu-1)} & & & & \\
\alpha_0 & \ldots & \alpha_k & & & & \\
 & \alpha_0 & \ldots & \alpha_k & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \alpha_0 & \ldots & \alpha_k & \\
 & & & \alpha_0^{(s-k+\nu+1)} & \ldots & \alpha_k^{(s-k+\nu+1)} & \\
 & & & \vdots & & \vdots & \\
 & & & \alpha_0^{(s)} & \ldots & \alpha_k^{(s)}
\end{bmatrix},$$

and

$$B = \begin{bmatrix} 0 & \dots & 0 & & & & \\ \beta_0^{(1)} & \dots & \beta_k^{(1)}, & & & & \\ \vdots & & \vdots & & & & \\ \beta_0^{(\nu-1)} & \dots & \beta_k^{(\nu-1)} & & & & \\ \beta_0 & \dots & \beta_k & & & & \\ & \beta_0 & \dots & \beta_k & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \beta_0 & \dots & \beta_k & \\ & & & \beta_0^{(s-k+\nu+1)} & \dots & \beta_k^{(s-k+\nu+1)}, & \\ & & & \vdots & & \vdots & \\ & & & \beta_0^{(s)} & \dots & \beta_k^{(s)} & \end{bmatrix}.$$

We recall that auxiliary methods cannot have the same coefficients as (8.6). Further details on the matrices A and B, M and their entries can be found in [66, p. 132]. Some properties and information on their eigenvalues can be found in [28] and in [30].

The size of the matrix M can be very large when s or m are large. If a direct method is used to solve the system (8.7), e.g., for a multi-dimensional FDE, then the operation count can be much higher, see also the comparisons sparse direct/iterative methods for a PDEs in [28]. Therefore, we concentrate again on Kyrlov iterative solvers. Note that in general it is not necessary to assemble explicitly the matrix M from equation (8.7), since to apply Krylov iterative solvers we need only to form the matrix vector products $M\mathbf{y}$. Thus, by the properties of Kronecker products, we have

$$\mathbf{x} = M\mathbf{y} = \mathrm{vec}(I_m Y A^T - h J_m Y B^T) = \mathrm{vec}(Y A^T - h J_m Y B^T),$$

where the operator $\mathrm{vec}(\cdot)$ stacks the columns of a matrix and Y is obtained by simply reshaping $\mathbf{y}$ as an $m \times s$ matrix.

Differently to PDEs discretized by finite differences or using finite elements, in case of FDEs, also Krylov iterative solvers with the block circulant preconditioners introduced in [28] can be not so effective because J_m is a dense matrix. The same conclusions are derived for all (block or not) preconditioners for the linear systems of other time-step integrators based, e.g., on linear multistep formulas or on Runge-Kutta methods; see [28].

The discretizations considered here for fractional differential equations produce Jacobian matrices J_m whose eigenvalues have negative real part. Therefore, it is natural to use the L–stable generalization of

BDF formulas proposed in [66], instead of the generalization of Adams-Moulton formulas used in [143]. The GBDF formula for a problem of the form (8.5) with k steps are obtained from the expression of the classical BDF formulas

$$\sum_{i=0}^{k} \alpha_i \mathbf{y}_{n+i} = h\beta_k \mathbf{f}_{n+k}. \tag{8.8}$$

Note that for (8.8) the second stability polynomial is $\sigma(z) = \beta_k z^k$. It is well known that the BDF formulas from order 7 onwards are 0–unstable and they are also not A–stable for any $k > 2$. On the other hand, if we use the underlying generalization of linear multistep formulas, we can build methods of both maximal order k and with potentially better stability properties. Specifically, we obtain formulas that are both $0_{\nu,k-\nu}$–stable and $A_{\nu,k-\nu}$–stable for all $k \geq 1$ by selecting the second stability polynomial $\sigma(z) = \beta_j z^j$ with $j = \nu$ and

$$\nu = \begin{cases} {}^{k+1}/_{2}, & k \text{ odd}, \\ {}^{k}/_{2} + 1, & k \text{ even}, \end{cases}$$

instead of the classical choice $j = k$. The stability region of these methods lies outside the curve

$$\Gamma_k = \{q \in \mathbb{C} : |\pi(z, q)| \equiv |\rho(z) - qz^{\nu}| = 1, \forall z \in \mathbb{C}\}.$$

Thus, by normalizing the coefficients, the GBDF with k steps, ν initial and $k - \nu$ final conditions can be written as

$$\sum_{i=-\nu}^{k-\nu} \alpha_{i+\nu} \mathbf{y}_{i+\nu} = h\mathbf{f}_n, \quad n = \nu, \ldots, s - k + \nu. \tag{8.9}$$

The latter is clearly an instance of the general formula in (8.6); see again [65] for the complete derivation. We stress that also $L_{\nu,k-\nu}$–stability matters in this case. Indeed the use of L–stable (and thus $L_{\nu,k-\nu}$–stable) methods permit to use sensibly larger time steps without compromising the qualitative behavior of the approximation when rapid decaying transients occurs in the solution. Here we use low order linear multistep formulas (maximum order 3) because the discretization of the fractional differential operator shows a order at most linear in our equispaced mesh. Note also that a higher order formula requires a higher computational effort to solve the related linear systems; see next sections. Moreover, a higher order formula (in both time and space)

requires a higher regularity of the solution, that is not guaranteed to hold for a fractional equation, even when the coefficients of the underlying FPDE are arbitrarily regular.

Theorem 8.1 (Brugnano and Trigiante [65])**.** *In exact arithmetic, a BVM with* $(\nu, k-\nu)$*–boundary conditions is convergent if it is consistent and* $0_{\nu,k-\nu}$*–stable.*

Therefore, we state our main convergence result as follows.

Proposition 8.2. *The GBDF formula* (8.9) *with* $k = 2$ *applied to Problem* (8.1) *discretized by the* 1*–shifted Grünwald–Letnikov formulas is convergent whenever* $y \in \mathscr{C}^{\alpha+1}$*.*

Proof. We wish to apply Theorem 8.1. Therefore, we only need to prove that the resulting method is consistent, since, as we have seen, GBDF formulas are $0_{\nu,k-\nu}$–stable. Let $u(x,t)$ be the true solution of (8.1). Then, the local truncation error $\tau(x,t)$ is consistent of order $O(h^2 + \Delta x)$ by [201, Theorem 2.7]. Similar arguments can be used in several spatial dimensions. □

In our opinion, using a discretization in time of order five as in [143] is expensive and the global accuracy does not increase in general: the low order approximation in space dominates the global error.

8.2 *Structured Preconditioners*

To solve linear systems (8.7), let us focus on the application of efficient iterative Krylov methods from Section 2.1, namely the *BiCGstab*(2) [284], *GMRES*(20) and *FGMRES* [245], coupled with block preconditioners taking into account their block structure. In the style of [28, 29, 40], we propose a preconditioner of the form

$$P = \breve{A} \otimes I - h\breve{B} \otimes \tilde{J}_m, \tag{8.10}$$

where $\breve{A}$ and $\breve{B}$ are circulant–like approximations of the Toeplitz matrices A and B, respectively, containing the coefficients of the LMF formulas (8.6) and of the additional LMFs, while $\tilde{J}_m$ is a suitable approximation of the Jacobian matrix detailed below.

By the properties of the Kronecker product, the eigenvalues of the preconditioner P are given by

$$\phi_i - h\psi_i\lambda_j, \qquad i = 1,\dots,s, \quad j = 1,\dots,m,$$

where $\{\phi_i\}$ and $\{\psi_i\}$ are the eigenvalues of the circulant–like approximations $\breve{A}$ and $\breve{B}$, respectively, and $\{\lambda_j\}$ are the eigenvalues of the selected approximation of J_m.

In [143] the authors proposed the following block–preconditioner based on the Strang circulant approximation (see [211] for further details) for the FDEs semidiscretized in space with the p–shifted Grünwald–Letnikov formula

$$P_{\mathfrak{s}} = \mathfrak{s}(A) \otimes I_m - h\mathfrak{s}(B) \otimes J_m, \tag{8.11}$$

where

$$\mathfrak{s}(A) = \begin{bmatrix} \alpha_\nu & \cdots & \alpha_\mu & & & & \alpha_0 & \cdots & \alpha_{\nu-1} \\ \vdots & \ddots & & \ddots & & & & \ddots & \vdots \\ \alpha_0 & & \ddots & & \ddots & & & & \alpha_0 \\ & \ddots & & \ddots & & \ddots & & 0 & \\ & & \ddots & & \ddots & & \ddots & & \\ & 0 & & \ddots & & \ddots & & \ddots & \\ & & & & \ddots & & \ddots & & \alpha_\mu \\ \alpha_\mu & & & & & \ddots & & \ddots & \vdots \\ \vdots & \ddots & & & & & \ddots & & \\ \alpha_{\nu+1} & \cdots & \alpha_\mu & & & & \alpha_0 & \cdots & \alpha_\nu \end{bmatrix},$$

and $s(B)$ can be defined similarly. The preconditioner (8.11) for a generic LMF in boundary value form was first introduced in 1998 in [27] and in [28] using also other circulant approximations, and later studied also in in [77]. In particular, (8.11) was introduced for LMF in boundary value form for solving a generic differential problem and thus also for an initial value problem generated by semidiscretization in space of the underlying FDE problem. In this framework, the following preconditioner, based on the modified Strang circulant introduced in [29] can be a better approach

$$P_{\tilde{\mathfrak{s}}} = \tilde{\mathfrak{s}}(A) \otimes I_m - h\tilde{\mathfrak{s}}(B) \otimes J_m. \tag{8.12}$$

The above, discussed in [29], is able to treat problems with severe ill-conditioning or also singularity of the block preconditioners based on Strang circulant approximation of a LMF. In particular $\tilde{\mathfrak{s}}(\cdot)$ is obtained simply as a rank–one correction of the natural Strang preconditioner $\mathfrak{s}(\cdot)$, i.e., $\tilde{\mathfrak{s}}(A) = \mathfrak{s}(A) + E$ where E is a rank–one circulant matrix given by

$$E = F^H \operatorname{diag}(\hat{\phi}_0\text{-}\phi_0, 0, \ldots, 0)\, F,$$

with $\hat{\phi}_0$ that, as suggested in [29], can be $\hat{\phi}_0 = 1/s+1$ or $\hat{\phi}_0 = \mathfrak{R}(\phi_s)$; see [27, 29, 30]. Surprisingly, none of the above mentioned researches on block circulant preconditioners for LMF in boundary value form has been mentioned in Gu et al. [143].

Differently from PDEs, J_m can be a dense matrix for FDEs. Hence, in order to reduce the computational complexity, the following two block–circulant with circulant blocks versions

$$P'_{\mathfrak{s}} = \mathfrak{s}(A) \otimes I_m - h\mathfrak{s}(B) \otimes \mathfrak{s}(J_m), \tag{8.13}$$

and

$$P'_{\tilde{\mathfrak{s}}} = \tilde{\mathfrak{s}}(A) \otimes I_m - h\tilde{\mathfrak{s}}(B) \otimes \tilde{\mathfrak{s}}(J_m), \tag{8.14}$$

based on the application of the same circulant preconditioner to the Jacobian matrix were also considered in [143].

The eigenvalues of the circulant approximation $\mathfrak{s}(\cdot)$ and $\tilde{\mathfrak{s}}(\cdot)$ can be read on the main diagonal of the matrix Λ; applying Theorem 2.12 with $\omega = 1$ and thus $\Omega = I$ is sufficient.

Remark 8.1. *We do not recommend the choice of the Strang circulant approximation for the Jacobian matrix J_m or for A in* (8.7)*. As clearly remarked in [27] and in [29], the Strang approximation for the matrix A in equation* (8.7) *is singular for every number of step $k \geq 1$, independently from the value of s, simply by the consistency requirements of the linear multistep formulas. Moreover, given the analysis of the spectral distribution of the matrix sequence $\{h^\alpha J_m\}_m$ we discussed at the beginning of the chapter, we do not recommend the use of the Strang preconditioner for the Jacobian matrix J_m as well.*

To overcome these serious issues of the Strang circulant approximations, in Gu, Huang, Zhao, Li, and Li [143] the strategy introduced by Bertaccini [29] for PDEs was used. The latter relies in shifting away from zero the eigenvalue of smallest modulus.

Here we focus on other preconditioners that do not need the correction mentioned in Remark 8.1. In particular, we consider the ω–circulant approximation $\omega(\cdot)$ introduced for LMF in boundary value form to integrate PDEs in [38, 40]

$$P_\omega = \omega(A) \otimes I_m - h\omega(B) \otimes J_m, \tag{8.15}$$

where

$$\omega(A) = \begin{bmatrix} \alpha_\nu & \cdots & \alpha_\mu & & & \omega\alpha_0 & \cdots & \omega\alpha_{\nu-1} \\ \vdots & \ddots & & \ddots & & & \ddots & \vdots \\ \alpha_0 & & \ddots & & \ddots & & & \omega\alpha_0 \\ & \ddots & & \ddots & & \ddots & 0 & \\ & & \ddots & & \ddots & & & \\ & 0 & & \ddots & & \ddots & & \\ \omega\alpha_\mu & & & & \ddots & & & \alpha_\mu \\ \vdots & \ddots & & & & \ddots & \ddots & \vdots \\ \omega\alpha_{\nu+1} & \cdots & \omega\alpha_\mu & & & \alpha_0 & \cdots & \alpha_\nu \end{bmatrix},$$

and $\omega(B)$ is defined similarly.

Since for FPDEs J_m is a dense matrix, for reducing the computational cost of matrix–vector multiplications using preconditioner (8.15) we firstly propose to use $\omega(J_m)$ instead of J_m in (8.15), i.e., an ω–circulant approximation also for J_m that is

$$P'_\omega = \omega(A) \otimes I_m - h\omega(B) \otimes \omega(J_m).$$

Our second proposal is based on exploiting the short–memory principle from Section 5.1.1. This means using a banded approximation of the Jacobian matrix J_m instead of a circulant or an ω–circulant one (for J_m). We apply the short–memory principle by the function $g_k(J_m) = [J_m]_k$ that just extracts the k lower and upper main diagonals of J_m producing the following *limited memory block ω-circulant preconditioner*

$$P_{\omega,k} = \omega(A) \otimes I_m - h\omega(B) \otimes g_k(J_m). \tag{8.16}$$

To further reduce the computational effort needed to apply $P_{\omega,k}$–circulant preconditioner at each iteration, instead of a direct method for sparse systems, we can consider the use of nested iterative methods, e.g., the *GMRES*(m) method. In particular, to solve the s auxiliary linear systems of the form

$$T_{j,k} \triangleq \phi_j I - h\psi_j\, g_k(J_m), \quad j = 1, \ldots, s,$$

required to apply all block circulant or block $P_{\omega,k}$–circulant preconditioners described above; see, e.g., [28] and [40] for technical details. In this way we are moving into the framework of preconditioners changing

during the iterations and then we need to use *Flexible GMRES* method or its restarted version; see Saad in [244, 245] and Section 2.1.3. In some cases, to ensure a fast convergence of the outer method (*FGMRES*), we need to use a preconditioner for the inner method (*GMRES*(m)). To this end, we propose the use of an approximate inverse Toeplitz preconditioner for $T_{j,k}$ based on the ω–circulant preconditioner from [116, 150]. Thus, we consider the ω–circulant extension $W_{j,n+k}$ of $T_{j,k}$, obtained as

$$W_{j,n+k} = \begin{bmatrix} \tilde{T}_{j,k} & T_{2,1}^H \\ T_{2,1} & T_{2,2} \end{bmatrix},$$

$$\text{with } T_{2,1} = \begin{bmatrix} \omega t_k & & & 0 & \dots & 0 & \overline{t_k} & \dots & \overline{t}_1 \\ \vdots & \ddots & & \vdots & \ddots & \vdots & & \ddots & \vdots \\ \omega t_1 & \dots & \omega t_k & 0 & \dots & 0 & & & \overline{t_k} \end{bmatrix},$$

where $\tilde{T}_{j,k}$ is the Toeplitz matrix obtained with the first column and row of $T_{j,k}$ and $\omega = \exp(i\theta)$ with $\theta \in [-\pi, \pi]$. In this way, the diagonal matrix $\Lambda_{j,n+k}$ containing the eigenvalues of $W_{j,n+k}$ is given by

$$\Lambda_{j,n+k} = F_{n+k}\Omega_{n+k}W_{j,n+k}\Omega_{n+k}^H F_{n+k}^H.$$

Once the eigenvalues have been computed, the inverse of the ω–circulant matrix is

$$W_{j,n+k}^{-1} = \begin{bmatrix} P & P_{1,2} \\ P_{1,2} & P_{2,2} \end{bmatrix} = \Omega_{n+k}^H F_{n+k}^H \Lambda_{j,n+k}^{-1} F_{n+k}\Omega_{n+k}, \tag{8.17}$$

taking care of the positions of the non–positive entries of $\Lambda_{j,n+k}$ and putting a zero in the corresponding positions of $\Lambda_{j,n+k}^{-1}$. Then, the preconditioner $P = \tilde{T}_{j,k}^{-1}$ in (8.17) is used for the inner *GMRES*(m) method.

Lemma 8.1 (Bertaccini and Durastante [34])**.** *Let us consider the approximation $g_k(J_m)$ for J_m. Then, for $\varepsilon > 0$ and $m > 0$ integer, there exists a bandwidth parameter $\tilde{k} = \tilde{k}(\varepsilon, m, \alpha) > 0$ such that $g_k(J_m)^{-1}J_m = I + N$ with $\|N\| \leq \varepsilon \ \forall k \geq \tilde{k}$.*

Proof. Fix $\varepsilon > 0$ and assume that $y(x)$ is such that $y(x) \leq M$ for $x \in \Omega = [x_L, x_R]$. Then, for each $L \in \Omega$ we can write the error as (see Section 5.1.1),

$$E(x) = |\,{}_{\mathrm{RL}}D_{a,x}^{\alpha} y(x) - {}_{\mathrm{RL}}D_{x-L,x}^{\alpha} y(x)| \leq \frac{ML^{-\alpha}}{|\Gamma(1-\alpha)|}.$$

We can find the required values of L by solving

$$|E(x)| \leq \varepsilon, \ (x_L + L \leq x \leq x_R), \ \Rightarrow \ L \geq \left(\frac{M}{\varepsilon |\Gamma(1-\alpha)|} \right)^{1/\alpha}.$$

Therefore, after repeating exactly the same argument for the other side fractional derivative, fixed a discretization step and a value of ε, we can choose a bandwidth k giving the wanted residual and such that its norm is less the ε. Otherwise, we can look at it from the spectral point of view. From Proposition 8.1 we find that $\{J_m\} \sim_{\text{GLT}} f$, with f defined on $[0,1] \times [-\pi, \pi]$), thus we can consider, at the same way, the spectral distribution $f_{\tilde{k}}$ of $g_{\tilde{k}}(J_m)$). This is obtained by replacing $e^{-\mathbf{i}\theta}(1-e^{\mathbf{i}\theta})^{\alpha}$ and $e^{\mathbf{i}\theta}(1-e^{\mathbf{i}\theta})^{-\alpha}$ in f by the first $\tilde{k}$ term of their real binomial expansion. Thus, $g_{\tilde{k}}(J_m)^{-1} J_m \sim_{\text{GLT}} f/f_{\tilde{k}}$, but this can be expressed as

$$\frac{f}{f_{\tilde{k}}} = 1 + n_{\tilde{k}},$$

where $n_{\tilde{k}}$ is again the function f in which we have replaced $e^{-\mathbf{i}\theta}(1-e^{\mathbf{i}\theta})^{\alpha}$ and $e^{\mathbf{i}\theta}(1-e^{\mathbf{i}\theta})^{-\alpha}$ by the first $m-\tilde{k}$ term of their real binomial expansion. We conclude referring to the decay property in (5.15) and recalling that the coefficients of $n_{\tilde{k}}$ are exactly the $\omega_j^{(\alpha)}$ for $j > \tilde{k}$, thus finding the minimum integer $\tilde{k}$ such that the bound $\|n_{\tilde{k}}\| < \varepsilon$ holds. Therefore, we immediately get also $\|n_k\| < \varepsilon \ \forall k \geq \tilde{k}$. □

Remark 8.2. *Observe that Lemma 8.1 is quite independent from the discretization adopted, since the first way of proving it depends only a structural properties of the fractional operators, namely the* short–memory principle, *which, as extensively discussed in Section 5.1.1, is inherited by different discretizations of the operators. Therefore, with a little additional effort, also the spectral part of the proof can be extended to other discretizations. A depiction of the result of Lemma 8.1 is given in Figure 8.1. We stress again that this property is lost if we use* any *circulant approximation for a variable coefficient case in both Problem* (8.1) *and* (8.2)*. See the discussion at the end of Chapter 8.*

Theorem 8.2 (Bertaccini and Durastante [34])**.** *Let us consider the limited memory block ω-circulant preconditioner* (8.16) *such that $\omega = \exp(i\omega\theta)$, $\theta = \pi$ and $k \geq \tilde{k}$, $\tilde{k}$ as in Lemma 8.1. Then, the eigenvalues of the preconditioned matrix $P_{\omega,k}^{-1} M$ are equal to $1 \in \mathbb{C}$ except for at most $2m\mu$ outliers.*

Proof. The claimed thesis follows by applying Lemma 8.1 and by the

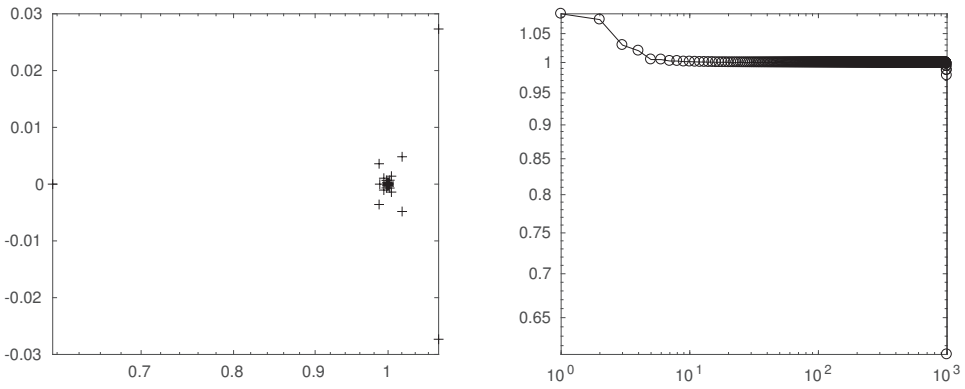


Figure 8.1. Lemma 8.1. Clustering on the eigenvalues (on the left) and on the singular values (on the right) for the preconditioner $g_k(J_m)^{-1}J_m$ for J_m from Problem (8.1) and $k = \lceil m/5 \rceil$.

same argument of [40, Theorem 4]. Let $E = M - P_{\omega,k}$, then

$$\begin{aligned} E &= ((A - \omega(A)) \otimes I_m) - h(B - \omega(B)) \otimes J_m - h\omega(B) \otimes (J_m - g_k(J_m)) \\ &= L_A \otimes I_m - hL_B \otimes J_m - h\omega(B) \otimes N. \end{aligned}$$

It is easy to check that both L_A and L_B are $(s+1)\times(s+1)$ matrices with nonzero entries at most in the following four corners: a $\nu \times (\mu+1)$ block in the upper left; a $\nu \times \nu$ block in the upper right; a $(\mu-\nu)\times(\mu+1)$ block in the lower right; and a $(\mu-\nu)\times(\mu-\nu)$ block in the lower left. Since $\mu > \nu$, $\operatorname{rank}(L_A) \leq \mu$ and $\operatorname{rank}(L_B) \leq \mu$. Then, by Lemma 8.1 we have that exists $k \geq \tilde{k}$ such that $\|N\| < \varepsilon$, thus we have that E is decomposed into a matrix of rank $2m\mu$ plus a matrix with norm lower than ε, this concludes the proof. □

As a direct consequence, we have the convergence result.

Corollary 8.1. *If the matrix $P_{\omega,k}^{-1} M$ is diagonalizable,* GMRES *converges in at most $2m\mu + 1$ iterations, independently of s, where μ is the number of the steps of the LMF formula.*

Proof. This is an immediate consequence of Theorem 8.2 and Theorem 2.7. □

We stress that the above result shows a number of iteration proportional to m. However, in practice, we experience convergence of iterations for *GMRES*, *GMRES*(r), $r > 1$, and BiCGstab, preconditioned by the *limited memory block ω–circulant preconditioner*, much less dependent on the mesh than what Corollary 8.1 suggests; see the tables in Section 8.3. On the other hand, if we choose, e.g., $k = \lceil m/5 \rceil$ for $P_{\omega,k}$

in (8.16), then the underlying Krylov iterative solvers converge in a number of iterations more or less constant with the mesh parameters. Unfortunately, by taking $k = \lceil m/5 \rceil$ we keep (almost) constant the iterations, but we increase the computational cost with m, suggesting that a choice of a constant k, can be a good (but of course somewhat problem–dependent) compromise.

Convergence results similar to Theorem 8.2 can be derived for other values of θ different from π. However, as observed in [40, Section 2.2] and confirmed by our numerical experiments, the $\{\omega\}$–circulant block preconditioners which give slightly "best" results have $\omega = -1$, i.e., $\theta = \pi$, and then are based on skew–circulant matrices.

8.3 *Numerical Examples*

We summarize in Table 8.1 the preconditioning strategies that are tested in our experiments.

The results have been obtained on a laptop running Linux with 8Gb memory and CPU Intel® Core™ i7–4710HQ CPU with clock 2.50 GHz and Matlab version R2016b.

We use here our implementation of *FGMRES*, based on the algorithms and suggestions discussed in Section 2.1.3 from [245], while we use *GMRES*(20) and BiCGstab provided by Matlab. BiCGstab(2) is implemented as in Section 2.1.4 so similarly to [284]. We report the number of matrix–vector operations performed by the solvers in the tables. Moreover, the main stopping criteria require the relative residuals less than $\varepsilon = 10^{-8}$. Here all the $\{\omega\}$-circulant approximations have $\omega = -1$, i.e., $\theta = \pi$, and then they are based on skew-circulant matrices. Motivations for this choice are detailed in [40, Section 2.2].

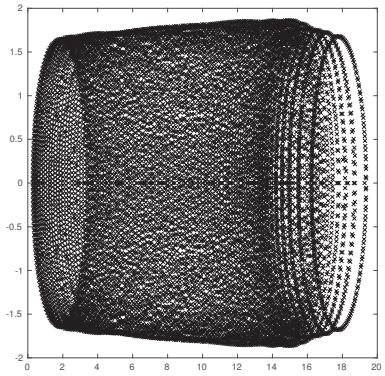
Experiment 1. As a first test case we consider the fractional diffusion equation (8.1) with coefficients

$$
\begin{aligned}
x_L = 0,\ x_R = 2,\ t_0 = 0,\ T = 1,\\
g(x,t) = -32e^{-t}\left\{x^2 + \frac{1}{8}(2-x)^2(8+x^2) + \dots\right.\\
\dots - \frac{3}{3-\alpha}\left[x^3 + (2-x)^3\right] + \dots\\
\left.\dots + \frac{3}{(4-\alpha)(3-\alpha)}\left[x^4 + (2-x)^4\right]\right\},\\
d_+(x,t) = \Gamma(3-\alpha)x^\alpha,\ d_-(x,t) = \Gamma(3-\alpha)(2-x)^\alpha,\\
u_0(x) = 4x^2(2-x)^2,
\end{aligned}
\tag{8.18}
$$

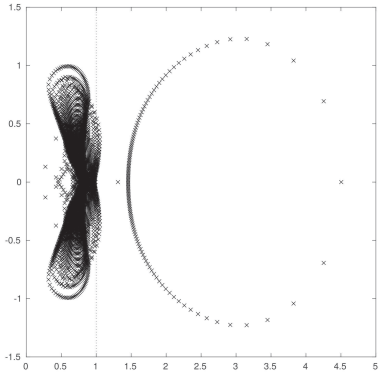
	Preconditioner for:			Computational
	A	B	J_m	cost
I	None	None	None	–
$P_{\mathfrak{s}}$	Strang	Strang	None	$O(ms\log(s)+sm^2)$
$P_{\tilde{\mathfrak{s}}}$	Modified Strang	Modified Strang	None	$O(ms\log(s)+sm^2)$
$P'_{\mathfrak{s}}$	Strang	Strang	Strang	$O(ms\log(ms))$
P_{ω}	ω–Circulant	ω–Circulant	None	$O(ms\log(s)+sm^2)$
P'_{ω}	ω–Circulant	ω–Circulant	ω–Circulant	$O(ms\log(ms))$
$P_{\omega,k}$	ω–Circulant	ω–Circulant	$g_k(J_m)$	$O(ms\log(s)+sk^2m)$
$P^{\text{FGMRES}}_{\omega,k}$	ω–Circulant	ω–Circulant	$g_k(J_m)$ *GMRES*	$O(ms\log(s)+s(2k-1)m)$

Table 8.1. Preconditioners tested in the numerical examples, details in Section 8.2.

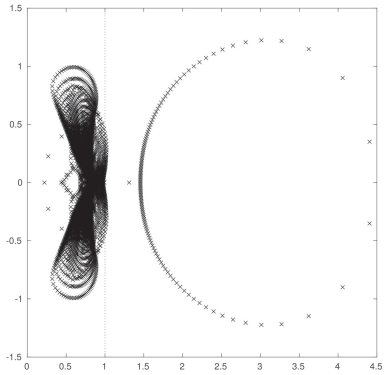
where the order of the fractional derivatives is $\alpha = 1.5$ and $\alpha = 1.8$, respectively. For this choice of the coefficients we have the exact solution of the FPDEs that is $u_e(x,t) = 4e^{-t^2}x^2(2-x)^2$ for any value of $\alpha \in (1,2)$. In Table 8.2 we show the results obtained with various preconditioning strategies. For this case, we use the GBDF formula with two step, that gives a more reasonable behavior of the error when mixed with the first order approximation used for the discretization in space, with *GMRES*(20), *FGMRES* and *BiCGstab*(2) iterative methods. Moreover, in Figure 8.2, we give both the spectrum of the unpreconditioned matrix M and of the preconditioned matrix M for all the proposed preconditioners. Consistently with the results in Table 8.2 and the



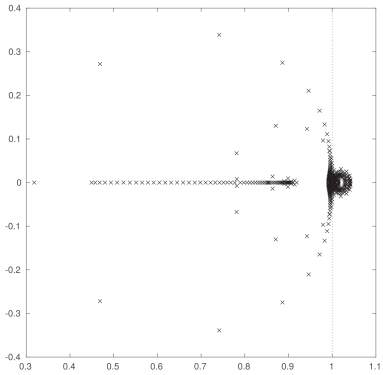
(a) Eigenvalues of the block matrix M in (8.7)



(b) Eigenvalues of $(P'_s)^{-1}M$



(c) Eigenvalues of $(P'_\omega)^{-1}M$



(d) Eigenvalues of $(P_{\omega,\lceil m/10\rceil})^{-1}M$

Figure 8.2. Experiment 1. Spectra of both the matrix of the system and of the preconditioned matrices with $\alpha = 2$ and 2 step GBDF formula with $m = 97$ and $s = 128$.

analysis in Section 8.2, the preconditioner based on the short–memory principle achieves the better clustering among the others. Observe also that, with *BiCGstab*(2), timings are greater than those obtained with *FGMRES*, even if the limited memory preconditioners $P_{\omega,\lceil m/10 \rceil}$ are always better than their competitors. We have omitted the numerical results for this case.

Experiment 2. We consider the fractional partial differential equation in two dimensions in (8.3) with the following choice of the coefficients

$$\begin{gathered} K_x = 2,\ K_y = 1.5,\ c(x,y) = 1 + 0.5\cos(xy), \\ \mathbf{b} = (\beta + 0.5\sin(4\pi x)\cos(5\pi y), \dots \\ \dots \alpha + 0.7\sin(7\pi y)\cos(4\pi x)), \\ g(x,y,t) = \sin(5\pi x)\sin(5\pi y)\exp(-t), \\ u_0(x,y) = xy(x-1)(y-1). \end{gathered} \tag{8.19}$$

The domain is $\Omega \times [0,T] = [0,1]^2 \times [0,1]$. In Table 8.3 we give the results for the solution of the semidiscrete problem with the GBDF formula with 2 steps and *GMRES*(20)/*FGMRES*(20) iterative methods with the various proposed preconditioners. Similarly to the other experiments, we observe that all the limited memory preconditioners, i.e., based on the short–memory principle, are optimal: the number of iterations to reach a prescribed tolerance is fixed, independent from the dimension. Moreover, as before, the approach with *FGMRES* turns out to be the fastest one. In this 2D case, we do not present the results with the circulant approximation of the Jacobian matrix, since it does not lead to a reasonable spectral approximation of the underlying block–matrix.

		GMRES(20)		I		P_s	
m	s	M_v	T(s)	M_v	T(s)	M_v	T(s)
25	32	359	0.498077	28	0.162611	28	**0.026580**
	64	417	0.065541	28	0.042177	29	0.043950
	128	618	0.230025	29	0.084474	29	0.075497
	256	1004	0.556410	28	**0.129884**	29	0.144670
	512	1700	1.367553	27	**0.229644**	28	0.254450
49	32	511	0.061850	28	0.074696	28	0.044302
	64	582	0.213668	28	0.079720	29	0.088193
	128	874	0.474627	29	0.164856	30	0.184416
	256	1361	1.110566	28	0.289902	30	0.355965
	512	2140	2.639608	28	0.579192	29	0.651599
97	32	803	0.300108	28	0.175620	28	0.131053
	64	1004	0.554717	28	0.254270	29	0.278705
	128	1479	1.108901	29	0.518899	30	0.588185
	256	2048	2.361870	28	0.896838	30	1.153711
	512	3066	6.686213	28	1.838700	29	2.046596
193	32	1633	0.928106	28	0.312824	28	0.328000
	64	2216	1.758452	28	0.845711	29	0.839759
	128	2833	3.458109	29	1.701079	30	1.625637
	256	3636	8.624317	28	2.893450	30	3.241947
	512	5192	25.985645	28	5.410698	29	5.354344
385	32	4483	3.821502	28	1.335643	28	1.265713
	64	4046	7.132265	28	2.835499	29	3.091758
	128	6815	21.917207	29	5.495737	30	8.458964
	256	8692	48.467187	28	13.032433	30	13.832191
	512	10024	114.332704	28	27.187875	29	25.970556
769	32	17445	34.674128	28	10.607206	28	9.517020
	64	16028	56.537935	28	19.210925	29	20.501788
	128	†	†	29	44.872253	30	49.015992
	256	†	†	28	81.530878	30	98.281073
	512	†	†	28	159.517824	29	181.350848
1537	32	†	†	28	72.905661	28	77.110471
	64	†	†	28	150.587943	29	168.761350
	128	†	†	29	335.425009	30	369.889844
	256	†	†	28	613.583401	30	743.860882
	512	†	†	28	1234.166093	29	1377.924956

continued on the next page

continued from previous page

		P_ω		P'_s		P'_ω	
m	s	M_v	T(s)	M_v	T(s)	M_v	T(s)
25	32	105	0.086074	106	0.064794	30	0.063938
	64	114	0.155477	108	0.091509	31	**0.035493**
	128	114	0.229144	115	0.179190	32	**0.048669**
	256	116	0.350810	116	0.345380	32	0.141315
	512	119	0.578919	118	0.562342	32	0.165098
49	32	160	0.147374	158	0.146482	31	**0.024588**
	64	174	0.228514	179	0.289489	31	**0.043588**
	128	184	0.380261	189	0.456471	32	**0.080920**
	256	192	0.678054	194	0.795719	32	**0.151359**
	512	195	1.302407	197	1.433853	32	**0.293215**
97	32	242	0.345816	253	0.440888	31	**0.057435**
	64	275	0.569038	290	0.712328	31	**0.098882**
	128	296	0.958323	312	1.192231	32	**0.192708**
	256	311	1.818854	324	2.246043	32	**0.374582**
	512	320	3.518119	335	4.218966	32	**0.747920**
193	32	359	0.907019	371	1.182327	31	**0.174783**
	64	437	1.579116	454	2.007617	32	**0.318097**
	128	502	3.061141	516	3.689773	32	**0.652213**
	256	538	5.941900	558	7.393889	32	**1.311898**
	512	567	11.399223	591	14.493837	32	**2.580167**
385	32	512	2.577256	517	3.118356	31	**0.565555**
	64	655	4.619358	671	5.792371	32	**1.131976**
	128	810	9.476015	826	11.498704	32	**2.122971**
	256	944	19.470493	963	24.454555	32	**4.353830**
	512	1035	45.186098	1057	53.588831	32	**8.243228**
769	32	722	6.652855	734	9.026140	31	3.160935
	64	1010	13.752854	1025	17.565095	32	6.743467
	128	1305	29.568440	1339	36.218418	32	14.076085
	256	1573	70.327472	1621	92.542980	32	28.237991
	512	1817	199.941485	1852	239.047024	32	57.684876
1537	32	978	20.342664	978	25.780316	31	15.895901
	64	1385	42.090657	1396	51.705245	32	33.590205
	128	1902	102.604220	1923	121.577188	32	67.754216
	256	2483	307.027933	2503	362.077657	32	135.465453
	512	2968	859.863198	2970	931.043836	32	278.942161

continued on the next page

continued from previous page

m	s	M_v	$P^{\text{FGMRES}}_{\omega,10}$ T(s)
25	32	12	0.390941
	64	3	0.130120
	128	3	0.249523
	256	12	1.929159
	512	12	3.750084
49	32	4	0.111798
	64	3	0.153173
	128	12	1.139513
	256	12	2.129826
	512	12	4.170408
97	32	3	0.099084
	64	12	0.719274
	128	12	1.351855
	256	12	2.551376
	512	12	4.802997
193	32	12	0.564502
	64	12	1.026921
	128	12	1.870541
	256	12	3.556693
	512	13	7.276354
385	32	12	0.926664
	64	12	1.678028
	128	12	3.047989
	256	12	5.621248
	512	13	11.728037
769	32	12	**1.543622**
	64	12	**2.712531**
	128	13	**5.460369**
	256	13	**10.281042**
	512	12	**18.724137**
1537	32	12	**2.313077**
	64	13	**4.522511**
	128	13	**8.845778**
	256	13	**17.389843**
	512	13	**33.900990**

Table 8.2. Experiment 1. Coefficients from equation (8.18), fractional order of differentiation $\alpha = 1.8$. The method used is the *GMRES*(20) for all the preconditioners except for $P^{\text{FGMRES}}_{\omega,10}$, using the approach with the *FGMRES* discussed in Section 8.2.

GMRES(20)		I		$P'_{\mathfrak{s}}$		$P_{\omega,\lceil m/10\rceil}$		$P^{\text{FGMRES}}_{\omega,10}$		$P^{\text{FGMRES}}_{\omega,5}$	
m	s	M_v	T(s)	M_v	T(s)	M_v	T(s)	M_v	T(s)	M_v	T(s)
25×25	32	266	**0.340921**	114	1.709925	34	2.509950	3	0.687541	3	0.220345
	64	354	**0.737725**	126	2.498359	33	4.031727	12	1.641349	11	1.411272
	128	520	**2.310232**	135	4.906858	33	8.018807	12	2.991839	12	2.874120
	256	838	7.673421	137	9.559409	32	14.279442	13	6.076410	13	**5.953192**
	512	1471	33.862683	138	18.976753	31	26.585789	13	12.048800	13	**11.433036**
49×49	32	512	3.470923	537	51.713398	34	26.305325	12	2.383546	12	**2.216880**
	64	633	8.502515	1214	228.157016	34	51.005210	12	4.383754	12	**3.791248**
	128	861	26.832703	1261	430.258106	33	100.230792	13	9.349177	13	**8.199694**
	256	1269	120.173661	1606	1192.707353	32	192.221346	13	19.039451	13	**17.025160**
	512	1985	443.243425	2391	3661.130769	32	433.940684	13	37.788963	13	**34.709295**
97×97	32	1209	77.714770	1919	1531.313896	34	1208.996561	13	10.958864	13	**9.126362**
	64	1362	220.144413	8051	12804.937808	34	2535.711822	13	20.057837	13	**17.836464**
	128	1622	577.350800	–	$> 4h$	34	5216.717315	13	38.867441	6	**15.510864**
	256	2275	1649.682022	–	$> 4h$	32	9198.798614	13	75.631715	13	**69.373086**
	512	3257	4636.613558	–	$> 4h$	31	16705.028289	14	160.233433	14	**147.280593**

Table 8.3. Experiment 2. Coefficients from equation (8.19), fractional order of differentiation $2\alpha = 1.1, 2\beta = 1.8$. We use *GMRES*(20) for all the proposed preconditioners with the exception of $P^{\text{FGMRES}}_{\omega,\cdot}$, used with *FGMRES*. See also the discussion in Section 8.2 for more details.

9

Future Perspectives

"The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say"

J.R.R. Tolkien, The Fellowship of the Ring

In this thesis we have addressed the numerical solution of some partial, fractional partial differential equations and optimization problems mainly by using preconditioners for sequences of sparse matrices and specific tools for *quasi–Toeplitz* structures.

The efforts for the problems discussed in the previous pages and resumed briefly below are prone to be extended in several directions, partly mentioned in the following items.

- In Chapter 3 we discussed the following update strategy for preconditioners in factorized form:

$$M_k^{-1} = W_k(D + E_k)^{-1}Z_k^T,$$

where the update is obtained by working on the triangular matrices W_k and Z_k; see (3.13). The other substantial component of this strategy is represented by the updating matrix E_k and by an efficient computation of $(D + E_k)^{-1}\mathbf{r}$ for a given vector $\mathbf{r}$. We saw that often an m–banded approximation of E_k (with m both $m \ll n$ and independent from n) permits to generate an updated preconditioner, that needs only the solution of a m–banded linear system and two matrix–vector product per application. Numerical tests show that in some specific cases the correction factors $D + E_k$ have a sparsity that can be exploited for the solution of the system with matrix $D + E_k$. One of the directions we

are already exploring consists in finding which data locality structures can help in designing faster and cheaper updates for some specific problems.

- In Chapter 5 we applied our update strategy from Chapter 3 to achieve a fast solution of linear and time dependent FPDEs (Fractional Partial Differential Equations), where only the derivatives in space can be fractional. We are working on a generalization of these techniques for nonlinear equations, with either fractional derivatives in space and possibly in time. This can lead to the solution of sequences of nonlinear algebraic equations that, when treated by quasi–Newton methods, require the solution of sequences of linear systems as discussed in Example 3.2. This task can be ultimately faced by exploiting the techniques in Chapters 3 and 5; see also the discussion at the end of Section 6.2.2.
- In Chapter 6 we considered an efficient algorithm for PDE constrained optimization. For this topic we are planning an extension of the techniques and instruments developed in Chapter 6 to face more challenging constrained optimization problems, both for the preconditioning update techniques for sequences of linear systems, and for the optimization routine. We emphasize that the latter can be extended to treat stronger nonlinearities, time–dependent constraints and also for considering cases in which the controls are multiplied by the state variable instead of being applied to the boundary condition or as a source, as it was, e.g., in (6.6).
- In Chapter 8 we proposed some hybrid-structured preconditioners for fast solution of FDEs. We plan to focus on a different approach for the application of the preconditioners proposed in Chapter 8. It is easy to observe that from the expression in (8.10) it is possible to recast the application of the underlying hybrid preconditioner to a vector $\mathbf{r}$, i.e., the solution of the linear system $\mathbf{x} = P^{-1}\mathbf{r}$, as the solution of a suitable *matrix equation*. Exploiting the spectral information obtained for P and its constituting blocks in the context of matrix equation solvers, we expect an advantage over the results in Chapter 8, both for storage and for timings.

A

A Brief Introduction to Fractional Calculus

"Natura non facit saltus"

G. W. Leibniz, Nouveaux essais (1704) IV, 16, 12

The origins of *fractional calculus* date back to the same years as calculus [88, 203, 241, 288]. When Leibniz developed his version of calculus and invented the notations

$$\frac{d^n f}{dx^n} \text{ and } \left(\frac{d}{dx}\right)^n f,$$

the question concerning the necessity of n being a natural number had been asked. In a letter by Leibniz to de L'Hospital, dated September 30, 1695[1], we read

> "John Bernoulli seems to have told you of my having mentioned to him a marvelous analogy which makes it possible to say in a way that successive differentials are in geometric progression. One can ask what would be a differential having as its exponent a fraction. You see that the result can be expressed by an infinite series. Although this seems removed from Geometry, which does not yet know of such fractional exponents, it appears that one day these paradoxes will yield useful consequences, since there is hardly a paradox without utility. Thoughts that mattered little in themselves may give occasion to more beautiful ones."

Some years later Euler [112] came back to the subject, observing that the choice of a fractional exponent can be seen as a suitable *interpolation* between some series expansion[2]

> "To round off this discussion, let me add something which certainly is more curious than useful. It is known that $d^n x$ denotes the differential of x of order n, and if p denotes any function of x and dx is taken to be constant, then $d^n p$ is homogeneous with dx^n; but whenever n is a positive whole number, the ratio of $d^n p$ to dx^n can be expressed algebraically; for example, if $n = 2$ and $p = x^3$, then $d^2(x^3)$ is to dx^2 as $6x$ to 1. We now ask, if n is a fractional number, what the value

1 Leibniz 1849-, II, XXIV, 197ff, translation by Mandelbrot [199, Chp. 41, Historical Sketches]

2 Translation from the Latin original by Stacy Langton.

> of that ratio should be. It is easy to understand the difficulty in this case; for if n is a positive whole number, d^n can be found by continued differentiation; such an approach is not available if n is a fractional number. But it will nevertheless be possible to disentangle the matter by using interpolation in progressions, which I have discussed in this essay."

The *target* is an extension of the ordinary calculus that is analogous to the extension of the Euler's $\Gamma(n)$ function on non–natural numbers.

In an early stage, contributions to the solution of this *paradox* were made successively by Lagrange [176], Laplace [180], Lacroix [175], Fourier [118], Abel [1] – who found the first application of fractional calculus for solving an integral equation coming from the tautochronous problem – Liouville [192], Riemann [238], Heaviside [155], Bateman [13], Hardy [153, 154], Weyl [287], Riesz [239] and many others, even if the applications that made it popular are indeed more recent.

The first attempts in giving a general definitions are by Lacroix [175] in which, from the formula for the nth derivative of $y = x^m$ for $n, m \in \mathbb{N}$, the formula

$$\left(\frac{d}{dx}\right)^{\alpha} x^{\beta} = \frac{\Gamma(\beta+1)}{\Gamma(\beta-\alpha+1)} x^{\beta-\alpha}, \ \alpha, \beta \in \mathbb{Q}, \tag{A.1}$$

was obtained. In particular, the 1/2th derivative of $y = x$ from the Leibniz's letter was recovered. Nevertheless, all the procedure is made accordingly with the formalistic approach of the time: no general or rigorous definition of the operation is given. Always based on formal manipulations are the Liouville's formulas [191], that extended the set of functions for which a fractional derivative could be computed to for exponential and rational functions. From the latter formula, another expression of Lacroix's formula (A.1) for the derivative of constants can be obtained and this led to a discrepancy between the two fractional derivatives. While (A.1) implies that the derivative of a constant is not zero, the Liouville's gives zero, as expected. As we will see, this is a feature that the subsequent progress on fractional calculus did not resolve; see [88]. We need to wait until the contribution of Fourier [118] for an *integral* representation of a fractional derivative, i.e.,

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(\xi) d\xi \int_{-\infty}^{\infty} \cos t(x-\xi) dt,$$

and thus, for $\alpha \in \mathbb{R}$, this representation yields (at least formally)

$$\frac{d^{\alpha} f(x)}{dx^{\alpha}} = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(\xi) d\xi \int_{-\infty}^{\infty} \cos\left[t^{\alpha}(x-\xi) + \frac{\pi\alpha}{2}\right] dt.$$

A.1 *Definition of Some Fractional Operators*

We focus on two contributions, the one by Riemann [238] and Liouville [191] extending the definition to obtain the fractional integral for a given $\alpha > 0$ and $a < b \in \mathbb{R} \cup \{\pm\infty\}$,

$$J^{\alpha}_{a,x} y(x) = \frac{1}{\Gamma(\alpha)} \int_a^x (x-\xi)^{\alpha-1} y(\xi) d\xi, \tag{A.2}$$

where $\Gamma(\cdot)$ is the *Euler gamma function* [2], i.e., the usual analytic continuation to all complex numbers (except the non–positive integers) of the convergent improper integral function

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx. \tag{A.3}$$

Definition A.2 arises naturally if one considers the solution of the ordinary linear differential non–homogeneous equation (Abel's equation),

$$\begin{cases} \dfrac{d^n y(x)}{dx^n} = f(x), & x \in [b,c], \quad f \in \mathcal{C}^{(0)}([b,c]), \\ y^{(k)}(a) = 0, & a \in (b,c), \quad k = 0, 1, \ldots, n-1. \end{cases}$$

Since a fundamental set of solutions of the corresponding homogeneous equation $d^n y(x)/dx^n$ is represented by $\{1, x, \ldots, x^{n-1}\}$, we obtain that, for any $a \in (b,c)$, the solution can be expressed as

$$y(x) = \frac{1}{(n-1)!} \int_a^x (x-\xi)^{n-1} f(\xi) d\xi = \frac{1}{\Gamma(n)} \int_a^x (x-\xi)^{n-1} f(\xi) d\xi,$$

from which (A.2) arises by substituting n by the real number α with $\Re(\alpha) > 0$. Thus, this fractional–order integral formula can be interpreted as a natural extension of an iterated integral; see again [88] for other possible derivation through *complex analysis* or *Laplace transform*. As a final step we summarize the discussion above in the following definition.

Definition A.1 (Riemann–Liouville Integral)**.** *Given* $f(x) \in \mathbb{L}^1([a,b])$ *and* $\mathbb{R} \ni \alpha > 0$*, then* (A.2) *is called* left–sided *fractional integral, while*

$$J^{\alpha}_{a,x} y(x) = \frac{1}{\Gamma(\alpha)} \int_x^b (\xi - x)^{\alpha-1} y(\xi) d\xi,$$

is called right–sided *fractional integral.*

Theorem A.1 (Fractional integral properties)**.** *Given* $a, b \in \mathbb{R}$ *with* $b > a$*,*

1. *Let Q be the reflection operator $Qf(x) = f(a+b-x)$, then*

$$QJ^{\alpha}_{a,x} = J^{\alpha}_{x,b}Q, \qquad QJ^{\alpha}_{x,b} = J^{\alpha}_{a,x}Q,$$

2. *Given $f, g \in \mathbb{L}^1(a,b)$ then the following formula for fractional integration by parts holds*

$$\int_a^b f(x)J^{\alpha}_{a,x}g(x)dx = \int_a^b g(x)J^{\alpha}_{x,b}f(x)dx,$$

3. *Given $\mathbb{R} \ni \alpha, \beta > 0$ and $f \in \mathbb{L}^1(a,b)$, then the* semigroup property of fractional integration *reads as*

$$J^{\alpha}_{a,x}J^{\beta}_{a,x}f = J^{\alpha+\beta}_{a,x}f, \quad J^{\alpha}_{x,b}J^{\beta}_{x,b}f = J^{\alpha+\beta}_{x,b}f, \qquad a.e.,$$

moreover, if $f \in \mathscr{C}([a,b])$ or $\alpha + \beta > 1$ then the relation is satisfied in every point.

Moving forward in this direction, the introduction of a *fractional derivative* of order α, with $\mathfrak{R}(\alpha) > 0$, becomes the next step. Let us observe that the standard derivative of order $n \in \mathbb{N}$ is only a *left inverse* of the n–fold integral $J^{n}_{a,x}y(x)$, since, in general, if used on the right can differ from the original function by a polynomial of degree $n-1$, i.e.,

$$J^{n}_{a,x}\left(\frac{d^n y(x)}{dx^n}\right) = y(x) - \sum_{k=0}^{n-1} y^{(k)}(a^+)\frac{(x-a)^k}{k!}, \quad x > a.$$

Therefore, what is reasonable to ask for a fractional derivative is only to be a left inverse of the Riemann–Liouville fractional integral and this induces the following definition.

Definition A.2 (Riemann-Liouville Derivative)**.** *For a given a function $y(x)$ with absolutely continuous first derivative, a given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m-1 < \alpha \leq m$, we define the left-side Riemann-Liouville fractional derivative as*

$$ {}_{RL}D^{\alpha}_{a,x}y(x) = \frac{1}{\Gamma(m-\alpha)}\left(\frac{d}{dx}\right)^m \int_a^x \frac{y(\xi)d\xi}{(x-\xi)^{\alpha-m+1}}, \tag{A.4}$$

and the right-side Riemann-Liouville fractional derivative as

$$ {}_{RL}D^{\alpha}_{x,b}y(x) = \frac{1}{\Gamma(m-\alpha)}\left(-\frac{d}{dx}\right)^m \int_x^b \frac{y(\xi)d\xi}{(\xi-x)^{\alpha-m+1}}. \tag{A.5}$$

Moreover, the two functions belong to space $\mathbb{L}^p([a,b])$ for $1 \leq p < 1/\alpha$.

Observe that, in general, the left– and right–hand side derivatives are not equal unless α is an even integer, a case in which the derivatives become *localized* and *equal*. On the other hand, if α is an odd integer, then these derivatives become again localized but opposite in sign, see also Proposition A.6. In general, the left–hand fractional derivative of the function y computed at a point $x \in (a, b)$ depends on *all function values* to the left of the point x, thus it is not localized! Similarly with the right–handed derivative and the point on the right; see Figure A.1.

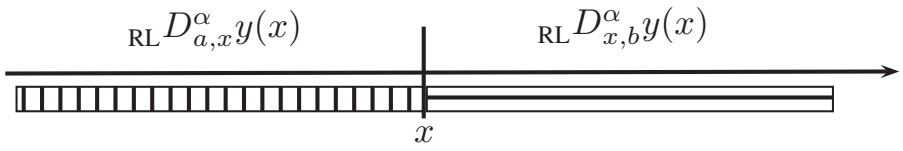


Figure A.1. Non locality for left– and right–sided Riemann–Liouville fractional derivative.

Let us consider now the *semigroup properties* also for the Riemann-Liouville fractional derivatives, as we have done in Theorem A.1.

Proposition A.1. *Assume that $\alpha, \beta \geq 0$, moreover let y be a function such that there exists $\phi \in \mathbb{L}^1([a, b])$ with $y = J^{\alpha+\beta}_{a,x}\phi$, then*

$$ {}_{RL}D^{\alpha}_{a,x}\,{}_{RL}D^{\beta}_{a,x}y = {}_{RL}D^{\alpha+\beta}_{a,x}y. $$

The regularity conditions imposed in the proposition above, that implies a certain convergence to zero of the fractional integrals, are not there only for technical reasons, i.e., there exist some cases where such a ϕ does not exists; see, e.g., [93]. Thus, no unconditional semigroup property of Riemann–Liouville fractional derivative exists. Nevertheless, this definition suffices for obtaining the features we requested at the beginning, i.e., the Riemann–Liouville derivative is indeed a left inverse of the Riemann–Liouville fractional integral.

Proposition A.2. *Let $\alpha > 0$. Then, for every $y \in \mathbb{L}^1([a, b])$ we have ${}_{RL}D^{\alpha}_{a,x}J^{\alpha}_{a,x}y = y$ almost everywhere.*

Proposition A.3. *Let $\alpha > 0$ and $m \in \mathbb{N}$ such that $m - 1 < \alpha \leq m$. Assume that y is such that $J^{m-\alpha}_{a,x}y$ has an absolutely continuous $(m-1)$th derivative. Then,*

$$ J^{\alpha}_{a,x}\,{}_{RL}D^{\alpha}_{a,x}y(x) = y(x) - \sum_{k=0}^{m-1} \frac{(x-a)^{\alpha-k-1}}{\Gamma(\alpha-k)} \lim_{z\to a^+} \frac{d^{m-k-1}}{dz^{m-k-1}} J^{m-\alpha}_{a,z} f(z). $$

Moreover, if there exists a $\phi \in \mathbb{L}^{(1)}([a, b])$ such that $y = J^{\alpha}_{a,x}\phi(x)$ then $J^{\alpha}_{a,x}\,{}_{RL}D^{\alpha}_{a,x}y(x) = y(x)$ almost everywhere.

Note that with this definition we recover also the Lacroix [175] expression for the fractional derivative of a power function (A.1). Thus, for $\alpha \notin \mathbb{N}$ the fractional derivative of $y(x) \equiv 1$ is *not* zero. Indeed,

$$ {}_{\mathrm{RL}}D^{\alpha}_{0,x}1 = \frac{x^{-\alpha}}{\Gamma(1-\alpha)}, \quad \alpha \geq 0,\ \alpha \notin \mathbb{N}, \quad x > 0. \tag{A.6} $$

A way to overcome this difficulty is by interchanging the process of differentiation and integration in Definition A.2, see [72]. Observe that this requires the mth derivative of $y(x)$ absolutely integrable.

Definition A.3 (Caputo Derivative)**.** *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m - 1 < \alpha \leq m$, the left-side Caputo fractional derivative of a function $y(x)$ such that $y^{(m)}$ is absolute integrable reads as*

$$ {}_{\mathrm{C}}D^{\alpha}_{a,x}y(x) = \frac{1}{\Gamma(m-\alpha)}\int_a^x \frac{y^{(m)}(\xi)d\xi}{(x-\xi)^{\alpha-m+1}}, \tag{A.7} $$

while the right-side

$$ {}_{\mathrm{C}}D^{\alpha}_{x,b}y(x) = \frac{(-1)^m}{\Gamma(m-\alpha)}\int_x^b \frac{y^{(m)}(\xi)d\xi}{(\xi-x)^{\alpha-m+1}}. \tag{A.8} $$

It is easy to recognize that in general

$$ {}_{\mathrm{RL}}D^{\alpha}_{a,x}y(x) = \frac{d^m}{dx^m}J^{m-\alpha}_{a,x}y(x) \neq J^{m-\alpha}_{a,x}\frac{d^m}{dx^m} = {}_{\mathrm{C}}D^{\alpha}_{a,x}y(x), $$

unless the function $y(x)$ is such that $y^{(k)}(a^+) = 0$ for $k = 0, \ldots, m-1$. In fact, assuming that we can exchange the derivative with the integral in Definition A.3, e.g., by a dominated convergence result, we have

$$ \begin{aligned} {}_{\mathrm{C}}D^{\alpha}_{a,t}y(x) &= {}_{\mathrm{RL}}D^{\alpha}_{a,t}y(x) - \sum_{k=0}^{m-1} f^{(k)}(a^+)\frac{x^{k-\alpha}}{\Gamma(k-\alpha+1)} \\ &= {}_{\mathrm{RL}}D^{\alpha}_{a,t}\left[y(x) - \sum_{k=0}^{m-1} f^{(k)}(a^+)\frac{x^k}{k!}\right]. \end{aligned} \tag{A.9} $$

It is obviously possible to investigate the composition of Riemann–Liouville integrals and Caputo Derivatives. What ones can find is that even the latter are a left inverse of the Riemann–Liouville integrals.

Proposition A.4. *If y is continuous and $\alpha \geq 0$, then ${}_{\mathrm{C}}D^{\alpha}_{a,x}J^{\alpha}_{a,x}y(x) = y(x)$.*

Again, we find that in general the Caputo derivative is not the right inverse of the Riemann–Liouville integral.

Proposition A.5. *Assuming that $\alpha \geq 0$, $m \in \mathbb{N}$ such that $m - 1 < \alpha \leq m$ and y has an absolutely continuous $(m-1)$th derivative. Then:*

$$J^{\alpha}_{a,x}\, {}_{C}D^{\alpha}_{a,x} y(x) = y(x) - \sum_{k=0}^{m-1} \frac{y^{(k)}(a^+)}{k!}(x-a)^k.$$

We have started our exploration having in mind the objective of "interpolating" between derivatives of integer orders, like the Euler Gamma function (A.3) interpolates the factorial over $\mathbb{R}$. In this sense we expect that the construction of fractional derivatives, either in the Riemann–Liouville or Caputo form, is such that if α is an integer, then the classical derivative is recovered. This is true only on one side as the following Proposition shows.

Proposition A.6 (Continuity w.r.t. α)**.** *Given a function $y(x)$, $\alpha > 0$ and $m \in \mathbb{N}$ such that $m - 1 < \alpha \leq m$. We have*

$$\lim_{\alpha \to m^-} {}_{RL}D^{\alpha}_{a,x} y(x) = \lim_{\alpha \to m^-} {}_{C}D^{\alpha}_{a,x} y(x) = \frac{d^m}{dx^m} y(x),$$

on the other hand,

$$\lim_{\alpha \to (m-1)^+} {}_{RL}D^{\alpha}_{a,x} y(x) = y^{(m-1)}(x),$$

and

$$\lim_{\alpha \to (m-1)^+} {}_{C}D^{\alpha}_{a,x} y(x) = y^{(m-1)}(x) - y^{(m-1)}(a^+).$$

The other approach we want to investigate is the Grünwald–Letnikov, based on the limit of a sum, filling both the gap between the usual definition of the derivative, through the limit of the incremental ratio of function, and the formulation as integral operators.

Definition A.4 (Grünwald [142]–Letnikov [184] fractional derivatives)**.** *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m - 1 < \alpha \leq m$, the left-side Grünwald–Letnikov fractional derivative of a function $y(x)$ is defined as*

$${}_{GL}D^{\alpha}_{a,x} y(x) = \lim_{\substack{h \to 0 \\ Nh = x-a}} \frac{1}{h^{\alpha}} \sum_{j=0}^{N} (-1)^j \binom{\alpha}{j} y(x - jh), \tag{A.10}$$

while the right-side

$${}_{GL}D^{\alpha}_{x,b} y(x) = \lim_{\substack{h \to 0 \\ Nh = b-x}} \frac{1}{h^{\alpha}} \sum_{j=0}^{N} (-1)^j \binom{\alpha}{j} y(x + jh). \tag{A.11}$$

Observe that if α goes to an integer m, (A.10) reduces to the derivative of integer order m, where $\binom{m}{j}$ is now the usual binomial coefficient. Generally speaking, the Definition A.4 of the Grünwald–Letnikov derivative, the Riemann–Liouville derivative, and thus the Caputo derivative are not equivalent. Only under suitable smoothness conditions for the function an equivalence can be recovered.

Proposition A.7. *Given $\alpha > 0$ and $m \in \mathbb{Z}^+$ such that $m - 1 < \alpha \leq m$ if $y \in \mathscr{C}^m([a, b])$, then for $x \in (a, b]$ and $x \in [a, b)$, we have:*

$$ {}_{RL}D^{\alpha}_{a,x}y(x) = {}_{GL}D^{\alpha}_{a,x}y(x), \qquad {}_{RL}D^{\alpha}_{x,b}y(x) = {}_{GL}D^{\alpha}_{x,b}y(x), $$

respectively.

Another natural question that is raised by Definition A.4 is: what happens if we replace the fractional order of α by $-\alpha$? We recover the original formulation of the fractional integral by Grünwald [142] and Letnikov [184], thus allowing for a formal unification of the two concepts of fractional derivatives and integrals.

The last type of fractional derivative that we are going to discuss is the *Riesz fractional derivative*:

Definition A.5 (Riesz Derivative)**.** *Given $\alpha > 0$ and $\alpha \notin \{2k + 1\}_{k\geq 0}$ and $m \in \mathbb{Z}^+$ such that $m - 1 < \alpha \leq m$, the symmetric Riesz derivative is given by*

$$ \frac{d^{\alpha}y(x)}{d|x|^{\alpha}} = \frac{1}{2\cos(\alpha\pi/2)}\left({}_{RL}D^{\alpha}_{a,x} + {}_{RL}D^{\alpha}_{x,b}\right). \tag{A.12} $$

A.1.1 Physical meaning of the fractional operators

As we have seen there exist many definitions of both fractional integrals and derivatives. On the other hand, for the classical calculus the situation is unambiguous. We have only one derivative and only one integral that are tight together by the *Fundamental Theorem of Calculus* and assume a well defined physical meaning: the integral $\int_a^b y(t)dt$ implies the displacement from a to b of a point moving at velocity $y(t)$. Similarly, if $s(t)$ is the displacement at time t, then $s'(t)$ stands for the velocity at time t while $s''(t)$ stands for the acceleration. On the other hand, as we have seen with Proposition A.6, neither the Riemann–Liouville fractional derivative (Definition A.2) nor the Caputo fractional derivative (Definition A.3) can be considered the mathematical generalization of the typical derivative. Also thinking at the $\Gamma(\cdot)$ function example we used at the beginning, we observe that the condition of being an

holomorphic function in the right half plane $\mathbb{C}^+$ satisfying $f(z+1) = zf(z)$ for each $z \in \mathbb{C}^+$ is not sufficient to have a true generalization of the factorial function. To select the proper extension making it unique, i.e., to prove the Wielandt's Theorem, an auxiliary conditions on boundedness is needed; see, e.g., [236]. The classical interpretation can be partially recovered through the interpretation of both fractional integrals (Definition A.1) and fractional derivatives (Definitions A.2, A.3 and A.5) in terms of *Stieltjes integral* with the opportune singular kernels, e.g.,

$$K_\alpha(\xi, x) = \begin{cases} -\dfrac{(x-\xi)}{\Gamma(\alpha+1)}, & \xi \in [a, x], \\ 0, & \xi < a, \end{cases}$$

for the left sided fractional integral with order α. In this way the fractional integral is reduced to the generalized displacement in the sense of $K_\alpha(\xi, x)$ if $y(t)$ is the velocity at time t. The fractional order of integration α controls the singularity of the integral, i.e., the smaller the index the stronger the singularity. The situation is completely similar, using the correct kernels, for the physical meaning of the fractional derivatives.

Another way to investigate the physical meaning of fractional operators is through the link between "random walk" processes and fractional dynamics [204]. One could be interested in diffusion processes that no longer follows Gaussian statistic, i.e., the classic Wiener process with Markov property tight to the standard Brownian motion. Therefore, we may want to consider deviations from the *linear time dependence*, Kt, of the mean squared displacement, $< \chi^2(t) >$. For example, this is the case of *anomalous diffusion* related to a non–linear growth in time of the mean squared displacement, i.e., $< \chi^2(t) > \sim K_\alpha t^\alpha$, where α is our fractional exponent manifesting in a power–law pattern.

A.2 *Fractional Partial Differential Equations*

Why should one bother, from the point of view of the applications, with the use of fractional calculus? The first answer in this direction can be hinted by thinking at the link between second order elliptic operators and Gaussian diffusion. The diffusion modeled with fractional operators, on the other hand, is *non–universal*, i.e., involves a parameter α, and *non–local*. As one could expect, there are in nature several phenomena that often violate universality and locality mirrored in the Gaussian models. Fractional diffusion equations have been developed to account for this *anomalous* features; see, e.g., [159, 197, 204, 235].

To proceed with our treatment we need now to introduce a substantial difference. In general, one can consider

- differential equations with fractional derivatives *in space*,
- differential equations with fractional derivatives *in time*,
- differential equations with fractional derivatives *in both space and time*.

For the problems whose numerical treatment is considered in this thesis, we will focus mostly on the *fractional derivatives in space* case; see [225]. This class of problems contains already several models of interests, e.g., the space fractional diffusion equation, the fractional advection–dispersion equation, the fractional advection–diffusion equation, the space fractional Fokker–Planck equation and the fractional partial differential equations with Riesz space fractional derivatives and many others.

Let us start by recalling two simple one-dimensional linear partial differential equations, the *parabolic equation*

$$\frac{\partial u(x,t)}{\partial t} = \kappa(x,t)\frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \qquad (x,t) \in Q = (L,R)\times(0,T]$$

and the *hyperbolic equation*

$$\frac{\partial u(x,t)}{\partial t} = v(x,t)\frac{\partial u(x,t)}{\partial x} + f(x,t), \qquad (x,t) \in Q = (L,R)\times(0,T].$$

We are interested in looking at their fractional generalization, i.e., we substitute the derivatives *in space* with fractional derivatives discussed in Appendix A.1. In this way we obtain the **fractional–diffusion** equation given by

$$\begin{cases} \dfrac{\partial u}{\partial t} = d_-(x,t)\,{}_{\mathrm{RL}}D^{\alpha}_{L,x}u + d_+(x,t)\,{}_{\mathrm{RL}}D^{\alpha}_{x,R}u + f, & (x,t)\in Q \\ u(x,0) = u_0(x), & x\in(L,R), \\ u(L,t) = u(R,t) = 0, & t\in(0,T], \end{cases}$$

where $d_+, d_-, f \in \mathscr{C}^0([L,R])$, and the transport coefficients d_+, d_- are such that $d_+, d_- \geq 0$, $\alpha \in [1,2]$, the function $f(x,t)$ is again a source/sink term. In particular, we can consider the additional advective term $-v(x,t)^{\partial u(x,t)}/_{\partial x}$ on the right–hand side in the case in which $\alpha \in (1,2]$ transforming the fractional diffusion into a **fractional advection–diffusion** equation; see, e.g., [252]. In some cases a difference is made between this form, called two–sided fractional diffusion equation, and the case in which only one of the left– and right–sided

derivatives appears. The case in which left– and right–sided fractional derivatives appear with the same coefficients is usually modeled as a **Riesz space-fractional diffusion equation**, i.e.,

$$\begin{cases} \dfrac{\partial u}{\partial t} = d(x,t)\dfrac{d^\alpha u}{d|x|^\alpha} + f, & (x,t) \in Q, \\ u(x,0) = u_0(x), & x \in (L,R), \\ u(L,t) = u(R,t) = 0, & t \in (0,T]. \end{cases}$$

Where $\alpha \in (1,2]$, $d(x,t) > 0$.

Another generalization of the classical PDE is represented by the **fractional advection–dispersion** equation in flux form for $Q = (-L,L)\times(0,T]$

$$\begin{cases} \dfrac{\partial u}{\partial t} = -\nu\dfrac{\partial u}{\partial x} + D\dfrac{\partial}{\partial x}\left(\dfrac{1+\eta}{2}J^{1-\alpha}_{-L,x}u + \dfrac{1-\eta}{2}J^{1+\alpha}_{x,L}u\right), & (x,t) \in Q \\ u(0,t) = g(t), & t \in [0,T], \\ u_t(0,t) = h(t), & t \in [0,T], \\ u(x,0) = u_0(x), & x \in [-L,L]. \end{cases},$$

for the skewness of the transport process $\eta \in [-1,1)$, controlling the bias in the dispersion, where $\alpha \in (0,1]$ is the order of the fractional derivatives and ν and D are macroscopic parameters; see, e.g., [131].

A.2.1 Sobolev spaces of fractional order

Sobolev spaces represent a tool for the analysis of the solutions of partial differential equations and, from the numerical point of view, the instrument needed to investigate the weak formulations, i.e., the Galerkin approximation in finite dimensional subspaces, together with the associated error analysis for the FPDEs.

The two main differences/difficulties, as reported in Appendix A.1, are that the fractional operators are *non local* and, moreover, their adjoint is not, in general, the negative of itself. Nevertheless, the tools that are needed for performing this analysis exist and have been already developed in another context: Sobolev spaces of fractional order or, briefly, fractional Sobolev spaces; see, e.g., the review in [92].

As in the standard case, there exist at least two equivalent ways for defining Sobolev spaces of, both integer and real (fractional) order, that are, namely, the so–called W– and H–definition. The first one arises with the introduction of *weak derivatives* through which an opportune norm is built. Let us work in $\mathbb{R}^d$ and consider the multi–index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$ such that its length is given by $|\boldsymbol{\alpha}| = \sum_{i=1}^d \alpha_i$, then for

any function y that is an m–differentiable function and any $\boldsymbol{\alpha}$ such that $|\boldsymbol{\alpha}| \leq m$ the $\boldsymbol{\alpha}$th derivative can be expressed as

$$D^{\boldsymbol{\alpha}} y(\mathbf{x}) = \frac{\partial^{|\boldsymbol{\alpha}|} y(\mathbf{x})}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}.$$

Thus we can obtain the *weak–derivatives*, extending the concept of derivative for function that are not in $\mathscr{C}^m$.

Definition A.6 (Weak derivative)**.** *Given $y, w \in \mathbb{L}^1(\Omega)$ for an open set $\Omega \subseteq \mathbb{R}^d$, a multi–index $\boldsymbol{\alpha}$, w is called a* weak $\boldsymbol{\alpha}$th derivative *of y if and only if*

$$\int_\Omega v(\mathbf{x}) D^{\boldsymbol{\alpha}} \phi(\mathbf{x})\, d\mathbf{x} = (-1)^{|\boldsymbol{\alpha}|} \int_\Omega w(\mathbf{x}) \phi(\mathbf{x})\, d\mathbf{x}, \qquad \forall\, \phi \in \mathscr{C}_0^\infty(\Omega).$$

At last to introduce Sobolev spaces we need to require some regularity of the boundary of the domain Ω.

Definition A.7 (Regular Domains)**.** *Let $\Omega \in \mathbb{R}^d$ be open and bounded, let V denote a function space on $\mathbb{R}^{d-1}$. We say that the boundary of Ω, i.e., $\partial\Omega$, is of class V if for each point $\bar{\mathbf{x}} \in \partial\Omega$ there exists an $r > 0$ and a function $g \in V$ such that*

$$\Omega \cap B_r(\bar{\mathbf{x}}) = \{\mathbf{x} \in B_r(\bar{\mathbf{x}}) \,:\, x_d > g(x_1, \ldots, x_{d-1})\},$$

where $B_r(\bar{\mathbf{x}})$ is the d–dimensional ball of radius r centered at $\bar{\mathbf{x}}$ and, if necessary, we admit a transformation of the coordinate system.

In particular, one is usually interested in Lipschitz, $\mathscr{C}^k$–regular and $\mathscr{C}^{k,\alpha}$–Hölder continuous ($\alpha \in (0, 1]$) boundary for V the corresponding function space.

Definition A.8 (Sobolev space of integer order)**.** *Let $k \in \mathbb{N}$, $p \in [1, +\infty]$. The* Sobolev space $W^{k,p}(\Omega)$ *for Ω a regular domain is the set of all the functions y such that for each multi–index $\boldsymbol{\alpha}$ with length less than k, the $\boldsymbol{\alpha}$th weak derivative exists and is a function in $\mathbb{L}^p(\Omega)$. Moreover, we define the norm:*

$$\|y\|_{W^{k,p}(\Omega)} = \begin{cases} \left(\displaystyle\sum_{|\boldsymbol{\alpha}| \leq k} |D^{\boldsymbol{\alpha}} y|^p \right)^{1/p}, & 1 \leq p < \infty, \\ \displaystyle\max_{|\boldsymbol{\alpha}| \leq k} \|D^{\boldsymbol{\alpha}} y\|_\infty, & p = \infty. \end{cases}$$

Let us now extend this definition to a real order, i.e., let us substitute the integer k with a real number $\alpha > 0$. We start fixing the *fractional exponent* $\alpha \in (0,1)$. Then for any $p \in [1, +\infty)$ we can extend the Definition A.8 by considering

$$W^{\alpha,p}(\Omega) = \left\{ u \in \mathbb{L}^p(\Omega) \,:\, \frac{|u(\mathbf{x}) - u(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^{d/p+\alpha}} \in \mathbb{L}^p(\Omega \times \Omega) \right\}, \tag{A.13}$$

that is indeed an intermediate Banach space between $\mathbb{L}^p(\Omega)$ and $W^{1,p}(\Omega)$ if endowed with the natural norm

$$\|u\|_{W^{s,p}(\Omega)} = \left(\int_\Omega |u|^p \, d\mathbf{x} + \iint_{\Omega\times\Omega} \frac{|u(\mathbf{x}) - u(\mathbf{y})|^p}{\|\mathbf{x} - \mathbf{y}\|^{d+\alpha p}} \, d\mathbf{x}\, d\mathbf{y} \right)^{1/p},$$

where

$$|u|_{W^{s,p}(\Omega)} = \left(\iint_{\Omega\times\Omega} \frac{|u(\mathbf{x}) - u(\mathbf{y})|^p}{\|\mathbf{x} - \mathbf{y}\|^{d+\alpha p}} \, d\mathbf{x}\, d\mathbf{y} \right)^{1/p}$$

is usually called the *Gagliardo* seminorm of u. Now, for the general case of a non integer $\alpha > 1$, it suffices to write $\alpha = k + \sigma$, where k is an integer and $\sigma \in (0,1)$. In this way we can extend easily Definition A.8 as

Definition A.9 (Sobolev space of fractional order)**.** *Given $\alpha = k + \sigma$ with $k \in \mathbb{N}$ and $\sigma \in (0,1)$, the space $W^{\alpha,p}(\Omega)$ consists of classes of functions $u \in W^{k,p}(\Omega)$ (see Definition A.8) whose distributional derivatives $D^{\boldsymbol{\alpha}} u$ for $|\boldsymbol{\alpha}| = k$ belong to $W^{\sigma,p}(\Omega)$ in the sense of* (A.13)*, and this is a Banach space with respect to the norm*

$$\|u\|_{W^{\alpha,p}(\Omega)} = \left(\|u\|^p_{W^{k,p}(\Omega)} + \sum_{|\boldsymbol{\alpha}|=k} \|D^{\boldsymbol{\alpha}} u\|^p_{W^{\sigma,p}(\Omega)} \right)^{1/p}.$$

What is needed now is a link between Definition A.9 and the fractional derivatives we have defined, i.e., Riemann–Liouville fractional derivative (Definition A.2) and Riesz fractional derivative (Definition A.5). To achieve this, we need to obtain the so–called H–definition of Sobolev spaces, thus restricting to the case $p = 2$, i.e., using $\mathbb{L}^2(\Omega)$ as starting space. So let $\mathfrak{S}$ be the Schwartz space of rapidly decaying $\mathscr{C}^\infty(\mathbb{R}^n)$ functions. Then we can consider the topology generated by the seminorms,

$$|\varphi|_n = \sup_{\mathbf{x}\in\mathbb{R}^n} (1 + \|\mathbf{x}\|)^N \sum_{|\boldsymbol{\alpha}|\le N} \|D^{\boldsymbol{\alpha}} \varphi(\mathbf{x})\|, \quad N \ge 0, \quad \varphi \in \mathfrak{S}({}^n\mathbb{R}).$$

Moreover, $\mathfrak{S}'$ is the topological dual of $\mathfrak{S}$, i.e., the set of all *tempered distributions*. Then for any function $\varphi \in \mathfrak{S}$ the Fourier transform of φ is denoted by

$$\mathfrak{F}\varphi(\xi) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{-i\xi x} \varphi(x) dx,$$

that can be extended from $\mathfrak{S}$ to $\mathfrak{S}'$. We can now consider the alternative definition of the space $H^s(\mathbb{R}^n) = W^{s,2}(\mathbb{R}^n)$ via the Fourier transform. Therefore, what is needed, for $s > 0$, is to prove that

$$\tilde{H}^s(\mathbb{R}^n) = \left\{ u \in \mathbb{L}^2(\mathbb{R}^n) \, : \, \int_{\mathbb{R}^n} (1 + \|\xi\|^{2s}) \|\mathfrak{F}u(\xi)\|^2 d\xi < +\infty \right\}, \quad \text{(A.14)}$$

is equivalent to the space $W^{s,2}(\mathbb{R}^n)$.

Proposition A.8. *Let $s \in (0,1)$. Then the fractional Sobolev space $W^{s,2}(\mathbb{R}^n)$ from Definition A.9 coincides with $\tilde{H}^s(\mathbb{R}^n)$ defined in* (A.14)*, thus we will identify it in any case simply as $H^s(\mathbb{R}^n) \equiv \tilde{H}^s(\mathbb{R}^n) \equiv W^{s,2}(\mathbb{R}^n)$. In particular, for any $u \in \mathbb{H}^S(\mathbb{R}^n)$ we find that*

$$\|u\|^2_{H^s(\mathbb{R}^n)} = 2C(n,s)^{-1} \int_{\mathbb{R}^n} |\xi|^{2s} |\mathfrak{F}u(\xi)|^2 \, d\xi,$$

where

$$C(n,s)^{-1} = \int_{\mathbb{R}^n} \frac{1 - \cos(\zeta)}{|\zeta|^{n+2s}} d\zeta.$$

The proof of this proposition, for which we refer to [92] and references therein, relies on the Plancherel's formula, i.e., to the fact that we are using as starting space $\mathbb{L}^2$.

As a final step, we can focus on the connection between the operators defined in Appendix A.1 and this formulation of the fractional Sobolev spaces. For doing this we will follow the approach in [110]. Therefore, we will start by giving some definitions of the objects we need. To keep the notation simple, we restrict ourselves to the case of $n = 1$, as already done with the definition of the fractional operators.

Definition A.10 (Left Fractional Derivative Space)**.** *Let $\mu > 0$. We define the semi–norm*

$$|u|_{J_L^\mu(\mathbb{R})} = \| {}_{RL}D^{\mu}_{-\infty,x} u \|_2,$$

the norm

$$\|u\|_{J_L^\mu} = (\|u\|_2^2 + |u|^2_{J_L^\mu})^{1/2},$$

and let $J_L^\mu(\mathbb{R})$ denote the closure of $\mathscr{C}_0^\infty(\mathbb{R})$ with respect to the norm $\|\cdot\|_{J_L^\mu}$.

Definition A.11 (Right Fractional Derivative Space)**.** *Let $\mu > 0$. We define the semi–norm*

$$|u|_{J_R^\mu(\mathbb{R})} = \| {}_{RL}D^{\mu}_{x,+\infty} u\|_2,$$

the norm

$$\|u\|_{J_R^\mu} = (\|u\|_2^2 + |u|_{J_R^\mu}^2)^{1/2},$$

and let $J_R^\mu(\mathbb{R})$ denote the closure of $\mathscr{C}_0^\infty(\mathbb{R})$ with respect to the norm $\|\cdot\|_{J_R^\mu}$.

Definition A.12 (Symmetric Fractional Derivative Space)**.** *Let $\mu > 0$ with $\mu \neq n - 1/2$, $n \in \mathbb{N}$. We define the semi–norm*

$$|u|_{J_S^\mu(\mathbb{R})} = | < {}_{RL}D^{\mu}_{x,+\infty} u, {}_{RL}D^{\mu}_{x,+\infty} u >_{\mathbb{L}^2(\mathbb{R})} |^{1/2},$$

the norm

$$\|u\|_{J_S^\mu} = (\|u\|_2^2 + |u|_{J_S^\mu}^2)^{1/2},$$

and let $J_S^\mu(\mathbb{R})$ denote the closure of $\mathscr{C}_0^\infty(\mathbb{R})$ with respect to the norm $\|\cdot\|_{J_S^\mu}$.

In this way we have mimicked the construction of the fractional Sobolev spaces via the Gagliardo norm, see Definition A.9. Now we need the analogous of Proposition A.8.

Proposition A.9 (Ervin and Roop [110])**.** *Given $\mu > 0$ the following facts hold true*

- *the spaces $J_L^\mu(\mathbb{R})$ and $H^\mu(\mathbb{R})$ are equal, with equivalent semi–norms and norms;*
- *the spaces $J_L^\mu(\mathbb{R})$ and $J_R^\mu(\mathbb{R})$ are equal, with equivalent semi–norms and norms;*
- *if $\mu \neq n - 1/2$, $n \in \mathbb{N}$, then the spaces $J_L^\mu(\mathbb{R})$ and $J_S^\mu(\mathbb{R})$ are equal, with equivalent semi–norms and norms.*

To extend the definitions and the equivalence to a bounded open subinterval $\Omega = (L, R) \subset \mathbb{R}$, it is sufficient to consider the closure in the relative norms (Definitions A.10, A.11 and A.12) of the space $\mathscr{C}_0^\infty(\Omega)$.

Proposition A.10 (Ervin and Roop [110])**.** *Let $\mu > 0$, $\mu \neq n - 1/2$, $n \in \mathbb{N}$. Then the space $J_{S,0}^\mu(\Omega)$, $J_{L,0}^\mu(\Omega)$, $J_{R,0}^\mu(\Omega)$ and $H_0^\mu(\Omega)$ are equal, with equivalent semi–norms and norms.*

A.3 *Some Classical Discretization Formulas*

Most of the discretization schemes for partial differential equations have been extended to FDEs. Thus, there exist methods for discretizing them in *strong* form, e.g., finite differences (FD) discretization, and methods that exploit the *weak* forms through the use of the Fractional Sobolev spaces from Appendix A.2.1.

Here we will focus on the use of finite differences schemes for our derivatives, while some use of *weak formulation* is used in Chapter 6 for treating FDE control problems that, nonetheless, will be solved indeed by using *strong formulation*.

A.3.1 Finite Differences for Riemann–Liouville Fractional Derivatives

For numerical calculation of fractional–order derivatives we can use the relation given by Proposition A.7 and Equation (A.9) to obtain finite difference discretizations of both Riemann–Liouville and Caputo derivatives. Thus for functions $y(x)$ suitably smooth, in the sense of Proposition A.7, we can use the Grünwald–Letnikov expression (A.10) to approximate the left–sided Riemann–Liouville derivative as,

$$ {}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} = \frac{1}{\Delta x^{\alpha}}\sum_{j=0}^{n}\omega_j^{(\alpha)}y(x_{n-j}) + O(\Delta x), \qquad \begin{array}{l}\{x_k = j\Delta x\}_{j=0}^{n},\\ \omega_j^{(\alpha)} = (-1)^j\binom{\alpha}{j},\end{array} $$

that is convergent approximation of order 1 for any $\alpha > 0$. For the sake of obtaining stable scheme for FDEs, we are interested also in a slight modification of this formula for $\alpha \in (1,2)$, that is the *shifted Grünwald–Letnikov* approximation formula.

Definition A.13 (p–shifted Grünwald–Letnikov formula)**.** *Given $\alpha \in (1,2)$, $y \in \mathscr{C}^2([0,1])$ with $y(0) = 0$ and $p \in \mathbb{N}$, then for $x \in (0,1]$ the* right shifted Grünwald–Letnikov formula *with p shifts to approximate the left Riemann–Liouville derivative is defined by:*

$$ {}_{RL}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} = \frac{1}{\Delta x^{\alpha}}\sum_{j=0}^{n+p}\omega_j^{(\alpha)}y(x_{n-j+p}) + o(\Delta x), \qquad \begin{array}{l}\{x_k = j\Delta x\}_{j\geq 0},\\ \omega_j^{(\alpha)} = (-1)^j\binom{\alpha}{j}.\end{array} \cdot $$

The best performances in terms of orders and stability for this formula is obtained when choosing a value of p that minimizes the quantity $|p-\alpha/2|$ that, for $\alpha \in (1,2)$, is equivalent to choose $p = 1$. Observe that for $\alpha = 2$ Definition A.13 reduces to the second order central

difference method for $y''(x)$. Let us investigate now what happens in the case $y(0) \neq 0$. Let us consider the function $y(x) = x^m$ for m a non–negative integer. By choosing again the grid $\{x_k = j\Delta x\}_{j\geq 0}$, $m = 0$ the residual term is given by:

$$\frac{1}{\Delta x^\alpha}\sum_{j=0}^{n}\omega_j^{(\alpha)}y(x_{n-j}) = {}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} + (1-\alpha)\frac{-\alpha x^{-1-\alpha}}{2\Gamma(1-\alpha)}\Delta x + O(\Delta x^2),$$

while form $m > 0$, we have

$$\frac{1}{\Delta x^\alpha}\sum_{j=0}^{n}\omega_j^{(\alpha)}y(x_{n-j}) = {}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} - \alpha\frac{\Gamma(m+1)x^{m-1-\alpha}}{2\Gamma(m-\alpha)}\Delta x + O(\Delta x^2),$$

and, similarly, for the shifted Grünwald–Letnikov formula, and for $m = 0$ the residual term is given by:

$$\frac{1}{\Delta x^\alpha}\sum_{j=0}^{n+1}\omega_j^{(\alpha)}y(x_{n-j}) = {}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} + (3-\alpha)\frac{-\alpha x^{-1-\alpha}}{2\Gamma(1-\alpha)}\Delta x + O(\Delta x^2),$$

while form $m > 0$, we obtain

$$\frac{1}{\Delta x^\alpha}\sum_{j=0}^{n+1}\omega_j^{(\alpha)}y(x_{n-j}) = {}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} + (2-\alpha)\frac{\Gamma(m+1)x^{m-1-\alpha}}{2\Gamma(m-\alpha)}\Delta x + \\ + O(\Delta x^2).$$

Therefore, whenever $y(0) \neq 0$, we find that the Grünwald–Letnikov formula does not have first–order accuracy and a suitable correction is needed. As we have observed when defining the Riemann–Liouville derivative, the derivative of a constant is not zero, thus we can use Equation (A.6) to obtain the correction we need

$$\begin{aligned}{}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} &= {}_{\mathrm{RL}}D^{\alpha}_{0,x}\left(y(x)-y(0)\right)\Big|_{x=x_n} + \frac{y(0)x_n^{-\alpha}}{\Gamma(1-\alpha)} \\ &\approx \frac{1}{\Delta x^\alpha}\sum_{j=0}^{n+p}\omega_j^{(\alpha)}\left(y(x_{n-j+p})-y(0)\right) + \frac{y(0)x_n^{-\alpha}}{\Gamma(1-\alpha)},\end{aligned} \tag{A.15}$$

that achieves again first order convergence and is exact when $y(x)$ is a constant. To obtain a second–order method we can further modify equation (A.15). Computing it again on $y(x) = x^m$ for $\mathbb{N} \ni m \geq 0$ and

eliminating the term that multiplies Δx we obtain the method:

$$
{}_{\mathrm{RL}}D^{\alpha}_{0,x}y(x)\Big|_{x=x_n} = \frac{1}{\Delta x^{\alpha}}\left[\frac{2-\alpha}{2}\sum_{j=0}^{n}\omega_j^{(\alpha)}\left(y(x_{n-j})-y(0)\right)\right.
$$
$$
\left.+\frac{\alpha}{2}\sum_{j=0}^{n+1}\omega_j^{(\alpha)}\left(y(x_{n-j+1})-y(0)\right)\right]+\frac{y(0)x_n^{-\alpha}}{\Gamma(1-\alpha)}+O(\Delta x^2),
$$

that is a second order method for any $y \in \mathscr{C}^2([0,1])$.

To use this method we need to compute in an efficient way the coefficients $\{\omega_j^{(\alpha)}\}_j$ that appear in the formula. Using the direct expression in term of the binomial, i.e., $\omega_j^{(\alpha)} = (-1)^j\binom{\alpha}{j}$, is obviously cumbersome. One of the possible approaches for this computation is to use the recurrence relationships:

$$
\omega_0^{(\alpha)} = 1, \quad \omega_j^{(\alpha)} = \left(1-\frac{\alpha+1}{j}\right)\omega_{j-1}^{(\alpha)}, \quad j \geq 1.
$$

Observe, moreover, that the coefficients $\{\omega_j^{(\alpha)}\}_j$ can be regarded as the coefficients of the power series expansion of the function $f_\alpha(z) = (1-z)^\alpha$, since:

$$
f_\alpha(z) = (1-z)^\alpha = \sum_{j=0}^{+\infty}(-1)^j\binom{\alpha}{j}z^j = \sum_{j=0}^{+\infty}\omega_j^{(\alpha)}z^j.
$$

From this relationship we have also a series expansion of $f_\alpha(\exp(-\imath\theta))$ for $\theta \in [-\pi,\pi]$, thus the $\{\omega_j^{(\alpha)}\}_j$ are also the coefficient of the Fourier expansion of $f_\alpha(\theta) \triangleq f_\alpha(\exp(-\imath\theta))$. Therefore, the coefficients $\{\omega_j^{(\alpha)}\}_j$ can be computed using any suitable implementation of the FFT.

The feature we just uncovered open also the path to several theoretical interpretation. Firstly let us observe that, by the usual link between regularity and order of magnitude of the Fourier coefficients of a regular function, this implies that the coefficients $\{\omega_j^{(\alpha)}\}_j$ present a polynomial decay depending on the value of α, i.e., $|\omega_j^{(\alpha)}| = O(j^{-\alpha-1})$, for $j \to +\infty$.; see the application of this fact in Chapters 5 and 8.

Second, consider the numerical computation of the Riemann–Liouville derivative of $y(x)$ through Definition A.13. If $\mathbf{y}$ is the vector containing the evaluation of $y(x)$ over the grid $\{x_k = j\Delta x\}_{j=0}^n$, i.e., $(\mathbf{y})_j = y_j = y(x_j)$, then the numerical computation can be achieved by

the matrix–vector product:

$$\frac{1}{\Delta x^{\alpha}} T_{n+1}(f_{\alpha})\mathbf{y} = \frac{1}{\Delta x^{\alpha}} \begin{bmatrix} \omega_1 & \omega_0 & 0 & \dots & 0 \\ \omega_2 & \omega_1 & \omega_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \omega_1 & \omega_0 \\ \omega_{n-1} & \dots & \dots & \omega_2 & \omega_1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix},$$

where $T_{n+1}(f_{\alpha})$ is the Toeplitz Matrix (Definition 2.4) generated by the function $f_{\alpha}(\theta)$. Moreover, the sequence of Toeplitz matrices is distributed in the sense of the singular values as $f_{\alpha}(\theta)$ (Definition 2.6): $\{T_{n+1}(f_{\alpha})\}_n \sim_{\sigma} f_{\alpha}(\theta)$; see [100] and references therein. In our work this result and its extensions are used in Chapter 8; see also [226, 228, 229] for further information on the matrix representations of discretized fractional differential operators.

The last relevant information we are going to extract from the function $f_{\alpha}(\theta)$ is the following: $f_{\alpha}(\theta)$ is the αth power of the function that generates the coefficients for the first order approximation of the first order derivative, i.e., $f_1(\theta) = 1 - \exp(-\imath\theta)$. From this observation in [196] it is obtained that the αth power of the $(p+1)$–point backward difference gives the pth order approximation of the αth derivative. Therefore, we can obtain the 2nd order expression of the αth derivative by considering the series expansion of the function

$$f_{\alpha}^{(2)}(z) = \left(\frac{3}{2} - 2z + \frac{1}{2}z^2\right)^{\alpha},$$

or by doing the substitution with $z = \exp(-\imath\theta)$ and proceeding by computing a FFT and at the same way for the formula of higher order; see again [196] for the details.

At last, we have performed all the analysis for the left–sided derivative. All can be restated for the right–sided. One simply substitute to the function $f_{\alpha}^{(p)}(\theta)$ its complex conjugate or looking at the transpose of $T_{n+1}(f_{\alpha}^{(p)}(\theta))^T = T_{n+1}(\overline{f_{\alpha}^{(p)}(\theta)})$, i.e., we start using the αth power of the functions that generate the coefficients for the first order approximation of the first order derivative but, this time, we use the $(p+1)$–point forward difference formula. In this case the standard Grünwald–Letnikov formula is expressed as:

$${}_{\mathrm{RL}}D_{x,1}^{\alpha} y(x)\Big|_{x=x_n} \approx \frac{1}{\Delta x^{\alpha}} \sum_{j=0}^{n} (-1)^j \binom{\alpha}{j} y(x_{n+j}) = \frac{1}{\Delta x^{\alpha}} \sum_{j=0}^{n} \omega_j^{(\alpha)} y(x_{n+j}),$$

while the p–shifted version is obtained as

$$\begin{aligned} {}_{\text{RL}}D^{\alpha}_{x,1}y(x)\Big|_{x=x_n} &\approx \frac{1}{\Delta x^{\alpha}}\sum_{j=0}^{n+p}(-1)^j\binom{\alpha}{j}y(x_{n+j-p}) \\ &= \frac{1}{\Delta x^{\alpha}}\sum_{j=0}^{n+p}\omega_j^{(\alpha)}y(x_{n+j-p}). \end{aligned}$$

Observe that in this case all the considerations and the conditions regarding the order of convergence are obtained by looking at the value of $y(x)$ at the right end of the interval, i.e., $y(1)$.

A.3.2 Finite Differences for Riesz Fractional Derivatives

In this section, we focus on some formulas for the Riesz derivative (Definition A.5) with order $\alpha \in (1,2)$. From the Definition, one can go for the use of the formulas from Appendix A.3.1 by

$$\begin{aligned} \frac{d^{\alpha}y(x)}{d|x|^{\alpha}}\Big|_{x=x_n} &= \frac{1}{2\cos(\alpha\pi/2)}\left({}_{\text{RL}}D^{\alpha}_{a,x} + {}_{\text{RL}}D^{\alpha}_{x,b}\right)y(x)\Big|_{x=x_n} \\ &\approx \frac{1}{2\Delta x^{\alpha}\cos(\alpha\pi/2)}\left(\sum_{j=0}^{n+p}\omega_j^{(\alpha)}y(x_{n-j+p}) + \sum_{j=0}^{n+p}\omega_j^{(\alpha)}y(x_{n+j-p})\right), \end{aligned}$$

that inherits all the theoretical features we have discussed in connection with the Grünwald–Letnikov formulas in the cases of one–sided Riemann–Liouville derivatives.

The other scheme to be considered is the *symmetrical fractional central difference operators*, that is obtained as

$$\frac{\partial^{\alpha}y(x)}{\partial|x|^{\alpha}} = -\frac{1}{2\cos(\alpha\frac{\pi}{2})\Gamma(2-\alpha)}\frac{d^2}{dx^2}\int_{\mathbb{R}}\frac{y(\xi)d\xi}{|x-\xi|^{\alpha-1}}, \quad \alpha\in(1,2]$$

and is equivalent to the *fractional central difference formula* [220]:

$$\delta^{(\alpha)}_{\Delta x}y(x) = \sum_{j=-\infty}^{+\infty}\frac{(-1)^j\Gamma(\alpha+1)}{\Gamma(\alpha/2-j+1)\Gamma(\alpha/2+j+1)}y(x-j\Delta x).$$

The latter can be by truncated as follows

$$\begin{aligned} \frac{d^{\alpha}y(x)}{d|x|^{\alpha}}\Big|_{x=x_n} &= \frac{1}{\Delta x^{\alpha}}\sum_{|j|\le n}\frac{(-1)^j\Gamma(\alpha+1)}{\Gamma(\alpha/2-j+1)\Gamma(\alpha/2+j+1)}y(x-j\Delta x) + O(\Delta x^2) \\ &= \frac{1}{\Delta x^{\alpha}}\sum_{|j|\le n}g_j^{(\alpha)}y(x-j\Delta x) + O(\Delta x^2) \qquad \text{(A.16)} \end{aligned}$$

becoming a second order formula; see also [75] for further details. For using this approximation we need again an efficient way to compute the coefficients $\{g_j^{(\alpha)}\}_j$.

Proposition A.11 (Çelik and Duman [75]). *Let $\{g_j^{(\alpha)}\}_j$ be the coefficients in* (A.16) *for $j \in \mathbb{Z}$, and $\alpha > -1$. Then*

- $g_0^{(\alpha)} \geq 0$, $g_{-j} = g_j \leq 0$ *for all* $|j| \geq 1$,
- $g_{j+1}^{(\alpha)} = \left(1 - \frac{\alpha+1}{\alpha/2+j+1}\right) g_j^{(\alpha)}$ *and* $g_0^{(\alpha)} = \Gamma(\alpha+1)/\Gamma(\alpha/2+1)^2$,
- $g_j^{(\alpha)} = O(j^{-\alpha-1})$ *for* $j \to +\infty$.

Observe that Proposition A.11 implies also that again the coefficients for the discretization formula are the Fourier coefficients of the bounded $\mathbb{L}^1$ function:

$$g_\alpha(z) = |2\sin(z/2)|^\alpha,$$

that g_α is the spectral symbol for the sequence of Toeplitz matrices obtained from the coefficients $\{g_j^{(\alpha)}\}_j$, see Definition 2.5.

Acknowledgments

> "Alas, eleventy one years is far too short a time to live among such excellent and admirable Hobbits. I don't know half of you half as well as I should like and I like less than half of you half as well as you deserve."
>
> J.R.R. Tolkien, The Fellowship of the Ring.

Research is not a work that one does on its own, thus the first due thanks go to all the people who made the existence of these pages possible. First of all, my two advisors Daniele Bertaccini and Stefano Serra–Capizzano, whose suggestions have enriched my way of seeing and doing research in mathematics. Then, the other collaborators with whom this research was written and discussed (in strict alphabetical order): Daniele Bertaccini, Stefano Cipolla, Marco Donatelli, Marina Popolizio and Stefano Serra–Capizzano.

A special thanks goes to the two anonymous Reviewers. Their work and their effort spurred me to improve the proposed material. In this regard, further thanks go back to Stefano Serra–Capizzano, Sven-Erik Ekström and Ali Dorostkar, who have invested much of their spare time to help me in the tense days of the review process.

To make good research, one of the fundamental points was finding myself in a humanly welcoming and scientifically sparkling environment. For this reason, I would like to thank my fellow doctoral students for their friendship, their feedbacks, and for all the challenging discussion we have entertained during this three years.

Last but not least, a sincere "thank you" goes to my friends and my family who, despite being completely outside the world of mathematics and probably not having yet understood what I do, have endured my complaints, my tiredness and all the negative effects of my work.

I thank you all for being there for me.

Fabio Durastante

Bibliography

[1] N. H. Abel. "Solutions de quelques problèmes à l'aide d'intégrales définies (1823)". In: *Œuvres complètes de Niels Henrik Abel* 1 (), pp. 11–18 (cit. on p. 248).

[2] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Applied mathematics series. Dover Publications, 1964. ISBN: 9780486612720 (cit. on pp. 182, 249).

[3] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Guettel. "Implementation of a restarted Krylov subspace method for the evaluation of matrix functions." In: *Linear Algebra Appl.* 429.10 (2008), pp. 2293–2314 (cit. on pp. 122, 140).

[4] M. Annunziato, A. Borzì, M. Magdziarz, and A. Weron. "A fractional Fokker–Planck control framework for subdiffusion processes". In: *Optim. Control. Appl. Methods* 37.2 (2016). oca.2168, pp. 290–304. ISSN: 1099-1514. DOI: 10.1002/oca.2168. URL: http://dx.doi.org/10.1002/oca.2168 (cit. on p. 167).

[5] H. Antil and E. Otarola. "A FEM for an optimal control problem of fractional powers of elliptic operators". In: *SIAM J. Control. Optim.* 53.6 (2015), pp. 3432–3456 (cit. on p. 167).

[6] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9 (1951), pp. 17–29 (cit. on pp. 24, 33).

[7] G. Astrakhantsev. "An iterative method of solving elliptic net problems". In: *Z. Vycisl. Mat. i. Mat. Fiz.* 11.2 (1971), pp. 171–182 (cit. on p. 75).

[8] F. Avram. "On bilinear forms in Gaussian random variables and Toeplitz matrices". In: *Probab. Theory Related Fields* 79.1 (1988), pp. 37–45 (cit. on pp. 62, 63).

[9] A. Axelsson and J. Verwer. "Boundary Value Techniques for Initial Value Problems in Ordinary Differential Equations". In: *Math. Comp.* 45 (1985), pp. 153–171 (cit. on p. 225).

[10] O. Axelsson and P. S. Vassilevski. "A black box generalized conjugate gradient solver with inner iterations and variable–step preconditioning". In: *SIAM J. Matrix Anal. Appl.* 12.4 (1991), pp. 625–644 (cit. on p. 45).

[11] R. E. Bank and T. Dupont. "An optimal order process for solving finite element equations". In: *Math. Comp.* 36.153 (1981), pp. 35–51 (cit. on p. 75).

[12] F. Bassi, A. Ghidoni, and S. Rebay. "Optimal Runge-Kutta smoothers for the p-multigrid discontinuous Galerkin solution of the 1D Euler equations". In: *J. Comput. Phys.* (230 2010), pp. 4153–4175 (cit. on p. 206).

[13] H. Bateman. "On dissipative systems and related variational principles". In: *Physical Review* 38.4 (1931), p. 815 (cit. on p. 248).

[14] B. Beckermann and S. Serra-Capizzano. "On the asymptotic spectrum of Finite Element matrix sequences". In: *SIAM J. Numer. Anal.* 45 (2007), pp. 746–769 (cit. on p. 196).

[15] S. Bellavia, D. Bertaccini, and B. Morini. "Nonsymmetric Preconditioner Updates in Newton–Krylov Methods for Nonlinear Systems". In: *SIAM J. Sci. Comput.* 33.5 (2011), pp. 2595–2619 (cit. on pp. 105, 106, 108, 111, 144, 156, 179, 180, 191).

[16] S. Bellavia, V. De Simone, D. Di Serafino, and B. Morini. "A preconditioning framework for sequences of diagonally modified linear systems arising in optimization". In: *SIAM J. Num. Anal.* 50.6 (2012), pp. 3280–3302 (cit. on p. 102).

[17] S. Bellavia, V. De Simone, D. Di Serafino, and B. Morini. "Updating constraint preconditioners for KKT systems in quadratic programming via low-rank corrections". In: *SIAM J. on Optimization* 25.6 (2015), pp. 1787–1808 (cit. on p. 102).

[18] S. Bellavia, V. D. Simone, D. Di Serafino, and B. Morini. "Efficient preconditioner updates for shifted linear systems". In: *SIAM J. Sci. Comput.* 33.4 (2011), pp. 1785–1809 (cit. on p. 102).

[19] M. Benzi. "Preconditioning Techniques for Large Linear Systems: A Survey". In: *J. Comp. Phys.* 182 (2002), pp. 418–477 (cit. on pp. 16, 76, 155, 179).

[20] M. Benzi and D. Bertaccini. "Approximate inverse preconditioning for shifted linear systems". In: *BIT* 43.2 (2003), pp. 231–244 (cit. on pp. 15, 138, 144, 156, 179, 180).

[21] M. Benzi, J. K. Cullum, and M. Tüma. "Robust approximate inverse preconditioning for the conjugate gradient method". In: *SIAM J. Sci. Comput.* 22.4 (2000), pp. 1318–1332 (cit. on pp. 78, 89, 91, 96).

[22] M. Benzi, C. Meyer, and M. Tüma. "A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method". In: *SIAM J. Sci. Comput.* 17.5 (1996), pp. 1135–1149 (cit. on pp. 78, 89, 91, 93, 96).

[23] M. Benzi and N. Razouk. "Decay bounds and $O(n)$ algorithms for approximating functions of sparse matrices". In: *Electron. Trans. Numer. Anal.* 28 (2007), pp. 16–39 (cit. on p. 130).

[24] M. Benzi and M. Tüma. "A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems". In: *SIAM J. Sci. Comput.* 19.3 (1998), pp. 968–994 (cit. on pp. 78, 89, 91).

[25] M. Benzi and M. Tüma. "Orderings for factorized sparse approximate inverse preconditioners". In: *Siam J. Sci. Comput.* 21.5 (2000), pp. 1851–1868 (cit. on pp. 126, 127).

[26] S. Bernstein. *Leçons sur les propriétés extrémales et la meilleure approximation des fonctions analytiques d'une variable réelle*. Paris, 1926 (cit. on p. 81).

[27] D. Bertaccini. "P–circulant preconditioners and the systems of the ODE codes". In: *Iterative methods in scientific computation IV*. Ed. by D. Kincaid. Ed. by A. Elster. Vol. 5. IMACS series in computational and applied mathematics. IMACS, 1999, pp. 179–193 (cit. on pp. 16, 230, 231).

[28] D. Bertaccini. "A circulant preconditioner for the systems of LMF–based ODE codes". In: *SIAM J. Sci. Comp.* 22.3 (2000), pp. 767–786 (cit. on pp. 16, 65, 102, 227, 229, 230, 232).

[29] D. Bertaccini. "Reliable preconditioned iterative linear solvers for some numerical integrators". In: *Numer. Linear Algebra Appl.* 8.2 (2001), pp. 111–125 (cit. on pp. 16, 229–231).

[30] D. Bertaccini. "The spectrum of circulant–like preconditioners for some general linear multistep formulas for linear boundary value problems". In: *SIAM J. Numer. Anal.* 40.5 (2002), pp. 1798–1822 (cit. on pp. 16, 227, 231).

[31] D. Bertaccini. "Efficient preconditioning for sequences of parametric complex symmetric linear systems". In: *Electron. Trans. Numer. Anal.* 18 (2004), pp. 49–64 (cit. on pp. 15, 106, 108, 122, 138, 144, 156, 179, 180).

[32] D. Bertaccini, M. Donatelli, F. Durastante, and S. Serra-Capizzano. "Optimizing a multigrid Runge–Kutta smoother for variable–coefficient convection–diffusion equations". In: *Linear Algebra Appl.* 533 (2017), pp. 507–535. ISSN: 0024-3795 (cit. on pp. 15, 195, 199–201).

[33] D. Bertaccini and F. Durastante. "Interpolating preconditioners for the solution of sequence of linear systems". In: *Comput. Math. Appl.* 72.4 (2016), pp. 1118–1130 (cit. on pp. 15, 103, 107, 108, 144, 180, 191).

[34] D. Bertaccini and F. Durastante. "Limited memory block preconditioners for fast solution of fractional PDEs". In: *J. Comp. Phys* x (2017). Submitted, pp. xx–xx (cit. on pp. 16, 223, 225, 233, 234).

[35] D. Bertaccini and F. Durastante. "Solving mixed classical and fractional partial differential equations using short–memory principle and approximate inverses". In: *Numer. Algorithms* 74.4 (2017), pp. 1061–1082. ISSN: 1572-9265. DOI: 10.1007/s11075-016-0186-8 (cit. on pp. 15, 143, 179, 186, 191).

[36] D. Bertaccini and S. Filippone. "Sparse approximate inverse preconditioners on high performance GPU platforms". In: *Comput. Math. Appl.* 71.3 (2016), pp. 693–711 (cit. on pp. 76, 78, 83, 86–89, 92–94, 96, 98, 104, 111, 130, 142, 144, 157, 179).

[37] D. Bertaccini, G. H. Golub, and S. Serra-Capizzano. "Spectral analysis of a preconditioned iterative method for the convection-diffusion equation". In: *SIAM J. Matr. Anal. Appl.* 29.1 (2007), pp. 260–278 (cit. on p. 195).

[38] D. Bertaccini and M. K. Ng. "Skew–circulant preconditioners for systems of LMF–based ODE codes". In: *International Conference on Numerical Analysis and Its Applications*. Springer. 2000, pp. 93–101 (cit. on pp. 16, 231).

[39] D. Bertaccini and M. K. Ng. "Band-Toeplitz Preconditioned GMRES Iterations for Time-Dependent PDEs". In: *BIT* 43 (2003), pp. 901–914 (cit. on p. 39).

[40] D. Bertaccini and M. K. Ng. "Block $\{\omega\}$-circulant preconditioners for the systems of differential equations". In: *Calcolo* 40.2 (2003), pp. 71–90 (cit. on pp. 65, 229, 231, 232, 235, 236).

[41] D. Bertaccini, M. Popolizio, and F. Durastante. "Adaptive updating techniques for the approximation of functions of large matrices". In: *arXiv preprint arXiv:1709.06351* (2017) (cit. on pp. 15, 121, 125, 126, 128, 129).

[42] D. Bertaccini and F. Sgallari. "Updating preconditioners for nonlinear deblurring and denoising image restoration". In: *Appl. Numer. Math.* 60 (2010), pp. 994–1006 (cit. on p. 102).

[43] J. Bey and G. Wittum. "Downwind numbering: Robust multigrid for convection-diffusion problems". In: *Appl. Numer. Math.* 23.1 (1997), pp. 177–192 (cit. on p. 195).

[44] D. Bianchi, A. Buccini, M. Donatelli, and S. Serra-Capizzano. "Iterated fractional Tikhonov regularization". In: *Inverse Problems* 31.5 (2015), p. 055005 (cit. on p. 65).

[45] P. Birken. "Optimizing Runge-Kutta smoothers for unsteady flow problems". In: *Electron. Trans. Numer. Anal.* 39 (2012), pp. 298–312 (cit. on pp. 195, 196, 202, 205, 206, 211, 214–219).

[46] P. Birken, J. Duintjer Tebbens, A. Meister, and M. Tüma. "Preconditioner updates applied to CFD model problems". In: *Appl. Numer. Math.* 58.11 (2008), pp. 1628–1641 (cit. on p. 102).

[47] A. Böttcher and S. M. Grudsky. *Toeplitz matrices, asymptotic linear algebra, and functional analysis*. Basel–Boston–Berlin: Birkhäuser Verlag, 2000 (cit. on p. 195).

[48] A. Böttcher and S. M. Grudsky. *Spectral properties of banded Toeplitz matrices*. Siam, 2005 (cit. on p. 61).

[49] A. Böttcher and S. M. Grudsky. *Toeplitz matrices, asymptotic linear algebra, and functional analysis*. Birkhäuser, 2012 (cit. on p. 61).

[50] A. Böttcher and B. Silbermann. *Introduction to large truncated Toeplitz matrices*. Springer Science & Business Media, 2012 (cit. on p. 61).

[51] D. Braess and W. Hackbusch. "A new convergence proof for the multigrid method including the V-cycle". In: *SIAM J. Numer. Anal.* 20.5 (1983), pp. 967–975 (cit. on p. 75).

[52] A. Brandt. "Algebraic multigrid theory: The symmetric case". In: *Appl. Math. Comput.* 19.1-4 (1986), pp. 23–56 (cit. on p. 74).

[53] R. K. Brayton, S. Director, G. D. Hachtel, and L. M. Vidigal. "A new algorithm for statistical circuit design based on quasi-Newton methods and function splitting". In: *IEEE Trans. Circuits Syst. I. Regul. Pap.* 26.9 (1979), pp. 784–794 (cit. on p. 213).

[54] R. P. Brent, F. G. Gustavson, and D. Y. Yun. "Fast solution of Toeplitz systems of equations and computation of Padé approximants". In: *J. Algorithms* 1.3 (1980), pp. 259–295 (cit. on p. 61).

[55] C. Brezinski and M. Redivo Zaglia. "Treatment of near–breakdown in the CGS algorithm". In: *Numer. Algorithms* 7.1 (1994), pp. 33–73 (cit. on p. 51).

[56] C. Brezinski and M. Redivo Zaglia. "Look–ahead in Bi-CGSTAB and other product methods for linear systems". In: *BIT* 35.2 (1995), pp. 169–201 (cit. on p. 50).

[57] C. Brezinski, M Redivo Zaglia, and H. Sadok. "Avoiding breakdown and near-breakdown in Lanczos type algorithms". In: *Numer. Algorithms* 1.2 (1991), pp. 261–284 (cit. on p. 48).

[58] C. Brezinski, M. Redivo Zaglia, and H. Sadok. "A breakdown–free Lanczos type algorithm for solving linear systems". In: *Numer. Math.* 63.1 (1992), pp. 29–38 (cit. on p. 48).

[59] C. Brezinski, M. Redivo Zaglia, and H. Sadok. "New look–ahead Lanczos–type algorithms for linear systems". In: *Numer. Mathematik* 83.1 (1999), pp. 53–85 (cit. on p. 50).

[60] R. Bridson and W.-P. Tang. "Ordering, anisotropy, and factored sparse approximate inverses". In: *SIAM J. Sci. Comp.* 21.3 (1999), pp. 867–882 (cit. on p. 89).

[61] R. Bridson and W.-P. Tang. "Refining an approximate inverse". In: *J. Comput. Appl. Math.* 123.1 (2000), pp. 293–306 (cit. on pp. 89, 154, 179).

[62] R. Bridson and W.-P. Tang. "Multiresolution approximate inverse preconditioners". In: *SIAM J. Sci. Comput.* 23.2 (2001), pp. 463–479 (cit. on p. 179).

[63] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. SIAM, 2000 (cit. on p. 70).

[64] F. E. Browder. "Nonlinear monotone operators and convex sets in Banach spaces". In: *Bull. Amer. Math. Soc.* 71.5 (1965), pp. 780–785 (cit. on p. 175).

[65] L. Brugnano and D. Trigiante. *Solving Differential Equations by Multistep Initial and Boundary Value Methods*. Stability and Control: Theory, Methods and Applications. Taylor & Francis, 1998. ISBN: 9789056991074 (cit. on pp. 228, 229).

[66] L. Brugnano and D. Trigiante. *Solving Differential Problems by Multistep Initial and Boundary Value Methods*. Gordon and Breach Science Publishers, Amsterdam, 1998 (cit. on pp. 225–228).

[67] G. Bruun. "z–transform DFT filters and FFT's". In: *IEEE T. Acoust. Speech* 26.1 (1978), pp. 56–63 (cit. on p. 64).

[68] W. Bu, Y. Tang, and J. Yang. "Galerkin finite element method for two–dimensional Riesz space fractional diffusion equations". In: *J. Comp. Phys.* 276 (2014), pp. 26–38 (cit. on pp. 173, 174).

[69] M.-C. Cai and X.-Q. Jin. "BCCB preconditioners for solving linear systems from delay differential equations". In: *Comput. Math. Appl.* 50.1 (2005), pp. 281–288 (cit. on p. 65).

[70] C. Calgaro, J.-P. Chehab, and Y. Saad. "Incremental incomplete LU factorizations with applications". In: *Numer. Linear Algebra Appl.* 17.5 (2010), pp. 811–837. ISSN: 1099-1506. DOI: 10.1002/nla.756. URL: http://dx.doi.org/10.1002/nla.756 (cit. on p. 102).

[71] C. Canuto, V. Simoncini, and M. Verani. "On the decay of the inverse of matrices that are sum of Kronecker products". In: *Linear Algebra Appl.* 452 (2014), pp. 21–39 (cit. on pp. 82, 153).

[72] M. Caputo. "Linear models of dissipation whose Q is almost frequency independent—II". In: *Geophys. J. Roy. Astron. Soc.* 13.5 (1967), pp. 529–539 (cit. on p. 252).

[73] A. J. Carpenter, A. Ruttan, and R. S. Varga. "Extended numerical computations on the 1/9 conjecture in rational approximation theory". In: *Rational Approximation and Interpolation*. Ed. by P. R. Graves-Morris, E. B. Saff, and R. S. Varga. Vol. 1105. Lecture Notes in Mathematics. Berlin: Springer-Verlag, 1984, pp. 383–411 (cit. on p. 123).

[74] D. Caughey and A. Jameson. "How many steps are required to solve the Euler equations of steady compressible flow: In search of a fast solution algorithm". In: *AIAA Journal* (2001), pp. 2001–2673 (cit. on p. 195).

[75] C. Çelik and M. Duman. "Crank–Nicolson method for the fractional diffusion equation with the Riesz fractional derivative". In: *J. Comput Phys.* 231.4 (2012), pp. 1743–1750 (cit. on pp. 143, 149, 267).

[76] R. Chan and T. F. Chan. "Circulant preconditioners for elliptic problems". In: *J. Numer. Lin. Alg. and Appl.* 1 (1992), pp. 77–101 (cit. on p. 65).

[77] R. Chan, M. Ng, and X.-Q. Jin. "Strang–type preconditioner for systems of LMF–based ODE codes". In: *IMA J. Numer. Anal.* 21.2 (2001), pp. 451–62 (cit. on p. 230).

[78] R. H. Chan, Q.-S. Chang, and H.-W. Sun. "Multigrid method for ill–conditioned symmetric Toeplitz systems". In: *SIAM J. Sci. Comput.* 19.2 (1998), pp. 516–529 (cit. on p. 75).

[79] Z. Chen, R. E. Ewing, R. D. Lazarov, S. Maliassov, and Y. A. Kuznetsov. "Multilevel preconditioners for mixed methods for second order elliptic problems". In: *Numer. Linear Algebra Appl.* 3.5 (1996), pp. 427–453 (cit. on p. 76).

[80] W. K. Ching. *Iterative methods for queuing and manufacturing systems*. Springer Science & Business Media, 2013 (cit. on p. 61).

[81] S. Cipolla and F. Durastante. "Fractional PDE constrained optimization: An optimize-then-discretize approach with L-BFGS and approximate inverse preconditioning". In: *Appl. Numer. Math.* 123.Supplement C (2018), pp. 43–57. ISSN: 0168-9274. DOI: https://doi.org/10.1016/j.apnum.2017.09.001 (cit. on pp. 15, 167, 169–172, 174, 181).

[82] W. J. Cody, G. Meinardus, and R. S. Varga. "Chebyshev rational approximations to e^{-x} in $[0, +\infty)$ and applications to heat-conduction problems". In: *J. Approx. Theory* 2.1 (March 1969), pp. 50–65 (cit. on p. 123).

[83] J. W. Cooley and J. W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Math. Comput.* 19.90 (1965), pp. 297–301 (cit. on p. 64).

[84] S. Dalton, N. Bell, L. Olson, and M. Garland. *Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations*. Version 0.5.0. 2014. URL: http://cusplibrary.github.io/ (cit. on pp. 144, 156, 169, 179, 181).

[85] P. D'Ambra, D. D. Serafino, and S. Filippone. "MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95". In: *ACM Trans. Math. Softw.* 37.3 (2010), p. 30 (cit. on p. 144).

[86] T. A. Davis and Y. Hu. "The University of Florida sparse matrix collection". In: *ACM Trans. Math. Software* 38.1 (2011), p. 1 (cit. on pp. 92, 136–138).

[87] D. De Cecchis, H. López, and B. Molina. "FGMRES preconditioning by symmetric/skew–symmetric decomposition of generalized Stokes problems". In: *Math. Comput. in Simulat.* 79.6 (2009), pp. 1862–1877 (cit. on p. 45).

[88] L. Debnath. "A brief historical introduction to fractional calculus". In: *International Journal of Mathematical Education in Science and Technology* 35.4 (2004), pp. 487–501 (cit. on pp. 247–249).

[89] S. Demko, W. F. Moss, and P. W. Smith. "Decay rates for inverses of band matrices". In: *Math. Comp.* 43.168 (1984), pp. 491–499 (cit. on pp. 79, 81, 125, 150).

[90] W. Deng. "Short memory principle and a predictor–corrector approach for fractional differential equations". In: *J. Comput. Appl. Math.* 206.1 (2007), pp. 174–188 (cit. on pp. 143, 150).

[91] J. Dennis Jr. and J. Moré. "Quasi–Newton Methods, Motivation and Theory". In: *SIAM Rev.* 19.1 (1977), pp. 46–89. DOI: 10.1137/1019005. eprint: http://dx.doi.org/10.1137/1019005. URL: http://dx.doi.org/10.1137/1019005 (cit. on p. 102).

[92] E. Di Nezza, G. Palatucci, and E. Valdinoci. "Hitchhiker's guide to the fractional Sobolev spaces". In: *Bull. Sci. Math.* 136.5 (2012), pp. 521–573 (cit. on pp. 170, 175, 257, 260).

[93] K. Diethelm. *The Analysis of Fractional Differential Equations: An Application–Oriented Exposition Using Differential Operators of Caputo Type*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 2010. ISBN: 9783642145735 (cit. on p. 251).

[94] H. Ding, C. Li, and Y. Chen. "High–order algorithms for Riesz derivative and their applications (II)". In: *J. Comput Phys.* 293 (2015), pp. 218–237 (cit. on p. 149).

[95] V. A. Dobrev, R. D. Lazarov, P. S. Vassilevski, and L. T. Zikatanov. "Two–level preconditioning of discontinuous Galerkin approximations of second–order elliptic equations". In: *Numer. Linear Algebra Appl.* 13.9 (2006), p. 753 (cit. on p. 76).

[96] S. Dolgov, J. W. Pearson, D. V. Savostyanov, and M. Stoll. "Fast tensor product solvers for optimization problems with fractional differential equations as constraints". In: *Appl. Math. Comput.* 273 (2016), pp. 604–623 (cit. on p. 167).

[97] M. Donatelli, A. Dorostkar, M. Mazza, M. Neytcheva, and S. Serra-Capizzano. "Function-based block multigrid strategy for a two-dimensional linear elasticity-type problem". In: *Comput. Math. Appl.* 74.5 (2017), pp. 1015–1028. DOI: 10.1016/j.camwa.2017.05.024 (cit. on p. 75).

[98] M. Donatelli, C. Garoni, C. Manni, S. Serra-Capizzano, and H. Speleers. "Spectral analysis and spectral symbol of matrices in isogeometric collocation methods". In: *Math. Comput.* 85.300 (2016), pp. 1639–1680 (cit. on p. 196).

[99] M. Donatelli, C. Garoni, C. Manni, S. Serra-Capizzano, and H. Speleers. "Symbol-based multigrid methods for Galerkin B-spline isogeometric analysis". In: *SIAM J. Numer. Anal.* 55.1 (2017), pp. 31–62 (cit. on p. 75).

[100] M. Donatelli, M. Mazza, and S. Serra-Capizzano. "Spectral analysis and structure preserving preconditioners for fractional diffusion equations". In: *J. Comp. Phys.* 307 (2016), pp. 262–279 (cit. on pp. 148, 180, 224, 225, 265).

[101] M. Donatelli, M. Molteni, V. Pennati, and S. Serra-Capizzano. "Multigrid methods for cubic spline solution of two point (and 2D) boundary value problems". In: *App. Numer. Math.* 104 (2016), pp. 15–29 (cit. on p. 196).

[102] C. C. Douglas. "Multi–grid algorithms with applications to elliptic boundary value problems". In: *SIAM J. Numer. Anal.* 21.2 (1984), pp. 236–254 (cit. on p. 75).

[103] I. S. Du, R. G. Grimes, and J. G. Lewis. *Users' guide for the Harwell-Boeing sparse matrix collection (Release I)*. Tech. rep. Report RAL-92-086, Atlas Centre Rutherford Appleton Laboratory, Didcot Oxon (UK), 1992. URL: http://www.cs.colostate.edu/~mroberts/toolbox/c++/sparseMatrix/hbsmc.pdf (cit. on p. 91).

[104] A. C. N. van Duin. "Scalable Parallel Preconditioning with the Sparse Approximate Inverse of Triangular Matrices". In: *SIAM J. Matrix Anal. Appl.* 20 (4 1999), pp. 987–0. DOI: 10.1137/s0895479897317788 (cit. on pp. 78, 83, 89).

[105] D. M. Dunlavy, T. G. Kolda, and E. Acar. *Poblano v1.0: A Matlab Toolbox for Gradient-Based Optimization*. Tech. rep. SAND2010-1422. Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Mar. 2010 (cit. on pp. 169, 180, 181).

[106] F. Durastante. "Interpolant Update of Preconditioners for Sequences of Large Linear Systems". In: *Mathematical Methods, Computational Techniques and Intelligent Systems (MAMECTIS '15)*. Vol. 41. WSEAS Press. 2015, pp. 40–47 (cit. on pp. 15, 103, 106, 111, 144).

[107] H. Dym and H. McKean. *Fourier Series and Integrals*. Probability and mathematical statistics. Academic Press, 1972. ISBN: 9780122264511 (cit. on p. 61).

[108] H. C. Elman. "A Stability Analysis of Incomplete LU Factorizations". In: *Math. Comp.* 47.175 (1986), pp. 191–217. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2008089 (cit. on p. 83).

[109] J. B. Elsner and A. A. Tsonis. *Singular spectrum analysis: a new tool in time series analysis*. Springer Science & Business Media, 2013 (cit. on p. 61).

[110] V. J. Ervin and J. P. Roop. "Variational formulation for the stationary fractional advection dispersion equation". In: *Numer. Methods Partial Differ. Equ.* 22.3 (2006), pp. 558–576 (cit. on pp. 169–171, 260, 261).

[111] J. van den Eshof and M. Hochbruck. "Preconditioning Lanczos approximations to the matrix exponential". In: *SIAM J. Sci. Comput.* 27.4 (2006), pp. 1438–1457 (cit. on pp. 122, 140).

[112] L. Euler. "De Progressionibus Transcendentibus, sev quarum Termini Generales Algebraice Dari Nequevent". In: *Commentarii academiae scientiarum Petropolitanae* (5 1738), pp. 36–57 (cit. on p. 247).

[113] S. Filippone and M. Colajanni. "PSBLAS: A library for parallel linear algebra computation on sparse matrices". In: *ACM Trans. on Math. Software* 26.4 (2000), pp. 527–550 (cit. on p. 144).

[114] G Fiorentino and S. Serra-Capizzano. "Multigrid methods for Toeplitz matrices". In: *Calcolo* 28.3 (1991), pp. 283–305 (cit. on p. 75).

[115] G. Fiorentino and S. Serra-Capizzano. "Multigrid methods for symmetric positive definite block Toeplitz matrices with nonnegative generating functions". In: *SIAM J. Sci. Comput.* 17.5 (1996), pp. 1068–1081 (cit. on p. 75).

[116] R. Fischer and T. Huckle. "Using ω-circulant matrices for the preconditioning of Toeplitz systems". In: *Selçuk J. Appl. Math* 4 (2003), pp. 71–88 (cit. on p. 233).

[117] B. Fornberg. “Classroom note: Calculation of weights in finite difference formulas”. In: *SIAM review* 40.3 (1998), pp. 685–691 (cit. on p. 147).

[118] J. Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822 (cit. on p. 248).

[119] L. Fox, H. D. Huskey, and J. H. Wilkinson. “Notes on the solution of algebraic linear simultaneous equations”. In: *Quart. J. Mech. and Applied Math.* 1 (1948), pp. 149–173 (cit. on p. 89).

[120] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. “An implementation of the Look-Ahead Lanczos algorithm”. In: *SIAM J. Sci. Stat. Comput.* 14.2 (1993), pp. 470–482 (cit. on p. 50).

[121] M. Frigo and S. Johnson. “The Design and Implementation of {FFTW3}”. In: *Proceedings of the IEEE* 93.2 (2005). Special issue on ”Program Generation, Optimization, and Platform Adaptation”, pp. 216–231 (cit. on p. 64).

[122] C. Garoni. “Topological foundations of an asymptotic approximation theory for sequences of matrices with increasing size”. In: *Linear Algebra Appl.* 513 (2017), pp. 324–341 (cit. on p. 68).

[123] C. Garoni, C. Manni, S. Serra-Capizzano, D. Sesana, and H. Speleers. “Spectral analysis and spectral symbol of matrices in isogeometric Galerkin methods”. In: *Math. Comput.* 86.305 (2017), pp. 1343–1373 (cit. on p. 196).

[124] C. Garoni and S. Serra-Capizzano. “The theory of locally Toeplitz sequences: a review, an extension, and a few representative applications”. In: *Bol. Soc. Mat. Mex.* 22.2 (2016), pp. 529–565 (cit. on p. 67).

[125] C. Garoni and S. Serra-Capizzano. “The theory of generalized locally Toeplitz sequences: a review, an extension, and a few representative applications”. In: *Large Truncated Toeplitz Matrices, Toeplitz Operators, and Related Topics*. Springer, 2017, pp. 353–394 (cit. on p. 67).

[126] C. Garoni and S. Serra-Capizzano. *The theory of Generalized Locally Toeplitz sequences: theory and applications - Vol I*. Berlin: Springer Monographs. ISBN: 978-3-319-53678-1: http://www.springer.com/gp/book/9783319536781, 2017 (cit. on pp. 16, 68, 69, 205).

[127] C. Garoni, S. Serra-Capizzano, and D. Sesana. "Tools for Determining the Asymptotic Spectral Distribution of non-Hermitian Perturbations of Hermitian Matrix-Sequences and Applications". English. In: *Integral Equations Operator Theory* 81.2 (2015), pp. 213–225. ISSN: 0378-620X. DOI: 10.1007/s00020-014-2157-6. URL: http://dx.doi.org/10.1007/s00020-014-2157-6 (cit. on p. 205).

[128] R. Garrappa and M. Popolizio. "On the use of matrix functions for fractional partial differential equations". In: *Math. Comput. Simulation* 81.5 (2011), pp. 1045–1056. ISSN: 0378-4754 (cit. on p. 122).

[129] J. R. Gilbert. "Predicting structure in sparse matrix computations". In: *SIAM J. Matrix Anal. Appl.* 15.1 (1994), pp. 62–79 (cit. on p. 21).

[130] I Gohberg and A Semencul. "On the inversion of finite Toeplitz matrices and their continuous analogs". In: *Mat. issled* 2 (1972), pp. 201–233 (cit. on p. 67).

[131] A Golbabai and K Sayevand. "Analytical modelling of fractional advection–dispersion equation defined in a bounded space domain". In: *Math. Comput. Model.* 53.9 (2011), pp. 1708–1718 (cit. on p. 257).

[132] L. Golinskii and S. Serra-Capizzano. "The asymptotic properties of the spectrum of nonsymmetrically perturbed Jacobi matrix sequences". In: *J. Approx. Theory* 144.1 (2007), pp. 84 –102. ISSN: 0021-9045. DOI: http://dx.doi.org/10.1016/j.jat.2006.05.002. URL: http://www.sciencedirect.com/science/article/pii/S0021904506000864 (cit. on pp. 32, 205).

[133] G. H. Golub and C. Greif. "On solving block–structured indefinite linear systems". In: *SIAM J. Sci. Comput.* 24.6 (2003), pp. 2076–2092 (cit. on p. 30).

[134] G. H. Golub and C. F. Van Loan. *Matrix computations*. 3rd ed. Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press, 1996 (cit. on pp. 21, 25, 35, 90).

[135] I. J. Good. "The interaction algorithm and practical Fourier analysis". In: *J. Royal Statist. Soc.* 20.2 (1958), pp. 361–372 (cit. on p. 64).

[136] S. Goossens, K. Tan, and D. Roose. "An efficient FGMRES solver for the shallow water equations based on domain decomposition". In: *Domain Decomposition Methods in Sciences and Engineering*. 1998, pp. 350–358 (cit. on p. 45).

[137] A. Greenbaum. "Analysis of a multigrid method as an iterative technique for solving linear systems". In: *SIAM J. Numer. Anal.* 21.3 (1984), pp. 473–485 (cit. on p. 75).

[138] A. Greenbaum. *Iterative methods for solving linear systems*. Vol. 17. Siam, 1997 (cit. on pp. 29, 36, 40).

[139] A. Greenbaum, V. Pták, and Z. Strakoš. "Any Nonincreasing Convergence Curve is Possible for GMRES". In: *SIAM J. Matrix Anal. Appl.* 17.3 (1996), pp. 465–469. DOI: 10.1137/S0895479894275030. eprint: http://dx.doi.org/10.1137/S0895479894275030. URL: http://dx.doi.org/10.1137/S0895479894275030 (cit. on p. 40).

[140] A. Greenbaum and Z. Strakoš. *Matrices that generate the same Krylov residual spaces*. Springer, 1994 (cit. on p. 40).

[141] U. Grenander and G. Szegö. *Toeplitz forms and their applications*. Vol. 321. University of California Press, 2001 (cit. on pp. 61–63, 67).

[142] A. K. Grünwald. "Über "begrenzte" Derivationen und deren Anwendung". In: *Z. Math. Phys.* 12 (1867), pp. 441–480 (cit. on pp. 253, 254).

[143] X.-M. Gu, T.-Z. Huang, X.-L. Zhao, H.-B. Li, and L. Li. "Strang–type preconditioners for solving fractional diffusion equations by boundary value methods". In: *J. Comput. Appl. Math.* 277 (2015), pp. 73–86 (cit. on pp. 228–231).

[144] H. Guo, G. A. Sitton, and C. S. Burrus. "The quick Fourier transform: an FFT based on symmetries". In: *IEEE Trans. Signal Process.* 46.2 (1998), pp. 335–341 (cit. on p. 64).

[145] M. M. Gupta and J. Zhang. "High accuracy multigrid solution of the 3D convection–diffusion equation". In: *Appl. Math. Comput.* 113.2 (2000), pp. 249–274 (cit. on p. 195).

[146] M. H. Gutknecht. "Variants of BICGSTAB for matrices with complex spectrum". In: *SIAM J. Sci. Comput.* 14.5 (1993), pp. 1020–1033 (cit. on p. 161).

[147] W. Hackbusch. "Convergence of multigrid iterations applied to difference equations". In: *Math. Comp.* 34.150 (1980), pp. 425–440 (cit. on p. 75).

[148] R. Haelterman, J. Vierendeels, and D. V. Heule. "A generalization of the Runge-Kutta iteration". In: *J. Comput. Appl. Math.* 224.1 (2009), pp. 152 –167. ISSN: 0377-0427. DOI: http://dx.doi.org/10.1016/j.cam.2008.04.021. URL: http://www.sciencedirect.com/science/article/pii/S0377042708001866 (cit. on pp. 196, 208, 212).

[149] N. Hale, N. J. Higham, and L. N. Trefethen. "Computing A^α, $\log(A)$, and related matrix functions by contour integrals". In: *SIAM J. Numer. Anal.* 46 (2008), pp. 2505–2523 (cit. on pp. 121–123).

[150] M. Hanke and J. G. Nagy. "Toeplitz approximate inverse preconditioner for banded Toeplitz matrices". In: *Numer. Algorithms* 7.2 (1994), pp. 183–199 (cit. on p. 233).

[151] P. C. Hansen. "Deconvolution and regularization with Toeplitz matrices". In: *Numer. Algorithms* 29.4 (2002), pp. 323–378 (cit. on p. 65).

[152] P. C. Hansen, J. G. Nagy, and D. P. O'leary. *Deblurring images: matrices, spectra, and filtering*. Vol. 3. Siam, 2006 (cit. on p. 61).

[153] G. H. Hardy and J. E. Littlewood. "Some properties of fractional integrals. I." In: *Mathematische Zeitschrift* 27.1 (1928), pp. 565–606 (cit. on p. 248).

[154] G. H. Hardy and J. E. Littlewood. "Some properties of fractional integrals. II". In: *Mathematische Zeitschrift* 34.1 (1932), pp. 403–439 (cit. on p. 248).

[155] O. Heaviside. *Electrical papers*. Vol. 2. Cambridge University Press, 2011 (cit. on p. 248).

[156] F. Hecht. "New development in FreeFem++". In: *J. Numer. Math.* 20.3-4 (2012), pp. 251–265. ISSN: 1570-2820 (cit. on pp. 113, 116).

[157] M. R. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems". In: *J. Res. Nat. Bur. Stand.* 49 (1952), pp. 409–436 (cit. on p. 25).

[158] N. J. Higham. *Functions of matrices. Theory and computation*. Philadelphia, PA: SIAM, 2008, pp. xxiv+425. ISBN: 0-12-558840-2 (cit. on pp. 121–123, 132, 141).

[159] R. Hilfer. *Applications of fractional calculus in physics*. World Scientific, 2000 (cit. on p. 255).

[160] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE constraints*. Vol. 23. Springer Science & Business Media, 2008 (cit. on pp. 167, 171).

[161] M. Hochbruck and C. Lubich. "On Krylov subspace approximations to the matrix exponential operator". In: *SIAM J. Numer. Anal.* 34.5 (1997), pp. 1911–1925 (cit. on pp. 122, 140).

[162] J. E. Hopcroft and J. D. Ullman. *Data structures and algorithms*. Vol. 175. Addison-Wesley Boston, MA, USA: 1983 (cit. on p. 87).

[163] T. Huckle. "Approximate sparsity patterns for the inverse of a matrix and preconditioning". In: *Appl. Numer. Math.* 30.2 (1999), pp. 291–303 (cit. on p. 78).

[164] T. Huckle. "Factorized sparse approximate inverses for preconditioning and smoothing". In: *Selcuk J. Appl. Math* 1 (2000), p. 63 (cit. on p. 78).

[165] T. Huckle. "Factorized sparse approximate inverses for preconditioning". In: *J Supercomput* 25.2 (2003), pp. 109–117 (cit. on p. 78).

[166] S. Jaffard. "Propriétés des matrices «bien localisées» près de leur diagonale et quelques applications". In: *Annales de l'IHP Analyse non linéaire*. Vol. 7.5. 1990, pp. 461–476 (cit. on pp. 83, 151).

[167] T. Kailath and J Chun. "Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices". In: *SIAM J. Matrix Anal. Appl.* 15.1 (1994), pp. 114–128 (cit. on p. 65).

[168] C. Kelley. *Solving Nonlinear Equations with Newton's Method*. 1st. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718898. URL: http://epubs.siam.org/doi/abs/10.1137/1.9780898718898 (cit. on p. 186).

[169] C. Kenney and A. J. Laub. "Padé error estimates for the logarithm of a matrix". In: *Internat. J. Control* 50.3 (1989), pp. 707–730 (cit. on p. 123).

[170] S. Kharchenko, L. Y. Kolotilina, A. Nikishin, and A. Y. Yeremin. "A robust AINV–type method for constructing sparse approximate inverse preconditioners in factored form". In: *Numer. Linear Algebra Appl.* 8.3 (2001), pp. 165–179 (cit. on p. 89).

[171] L. Knizhnerman and V. Simoncini. "A new investigation of the extended Krylov subspace method for matrix function evaluations". In: *Numerical Linear Algebra with Applications* 17.4 (2010), pp. 615–638 (cit. on pp. 122, 140).

[172] D. Knoll and D. Keyes. "Jacobian–free Newton–Krylov methods: a survey of approaches and applications". In: *J. Comp. Phys.* 193 (2 2004), pp. 357–397. DOI: 10.1016/j.jcp.2003.08.010 (cit. on p. 102).

[173] T. Körner. *Fourier Analysis*. Cambridge University Press, 1989. ISBN: 9780521389914 (cit. on p. 61).

[174] I. Krishtal, T. Strohmer, and T. Wertz. "Localization of matrix factorizations". In: *Found. Comput. Math.* 15.4 (2015), pp. 931–951 (cit. on p. 151).

[175] S. F. Lacroix. *Traité du calcul différentiel et du calcul intégral*. 2nd ed. Vol. 3. Courcier, 1819 (cit. on pp. 248, 252).

[176] J. L. Lagrange. *Sur une nouvelle espèce de calcul rélatif à la différentiation & à l'intégration des quantités variables*. Académie royale des sciences et belles lettres, 1772 (cit. on p. 248).

[177] V. Lampret. "Estimating the sequence of real binomial coefficients". In: *J. Ineq. Pure and Appl. Math.* 7.5 (2006) (cit. on p. 147).

[178] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *J. Res. Nat. Bur. Stand.* 45 (1950), pp. 255–282 (cit. on pp. 24, 26).

[179] C. Lanczos. "Solution of systems of linear equations by minized iterations". In: *J. Res. Nat. Bur. Stand.* 49 (1952), pp. 33–53 (cit. on p. 47).

[180] P. S. de Laplace. *Théorie analytique des probabilités*. Vol. 7. Courcier, 1820 (cit. on p. 248).

[181] A. J. Laub. *Matrix analysis for scientists and engineers*. SIAM, 2005 (cit. on p. 203).

[182] S.-L. Lei and H.-W. Sun. "A circulant preconditioner for fractional diffusion equations". In: *J. Comp. Phys.* 242 (2013), pp. 715–725 (cit. on p. 180).

[183] F. Lemeire. "Bounds for condition numbers of triangular and trapezoid matrices". English. In: *BIT* 15.1 (1975), pp. 58–64. ISSN: 0006-3835. DOI: 10.1007/BF01932996. URL: http://dx.doi.org/10.1007/BF01932996 (cit. on p. 107).

[184] A. Letnikov. "Theory of differentiation of fractional order". In: *Mat. Sb* 3.1 (1868) (cit. on pp. 253, 254).

[185] G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich. *Trends in PDE constrained optimization*. Vol. 165. Springer, 2014 (cit. on p. 167).

[186] R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, 2007. ISBN: 9780898716290. URL: http://books.google.it/books?id=qsvmsXe8Ug4C (cit. on pp. 61, 82, 102).

[187] C. Li, A. Chen, and J. Ye. "Numerical approaches to fractional calculus and fractional ordinary differential equation". In: *J. Comput Phys.* 230.9 (2011), pp. 3352–3368 (cit. on p. 143).

[188] C. Li and F. Zeng. *Numerical methods for fractional calculus*. Vol. 24. CRC Press, 2015 (cit. on p. 143).

[189] N. Li and Y. Saad. "Crout versions of ILU factorization with pivoting for sparse symmetric matrices". In: *Electron. Trans. Numer. Anal.* 20 (2005), pp. 75–85 (cit. on p. 83).

[190] F. Lin, X. Jin, and S. Lei. "Strang–type preconditioners for solving linear systems from delay differential equations". In: *BIT* 43.1 (2003), pp. 139–152 (cit. on p. 65).

[191] J. Liouville. "Mémoire sur quelques questions de géométrie et de mécanique, et sur un nouveau genre de calcul pour résoudre ces questions". In: *J. Ècole Polytech.* (13 1832), pp. 1–69 (cit. on pp. 248, 249).

[192] J. Liouville. "Note sur une formule pour les diffèrentielles á indices quelconques, á l'occasion d'un Mèmoire de M. Tortolini." fre. In: *Journal de Mathématiques Pures et Appliquées* (1855), pp. 115–120. URL: http://eudml.org/doc/235143 (cit. on p. 248).

[193] D. C. Liu and J. Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Math. Program.* 45.1-3 (1989), pp. 503–528 (cit. on p. 180).

[194] L. Lopez and V. Simoncini. "Analysis of projection methods for rational function approximation to the matrix exponential". In: *SIAM J. Numer. Anal.* 44.2 (2006), 613–635 (electronic) (cit. on p. 140).

[195] Y. Y. Lu. "Computing the logarithm of a symmetric positive definite matrix". In: *Applied numerical mathematics* 26.4 (1998), pp. 483–496 (cit. on p. 137).

[196] C. Lubich. "Discretized fractional calculus". In: *SIAM J. Math. Anal.* 17.3 (1986), pp. 704–719 (cit. on pp. 143, 147, 265).

[197] F. Mainardi. *Fractional calculus and waves in linear viscoelasticity: an introduction to mathematical models*. World Scientific, 2010 (cit. on p. 255).

[198] J. Mandel. "Algebraic study of multigrid methods for symmetric, definite problems". In: *Appl. Math. Comput.* 25.1 (1988), pp. 39–56 (cit. on p. 73).

[199] B. Mandelbrot. *The Fractal Geometry of Nature*. Henry Holt and Company, 1982. ISBN: 9780716711865 (cit. on p. 247).

[200] S. McCormick. "Multigrid methods for variational problems: general theory for the V–cycle". In: *SIAM J. Numer. Anal.* 22.4 (1985), pp. 634–643 (cit. on p. 73).

[201] M. M. Meerschaert and C. Tadjeran. "Finite difference approximations for fractional advection–dispersion flow equations". In: *J. Comput. Appl. Math.* 172.1 (2004), pp. 65–77 (cit. on pp. 143, 148, 157, 169, 176, 229).

[202] G. Meinardus and L. L. Schumaker. *Approximation of functions: theory and numerical methods*. 1st ed. Springer Tracts in Natural Philosophy. Springer, 1967. ISBN: 9783540039853,3540039856 (cit. on p. 81).

[203] J. Meldrum and N. Bourbaki. *Elements of the History of Mathematics*. Springer Berlin Heidelberg, 2013. ISBN: 9783642616938 (cit. on p. 247).

[204] R. Metzler and J. Klafter. "The random walk's guide to anomalous diffusion: a fractional dynamics approach". In: *Phys. Rep.* 339.1 (2000), pp. 1–77 (cit. on p. 255).

[205] G. Meurant. "On the incomplete Cholesky decomposition of a class of perturbed matrices". In: *Siam J. Sci. Comput.* 23.2 (2000), pp. 419–429 (cit. on p. 102).

[206] C. Moler and C. Van Loan. "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later". In: *SIAM Review* 45.1 (2003), pp. 3–49 (cit. on p. 121).

[207] J. J. Moré and D. J. Thuente. "Line search algorithms with guaranteed sufficient decrease". In: *ACM Trans. Math. Software* 20.3 (1994), pp. 286–307 (cit. on p. 180).

[208] I. Moret. "Rational Lanczos approximations to the matrix square root and related functions". In: *Numerical Linear Algebra with Applications* 16 (2009), pp. 431–445 (cit. on pp. 122, 140).

[209] I. Moret and P. Novati. "RD-Rational Approximations of the Matrix Exponential". In: *BIT, Numerical Mathematics* 44.3 (2004), pp. 595–615 (cit. on pp. 122, 140).

[210] I. Moret and M. Popolizio. "The restarted shift-and-invert Krylov method for matrix functions". In: *Numerical Linear Algebra with Appl.* 21.1 (2014), pp. 68–80 (cit. on pp. 122, 140).

[211] M. K. Ng. *Iterative methods for Toeplitz systems*. Oxford University Press, USA, 2004 (cit. on pp. 16, 230).

[212] M. K. Ng and J. Pan. "Approximate inverse circulant–plus–diagonal preconditioners for Toeplitz–plus–diagonal matrices". In: *SIAM J. Sci. Comput.* 32.3 (2010), pp. 1442–1464 (cit. on pp. 144, 154).

[213] M. K. Ng, S. Serra-Capizzano, and C. Tablino-Possio. "Multigrid preconditioners for symmetric Sinc systems". In: *ANZIAM Journal* 45 (2004), pp. 857–869 (cit. on p. 144).

[214] M. K. Ng, S. Serra-Capizzano, and C. Tablino-Possio. "Numerical behaviour of multigrid methods for symmetric Sinc–Galerkin systems". In: *Numer. Linear Algebra Appl.* 12.2-3 (2005), pp. 261–269 (cit. on p. 144).

[215] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006 (cit. on p. 180).

[216] Y. Notay. "Optimal V–cycle algebraic multilevel preconditioning". In: *Numer. Linear Algebra Appl.* 5.5 (1998), pp. 441–459 (cit. on p. 76).

[217] Y. Notay. "An aggregation–based algebraic multigrid method". In: *Electron. Trans. Numer. Anal.* 37.6 (2010), pp. 123–146 (cit. on p. 195).

[218] Y. Notay. "Aggregation-based algebraic multigrid for convection-diffusion equations". In: *SIAM J. Sci. Comput.* 34.4 (2012), A2288–A2316 (cit. on p. 195).

[219] D. Noutsos, S. Serra-Capizzano, and P. Vassalos. "Matrix algebra preconditioners for multilevel Toeplitz systems do not insure optimal convergence rate". In: *Theor. Comput. Sci.* 315.2-3 (2004), pp. 557–579 (cit. on p. 225).

[220] M. D. Ortigueira. "Riesz potential operators and inverses via fractional centred derivatives". In: *Int. J. Math. Math. Sci.* 2006 (2006) (cit. on pp. 149, 178, 224, 266).

[221] J. Pan, R. Ke, M. K. Ng, and H.-W. Sun. "Preconditioning techniques for diagonal–times–Toeplitz matrices in fractional diffusion equations". In: *SIAM J. Sci. Comput.* 36.6 (2014), A2698–A2719 (cit. on pp. 144, 154, 180).

[222] S. V. Parter. "On the distribution of the singular values of Toeplitz matrices". In: *Linear Algebra Appl.* 80 (1986), pp. 115–130 (cit. on pp. 62, 63).

[223] X. Ping, R. Chen, K. Tsang, and E. K. Yung. "The SSOR-preconditioned inner outer flexible GMRES method for the FEM analysis of EM problems". In: *Microw. Opt. Techn. Let.* 48.9 (2006), pp. 1708–1712 (cit. on p. 45).

[224] G. Plank, M. Liebmann, R. W. dos Santos, E. J. Vigmond, and G. Haase. "Algebraic multigrid preconditioner for the cardiac bidomain model". In: *IEEE Trans. Biomed. Eng.* 54.4 (2007), pp. 585–596 (cit. on p. 76).

[225] I. Podlubny. *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*. Vol. 198. Academic press, 1998 (cit. on pp. 143–145, 148, 150, 169, 172, 179, 256).

[226] I. Podlubny. "Matrix approach to discrete fractional calculus". In: *Fract. Calc. Appl. Anal.* 3.4 (2000), pp. 359–386 (cit. on pp. 143, 146, 265).

[227] I. Podlubny. "Geometric and physical interpretation of fractional integration and fractional differentiation." English. In: *Fract. Calc. Appl. Anal.* 5.4 (2002), pp. 367–386. ISSN: 1311-0454; 1314-2224/e (cit. on p. 143).

[228] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, and B. M. V. Jara. "Matrix approach to discrete fractional calculus II: Partial fractional differential equations". In: *J. Comput Phys.* 228.8 (2009), pp. 3137–3153 (cit. on pp. 143, 146, 265).

[229] I. Podlubny, T. Skovranek, B. M. V. Jara, I. Petras, V. Verbitsky, and Y. Chen. "Matrix approach to discrete fractional calculus III: non–equidistant grids, variable step length and distributed orders". In: *Philos. T. R. Soc. A* 371.1990 (2013), p. 20120153 (cit. on pp. 143, 146, 265).

[230] M. Popolizio. "A matrix approach for partial differential equations with Riesz space fractional derivatives". In: *Eur. Phys. J-Spec. Top.* 222.8 (2013), pp. 1975–1985 (cit. on pp. 143, 150, 154).

[231] M. Popolizio and V. Simoncini. "Acceleration techniques for approximating the matrix exponential". In: *SIAM J. Matrix Analysis Appl.* 30.2 (2008), pp. 657–683 (cit. on pp. 122, 140).

[232] A. Quarteroni. *Numerical Models for Differential Problems*. Vol. 2. MS&A. Springer Science & Business Media, 2010. 601 pp. (cit. on p. 173).

[233] A. Rafiei and F Toutounian. "New breakdown–free variant of AINV method for nonsymmetric positive definite matrices". In: *J. Comput. Appl. Math.* 219.1 (2008), pp. 72–80 (cit. on p. 96).

[234] A Ramage. "A multigrid preconditioner for stabilised discretisations of advection–diffusion problems". In: *J. Comput. Appl. Math.* 110.1 (1999), pp. 187–203 (cit. on p. 76).

[235] S. S. Ray. *Fractional calculus with applications for nuclear reactor dynamics*. CRC Press, 2015 (cit. on p. 255).

[236] R. Remmert. "Wielandt's Theorem About the Γ–Function". In: *Amer. Math. Monthly* 103.3 (1996), pp. 214–220. ISSN: 00029890, 19300972 (cit. on p. 255).

[237] J. C. De los Reyes. *Numerical PDE–constrained optimization*. Springer, 2015 (cit. on pp. 167, 168, 171, 175, 176).

[238] B. Riemann. "Versuch einer allgemeinen Auffassung der Integration und Differentiation". In: *Gesammelte Werke* 62 (1876), pp. 331–344 (cit. on pp. 248, 249).

[239] M. Riesz. "L'intégrale de Riemann–Liouville et le problème de Cauchy". In: *Acta mathematica* 81.1 (1949), pp. 1–222 (cit. on p. 248).

[240] R. B. Rood. "Numerical advection algorithms and their role in atmospheric transport and chemistry models". In: *Rev. Geophys.* 25.1 (1987), pp. 71–100 (cit. on p. 202).

[241] B. Ross. "The development of fractional calculus 1695–1900". In: *Historia Mathematica* 4.1 (1977), pp. 75–89 (cit. on p. 247).

[242] J. Ruge and K. Stüben. "Algebraic Multigrid". In: *Multigrid Methods*. Ed. by S. F. McCormick. Frontiers In Applied Mathematics. Philadelphia, Pennsylvania: SIAM, 1987. Chap. 4, pp. 73–130 (cit. on pp. 70, 75).

[243] Y. Saad. "Analysis of some Krylov subspace approximations to the matrix exponential operator". In: *SIAM J. Numer. Anal.* 29 (1992), pp. 209–228 (cit. on pp. 121, 122, 140).

[244] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003 (cit. on pp. 14, 22, 24, 25, 27, 30, 31, 36, 44, 47, 51, 76, 83, 233).

[245] Y. Saad. "A flexible inner–outer preconditioned GMRES algorithm". In: *SIAM J. Sci. Comput.* 14.2 (1993), pp. 461–469 (cit. on pp. 44, 47, 229, 233, 236).

[246] Y. Saad. "ILUT: A dual threshold incomplete LU factorization". In: *Numer. Linear Algebra Appl.* 1.4 (1994), pp. 387–402. URL: http://www-users.cs.umn.edu/~saad/PDF/umsi-92-38.pdf (cit. on pp. 83, 86).

[247] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003 (cit. on pp. 111, 155).

[248] Y. Saad. *Numerical Methods for Large Eigenvalue Problems 2nd edition*. SIAM, 2011 (cit. on p. 33).

[249] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM J. Sci. Stat. Comput.* 7.3 (1986), pp. 856–869 (cit. on pp. 35, 205).

[250] D. Sakrison. "An extension of the theorem of Kac, Murdock and Szegö to N dimensions (Corresp.)" In: *IEEE Trans. Inform. Theory* 15.5 (1969), pp. 608–610 (cit. on p. 67).

[251] S. G. Samko, A. A. Kilbas, and O. I. Marichev. *Fractional integrals and derivatives*. Theory and Applications. Yverdon: Gordon and Breach, 1993 (cit. on p. 171).

[252] R. Schumer, M. M. Meerschaert, and B. Baeumer. "Fractional advection-dispersion equations for modeling transport at the Earth surface". In: *J. Geophys. Res. Earth Surf.* 114.F4 (2009) (cit. on p. 256).

[253] S. Serra-Capizzano. "Distribution results on the algebra generated by Toeplitz sequences: a finite-dimensional approach". In: *Linear Algebra Appl.* 328.1-3 (2001), pp. 121–130 (cit. on p. 68).

[254] S. Serra-Capizzano. "Generalized Locally Toeplitz sequences: spectral analysis and applications to discretized partial differential equations". In: *Linear Algebra Appl.* 366 (2003). Special issue on Structured Matrices: Analysis, Algorithms and Applications, pp. 371–402. ISSN: 0024-3795 (cit. on pp. 67, 68, 196).

[255] S. Serra-Capizzano. "The GLT class as a generalized Fourier analysis and applications". In: *Linear Algebra Appl.* 419.1 (2006), pp. 180–233 (cit. on pp. 67, 196).

[256] S. Serra-Capizzano and C. Tablino-Possio. "Multigrid methods for multilevel circulant matrices". In: *SIAM J. Sci. Comput.* 26.1 (2004), pp. 55–85 (cit. on p. 75).

[257] S. Serra-Capizzano and E Tyrtyshnikov. "Any circulant-like preconditioner for multilevel matrices is not superlinear". In: *SIAM J. Matrix Anal. Appl.* 21.2 (2000), pp. 431–439 (cit. on p. 225).

[258] S. Serra-Capizzano and E Tyrtyshnikov. "How to prove that a preconditioner cannot be superlinear". In: *Math. Comp.* 72.243 (2003), pp. 1305–1316 (cit. on p. 225).

[259] J. Sherman and W. J. Morrison. "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix". In: *Ann. Of Math. Stat.* 21 (1 Mar. 1950), pp. 124–127. DOI: 10.2307/2236561 (cit. on p. 84).

[260] V. Simoncini and D. B. Szyld. "Flexible inner–outer Krylov subspace methods". In: *SIAM J. Numer. Anal.* 40.6 (2002), pp. 2219–2239 (cit. on p. 47).

[261] V. Simoncini and D. B. Szyld. "Theory of inexact Krylov subspace methods and applications to scientific computing". In: *SIAM J. Sci. Comput.* 25.2 (2003), pp. 454–477 (cit. on p. 45).

[262] V. Simoncini and D. B. Szyld. "Recent computational developments in Krylov subspace methods for linear systems". In: *Numer. Linear Algebra Appl.* 14.1 (2007), pp. 1–59 (cit. on p. 45).

[263] G. L. G. Sleijpen and H. A. van der Vorst. "Maintaing convergence properties of BiCGstab methods in finite precision arithmetic". In: *Numer. Algorithms* 10 (1995), pp. 203–223 (cit. on p. 56).

[264] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema. "BiCGstab(l) and other hybrid Bi–Cg methods". In: *Numer. Algorithms* 7 (1994), pp. 75–109 (cit. on p. 56).

[265] G. L. Sleijpen and D. R. Fokkema. "BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum". In: *Electron. Trans. Numer. Anal.* 1.11 (1993), p. 2000 (cit. on p. 56).

[266] P. Sonneveld. "CGS, a fast Lanczos–type solver for nonsymmetric linear systems". In: *SIAM J. Sci Stat. Comput.* 10 (1989), pp. 36–52 (cit. on p. 51).

[267] H. M. Srivastava and J. J. Trujillo. *Theory and applications of fractional differential equations*. Elsevier, Amsterdam, 2006 (cit. on p. 143).

[268] W. Stewart. "MARCA: Markov Chain Analyzer, A Software Package for Markov Modeling". In: *Numerical Solution of Markov Chains* 8 (1991), p. 37 (cit. on pp. 135, 137, 140).

[269] K. Stüben. "A review of algebraic multigrid". In: *J. Comput. Appl. Math.* 128.1 (2001), pp. 281–309 (cit. on pp. 75, 76).

[270] C.-H. Tai, J.-H. Sheu, and B. Van Leer. "Optimal multistage schemes for Euler equations with residual smoothing". In: *AIAA Journal* 33.6 (1995), pp. 1008–1016 (cit. on p. 206).

[271] P. L. Tchebychev. "Sur les polynômes réprésentant le mieux les valeurs des fonctions fractionnaires élémentaires pour les valeurs de la variable contenues entre deux limites données." In: *Oeuvres*. Ed. by S. Petersburg. Vol. II. Commissionaires de l'Académie impériale des sciences, 1907, pp. 669–678. (Cit. on p. 81).

[272] J. D. Tebbens and M. Tüma. "Efficient Preconditioning of Sequences of Nonsymmetric Linear Systems". In: *SIAM J. Sci. Comp.* 29 (5 2007), pp. 1918–1941. DOI: 10.1137/06066151x (cit. on p. 102).

[273] J. D. Tebbens and M. Tüma. "Preconditioner updates for solving sequences of linear systems in matrix-free environment". In: *Numer. Linear Algebra Appl.* 17 (6 2010), pp. 997–1019. DOI: 10.1002/nla.695 (cit. on p. 102).

[274] L. Thomas. "Using a computer to solve problems in physics". In: *Applications of Digital Computers*. Ed. by W. Freiberger and W. Prager. Boston: Ginn and Company, 1963, pp. 44–45 (cit. on p. 64).

[275] P. Tilli. "Locally Toeplitz sequences: spectral properties and applications". In: *Linear Algebra Appl.* 278.1 (1998), pp. 91–120 (cit. on pp. 67, 68).

[276] O. Toeplitz. "Zur Theorie der quadratischen und bilinearen Formen von unendlichvielen Veränderlichen". In: *Math. Ann.* 70.3 (1911), pp. 351–376 (cit. on p. 61).

[277] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000 (cit. on pp. 70, 71, 75).

[278] E. Tyrtyshnikov. "A unifying approach to some old and new theorems on distribution and clustering". In: *Linear Algebra Appl.* 232 (1996), pp. 1–43 (cit. on pp. 63, 67).

[279] E. Tyrtyshnikov. *A Brief Introduction to Numerical Analysis*. Boston: Birkhauser, 1997 (cit. on p. 32).

[280] E. Tyrtyshnikov and N. Zamarashkin. "Spectra of multilevel Toeplitz matrices: advanced theory via simple matrix relationships". In: *Linear Algebra Appl.* 270.1 (1998), pp. 15–27 (cit. on pp. 63, 67).

[281] B. Van Leer, W.-T. Lee, P. L. Roe, K. G. Powell, and C.-H. Tai. "Design of optimally smoothing multistage schemes for the Euler equations". In: *Commun. Appl. Numer. M.* 8.10 (1992), pp. 761–769 (cit. on pp. 195, 206).

[282] R. Varga. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer-Verlag, 1991 (cit. on p. 102).

[283] P. Vassilevski. "Hybrid V–cycle algebraic multilevel preconditioners". In: *Math. Comp.* 58.198 (1992), pp. 489–512 (cit. on p. 76).

[284] H. A. Van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003 (cit. on pp. 45, 50, 56, 229, 236).

[285] H. A. Van der Vorst. "Bi–CGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems". In: *SIAM J. Sci. Stat. Comput.* 13.2 (1992), pp. 631–644 (cit. on pp. 50, 51, 55, 205).

[286] J. Wendel. "Note on the gamma function". In: *Am. Math. Mon.* 55.9 (1948), pp. 563–564 (cit. on p. 147).

[287] H. Weyl. "Bemerkungen zum begriff des differentialquotienten gebrochener ordnung". In: *Zürich. Naturf. Ges* 62 (1917), pp. 296–302 (cit. on p. 248).

[288] N. Wheeler. *Construction & Physical Application of the Fractional Calculus*. Reed College Physics Department. 1997. URL: `https://goo.gl/5GdFfL` (cit. on p. 247).

[289] J. H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, 1965 (cit. on pp. 47, 48).

[290] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation Vol. II – Linear Algebra*. Springer–verlag, 1971 (cit. on p. 59).

[291] S. Winograd. "On computing the discrete Fourier transform". In: *Math. Comput.* 32.141 (1978), pp. 175–199 (cit. on p. 64).

[292] Q. Yang, F. Liu, and I Turner. "Numerical methods for fractional partial differential equations with Riesz space fractional derivatives". In: *Appl. Math. Model.* 34.1 (2010), pp. 200–218 (cit. on pp. 143, 162).

[293] H Zhang, F. Liu, and V. Anh. "Galerkin finite element approximation of symmetric space–fractional partial differential equations". In: *Applied Mathematics and Computation* 217.6 (2010), pp. 2534–2545 (cit. on pp. 170, 171).

[294] J. Zhang. "Accelerated multigrid high accuracy solution of the convection-diffusion equation with high Reynolds number". In: *Numer. Methods Partial Differential Equations* 13.1 (1997), pp. 77–92 (cit. on p. 195).

[295] A. Zygmund. *Trigonometric series*. Vol. 1. Cambridge university press, 2002 (cit. on p. 61).