Delft University of Technology

Induced Dimension Reduction algorithms for solving non-symmetric sparse matrix problems

Astudillo Rengifo, Reinaldo

DOI
10.4233/uuid:85261582-62d7-4b17-9c18-a1b2d11a07b6

Publication date
2018

Document Version
Final published version

Citation (APA)
Astudillo Rengifo, R. (2018). Induced Dimension Reduction algorithms for solving non-symmetric sparse matrix problems DOI: 10.4233/uuid:85261582-62d7-4b17-9c18-a1b2d11a07b6

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Induced Dimension Reduction algorithms for solving non-symmetric sparse matrix problems

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
vrijdag 16 maart 2018 om 10:00 uur

door

Reinaldo Antonio ASTUDILLO RENGIFO
Magister Scientiarum Mención: Ciencias de la Computación,
Universidad Central de Venezuela, Caracas, Venezuela

geboren te Caracas, Venezuela

Dit proefschrift is goedgekeurd door de promotor en de copromotor.

Samenstelling promotiecommissie bestaat uit:

Rector Magnificus	voorzitter
Prof.dr.ir. C. Vuik	Technische Universiteit Delft, promotor
Dr.ir. M.B. van Gijzen	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof.dr. D.B. Szyld	Temple University, Verenigde Staten van Amerika
Prof.dr. K. Abe	Gifu Shotoku Gakuen University, Japan
Prof.dr.ir. A. van Keulen	Technische Universiteit Delft
Prof.dr.ir. A.W. Heemink	Technische Universiteit Delft
Dr. M.E. Hochstenbach	Technische Universiteit Eindhoven

Keywords: Krylov subspace methods, Induced Dimension Reduction, system of linear equations, eigenvalues/eigenvectors approximation, solution of matrix linear problems.

Induced Dimension Reduction algorithms for solving non-symmetric sparse matrix problems
Dissertation at Delft University of Technology.
Copyright © 2018 by R.A. Astudillo Rengifo

ISBN 978-94-6366-019-8

Printed by: ProefschriftMaken (`www.proefschriftmaken.nl`)
Cover design by: Luis Jiménez (`http://be.net/luisdjt`)

To my grandmother Carmen Astudillo

"[IDR] It is the best method that we have at the moment ..."
Prof. Martin Gutknecht at the *GAMM meeting, Germany, 2008.*

Contents

Samenvatting

Induced Dimension Reduction algoritmes voor het oplossen van niet-symmetrisch ijle matrices

Reinaldo Antonio Astudillo Rengifo

In veel wetenschappelijke en technische toepassingen leidt het discretiseren van partiële differentiaalvergelijkingen tot verscheidene matrixproblemen. Deze scriptie gaat over de ontwikkeling van nieuwe algoritmes voor het oplossen van zulke problemen. Speciale aandacht gaat uit naar problemen met asymmetrische ijle matrices. De nieuwe algoritmes zijn gebaseerd op de Induced Dimension Reduction methodes [IDR(s)].

IDR(s) is een Krylovruimte methode die in 2008 werd geïntroduceerd om systemen van lineaire vergelijkingen op te lossen. IDR(s) heeft veel aandacht gekregen vanwege zijn stabiliteit en snelle convergentie. Het is daarom logisch om te onderzoeken of IDR(s) uit te breiden is naar andere matrixproblemen, en zo ja, hoe deze uitbreidingen zich verhouden tot gevestigde methodes.

De voornaamste matrixproblemen in deze scriptie zijn: het standaard eigenwaardeprobleem, het kwadratische eigenwaardeprobleem, het oplossen van systemen van lineaire vergelijkingen, het oplossen van reeksen van systemen van lineaire vergelijkingen en lineaire matrixvergelijkingen. De focus ligt op voorbeelden die voortkomen uit het discretiseren van partiële differentiaalvergelijkingen.

Eerst analyseren wij de recurrente betrekkingen van IDR(s) en leiden we een Hessenberg decompositie af. Dit stelt ons in staat om een deelverzameling van eigenwaarden en eigenvectoren te benaderen. We laten zien hoe het conditiegetal van de basis van de Krylovruimte, gegenereerd door IDR(s), gerelateerd is aan het verschil tussen de Ritz waarden van ons voorgestelde algoritme en de Ritz waarden afkomstig van de Arnoldi methode. We passen ook Sorensens impliciete herstarttechniek toe op ons voorgestelde algoritme. In de numerieke voorbeelden laat IDR(s) voor eigenwaarden competitief gedrag

zien vergeleken met de bekende Implicitly Restarted Arnoldi methode (IRAM). Verder passen we de voorgestelde IDR(s) eigensolver toe op het kwadratische eigenwaardeprobleem.

Met gebruik van de Hessenberg relatie verkregen met IDR(s) laten we zien hoe Ritz waarden en de bijbehorende Ritz vectoren verkregen kunnen worden tijdens het toepassen van IDR(s) voor het oplossen van systemen van lineaire vergelijkingen. Deze methodologie passen we toe op twee verschillende problemen: het oplossen van systemen van lineaire vergelijkingen en het oplossen van reeksen van systemen van lineaire vergelijkingen. Eerst maken we een IDR(s) die Ritz waarden berekent, deze worden gebruikt als inputparameters om het oplossen van systemen van lineaire vergelijken te versnellen. Ten tweede gebruiken we Ritz vectoren om systemen van lineaire vergelijkingen met IDR(s) op te lossen. De kerngedachte wordt gevormd door het berekenen van een paar Ritz vectoren tijdens het oplossen van het eerste systeem van lineaire vergelijkingen, en deze te gebruiken om de convergentie te verbeteren van de volgende systemen van lineaire vergelijkingen.

We gebruiken IDR(s) ook voor het oplossen van lineaire matrixvergelijkingen. Door gebruik te maken van een generalisatie van de IDR stelling, passen we IDR(s) toe op lineaire matrixvergelijkingen zoals: de Lyapunov vergelijking, de Sylvester vergelijking, bloksystemen van lineaire vergelijkingen, en de multi-shift Helmholtz vergelijking. We ontwerpen ook een nieuwe preconditioner voor de matrixvergelijking verkregen uit de multi-shift Helmholtz vergelijking. Deze voorgestelde preconditioner gebruikt de incomplete LU-factorisatie van de verschoven Laplaciaanse matrix, in plaats van de exacte LU-factorisatie, waardoor hij geschikt is voor problemen van grote omvang. We combineren deze incomplete LU-factorisatie met een andere operator die de eigenwaarden roteert van de operator die bij de matrixvergelijking hoort. Deze preconditioner versnelt de convergentie van de iteratieve methode terwijl hij het gebruik van een groter bereik van verschuivingen mogelijk maakt.

We kunnen de impact van het onderzoek dat uitgevoerd is tijdens dit project als volgt samenvatten. Het ontwikkelen van nieuwe, op IDR(s) gebaseerde methodes met korte recurrenties, voor het oplossen van matrixproblemen, zou een numeriek alternatief kunnen vormen voor methodes gebaseerd op Lanczos en Arnoldi methodes, vanwege de numerieke stabiliteit en het CPU- en geheugengebruik. Aan de theoretische kant kan het onderzoek naar spectrale informatie verkregen met de IDR(s) methode voor systemen van lineaire vergelijkingen helpen om zijn convergentie-eigenschappen te begrijpen. Daarnaast is de ontwikkeling van IDR(s) voor matrixvergelijkingen een eerste stap richting de ontwikkeling van geavanceerdere algoritmes die gebruik kunnen maken van de lagerangsbenaderingen van de oplossingen. In het geval van het oplossen van de multi-shift Helmholtz vergelijking, maakt de voorgestelde aanpak voor matrixvergelijkingen het gebruik van flexibelere preconditioners mogelijk, zoals incomplete factorisaties en het gebruikt van deflatie- en augmentatietechnieken.

Summary

Induced Dimension Reduction algorithms for solving non-symmetric sparse matrix problems

Reinaldo Antonio Astudillo Rengifo

In several applications in science and engineering, different types of matrix problems emerge from the discretization of partial differential equations. This thesis is devoted to the development of new algorithms to solve this kind of problems. In particular, when the matrices involved are sparse and non-symmetric. The new algorithms are based on the Induced Dimension Reduction method [IDR(s)].

IDR(s) is a Krylov subspace method originally proposed in 2008 to solve systems of linear equations. IDR(s) has received considerable attention due to its stable and fast convergence. It is, therefore, natural to ask if it is possible to extend IDR(s) to solve other matrix problems, and if so, to compare those extensions with other well-established methods. This work aims to answer these questions.

The main matrix problems considered in this dissertation are: the standard eigenvalue problem, the quadratic eigenvalue problem, the solution of systems of linear equations, the solution of sequences of systems of linear equations, and linear matrix equations. We focus on examples that arise from the discretization of partial differential equations.

First, we analyze the IDR(s) recurrence formulas, and derive a Hessenberg decomposition. This allows us to create an IDR-based algorithm for approximating a subset of eigenvalues and eigenvectors of a given matrix. We illustrate how the condition number of the Krylov subspace basis generated by IDR(s) is related to the difference between the Ritz values from our proposed algorithm and the Ritz values generated by the Arnoldi method. We also apply Sorensen's implicit restarting technique to our proposed algorithm. In the numerical examples, IDR(s) for eigenvalues shows a competitive behavior in comparison to the well-known Implicitly Restarted Arnoldi method (IRAM). Additionally, we apply the proposed IDR(s) eigensolver to the quadratic eigenvalue problem.

Making further use of the Hessenberg relation obtained from IDR(s), we show how to obtain the Ritz values and their corresponding Ritz vectors during the application of IDR(s) for the solution of systems of linear equations. We apply this methodology to two different problems: the solution of systems of linear equations and solving sequences of systems of linear equations. First, we create an IDR-solver that computes the Ritz values, and then uses them as input parameters to speed-up the solution of systems of linear equations. Second, we use the Ritz vectors to solve sequences of systems of linear equations with IDR(s). The main idea is to compute few Ritz vectors during the solution of the first system of linear equations, and use them to accelerate the convergence of the rest of the systems of linear equations.

We also use IDR(s) to solve linear matrix equations. Using a generalization of the IDR Theorem, we apply IDR(s) for solving linear matrix equations, such as, the Lyapunov equation, Sylvester equation, block systems of linear equations, and multi-shift Helmholtz equation. Finally, we design a new preconditioner for the matrix equation obtained from the multi-shift Helmholtz problem. This proposed preconditioner uses the incomplete LU factorization of the shifted Laplacian matrix rather than the exact LU factorization of this matrix, which makes it suitable for solving large-scale problems. We combine the use of the incomplete LU factorization with another operator that rotates the eigenvalues of the operator associated with the matrix equation. This preconditioner accelerates the convergence of the iterative method while enabling the use of a larger range of shifts.

We can summarize the impact of the research conducted in this project as follows. The development of new short-recurrence methods to solve matrix problems based on IDR(s) might represent an intermediate computational alternative to Lanczos-based methods and Arnoldi-based methods, from the point of view of numerical stability, and CPU and memory consumption. On the theoretical side, the study of the spectral information obtained from the IDR(s) method for solving systems of linear equations might help understanding its convergence properties. Additionally, the development of IDR(s) for matrix equations is a first step towards the development of more advanced algorithms that can exploit low-rank representations of the solutions. In the case of the solution of the multi-shift Helmholtz equation, the proposed matrix

equation approach allows the use of more flexible preconditioners, such as, incomplete factorizations, and the use of deflation and augmentation techniques.

Notation

Throughout this document, we adopt the following notation,

Description	**Meaning**	**Example**
Symbol $\mathbb{C}$	Set of complex numbers	$\mathbb{C}$
Symbol $\mathbb{R}$	Set of real numbers	$\mathbb{R}$
Symbol $\mathbb{N}$	Set of natural numbers	$\mathbb{N}$
Symbol $\mathbb{N}^+$	Set of positive natural numbers	$\mathbb{N}^+$
Capital letters	Matrices	A, B, $C, \ldots$
Capital letter T as superindex	Transpose operator	A^T, B^T, $\mathbf{x}^T, \ldots$
Capital letter H as superindex	Conjugate transpose operator	A^H, B^H, $\ldots$
Capital I with or without subindex n	Identity matrix of order n. The subindex is dropped when the dimension is clear from the context	I or I_n
Bolded lower letters	Columns vectors	$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}, \ldots$
Bolded number 0	Zero vector	$\mathbf{0}$
Greek lower letters	Complex or real scalars	α, β, $\gamma, \ldots$
Bolded lower letters with subindex	Vectors in a sequence or columns of a matrix (represented with same letter in capital)	$\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{a}_i, \ldots$
Function max	The largest of two numbers	$\max(a, b)$
Function min	The smallest of two numbers	$\min(a, b)$
Symbol $\otimes$	Kronecker product	$A \otimes B$
Function trace	The trace of a matrix	$\text{trace}(A) = \sum_{i=1}^{n} a_{ii}$

Bolded lower e with subindex i	ith canonical vector	$\mathbf{e}_1, \mathbf{e}_2, \dots$
Calligraphic capital letters	Vectors (sub)spaces	$\mathcal{G}, \mathcal{S}, \dots$
Calligraphic capital A	(Chapter 5) General linear operator	$\mathcal{A}$
Calligraphic capital P	(Chapter 5) General linear operator preconditioner	$\mathcal{P}$
Calligraphic capital I	(Chapter 5) Identity as linear operator	$\mathcal{I}$
Lower letters	Functions	$f, g, h, \dots$
Lower letter i	Imaginary unit $i^2 = -1$	$5 + 9i$
Symbol $\Re$	Real part of a complex number	$\Re(z)$
Symbol $\Im$	Imaginary part of a complex number	$\Im(z)$
Symbol $\langle \cdot, \star \rangle$	Euclidean inner product	$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y}^H \mathbf{x}$
Symbol $\langle \cdot, \star \rangle_F$	Frobenius inner product	$\langle A, B \rangle_F = \text{trace}(B^H A)$
Symbol $\|\cdot\|$	Euclidean norm	$\|\mathbf{x}\|$
Symbol $\|\cdot\|_F$	Frobenius norm	$\|A\|_F$
Symbol $\frac{du}{dx}$	When u is a real-valued function, this represents the derivative of u with respect to x	$\frac{du}{dx}$
Symbol $\frac{\partial u}{\partial x}$	When u is a multi-variate function, this represents the partial derivative of u with respect to x	$\frac{\partial u}{\partial x_i}$
Symbol ∇u	When u is a multi-variate function, this represents a vector with all partial derivatives	$\nabla u = [\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3}]^T$
Symbol $\triangle u$	When u is a multi-variate function, this represents the Laplacian operator applied to u.	$\triangle u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2}$
Function vec	It represents a vector created from a matrix by stacking its columns	$\text{vec}(A)$

CHAPTER 1

Introduction and preliminaries

The core of this work is the development of algorithms to solve matrix problems. In particular, the algorithms developed here are based on the Induced Dimension Reduction method (IDR(s)) [95]. IDR(s) is an iterative Krylov method to solve systems of linear equations. IDR(s) has proved to be a competitive option to solve system of linear equations with respect other well-established iterative Krylov methods. For this reason, we investigate how to adapt this method to solve other types of matrix problems such as eigenvalues/eigenvector approximation, quadratic eigenvalues problem, sequences of systems of linear equations or matrix equations.

The discretization of partial differential equations to solve models of different phenomena in sciences and engineering, is the most common source of matrix problems. One main characteristic of the matrices resulting of this discretization process (via finite elements or finite differences among other techniques) is the sparsity. Sparse matrices are matrices of which most of their elements are zeros. This allows the creation of efficient computer data structures to store and handle this kind of matrices with large dimension. However, these large dimensions make unfeasible the application of matrix decompositions such as SVD, or LU, or algorithms as Gaussian elimination or Francis QR [38, 39]. For this reason, it is important to investigate other options to solves this kind of problems, such as iterative Krylov methods.

Another important characteristic of the matrix problems investigated in this dissertations is the non-symmetry. For solving system of linear equations where the coefficient matrix is symmetric and positive definite (SPD), preconditioned Conjugate Gradient (CG) [46] is the most well-established option.

Lanczos algorithm [55], [22] is the most common option for approximating the eigenvalues and eigenvectors of symmetric matrices. Nevertheless, it is difficult to choose a clear winner among Krylov method when the coefficient matrix is non-symmetric [65].

The remainder of this chapter is organized as follows. In the next section, we present in detail the class of matrix problems that we deal with in this dissertation, their importance, their corresponding assumptions, and some motivating examples. Section 1.2 provides a summary of the structure of this document. In sections 1.3, 1.4, and 1.5, we present a brief review of the most well-known Krylov subspace methods for solving system of linear equations, approximate eigenvalues and eigenvectors, and solving matrix equations respectively.

1.1 Linear matrix problems

As we mention in the introduction, we consider different types of matrix problems. The common characteristics of these problems is that the matrices involved are sparse and non-symmetric.

IDR(s) was designed specifically for the solution of system of linear equations

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

which is one of most ubiquitous problems in science and engineering. In (1.1), A is $n \times n$ complex or real matrix called the coefficient matrix, the vector $\mathbf{b}$ or right-hand size vector is a given vector of dimension n, and $\mathbf{x}$ is the unknown vector. This kind of problem arises naturally after the discretization of partial differential equations. In this research, we only consider the case when (1.1) has a unique solution.

A variation of the problem (1.1) is that of a sequence of systems of linear equations

$$A_i\mathbf{x}_i = \mathbf{b}_i \qquad \text{for } i = 1, \dots, m. \tag{1.2}$$

In this document, we only consider the case where the coefficient matrix is constant and non-singular. Then problem (1.2) can be rewritten as

$$A\mathbf{x}_i = \mathbf{b}_i \qquad \text{for } i = 1, \dots, m,$$

and each system of equations has a unique solution. The right-hand size vectors may not be all available simultaneously. This type of problems emerge from the discretization of linear time-dependent differential equations and the solution of systems of non-linear equations using modified Newton-type methods with constant Jacobian matrix

We also consider the eigenvalue problem. Given a matrix A find a subset of pairs $(\lambda, \mathbf{x})$ such that λ is a complex scalar and $\mathbf{x}$ is a non-null vector in $\mathbb{C}^n$ and

$$A\mathbf{x} = \lambda\mathbf{x}; \tag{1.3}$$

the vectors $\mathbf{x}$ are called eigenvectors of A, while the scalars λ are referred as eigenvalues. This problem typically arises solution of first-order differential equations, stability analysis, and partial differential equations. One strategy to obtain approximations to the eigenpairs of the large and sparse matrix A is to create a Hessenberg decomposition as

$$AQ_m = Q_m H_m + \mathbf{q}\mathbf{e}_m^T, \tag{1.4}$$

where $Q_m \in \mathbb{C}^{n\times m}$, H_m is a $m \times m$ upper Hessenberg matrix, and $\mathbf{q} \in \mathbb{C}^n$ with $m < n$. Part of the eigenvalues of the matrix H_m are approximations to a subset of eigenvalues of A, and the corresponding approximation to the eigenvectors of A are obtained as a linear combination of the columns of the matrix Q_m.

We also consider the numerical solution of linear matrix equations

$$\sum_{j=1}^{k} A_j X B_j = C. \tag{1.5}$$

The matrices $A_j \in \mathbb{C}^{n\times n}$, $B_j \in \mathbb{C}^{m\times m}$, and $C \in \mathbb{C}^{n\times m}$ are given matrices. The matrix X is the unknown $n \times m$ matrix. The general matrix equation (1.5) is equivalent to the following system of linear equations

$$\left(\sum_{j=1}^{k} B_k^T \otimes A_k\right) \operatorname{vec}(X) = \operatorname{vec}(C), \tag{1.6}$$

where vec(Y) is the operation of creating a vector of order $n \times m$ by stacking the columns of the matrix Y, and the operator $\otimes$ is referring to the Kronecker product (see [54]).

Several applications of the control theory lead to the solution of the particular cases of (1.5), such as, the Lyapunov equation

$$AX + XA^T = B, \tag{1.7}$$

or the Sylvester equation

$$AX + XB = C. \tag{1.8}$$

1.2 Dissertation outline and abstracts

This document is structured as follows,

- **Chapter 2: The Induced Dimension Reduction method** In this chapter, we review the Induced Dimension Reduction method. We present a summary of the development and evolution of the IDR(s). We emphasize in the description of the numerical properties of IDR(s) and its implementation. Also to provide to the reader an idea about the computational behavior of the IDR(s), we solve different system of linear equations that arise from the discretization of a simple model of the convection-diffusion equation. We present a comparison with other well-known iterative methods to solve systems of linear equations as GMRES, BiCG and BiCGStab.

- **Chapter 3: Induced Dimension Reduction method for solving eigenvalue problem** In chapter 3, we adapt IDR to solve the problem of approximating the eigenvalues and eigenvectors of sparse and non-symmetric matrices. From the IDR(s) calculation, we create a standard Hessenberg decompositions of the form (1.4), that allows us to approximate a subset of eigenpairs of the matrix A. In order to obtain the eigenvalues of the interest for our application, we apply the implicit restarting technique [97].

  Additionally, we apply our proposed IDR(s) algorithm to solve the quadratic eigenvalue problem

$$(\lambda^2 M + \lambda D + K)\mathbf{x} = \mathbf{0}. \tag{1.9}$$

- **Chapter 4: Accelerating the Induced Dimension Reduction method using spectral information** As a follow-up of the approximation of eigenvalues and eigenvector with IDR(s), we incorporate this information to accelerate the convergence of IDR(s) for solving a linear systems. Ritz-IDR(s) [87] was the first variant of IDR(s) that uses few approximations of the eigenvalues to accelerate its convergence. Nevertheless, it requires a previous call to an eigensolver routines as the Arnoldi method [4]. We create a self-contained variant of Ritz-IDR(s) that does not use an external routine for the eigenvalue approximation. We name it self-contained Ritz-IDR(s) or SC-Ritz-IDR(s). The approximated eigenvectors are used in the context of solving a sequence of systems of linear equations where the coefficient matrix is constant. During the solution of the first linear system in the sequence, we compute approximations of a subset of eigenvectors of the coefficient matrix. Then, we use these eigenvectors to enrich the Krylov subspace used by IDR(s) in the solution of the subsequent linear systems.

- **Chapter 5: IDR(s) for solving linear matrix equations** IDR(s) is based on the IDR(s) Theorem (see Theorem 2.1 in [95]). IDR(s) Theorem defines a sequence of nested and shrinking subspaces in $\mathbb{C}^n$ where the residuals of the approximations are created. This sequence of nested and shrinking can also be defined in any finite dimensional subspace and not only in $\mathbb{C}^n$. Taking advantages of this fact, we propose an IDR(s) for solving matrix equations (equivalent to a block version of IDR(s)).

In chapter 6, we give the general conclusions of this dissertation and future lines of research. For sake of completeness, the rest of this chapter is devoted to present an overview of the mainstream Krylov iterative method to solve non symmetric systems of linear equation and to approximate eigenvalues and eigenvectors.

1.3 Solving systems of linear equations

In this section, we review some of the most well-known Krylov iterative methods to solve systems of linear equations. In particular, we examine the Bi-Conjugate Gradient method (BiCG) [37], the Generalized Minimal Residual method (GMRES) [81], and the Bi-Conjugate Gradient stabilized method [101]. These methods are used for comparison purpose throughout this document. For a more comprehensive description of this topic, we refer the reader to [79].

When the coefficient matrix A is large and sparse, direct methods to solve system of linear equations become too expensive from the computational point of view. Examples of direct method as the classical Gaussian elimination or LU factorization have a computational complexity of $\mathcal{O}(n^3)$. On the other hand, iterative methods to solve system of linear equations create a sequence of vectors $\{\mathbf{x}_k\}_{k=1}^{\infty}$, that under certain conditions, converges to the solution of (1.1). Iterative methods are suitable for large-scale settings.

In this dissertation, we deal with the iterative projection method onto m-dimensional subspace $\hat{\mathcal{K}}$ and orthogonal to the m-dimensional subspace $\mathcal{L}$. These iterative methods find the approximate solution $\mathbf{x}_m$ in the affine subspace $\mathbf{x}_0 + \hat{\mathcal{K}}$ imposing the Petrov-Galerkin condition, i.e., $\mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$ orthogonal to $\mathcal{L}$. The subspace $\hat{\mathcal{K}}$ is called search space, while $\mathcal{L}$ is called restriction space. In particular, we are interested in iterative Krylov methods, which are projection methods whose search subspace is the Krylov subspace.

Definition 1. Let $A \in \mathbb{C}^{n\times n}$, and let $\mathbf{v} \in \mathbb{C}^n$ be a non-zero vector. The mth

Krylov subspace associated with A and $\mathbf{v}$, denoted by $\mathcal{K}_m(A,\, \mathbf{v})$, is defined as

$$\mathcal{K}_m(A,\, \mathbf{v}) = \operatorname{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \dots, A^{m-1}\mathbf{v}\}. \tag{1.10}$$

Where $\operatorname{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$ is set of all linear combinations of the vectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$.

1.3.1 Biconjugate gradients method

BiCG was proposed by R. Fletcher in 1976 [37]. This is a projection method whose searching subspace is

$$\mathcal{K}_m(A, \mathbf{v}_1) = \operatorname{span}\{\mathbf{v}_1,\, A\mathbf{v}_1,\, \dots,\, A^{m-1}\mathbf{v}_1\}.$$

and the residual vector is orthogonal to the subspace

$$\mathcal{K}_m(A^H, \mathbf{w}_1) = \operatorname{span}\{\mathbf{w}_1,\, A^H\mathbf{w}_1,\, \dots,\, (A^H)^{m-1}\mathbf{w}_1\},$$

where $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|$ and $\mathbf{w}_1$ is a non-zero vector such that $\langle \mathbf{v}_1, \mathbf{w}_1 \rangle = 1$.

The basis for the subspaces $\mathcal{K}_m(A, \mathbf{v}_1)$ and $\mathcal{K}_m(A^H, \mathbf{w}_1)$ are created using the Lanczos bi-orthogonalization method [55]. Despite the fact that the BiCG algorithm has low computational cost in terms of memory, inner products and vectors operations, it requires two matrix-vector multiplications per iteration (a matrix-vector multiplication using A and another using A^H). BiCG finds the exact solution in n iteration using exact arithmetic ($2n$ matrix-vector products). Algorithm 1 presents an implementation of the BiCG method.

Algorithm 1 BiCG

```
1: Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $tol \in (0, 1)$, $\mathbf{x}_0 \in \mathbb{C}^n$.
2: $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$. Choose $\hat{\mathbf{r}}_0$ such that $\langle \mathbf{r}_0, \hat{\mathbf{r}}_0 \rangle \neq 0$
3: for $j = 0$ to convergence do
4:     $\alpha_j = \langle \mathbf{r}_j, \hat{\mathbf{r}}_j \rangle / \langle A\mathbf{p}_j, \hat{\mathbf{p}}_j \rangle$
5:     $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6:     $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A\mathbf{p}_j$
7:     $\hat{\mathbf{r}}_{j+1} = \hat{\mathbf{r}}_j - \alpha_j A^H \hat{\mathbf{p}}_j$
8:     $\beta_j = \langle \mathbf{r}_{j+1}, \hat{\mathbf{r}}_{j+1} \rangle / \langle \mathbf{r}_j, \hat{\mathbf{r}}_j \rangle$
9:     $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
10:    $\hat{\mathbf{p}}_{j+1} = \hat{\mathbf{r}}_{j+1} + \beta_j \hat{\mathbf{p}}_j$
11: end for
12: Return $\mathbf{x}_{j+1}$
```

1.3.2 Generalized minimal residual method

The Generalized Minimal Residual method (GMRES) was introduced by Saad and Schultz in [81]. GMRES finds the approximation $\mathbf{x}_m$ at iteration m in the subspace $\mathcal{K}_m(A, \mathbf{v}_1)$, and imposes the residual to be orthogonal to $A\mathcal{K}_m(A, \mathbf{v}_1)$ with $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|$. GMRES can be formulate as

$$\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m \tag{1.11}$$

with

$$(AV_m)^H\mathbf{r}_m = \mathbf{0}$$

or equivalently

$$\mathbf{y}_m = \min_{\mathbf{y}} \|\mathbf{b} - A(\mathbf{x}_0 + V_m\mathbf{y})\|, \tag{1.12}$$

where V_m is a matrix whose columns form a basis for $\mathcal{K}_m(A, \mathbf{v}_1)$.

Equations (1.11) and (1.12) ensure the optimal residual norm condition, i.e., GMRES obtains an approximation $\mathbf{x}_m$ that generates the minimum residual norm possible in the subspace $\mathcal{K}_m(A, \mathbf{r}_0)$, as a consequence, GMRES finds the exact solution of the linear system in n iterations (n matrix-vector) products in exact arithmetic.

In order to build the matrix V_m and solve efficiently (1.12), GMRES is commonly implemented using the Arnoldi method [4]. However, the computational cost and memory requirement of the Arnoldi method grow as m increases. This makes GMRES prohibitive for large problems. A option to overcome this issue is to restrict the maximum value of m, and restart the process with approximation $\mathbf{x}_m$ as the new starting vector. This variant is known as the restarted GMRES or GMRES(m).

Algorithm 2 GMRES(m)

```
 1: Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $tol \in (0, 1)$, $m \in \mathbb{N}^+$, $\mathbf{x}_0 \in \mathbb{C}^n$.
 2: $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
 3: for $j = 1$ to $m$ do
 4:     $\mathbf{w} = A\mathbf{v}_j$
 5:     for $i = 1$ to $j$ do
 6:         $H_{i,j} = \langle \mathbf{w}, \mathbf{v}_i \rangle$
 7:         $\mathbf{w} = \mathbf{w} - H_{i,j}\mathbf{v}_i$
 8:     end for
 9:     $H_{j+1,j} = \|\mathbf{w}\|$, and $\mathbf{v}_{j+1} = \mathbf{w}/H_{j+1,j}$
10: end for
11: Define $V_m = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$
12: Compute $\mathbf{y}_m = \text{argmin}_y \|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y}_m\|$        $\triangleright$ $\bar{H}_m \in \mathbb{C}^{m+1\times m}$
13: $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$
14: if convergence test fails then
15:     $\mathbf{x}_0 = \mathbf{x}_m$ goto 2.
16: end if
17: Return $\mathbf{x}_m$
```

1.3.3 Biconjugate gradient stabilized method

This method was proposed by H. van der Vorst in 1992 [101]. BiCGStab is a variant of the Conjugate Gradient Square (CGS)[†] proposed by Sonneveld [93]. CGS replaces the matrix-vector product with A^H present in each iteration of the BiCG via squaring the residual polynomial. However, CGS might suffer from irregular convergence, due to this squaring operation which makes CGS sensitive to round-off errors [101].

The main idea behind BiCGStab is to multiply the CGS residual polynomial by a degree one polynomials to counterbalance the irregular behavior of CGS. BiCGStab is one of the most widely used short-recurrences methods, it uses two matrix-vector products per iteration and low memory consumption. Nevertheless, when the eigenvalues of the matrix A have imaginary parts that are large relative to the real parts, BiCGStab may suffer of stagnation (see [88]). Algorithm 3 shows an implementation of the BiCGStab method.

Algorithm 3 BiCGStab

```
 1: Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $tol \in (0, 1)$, $\mathbf{x}_0 \in \mathbb{C}^n$.
 2: $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$. Choose $\hat{\mathbf{r}}_0 \in \mathbb{C}^n$, s.t. $\langle \mathbf{r}_0, \hat{\mathbf{r}}_0\rangle \neq 0$
 3: $\mathbf{p}_0 = \mathbf{r}_0$
 4: for $j = 0$ to convergence do
 5:     $\alpha_j = \langle \mathbf{r}_j, \hat{\mathbf{r}}_0\rangle / \langle A\mathbf{p}_j, \hat{\mathbf{r}}_0\rangle$
 6:     $\mathbf{s}_j = \mathbf{r}_j - \alpha_j A\mathbf{p}_j$
 7:     $\omega_j = \langle \mathbf{s}_j, A\mathbf{s}_j\rangle / \langle A\mathbf{s}_j, A\mathbf{s}_j\rangle$
 8:     $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j$
 9:     $\mathbf{r}_{j+1} = \mathbf{s}_j - \alpha_j A\mathbf{s}_j$
10:     $\beta_j = \frac{\langle \mathbf{r}_{j+1}, \hat{\mathbf{r}}_0\rangle}{\langle \mathbf{r}_j, \hat{\mathbf{r}}_0\rangle} \frac{\alpha_j}{\omega_j}$
11:     $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j(\mathbf{p}_j - \omega_j A\mathbf{p}_j)$
12: end for
13: Return $\mathbf{x}_{j+1}$
```

1.3.4 Long and short-recurrences methods and optimal residual

The Conjugate Gradient method, applied to a SPD matrix A, has three optimal properties. First, CG creates an approximation $\mathbf{x}_m$ in the space $\mathcal{K}_m(A, \mathbf{r}_0)$ using exactly m matrix-vector products. Second, the residual $\mathbf{r}_m$ is minimal in A-norm among all the possible approximations in $\mathcal{K}_m(A, \mathbf{r}_0)$. Third, CG method is a short-recurrences method, a short-recurrences method only uses a fixed number of vectors independent from the number of iterations. In particular, CG uses three-term vector recurrences.

[†]Historical note: BiCGStab was first named as CGStab [44].

These three optimal conditions of CG are not present simultaneously in the Krylov methods for solving non-symmetric systems of linear equations. Full GMRES creates an approximation such that the optimal residual vector is minimal in Euclidean norm for all the possible approximation in the Krylov space $\mathcal{K}_m(A, \mathbf{r}_0)$. However, full GMRES is a long-recurrences method, i.e., the number of vectors used by the method depend directly on the number of iterations (or number of matrix-vector products). For this reason, the computational cost of GMRES grows with each iteration. On the other hand, BiCG and BiCGStab use short-recurrences. These methods sacrifice the optimality condition of minimizing the norm of the residuals per iteration for the sake of limiting the CPU and memory consumption.

1.4 Solving the eigenvalue problem

In this section, we turn our attention to describe a Krylov method to approximate eigenvalues and eigenvectors of sparse matrices. Given a matrix A and non-zero initial vector $\mathbf{v}$, the Arnoldi method [4], after m steps, creates a matrix with m orthogonal columns V_m, a upper Hessenberg H_m, and a vector $\mathbf{f}$, such that, $V_m^H \mathbf{f} = \mathbf{0}$. The matrices A, V_m, H_m, and the vector $\mathbf{f}$ hold the so-called Hessenberg decomposition

$$AV_m = V_m H_m + \mathbf{f}\mathbf{e}_m^T. \tag{1.13}$$

Also, the columns of V_m form a basis for the Krylov subspace $\mathcal{K}_m(A, \mathbf{v})$, and it can be proved that

$$V_m^H A V_m = H_m.$$

Hessenberg decompositions, such as (1.13), play an important role in the approximation of eigenvalues and eigenvectors of large and sparse matrices. First, let the pairs $\{(\lambda_i, \hat{\mathbf{y}})\}_{i=0}^m$ be the set of eigenvalues and the corresponding eigenvectors of the matrix H_m. Then, a subset of eigenpairs of the matrix A can be approximated by the pairs $\{(\lambda_i,\, \mathbf{y}_i = V_m \hat{\mathbf{y}})\}_{i=1}^m$. The scalars λ_i are called Ritz-values of the matrix A and the vectors $\mathbf{y}_i$ are the associated Ritz-vectors.

The interest in Arnoldi as eigensolver method is to a large extent owing the work of Y. Saad during the decade of 1980 [80]. A large part of this work was focus in how to improve the accuracy of the Ritz-values and Ritz-vectors, or how to approximate a specific subset of eigenvalues of the matrix A. Also, it is important to limit the use of computational resources by the Arnoldi method when m grows. For this reason, different restarting techniques where applied to the Arnoldi method (see for example [74], [76], [77]). In 1992, D. C. Sorensen presented the implicitly restarted technique for the Arnoldi method. This is an iterative scheme that fixed the dimension m and restarts the Arnoldi algorithm by applying the Francis QR method to the matrix H_m.

In Chapter 2, we combine this implicit restarting technique to our proposed IDR(s) algorithm for eigenvalues. Algorithm 4 presents an implementation of Arnoldi using the Gram-Schmidt process.

Algorithm 4 Arnoldi method

```
 1: Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{x} \neq \mathbf{0} \in \mathbb{C}^n$, $m \in \mathbb{N}^+$.
 2: $\mathbf{v}_1 = \mathbf{x}/\|\mathbf{x}\|$
 3: for $j = 1$ to $m$ do
 4:     $\mathbf{w} = A\mathbf{v}_j$
 5:     for $i = 1$ to $j$ do
 6:         $H_{i,j} = \langle \mathbf{w}, \mathbf{v}_i \rangle$
 7:         $\mathbf{w} = \mathbf{w} - H_{i,j}\mathbf{v}_i$
 8:     end for
 9:     $H_{j+1,j} = \|\mathbf{w}\|$, and $\mathbf{v}_{j+1} = \mathbf{w}/H_{j+1,j}$
10: end for
11: Define $V_m = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$
```

1.5 Solving matrix linear equations

The Krylov methods have been adapted also to solve linear matrix equations of the form (1.5). There are two different kinds of Krylov methods to solve the matrix linear equations. First, the projection methods, that in general project the problem into a smaller Krylov subspace, $\mathcal{K}_m(A, \mathbf{x})$ taking into account the low-rank of the solution (see for example [78] for the particular case of the Lyapunov equation). To accelerate the solution of matrix linear equation, the state-of-the-art projection methods find a solution in the so-called extended Krylov subspace

$$\mathcal{K}_m(A, \mathbf{x}) \oplus \mathcal{K}_m(A^{-1}, \mathbf{x}),$$

where the symbol $\oplus$ represents the sum of subspaces (see for example [27]). The second approach consists of considering the unknown solution matrix as a (long) vector of unknowns and solving a systems of linear equations, where the coefficient matrix is formed from the sum of Kronecker products (see for example [48]).

1.6 Context of this research

The Induced Dimension Reduction method (IDR(s)), since its publication in 2008, has been adapted to different kind of matrix problems for several researchers. In this section, we reference some of the works involving the Induced Dimension Reduction method.

IDR(s) is a family of algorithms to solve systems of linear equations. There exist several ways to translate the IDR(s) Theorem (see Theorem 2.1 in [95]) into practical algorithms. This fact is going to be explained in more detail in the next chapter. Different variants of IDR(s) for solving systems of linear equations have been proposed to improved its convergence, numerical stability and even to adapt it for high performance architectures. Examples of these works are, the IDR(s) with biorthogonal residuals by M. B. van Gijzen and P. Sonneveld [103]; minimum residual IDR(s) variants by M. B. van Gijzen, G. L. G. Sleijpen and J.-P. M. Zemke [102], L. Du, T. Sogabe, and S.-L. Zhang [29], J.-P. M. Zemke [108]; IDR with partial orthogonalization J.-P. M. Zemke [108], IDR(s) has been adapted for exploiting high performance computing architectures by T. P. Collignon and M. B. van Gijzen [23] and also by H. Anzt et al. in [3].

As contributions to the theory of IDR(s), G. L. G. Sleijpen, P. Sonneveld, and M. B. van Gijzen prove that IDR(1) and BiCGStab are mathematically equivalent. Also, they exploited this relation to created IDRstab based on BiCGStab(ℓ) [88]. V. Simoncini and D. B. Szyld proved that this method can also be viewed as a projection method with Petrov-Galerkin conditions [87].

The first adaptation of IDR(s) to approximate eigenvalues was proposed by M. H. Gutknecht and J.-P. M. Zemke [45]. In this work, the authors use the IDR(s) method to construct a generalized Hessenberg decomposition

$$AW_mU_m = W_mH_m + \mathbf{v}\mathbf{e}_m^T, \tag{1.14}$$

where $W_m \in \mathbb{C}^{n\times m}$ is a basis for a Krylov subspace (not explicitly built). U_m is an upper triangular $m \times m$ matrix, $H_m \in \mathbb{C}^{m\times m}$ is a upper Hessenberg matrix, and $\mathbf{f}$ is a vector in $\mathbb{C}^n$. A portion of the eigenvalues of A are approximated by the eigenvalues obtained from the solution of the generalized eigenvalue problem

$$H_m\mathbf{y}_i = \lambda_i U_m\mathbf{y}_i,$$

also known as the eigenvalue pencil (H_m, U_m). Due to the fact that W_m is not explicitly created the eigenvectors are not approximated.

Additional variants of IDR(s) have been adapted to other related problems as, solving multi-shift linear systems (see for example M. B. van Gijzen, G. L. G. Sleijpen, and J.-P. M. Zemke [102], L. Du, T. Sogabe, and S.-L. Zhang [30], and M. Baumann and M. B. van Gijzen [13]), solving block linear systems (L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S.-L. Zhang, [28]), or sequences of system of linear equations (M. P. Neuenhofen [66]).

CHAPTER 2

The Induced Dimension Reduction method

In the previous chapter, we gave a general introduction to the most well-known Krylov subspace methods to solve systems of linear equations and eigenvalue problems. In this chapter, we present a review of the development and theory of the Induced Dimension Reduction method (IDR(s)), which is the core algorithm in this dissertation.

IDR(s) was introduced as a method to solve systems of linear equations in [95] as a generalization of the IDR method first presented in [105]. IDR methods are based on a relatively unusual approach of forcing the residual to be in a sequence of nested and shrinking subspaces, instead of the classical approach of finding the approximation in a subspace of bigger dimension at each iteration.

We present here a detailed introduction to the Induced Reduction Dimension method. In section 2.1, we review the early version of IDR. Then Section 2.2 presents how this early IDR evolves to its generalization the IDR(s) method. We also describe the biortho IDR(s) [103], which is a numerically stable variant of the IDR(s). In Section 2.4, we also present a comparison of the computational behavior of IDR(s) with respect to other short-recurrences Krylov methods, specifically the Bi-Conjugate Gradient Method (BiCG), restarted Generalized Minimal Residual (GMRES(m)), and Bi-Conjugate Gradient Stabilized method (BiCGStab).

2.1 The early Induced Dimension Reduction method

The initial version of IDR was presented in a Symposium held by the International Union of Theoretical and Applied Mechanics at the University of Paderborn in Germany in 1979. In the proceedings of this Symposium, IDR was introduced in the paper titled "Numerical experiments with a multiple grid and a preconditioned Lanczos type method" by P. Wesseling and P. Sonneveld [105]. In this paper, the authorship of the IDR method was explicitly attributed to P. Sonneveld.

P. Sonneveld was trying to generalized the secant method to n dimensions. In his efforts, he proposed the following three terms recurrences equations to solve system of linear equations

$$\gamma_i = \frac{\langle \mathbf{r}_{i-1}, \mathbf{p}\rangle}{\langle \mathbf{r}_{i-1} - \mathbf{r}_{i-2}, \mathbf{p}\rangle} \tag{2.1}$$

$$\mathbf{v}_i = \mathbf{r}_{i-1} - \gamma_i(\mathbf{r}_{i-1} - \mathbf{r}_{i-2}), \tag{2.2}$$

$$\mathbf{r}_i = (I - \omega_j A)\mathbf{v}_i, \tag{2.3}$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \omega_i \mathbf{v}_i + \gamma_i(\mathbf{x}_{i-1} - \mathbf{x}_{i-2}) \tag{2.4}$$

for $i = 2, 3, \ldots$, and $\mathbf{x}_0$ given, and $\mathbf{p} \in \mathbb{C}^n$ a vector such that $\langle \mathbf{r}_0, \mathbf{p}\rangle \neq 0$. The first approximation and its residual is obtained as following

$$\mathbf{x}_1 = \mathbf{x}_0 + \omega_1 \mathbf{r}_0, \tag{2.5}$$

and

$$\mathbf{r}_1 = \mathbf{r}_0 - \omega_1 A \mathbf{r}_0. \tag{2.6}$$

The scalars ω_i are selected every even step as the minimizer of $\|\mathbf{r}_i\|$. The IDR method showed a competitive computational behavior with respect of the mainstream non-symmetric linear solvers at the end of the decade of the 1970s (see for example Figure 2.1).

It was shown that the residuals produced by (2.1)–(2.6) belong to the sequence of subspaces $\mathcal{G}_j$ defined as

$$\mathcal{G}_j \equiv (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S}), \qquad \text{for } j = 1, 2, \ldots. \tag{2.7}$$

Where $\mathcal{S} \equiv \{\mathbf{x} : \langle \mathbf{x}, \mathbf{p}\rangle = 0\}$ or the space orthogonal to the vector $\mathbf{p}$. The sequence of subspaces $\mathcal{G}_j$ has two important properties. First, the subspaces are nested, this is $\mathcal{G}_{j-1} \subset \mathcal{G}_j$. Second, $\mathcal{G}_j \equiv \{\mathbf{0}\}$ for some j. This implies a finite termination behavior (in exact arithmetic) after $2n$ matrix-vector products.

Despite to be almost unnoticed for long time, IDR has a major impact on the development of other well-known Krylov subspace methods as CGS, BiCGStab, and of course its own generalization IDR(s). In the next section, we present a review of IDR(s), which is the core method for the development of the research presented in this document.

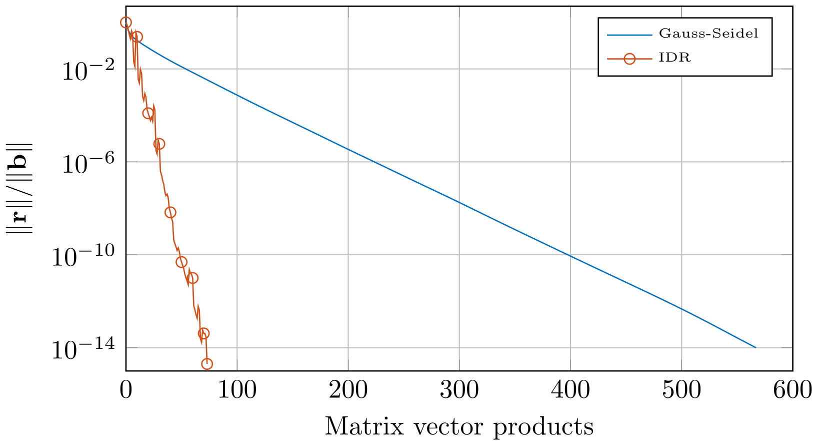

Figure 2.1: Comparison of the converge of Gauss-Seidel and IDR [105] for solving the system of linear equation of order 400 where the coefficient matrix is the Toeplitz tridiagonal matrix `A = tridiag(-1, 3, 1.9)`.

Algorithm 5 Induced Dimension Reduction method

```
1: procedure IDR($A$, $\mathbf{b}$, $tol$, $\mathbf{x}_0$)
2:     Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $tol \in (0, 1)$, $\mathbf{x}_0 \in \mathbb{C}^n$.
3:     Select $\mathbf{p} \in \mathbb{C}^n$
4:     $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\Delta\mathbf{g}_0 = \Delta\mathbf{y}_0 = \mathbf{0} \in \mathbb{C}^n$
5:     $\gamma_0 = 0.0$
6:     for $i = 1, \dots,$ convergence do
7:         $\mathbf{s}_i = \mathbf{r}_{i-1} + \gamma_{i-1}\Delta\mathbf{g}_{i-1}$
8:         $\mathbf{t}_i = A\mathbf{s}_i$
9:         if $i$ is even then
10:            $\omega_i = \frac{\langle \mathbf{s}_i, \mathbf{t}_i\rangle}{\langle \mathbf{t}_i, \mathbf{t}_i\rangle}$
11:        else
12:            $\omega_i = \omega_{i-1}$
13:        end if
14:        $\Delta\mathbf{x}_i = \gamma_{i-1}\Delta\mathbf{y}_{i-1} - \omega_i\mathbf{s}_i$
15:        $\Delta\mathbf{r}_i = \gamma_{i-1}\Delta\mathbf{g}_{i-1} - \omega_i\mathbf{t}_i$
16:        $\mathbf{x}_i = \mathbf{x}_{i-1} + \Delta\mathbf{x}_i$
17:        $\mathbf{r}_i = \mathbf{r}_{i-1} + \Delta\mathbf{r}_i$
18:        if $i$ is even then
19:            $\Delta\mathbf{y}_i = \Delta\mathbf{y}_{i-1}$
20:            $\Delta\mathbf{g}_i = \Delta\mathbf{g}_{i-1}$
21:        else
22:            $\Delta\mathbf{y}_i = \Delta\mathbf{x}_i$
23:            $\Delta\mathbf{g}_i = \Delta\mathbf{r}_i$
24:        end if
25:        $\gamma_i = \frac{\langle \mathbf{r}_i, \mathbf{p}\rangle}{\langle \Delta\mathbf{g}_i, \mathbf{p}\rangle}$
26:    end for
27:    Return $\mathbf{x}_i$.
28: end procedure
```

2.2 The modern IDR(s) method

P. Sonneveld and M. B. van Gijzen revisited the original IDR method [105]. They present the Induced Dimension Reduction method (IDR(s)) in [95] as a generalization of IDR[†]. This is a framework to solve systems of linear equations and is based on the IDR(s) Theorem,

Theorem 1. *[IDR(s) Theorem] Let A be any matrix in $\mathbb{C}^{n\times n}$, let $\mathbf{v}_0$ be any nonzero vector in $\mathbb{C}^n$, and let $\mathcal{G}_0$ be the full Krylov subspace $\mathcal{K}_n(A, \mathbf{v}_0)$. Let $\mathcal{S}$ be any (proper) subspace of $\mathbb{C}^n$ such that $\mathcal{S}$ and $\mathcal{G}_0$ do not share a nontrivial invariant subspace of A, and define the sequence $\mathcal{G}_j$, $j = 1, 2, \ldots$ as*

$$\mathcal{G}_j \equiv (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S})$$

where ω_j's are nonzero scalars. Then

1. *$\mathcal{G}_{j+1} \subset \mathcal{G}_j$, for $j \geq 0$ and*
2. *dimension($\mathcal{G}_{j+1}$) $<$ dimension($\mathcal{G}_j$) unless $\mathcal{G}_j = \{\mathbf{0}\}$.*

Proof. See [95]. □

In a similar way as its predecessor IDR, IDR(s) forces each residual to be in the $\mathcal{G}_j$ spaces, which are shrinking and nested subspaces, while in parallel it obtains the approximate solution to the system of equations. One of the important ideas introduced in [95] is the selection of the shadow space $\mathcal{S}$ as the null space of a hyperplane $P = [\mathbf{p}_1, \mathbf{p}_1, \ldots, \mathbf{p}_s]_{n\times s}$ instead of the null space of a single vector $\mathbf{p}_1$ as in the original IDR. This change provides more reduction in the residual of the problem, for example, IDR(s) requires $n + n/s$ matrix-vector multiplications to obtain the solution of the system of linear equations in exact arithmetic, in contrast to the $2n$ matrix-vector products required by IDR under the same conditions.

In order to describe the recurrence formulas of IDR(s), let us first consider the general Krylov subspace recurrences, these are

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k - \sum_{i=1}^{\hat{l}} \gamma_i \Delta\mathbf{x}_{k-i} \tag{2.8}$$

and the residual

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{v}_k - \sum_{i=1}^{\hat{l}} \gamma_i \Delta\mathbf{r}_{k-i}, \tag{2.9}$$

[†]Historical note: In 2006, Dr. J.-P. M. Zemke from Hamburg-Harburg University of Technology emailed to P. Sonneveld asking what had happened with the old IDR method. This question revives the interest of P. Sonneveld in IDR.

where $\Delta\mathbf{u}_k$ is the forward difference operator defined as $\Delta\mathbf{u}_k = \mathbf{u}_{k+1} - \mathbf{u}_k$, and the vector $\mathbf{v}_k$ is a computable vector in the Krylov subspace $\mathcal{K}_k(A, \mathbf{r}_0) \setminus \mathcal{K}_{k-1}(A, \mathbf{r}_0)$. The integer $\hat{l}$ represents the depth of the recursions. When $\hat{l} = n$, (2.8) and (2.9) are called long-recurrences. As an example of a method that uses long-recurrences we have GMRES, and the main drawback of this type of method is that the computational requirements grow with k. A short-recurrence method uses a small fixed value for $\hat{l}$. Examples of this kind of methods are BiCG, BiCGStab, and CGS among others.

IDR(s) is also a short-recurrence method. It uses recursions of $s+1$ vectors (typical choices are $s < 20$), this is $\hat{l} = s$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k - \sum_{i=1}^{s} \gamma_i \Delta\mathbf{x}_{k-i} \tag{2.10}$$

and the residual

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{v}_k - \sum_{i=1}^{s} \gamma_i \Delta\mathbf{r}_{k-i}. \tag{2.11}$$

Selecting the vector $\mathbf{v}_k$ as

$$\mathbf{v}_k = \mathbf{r}_k - \sum_{i=1}^{s} \gamma_i \Delta\mathbf{r}_{k-i}, \tag{2.12}$$

we can rewrite (2.11) as

$$\mathbf{r}_{k+1} = (I - \alpha_k A)\mathbf{v}_k. \tag{2.13}$$

In order to find the scalars in (2.10) and (2.11), we impose the condition that $\mathbf{r}_{k+1} \in \mathcal{G}_{j+1}$. As one can see in (2.13), we have that $\alpha_k = \omega_{j+1}$, and we have to impose that $\mathbf{v}_k \in \mathcal{G}_j \cap \mathcal{S}$. This leads to the solution of the following $s \times s$ system of linear equations

$$([\mathbf{p}_1, \mathbf{p}_1, \ldots, \mathbf{p}_s])^H [\Delta\mathbf{r}_{k-s}, \ldots, \Delta\mathbf{r}_{k-1}]\mathbf{c} = ([\mathbf{p}_1, \mathbf{p}_1, \ldots, \mathbf{p}_s])^H \mathbf{r}_k, \tag{2.14}$$

where $\mathbf{c} = [\gamma_1, \gamma_2, \ldots, \gamma_s]$. Using these calculations, IDR(s) obtains $\mathbf{r}_{k+1}$ in the subspace $\mathcal{G}_{j+1}$, then using the fact that $\mathcal{G}_{j+1} \subset \mathcal{G}_j$, IDR($s$) repeats this calculations to obtain at least $s+1$ residual vectors in $\mathcal{G}_{j+1}$. Then, IDR(s) is ready to create new residual vector in $\mathcal{G}_{j+2}$ with their respective approximations.

The parameter $\alpha \equiv \omega_{j+1}$ can be chosen freely for the first residual vector in a $\mathcal{G}_{j+1}$ space, however, the same value should be used for the computation of the other residual vectors in $\mathcal{G}_{j+1}$. There are different options to select the value ω_{j+1}, for example, the value that minimizes the norm of $\mathbf{r}_{k+1}$, the "maintaining the convergence" strategy proposed in [90] and used in IDR(s) in [103], or the inverse of the Ritz-values as is proposed in [87]. A prototype implementation of the IDR(s) is shown in Algorithm 6.

Algorithm 6 IDR(s) algorithm

```
1: procedure IDR($A$, $\mathbf{b}$, $s$, $\mathbf{x}_0$)
2:     Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $s \in \mathbb{N}^+$, $\mathbf{x}_0 \in \mathbb{C}^n$.
3:     $P$ a random matrix in $\mathbb{C}^{n\times s}$.
4:     $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
5:     for $j = 0, \ldots, s-1$ do          ▷ $s$ minimum norm steps to get $s$ extra vectors in $\mathcal{G}_0$
6:         $\mathbf{v} = A\mathbf{r}_j$; $\omega = \frac{\langle \mathbf{r}_j, \mathbf{v}\rangle}{\langle \mathbf{v}, \mathbf{v}\rangle}$
7:         $\Delta\mathbf{x}_j = \omega\mathbf{r}_j$; $\Delta\mathbf{r}_j = -\omega\mathbf{v}$
8:         $\mathbf{r}_{j+1} = \mathbf{r}_j + \Delta\mathbf{r}_j$; $\mathbf{x}_{j+1} = \mathbf{x}_j + \Delta\mathbf{x}_j$
9:     end for
10:    $\Delta R_{j+1} = [\Delta\mathbf{r}_j, \ldots, \Delta\mathbf{r}_0]$; $\Delta X_{n+1} = [\Delta\mathbf{x}_j, \ldots, \Delta\mathbf{x}_0]$
11:    $j = s$
12:    while not convergence do                          ▷ Building the $\mathcal{G}_j$ spaces
13:        for $k = 0$ to $s$ do                         ▷ Loop inside the $\mathcal{G}_j$ spaces
14:            Solve $\mathbf{c}$ from $P^H \Delta R_n \mathbf{c} = P^H \mathbf{r}_n$
15:            $\mathbf{v} = \mathbf{r}_j - \Delta R_j \mathbf{c}$
16:            if $k =$ then
17:                $\mathbf{t} = A\mathbf{v}$
18:                $\omega = \frac{\langle \mathbf{v}, \mathbf{t}\rangle}{\langle \mathbf{t}, \mathbf{t}\rangle}$
19:                $\Delta\mathbf{r}_j = -\Delta R_j \mathbf{c} - \omega\mathbf{t}$
20:                $\Delta\mathbf{x}_j = -\Delta X_j \mathbf{c} + \omega\mathbf{v}$
21:            else
22:                $\Delta\mathbf{x}_j = -\Delta X_j \mathbf{c} + \omega\mathbf{v}$
23:                $\Delta\mathbf{r}_j = -A\Delta\mathbf{x}_j$
24:            end if
25:            $\mathbf{r}_{j+1} = \mathbf{r}_j + \Delta\mathbf{r}_j$
26:            $\mathbf{x}_{j+1} = \mathbf{x}_j + \Delta\mathbf{x}_j$
27:            $j = j + 1$
28:            $\Delta R_j = [\Delta\mathbf{r}_{j-1}, \ldots, \Delta\mathbf{r}_{j-s}]$
29:            $\Delta X_j = [\Delta\mathbf{x}_{j-1}, \ldots, \Delta\mathbf{x}_{j-s}]$
30:        end for
31:    end while
32:    Return $\mathbf{x}_j$.
33: end procedure
```

2.3 Generalization of the IDR(s) recurrences

Residual vectors of IDR(s) are uniquely defined every ($s+1$)th step (matrix-vector multiplications), nevertheless there is some freedom during the creation of the intermediate residuals (see section 5.1 of [95]). In exact arithmetic, the selection of the intermediate residuals does not affect the residuals obtained at every $s+1$ matrix-vector multiplications. However, this is not the case for numerical computations. The selection of the intermediate residuals affects the residuals at every ($s+1$)th step and also the stability and efficiency of IDR(s).

Before, we describe biortho IDR(s), which is the IDR(s) variant used in this dissertation, we present a general framework to obtain different variants of IDR(s). In (2.10) and (2.11), it is not necessary to use the set of vectors

$\Delta\mathbf{x}_i$ and $\Delta\mathbf{r}_i$. These equations can be generalized as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \omega_{j+1}\mathbf{v}_k + \sum_{i=1}^{s} \gamma_i \mathbf{u}_{k-i}$$

and the residual in $\mathcal{G}_{j+1}$ as

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \omega_{j+1} A\mathbf{v}_k - \sum_{i=1}^{s} \gamma_i \mathbf{g}_{k-i}.$$

where

$$\mathbf{u}_i = A\mathbf{g}_i, \qquad \text{with } \mathbf{g}_i \in \mathcal{G}_j \qquad \text{for } i = k-s, \dots, k-1. \tag{2.15}$$

The vectors $\mathbf{u}_i$ and $\mathbf{g}_i$ are called the direction vectors. It is possible to exploit the fact that other residuals and the vectors $\mathbf{g}_i$ have been already created in the same subspace $\mathcal{G}_{j+1}$ and any linear combination of these vectors is also in $\mathcal{G}_{j+1}$. Let us assume that m residual vectors have been created in $\mathcal{G}_{j+1}$ with their respective direction vectors and approximations. We can generalize the IDR(s) formulas as

$$\mathbf{v}_{k+m} = \mathbf{r}_{k+m} - \sum_{i=1}^{s} \gamma_i \mathbf{g}_{k-m-i}, \tag{2.16}$$

where the scalars $\{\gamma_i\}_{i=1}^{s}$, akin to (2.14), are obtained from the solution of the system of linear equations

$$([\mathbf{p}_1,\, \mathbf{p}_1, \dots, \mathbf{p}_s])^H [\mathbf{g}_{k+m-1},\, \dots, \mathbf{g}_{k+m-s}]\mathbf{c} = ([\mathbf{p}_1,\, \mathbf{p}_1, \dots, \mathbf{p}_s])^H \mathbf{r}_{k+m}. \tag{2.17}$$

The direction vectors are computed first as

$$\hat{\mathbf{u}}_{k+m} = \sum_{i=1}^{s} \gamma_i \mathbf{u}_{k-m-i} + \omega_{j+1}\mathbf{v}_{k+m}, \tag{2.18}$$

and

$$\hat{\mathbf{g}}_{k+m} = A\mathbf{u}_{k+m},$$

then using the fact that $\mathbf{g}_{k+i}$ are in $\mathcal{G}_{j+1}$ for $i = 1, \dots, m-1$, we can update any vector $\mathbf{g}_{k+m}$ as following

$$\mathbf{g}_{k+m} = \hat{\mathbf{g}}_{k+m} - \sum_{i=1}^{m-1} \alpha_i \mathbf{g}_{k+m}, \qquad \text{then } \mathbf{g}_{k+m} \in \mathcal{G}_{j+1}, \tag{2.19}$$

and to maintain the relation (2.15), the vector $\mathbf{u}_{k+i}$ should be updated as

$$\mathbf{u}_{k+m} = \hat{\mathbf{u}}_{k+m} - \sum_{i=1}^{m-1} \alpha_i \mathbf{u}_{k+i}. \tag{2.20}$$

Also the formulas for the approximation and the residual vector are generalized as

$$\mathbf{r}_{k+m+1} = \mathbf{r}_{k+m} - \sum_{i=1}^{m} \beta_i \mathbf{g}_{k+i}, \tag{2.21}$$

and

$$\mathbf{x}_{k+m+1} = \mathbf{x}_{k+m} + \sum_{i=1}^{m} \beta_i \mathbf{u}_{k+i} \tag{2.22}$$

The different implementations for the IDR(s) method are based on the different ways to create the intermediate residuals selecting the parameters α_i and β_i (see for example [29], [103], [102], and [108]).

2.3.1 IDR(s) with biorthogonal residuals

In this section, we describe the IDR(s) with biorthogonal residuals or biortho-IDR(s). In this variant, the freedom in (2.19) and (2.21) is filled with the following conditions,

$$\langle \mathbf{g}_{k+m}, \mathbf{p}_i \rangle = 0 \qquad \text{for } i = 1, 2, \ldots, m-1, \tag{2.23}$$

and

$$\langle \mathbf{r}_{k+m+1}, \mathbf{p}_i \rangle = 0 \qquad \text{for } i = 1, 2, \ldots, m. \tag{2.24}$$

The imposition of the conditions (2.23) and (2.24) leads to important changes with respect the original IDR(s) variant presented in [95]. Firstly, the initial residual vector in $\mathcal{G}_{j+1}$ can be written as

$$\mathbf{r}_{k+1} = (I - \omega_{j+1} A)\mathbf{r}_k, \tag{2.25}$$

with its approximation

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \omega_{j+1}\mathbf{r}_k. \tag{2.26}$$

There are other simplifications for the creation of the subsequent residuals $\mathbf{r}_{j+m}$ in the subspace $\mathcal{G}_{j+1}$. For example, the coefficient matrix of system of linear equations (2.17) becomes a lower triangular matrix, and its right-hand size vector has zeros in its first m entries. For this reason, (2.16) can be written as

$$\mathbf{v}_{k+m} = \mathbf{r}_{k+m} - \sum_{i=k}^{s} \gamma_i \mathbf{g}_{k-m-i}, \tag{2.27}$$

and the direction vectors are expressed as (2.18)–(2.20), selecting the parameters α_i such that (2.23) holds. Lastly, we can obtain $\mathbf{r}_{k+m+1}$ and with its corresponding approximation $\mathbf{x}_{k+m+1}$ as

$$\mathbf{r}_{k+m+1} = \mathbf{r}_{k+m} - \beta_m \mathbf{g}_{k+m}, \tag{2.28}$$

Algorithm 7 IDR(s) with biorthogonal residuals algorithm

```
1: procedure IDR($A$, $\mathbf{b}$, $s$, $\mathbf{x}_0$)
2:     Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $s \in \mathbb{N}^+$, $\mathbf{x}_0 \in \mathbb{C}^n$.
3:     $P$ a random matrix in $\mathbb{C}^{n\times s}$.
4:     $\mathbf{x} = \mathbf{x}_0$, $\mathbf{r} = \mathbf{b} - A\mathbf{x}$
5:     $G = 0 \in \mathbb{C}^{n\times s}$, $U = 0 \in \mathbb{C}^{n\times s}$.
6:     $M = I_s \in \mathbb{C}^{s\times s}$.
7:     $\omega = 1.0$
8:     while not convergence do                                   ▷ Loop over $\mathcal{G}_j$ spaces
9:         $\mathbf{f} = P^H\mathbf{r}$
10:        for $k = 1$ to $s$ do              ▷ Compute $s$ independent vectors $\mathbf{g}_k$ in $\mathcal{G}_j$ space
11:            Solve $\mathbf{c}$ from $M\mathbf{c} = \mathbf{f}$, $(\gamma_1, \ldots, \gamma_s)^H = \mathbf{c}$            ▷ Note that $M = P^H G$
12:            $\mathbf{v} = \mathbf{r} - \sum_{i=k}^{s} \gamma_i \mathbf{g}_i$
13:            $\mathbf{v} = B^{-1}\mathbf{v}$                                ▷ Preconditioning operation
14:            $\mathbf{u}_k = \omega\mathbf{v} + \sum_{i=k}^{s} \gamma_i \mathbf{g}_i$
15:            $\mathbf{g}_k = A\mathbf{u}_k$
16:            for $i = 1$ to $k-1$ do                              ▷ Make $\mathbf{g}_k$ orthogonal to $P$
17:                $\alpha_i = \langle \mathbf{g}_k, \mathbf{p}_i \rangle / \mu_{i,i}$
18:                $\mathbf{g}_k = \mathbf{g}_k - \alpha_i \mathbf{g}_i$
19:                $\mathbf{u}_k = \mathbf{u}_k - \alpha_i \mathbf{u}_i$
20:            end for
21:            $M_{i,k} = \langle \mathbf{g}_k, \mathbf{p}_i \rangle$ for $i = k, \ldots, s$                     ▷ Update $M$
22:            $\beta_k = f_k / M_{k,k}$         ▷ Make the residual orthogonal to $\mathbf{p}_i$ for $i = 1, \ldots, k$
23:            $\mathbf{r} = \mathbf{r} - \beta_k \mathbf{g}_k$
24:            $\mathbf{x} = \mathbf{x} + \beta_k \mathbf{u}_k$
25:            if $k + 1 \leq s$ then
26:                $\mathbf{f}_i = 0$ for $i = 1, \ldots, k$
27:                $\mathbf{f}_i = \mathbf{f}_i - \beta_k M_{i,k}$ for $i = k+1, \ldots, s$
28:            end if
29:            Overwrite $k$th columns of $G$ and $U$ by $\mathbf{g}_k$ and $\mathbf{u}_k$ respectively.
30:        end for                                                    ▷ Entering $\mathcal{G}_{j+1}$
31:        $\mathbf{v} = B^{-1}\mathbf{r}$                                        ▷ Preconditioning operation
32:        $\mathbf{t} = A\mathbf{v}$
33:        $\omega$ is selected using the converge maintenance strategy [103].
34:        $\mathbf{r} = \mathbf{r} - \omega\mathbf{t}$
35:        $\mathbf{x} = \mathbf{x} + \omega\mathbf{v}$
36:    end while
37:    Return $\mathbf{x}$.
38: end procedure
```

and

$$\mathbf{x}_{k+m+1} = \mathbf{x}_{k+m} + \beta_m \mathbf{u}_{k+m}. \tag{2.29}$$

The parameter β_m is selected such that $\langle \mathbf{r}_{k+m+1}, \mathbf{p}_m \rangle = 0$, with this selection of the scalars α_i and β_i both conditions (2.23) and (2.24) hold. Algorithm 7 shows an implementation of biortho-IDR(s).

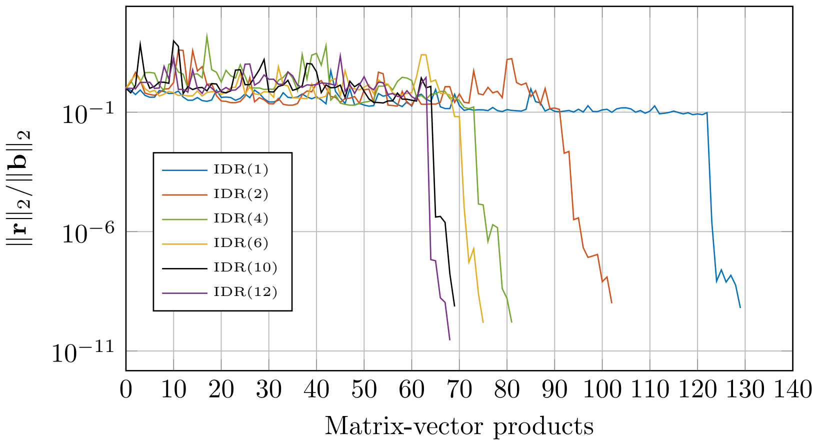

Figure 2.2: Illustration of the finite termination property of IDR(s). The system of linear equations of size 60×60 is obtained from the discretization of (2.30). One can see the drastic reduction in the norms at every $60 + 60/s$ matrix vector products

2.3.2 Three properties of the IDR(s) method

In this section, we remark three important properties of IDR(s) method,

1. Under the assumptions of Theorem 1, IDR(s) finds the exact solution of the linear system of equations $A\mathbf{x} = \mathbf{b}$ after $n + n/s$ matrix-vector products using exact arithmetic (see Corollary 3.2 in [95]). To illustrate this phenomenon, let us consider the example 6.1 in [95]. The system of linear equations arises from the discretization of the 1D convection-diffusion equation

$$-\frac{d^2u}{dx^2} + w\frac{du}{dx} = 0 \qquad \text{for } x \in (0,\, 1), \tag{2.30}$$

with the Dirichlet boundary conditions $u(0) = u(1) = 1$, using central finite differences with mesh size $h = \frac{1}{61}$, and the convection parameter $w = 61$. Figure 2.2 shows the evolution of residual norm as function of the number of the matrix-vector products. In this figure, one can see the drastic reduction of the residual norms, at the points where the finite termination of the IDR(s) should occur for the specific parameter s.

2. In [89], the authors showed that IDR(s) with biorthogonal residual with $s = 1$ is mathematically equivalent to the BiCGStab method.

3. For difficult problems, IDR(s) outperforms BiCGStab for $s > 1$, an example of this is the convection-diffusion equation in the convection dominated case. In section 2.4, we present several numerical tests to exemplify

this behavior. Also in section 2.4.3, we present an analysis of the residuals of IDR(s) and BiCGStab in terms of the polynomials of the matrix A.

2.3.3 Polynomial residual formulas

An important property of the residual formulas in IDR(s) to be used in the next chapter this dissertation is that any residual vector $\mathbf{r}_k$ in $\mathcal{G}_j$ can be written as

$$\mathbf{r}_k = \Omega_j(A)\Psi(A)_{k-j}\mathbf{r}_0, \tag{2.31}$$

where

$$\Omega_j(t) = \prod_{i=0}^{j}(1-\omega_i t), \quad \omega_i \neq 0, \quad i = 1, \dots, j, \tag{2.32}$$

$\Omega_0(t) = 1$, and $\Psi_m(t)$ is a multi-Lanczos-type polynomial [106] of order m, that uses $s+2$ terms recurrences such that $\Psi_0 = 1$ (see section 5 in [95]).

2.3.4 IDR(s) as a Petrov-Galerkin method

V. Simoncini and D. B. Szyld showed that IDR(s) can also be viewed as a Petrov-Galerkin method in [87]. Particularly IDR(s) finds the approximation $\mathbf{x}_{k+1}$ in the right or search subspace $\mathbf{x}_0+\mathcal{K}_{k+1}(A, \mathbf{r}_0)$, by imposing the condition that $\mathbf{r}_{k+1}$ is orthogonal to the subspace $\mathcal{W}_j$, defined as

$$\mathcal{W}_j = \Omega_j(A^H)^{-1}\mathcal{K}_j(A^H, P), \tag{2.33}$$

where $\Omega_j(A)$ is the polynomial defined in (2.32), and $\mathcal{K}_j(A^H, P)$ is the block Krylov subspace of order j, associated with the matrix A and the block P.

2.4 Numerical experiments

In this section, we consider the following simple convection-diffusion-reaction model problem

$$-\epsilon\triangle u + \mathbf{v}^T\nabla u + \rho u = f, \qquad \text{in } \Omega = [0,\, 1]^d \tag{2.34}$$

with $d = 2$ or $d = 3$, and Dirichlet boundary conditions $u = 0$ on $\partial\Omega$. In (2.34), u represents the concentration of solute, $\mathbf{v} \in \mathbb{R}^d$ is the velocity of the medium or convection vector, $\epsilon > 0$ represents the diffusion coefficient, ρ the reaction coefficient, and f represents the source-term function.

We compare the computational behavior of the Induced Dimension Reduction method (IDR(s)), with other short-recurrences Krylov methods, specifically the Bi-Conjugate Gradient Method (BiCG), GMRES(m), and Bi-Conjugate Gradient Stabilized method (BiCGStab) for solving systems of linear equations derived from the discretization of (2.34). It is well known that the convergence rate of the Krylov methods is strongly influenced by the numerical properties of the coefficient matrix A, which depend on the physical parameters of (2.34). For example, in the convection-dominated case, i.e., $\|\mathbf{v}\| \gg \epsilon$, the coefficient matrix A has almost purely imaginary eigenvalues and this can slow down the convergence of Krylov methods.

All the experiments presented in this section are the discretization of (2.34) with homogeneous Dirichlet boundary conditions over the unit cube, The right-hand-side function f is defined by the solution $u(x, y, z) = x(1-x)y(1-y)z(1-z)$. We use as stopping criterion that

$$\frac{\|\mathbf{b} - A\mathbf{x}_k\|}{\|\mathbf{b}\|} < 10^{-8}.$$

The discretization of (2.34) using central finite differences may produce non-physical oscillations in the numerical solution of convection or reaction-dominated problems. This problem can be solved by discretizing the convection term using upwind schemes. However, we use central finite differences rather than upwind discretization in this set of problems, to illustrate the effect of unfavorable numerical conditions for the Krylov subspace solvers.

The experiments presented in this chapter are based on the article:

R. Astudillo and M. B. van Gijzen. The Induced Dimension Reduction method applied to convection-diffusion-reaction problems. In *Numerical Mathematics and Advanced Applications ENUMATH 2015*, B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, and Ö. Uğur, editors, Cham, ZG, Switzerland, 2016, pages 295–303. Springer .

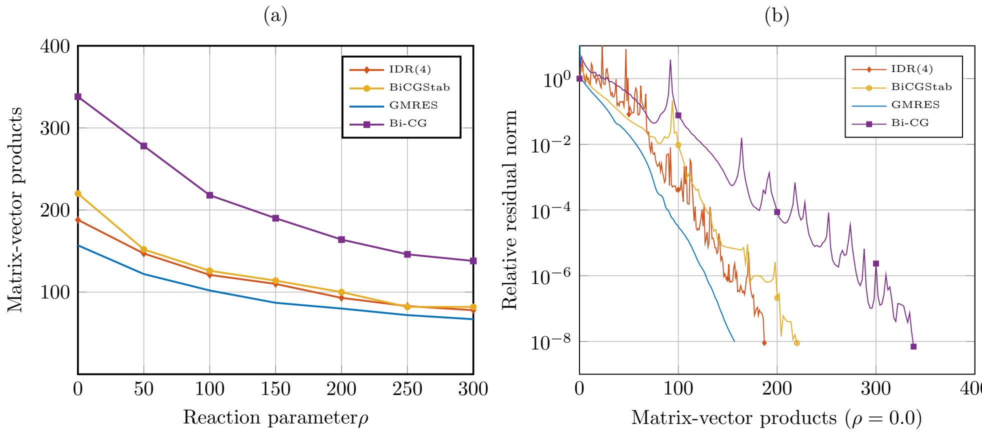

Figure 2.3: *Experiment 2.1.* (a) Number of matrix-vector products required to converge as a function of the parameter ρ for a diffusion-dominated problem. (b) Comparison of the residual norms. The physical parameters are $\epsilon = 1.0$, $\mathbf{v} = (1.0,\, 1.0,\, 1.0)^T/\sqrt{3}$, and $\rho = 0.0$.

Experiment 2.1. In this example, we consider the parameters $\epsilon = 1.0$ and $\mathbf{v} = (1.0,\, 1.0,\, 1.0)^T/\sqrt{3}$. We want to illustrate the effect of non-negative reaction parameter over the Krylov solver, then, we select $\rho \in \{0,\, 50, \dots,\, 300\}$. Figure 2.3 (a) shows the number of matrix-vector multiplication required for each Krylov method as a function of the reaction parameter ρ. In these problems, the increment of the reaction parameter produces a reduction in the number of matrix-vector products required for each Krylov method. All the methods perform very efficiently for these examples. Figure 2.3 (b) shows the evolution of the residual norm for $\rho = 0.0$. The execution times are: IDR(4) 0.62s, BiCGStab 0.64s, BiCG 0.92s, and GMRES 2.83s.

Experiment 2.2. In order to illustrate the effect of the magnitude of the convection velocity, we consider $\epsilon = 1.0$, $\rho = -50.0$, and $\mathbf{v} = \beta(1.0,\, 1.0,\, 1.0)^T/\sqrt{3}$ with $\beta \in \{100.0,\, 200.0, \dots, 800.0\}$. As the parameter β grows we obtain a more convection-dominated problem. Figure 2.4 (a) shows how many matrix-vector products are required for each Krylov method as function of the convection speed. The problem is more convection-dominated as $\|\mathbf{v}\|$ grows. It is interesting to remark the linear growth of the number of matrix-vector product for BiCGStab. Figure 2.4 (b) shows the evolution of the residual norm for $\beta = 800.0$. The execution times are IDR(4) 1.24s, BiCGStab 5.64s, BiCG 1.01s, and GMRES 3.26s.

Experiment 2.3. Here we use the same set of problems presented in experiment 1, but selecting negative reaction parameters, we consider
$\rho \in \{-300,\, -250, \dots,\, -50\}$. In Figure 2.5 (a), one can see how negative

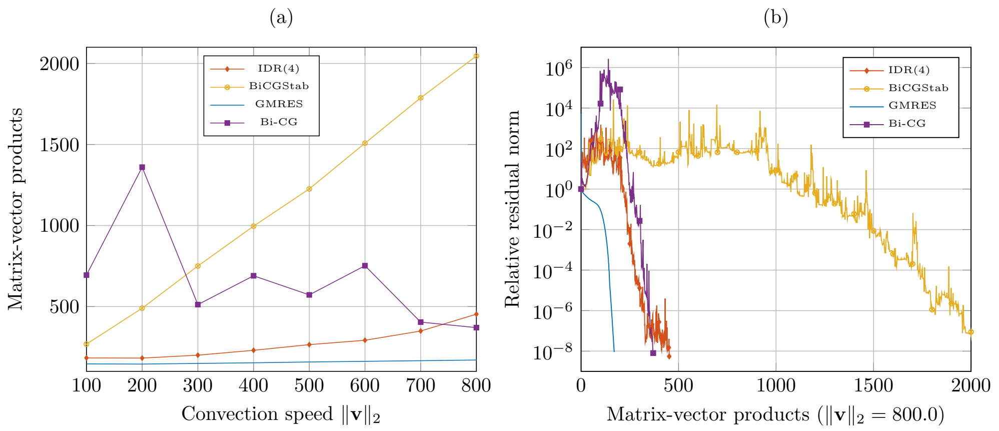

Figure 2.4: *Experiment 2.2.* (a) Number of matrix-vector products required to converge as a function of the convection speed. (b) Comparison of the residual norms. The physical parameters are $\epsilon = 1.0$, $\mathbf{v} = 800.0 \times (1.0,\ 1.0,\ 1.0)^T/\sqrt{3}$, and $\rho = -50.0$.

reaction parameters generate a considerable increment of the matrix-vector multiplications needed for solving the corresponding linear system. BiCGStab performs poorly for large negative reaction parameter. Figure 2.5 (b) shows the evolution of the residual norm for $\epsilon = 1$ and $\rho = 300.0$. The execution times are: IDR(4) 4.02s, BiCGStab 15.38s, BiCG 3.52 s, and GMRES 28.57s.

2.4.1 IDR(s) and BiCG

Despite being a method that is not drastically affected by the increment of the reaction parameter or the convection speed, BiCG is not the faster method in terms matrix-vector products required. BiCG requires two matrix-vector multiplications to produce one new approximation. IDR(4) in most of the experiments requires less matrix-vector multiplication to get the desired residual tolerance. Only in the highly convection-dominated examples presented in Experiment 2.2, BiCG presents a similar behavior as IDR(4). A discussion of the phenomena is presented in section 2.4.3.

2.4.2 IDR(s), GMRES, and restarted GMRES

In the numerical experiments presented in the previous section, full GMRES is the methods that uses less matrix-vector products to obtain the desired residual reduction. This result is expected because the optimal residual condition of GMRES. Nevertheless, the computational requirements of full GMRES grow in every iteration. Restarting GMRES or GMRES(m) is an option to overcome this issue. The idea of GMRES(m) is to limit to a maximum of m matrix-vector products, and then restart the process using the last approximation as

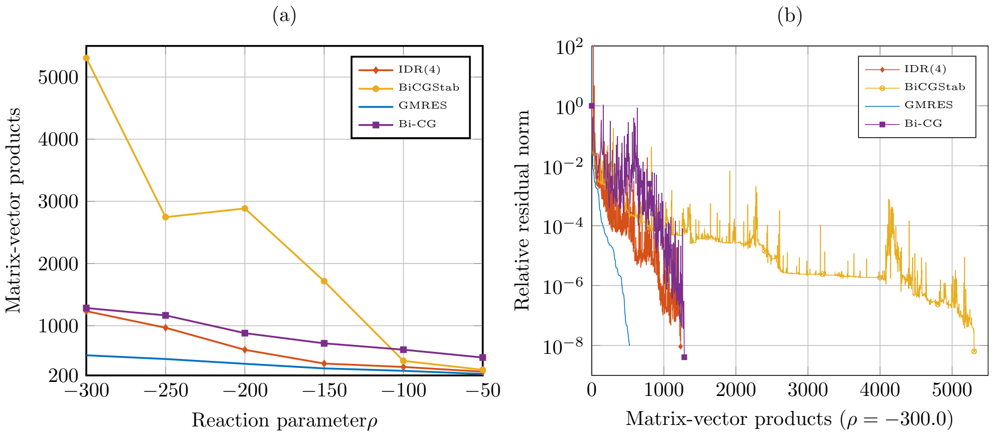

Figure 2.5: *Experiment 2.3.* (a) Number of matrix-vector products required to converge as a function of the parameter ρ. (b) Comparison of the residual norms. The physical parameters are $\epsilon = 1$, $\mathbf{v} = (1.0,\ 1.0,\ 1.0)^T/\sqrt{3}$, and $\rho = -300.0$.

initial vector. The optimal residual property is lost in this restarted scheme.

In terms of memory consumption, GMRES(m) is equivalent to IDR(s) when $m = 3(s+1)$. In order to compare the behavior of GMRES(m) and IDR(4), we consider the discretization of (2.34) with parameters: $\epsilon = 1$, $\mathbf{v} = (1.0,\ 1.0,\ 1.0)^T/\sqrt{3}$ and $\rho = 40.0$, and we take as restarted parameter $m = 15,\ 16, \ldots, 170$. Figure 2.6 shows the number of matrix-vector multiplication required for GMRES(m) for different values of m. GMRES(160) and IDR(4) solve this system using the same number of matrix-vector products (262), however, GMRES(160) consumes approximately ten times more memory than IDR(4). Moreover, CPU time for GMRES(160) is 4.60s while IDR(4) runs in only 0.79s.

2.4.3 IDR(s) and BiCGStab

One can see in the experiments that BiCGStab performs poorly for convection-dominated problems. This can be explained throughout the study of the residual formulas for BiCGStab. BiCGStab uses two matrix-vector products per iteration, and the residual vector in BiCGStab can be written as

$$\mathbf{r}_k^{(B)} = \Omega_k(A)\phi_k(A)\mathbf{r}_0,$$

where $\phi_k(t)$ is residual associated with BiCG and $\Omega_k(t)$ is also known as the Minimal Residual (MR) polynomial defined in (2.32).

The parameter ω_k are selected such that $\|\mathbf{r}_k^{(B)}\|$ is minimized. However, for indefinite matrices or real matrices that have non-real eigenvalues with an imaginary part that is large relative to the real part, the parameter ω_k is close

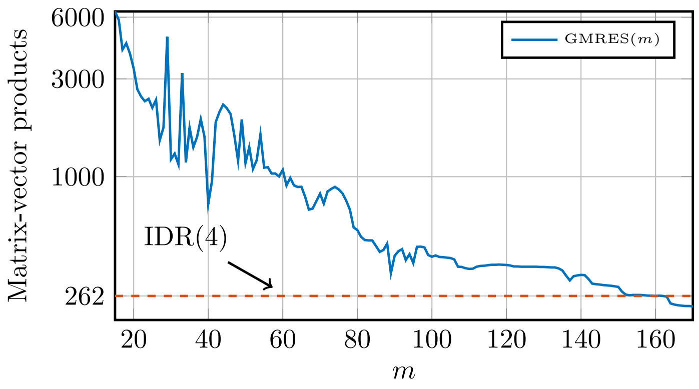

Figure 2.6: (GMRES(m) and IDR(s) comparison) Number of matrix-vector products required for GMRES(m) as a function of the parameter m.

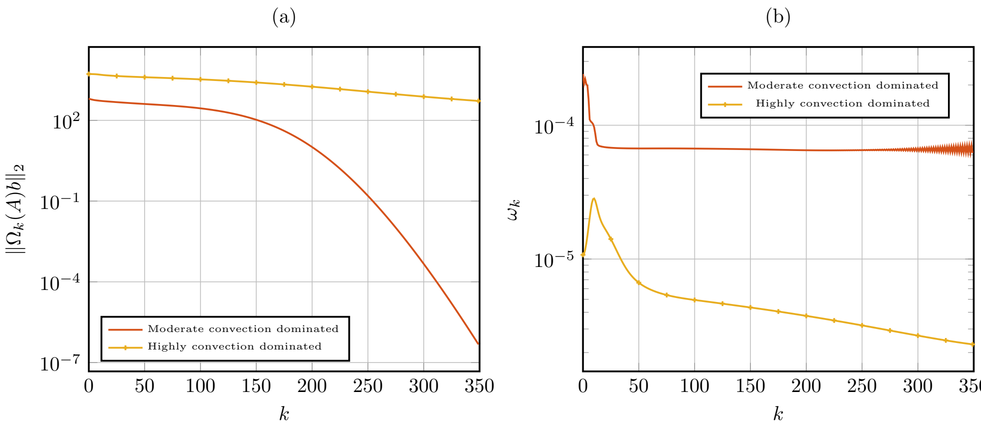

Figure 2.7: (a) Behavior of the norm of the MR-polynomial $\Omega_k(A)$. (b) Values of the parameter ω_k.

to zero (see [90]), and the MR-polynomial suffers from slow convergence or numerical instability. To illustrate this we show the behavior of the polynomial $\Omega_k(A)$ applied to two different matrices from the second set of experiments. We consider $\beta = 100.0$ and $\beta = 800.0$ labeled in Figure 2.7 as moderate convection-dominated and highly convection-dominated respectively.

The convergence of IDR(s) is also affected by the convection speed for a similar reason as was explained for BiCGStab. According to (2.31), the IDR(s) residual vector $\mathbf{r}_k$ in the subspace $\mathcal{G}_j$ can be written as

$$\mathbf{r}_k^{(I)} = \Omega_j(A)\psi_{k-j}(A)\mathbf{r}_0,$$

where $\psi_{k-j}(t)$ is a multi-Lanczos polynomial. For IDR(s) the degree of the polynomial $\Omega_k(t)$ increases by one every $s+1$ matrix-vector products, while in

BiCGStab the degree of the MR polynomial grows by one every two matrix-vector products (one iteration). For this reason, IDR(s) controls the negative effects of the MR-polynomial when A has complex spectrum or is an indefinite matrix.

The bad convergence for convection-dominated problems of BiCGStab has been observed by several authors, and it has given rise to BiCGStab(ℓ) [88]. This method uses polynomial factors of degree ℓ, instead of MR-polynomial. A similar strategy has been implemented in IDR(s) which led to the method IDRstab [92]. For the comparison of the convergence of BiCGStab(ℓ) and IDRstab with IDR(s) we refer the reader to [92].

2.5 Discussion and remarks

In the first part of this chapter, we have presented a historical review of its evolution, the most important of its variants, its numerical properties, and implementations.

Throughout the numerical experiments, we have shown that IDR(s) is a competitive option to solve system of linear equations arising in the discretization of the convection-diffusion-reaction equation.

GMRES, BiCG, and IDR(s) exhibit a stable behavior in the most numerically difficult examples conducted in this work. Despite performing more matrix-vector products to obtain convergence, IDR(s) consumes less CPU time than GMRES. We show that for diffusion-dominated problems with a positive reaction term the convergence of the BiCGStab and IDR(s) are very similar, and for this kind of problems it is often preferable to simply choose $s = 1$. However, for the more difficult to solve convection-dominated problems, or problems with a negative reaction term, IDR(s), with $s > 1$ greatly outperforms BiCGStab.

CHAPTER 3

Induced Dimension Reduction method for solving eigenvalue problems

This chapter presents a new algorithm to compute eigenpairs of large non-symmetric matrices. Using the Induced Dimension Reduction method (IDR(s)), we obtain a Hessenberg decomposition, from which we approximate the eigenvalues and eigenvectors of a matrix. This decomposition has two main advantages. First, IDR(s) is a short-recurrence method, which is attractive for large scale computations. Second, the IDR(s) polynomial used to create this Hessenberg decomposition is also used as a filter to discard the unwanted eigenvalues. Additionally, we incorporate the implicit restarting technique proposed by D. C. Sorensen [97], to approximate specific portions of the spectrum and improve the convergence.

This chapter is based on the articles:

R. Astudillo and M. B. van Gijzen. A restarted Induced Dimension Reduction method to approximate eigenpairs of large unsymmetric matrices. *J. Comput. Appl. Math.*, 296: 24–35, 2016 ,

R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction Method to solve the quadratic eigenvalue problem. In *Numerical Analysis and its Applications: 6th International Conference, NAA 2016, Lozenetz, Bulgaria, June 15-22, 2016, Revised Selected Papers*, I. Dimov, L. Vulkov, and I. Faragó, editors, Cham, ZG, Switzerland, 2017, pages 203–211. Springer .

3.1 Introduction

A variety of applications involve the solution of the eigenvalue problem. This problem consists in finding a subset of pairs $(\lambda, \mathbf{x})$ of a matrix $A \in \mathbb{C}^{n\times n}$, such that

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

where $\lambda \in \mathbb{C}$ is called eigenvalue, and the nonzero vector $\mathbf{x} \in \mathbb{C}^n$ is its corresponding eigenvector. When the matrix A is large and non-symmetric, solving the eigenvalue problem becomes computationally challenging.

Methods to approximate a subset of eigenpairs of large non-symmetric matrices are usually based on the construction of a standard Hessenberg decomposition associated with the matrix A, i. e.,

$$AU_m = U_m B_m + \mathbf{u}_{m+1}\mathbf{e}_m^T, \tag{3.1}$$

where $U_m \in \mathbb{C}^{n\times m}$, B_m is a Hessenberg matrix of order m, $\mathbf{u}_{m+1} \in \mathbb{C}^n$, and $\mathbf{e}_m$ is the mth canonical vector, with m being typically much smaller than n. Under certain conditions, the eigenvalues of the matrix B_m approximate a subset of eigenvalues of the matrix A.

IDR(s) as a method to compute eigenvalues was first studied by M. H. Gutknecht and J.-P. M. Zemke in [45]. In this chapter, we describe another way to obtain the eigenvalues using an underlying Hessenberg decomposition of the form (3.1) from IDR(s). We combine this Hessenberg decomposition with the implicit restarting technique introduced by D. C. Sorensen [97] for Arnoldi to approximate the eigenpairs of interest. Additionally, we suggest a parameter selection for our proposed method which defines a filter polynomial for the spectrum.

This chapter is structured as follows. In section 2, we present an overview of the Hessenberg decompositions, which are the basis for large scale eigenvalues/eigenvectors approximation. Section 3 explains how to compute a Hessenberg decomposition based on the IDR method. Two techniques to refine the information obtained from the IDR-Hessenberg factorization are discussed in section 4. In Section 5, we present numerical experiments to illustrate the behavior of the method proposed.

3.2 Background on Hessenberg decompositions

In (3.1), the columns of the matrix U_m represent a basis for the Krylov subspace

$$\mathcal{K}_m(A, \mathbf{x}) = \{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, \ldots, A^{m-1}\mathbf{x}\}. \tag{3.2}$$

The upper Hessenberg matrix B_m is the projection and restriction of the matrix A on $\mathcal{K}_m(A, \mathbf{x})$. Projections onto Krylov subspaces are the basis for several methods to solve system of linear equations and eigenpairs approximation (see for example [79, 80]). To compute eigenvalues of large, non-symmetric, and sparse matrices, the most common options among the Krylov methods are the Bi-Lanczos [55] and the Arnoldi method [4]. Each of them creates a different Hessenberg decomposition associated with the matrix A. The Bi-Lanczos method uses a short-recurrence formulas to create two Hessenberg tridiagonal decompositions of the form

$$AV_m = V_mT_m + \mathbf{f}\mathbf{e}_m^T$$

and

$$A^HW_m = W_mT_m^H + \mathbf{s}\mathbf{e}_m^T,$$

where $\mathbf{e}_m$ is the mth canonical vector, $\mathbf{f}$ and $\mathbf{s} \in \mathbb{C}^n$, $T_m \in \mathbb{C}^{m\times m}$ is a tridiagonal matrix, the matrix $V_m \in \mathbb{C}^{n\times m}$ is a basis for $\mathcal{K}_m(A, \mathbf{v}_1)$, $W_m \in \mathbb{C}^{n\times m}$ is a basis for $\mathcal{K}_m(A^H, \mathbf{w}_1)$ and the matrices V_m and W_m are bi-orthogonal ($W_m^HV_m = I_m$). However, despite being an efficient short-recurrence method, Bi-Lanczos is numerically unstable (see [69] for a discussion).

Arnoldi method, on the other hand, builds a Hessenberg decomposition

$$AV_m = V_mH_m + \mathbf{f}\mathbf{e}_m^T,$$

where $\mathbf{f} \in \mathbb{C}^n$ and V_m is a matrix with orthogonal columns and represents a basis for $\mathcal{K}_m(A, \mathbf{v}_1)$. This method is widely used to approximate a subset of the eigenpairs of A; nevertheless, its computational and memory cost increases per iteration. An option to overcome this issue is to restart the process (see [74]). Other Hessenberg decompositions to approximate eigenpairs based on Newton and Chebyshev polynomials can be found in [50, 51, 9, 71].

In the original implementations of IDR(s), the authors do not create explicitly any Hessenberg decomposition [95, 103]. M. H. Gutknecht and J.-P. M. Zemke in [45] deduce a generalized Hessenberg decomposition from the IDR(s) method of the form

$$AW_mU_m = W_m\hat{H}_m + \mathbf{w}\mathbf{e}_m^T,$$

from which only the eigenvalues values of A are approximated by the solution of the generalized eigenvalue

$$\hat{H}_m\mathbf{y}_i = \lambda_iU_m\mathbf{y}_i.$$

Here the matrices U_m and $\hat{H}_m$ are in $\mathbb{C}^{m\times m}$, with U_m upper triangular and, H_m is a Hessenberg matrix. The matrix W_m is not explicitly built.

In the next section, we present an IDR(s)-based Hessenberg decomposition which generates, in exact arithmetic, the same eigenvalues as the generalized

Hessenberg decomposition presented in [45]. It is worth remarking two advantages of the IDR(s) Hessenberg decomposition proposed here. First, we explicitly build the matrix W_m, and we can approximate the eigenvectors. Second, the IDR(s) decomposition is of the form (3.1) and this is particularly suitable to apply the implicitly restarted technique of D. C. Sorensen [97].

3.3 A Hessenberg decomposition based on the IDR(s) method

This section proposes a method to build a standard Hessenberg decompositions using the IDR(s) method. First, we review the generalized Hessenberg decomposition presented in [45], then, we present an equivalent standard Hessenberg decomposition. A vector $\mathbf{w}_{i+1}$ in $\mathcal{G}_j$, according to [95], can be written as

$$\mathbf{w}_{i+1} = (A - \mu_j I)\left(\mathbf{w}_i - \sum_{\ell=1}^{s} c_\ell \mathbf{w}_{i-\ell}\right), \tag{3.3}$$

where the $s+1$ vectors $\mathbf{w}_{i-s}, \mathbf{w}_{i-s-1}, \ldots, \mathbf{w}_i$ belong to $\mathcal{G}_{j-1}$, $\mu_j \in \mathbb{C}$ with $\frac{i}{s+1} = j$, and the constants c_ℓ are obtained from the solution of the $s \times s$ system of linear equations

$$(P^H[\mathbf{w}_{i-s}, \mathbf{w}_{i-s+1}, \ldots, \mathbf{w}_{i-1}])\mathbf{c} = P^H\mathbf{w}_i.$$

Using (3.3), we have

$$A\mathbf{w}_i = \mathbf{w}_{i+1} + \mu_{j+1}\mathbf{w}_i - \mu_{j+1}\sum_{\ell=1}^{s} c_\ell \mathbf{w}_{i-\ell} + \sum_{\ell=1}^{s} c_\ell A\mathbf{w}_{i-\ell}, \tag{3.4}$$

or equivalently

$$A\mathbf{w}_i - \sum_{\ell=1}^{s} c_\ell A\mathbf{w}_{i-\ell} = \mathbf{w}_{i+1} + \mu_{j+1}\mathbf{w}_i - \mu_{j+1}\sum_{\ell=1}^{s} c_\ell \mathbf{w}_{i-\ell}.$$

From the latter equation, the authors in [45] propose a generalized Hessenberg decomposition

$$AW_mU_m = W_mH_m + \mathbf{w}\mathbf{e}_m^T,$$

where U_m is an upper triangular matrix and $\hat{H}_m$ is an upper Hessenberg matrix;

their columns are defined as

$$\mathbf{u}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\begin{bmatrix} c_1 \\ \vdots \\ c_s \end{bmatrix} \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{and} \quad \hat{\mathbf{h}}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\mu_{j+1}\begin{bmatrix} c_1 \\ \vdots \\ c_s \end{bmatrix} \\ \mu_{j+1} \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The matrix pencil $(\hat{H}_m, U_m)$ is called the Sonneveld pencil. The eigenvalues of this pencil are divided into two sets: $\{\mu_k\}_{k=1}^{t}$ with $t = \frac{m-1}{s+1}$, and the approximations to the eigenvalues of A or Ritz values $\{\theta_k\}_{k=t}^{m}$. We create an IDR(s)-based standard Hessenberg decomposition of the form (3.1). Setting $W_k = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k]$, and assuming that $A\mathbf{w}_{i-\ell}$ can be written as a linear combination of the vectors $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_{i-\ell}$, $\mathbf{w}_{i-\ell+1}$, for $i = 1, 2, \ldots, i-1$, we obtain

$$A\mathbf{w}_{i-\ell} = W_{i-\ell+1}\mathbf{h}_{i-\ell}. \tag{3.5}$$

Combining (3.4) and (3.5), we obtain

$$A\mathbf{w}_i = W_{i+1}\mathbf{h}_i$$

where

$$\mathbf{h}_i = \left(\begin{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \\ -\mu_{j+1}\begin{bmatrix} c_1 \\ \vdots \\ c_s \end{bmatrix} \\ \mu_{j+1} \\ 1 \end{bmatrix} + \sum_{\ell=1}^{s} c_\ell \mathbf{h}_{i-\ell} \right) \qquad \text{for } i = s+1, \ldots . m. \tag{3.6}$$

Applying (3.6) for $i = 1, 2, \ldots, m$, we obtain a standard Hessenberg decomposition that we call the **IDR factorization**

$$\begin{aligned} AW_m &= W_{m+1}\bar{H}_m && (3.7) \\ &= W_m H_m + \mathbf{w}_{m+1}\mathbf{e}_m^T. && (3.8) \end{aligned}$$

The matrix W_m is a non-orthogonal basis for the Krylov subspace and the Hessenberg matrix H_m has exactly the same eigenvalues as the matrix Sonneveld matrix pencil (see Figure 3.1 for a comparison of Ritz values obtained from the IDR factorization, the Sonneveld pencil, and the Arnoldi method). This result is summarized in the following theorem.

Theorem 2. *Matrix H_m, whose columns are defined in* (3.6)*, can be written as*

$$H_m = \hat{H}_m U_m^{-1}$$

where the matrices $(\hat{H}_m, U_m)$ define the Sonneveld pencil proposed in [45].

Proof. (Induction over the columns of H_m). For $1 \leq i \leq s+1$, let us assume a starting standard Hessenberg decomposition for the Sonneveld pencil. As an inductive step let us assume that $H_{m\times i} = \hat{H}_{m\times i} U_i^{-1}$, or if we represent the columns of the inverse U_m as $[\mathbf{u}^{-1}]_i$, we can write the previous expression as $\mathbf{h}_k = \hat{H}_{m\times k}[\mathbf{u}^{-\mathbf{1}}]_k$ for $1 \leq k \leq i$. For $k = i+1$, taking into account the structure of the matrix U_m, we obtain that the $(i+1)$th column is

$$\hat{H}_{m\times i+1}[\mathbf{u}^{-\mathbf{1}}]_{i+1} = \hat{H}_{m\times i+1}\left(\sum_{\ell=1}^{s} c_\ell [\mathbf{u}^{-\mathbf{1}}]_{i-\ell} + \mathbf{e}_{i+1}\right)$$

Using the induction hypothesis, we obtain

$$\hat{H}_{m\times i+1}[\mathbf{u}^{-\mathbf{1}}]_{i+1} = \sum_{\ell=1}^{s} c_\ell \mathbf{h}_{i-\ell} + \hat{\mathbf{h}}_{i+1},$$

and this is exactly equal to the proposed formula of the $(i+1)$th column of H_m in (3.6). □

If we assume that k vectors have been created in $\mathcal{G}_j$, then any linear combination of these is also a vector in $\mathcal{G}_j$. Therefore, we can rewrite this equation as

$$\mathbf{w}_{i+1} = (A - \mu_j I)\left(\mathbf{w}_i - \sum_{\ell=1}^{s} c_\ell \mathbf{w}_{i-\ell}\right) + \sum_{\ell=0}^{k-1} \beta_{i-\ell}\mathbf{w}_{i-\ell} \tag{3.9}$$

The selection of the parameters β's yields different versions of IDR(s). For example, choosing the parameter β_ℓ to impose biorthogonality between the set of vectors $\{\mathbf{w}_{i-\ell}\}_{\ell=0}^{k-1}$ and $\{\mathbf{P}_\ell\}_{\ell=1}^{k}$ [103], or to make the vector $\mathbf{w}_{i+1}$ orthogonal to the previous vectors in the subspace [102].

Algorithm 8 outlines the process to create an IDR factorization of size m in which β_ℓs are selected to orthogonalize the vector in $\mathcal{G}_j$.

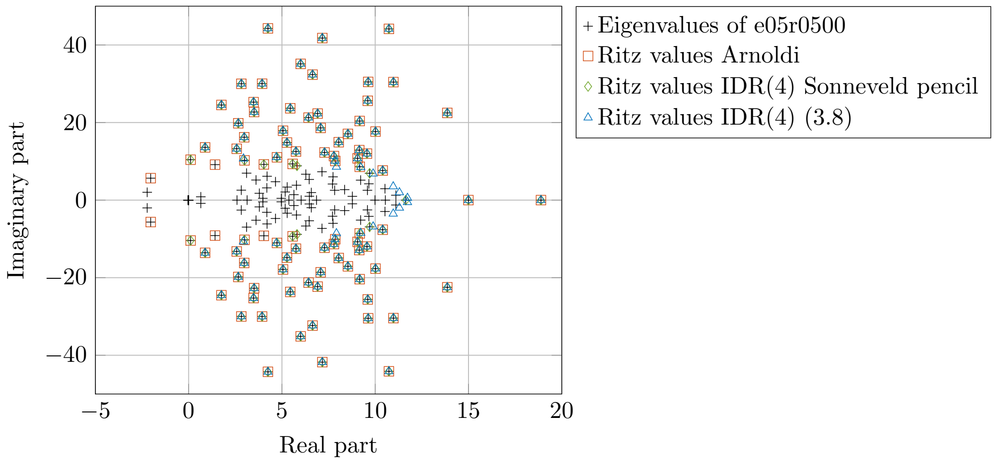

Figure 3.1: Eigenvalues of matrix e05r0500, and Ritz values generated by Arnoldi, IDR($s = 4$) Sonneveld pencil, and our proposed IDR($s = 4$).

Algorithm 8 IDR(s) Process applied to a matrix A

```
1:  Given $s \in \mathbb{N}^+$, $P \in \mathbb{C}^{n\times s}$, $W \in \mathbb{C}^{n\times s+1}$ and $H \in \mathbb{C}^{s+1\times s}$, such that $AW_s = W_{s+1}\bar{H}_s$
2:  for $i = s+1, \ldots, m$ do
3:      if $i$ is multiple of $s+1$ then
4:          Choose the parameter $\mu_j$ for the subspace $\mathcal{G}_j$
5:          $k = 0$
6:      end if
7:      Solve the $s \times s$ system of linear equations
            $(P^H[\mathbf{w}_{i-s}, \mathbf{w}_{i-s+1}, \ldots, \mathbf{w}_{i-1}])\mathbf{c} = P^H\mathbf{w}_i$
8:      $\mathbf{v} = \mathbf{w}_i - \sum_{\ell=1}^{s} c_\ell \mathbf{w}_{i-\ell}$          ▷ $\mathbf{v} \in \mathcal{G}_{j-1} \cap P^\perp$
9:      $\mathbf{w}_{i+1} = (A - \mu_j I)\mathbf{v}$          ▷ New vector in $\mathcal{G}_j$
10:     Create the $i$+1th column of $H$ according to (3.6).
11:     $\beta_{i-\ell} = \langle \mathbf{w}_{i+1}, \mathbf{w}_{i-\ell}\rangle$ and $h_{i-\ell,i} = h_{i-\ell,i} + \beta_{i-\ell}$ for $\ell = 0, 2, \ldots, k-1$
12:     $\mathbf{w}_{i+1} = \mathbf{w}_{i+1} - \sum_{\ell=0}^{k-1} \beta_{i-\ell}\mathbf{w}_{i-\ell}$
13:     $\beta_i = \|\mathbf{w}_{i+1}\|$ and $h_{i+1,i} = \beta_i$
14:     $\mathbf{w}_{i+1} = \mathbf{w}_{i+1}/\beta_i$
15:     $W_{i+1} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_i, \mathbf{w}_{i+1}]$          ▷ Update the IDR factorization
16:     $k = k + 1$
17: end for
```

The matrix H_m, created by Algorithm 8, is called the *Sonneveld matrix* in [45]. At this stage, it is worth commenting on the main differences between the work presented in this chapter and the work in [45]. First, we create a standard Hessenberg decomposition rather than the Generalized Hessenberg decomposition proposed in [45]. The standard Hessenberg decomposition is suitable to apply the implicit restarting technique. Another difference is that by means of our proposed IDR factorization, the eigenvectors of the matrix A

can be approximated without extra calculations. The authors in [45] remove the μ_j values from the spectrum of the Sonneveld pencil $(\hat{H}_m, U_m)$; this process is called *purification* and it creates a smaller pencil that only contains the approximations of the eigenvalues of A. The disadvantage of the purification process is that the Krylov basis has to be recalculated to approximate eigenvectors. The numerical stability difference between our proposed method and the Sonneveld pencil has been recently addressed in [107] by J.-P. M. Zemke.

3.3.1 Operation count and memory consumption

The Arnoldi method, in the mth iteration, requires one matrix-vector multiplication and $\frac{m(m+1)}{2} + 1$ inner products. For IDR(s), the number of inner products does not depend on the iteration number m. In Algorithm 8, every $s+1$ iterations, performs $\frac{3s^2+5s}{2} + 1$, inner products, $s+1$ matrix-vectors products, and it also requires the solution of $s+1$ systems of linear equations of order s. All of this indicates that the computational work of IDR(s) does not grow in every iteration, in contrast to the Arnoldi method. In terms of computational work, IDR(s) is an intermediate option between Bi-Lanczos and Arnoldi method.

IDR(s) and Arnoldi have similar memory requirements. In the mth iteration, IDR has to store the Hessenberg matrix $\bar{H}_m$ of size $(m+1) \times m$, the matrix W_{m+1} of size $n \times (m+1)$, and additionally the matrix P of size $n \times s$. In some application, however, where only the eigenvalues are required, the IDR(s) could be adapted to low memory requirements. IDR(s) would not need to save all the columns of the matrix W_m; it would only required the last $s+1$ vectors of W_{m+1}, the matrix $\bar{H}_m$, and the matrix P.

3.3.2 Approximation of the eigenpairs and stopping criterion

To obtain an approximation of the eigenpairs of the matrix A, we first compute an eigenpair of the small matrix H_m, i. e.

$$H_m \mathbf{y}^{(i)} = \theta_i \mathbf{y}^{(i)} \qquad \text{with } \|\mathbf{y}^{(i)}\| = 1.$$

Then, setting our eigenpair approximation as $(\theta_i, \mathbf{x}^{(i)} = W_m \mathbf{y}^{(i)})$, and using the relation (3.7), we have that

$$A\mathbf{x}^i - \theta_i \mathbf{x}^i = AW_m \mathbf{y}^i - \theta_i W_m \mathbf{y}^i = w_{m+1} \mathbf{e}_m^T \mathbf{y}^i.$$

From the previous equation and setting $[\mathbf{y}^{(i)}]_m$ as the mth component of the vector $y^{(i)}$, we can obtain the following bound

$$\|A\mathbf{x}^{(i)} - \theta_i \mathbf{x}^{(i)}\| \leq \|\mathbf{w}_{m+1}\| |[\mathbf{y}^{(i)}]_m|, \tag{3.10}$$

or, if we normalize the vector $\mathbf{w}_m$

$$\|A\mathbf{x}^{(i)} - \theta_i \mathbf{x}^{(i)}\| \leq h_{m+1,m}|[\mathbf{y}^{(i)}]_m|. \tag{3.11}$$

However, it is not suitable to use $(h_{m+1,m}[\mathbf{y}^{(i)}]_m) \leq \epsilon$ as stopping criterion, because W_m is not a matrix with orthogonal columns, and consequently, the norm of this matrix produces scaling effects over (3.10). For this reason, we consider

$$\frac{\|A\mathbf{x}^{(i)} - \theta_i \mathbf{x}^{(i)}\|}{\|W_m\|} \leq \epsilon,$$

Then we need that

$$h_{m+1,m}|[\mathbf{y}^{(i)}]_m|\|W_m\| \leq \epsilon.$$

In this way, we avoid the scaling effect of the matrix W_m to the residual bound. Furthermore, it is not necessary to compute the norm of W_m in every iteration. An observation made in [102] is that if the matrix W_m has m blocks of size $s+1$ orthogonal vectors then

$$\|W_m\| \leq \sqrt{m}.$$

We stop the process when

$$h_{m+1,m}|[\mathbf{y}^{(i)}]_m|\sqrt{m} \leq \epsilon. \tag{3.12}$$

3.3.3 Relation between the Arnoldi and other Hessenberg decompositions

In this section, we review the relation between different Hessenberg decompositions. In particular, we are interested in the difference between a Hessenberg decomposition obtained by Arnoldi and an IDR(s)-Hessenberg decomposition. Let us assume that after m steps of the Arnoldi method applied to the matrix $A \in \mathbb{C}^{n\times n}$, with an initial vector $\mathbf{x} \in \mathbb{C}^n$ and without breakdown, we obtain the following Hessenberg decomposition

$$AV_m = V_m H_m + \mathbf{f}\mathbf{e}_m^T = V_{m+1}\bar{H}_m. \tag{3.13}$$

On the other hand, let us consider another Hessenberg decomposition associated with the matrix A and the same initial vector $\mathbf{x}$

$$AX_m = X_m G_m + \mathbf{g}\mathbf{e}_m^T = X_{m+1}\bar{G}_m, \tag{3.14}$$

where the columns of the matrix X_{m+1} do not form an orthogonal set. One can relate (3.13) and (3.14) using the reduced QR factorization of the matrix X_{m+1}

$$X_{m+1} = Q_{m+1}R_{m+1}, \tag{3.15}$$

and obtain

$$AQ_m = Q_{m+1}R_{m+1}\bar{H}_m R_m^{-1}. \tag{3.16}$$

Comparing (3.13) and (3.16), we obtain by uniqueness of the Arnoldi Hessenberg decomposition (see Theorem 2.4 in [97]), that $Q_{m+1} = V_{m+1}$, and

$$\bar{H}_m = R_{m+1}\bar{G}_m R_m^{-1}. \tag{3.17}$$

The latter equation can be rewritten as

$$H_m = R_m G_m R_m^{-1} + \frac{g_{m+1,m}}{r_{m,m}}\mathbf{r}\mathbf{e}_m^T, \tag{3.18}$$

where $\mathbf{r} = [r_{i,m+1}]_{i=1}^{m}$. Other formulas related to (3.18) can be found in [36] and [71]. A direct consequence of this equation is that the Ritz values obtained from an Arnoldi Hessenberg decomposition, and the Ritz values obtained from a factorization that generates a non-orthogonal Krylov basis, are not the same. The condition number of the Krylov basis generated X_m, which is at the same the condition number of R_m, gives an indication how far the eigenvalues of G_m are from the approximated eigenvalues resulting from the Arnoldi process. To exemplify this, we consider the following matrix, in Matlab notation

$$A = \text{sprandn}(100, 100, 0.2). \tag{3.19}$$

In Figure 3.2, we plot the Ritz values obtained by $\bar{H}_{15}$ generated by Arnoldi, and the Ritz values obtained by $\bar{G}_{15} = R_{16}^{-1}\bar{H}_{15}R_{15}$. We select randomly three groups of 10000 matrices R each, with different condition numbers. One can observe that the Ritz values tend to be more clustered around Arnoldi's Ritz values, when the condition numbers of the matrices R decrease.

Now, let us turn to the case of IDR(s), we consider again the matrix (3.19), and we obtain different IDR(s)-Hessenberg decompositions

$$AW_{70} = W_{70}H_{70} + \mathbf{w}\mathbf{e}_{70}^T,$$

for $s = 1, 2, 3 \ldots, 35$. The upper part of Figure 3.3 shows the evolution of the condition number of the matrix W_{70} generated by IDR(s), when the value s increases. One can observe how the condition number of W_{70} decreases while parameter s increases. Despite the high condition numbers of the matrices W_{70}, IDR(s) generates, in this experiment, a good approximation of the largest magnitude Ritz value $\lambda_1^{(A)}$ generated by Arnoldi (see lower Figure 3.3). This analysis suggests that the Ritz values of largest magnitude, generated by IDR(s), are closer to some of the Ritz values generated by Arnoldi when we select large values of s. In [94], P. Sonneveld drew a similar conclusion for IDR(s) in the context of solving systems of linear equations; using stochastic analysis, he related the behavior of the Arnoldi-based method GMRES [81] and IDR(s) when s tends to infinity.

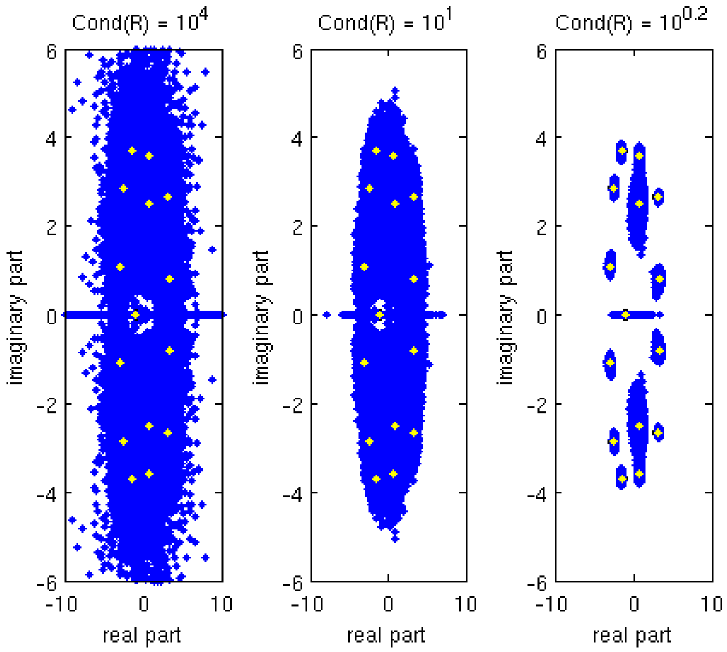

Figure 3.2: For the matrix (3.19), we plot the Ritz values of $\bar{H}_{10}$ generated by Arnoldi (yellow dots) and the Ritz values of matrices G_{10} (blue dots), where $\bar{G}_{10} = R_{11}^{-1}\bar{H}_{10}R_{10}$, Matrices R are generated randomly in three groups. In each group, the matrices R have a fixed condition number.

3.4 Refining the spectral information

In some applications it is important to find eigenvalues and their corresponding eigenvectors in a specific region of the complex plane. For example, the eigenvalues with largest real part for stability analysis, or the nearest eigenvalues to a given point for vibration analysis. For this reason, we implement two techniques to refine the spectral information obtained from the Hessenberg relation described in the previous section.

3.4.1 A polynomial filter based on the selection of the parameters μ_j

The explicit restart is one of the first ideas to restart a Hessenberg decomposition [74]. This is based on initiating the process with an improved initial vector. The new initial vector can be a linear combination of the approximated wanted eigenvectors, or a vector of the form

$$\mathbf{v}_1^+ = p_k(A)\mathbf{v}_1, \tag{3.20}$$

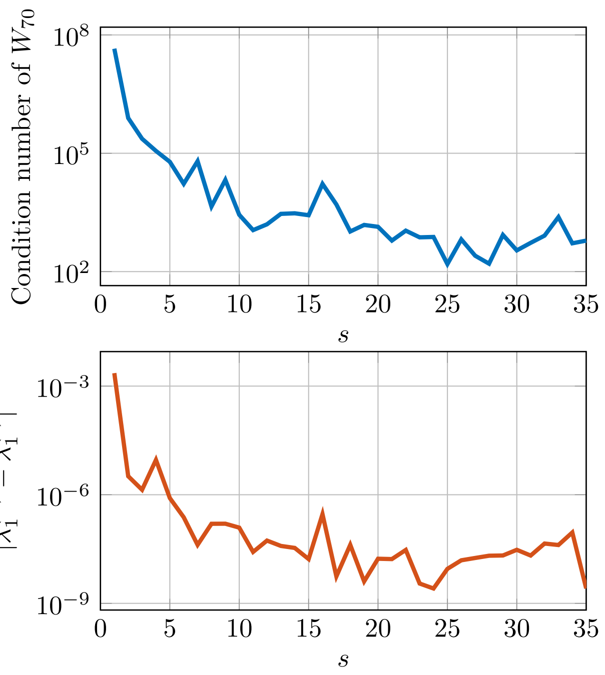

Figure 3.3: Upper: condition number of the matrix W_{70} generated by IDR(s) as function of s. Lower: difference between the largest magnitude Ritz values generated by Arnoldi ($\lambda_1^{(A)}$) and IDR(s) ($\lambda_1^{(I)}$) as function of the value s.

where p_k is a polynomial which amplifies the components of $\mathbf{v}_1$ toward the desired eigenvectors and reducing those in the remaining eigenvectors (see [76]). The polynomial p_k is called a filter polynomial. An example of the form of p_k might be

$$p_k(t) = (t - \omega_1)(t - \omega_2)\dots(t - \omega_k). \tag{3.21}$$

Different options to select the parameter $\{\omega_i\}_{i=1}^k$ have been studied in [76], [97], and [62].

For IDR(s), the authors prove in [95] that every vector in the subspaces $\mathcal{G}_j$ satisfies

$$\mathbf{w}_k = \Psi_{m-j}(A)\Omega_j(A)\mathbf{w}_1, \quad \text{where} \quad \Omega_j(t) = (t-\mu_1)(t-\mu_2)\dots(t-\mu_j), \tag{3.22}$$

and $\Psi_{m-j}(A)$ is a polynomial of degree $m-j$ and its coefficients are fully determined by the IDR(s) procedure. The vectors $\{\mathbf{w}_i\}_{i=1}^m$ also form the Krylov subspace basis in (3.7). There is an analogy between (3.20) and (3.21), and (3.22). We exploit this fact by applying the polynomial Ω_j as a polynomial filter. This idea is similar to the one presented by Saad in [76]. We select the parameters μ_i in (3.22) to minimize the infinity norm of Ω_j in the area where the unwanted eigenvalues are localized. This is achieved by choosing μ_i as the Chebyshev nodes on the interval $[l, u]$, where l and u are the foci of the ellipse that encloses the unwanted portion spectrum of the matrix H_m (see Figure 3.4). The polynomial filter $\Omega_j(A)$ is not fixed. The degree of the polynomial $\Omega_j(A)$ grows when the IDR process creates vectors in a new subspace $\mathcal{G}_j$. We stress that the IDR polynomial filter is implicitly applied to the vector $\mathbf{w}_k$: it does not require any additional computation; it is achieved by a special choice of the parameters μ_j.

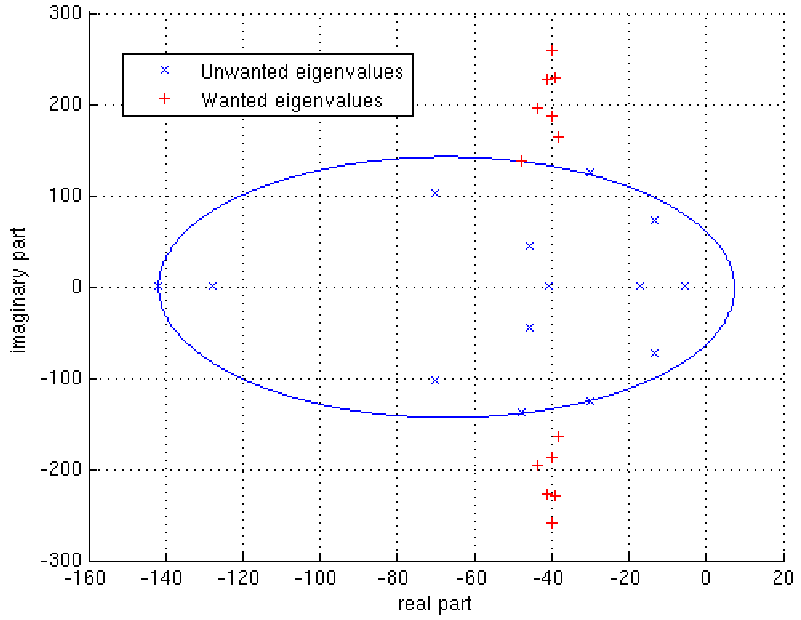

Figure 3.4: Select μ_j to minimize the norm of the IDR polynomial (3.22) in the ellipse which encloses the unwanted eigenvalues.

3.4.2 Implicitly restarting

The most successful variant to approximate subsets of eigenvalues and their corresponding eigenvectors of large and sparse matrices is the Implicitly Restarted Arnoldi Method (IRAM) proposed by D. C. Sorensen in 1992. This method is also the basis for the software package ARPACK [58]. The idea is to truncate the Hessenberg decomposition by removing the uninteresting part of the spectrum using QR steps. After this truncation, the Hessenberg decomposition is expanded to improve the Ritz values and Ritz vectors in the direction of the wanted portion of the spectrum.

The main idea of the implicit restart is to apply orthogonal transformations to the Hessenberg decomposition to reorder the Ritz values. To illustrate this, let us consider a Hessenberg decomposition of size $m+1$

$$AZ_{m+1} = Z_{m+1}H_{m+1} + \mathbf{z}_{m+2}\mathbf{e}_{m+1}^T, \tag{3.23}$$

and suppose that λ is an unwanted Ritz value (it is also called exact shift). Consider the orthogonal matrix Q and the upper triangular matrix R, such that

$$H_{m+1} - \lambda I = QR.$$

If we multiply (3.23) by Q on the right, and define $\hat{H}_{m+1} = Q^T H_{m+1} Q$, and $\hat{Z}_{m+1} = Z_{m+1}Q$, we obtain

$$A\hat{Z}_{m+1} = \hat{Z}_{m+1}\hat{H}_{m+1} + \mathbf{z}_{m+2}\mathbf{e}_{m+1}^T Q.$$

Now, if we discard the vectors $\hat{\mathbf{z}}_{m+2}$, and truncate the matrix $\hat{Z}_{m+1}$ to m columns, we obtain a new Hessenberg decomposition of size m

$$A\hat{Z}_m = \hat{Z}_m\hat{H}_m + \hat{\mathbf{z}}_{m+1}\mathbf{e}_m^T Q.$$

which does not contain the unwanted Ritz value λ.

The implicitly restarted Arnoldi has been analyzed in different works for example: [57], [62], and [56]. Another variant of implicit restarting using the Schur factorization was proposed by Stewart in [98], and a new implementation was recently proposed by Bujanović and Drmač in [21].

In the context of an IDR factorization, we implement the implicit restarting technique taking advantage of the input parameter of Algorithm 8. After the creation of an IDR factorization of size m, we discard the unwanted Ritz values using the implicit restarting technique, and then we truncate it to obtain a new Hessenberg factorization of size s which is the input parameter of the iterative process. The value of s should be greater or equal than the number of wanted eigenpairs. Algorithm 9 outlines the IDR(s) with implicit restarting.

Algorithm 9 Implicit restarting of an IDR factorization

```
1: Given an initial Hessenberg relation of size $s$. The value of $s$ should be greater or equal
   to the number of wanted eigenvalues and $m > s$.
2: Expand the initial factorization using Algorithm 8, to obtain the IDR factorization of
   size $m$:
       $AW_s = W_sH_s + \mathbf{w}_{s+1}\mathbf{e}_s^T \rightarrow AW_m = W_mH_m + \mathbf{w}_{m+1}\mathbf{e}_m^T$
3: Reorder the IDR factorization, using as a shift $\lambda \in \{\mu_1, \dots, \mu_j\} \cup$
   {the unwanted eigenvalues}.
4: Truncate the IDR factorization to obtain the new Hessenberg relation of size $s$ in which
   the unwanted Ritz values where eliminated.
5: Test for convergence. If no convergence **go to 2** with the new Hessenberg relation else
   **return**
```

3.5 Numerical experiments

In this section, we present six numerical experiments to illustrate the computational performance of the IDR(s) for eigenvalues computations. First, we compare the basic IDR (Algorithm 8) with the basic Arnoldi. In the other experiments, we compare the implicitly restarted version of IDR and Arnoldi. All the experiments were executed using Matlab 8.0 (R2012b) on a computer with an I7 Intel processor 2.4GHz, 4GB of RAM memory.

In the case of the parameter selection for IDR(s) with implicit restart, we select s as the number of wanted eigenvalues, and the parameter m is selected as $2 \times s$. $P \in \mathbb{R}^{n \times s}$ is a random matrix with orthogonal columns. The initial guess is selected randomly and we use this vector as initial guess in both IDR and IRAM. In the first iteration the initial Hessenberg factorization for IDR(s) of size s is computed using the Arnoldi method. The selection of the parameter μ_j is discussed in Section 4.1. We adapt the formula (3.12) for IDR(s) with implicit restart for multiple eigenpairs in the following way

$$h_{m+1,m} \max_{1 \le i \le s} |[\mathbf{y}^{(i)}]_m| \sqrt{m} \le \epsilon.$$

where $\epsilon = 10^{-10} \|A\|_F$.

From the second to the sixth experiment, we compare the CPU time of IDR(s) with implicit restart implemented in Matlab and two implementations of IRAM: the first one is a Matlab interpreted code and the second one is the built-in command `eigs`. The command `eigs` is an interface for the ARPACK's FORTRAN-native-code. Therefore, in most of the experiments the command `eigs` produces the shortest CPU time, when this is compared with other native Matlab codes.

Experiment 3.1. We consider a random sparse matrix of dimension 1000. This matrix is generated in Matlab using the following command[†]

$$\text{A} = \text{sprandn(1000,1000,0.1);}$$

We compare the basic versions of IDR($s = 10$) and Arnoldi to find the eigenvalue of A with largest module $\lambda = -10.0581 + 0.27421i$. The parameters μ_j and P for IDR are selected randomly. Figure 3.5 shows the evolution of the absolute error for each algorithm.

We stop the algorithms when the absolute error is smaller than 10^{-10}. The Arnoldi method takes 244 matrix-vector multiplications to obtain the desired reduction in the absolute error, while IDR($s = 10$) uses 324 matrix-vector multiplications. The execution time for Arnoldi is 0.184 seconds and for IDR($s = 10$) is 0.09 seconds.

[†]Using the default values for the random number generator with the command `rng('default')`.

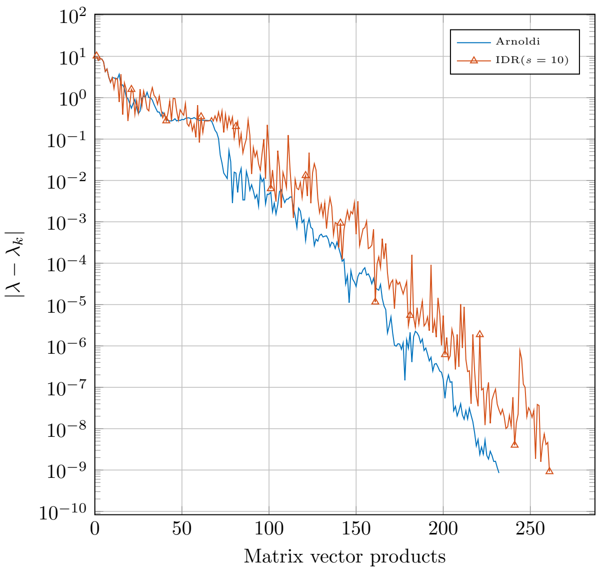

Figure 3.5: Experiment 3.1. Evolution of the absolute error between $\lambda = -10.0581 + 0.27421i$ and its approximations $\hat{\lambda}$ obtained from Arnoldi and IDR($s = 10$).

Method	Restarts	Time (sec.)	Residual bound	Max. diff. from `eigs`
IRAM($k = 15$,$p = 32$)	141	1.19	5.86×10^{-09}	6.70×10^{-14}
IDR($s = 15$,$m = 32$)	91	0.80	3.70×10^{-09}	2.41×10^{-08}
IDR($s = 15$,$m = 48$)	34	0.77	2.49×10^{-09}	1.83×10^{-08}

Time `eigs` command: 0.72 sec.

Table 3.1: Experiment 3.2. Comparison between IRAM and IDR target: the 15 right most eigenvalues

Experiment 3.2. As the second example, we consider the Toeplitz tridiagonal matrix

```
A = gallery('tridiag',n,-1,2,-1)
```

of order 1000. Table 3.1 shows the comparison between IRAM and the implicitly restarted IDR to compute the 15 largest algebraic eigenvalues of this matrix. Figure 3.6 shows the absolute error of the methods computing the largest eigenvalue of this matrix $\lambda = 2 + 2\cos(\frac{\pi}{1001})$ in 85 implicit restarting cycles.

Experiment 3.3. We consider the real non-symmetric matrix CK656 from the collection Non-Hermitian Eigenvalue Problem [8]. In this example we compute the 24 eigenvalues with largest module [8]. Table 3.2 shows the comparison of IRAM and the implicitly restarted IDR.

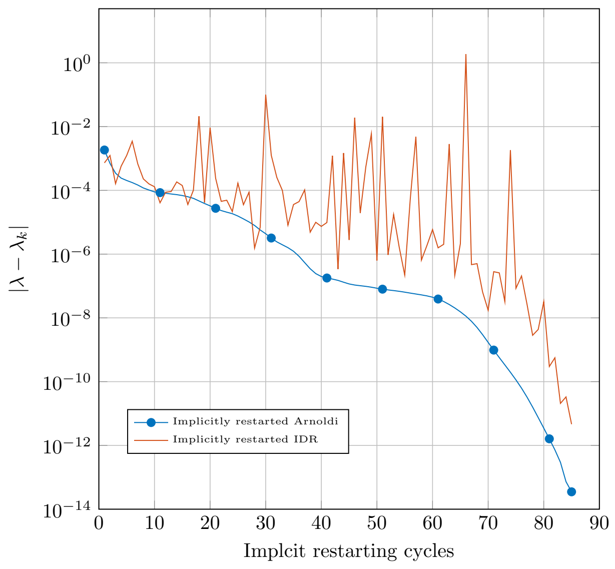

Figure 3.6: Experiment 3.2. Evolution of the absolute error between $\lambda = 2 + 2\cos(\frac{\pi}{1001})$ and its approximations $\hat{\lambda}$ obtained from IRAM and the Implicitly restarted IDR($s = 15$).

Method	Restarts	Time (sec.)	Residual bound	Max. diff. from `eigs`
IRAM($k = 24$,$p = 50$)	19	0.31	4.83×10^{-10}	5.21×10^{-15}
IDR($s = 24$,$m = 50$)	20	0.35	1.69×10^{-13}	1.62×10^{-12}
IDR($s = 24$,$m = 75$)	8	0.34	1.31×10^{-09}	2.29×10^{-09}
IDR($s = 24$,$m = 100$)	6	0.58	1.13×10^{-09}	9.55×10^{-13}

Time `eigs` command: 0.17

Table 3.2: Experiment 3.3. Comparison between IRAM and IDR asked for the 24 leftmost eigenvalues

Experiment 3.4. In the fourth experiment, we compute 12 of the largest magnitude eigenvalues of the matrix AF23560 from the Non-Hermitian Eigenvalue Problem Collection (NEP) [8]. AF23560 is a real non-symmetric matrix of order 23560. Table 3.3 presents the comparison between IRAM and the implicitly restarted IDR with different parameter selection.

Experiment 3.5. We consider the real non-symmetric matrix HOR131 of dimension 434×434. We aim to compute the 8 eigenvalues with largest real part. Table 3.4 shows the results obtained from IRAM and different parameters of the implicitly restarted IDR.

Experiment 3.6. In the sixth experiment, we consider the matrix that arises from the finite difference discretization of the 2D Schrödinger equation. This

Method	Restarts	Time (sec.)	Residual bound	Max. diff. from `eigs`
IRAM($k=12$,$p=26$)	6	0.64	2.78×10^{-08}	4.74×10^{-14}
IDR($s=12$,$m=26$)	7	0.47	7.23×10^{-08}	1.72×10^{-10}
IDR($s=12$,$m=39$)	3	0.44	2.23×10^{-07}	9.74×10^{-09}
IDR($s=12$,$m=53$)	2	0.54	7.66×10^{-07}	2.48×10^{-09}

Time `eigs` command: 0.26

Table 3.3: Experiment 3.4. Comparison between IRAM and IDR asked for the 12 eigenvalues of largest magnitude

Method	Restarts	Time (sec.)	Residual bound	Max. diff. from `eigs`
IRAM($k=8$,$p=18$)	6	0.03	2.19×10^{-13}	4.54×10^{-15}
IDR($s=8$,$m=18$)	13	0.06	4.46×10^{-11}	8.05×10^{-10}
IDR($s=8$,$m=27$)	7	0.19	1.38×10^{-10}	2.65×10^{-10}
IDR($s=8$,$m=36$)	3	0.04	8.23×10^{-13}	1.70×10^{-09}

Time `eigs` command: 0.02

Table 3.4: Experiment 3.5. Comparison between IRAM and IDR asked for the 8 eigenvalues of largest real

equation models the energy levels of the confined hydrogen atom, and is given by

$$-\triangle u(x,\,y)-\frac{2u(x,\,y)}{\|(x,\,y)\|}=\lambda u(x,\,y)\qquad (x,\,y)\in(-16,\,16)\times(-16,\,16)\,,$$

with homogeneous Dirichlet boundary conditions. We use a nonuniform mesh refined near the origin and obtain a matrix of size 44100×44100. We are interested to approximate the 16 leftmost eigenvalues. We apply IRAM and the Implicitly Restarted IDR to the matrix $(A-\sigma I)^{-1}$, where $\sigma=-2.1$. Table 3.5 shows the comparison between these two methods.

Method	Restarts	Time (sec.)	Residual bound	Max. diff. from `eigs`
IRAM($k=16$,$p=34$)	10	11.19	2.69×10^{-11}	2.06×10^{-11}
IDR($s=16$,$m=34$)	12	11.88	9.73×10^{-11}	6.01×10^{-08}
IDR($s=16$,$m=50$)	6	12.6	2.25×10^{-11}	1.72×10^{-09}

Time `eigs` command: 2.92

Table 3.5: Experiment 3.6. Comparison between IRAM and IDR asked for the 16 leftmost eigenvalues

3.6 IDR(s) to solve the Quadratic Eigenvalue Problem

In this section, we are interested in applying the Induced Dimension Reduction method to solve the Quadratic Eigenvalue Problem (QEP). The QEP is defined as finding a subset of pairs $(\lambda, \mathbf{x})$, where $\lambda \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^N$, such that

$$(\lambda^2 M + \lambda D + K)\mathbf{x} = \mathbf{0}, \tag{3.24}$$

where M, D, and K are (sparse) matrices of order n often referred as mass, damping, and stiffness matrices, respectively. The QEP appears in different areas as vibration analysis, dynamical systems, or stability of flows in fluid mechanics (see [99] and their references therein). One of the most common options to solve the quadratic eigenvalue problem (3.24) is to linearized it to an standard eigenvalue problem. First, problem (3.24) can be written as a generalized eigenvalue problem, i.e.,

$$C\mathbf{y} = \lambda G\mathbf{y}, \tag{3.25}$$

where

$$C = \begin{bmatrix} -D & -K \\ I & 0 \end{bmatrix}, \quad \text{and} \quad G = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}.$$

Second, if the matrix M is not singular, (3.25) can be rewritten as standard eigenvalue problem

$$A\mathbf{y} = \lambda \mathbf{y}, \tag{3.26}$$

with

$$A = \begin{bmatrix} -M^{-1}D & -M^{-1}K \\ I & 0 \end{bmatrix}. \tag{3.27}$$

It is easy to check that the eigenvectors of A and the eigenvectors of (3.24) are related by

$$\mathbf{y}_i = \begin{bmatrix} \lambda_i \mathbf{x}_i \\ \mathbf{x}_i \end{bmatrix}.$$

Then, one can apply any eigensolver software for the standard eigenvalue (3.26) and obtain approximate solutions of the quadratic eigenvalue problem (3.24). This approach has two main disadvantages. First, it solves a standard eigenvalue problem of double the dimension of the original quadratic eigenvalue problem. Second, some properties of the matrices M, D, and K are lost during the linearization; for example, matrices M, D and K can be symmetric positive definite (SPD) matrices but the matrix A does not keep the SPD property.

To overcome the disadvantages of using the linearization (3.26), the authors in [10] propose a method called Second Order Arnoldi (SOAR), which is a modification of the Arnoldi method [4]. By exploiting the block structure of the matrix A, the SOAR method uses approximately half of the memory of the classical Arnoldi method applied to problem (3.26). Also and more importantly, this method preserves essential structures and properties of the matrices involved.

The Arnoldi method has as main drawback its demanding computational requirements. For this reason, we study the Induced Dimension Reduction Method for eigenvalue problem, presented in the previous sections, to solve the Quadratic Eigenvalue Problem as an alternative to the Arnoldi method.

3.6.1 Second order IDR(s)

One can exploit the block structure of the $2n \times 2n$ matrix A in (3.27) for the creation a standard Hessenberg relation. Let us consider (3.7), with the matrix W_m rewritten in two block matrices of size $n \times m$ as

$$W_m = \begin{bmatrix} W_m^{(U)} \\ W_m^{(L)} \end{bmatrix},$$

Then (3.7) can be written as

$$-M^{-1}DW_m^{(U)} - M^{-1}KW_m^{(L)} = W_m^{(U)}H_m + \mathbf{w}_{m+1}^{(U)}\mathbf{e}_m^T \tag{3.28}$$

$$W_m^{(U)} = W_m^{(L)}H_m + \mathbf{w}_{m+1}^{(L)}\mathbf{e}_m^T. \tag{3.29}$$

From (3.29), and assuming that the first column vector ($\mathbf{w}_1$) of the matrix W_m has the following pattern

$$\mathbf{w}_1 = \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix}, \qquad \text{with } \mathbf{u} \neq \mathbf{0} \in \mathbb{R}^n,$$

we have (using the Matlab subindex notation)

$$W_m^{(U)} = W_{m+1}^{(L)}\bar{H}_m = W_{m+1}^{(L)}(:, 2:m+1)\bar{H}_m(2:m+1, 1:m). \tag{3.30}$$

The equation (3.28) can be rewritten as

$$-M^{-1}DW_m^{(U)} - M^{-1}KW_m^{(U)}T_m = W_m^{(U)}H_m + \mathbf{w}_{m+1}^{(U)}\mathbf{e}_m^T, \tag{3.31}$$

where

$$T_m = \begin{bmatrix} \mathbf{0} & \bar{H}_m(2:m, 1:m-1)^{-1} \\ 0 & \mathbf{0} \end{bmatrix}. \tag{3.32}$$

Equations (3.31) and (3.32) suggest a formula to compute the column vectors of the matrix $W_m^{(L)}$ as a linear combination of the column vector of $W_m^{(U)}$. Algorithm 10 shows a possible implementation of these ideas. This method needs only half of the memory

Algorithm 10 SOIDR(s) for solving the QEP.

```
1: Given $s \in \mathbb{N}^+$, $P \in \mathbb{C}^{n\times s}$, $M$, $D$, and $K$.
2: Run SOAR to obtain $W \in \mathbb{C}^{n\times s+1}$ and $H \in \mathbb{C}^{s+1\times s}$, s.t.
   $$A \begin{bmatrix} W^{(U)} \\ W^{(L)} \end{bmatrix}_s = \begin{bmatrix} W^{(U)} \\ W^{(L)} \end{bmatrix}_{s+1} \bar{H}_s.$$
3: $T_s = \bar{H}_s(2:s, 1:s-1)^{-1}$
4: for $i = s+1, \dots, m$ do
5:     if $i$ is multiple of $s+1$ then
6:         Choose the parameter $\mu_j$ for the subspace $\mathcal{G}_j$.
7:     end if
8:     Solve $(P^H[\mathbf{w}^{(U)}_{i-s}, \mathbf{w}^{(U)}_{i-s+1}, \dots, \mathbf{w}^{(U)}_{i-1}])\mathbf{c} = P^H\mathbf{w}^{(U)}_i$.
9:     $\mathbf{v} = \mathbf{w}^{(U)}_i - \sum_{\ell=1}^{s} \beta_\ell \mathbf{w}^{(U)}_{i-\ell}$.
10:    Compute the latest column of $W^{(L)}_i$ as $\mathbf{v}^{(L)}$ using (3.30) and (3.9).
11:    $\mathbf{w}^{(U)}_{i+1} = -M^{-1}(D\mathbf{v} + K\mathbf{v}^{(L)}) - \mu_j\mathbf{v}$.
12:    Create the $i$th column of $H$ according to (3.6).
13:    Update $T_i$ using (3.32).
14:    $W^{(U)}_{i+1} = [\mathbf{w}^{(U)}_1, \mathbf{w}^{(U)}_2, \dots, \mathbf{w}^{(U)}_i, \mathbf{w}^{(U)}_{i+1}]$.
15: end for
16: Compute the eigenpairs $\{(\lambda_i, \mathbf{z}_i)\}_{i=1}^m$ s.t. $H_m\mathbf{z}_i = \lambda_i\mathbf{z}_i$.
17: return $\{(\lambda_i, W^{(U)}_m\mathbf{z}_i)\}_{i=1}^m$.
```

Method	Time [s]
SOAR	3.67
SOIDR(4)	3.78
IDR(4)	2.41

Table 3.6: *Experiment 3.8.* Execution time comparison for SOAR, SOIDR(4), and IDR(4) after 40 matrix vector products.

3.6.2 Numerical experiments

Experiment 3.7. The purpose of this example is to compare the convergence of Arnoldi, SOAR, and SOIDR for the exterior eigenvalues of the QEP. The matrices M, D, and K are random sparse of order 400. Figure 3.7 shows a comparison between the errors of the Ritz values generated by Arnoldi, SOAR, and SOIDR(4) or 35 Ritz values after 40 matrix vector products.

Experiment 3.8. In our second experiment we measure the execution times for SOAR, SOIDR(4), and IDR(4) for eigenvalues [5]. We only compute a set of 10 the eigenvalues of (3.24) and the matrices M, D and K are random matrices of size 6000×6000. Table 3.6 shows the CPU times for each method.

Experiment 3.9. This example was presented in [91], and models the propagation of sound waves in a room with five solid walls and one wall of a sound-

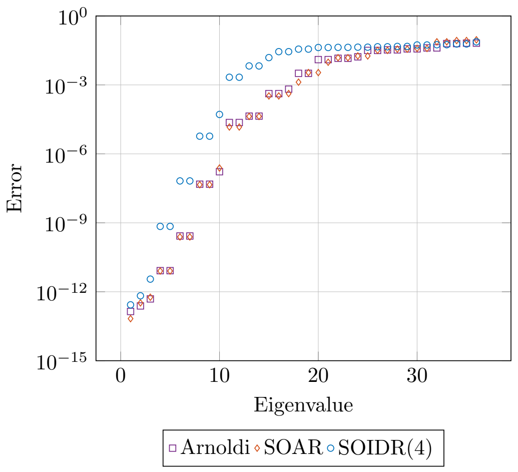

Figure 3.7: *Experiment 3.7.* Convergence for 35 Ritz values after 40 matrix vector products. One can see a similar convergence behavior, however some Ritz values of the SOIDR(4) have a larger error.

absorbing material

$$\frac{\lambda}{c^2}p - \triangle p = 0 \quad \text{in } [-2.0,\, 2.0]^3 \tag{3.33}$$

where c is the speed of sound (340 meter/second) and the boundary conditions are

$$\frac{\partial p}{\partial n} = 0 \qquad \text{for the solid walls,} \tag{3.34}$$

and

$$\frac{\partial p}{\partial n} = -\frac{\lambda}{cZ_n}p \qquad \text{for the absorbing wall.} \tag{3.35}$$

Selecting an impedance $Z_n = 0.2 - 1.5i$, this problem has an analytical eigenvalue $-5.19 + 217.5i$. We discretized (3.33)–(3.35) using finite element and obtain matrices of order 1681. Figure 3.9 shows the evolution of the error of the Ritz values generated by SOAR and SOIDR(2).

3.7 Discussion and remarks

This chapter has introduced an algorithm to compute eigenpairs of large matrices using a Hessenberg decomposition based on the IDR(s) method. The main advantage of the proposed Hessenberg decomposition is its low computational cost since it only uses recurrences of size $s + 1$. We have implemented two techniques to refine the spectral information obtained. The first technique is based on the parameter selection for our proposed algorithm and the second technique is Sorensen's implicitly restart.

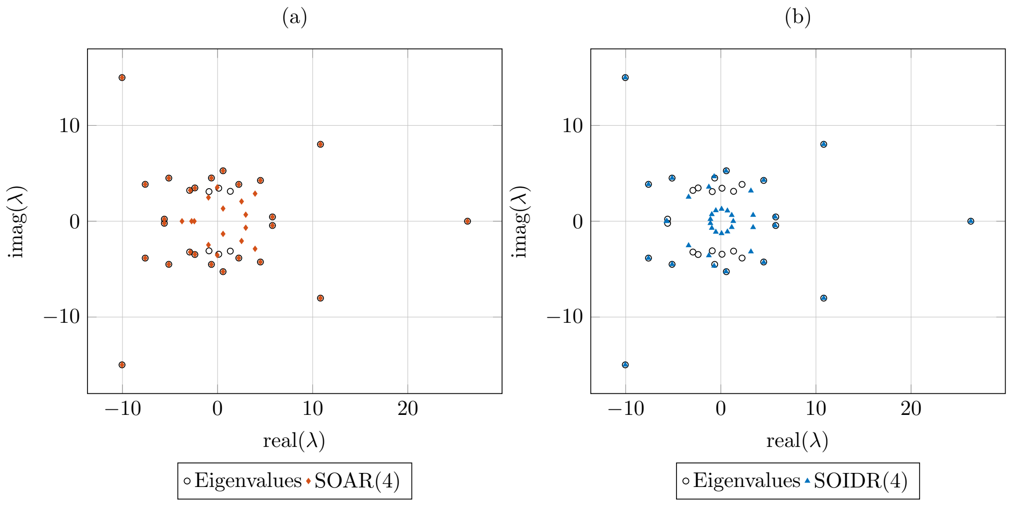

Figure 3.8: *Experiment 3.7.* (a) Exterior eigenvalues and their approximation by SOAR. (b) Exterior eigenvalues and their approximation by SOIDR(4).

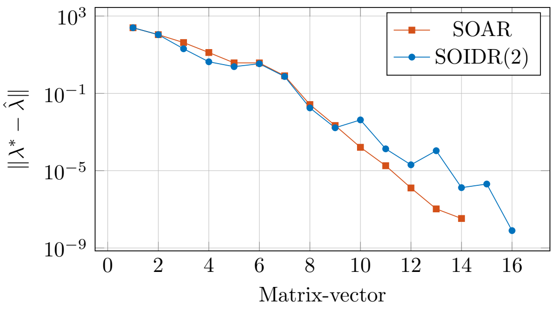

Figure 3.9: *Experiment 3.9.* Error convergence for SOAR and SOIDR(2) to the known eigenvalues $\lambda^* = -5.19 + 217.5i$.

The Krylov subspace basis created by our IDR-Hessenberg decomposition is only locally orthogonal, which might have a negative effect on the convergence speed or numerical stability. However, IDR(s) for eigenvalues shows competitive performance with respect to IRAM in the numerical examples presented in this chapter. This interesting fact, in conjunction with the computational efficiency to compute the IDR-Hessenberg factorization, can also be exploited in applications when only the eigenvalues are required.

We have also developed a second order IDR algorithm (SOIDR) based on the ideas underlying SOAR. In contrast to IDR for the standard eigenvalue problem, SOIDR is not short recurrence. The memory requirements of SOIDR are comparable to that of SOAR. In our numerical test about solving QEP, SOAR exhibits a faster convergence, and is therefore the preferred method.

CHAPTER 4

Accelerating the Induced Dimension Reduction method using spectral information

The Induced Dimension Reduction method (IDR(s)) is a short-recurrences Krylov method to solve systems of linear equations. In this chapter, we accelerate this method using spectral information. We construct a Hessenberg relation from the IDR(s) residual recurrences formulas, from which we approximate the eigenvalues and eigenvectors. Using the Ritz values, we propose a self-contained variant of the Ritz-IDR(s) method [87] for solving a system of linear equations. In addition, the Ritz vectors are used to speed-up IDR(s) for the solution of sequence of systems of linear equations.

4.1 Introduction

In this chapter, we are interested in accelerating the convergence of the Induced Dimension Reduction method (IDR(s)) [95] to solve a system of linear

This chapter is based on the article:

R. Astudillo, J. M. de Gier, and M. B. van Gijzen. Accelerating the Induced Dimension Reduction method using spectral information. Technical Report 17-04, Delft Unversity of Technology, Delft Institute of Applied Mathematics, Delft, ZH, The Netherlands, 2017 .

equations

$$Ax = \mathbf{b}, \qquad \text{with } A \in \mathbb{C}^{n\times n} \text{ and } \mathbf{b} \in \mathbb{C}^n, \tag{4.1}$$

and also to solve sequences of systems of linear equations

$$A\mathbf{x}^{(i)} = \mathbf{b}^{(i)}, \qquad \text{with } A \in \mathbb{C}^{n\times n} \text{ and } \mathbf{b}^{(i)} \in \mathbb{C}^n, \text{ for } i = 1, 2, \ldots, p. \tag{4.2}$$

The vectors $\mathbf{x}$, $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)}$ represent the unknowns in $\mathbb{C}^n$, and we only consider the case when the coefficient matrix A is a non-Hermitian and non-singular matrix.

IDR(s) is a Krylov subspace method which has been proved to be effective for solving large and sparse systems of linear equations. Both theoretical and practical aspects of the IDR(s) have been studied in different works, e.g., [87], [49], [103], [23], and [94] among others. Simoncini and Szyld reformulate IDR(s) as a Petrov-Galerkin method in [87]. The authors prove that in IDR(s) the subspace of constraints or left space is a block rational Krylov subspace. Based on this connection with the rational subspaces, they propose to use the Ritz values to accelerate the convergence of IDR(s). This idea originates Ritz-IDR(s), which is an effective IDR(s) variant to solve systems of linear equations (4.1) where the spectrum is highly complex.

To obtain a subset of the Ritz values, Ritz-IDR(s) requires a preceding call to an external sparse eigensolver routine, for example the Arnoldi method [4] or Bi-Lanczos method [55]. In the first part of this chapter, we present a self-contained version of the Ritz-IDR(s), i. e., a Ritz-IDR(s) variant that does not require an external call to an eigensolver routine. We compute the upper Hessenberg matrix H_m from a Hessenberg relation as

$$AW_m = W_m H_m + \mathbf{f}\mathbf{e}_m^T, \tag{4.3}$$

during the first iterations of IDR(s). Then, we obtain the Ritz values from the matrix H_m, and use them as input parameter of the subsequent iterations of IDR(s).

In the second part of this chapter, we apply IDR(s) to solve sequences of systems of linear equations (4.2). We only consider the case when the coefficient matrix does not change and the right-hand side vectors $\{\mathbf{b}^{(i)}\}_{i=1}^{p}$ are not available simultaneously. This kind of problems arises naturally from the discretization of linear time-dependent differential equations and the solution of systems of non-linear equations using modified Newton-type methods with constant Jacobian matrix.

Subspace recycling is a common technique to accelerate the Krylov method (see for example [61], [25], and [1] among others). This process consists of approximating invariant subspaces or calculating a “good” Krylov subspace basis and use this information to save matrix-vector products at the solution of the system of linear equations. For methods as GMRES [81] and GCR [32] the recycling idea has been incorporated to accelerate the solution of a single linear system of equations in [61] and [25] respectively. In the case of solving sequences of systems of linear equations, these methods have been adapted in [68] and [67]. Also, other Krylov methods have been adapted to solve sequences of systems of linear equations, for example BiCG in [2], GMRES(m) in [67], and IDRstab in [66].

GCROT [68] and GMRES are long-recurrences methods, with an optimal residual minimization property, but also these methods can be expensive in terms of memory and CPU consumption. For this reason, we propose an IDR(s) variant, that is a short-recurrences and memory-limited method to solve (4.2). First, we show how to obtain Ritz values and Ritz vectors from IDR(s) for solving a system of linear equations. Second, we present how to enrich the searching subspace of IDR(s) with the Ritz vectors. Finally, we apply IDR(s) with the Ritz vectors to solve sequences of linear equations as a main application of this enrichment.

This chapter is organized as follows. A review of IDR(s) and its recurrences is presented in the second section. In section 3, we present an IDR(s) variant to solve system of linear equations. We present how to obtain an underlying Hessenberg relation from the IDR(s) residual recurrences. This allows us to find approximation to the eigenvalues of the coefficient matrix involved. This eigenvalues approximations are used to accelerate the IDR(s) method. Section 4.3.1 shows the numerical examples related to the solution of system of linear equations. In section 4, we explain how to add the Ritz vectors to the initial searching space of IDR(s) to save computational effort. As a main application of this idea, we apply IDR(s) to solve a sequence of system of linear equations. Using the Hessenberg relation deduced in section 3, we compute a set of Ritz vectors during the solution of the first system of linear equation. These Ritz vectors are used to accelerate the subsequent systems of linear equations. Numerical experiments for the solution of a sequence of systems of linear equations using IDR(s) are presented in section 4.4.2. In section 5, we present the general conclusions and remarks.

4.2 Review on IDR(s)

In this section, we first review the recurrence formulas of IDR(s) for solving a system of linear equations, and then we discuss the work of Simoncini and Szyld in [87].

The Induced Dimension Reduction method is based on the following theorem. The main idea is to create approximation vectors $\mathbf{x}_m$ such that their corresponding residual vectors $\mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$ belong to the nested and shrinking subspaces $\mathcal{G}_j$. IDR(s) creates $s+1$ residual vectors in $\mathcal{G}_j$, and uses those vectors for the creation of the $s+1$ subsequent residuals in $\mathcal{G}_{j+1}$. This process is repeated iteratively until convergence.

Our implementation of IDR(s) is based on IDR(s) with biorthogonal residuals (see [103]). In practice, this variant has proved to be more stable, and is also slightly less expensive. Next, we present the recurrences used by this IDR(s) variant. For sake of simplicity, we introduce new notation. The subspace $\mathcal{S}$ is represented by the left null space of some full-rank $n \times s$ matrix $P = [\mathbf{p}_1, \ldots \mathbf{p}_s]$ (called shadow space). The superindex of a vector or a scalar represents the number of subspace $\mathcal{G}_j$ where the current residual belongs. The subindex represents the position in the sequence of intermediate residuals. For $\mathbf{r}_k^{(j)}$ represents the kth residual in $\mathcal{G}_j$. The first residual vectors in $\mathcal{G}_{j+1}$ and its respective approximation are

$$\mathbf{x}_0^{(j+1)} = \mathbf{x}_s^{(j)} + \omega_{j+1}\mathbf{r}_s^{(j)},$$

and

$$\mathbf{r}_0^{(j+1)} = (I - \omega_{j+1}A)\mathbf{r}_s^{(j)}, \tag{4.4}$$

and the recurrences to create the intermediate residuals in $\mathcal{G}_{j+1}$, are

$$\mathbf{x}_k^{(j+1)} = \mathbf{x}_{k-1}^{(j+1)} + \beta_k^{(j+1)}\mathbf{u}_k^{(j+1)},$$

and

$$\mathbf{r}_k^{(j+1)} = \mathbf{r}_{k-1}^{(j+1)} - \beta_k^{(j+1)}\mathbf{g}_k^{(j+1)}, \quad \text{for } k = 1, 2, \ldots, s. \tag{4.5}$$

The scalar $\beta_k^{(j+1)}$ is selected such that

$$\langle \mathbf{r}_k^{(j+1)}, \mathbf{p}_k \rangle = 0. \tag{4.6}$$

The direction vectors are defined as

$$\mathbf{u}_k^{(j+1)} = \hat{\mathbf{u}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)}\mathbf{u}_i^{(j+1)}, \tag{4.7}$$

and

$$\mathbf{g}_k^{(j+1)} = \hat{\mathbf{g}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)}\mathbf{g}_i^{(j+1)}, \tag{4.8}$$

where the vector $\hat{\mathbf{u}}_k^{(j+1)}$ and $\hat{\mathbf{g}}_k^{(j+1)}$ are

$$\hat{\mathbf{g}}_k^{(j+1)} = A\hat{\mathbf{u}}_k^{(j+1)}, \tag{4.9}$$

$$\hat{\mathbf{u}}_k^{(j+1)} = \omega_{j+1}\left(\mathbf{r}_{k-1}^{(j+1)} - \sum_{i=s-k}^{s} \gamma_i^{(j+1)}\mathbf{g}_i^{(j)}\right) + \sum_{i=s-k}^{s} \gamma_i^{(j+1)}\mathbf{u}_i^{(j)}. \tag{4.10}$$

The scalars $\{\alpha_i^{(j+1)}\}_{i=1}^{k-1}$ in (4.7) and (4.8) are selected, such that

$$\langle \mathbf{g}_k^{(j+1)}, \mathbf{p}_i\rangle = 0 \quad \text{for } i = 1, \ldots, k-1, \tag{4.11}$$

and the scalars $\{\gamma_i\}_{i=k}^{s}$ in (4.10) are selected as

$$\left\langle \mathbf{r}_{k-1}^{(j+1)} - \sum_{i=k}^{s} \gamma_i^{(j+1)}\mathbf{g}_i^{(j)}, \mathbf{p}_j\right\rangle = 0. \tag{4.12}$$

The conditions (4.6), (4.11), and (4.12) not only ensure that the residual $\mathbf{r}_k^{(j+1)}$ belongs to $\mathcal{G}_{j+1}$, but also, that the residual $\mathbf{r}_k^{(j+1)}$ is orthogonal to the vectors $\mathbf{p}_1$, $\mathbf{p}_2$, $\ldots$, $\mathbf{p}_k$ for $k = 1, 2, \ldots, s$.

An important property needed for the deduction of the IDR(s)-Hessenberg relation to be presented in the section 4.3, is that for any IDR(s) variant a residual in $\mathcal{G}_j$ can be also written as

$$\mathbf{r}_k^{(j)} = \Omega_j(A)\Psi(A)_{s\times j+k}\mathbf{r}_0, \tag{4.13}$$

where

$$\Omega_j(t) = \prod_{i=0}^{j}(1-\omega_i t), \quad \omega_i \neq 0, \quad i = 1, \ldots, j, \tag{4.14}$$

$\Omega_0(t) = 1$, and $\Psi_m(t)$ is a multi-Lanczos-type polynomial [106] of order m, that uses $s+2$ terms recurrences such that $\Psi_0 = 1$ (see section 5 in [95]). When the first residual vector is created in $\mathcal{G}_{j+1}$, the polynomial $\Omega_j(A)$ increases by one degree. Then, the degree of the polynomial $\Psi_m(A)$ is increased by one for each matrix-vector product during the creation of the others intermediate residuals.

4.2.1 IDR(s) as a Petrov-Galerkin method and Ritz-IDR(s)

As we mention in the introduction of this chapter, Simoncini and Szyld showed that IDR(s) can be viewed as a Petrov-Galerkin method in [87]. Particularly IDR(s) finds the approximation $\mathbf{x}_{k+1}$ in the right or search subspace $\mathbf{x}_0 + \mathcal{K}_{k+1}(A, \mathbf{r}_0)$, by imposing the condition that $\mathbf{r}_{k+1}$ is orthogonal to the subspace $\mathcal{W}_j$, defined as

$$\mathcal{W}_j = \Omega_j(A^H)^{-1}\mathcal{K}_j(A^H, P), \tag{4.15}$$

where $\Omega_j(A)$ is the polynomial defined in (4.14), and $\mathcal{K}_j(A^H, P)$ is the block Krylov subspace of order j, associated with the matrix A and the block P.

This link between IDR(s) and the rational block subspaces leads to the development of the variant Ritz-IDR(s). The authors in [87] argue that selecting the scalars ω_j as the inverse of Ritz values of the coefficient matrix A, is a good choice for the creation of the left space $\mathcal{W}_j$. This selection enriches the left subspace with information about the associated eigencomponents. This would damp the eigenvector components from the residual vector in a quick way, which leads to a faster convergence. The Ritz values required are computed with a call to an eigenvalue routine as the Arnoldi method. Note that Ritz-IDR(s) might require complex arithmetics even when the matrix and right-hand size vector are real, in the case when complex Ritz values are encountered.

In the following section we present how to obtain an Hessenberg relation from the IDR(s)-recurrences. Using this Hessenberg relation, we can obtain approximations to the eigenvalues of the coefficient matrix, and in this form we obtain a self-contained variant of the Ritz-IDR(s). To distinguish it, we label our algorithm as SC-Ritz-IDR(s).

4.3 Part 1: Accelerating IDR(s) using the Ritz values

IDR(s) has been previously used to obtain spectral information of a matrix. In [45], the authors adapt IDR(s) to solve the eigenvalue problem, and they obtain the matrices $\hat{H}_m$ and T_m from a generalized Hessenberg relation

$$AW_mT_m = W_m\hat{H}_m + \hat{\mathbf{f}}\mathbf{e}_m^T.$$

where $W_m \in \mathbb{C}^{n\times m}$ (not explicitly available) represents a Krylov subspace basis for $\mathcal{K}(A, \mathbf{w}_1)$, T_m is an s-banded, upper triangular matrix; $\hat{H}$ is an s-banded, upper Hessenberg matrix, and $\hat{\mathbf{f}} \in \mathbb{C}^n$. The approximation of the eigenvalues of A are obtained from the eigenvalue pencil $(\hat{H}_m, T_m)$. In [5], the authors create a standard Hessenberg relation

$$AW_m = W_mH_m + \mathbf{f}\mathbf{e}_m^T, \tag{4.16}$$

where $W_m \in \mathbb{C}^{n\times m}$, and H_m is a Hessenberg matrix. This matrix H_m has the same eigenvalues as the matrix pencil $(\hat{H}_m, T_m)$.

The mentioned works [45] and [5] target specifically the eigenvalue/eigenvector approximation problem. Next, we describe how to obtain a matrix H_m part of a standard Hessenberg relation (4.16) from the underlying IDR(s)-recurrences used to solve systems of linear equation. This allows us to obtain the solution of a system of linear equation whose coefficient matrix is A, and at the same time obtain approximations to the eigenvalues of this matrix. Particularly, we use this spectral information as is suggested in [87], and we proposed a Ritz-IDR(s) variant labeled as SC-Ritz-IDR(s).

To derive this Hessenberg matrix, let us consider the IDR(s) relations described in section 4.2. Substituting (4.8)–(4.10) in (4.5), we obtain

$$\begin{aligned}
\frac{\mathbf{r}_{k-1}^{(j+1)} - \mathbf{r}_k^{(j+1)}}{\beta_k^{(j+1)}} &= \hat{\mathbf{g}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= A\hat{\mathbf{u}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= A\left[\omega_{j+1}\left(\mathbf{r}_{k-1}^{(j+1)} - \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{g}_i^{(j)}\right) + \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{u}_i^{(j)}\right] - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= \omega_{j+1} A\mathbf{r}_{k-1}^{(j+1)} - \omega_{j+1} A \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{g}_i^{(j)} + \sum_{i=s-k}^{s} \gamma_i^{(j+1)} A\mathbf{u}_i^{(j)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= \omega_{j+1} A\mathbf{r}_{k-1}^{(j+1)} - \omega_{j+1} A \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{g}_i^{(j)} + \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{g}_i^{(j)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= \omega_{j+1} A\mathbf{r}_{k-1}^{(j+1)} + (I - \omega_{j+1} A) \sum_{i=s-k}^{s} \gamma_i^{(j+1)} \mathbf{g}_i^{(j)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)} \mathbf{g}_i^{(j+1)} \\
&= \omega_{j+1} A\mathbf{r}_{k-1}^{(j+1)} + (I - \omega_{j+1} A) \sum_{i=s-k}^{s} \frac{\gamma_i^{(j+1)}}{\beta_i^{(j)}} (\mathbf{r}_{i-1}^{(j)} - \mathbf{r}_i^{(j)}) - \sum_{i=1}^{k-1} \frac{\alpha_i^{(j+1)}}{\beta_i^{(j+1)}} (\mathbf{r}_{i-1}^{(j+1)} - \mathbf{r}_i^{(j+1)}).
\end{aligned}$$

From the equations above, we obtain the following relation

$$\begin{aligned}
\omega_{j+1} A\mathbf{r}_{k-1}^{(j+1)} = \frac{\mathbf{r}_{k-1}^{(j+1)} - \mathbf{r}_k^{(j+1)}}{\beta_k^{(j+1)}} &- (I - \omega_{j+1} A) \sum_{i=s-k}^{s} \frac{\gamma_i^{(j+1)}}{\beta_i^{(j)}} (\mathbf{r}_{i-1}^{(j)} - \mathbf{r}_i^{(j)}) \\
&+ \sum_{i=1}^{k-1} \frac{\alpha_i^{(j+1)}}{\beta_i^{(j+1)}} (\mathbf{r}_{i-1}^{(j+1)} - \mathbf{r}_i^{(j+1)}).
\end{aligned} \tag{4.17}$$

Using (4.13), we obtain that each vector in $\mathcal{G}_j$ can be written as

$$\mathbf{r}_i^{(j)} = \Omega_j(A)\hat{\mathbf{r}}_i^{(j)} \qquad \text{for } i = 0, \ldots, s, \tag{4.18}$$

and equivalently, any residuals in $\mathcal{G}_{j+1}$ can be written as

$$\mathbf{r}_i^{(j+1)} = \Omega_{j+1}(A)\hat{\mathbf{r}}_i^{(j+1)} \qquad \text{for } i = 0, \ldots, s. \tag{4.19}$$

Taking into account (4.19) and (4.18), we can multiply (4.17) by $\Omega_{j+1}(A)^{-1}$ and obtain

$$\begin{aligned}\omega_{j+1} A \hat{\mathbf{r}}_{k-1}^{(j+1)} = {}& \frac{\hat{\mathbf{r}}_{k-1}^{(j+1)} - \hat{\mathbf{r}}_{k}^{(j+1)}}{\beta_k^{(j+1)}} - \sum_{i=s-k}^{s} \frac{\gamma_i^{(j+1)}}{\beta_i^{(j)}} (\hat{\mathbf{r}}_{i-1}^{(j)} - \hat{\mathbf{r}}_{i}^{(j)}) \\ & + \sum_{i=1}^{k-1} \frac{\alpha_i^{(j+1)}}{\beta_i^{(j+1)}} (\hat{\mathbf{r}}_{i-1}^{(j+1)} - \hat{\mathbf{r}}_{i}^{(j+1)}).\end{aligned} \tag{4.20}$$

The set of vectors $\hat{\mathbf{r}}_i$ represents the Krylov basis associated with the polynomial $\Psi(A)$. In fact, one can see that the basis grows with the degree of the polynomial $\Psi(A)$. Substituting (4.18) and (4.19) in (4.4), we obtain that

$$\hat{\mathbf{r}}_0^{(j+1)} = \hat{\mathbf{r}}_s^{(j)}. \tag{4.21}$$

This implies that every $s+1$ matrix-vector products, IDR(s) creates s new vectors basis $\hat{\mathbf{r}}_i$. Using (4.21), we can rewrite (4.20) as

$$\begin{aligned}\omega_{j+1} A \hat{\mathbf{r}}_{k-1}^{(j+1)} = {}& -\frac{\gamma_{s-k}^{(j+1)}}{\beta_{s-k}^{(j+1)}} \hat{\mathbf{r}}_{s-k-1}^{(j)} - \sum_{i=s-k}^{s-1} \left(\frac{\gamma_{i+1}^{(j+1)}}{\beta_{i+1}^{(j)}} - \frac{\gamma_i^{(j+1)}}{\beta_i^{(j)}} \right) \hat{\mathbf{r}}_i^{(j)} \\ & + \left(\frac{\gamma_s^{(j+1)}}{\beta_s^{(j+1)}} + \frac{\alpha_1^{(j+1)}}{\beta_1^{(j+1)}} \right) \hat{\mathbf{r}}_s^{(j)} + \sum_{i=1}^{k-1} \left(\frac{\alpha_{i+1}^{(j+1)}}{\beta_{i+1}^{(j+1)}} - \frac{\alpha_i^{(j+1)}}{\beta_i^{(j+1)}} \right) \hat{\mathbf{r}}_i^{(j+1)} \\ & - \frac{1}{\beta_k^{(j+1)}} \hat{\mathbf{r}}_k^{(j+1)}.\end{aligned} \tag{4.22}$$

One can see in (4.22) that the vector $A\hat{\mathbf{r}}_{k-1}^{(j+1)}$ is a linear combination of the vectors $\{\hat{\mathbf{r}}_i^{(j)}\}_{i=s-k-1}^{s}$ and $\{\hat{\mathbf{r}}_i^{(j+1)}\}_{i=1}^{k}$. This defines a Hessenberg relation of the form

$$A\hat{R}_{\bar{m}} = \hat{R}_{\bar{m}+1} \bar{H}_{\bar{m}}, \tag{4.23}$$

where $\bar{m}$ is the number of intermediate residuals created by IDR(s), and $\hat{R}_{\bar{m}}$ is a Krylov subspace basis defined as

$$\hat{R}_{\bar{m}} = [\hat{\mathbf{r}}_0^{(0)}, \ldots, \hat{\mathbf{r}}_s^{(0)}, \hat{\mathbf{r}}_1^{(1)}, \ldots, \hat{\mathbf{r}}_s^{(1)}, \ldots, \hat{\mathbf{r}}_1^{(j)}, \ldots, \hat{\mathbf{r}}_s^{(j)}, \hat{\mathbf{r}}_1^{(j+1)}, \ldots, \hat{\mathbf{r}}_k^{(j+1)}]_{n \times \bar{m}}. \tag{4.24}$$

The vectors $\hat{\mathbf{r}}_i$ are not constructed explicitly, however, it is easy to see that

$$\hat{\mathbf{r}}_0^{(0)} = \mathbf{r}_0. \tag{4.25}$$

The matrix H is an upper and $s+1$ banded Hessenberg matrix whose columns are defined as

$$\mathbf{H}_\ell = \begin{bmatrix} 0 \\ \vdots \\ h_{\ell-s,\ell} \\ \vdots \\ h_{\ell+1,\ell} \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{C}^{\bar{m}+1},$$

where

$$\begin{bmatrix} h_{\ell-s,\ell} \\ h_{\ell-s+1,\ell} \\ \vdots \\ h_{\ell-s+m,\ell} \\ h_{\ell-s+m+1,\ell} \\ \vdots \\ h_{\ell+1,\ell} \end{bmatrix} = \frac{1}{\omega_{j+1}} \begin{bmatrix} -\frac{\gamma_{s-k}^{(j+1)}}{\beta_{s-k}^{(j+1)}} \\ \frac{\gamma_{s-k+1}^{(j+1)}}{\beta_{s-k+1}^{(j)}} - \frac{\gamma_{s-k}^{(j+1)}}{\beta_{s-k}^{(j)}} \\ \vdots \\ \frac{\gamma_{s}^{(j+1)}}{\beta_{s}^{(j)}} + \frac{\alpha_{1}^{(j+1)}}{\beta_{1}^{(j+1)}} \\ \frac{\alpha_{2}^{(j+1)}}{\beta_{2}^{(j+1)}} - \frac{\alpha_{1}^{(j+1)}}{\beta_{1}^{(j+1)}} \\ \vdots \\ -1/\beta_{k}^{(j+1)} \end{bmatrix} \in \mathbb{C}^{s+2},$$

Our implementation of SC-Ritz-IDR(s) is based on the IDR(s) with biorthogonal residuals. The memory consumption of SC-Ritz-IDR(s) is similar to that of IDR(s) (see section 3.5 in [103]). The sets of coefficients $\{\alpha_i\}_{i=1}^{s}$, $\{\gamma_i\}_{i=1}^{s}$, and $\{\beta_i\}_{i=1}^{s}$, used in SC-Ritz-IDR(s), are stored in three extra vectors of size s. Algorithm 11 shows an implementation of SC-Ritz-IDR(s).

4.3.1 Numerical experiments

To illustrate the numerical behavior of the proposed algorithm, we repeat all the experiments presented in [87]. We compare our proposed variant SC-Ritz-IDR(s) with IDR(s), Ritz-IDR(s) and full GMRES. All the experiments are performed in Matlab 2015a running on a 64 bit GNU/Debian Linux computer with 32 GB of RAM. The right-hand side vector $\mathbf{b} = \hat{\mathbf{b}}/\|\hat{\mathbf{b}}\|$ with $\hat{\mathbf{b}} = \mathbf{1}$, and the initial vector is $\mathbf{x}_0 = \mathbf{0}$. As stopping criterion, we use

$$\frac{\|\mathbf{b} - A\mathbf{x}_k\|}{\|\mathbf{b}\|} < \epsilon,$$

with $\epsilon = 10^{-10}$.

Algorithm 11 IDR(s) accelerated with Ritz values

```
1: procedure IDR($A$, $\mathbf{b}$, $s$, $tol$, $\mathbf{x}_0$)
2:     Input: $A \in \mathbb{C}^{n\times n}$, $\mathbf{b} \in \mathbb{C}^n$, $s \in \mathbb{N}^+$, $tol \in (0, 1)$, $\mathbf{x}_0 \in \mathbb{C}^n$.
3:     $\mathbf{x} = \mathbf{x}_0$, $\mathbf{r} = \mathbf{b} - A\mathbf{x}$
4:     $P$ a random matrix in $\mathbb{C}^{n\times s}$.
5:     $G = 0 \in \mathbb{C}^{n\times s}$, $U = 0 \in \mathbb{C}^{n\times s}$
6:     $M = I_s \in \mathbb{C}^{s\times s}$.
7:     $\omega = 1.0$, $\ell = 0$, $H_{\bar{m}} = 0 \in \mathbb{C}^{\bar{m}+1\times\bar{m}}$, $\mathbf{c} = \mathbf{0}$, $\alpha = \mathbf{0}$, $\beta = \mathbf{0} \in \mathbb{C}^s$.
8:     while $\|\mathbf{r}\| \leq tol \times \|\mathbf{b}\|$ do                          ▷ Loop over$\mathcal{G}_j$ spaces
9:         $\mathbf{f} = P^H\mathbf{r}$
10:        for $k = 1$ to $s$ do            ▷ Compute $s$ independent vectors $\mathbf{g}_k$ in $\mathcal{G}_j$ space
11:            Solve $\mathbf{c}$ from $M\mathbf{c} = \mathbf{f}$, $(\gamma_1, \dots, \gamma_s)^H = \mathbf{c}$          ▷ Note that $M = P^HG$
12:            $\mathbf{v} = \mathbf{r} - \sum_{i=k}^{s} \gamma_i\mathbf{g}_i$
13:            $\mathbf{v} = B^{-1}\mathbf{v}$                                 ▷ Preconditioning operation
14:            $\mathbf{u}_k = \omega\mathbf{v} + \sum_{i=k}^{s} \gamma_i\mathbf{g}_i$
15:            $\mathbf{g}_k = A\mathbf{u}_k$
16:            for $i = 1$ to $k - 1$ do                          ▷ Make $\mathbf{g}_k$ orthogonal to $P$
17:                $\alpha_i = \langle\mathbf{g}_k, \mathbf{p}_i\rangle/\mu_{i,i}$
18:                $\mathbf{g}_k = \mathbf{g}_k - \alpha_i\mathbf{g}_i$
19:                $\mathbf{u}_k = \mathbf{u}_k - \alpha_i\mathbf{u}_i$
20:            end for
21:            $\mu_{i,k} = \langle\mathbf{g}_k, \mathbf{p}_i\rangle$, $M_{i,k} = \mu_{i,k}$, for $i = k, \dots, s$              ▷ Update $M$
22:            $\beta_k = \phi_k/\mu_{k,k}$                        ▷ Now $\langle\mathbf{r}, \mathbf{p}_i\rangle = 0$ for $i = 1, \dots, k$
23:            $\mathbf{r} = \mathbf{r} - \beta_k\mathbf{g}_k$
24:            $\mathbf{x} = \mathbf{x} + \beta_k\mathbf{u}_k$
25:            if $k + 1 \leq s$ then
26:                $\mathbf{f}_i = 0$ for $i = 1, \dots, k$
27:                $\mathbf{f}_i = \mathbf{f}_i - \beta_k M_{i,k}$ for $i = k + 1, \dots, s$
28:            end if
29:            $\ell = \ell + 1$
30:            if $\ell \leq \bar{m}$ then
31:                $H_{\ell-s:\ell-k,\ell} = \mathbf{c}_{k:s}/\beta_{k:s}$
32:                $H_{\ell-k+1:\ell-1,\ell} = \alpha_{1:k-1}/\beta_{1:k-1}$
33:                $H_{\ell,\ell} = 1.0/\beta_k$
34:                $H_{\ell-s+1:\ell+1,\ell} = H_{\ell-s+1:\ell+1,\ell} + H_{\ell-s:\ell,\ell}$
35:                $H_{\ell-s:\ell+1,\ell} = H_{\ell-s:\ell+1,\ell}/\omega$
36:            end if
37:            Overwrite $k$th columns of $G$ and $U$ by $\mathbf{g}_k$ and $\mathbf{u}_k$ respectively.
38:        end for                                                   ▷ Entering $\mathcal{G}_{j+1}$
39:        $\mathbf{v} = B^{-1}\mathbf{r}$                                     ▷ Preconditioning operation
40:        $\mathbf{t} = A\mathbf{v}$
41:        if $\ell \leq \bar{m}$ then                                         ▷ Select new $\omega$
42:            $\omega$ is selected using the converge maintenance strategy [103].
43:        else
44:            $\omega$ is selected using the spectral information provided by $H_{\bar{m}}$.
45:        end if
46:        $\mathbf{r} = \mathbf{r} - \omega\mathbf{t}$
47:        $\mathbf{x} = \mathbf{x} + \omega\mathbf{v}$
48:    end while
49:    Return $\mathbf{x}$ and $H_{\bar{m}}$ (if required).
50: end procedure
```

For Ritz-IDR(s) and SC-Ritz-IDR(s), we use as parameter

$$\omega_j = \frac{1}{\lambda_i}, \tag{4.26}$$

where λ_i is an eigenvalue of the matrix $H_{\bar{m}}$. We select $\bar{m} = 20$ and the 15 smallest magnitude eigenvalues. For Ritz-IDR(s), the matrix $H_{\bar{m}}$ is obtained with a preliminary call to the Arnoldi method. In the case of SC-Ritz-IDR(s) the matrix $H_{\bar{m}}$ is computed as is explained in section 4.3. Before the creation of the matrix $H_{\bar{m}}$, SC-Ritz-IDR(s) uses the converge maintenance strategy, proposed in [103], to select the first ω_j parameters.

Convection-diffusion-reaction equation examples

The linear systems of equations used in the next three examples are based on the finite difference discretization of the simple convection-diffusion-reaction model problem

$$-\epsilon \triangle u + \mathbf{v}^T \nabla u + \rho u = f, \qquad \text{in } \Omega = [0,\, 1]^d \tag{4.27}$$

with $d = 2$ or $d = 3$, and homogeneous Dirichlet boundary conditions on $\partial\Omega$. Particularly, it is known that IDR(s) with $s > 1$ outperforms BiCGStab [101] when the $\|\mathbf{v}\| \gg \epsilon$ (see for example [92] and [6]).

Experiment 4.1. In this example the coefficient matrix A is given by the finite difference discretization of (4.27) for the 2D case. The physical parameters used are $\epsilon = 1$, $\mathbf{v} = [4,\, 0]^T$, and $\rho = 400$. We discretize the domain Ω using 21 points in each direction. Figure 4.1 (a) shows the convergence of the norm of the residual for the matrix A of order 400 generated with the parameters described. Ritz-IDR(s) and SC-Ritz-IDR(s) do not show any improvement over IDR(s). However, using a convection-dominated taking 41 points in each direction and $\epsilon = 1$, $\mathbf{v} = [80,\, 0]^T$, and $\rho = 1600$, we can see in Figure 4.1 (b) a better performance of Ritz-IDR(s) and SC-Ritz-IDR(s) over IDR(s).

Experiment 4.2. We consider two matrices of order 8000 from the discretization of the 3D problem (4.27) with $\epsilon = 1$, $\mathbf{v} = \beta[1,\, 1,\, 1]$, and $\rho = 0$. First using $\beta = 100$, we can see in Figure 4.2 (a) a similar behavior between the IDR(s) variants. However, Ritz-IDR(s) and SC-Ritz-IDR(s) are clearly superior with respect to the IDR(s) when the parameter β is increased to 500 (see Figure 4.2 (b)).

Experiment 4.3. The coefficient matrix used in this example is the unsymmetric matrix of order 8000 that comes from the finite difference discretization of the 3D (4.27), with parameter $\epsilon = 1$, $\rho = 0$, and $\mathbf{v} = [0,\, 0,\, 1000]^T$. As in part (b) of previous example, IDR(4) does not converge for the maximum number of iterations allowed, while Ritz-IDR(4) and SC-Ritz-IDR(4) converge using almost the same number of matrix-vector products (see Figure 4.3).

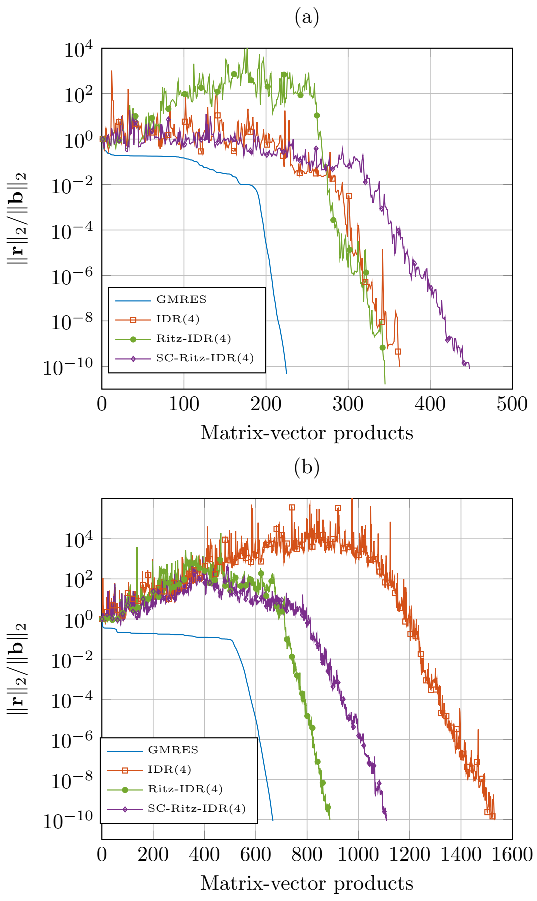

Figure 4.1: *(Example 4.1)* Evolution of the residual norm of full GMRES, IDR(4), Ritz-IDR(4), and SC-Ritz-IDR(4). (a) Diffusion-dominated example. (b) Convection-dominated example.

Examples from Matrix Market

The matrices used in the next two examples are part of the Matrix Market collection [18].

Experiment 4.4. We consider the highly indefinite matrix Sherman5 of order 3312. As is reported in [87], Ritz-IDR(s) diverges for this example. SC-Ritz-IDR(s) exhibits a similar behavior. On the other hand, Figure 4.4 shows that both Ritz-IDR(s) variants converge using the Incomplete LU factorization of the matrix $A + I$ as preconditioner with threshold tolerate 10^{-2}. In this example, IDR(s) and its variant behave similarly in term of matrix-vector products required.

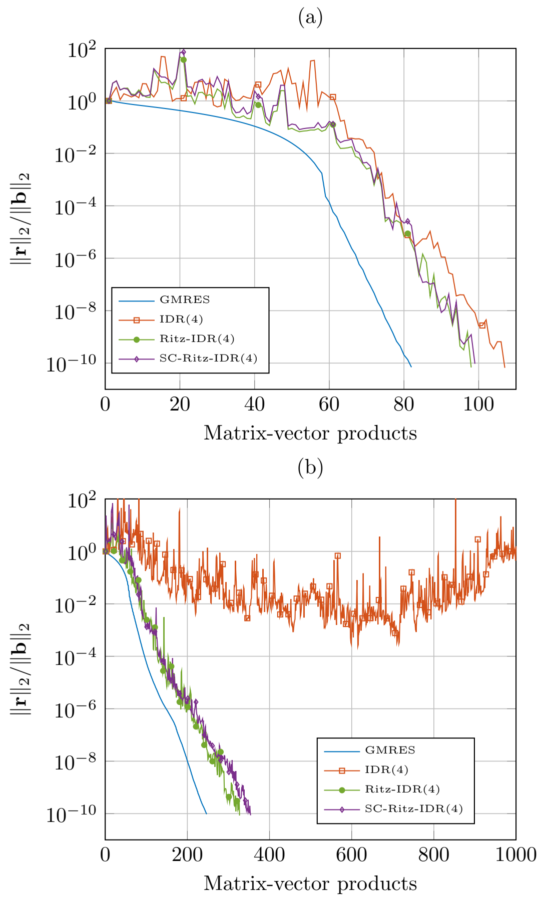

Figure 4.2: *(Example 4.2)* Evolution of the residual norm of full GMRES, IDR(4), Ritz-IDR(4), and SC-Ritz-IDR(4). (a) Diffusion-dominated example. (b) Convection-dominated example.

Experiment 4.5. In this example, we consider the linear system of equations ADD20 which arises from computer component design. In this example, we stop the algorithms when the relative residual norm is less than 1×10^{-8}. As is proposed in [87], we also consider 20 Leja points located in the interval where the 20 real Ritz values are located. The Leja points are computed using the algorithm proposed in [7]. Figure 4.5 shows a similar behavior between all the IDR(s) variants.

The numerical results show a similar behavior between Ritz-IDR(s) and SC-Ritz-IDR(s) in terms of matrix-vector products and convergence. Moreover, their computational requirements are virtually the same. The only difference is that SC-Ritz-IDR(s) requires storing a Hessenberg matrix of size

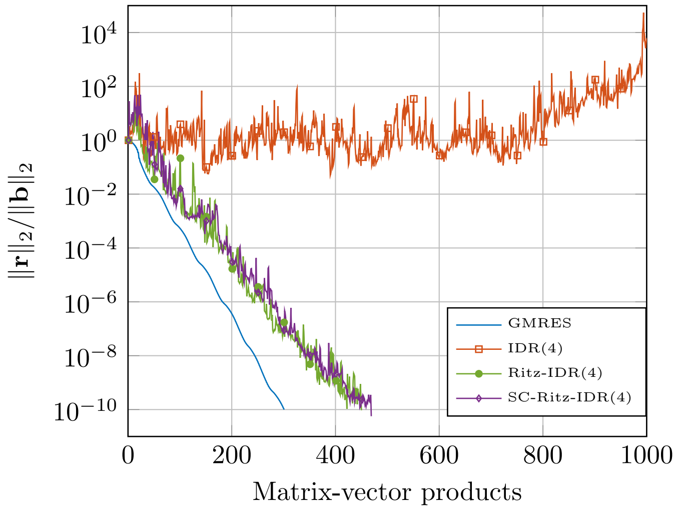

Figure 4.3: *(Example 4.3)* Evolution of the residual norm of full GMRES, IDR(4), Ritz-IDR(4), and SC-Ritz-IDR(4).

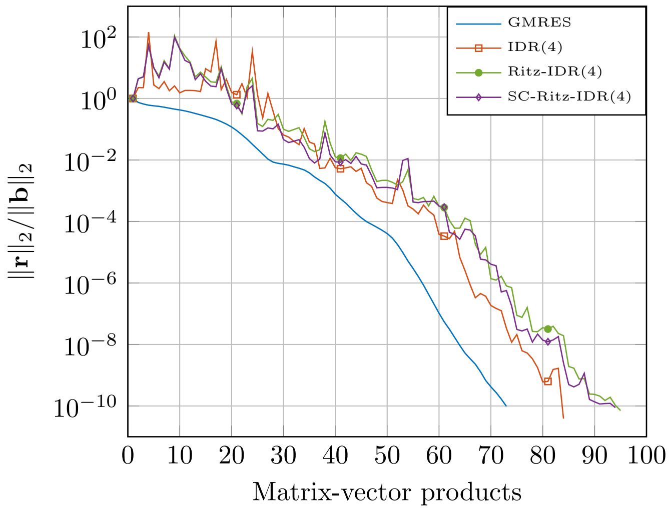

Figure 4.4: *(Example 4.4)* Evolution of the residual norm of full GMRES, IDR(4), Ritz-IDR(4), and SC-Ritz-IDR(4) for the matrix Sherman5 using ILU preconditioner.

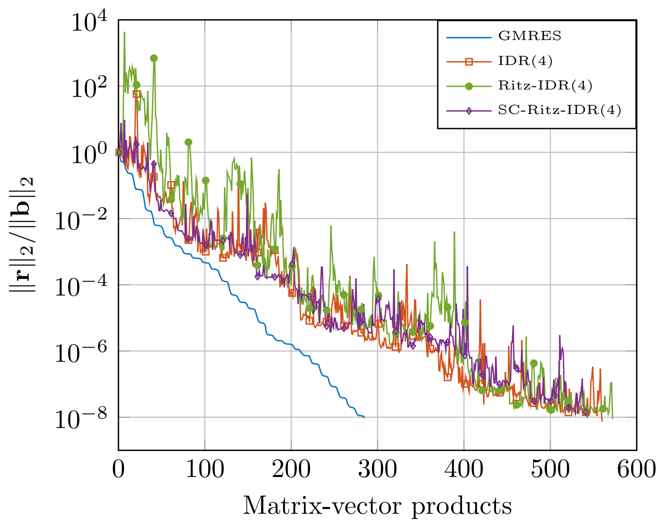

Figure 4.5: *(Example 4.5)* Evolution of the residual norm of full GMRES, IDR(4), Ritz-IDR(4), and SC-Ritz-IDR(4) for the matrix ADD20.

$\bar{m}$. The main advantage of SC-Ritz-IDR(s) is that it computes the Ritz-values "on-the-fly". Therefore, unlike in Ritz-IDR(s), the overhead of a call to an external eigensolver is avoided in SC-Ritz-IDR(s).

4.4 Part 2: Accelerating IDR(s) using Ritz vectors

In the previous sections we use the recurrences of IDR(s) to obtain an upper Hessenberg matrix H. From this matrix H, we obtain the Ritz values to accelerate the IDR(s) method. In this section, we incorporate the Ritz vectors to the Krylov basis generated by IDR(s). First, we present how to add additional vectors to the IDR(s) searching subspace basis, i.e., the augmented Krylov subspace

$$\mathcal{K}_{s+m}(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0,\, \mathbf{y}_1,\, \ldots, \mathbf{y}_s,\, A\mathbf{r}_0,\, \ldots, A^{m-1}\mathbf{r}_0\}. \tag{4.28}$$

Secondly, we use the matrix H to recover the Ritz vectors of the coefficient matrix, and add these Ritz vectors in IDR(s).

To add additional direction vectors to the Krylov basis created by IDR(s), we exploit the fact that $\mathcal{G}_0$ is $\mathbb{C}^n$. We can choose freely the first $s+1$ linearly independent direction vectors in IDR(s) and obtain their corresponding approximations and associated residuals. In the case of the biorthogonal variant, we have to ensure that each residual $\mathbf{r_i}$ is orthogonal to $\mathbf{p}_j$ for $i = 1, 2, \ldots, s$ and $j = 1, 2, \ldots, i$, and each vector $\mathbf{g}_i$ is orthogonal to $\mathbf{p}_j$ for $i = 1, 2, \ldots, s$

and $j = 1, 2, \ldots, i-1$. In order to do so, we present the Algorithm 12, to create the first s biorthogonal residuals.

Algorithm 12 Injecting basis vectors in $\mathcal{G}_0$

```
1:  Input: ${\mathbf{y}_i}_{i=1}^s$
2:  for $k = 1$ to $s$ do
3:      $\mathbf{u}_k = \mathbf{y}_k$
4:      $\mathbf{g}_k = A\mathbf{u}_k$
5:      for $i = 1$ to $k-1$ do                         ▷ Make $\mathbf{g}_k$ orthogonal to $P$
6:          $\alpha = \langle \mathbf{g}_k, \mathbf{p}_i \rangle / \mu_{i,i}$
7:          $\mathbf{g}_k = \mathbf{g}_k - \alpha \mathbf{g}_i$
8:          $\mathbf{u}_k = \mathbf{u}_k - \alpha \mathbf{u}_i$
9:      end for
10:     $\mu_{i,k} = \langle \mathbf{g}_k, \mathbf{p}_i \rangle$, $M_{i,k} = \mu_{i,k}$, for $i = k, \ldots, s$     ▷ Update $M$
11:     $\beta = \phi_k / \mu_{k,k}$     ▷ Make the residual orthogonal to $\mathbf{p}_i$ for $i = 1, \ldots, k$
12:     $\mathbf{r} = \mathbf{r} - \beta \mathbf{g}_k$
13:     $\mathbf{x} = \mathbf{x} + \beta \mathbf{u}_k$
14:     $\phi_i = 0$ for $i = 1, \ldots, k$
15:     $\phi_i = \phi_i - \beta \mu_{i,k}$ for $i = k+1, \ldots, s$
16: end for                                              ▷ Entering $\mathcal{G}_{j+1}$
```

To add the vectors $\{\mathbf{y}_i\}_{i=1}^s$ to the IDR(s), we should replace Algorithm 12 by the lines 5 and 6 in Algorithm 11. As is proposed in [61], [25], and [63], we use as extra basis vectors the Ritz vector associated with the smallest-magnitude Ritz values.

Experiment 4.6. *(A Motivating Example.)* To exemplify the idea of using the spectral information in the initial subspace $\mathcal{G}_0$, we consider solving a system of linear equations with the following bidiagonal matrix

$$A = \begin{bmatrix} 1\times10^{-8} & 1\times10^{-5} & & & & & \\ & 2\times10^{-8} & 1\times10^{-5} & & & & \\ & & \ddots & & & & \\ & & & 5\times10^{-8} & 1\times10^{-5} & & \\ & & & & 6 & 1\times10^{-5} & \\ & & & & & \ddots & \\ & & & & & & 100 \end{bmatrix}_{100\times100}, \tag{4.29}$$

and the right-hand side vector is $\mathbf{b} = \mathbf{1}$. We compare IDR(5) and IDR(5) with recycling. As recycling vectors, we use the five eigenvectors associated with the smallest magnitude eigenvalues of the bidiagonal matrix A. The initial guess vector is $\mathbf{x}_0 = \mathbf{0}$. Figure 4.6 shows the evolution of the norms of the residuals,

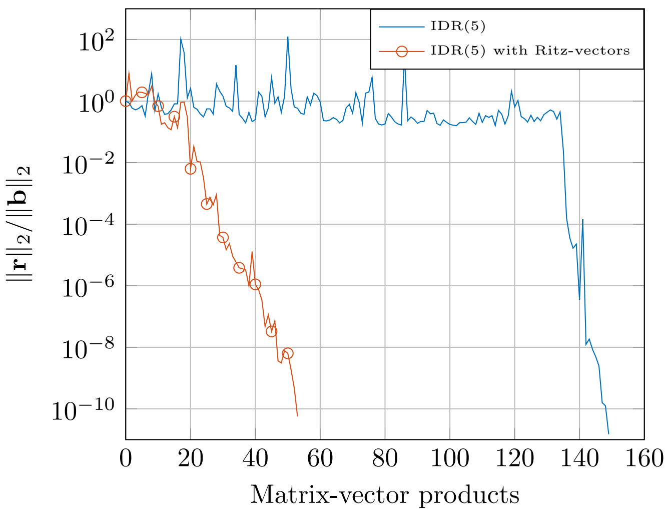

Figure 4.6: *(Example 4.6)* Evolution of the residual norm of full IDR(5) and IDR(5) with recycling with the four eigenvectors associated with the smallest magnitude eigenvalues of the matrix (4.29).

one can see a considerable reduction in the number of matrix-vector products for IDR(s) with recycling.

It is worth mentioning the recently proposed M(s)STAB(ℓ) method by Neuenhofen [66]. M(s)STAB(ℓ) is a variant of the IDRstab [92], that is specialized to solve sequences of systems of linear equations where the coefficient matrix is constant. Based on a generalization of the IDR theorem, M(s)STAB(ℓ) uses as initial vectors basis in $\mathcal{G}_0$ the last $s+1$ vectors in the subspace $\mathcal{G}_j$, which were created during the solution of the previous system of linear equation. In this form, this method reduces the computation and accelerates the solution of (4.2).

4.4.1 Adding the Ritz vectors to IDR(s): application to sequence of system of linear equations

Here we present the main application of IDR(s) with recycling, the solution of a sequence of systems of linear equations. We consider the case where the coefficient matrix A is constant, and the right-hand side vectors $\{\mathbf{b}^{(i)}\}_{i=1}^{p}$ are not available simultaneously.

The main idea is to compute a subset of Ritz vectors of the matrix A during the solution of the first system of linear equation, and then use these Ritz vectors to accelerate the solution of the subsequent systems of linear equations. The upper Hessenberg matrix $H_{\bar{m}} \in \mathbb{C}^{\bar{m}\times\bar{m}}$ is computed using Algorithm 11. To compute the Ritz vectors after the first execution of IDR(s), we need to compute the Krylov basis $\hat{R}$ in (4.23). To compute this $\hat{R}$, we use (4.13) and obtain that,

$$\hat{\mathbf{r}}_0 = \mathbf{r}_0, \tag{4.30}$$

and taking into account the upper Hessenberg structure of the matrix $H_{\bar{m}}$, we obtain the following recurrence formula for the vector $\hat{\mathbf{r}}_i$

$$\hat{\mathbf{r}}_i = \frac{1}{h_{i+1,i}} \left[A\hat{\mathbf{r}}_{i-1} - \sum_{j=\max(0,i-s)}^{i-1} h_{j,i}\hat{\mathbf{r}}_j \right]. \tag{4.31}$$

Because (4.31) uses only the last $s+1$ vectors, we can even obtain the Ritz vector saving temporally only the last $s+1$ basis vectors. Algorithm 13 presents how to obtain the Ritz vectors of A, after we have obtained the matrix H.

Algorithm 13 Obtaining the Ritz vectors

```
 1: procedure Ritz vectorsIDR($A$, $s$, $H$, $\mathbf{r}_0$)
 2:     Input: $A \in \mathbb{C}^{n\times n}$, $s \in \mathbb{N}^+$, $\mathbf{x} \in \mathbb{C}^n$.
 3:     Obtain $(\lambda_i, \hat{\mathbf{y}}_i)$ as the eigenpairs associated with the smallest magni-
    tude eigenvalues of $H$.
 4:     $\hat{\mathbf{r}}_0 = \mathbf{r}_0$
 5:     $Y = \hat{\mathbf{r}}_0 \times [[\hat{\mathbf{y}}_1]_1, [\hat{\mathbf{y}}_2]_1, \ldots, [\hat{\mathbf{y}}_{\bar{m}}]_1]$
 6:     for $i = 1$ to $\bar{m} - 1$ do
 7:         $\hat{\mathbf{r}}_i = \frac{1}{h_{i+1,i}} \left[ A\hat{\mathbf{r}}_{i-1} - \sum_{j=\max(0,i-s)}^{i-1} h_{j,i}\hat{\mathbf{r}}_j \right]$
 8:         $Y = Y + \hat{\mathbf{r}}_i \times [[\hat{\mathbf{y}}_1]_{i+1}, [\hat{\mathbf{y}}_2]_{i+1}, \ldots, [\hat{\mathbf{y}}_{\bar{m}}]_{i+1}]$
 9:     end for
10:     return $\{\lambda\}_{i=1}^{\bar{m}}, Y$.
11: end procedure
```

Once we compute the s Ritz vectors associated with the smallest magnitude, we proceed to use these vectors in IDR(s) with recycling to solve the remaining systems of linear equations. Algorithm 14 summarizes this procedure.

4.4.2 Numerical experiments

In this section, we conduct two numerical experiments of solving sequences of systems of linear equations (Algorithm 14). We use the same computer setting as is described in section 4.3.1. The stopping criterion consider in this experiment is

$$\frac{\|\mathbf{b}_i - A\mathbf{x}_k\|}{\|\mathbf{b}_i\|} < 10^{-6}, \qquad \text{for } i = 1, 2, \ldots, p.$$

Algorithm 14 IDR(s) with recycling for sequences of system of linear equations

```
1: procedure IDR($A$, $\{\mathbf{b}_i\}$, $s$, $tol$, $\mathbf{x}_0$)
2:     call IDR($A$, $\mathbf{b}_1$, $s$, $tol$, $\mathbf{x}_0$) to obtain $\mathbf{x}_1$ and the matrix $H_{\bar{m}}$ (Algorithm 11).
3:     call Ritz vectorsIDR($A$, $s$, $H_{\bar{m}}$, $\mathbf{r}_1$) to obtain the Ritz vectors $\{\mathbf{y}_j\}_{j=1}^{s}$ (Algorithm 13)
4:     for each right-hand side vector $\mathbf{b}_i$ with $i = 2, 3, \ldots, p$ do
5:         call IDR($s$) to solve $A\mathbf{x}_i = \mathbf{b}_i$ with the Ritz-vector $\{\mathbf{y}_j\}_{j=1}^{s}$.
6:     end for
7:     return $\mathbf{x}_i$ for $i = 1, \ldots, p$
8: end procedure
```

The initial guess for the first system of linear equations is the zero vector, and for the subsequent linear systems, we use the approximate solution of the previous linear system of equations.

Experiment 4.7. In this example, we consider the linear time-dependent convection-diffusion-reaction

$$\frac{\partial u}{\partial t} + \mathbf{v}^T \nabla u = \epsilon \Delta u + \rho u + f \tag{4.32}$$

with homogeneous Dirichlet conditions on the unit cube, and $u(t_0) = \mathbf{0}$, $\mathbf{v} = [1,\ 1,\ 1]$, $\epsilon = 0.1$ (diffusion-dominated) or $\epsilon = 0.005$ (convection-dominated), the reaction parameter ρ is 5, the function f is obtained from

$$u = \sqrt{x(1-x)y(1-y)z(1-z)}.$$

We solve (4.32) using Euler backward for time integration for $t \in [0,\ 10]$ with $\delta t = 1$. For space discretization, we use central finite differences with $h = 0.02$ obtaining a linear system of equations of size 125000×125000 per time-step. Figures 4.7 and 4.8 show the residual norm behavior for full GMRES, GCROT, and IDR(s) with and without Ritz vector enrichment. First, we can see a good decrement in number of matrix-vector multiplication when IDR(s) is enriched with the Ritz vectors. Second, the long-recurrences methods solve all the systems of linear equations using less number of matrix-vector multiplications. However, Tables 4.1 and 4.2 show that IDR(s) with Ritz vectors solves the convection and diffusion-dominated problems much faster that GMRES and GCROT, and other short-recurrences methods.

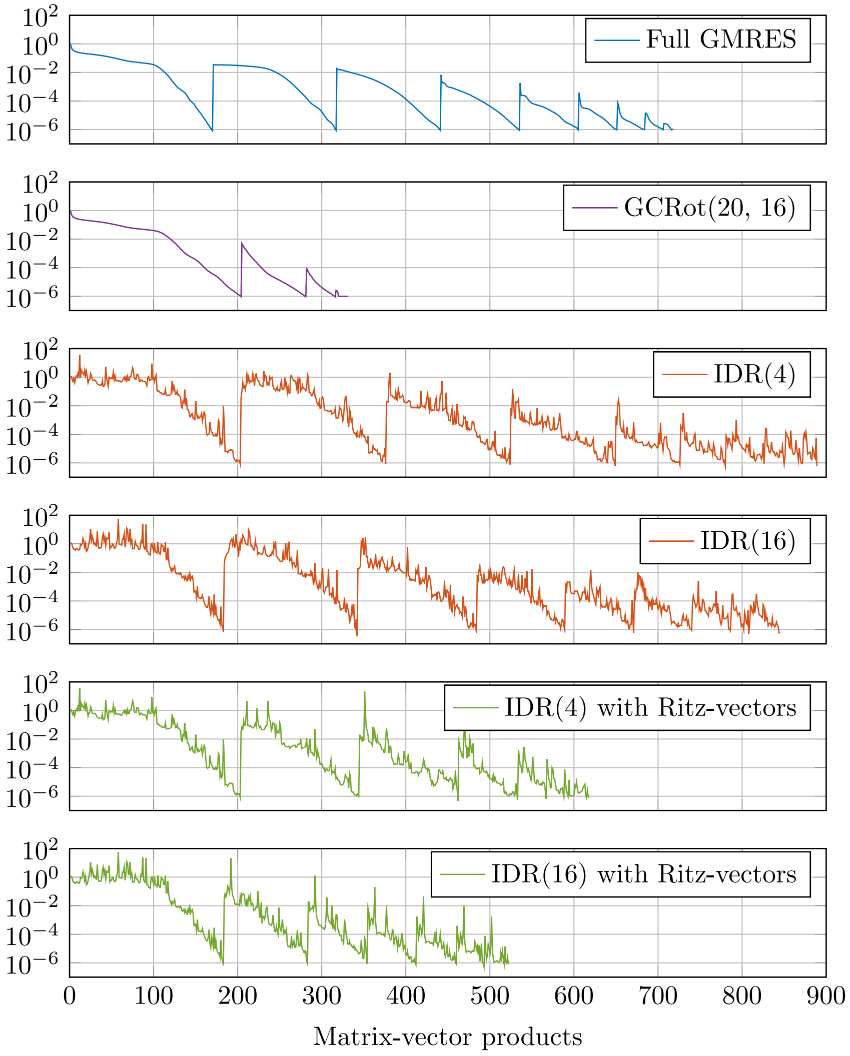

Figure 4.7: *(Example 4.7)*. Convergence residual history for the solution of (4.32) (diffusion-dominated example)

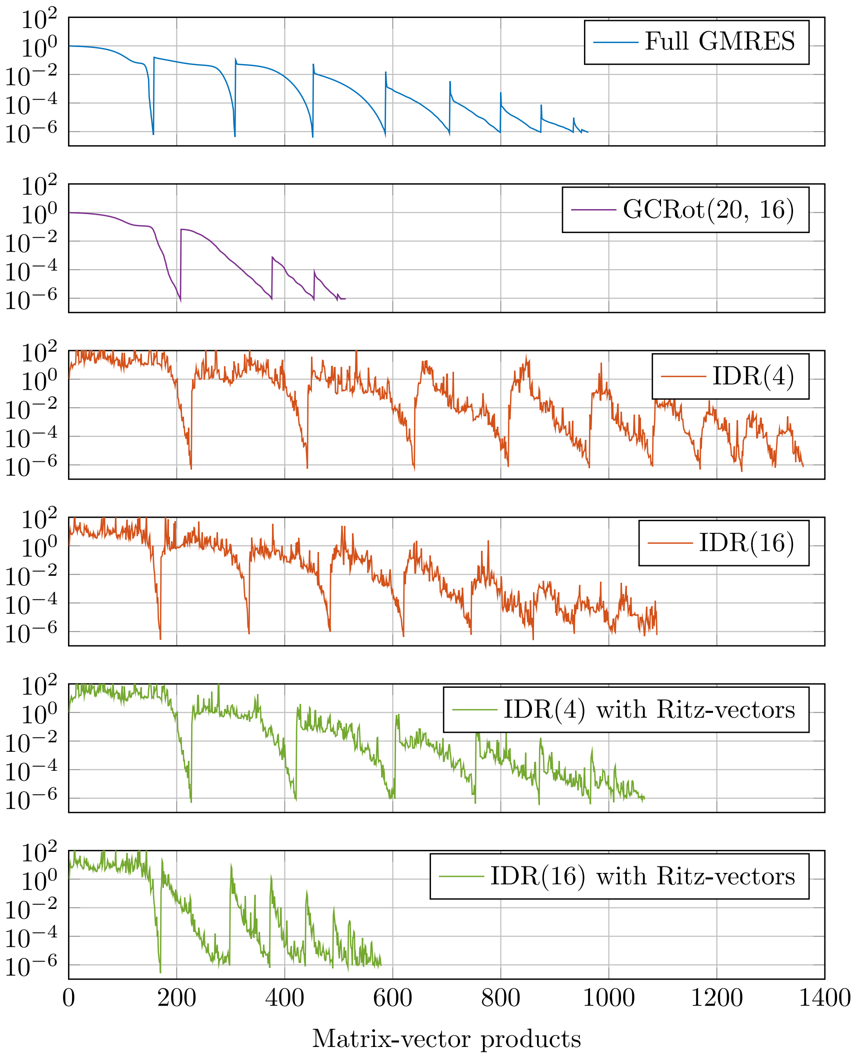

Figure 4.8: *(Example 4.7).* Convergence residual history for the solution of (4.32) (convection-dominated example)

Method	MATVECs	CPU time [s]
Full GMRES	718	185.93
GCRot(20, 4)	525	33.72
GCRot(20, 16)	332	51.2
BiCG [37]	1946	19.85
BiCGStab	1900	12.13
QMR [40]	1884	22.62
IDR(4) without recycling	889	20.34
IDR(4) with recycling	618	16.86
IDR(16) without recycling	845	36.61
IDR(16) with recycling	523	34.15

Table 4.1: *(Example 4.7).* Matrix-vector multiplications and time used for each method in the solution of (4.32) (diffusion dominated example)

Method	MATVECs	CPU time [s]
Full GMRES	962	281.43
GCRot(20, 4)	1380	96.01
GCRot(20, 16)	514	83.61
IDR(4) without recycling	1360	31.46
IDR(4) with recycling	1066	22.57
IDR(16) without recycling	1089	54.44
IDR(16) with recycling	578	37.58

Table 4.2: *(Example 4.7).* Matrix-vector multiplications and time used for each method in the solution of (4.32) (convection dominated example)

4.5 Discussion and remarks

In this chapter, we have derived a Hessenberg relation from the IDR(s) method while it solves a system of linear equations. This is a key component to obtain approximations to the eigenvalues and eigenvectors of the coefficient matrix involved. We have used this spectral information to accelerate the IDR(s) method.

In the first part of this chapter, we have proposed a Ritz-IDR(s) variant, named SC-Ritz-IDR(s), to solve systems of linear equations based on the work by Simoncini and Szyld [87]. This algorithm uses the inverse of the Ritz values as parameter ω_j for the creation of the residuals vectors into the subspaces $\mathcal{G}_j$. In contrast to Ritz-IDR(s), our proposed variant SC-Ritz-IDR(s) is a self-contained algorithm, i. e., it does not use an external sparse eigensolver to compute the Ritz values. In terms of CPU requirements and memory consumption, SC-Ritz-IDR(s) has a similar computational behavior as Ritz-IDR(s) [87]. Implementations of both methods Ritz-IDR(s) and SC-Ritz-IDR(s) may use complex arithmetic, even when the coefficient matrix and the right-hand side vectors are real, in the case of complex Ritz values as parameters ω_j.

In the second part of the chapter, we have explained how to enrich the search subspace of IDR(s) with the Ritz vectors. In particular, we have applied this enrichment to IDR(s) for solving sequences of systems of linear equations. After approximating the eigenvector during the solution of the first system of linear equations, IDR(s) uses this spectral information for the subsequent systems of equations. Numerical experiments show a significant reduction of the computational time.

CHAPTER 5

Induced Dimension Reduction method for solving linear matrix equations and preconditioners

This chapter discusses the solution of large-scale linear matrix equations using the Induced Dimension reduction method. IDR(s) was originally presented to solve system of linear equations, and is based on the IDR(s) theorem. We generalize the IDR(s) theorem to solve linear problems in any finite-dimensional space. This generalization allows us to develop IDR(s) algorithms to approximate the solution of linear matrix equations.

Additionally, we present two types of preconditioners to solve linear matrix equations. First, we propose a simple preconditioner to solve the Sylvester equation based on a fixed-point iteration. Second, we present a preconditioner for the reinterpretation of the multi-shift Helmholtz equation as a matrix equation. Several numerical examples are presented to illustrate the performance of IDR(s) for solving linear matrix equations.

This chapter is based on the articles:

R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction method for solving linear matrix equations. *Procedia Comput. Sci.*, 80:222–232, 2016 ,

M. Baumann, R. Astudillo, Y. Qiu, E. Y. M. Ang, M. B. van Gijzen, and R.-É. Plessix. An MSSS-preconditioned matrix equation approach for the time-harmonic elastic wave equation at multiple frequencies. *Comput. Geosci.*, 22(1):43–61, 2018 .

5.1 Introduction

In this chapter we extended the Induced Reduction Dimension method (IDR(s) [95]) to approximate the solution of linear matrix equations of the form

$$\sum_{j=1}^{k} A_j X B_j = C, \tag{5.1}$$

where the matrices $A_1, A_2, \dots, A_k$ are in $\mathbb{C}^{n\times n}$, B_1, B_2, $\dots, B_k$ are in $\mathbb{C}^{m\times m}$, $C \in \mathbb{C}^{n\times m}$, and $X \in \mathbb{C}^{n\times m}$ is the unknown matrix. Solving (5.1) is equivalent to solve a system of linear equations. Defining vec(X) as the vector of order $n \times m$ created by stacking the columns of the matrix X, we can write (5.1) as

$$\left(\sum_{j=1}^{k} B_k^T \otimes A_k\right) \mathrm{vec}(X) = \mathrm{vec}(C). \tag{5.2}$$

Throughout this chapter, we only consider the case where the coefficient matrix of the system of linear equations (5.2) is non-singular, i. e., (5.1) has guaranteed the existence and uniqueness of their solution. In (5.2) the conditions to ensure non-singularity of its coefficient matrix are not fully established. However in two specific cases as, the Sylvester and Lyapunov equation, the condition for existence and uniqueness of their solution are known. For the Sylvester equation

$$AX + XB = C, \tag{5.3}$$

the condition for the existence and uniqueness of the solution is that the matrices A and $-B$ do not have any common eigenvalue. The Lyapunov equation

$$AX + XA^T = C, \tag{5.4}$$

has a unique solution when the eigenvalues of A hold that $\lambda_i + \lambda_j \neq 0$ for $1 \leq i, j \leq n$ (see for example [43]).

Linear matrix equations of the form (5.1) appear in different areas like complex networks, and control and system theory (see [86] and references therein). Another important source of linear matrix equations is the numerical solution of differential equations. Discretization of differential equations lead to linear systems or parametrized linear systems, and in some cases, they can be rewritten as a Sylvester equations. In this chapter, we emphasize this kind of examples. We present numerical tests of Sylvester equations originated from the discretization of time-dependent linear systems, and convection-diffusion equations.

Methods to solve (5.1) have mostly focused on the solution of the particular cases of the Lyapunov and Sylvester equations. For small and dense matrices, one of the earliest algorithms to solve Sylvester equation (5.3) was proposed by Bartels and Stewart [11]. This algorithm relies on the computation of the Schur decomposition of the matrices A and B, and then solve a block upper triangular matrix. An improvement of the Bartels-Stewart method was introduced by Golub, Nash and Van Loan in [42] using the Hessenberg factorization of matrices A and B.

Solving large scale linear matrix equation is an active research area. The Alternating Direction Implicit (ADI) method, proposed by Peaceman and Rachford [70], has been adapted to solve the Sylvester equation by Ellner and Wachspress in [33]. The ADI method has been widely used for solving matrix equations, for example, Benner, Li, and Truhar extended the ADI method for low-rank requirements in [17]. However, the parameter selection in ADI is not trivial and its performance strongly depends on it (see for example [34], and [16]).

Krylov subspace methods have also been applied to solve matrix equations. Saad in [78] proposed to solve low-rank Lyapunov equation (5.4) using a projection over the Krylov subspace

$$\mathcal{K}_m(A, X) = \text{span}\{X,\, AX,\, \ldots,\, A^{m-1}X\}.$$

To improve the speed of these projection method, based of the work Druskin and Knizhnerman [26], Simoncini proposed the use of the extended Krylov subspace

$$\mathcal{E}\mathcal{K}_m(A, X) = \mathcal{K}_m(A, X) + \mathcal{K}_m(A^{-1}, A^{-1}X),$$

for solving the low-rank Lyapunov equation [85]. For the case of the Sylvester equation, the extended Krylov subspace projection method was used by Druskin and Simoncini in [27]. The main disadvantages of those methods is the use of the inverse of the matrices involved or its factorization, which might be prohibitive for large and unstructured matrices.

Another approach to solve linear matrix equations based on Krylov methods, is to consider the relation between a linear matrix equations and solving linear systems. Hochbruck and Starke applied the Quasi-Minimal Residual method (QMR) [40] to solve the system of linear equations obtained from the Lyapunov equation. A similar idea was proposed by Jbilou, Messaoudi, and Sadok in [48] to solve Lyapunov and Sylvester equations using block versions of the Full Orthogonalization method (FOM) [75] and the Generalized Minimal Residual method (GMRES) [81] called Global variants. This approach does not require the computation of any inverse or factorization of a matrix, however, it can suffer from slow convergence, in which case it is necessary to apply preconditioners. For a more detailed description of the state-of-art of matrix equation solvers see [86].

In this chapter, we propose a variant of Induced Dimension Reduction method (IDR(s)) for solving linear matrix equations. IDR(s) has been recently adapted to solve other related problems like solving block linear systems [28], multi-shift linear systems [13, 102], and eigenvalue problems [5, 45]. IDR(s) is based on the IDR(s) theorem. In this chapter, we generalize the IDR(s) theorem to solve linear problems in any finite-dimensional space. Using this generalization, we develop an IDR(s) algorithm to approximate the solution of linear matrix equations.

5.2 IDR(s) for linear operators

This section extends the IDR(s) method to solve linear matrix equations (5.1). We present an alternative form of the IDR(s) theorem. First, we would like to draw the attention of the reader to the proof of Theorem 3.1 in [95]. In this proof, the authors only use the properties of $\mathbb{C}^n$ as a linear subspace, and that A is a linear operator on this linear subspace. Using these facts, we can generalize the IDR(s) theorem to any finite-dimensional linear subspace $\mathcal{D}$ with $\mathcal{A}$ as linear operator defined on the same linear subspace. Corollary 2.1 summarizes this result.

Corollary 2.1. Let $\mathcal{A}$ be any linear operator over a finite dimensional subspace $\mathcal{D}$ and $\mathcal{I}$ the identity operator over the same subspace. Let $\mathcal{S}$ any (proper) subspace of $\mathcal{D}$. Define $\mathcal{G}_0 \equiv \mathcal{D}$, if $\mathcal{S}$ and $\mathcal{G}_0$ do not share a nontrivial invariant subspace of the operator $\mathcal{A}$, then the sequence of subspace $\mathcal{G}_j$, defined as

$$\mathcal{G}_j \equiv (\mathcal{I} - \omega_j \mathcal{A})(\mathcal{G}_{j-1} \cap \mathcal{S}) \quad j = 0, 1, 2 \ldots,$$

with ω_j's nonzero scalars, have the following properties,

1. $\mathcal{G}_{j+1} \subset \mathcal{G}_j$, for $j \geq 0$ and
2. dimension$(\mathcal{G}_{j+1}) <$ dimension$(\mathcal{G}_j)$ unless $\mathcal{G}_j = \{\mathbf{0}\}$.

Proof. The proof is analogous to the one presented in [95]. $\square$

Corollary 2.1 is closely related to another generalization of IDR(s) theorem presented by Du et al. in [28], which was used to derive an IDR(s) method for solving block system of linear equations. However, Corollary 2.1 has a broader scope. This corollary emphasizes that IDR(s) might be generalized to solve problems in any finite dimensional spaces. In particular, we apply this corollary to solve different types of linear matrix equations.

In a similar way as in [47], let us rewrite problem (5.1) as

$$\mathcal{A}(X) = C, \tag{5.5}$$

where $\mathcal{A}(X) = \sum_{j=1}^{k} A_j X B_j$. Using Corollary 2.1, we are able to create residuals $R_k = C - \mathcal{A}(X_k)$ of the problem (5.5) in the shrinking and nested subspaces $\mathcal{G}_j$ and obtain the approximations X_k. Only changing the definition of the operator $\mathcal{A}$ and the subspace $\mathcal{D}$, we are able to approximate the solution of the linear matrix equation using IDR(s) without forming the equivalent system of linear equation as sum of Kronecker products.

Throughout the rest of this chapter, the examples of IDR(s) for solving linear matrix equations are based on the biorthogonal residual variant presented in [103] (see Algorithm 15). The two main changes from the original IDR(s) are the substitution of the product $A\mathbf{x}$ by the application of the linear operator $\mathcal{A}(X)$ and the use of the Frobenius inner product. Algorithms 15 outlines the biorthogonal residual version of IDR(s) for solving linear matrix equations.

5.3 Preconditioning

The use of preconditioners in iterative methods is a key element to accelerate or improve the convergence. However, in the context of solving linear matrix equations $\mathcal{A}(X) = C$, there is not a straightforward definition of the application of a preconditioner. An option for applying the preconditioning operation to V is to obtain an approximation to the problem

$$\mathcal{A}(X) = V.$$

For example in the case of the Sylvester equation, the preconditioner applied to V computes an approximate solution of

$$AX + XB = V. \tag{5.6}$$

Similar to the preconditioners for solving linear systems, it is needed to approximate the solution of (5.6) in an computational cheap way. In next section, we present a simple preconditioner based on fixed-point iteration.

5.3.1 Fixed-point (FP) and Inexact Fixed-point (FP-ILU) preconditioners for the Sylvester equation

In this section we present a simple preconditioning scheme for the iterative method to solve the Sylvester equation

$$AX + XB = V. \tag{5.7}$$

Algorithm 15 Preconditioned IDR(s) for matrix equations with biorthogonal residuals

```
 1: procedure IDR($s$)
 2:     Input: $\mathcal{A}$ as linear matrix linear operator from $\mathbb{C}^{n\times b} \to \mathbb{C}^{n\times b}$, $C \in \mathbb{C}^{n\times b}$, $tol \in$
    (0, 1), $s \in \mathbb{N}^+$, $P \in \mathbb{C}^{n\times(s\times b)}$, $X \in \mathbb{C}^{n\times b}$, $\mathcal{M}$ as preconditioner    ▷ $X$ is initial guess
 3:     $G = 0 \in \mathbb{C}^{n\times(s\times b)}$, $U = 0 \in \mathbb{C}^{n\times(s\times b)}$    ▷ Initialization
 4:     $M = I_s \in \mathbb{C}^{s\times s}$, $\omega = 1$
 5:     $R = C - \mathcal{A}(X)$
 6:     while $\|R\|_F \le tol \times \|C\|_F$ do    ▷ Loop over$\mathcal{G}_j$ spaces
 7:         Compute $\mathbf{f} = [\langle R, P_i\rangle_F]_i$ for $i = 1, \ldots, s$
 8:         for $k = 1$ to $s$ do    ▷ Compute $s$ independent vectors $g_k$ in $\mathcal{G}_j$ space
 9:             Solve $\mathbf{c}$ from $M\mathbf{c} = \mathbf{f}$, $(\gamma_1, \ldots, \gamma_s)^H = \mathbf{c}$    ▷ Note that $M = P^H G$
10:             $V = R - \sum_{i=1}^{s} \gamma_i G_i$
11:             Applied preconditioner to $V$: $V = \mathcal{M}(V)$    ▷ Preconditioning operation
12:             $U_k = U\mathbf{c} + \omega V$
13:             $G_k = \mathcal{A}(U_k)$
14:             for $i = 1$ to $k - 1$ do    ▷ Make $\mathbf{g}_k$ orthogonal to $P$
15:                 $\alpha = \langle G_k, P_i\rangle_F / \mu_{i,i}$
16:                 $G_k = G_k - \alpha G_i$
17:                 $U_k = U_k - \alpha U_i$
18:             end for
19:             $\mu_{i,k} = \langle G_k, P_i\rangle_F$, $M_{i,k} = \mu_{i,k}$, for $i = k, \ldots, s$    ▷ Update $M$
20:             $\beta = \phi_k / \mu_{k,k}$    ▷ Make the residual orthogonal to $\mathbf{p}_i$ for $i = 1, \ldots, k$
21:             $R = R - \beta G_k$
22:             $X = X + \beta U_k$
23:             if $k + 1 \le s$ then
24:                 $\phi_i = 0$ for $i = 1, \ldots, k$
25:                 $\phi_i = \phi_i - \beta\mu_{i,k}$ for $i = k + 1, \ldots, s$
26:             end if
27:             Overwrite $k$th blocks of $G$ and $U$ by $\mathbf{G}_k$ and $\mathbf{U}_k$ respectively.
28:         end for    ▷ Entering $\mathcal{G}_{j+1}$
29:         $V = R$
30:         Applied preconditioner to $V$: $V = \mathcal{M}(V)$    ▷ Preconditioning operation
31:         $T = \mathcal{A}(V)$
32:         Select the parameter $\omega$ (see different options in [90, 103, 87])
33:         $R = R - \omega T$
34:         $X = X + \omega V$
35:     end while
36:     return $X$
37: end procedure
```

The solution of equation (5.7) is also the solution of the fixed-point iteration

$$AX_{k+1} = -X_k B + V \tag{5.8}$$

We propose as preconditioner a few steps of the fixed-point iteration (5.8). If matrix A is difficult to invert, we propose the application of few steps of the following iteration

$$M\hat{X}_{k+1} = -\hat{X}_k B + V, \tag{5.9}$$

where M is an approximation to the matrix A. This can be considered as an inexact fixed-point iteration. Particularly, we approximate A using the Incomplete LU factorization. Fixed-point iterations (5.8) and (5.9) do not have the same solution. However, if it is assumed that M is a good approximation of A and equation (5.7) is well-conditioned, one can expect that the solution of the fixed-point iteration (5.9) is close to the solution of the Sylvester equation (5.7).

We use as preconditioning operator $\mathcal{M}(V)$ the fixed-point iteration (5.9), or if it is possible to solve block linear system with A efficiently, we use iteration (5.8). The fixed-point iteration (5.8) for solving Sylvester equation has been analyzed in [60]. A sufficient condition for the iteration (5.8) to converge to its fixed-point is that $\|A^{-1}\|\|B\| < 1$ when A is non-singular. Using this result, it is easy to see that the inexact iteration (5.9) also converge to its fixed-point if M is non-singular and $\|M^{-1}\|\|B\| < 1$. For this reason, we can compute $M = LU + E$, the incomplete LU factorization of A, using strategies based on monitoring the growth of the norm of the inverse factors of L and U as the ones proposed in [19, 20], or scaling matrices such that $\|M^{-1}\|\|B\| < 1$ is satisfied. Another similar method based on the classical fixed-point iteration was presented in [83]. In particular, the authors in [83] solve the generalized Lyapunov equation combining the fixed point iteration with rank-truncation to improve the efficiency in large-scale settings.

Time-dependent linear system

In this section, we analyze the fixed-point iteration (5.8) for solving linear equations from the discretization of a time-dependent linear system

$$\frac{d\mathbf{y}}{dt} = A\mathbf{y} + g(t), \qquad t_0 \le t \le t_m \qquad \text{with } \mathbf{y}(t = t_0) = \mathbf{y}_0. \tag{5.10}$$

Solving (5.13) with backward Euler with constant time-step δ_t, we obtain a Sylvester equation

$$-AY + Y\frac{D}{\delta_t} = G + \frac{\mathbf{y}_0}{\delta_t}\mathbf{e}_1^T,$$

where $G = [g(t_1),\, g(t_2),\, \ldots,\, g(t_m)]_{n\times m}$, D is the bidiagonal and upper matrix

$$D = \begin{bmatrix} 1 & -1 & 0 & \ldots & 0 \\ 0 & 1 & -1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & & -1 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}_{m\times m},$$

and $\mathbf{e}_1$ represents the first canonical vector of order m.

Corollary 2.2. If the matrix $-A$ is stable, then the Sylvester equation (5.10) has a unique solution.

Proof. Because $-A$ is stable, all its eigenvalues of have negative real part, while D/δ_t has all its eigenvalues equal to $1/\delta_t$. The matrices $-A$ and D do not share any eigenvalue, as a consequence, Sylvester equation (5.13) has a unique solution. □

Due the fact that solving block linear systems with the matrix D is computationally inexpensive, we propose to use the following fixed-point iteration as preconditioner

$$\frac{D^T}{\delta_t} Y_{k+1}^T = Y_k^T A^T + G^T + \frac{\mathbf{e}_1}{\delta_t}\mathbf{y}_0^T. \tag{5.11}$$

Following corollary shows that there always exists a sufficiently small δ_t such that iteration (5.11) converges.

Corollary 2.3. If the matrix $-A$ is stable, there exists δ_t sufficiently small such that the iteration (5.11) applied to the equation (5.10) converges to its fixed-point.

Proof. A sufficient condition for the convergence of iteration is that $\|\delta_t D^{-1}\|\|A\| < 1$ for some norm $\|\star\|$. Using that for any matrix K and any scalar $\epsilon > 0$, there exist a norm $\|\star\|_{\alpha(\epsilon)}$, such that

$$\|K\|_{\alpha(\epsilon)} < \rho(K) + \epsilon,$$

where $\rho(K)$ is the spectral radius of the matrix K, we obtain

$$\|\delta_t D^{-1}\|_{\alpha(\epsilon)}\|A\|_{\alpha(\epsilon)} < (\rho(\delta_t D^{-1}) + \epsilon)\|A\|_{\alpha(\epsilon)}.$$

$\rho(\delta_t D^{-1}) = \delta_t$, then

$$\|\delta_t D^{-1}\|_{\alpha(\epsilon)}\|A\|_{\alpha(\epsilon)} < (\delta_t + \epsilon)\|A\|_{\alpha(\epsilon)}.$$

To ensure that $\|\delta_t D^{-1}\|\|A\| < 1$, for and sufficiently small ϵ, we can select δ_t such that

$$\delta_t \leq \frac{1 - \epsilon\|A\|_{\alpha(\epsilon)}}{\|A\|_{\alpha(\epsilon)}}.$$

In consequence, the iteration (5.11) converges to its fixed-point (the solution of (5.13)). □

5.4 Numerical examples

In this section we present five numerical experiments to illustrate the numerical behavior of IDR(s) for solving matrix equations and compare it with other block Krylov subspace solvers. The first three problems are illustrative small examples for different types of matrix equations. In the fourth and fifth examples, we consider more realistic applications.

The numerical experiments presented in this section were implemented in Python 3.5.3 running on GNU/ Debian Linux on an Intel computer with four cores I5 and 32GB of RAM. We use as stopping criterion

$$\frac{\|C - \mathcal{A}(X)\|_F}{\|C\|_F} \leq 10^{-8}. \tag{5.12}$$

5.4.1 Small examples

Experiment 5.1. (Solving a Lyapunov equation) In this example, we solve the Lyapunov equation using IDR(s) for $\mathcal{A}(X) = C$ with $\mathcal{A}(X) = AX + XA^T$. We compare IDR($s = 4$) for matrix equations with BiCGStab [101] and GMRES [81]. As matrix A, we choose the negative of the anti-stable matrix CDDE6 from the Harwell-Boeing collection, and matrix $C = \mathbf{c}\mathbf{c}^T$, with $\mathbf{c} = \text{rand}(961, 1)$. Although for IDR($s$) the solution of the Lyapunov equation takes more iteration (165), this is the faster method regarding CPU-time. IDR(s) consumes 7.52 secs., while GMRES runs in 17.53 secs. (131 iterations) and BiCGStab takes 13.32 secs. (566 iterations).

Experiment 5.2. (Solving a time-dependent linear system) We consider the time-dependent linear system,

$$\frac{d\mathbf{y}}{dt} = A\mathbf{y} + g(t), \qquad t_0 \leq t \leq t_m \qquad \text{with } \mathbf{y}(t = t_0) = \mathbf{y}_0. \tag{5.13}$$

Solving (5.13) with backward Euler with constant time-step δ_t, we obtain a Sylvester equation

$$-AY + Y\frac{D}{\delta_t} = G + \frac{\mathbf{y}_0}{\delta_t}\mathbf{e}_1^T,$$

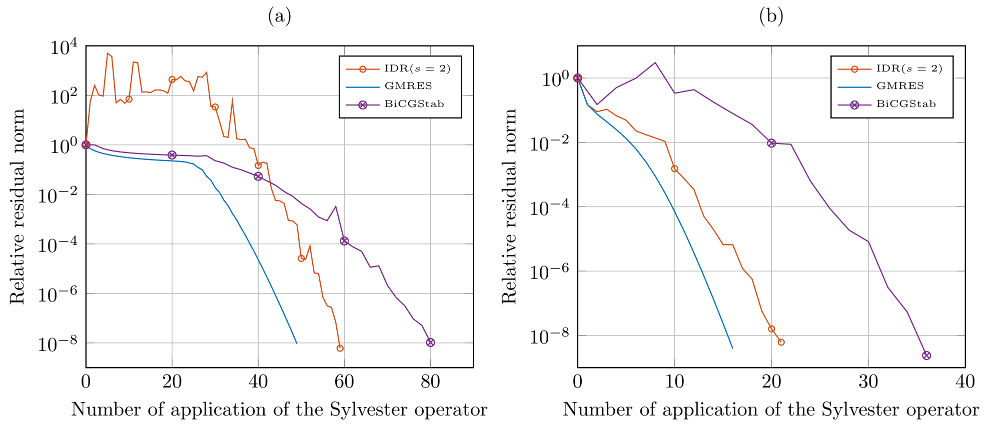

Figure 5.1: *Example 5.2.* (a) Residual norm for IDR($s = 2$), BiCGStab, and GMRES solving a Sylvester equation. (b) Residual norm for the preconditioned IDR($s = 2$), BiCGStab, and GMRES using two steps of (5.8).

where $G = [g(t_1), g(t_2), \ldots, g(t_m)]_{n\times m}$, D is the upper and bidiagonal matrix

$$D = \begin{bmatrix} 1 & -1 & 0 & \ldots & 0 \\ 0 & 1 & -1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & & -1 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}_{m\times m}, \tag{5.14}$$

and $\mathbf{e}_1$ represents the first canonical vector of order m. Specifically, We consider the 1D time-dependent convection-diffusion equation

$$\frac{du}{dt} - \epsilon\frac{d^2u}{dx^2} + \omega\frac{du}{dx} = 0, \qquad 0 \le t \le 1, \quad u_{t_0} = 1, \tag{5.15}$$

with convection parameter $\omega = 1.0$ and diffusion term $\epsilon = 10^{-3}$, $x \in [0, 100]$, with Dirichlet boundary conditions. We discretized this equation using the central finite differences and Euler backward for time integration, with $\delta_t = 0.05$ ($m = 20$), $\delta_x = 0.1$ ($A \in \mathbb{R}^{1000\times 1000}$). Figure 5.1 shows the evolution of the residual norm for IDR(s) and different Krylov method without and with preconditioner (5.8) respectively. We apply two steps of the preconditioner (5.8) inverting the matrix D defined as (5.14).

Experiment 5.3. (Solving a multi-shift linear system as a Sylvester equation) The multi-shift linear system of equation

$$(A - \sigma_i I)\mathbf{x}_i = \mathbf{b}_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

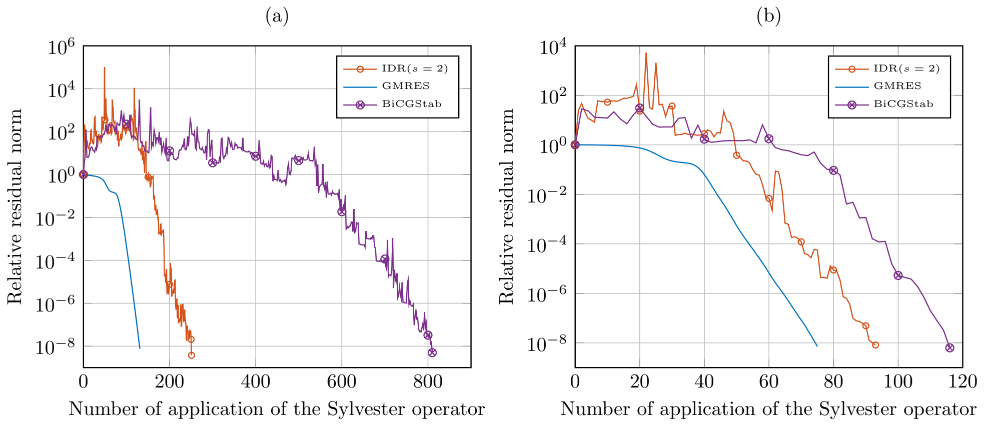

Figure 5.2: *Experiment 5.3.* (a) Residual norm for IDR(s=2), BiCGStab, and GMRES solving a Sylvester equation. (b) Residual norm for the preconditioned IDR(s=2), BiCGStab, and GMRES using two steps of (5.9).

can also be rewritten as a Sylvester equation

$$AX - XD = B,$$

where $D = diag([\sigma_1, \sigma_2, \ldots, \sigma_m])$, $X \in \mathbb{C}^{n\times m}$, and $B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m]^T \in \mathbb{C}^{n\times m}$. We consider the discretization of the convection-diffusion-reaction equation

$$-\epsilon \triangle u + \mathbf{v}^T \nabla u - r_i u = f,$$

with $\epsilon = 1$, $\mathbf{v} = [0, 250/\sqrt{5}, 500/\sqrt{5}]^T$, with $r_i \in \{0, 200, 400, 600, 800, 1000\}$, and homogeneous Dirichlet boundary conditions in the unit cube using central finite differences obtaining a matrix A of size 59319×59319. The right-hand side vector is defined by the solutions $u_i(x, y, z) = x(1-x)y(1-y)z(1-z)$ for each parameter r_i. Figures 5.2 shows the behavior of the relative residual norm for GMRES, IDR(s), and BiCGStab with and without preconditioner.

5.4.2 More realistic examples

The previous numerical examples are rather academic, in this section we present more realistic examples.

Experiment 5.4. (Solving a block system of linear equations) We consider a convection-diffusion problem from ocean circulation simulation (see [104]). The following model

$$-r \triangle \psi - \beta \frac{\partial \psi}{\partial x} = (\nabla \times F)_z \qquad \text{in } \Omega \tag{5.16}$$

describes the steady barotropic flow in a homogeneous ocean with constant depth. The function ψ represents the stream function, r is the bottom friction coefficient, β is the Coriolis parameter, and F is given by

$$F = \frac{\tau}{\rho H}, \tag{5.17}$$

where τ represents the external force field caused by the wind stress, H the average depth of the ocean, and ρ the water density. The stream function is constant on continent boundaries, i. e.,

$$\psi = C_k \qquad \text{on } \partial\Omega_k \quad \text{for } k = 1, 2, \ldots, K, \tag{5.18}$$

with K is the number of continents. The values of C_k are determined by the integral condition

$$\oint_{\partial\Omega_k} r \frac{\partial \psi}{\partial n}\, ds = -\oint_{\partial\Omega_k} F \times s\, ds. \tag{5.19}$$

We discretize (5.16)–(5.19) using the finite elements technique described in [104]. The physical parameters used can also be found in the same reference. We obtain a coefficient matrix A of order 42248, and we have to solve a sequence of twelve systems of linear equations

$$A\mathbf{x}_i = \mathbf{b}_i \qquad \text{with } i = 1, 2, \ldots, 12. \tag{5.20}$$

Each of these system of linear equations represent the data for each month of the year. We compare the time for solving (5.20) using two approaches, solving all the linear systems separately, and solving (5.20) as a linear matrix equation (a block linear system, i.e., all the right-hand side vectors and the same time). In all the cases, we applied incomplete LU of the matrix A with drop tolerance 10^{-4} as preconditioner. Table 5.1 shows the time comparison between the different methods using both approaches. Figure 5.3 shows the solution computed using IDR($s = 4$) for matrix equations.

In Table 5.2, the increment in number of matrix-block products is higher than a fact of two if the grid size is halved. This rather disappointing behavior seems to be caused by the dropping strategy in the ILU preconditioner, which gives a worse performance for increasingly finer grids. To exclude this effect, we next consider diagonal scaling as preconditioner for IDR(s) in Table 5.3.

Experiment 5.5. (Solving a multi-shift linear system as a generalized Sylvester equation) In this example, we are interested in the solution of numerical solution of the multi-shift problem that arises from the Helmholtz equation. Later in this section, we also propose a new preconditioner based on the ILU for this problem. Helmholtz equation

$$-\triangle u(\mathbf{x}) - \left(\frac{2\pi f}{c(\mathbf{x})}\right)^2 u(\mathbf{x}) = s(\mathbf{x}), \qquad \text{over } \Omega, \tag{5.21}$$

Method	Solving (5.20) separately		Solving (5.20) as a block linear system	
	Time [s]	# Mat-Vec	Time [s]	# Mat-Block
IDR($s = 4$)	17.03	2498	12.29	190
BiCGStab	22.66	3124	15.35	282
BiCG	25.58	4070	20.62	338
GMRES(100)	42.82	4200	38.30	400

Table 5.1: *Experiment 5.4.* Comparison of solving (5.20) as a sequence of linear system or as a matrix equation (a block linear system). This exemplifies one of the advantages of using a block-solvers over their sequential counterparts, the time reduction due to the extensive use of block subroutines (BLAS Level 3) over several calls to single vectors routines. Mat-Vec indicates the number of matrix-vector products and Mat-Block indicates the number of matrix-block multiplications.

Degree	Matrix size	Time [s]	# Mat-Block
4	2594×2594	0.02	5
3	4630×4630	0.11	18
2	10491×10491	0.42	29
1	42249×42249	12.29	190

Table 5.2: *Experiment 5.4.* Solving the ocean model problem as a matrix equation (a block linear system) using IDR($s = 4$) with ILU preconditioner. Degree is the grid size in the ocean model.

models physical phenomenon of wave propagation on the frequency domain. The function u is the pressure field, $\mathbf{x}$ is the spatial variable, $s(\mathbf{x})$ represents the source term, f is the angular frequency, and the function c is the medium velocity.

In different applications, it is necessary to resemble a infinite space domain. For this reason, it is necessary to impose suitable boundary conditions. Example of these type of boundary conditions are the absorbing boundary conditions or Sommerfeld boundary conditions given by

$$\frac{\partial u}{\partial n} - i\left(\frac{2\pi f}{c(\mathbf{x})}\right)^2 u(\mathbf{x}) = 0, \quad \text{on} \quad \partial\Omega. \tag{5.22}$$

Denoting $\sigma = 2\pi f$, the discretization of (5.21)–(5.22) leads to a system of linear equations of the form

$$(K + i\sigma C - \sigma^2 M)\mathbf{u} = \mathbf{s}, \tag{5.23}$$

where the vector $\mathbf{u}$ contains the unknown scalar, K is the Laplacian matrix, C represents the boundary conditions, M is the mass matrix, and $\mathbf{s}$ is load vector.

In practice, the wave-number σ makes the coefficient matrix of (5.23) an ill-conditioned and indefinite matrix. Also in different application as the ones

Degree	Matrix size	Time [s]	# Mat-Block
4	2594×2594	0.44	557
3	4630×4630	1.36	773
2	10491×10491	5.09	1204
1	42249×42249	58.03	2883

Table 5.3: *Experiment 5.4.* Solving the ocean model problem as a matrix equation (a block linear system) using IDR($s = 4$) with diagonal preconditioner. One can see a linear increment with a factor of two in the number of matrix-block products required if the grid size is halved.

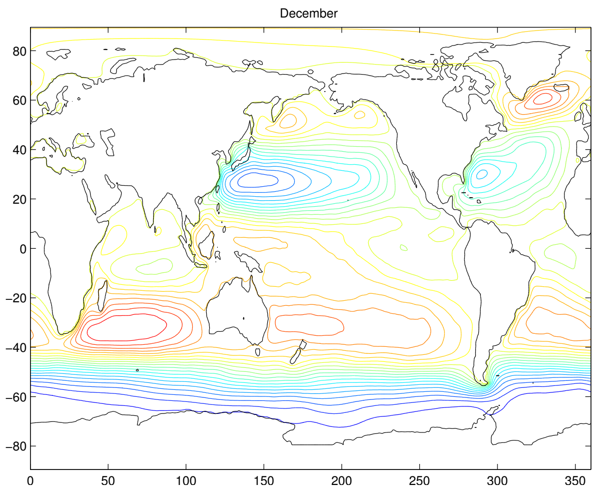

Figure 5.3: *Experiment 5.4.* Solution of the ocean problem.

described in [73], [64], and [82], it is necessary to solve the system of linear equations (5.23) using several values for the angular frequencies. This leads to the problem of solving a multi-shift system of linear equations

$$(K + i\sigma_j C - \sigma_j^2 M)\mathbf{u}_j = \mathbf{s}, \qquad \text{for } j = 1, 2, \ldots, m. \tag{5.24}$$

Defining the unknown block matrix as $X = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \in \mathbb{C}^{n\times m}$, we can reformulate (5.24) as a matrix equation

$$\mathcal{A}(X) = B, \tag{5.25}$$

where

$$\mathcal{A}(X) \equiv KX + iCX\Sigma - MX\Sigma^2, \tag{5.26}$$

$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_m)$, and $B = \mathbf{s}\,[1, \ldots, 1]_m$.

In this example, we are interested in the solution of (5.25). We consider *the wedge problem* introduced in [72]. This problem is an example of acoustic

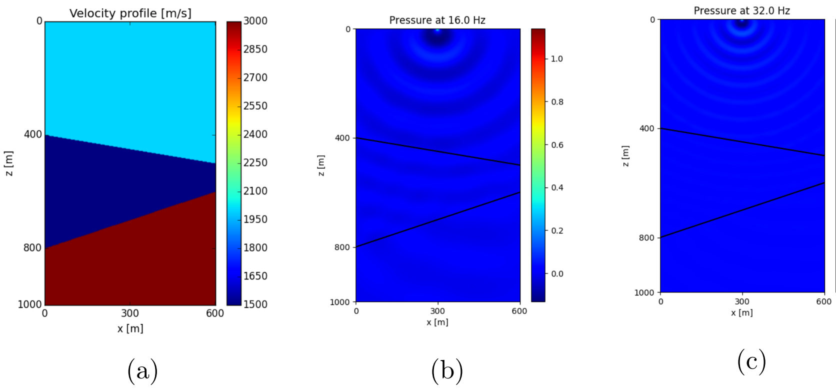

Figure 5.4: *(Example 5)* (a) Velocity distribution. (b) Solution at 16.0 Hz. (c) Solution at 32.0 Hz.

wave propagation modeled by the Helmholtz equation (5.21) with the boundary conditions (5.22). The computational domain Ω is $[0,\, 600]\times[0,\, 1000]$, and this is divided into three layers, each of them with a different sound velocity given by $c(\mathbf{x})$ (see Figure 5.4 (a)). We consider the Helmholtz model with damping proposed in [35],

$$-\triangle u(\mathbf{x}) - (1-\epsilon\, i)\sigma^2 u(\mathbf{x}) = s(\mathbf{x}), \qquad \text{over } \Omega, \tag{5.27}$$

where ϵ is the damping in the medium, and it is selected up to 5%, this mean $\epsilon \in [0,\, 0.05]$, and the source term is $s(x) = \delta(\mathbf{x} - \mathbf{x}_c)$ with $\mathbf{x}_c = (300,\, 0)^T$.

We discretize (5.27) using finite element method with B-splines of degree 1 as basis functions[†] (see [24] chapter 2). We use a uniform grid using $d_x = 2.0$ and $d_z = 2.0$ for spacing in $x-$ and $z-$ direction respectively. We apply IDR(6) to solve the matrix equations (5.26), and use as preconditioner operator

$$\mathcal{P}_\tau^{-1}(X) \equiv (K + i\tau C - \tau^2 M)^{-1} X, \tag{5.28}$$

for a given τ. The selection of this parameter τ is discussed in the next section. The solution of the block system of linear equations in (5.28) is computed via the LU factorization of the matrix $K + i\tau C - \tau^2 M$. Table 5.4 shows the CPU time and the number of iterations used by IDR(6) for solving the wedge problem, also it shows the final relative residual. One can see that the matrix equation approach obtains a large number of iterations when a wide range of frequencies is consider. Similar results are presented in [15] and [12] for the elastic wave equation.

[†]We use the open source FEM package nultis `http://nutils.org`.

Frequencies	Time [s]	Iterations	Residual
$\omega = [1.0,\ 1.5,\ 2.0]$	12.21	48	2.75×10^{-9}
$\omega = [1.0,\ 2,0,\ 4.0]$	29.45	115	4.29×10^{-9}
$\omega = [1.0,\ 4.0,\ 8.0]$	114.42	445	9.63×10^{-9}
$\omega = [1.0,\ 8.0,\ 16.0]$	1036.97	4028	8.45×10^{-9}
$\omega = [1.0,\ 16.0,\ 32.0]$	*	*	*

Table 5.4: Results for the problem $\mathcal{A}(\mathcal{P}^{-1}(X)) = B$: Time, number of iteration, and final residual norm for IDR(6) solving the problem (5.27) using (5.28) as preconditioner. The cpu time to create the preconditioner (LU factorization) is 5.88 secs. The symbol '*' means that the preconditioned IDR(6) method failed to satisfy (5.12) at the maximum of 5000 iterations.

Matrix equation formulation with ILU and spectral scaling strategy preconditioner

The matrix equation approach (5.25)–(5.26) is sensitive to the selection of a wide range the frequencies σ_i. This is to large extend due to the fact that a block Krylov solver builds a block Krylov subspace that contains the union of the spectra of each shifted system of linear equations involved. In this section, we review a recently proposed preconditioner to overcome this problem [14]. Later, we propose a new variant of this preconditioner that uses the incomplete LU factorization.

To start the review on the two-level preconditioner proposed in [14], we consider a simplify version of the operator (5.26), where the matrix C is the null matrix and the damping ϵ is zero. The two-level preconditioner for matrix equation is based on two main insights develop for multi-shift linear system of equations [14]. First, it uses a scaled version of the preconditioner operator (5.28), and the authors obtained the seed value τ that optimize convergence bound of full GMRES. This scaled version of the preconditioner (5.28) can be written as operator as

$$\mathcal{P}_{\tau}^{-1}(\mathcal{R}_1(X)),$$

where

$$\mathcal{R}_1(X) \equiv X \begin{bmatrix} 1/(1-\eta_1) & & \\ & \ddots & \\ & & 1/(1-\eta_m) \end{bmatrix}_{m \times m},$$

with $\eta_i = \sigma_i^2/(\sigma_i^2 - \tau)$. Second, the matrix associated with the operator

$\mathcal{A}(\mathcal{P}_\tau^{-1}(\mathcal{R}_1(X)))$ is

$$\begin{bmatrix} 1/(1-\eta_1)(K-\sigma_1^2 M)(K-\tau^2 M)^{-1} & & \\ & \ddots & \\ & & 1/(1-\eta_m)(K-\sigma_m^2 M)(K-\tau^2 M)^{-1} \end{bmatrix}_{(n\times m)\times(n\times m)}, \tag{5.29}$$

and its eigenvalues are located in regions bounded by circles. These circles are known explicitly using the information of the frequencies σ_i, the seed value τ, and the damping parameter of the problem (5.27). In more detail, for the systems of linear equations

$$\frac{1}{1-\eta_i}(K-\sigma_i^2 M)(K-\tau^2 M)^{-1}\mathbf{y}_i = \mathbf{b}, \qquad \text{for } i = 1,\ldots,m, \tag{5.30}$$

Lemma 4.1 in [14] states the following key results when the matrix K is symmetric positive semi-definite and M is symmetric positive definite

- (5.30) is equivalent to the preconditioner multi-shift system of linear equations

$$(K(K-\tau^2 M)^{-1} - \eta_i I)\mathbf{y}_i = \mathbf{b}, \qquad \text{for } i = 1,\ldots,m; \tag{5.31}$$

- The eigenvalues of the coefficient matrix of the kth system of linear equation of (5.30) are enclosed by the circle c_k of radii R_k defined as

$$c_k = \left(\frac{1}{2} - \frac{\sigma_k^2(\sigma_k^2 - \Re(\tau)}{(\sigma_k^2-\Re(\tau))^2+\Im(\tau)}, \frac{\Re(\tau)}{2\Im(\tau)} - \frac{\sigma_k^2\Im(\tau)}{(\sigma_k^2-\Re(\tau))^2+\Im(\tau)}\right),$$

$$R_k = \frac{1}{2}\sqrt{1+\left(\frac{\Re(\tau)}{\Im(\tau)}\right)^2};$$

- An optimal selection of the parameter τ for the preconditioner (5.28) is

$$\tau^* = \frac{2(\sigma_{\min}\sigma_{\max})^2}{\sigma_{\min}^2+\sigma_{\max}^2} - i\,\frac{\sqrt{|(\sigma_{\max}^2-\sigma_{\min}^2)^2|(\sigma_{\min}\sigma_{\max})^2}}{\sigma_{\min}^2+\sigma_{\max}^2}. \tag{5.32}$$

  This parameter τ^* minimizes the classical convergence bound for GMRES [79].

The two-level preconditioner is designed using the information of Lemma 4.1 in [14]. The seed parameter τ for (5.28) is selected as (5.32). Due the fact that the eigenvalues of each diagonal block of (5.29) are in the circle with center c_i and radius R_i, the authors in [14] proposed to apply a post-diagonal scaling to obtain a clustered spectrum. This clustered spectrum is more favorable for

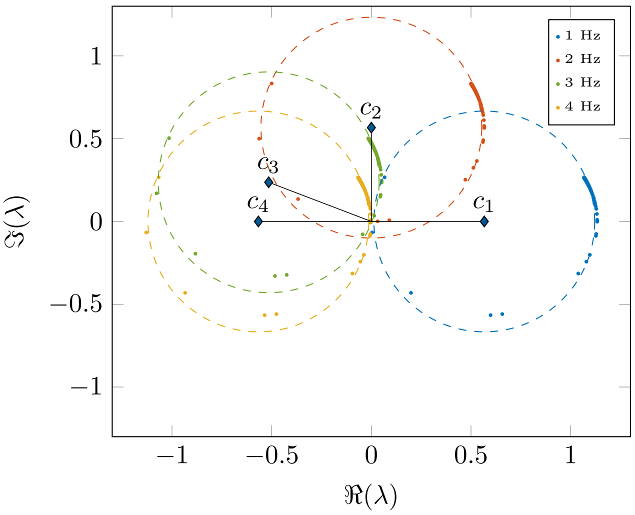

Figure 5.5: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\mathcal{P}^{-1}(\mathcal{R}_1(X)))$, and circles $(c_k,\, R_k)$.

the convergence of the Krylov iterative methods. Let us clarify this procedure with an example. Consider the Helmholtz equation (5.21), with Dirichlet boundary conditions, and frequencies $\{1.0,\, 2.0,\, 3.0, 4.0\}$. Figure 5.5 shows the eigenvalues of the matrix associated with $\mathcal{A}(\mathcal{P}^{-1}(\mathcal{R}_1(X)))$. Each cluster of eigenvalues corresponds to a diagonal block of the matrix (5.29), and these eigenvalues are located in circles described by c_k and R_k for $k = 1,\, 2,\, 3,\, 4$. This fact is exploited by defining the rotation operator $\mathcal{R}_2$ as

$$\mathcal{R}_2(X) \equiv X \begin{bmatrix} e^{-i\,(\theta_1-\theta_1)} & & \\ & \ddots & \\ & & e^{-i(\theta_m-\theta_1)} \end{bmatrix}_{m\times m},$$

where θ_k is the angular component of the point c_k. The application of the operator $\mathcal{R}_2$ over $\mathcal{A}(\mathcal{P}^{-1}(\mathcal{R}_1(X)))$ rotates the different cluster of eigenvalues onto the first circle $(c_1,\, R_1)$ (see Figure 5.6).

For our working example presented in the previous section, Table 5.5 shows the result of the solution of matrix equation (5.26) using the two-level preconditioner [14]. One can see the positive effect of the selection of τ as (5.32) and the application of the spectral rotation $\mathcal{R}_2$.

Nevertheless, if we substitute the operator (5.28) by its approximation using the incomplete LU factorization

$$\hat{\mathcal{P}}^{-1}(X) = \hat{U}^{-1}\hat{L}^{-1}X, \tag{5.33}$$

where $\hat{L}$ and $\hat{U}$ are the factors of the incomplete LU factorization of the matrix

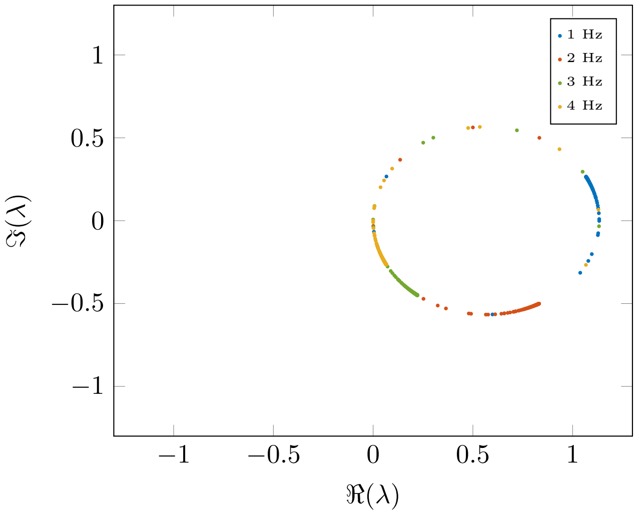

Figure 5.6: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\mathcal{P}^{-1}(\mathcal{R}_1(\mathcal{R}_2(X))))$.

Frequencies	Time	Iterations	Residual
$\omega = [1.0,\ 1.5,\ 2.0]$	8.10	32	7.87×10^{-9}
$\omega = [1.0,\ 2,0,\ 4.0]$	19.10	63	6.18×10^{-9}
$\omega = [1.0,\ 4.0,\ 8.0]$	43.06	168	5.82×10^{-9}
$\omega = [1.0,\ 8.0,\ 16.0]$	143.34	557	9.76×10^{-9}
$\omega = [1.0,\ 16.0,\ 32.0]$	435.23	1692	9.97×10^{-9}

Table 5.5: Results for the problem $\mathcal{A}(\mathcal{P}^{-1}(\mathcal{R}_1(\mathcal{R}_2(X)))) = B$: Time, number of iteration, and final residual norm for IDR(6) solving the problem (5.27) using the two-level preconditioner [14]. The CPU time to create the preconditioner (LU factorization and rotation matrix) is 6.08 secs.

Frequencies	Time	Iterations	Residual
$\omega = [1.0,\ 1.5,\ 2.0]$	66.93	385	9.83×10^{-9}
$\omega = [1.0,\ 2,0,\ 4.0]$	80.93	465	8.81×10^{-9}
$\omega = [1.0,\ 4.0,\ 8.0]$	241.33	1387	6.27×10^{-9}
$\omega = [1.0,\ 8.0,\ 16.0]$	*	*	*
$\omega = [1.0,\ 16.0,\ 32.0]$	*	*	*

Table 5.6: Results for the problem $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(X))) = B$: Time, number of iteration, and final residual norm for IDR(6) solving the problem (5.27) using (5.33) as preconditioner. The cpu time to create the preconditioner (ILU factorization) is 2.44 secs.

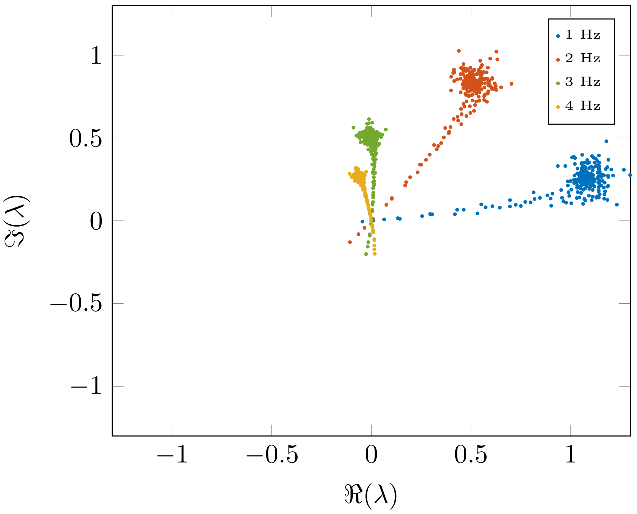

Figure 5.7: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(X)))$.

$(K + i\tau C - \tau^2 M)^{-1}$, the spectrum of the matrix associated with the operator $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(X)))$ is not bounded by the circles $(c_k,\ R_k)$ (see Figure 5.7). For this reason, we propose a different spectral rotation for the case of the use of the incomplete LU factorization.

First, let us define as the landmarks $\hat{c}_k$ as the set of approximations to the largest magnitude eigenvalues computed after p steps of the power method applied to of each system of linear equations (5.30). Figure 5.8 shows the axis generated by the points $\hat{c}_k$. We proposed the use of the angular component of the landmark points $\hat{c}_k$ as angles for the rotation of each cluster of eigenvalues.

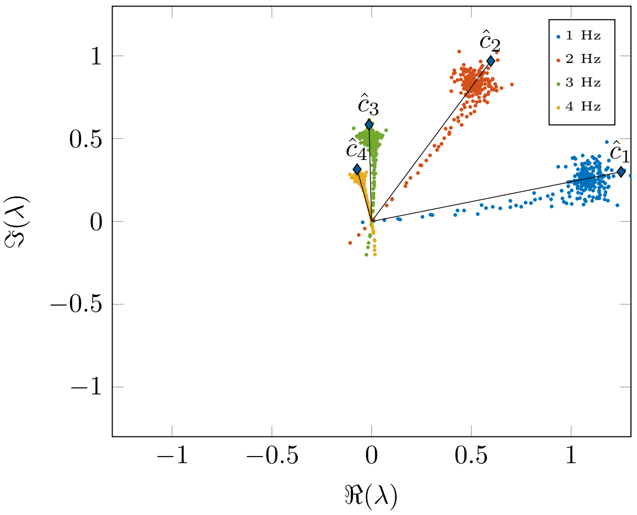

Figure 5.8: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(X)))$ and landmarks $\hat{c}_k$.

We define the new rotation operator $\hat{\mathcal{R}}_1(X)$ as

$$\hat{\mathcal{R}}_1(X) \equiv X \begin{bmatrix} e^{-i\hat{\theta}_1} & & \\ & \ddots & \\ & & e^{-i\hat{\theta}_m} \end{bmatrix}_{m\times m}, \tag{5.34}$$

with $\hat{\theta}_k$ the angular component of $\hat{c}_k$. Figure 5.9 shows the effect of the operator $\hat{\mathcal{R}}_1$. Algorithm 16 shows an implementation for computing the landmarks $\hat{c}_k$. Additionally, we apply the scaling $\hat{\mathcal{R}}_2$, defined as

$$\hat{\mathcal{R}}_2(X) \equiv X \begin{bmatrix} 1/|\hat{c}_1| & & \\ & \ddots & \\ & & 1/|\hat{c}_m| \end{bmatrix}_{m\times m}. \tag{5.35}$$

Figure 5.10 shows the effect of the scaling on the spectrum.

The application of the operator $\mathcal{R}_1$ correspond to the preconditioner [14], and it was apply in the Figures 5.7–5.9 only for illustrative purposes. To implement our proposed preconditioner, we only need to apply the rotation matrices $\hat{\mathcal{R}}_1$ and $\hat{\mathcal{R}}_2$. Algorithm 17 outline the creation of our proposed preconditioner. Table 5.7 shows how this new preconditioner improves the result over the plain application of the incomplete LU factorization (see Table 5.7).

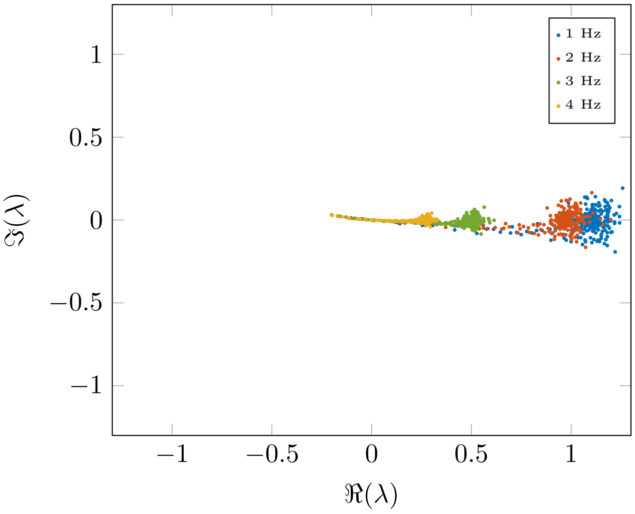

Figure 5.9: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(\hat{\mathcal{R}}_1(X))))$.

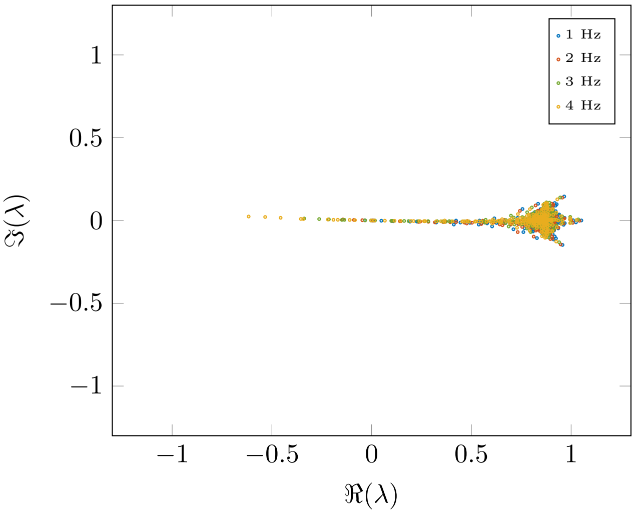

Figure 5.10: Eigenvalues of the matrix associated with the operator $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\mathcal{R}_1(\hat{\mathcal{R}}_1(\hat{\mathcal{R}}_2(X)))))$.

Algorithm 16 Block power method to compute the p-steps landmarks

```
1: Given $\mathcal{A}$ a linear operator defined as (5.26).
2: $X_0 = \text{randn}(n,m) + i \times \text{randn}(n,m)$
3: for $j = 1,\dots,p$ do
4:     $\hat{X}_j = \mathcal{A}(\hat{\mathcal{P}}_\tau^{-1}(X_{j-1}))$
5:     $X_j = \hat{X}_j / \|\hat{X}_j\|_F$
6: end for
7: $Y = \mathcal{A}(\mathcal{P}_\tau^{-1}(X_k))$
8: Compute $\hat{c}_j = \langle \mathbf{y}_j, \mathbf{x}_j \rangle / \langle \mathbf{x}_j, \mathbf{x}_j \rangle$ for $j = 1,\dots,m$
9: return $\hat{c}_1$, $\hat{c}_2$, $\dots$, $\hat{c}_m$
```

Algorithm 17 Creation of the ILU preconditioner with rotation and scaling

```
1: Given $\mathcal{A}$ a linear operator defined as (5.26).
2: Select a seed value $\tau$.                    ▷ we use $\tau = (1 - 0.5i)\sigma_{\min}$.
3: Compute the incomplete LU factorization of the matrix
       $$K + i\tau C - \tau^2.M$$
4: Compute the landmarks using $p$ steps of Algorithm 16.
5: Compute operator $\hat{\mathcal{R}}_1$ as (5.34).
6: Compute operator $\hat{\mathcal{R}}_2$ as (5.35).
```

Frequencies	Time	Iterations	Residual
$\omega = [1.0,\ 1.5,\ 2.0]$	18.05	104	8.59×10^{-9}
$\omega = [1.0,\ 2,0,\ 4.0]$	23.98	138	6.03×10^{-9}
$\omega = [1.0,\ 4.0,\ 8.0]$	46.92	270	7.63×10^{-9}
$\omega = [1.0,\ 8.0,\ 16.0]$	212.78	1222	8.29×10^{-9}
$\omega = [1.0,\ 16.0,\ 32.0]$	326.33	1875	9.31×10^{-9}

Table 5.7: Results for the problem $\mathcal{A}(\hat{\mathcal{P}}^{-1}(\hat{\mathcal{R}}_1(\hat{\mathcal{R}}_2(X)))) = B$: Time, number of iteration, and final residual norm for IDR(6) solving the problem (5.27) using our proposed preconditioner. The CPU time to create the preconditioner (ILU factorization, application of Algorithm 16 (20 steps), and rotation matrix) is 4.20 secs.

Multi-shift methods and matrix equation reformulation

For the problem (5.24), the most widely used methods are the Krylov multi-shift methods. These methods are based in a linearization of (5.24) as following,

$$\left(\begin{bmatrix} iC & K \\ I & 0 \end{bmatrix} - \sigma_j \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}\right) \begin{bmatrix} \sigma_j \mathbf{u}_j \\ \mathbf{u}_j \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}. \tag{5.36}$$

Under the assumption that the mass matrix M is non-singular, (5.36) can be rewritten as

$$(A - \sigma_j I)\mathbf{x} = \mathbf{b}. \tag{5.37}$$

The Krylov subspace methods can be adapted to solve this problem efficiently by exploiting the shift invariant property of the Krylov subspaces, i. e.,

$$\mathcal{K}_m(A, \mathbf{x}) = \mathcal{K}_m(A - \alpha I, \mathbf{x}), \quad \text{for } \alpha \in \mathbb{C}. \tag{5.38}$$

This shift invariant property enables to create efficient solvers for the problem (5.37). The main idea is to create only one Krylov subspace basis and used it for solving the m systems of linear equations in (5.38). Some examples of these multi-shift methods are [41], [84], and [102].

However it is not straightforward to apply a preconditioner and exploit the multi-shift invariant property of the Krylov subspaces at the same time. For this reason, K. Meerbergen proposed in [59] the use of the shifted A as preconditioner using the fact that

$$(A - \sigma_j I)(A - \tau I)^{-1} = A(A - \tau I)^{-1} - \frac{\sigma_j}{\sigma_j - \tau} I, \quad \text{for } \tau \in \mathbb{C}. \tag{5.39}$$

In [15], Baumann and van Gijzen present a comparison between the multi-shift method and the matrix equation reformulation for the elastic wave equation. From the conclusions presented in that work, it is worth remarking that solving the matrix presents a worse computational performance with respect the multi-shift Krylov method with inner-outer preconditioner presented in [13]. However, the matrix equation reformulation has different kind of advantages. First, it allows the use of inexact preconditioners as the one presented in the previous section. The inexact preconditioners are important for the solving 3D examples where the memory is a computational bottleneck (see for example 3 in [15]). Also, using the matrix equation reformulation can be useful for solving the multi-shift Helmholtz equation with several right-hand size vectors or source points (see [12]). Spectral deflation and augmentation might be applicable using the matrix equation framework (see [31]).

5.5 Discussion and remarks

In this chapter we have presented a generalization of the IDR(s) theorem [95] valid for any finite-dimensional space. Using this generalization, we have presented a framework of IDR(s) for solving linear matrix equations. This document also presents several numerical examples of IDR(s) solving linear matrix equations, among them, the most common linear matrix equations like Lyapunov and Sylvester equation. In the examples solving Lyapunov and Sylvester equations, full GMRES required less iterations to converge than IDR and BiCGStab. However, IDR(s) presented a better performance in CPU time.

Additionally, we have introduced two preconditioners based on fixed-point iteration to solve the Sylvester equation. The first preconditioner is a fixed-point iteration of the form

$$AX_{k+1} = -X_k B + C,$$

that required the explicit inverse or the solving block linear systems with the matrix A. Whenever it is not possible the inversion or solving block linear system with the matrix A in an efficient way, we use the inexact iteration

$$MX_{k+1} = -X_k B + C,$$

where $M = LU$ the incomplete LU factorization of A.

For the case of solving the Helmholtz equation, we have proposed a variant of the two-level preconditioner [14] for the matrix equation reformulation. This variant uses the incomplete LU factorization, which is suitable for large scale problems when is expensive to compute the exact LU factorization. The proposed preconditioner uses the power method to obtain a rotation which gives a more clustered eigenvalues favorable for the convergence of the Krylov method.

CHAPTER 6

Conclusions and future work

The main goal of this work have been the development of new algorithms for solving matrix problems based on the Induced Dimension Reduction method. The Induced Dimension Reduction is a Krylov method to solve system linear of equations. Its rather unusual approach of enforcing the residuals vectors into the shrinking subspaces $\mathcal{G}_j$, instead of the classical approach of finding the approximation in a dimensional increasing subspaces, makes IDR(s) an efficient option among the short-recurrences Krylov methods.

Following, we summarize the principal conclusions of this research in more detail.

In chapter 2, we have presented an overview of the development and mathematics behind the Induced Dimension Reduction methods. We have compared the computational behavior of IDR(s) with respect to other short-recurrences Krylov methods as restarted GMRES, BiCG, and BiCGStab, for solving systems of linear equations that arises from the simple model of the convection-diffusion equation. We have analyzed the causes why IDR(s) for $s > 1$ outperforms BiCGStab for the system of linear equations obtained from the discretization of the convection-diffusion equation particularly in the convection dominated case.

In chapter 3, we have deduced a Hessenberg decomposition

$$AW_m = W_mH_m + \mathbf{w}_{m+1}\mathbf{e}_m^T.$$

From the calculation of the IDR(s) to create vectors in the $\mathcal{G}_j$ subspaces. This allows to use IDR(s) to approximate a subset of eigenpairs (λ_i, $\mathbf{x}_i$) of the matrix A, i. e.,

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i.$$

We have implemented the implicit restarting technique, which gives a competitive result with the Implicitly Restarted Arnoldi method.

Once we have established how to obtain approximations to the eigenvalues and eigenvectors from the IDR(s) method, we have used this information to accelerate systems of linear equations and sequences of systems of linear equations, respectively, in chapter 4. As one of the principal findings of this chapter, we can remark the development of a self-contained version the linear solver proposed by Simoncini and Szyld the Ritz-IDR method. Both the Ritz-IDR(s) method and our proposed version SC-Ritz-IDR(s) use the Ritz values as input parameter. However, the SC-Ritz-IDR(s) method does not need an external call to an eigensolver routine to compute the Ritz values required. Another important finding is the deduction of an initial subspace recycling technique for IDR(s). We have use this procedure to add Ritz vectors to the initial search space of IDR(s) during the solution of sequences of systems of linear equations.

In chapter 5, we have generalized the IDR(s) theorem for block linear systems, more specifically ones obtained from linear matrix equations. We have considered multi-shift Helmholtz equation as a matrix equation. At first sight this reformulation does not provide too many advantages over the established multi-shift method. However in chapter 5, we have seen how with this reformulation a more flexible framework for preconditioning is achieved; for example, the use of incomplete LU factorization (other works in this direction are [96] and [12]).

Through several examples conducted in this thesis, we have shown the advantage of the IDR(s) as a short-recurrences method for solving different types of non-symmetric matrix problems. For example, we can see the IDR(s) process for eigenvalues proposed in chapter 2 as an intermediate option between the Arnoldi method and the Lanczos method, in terms of number of computer requirements and convergence for solving non-symmetric eigenvalues problems. Another example is the computational behavior of IDR(s) for solving linear systems with respect to other solvers as GMRES and BiCGStab. This makes IDR(s) worthy to be considered as method of preference for large scale problems.

Further research

Due the flexibility offered by IDR(s) and the diversity of problems where we have applied it, there exist several possibilities for further research about this method.

The freedom of choosing any linear combination of vectors in $\mathcal{G}_j$ to generate a new vector in the same subspace implies that the Hessenberg decomposition presented in chapter 3 is not unique and it can be generalized. Specifically using (3.9), we can obtain a family of IDR(s) decompositions as

$$AW_mU_m = W_m(H_m + R_m) + \mathbf{w}_{m+1}\mathbf{e}_m^T, \tag{6.1}$$

where $W_m \in \mathbb{C}^{n\times m}$ is a basis for the Krylov subspace and every block of $s+1$ columns belongs to a $\mathcal{G}_j$ subspace, H_m is an $(s+2)$-banded and upper Hessenberg matrix, U_m is an $(s+1)$-banded upper triangular matrix, and the matrix R_m is defined as $R_m = diag(R_{11}, \ldots, R_{jj})$, where $R_{ii} \in \mathbb{C}^{(s+1)\times(s+1)}$ are upper triangular matrices. At this point, we have obtained a family of pencil eigenvalue problems $(H_m + R_m, U_m)$ that can be used to approximate the eigenvalues of the matrix A (see Figure 6.1). The elements of the matrix R_m might be chosen to improve the Ritz values obtained (see also [107]).

In chapter 4, the following aspects deserve further investigation. The Ritz values and Ritz vectors were used to accelerate the IDR(s), however, we did not take into account the accuracy of these Ritz eigenpairs. This might be important to generalize the ideas proposed in this chapter for solving sequences of system of linear equations where the coefficient matrix is not fixed.

A potential improvement for the block IDR(s) version presented in chapter 5 is the application of low-rank techniques to address high-dimensional problems. A convenient low-rank technique can be the application of tensors in Hierarchical Tucker Format (see for example [52] and [53]).

Another topic which offers opportunity for further development is the preconditioner for the multi-shift Helmholtz matrix equation presented in chapter 5. During the creation of that preconditioner, we might incorporate information of the eigenvalues and eigenvectors computed by the IDR(s) iteration itself (chapter 4 this work). Also, it is worth mentioning that the reinterpretation of the multi-shift Helmholtz problem as a matrix equation has the advantage of not forcing the co-linearity as the multi-shift Krylov methods (see [41], [84], and [13]). For this reason, other techniques as incomplete factorizations (this work) deflation and augmentation can be used in the matrix equation approach (see for example [31]).

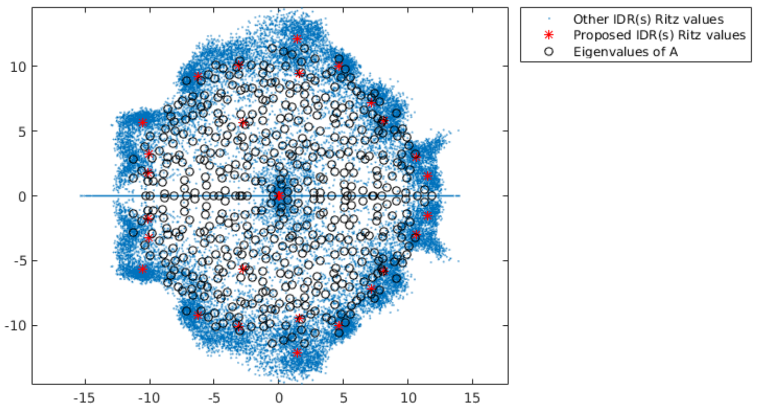

Figure 6.1: Eigenvalues and Ritz values obtained after 30 steps of IDR(6) for a random sparse matrix A of dimension 500. The circles represent the eigenvalues of the matrix A, the red stars represent the Ritz values obtained for our proposed IDR(s) method (chapter 3), and the blue dots represents other Ritz values generated by IDR(s) where the matrix R_m is chosen randomly with norm 1. One can see that these Ritz values generate clusters. The matrix R_m might be used to obtain better approximation of the Ritz values. This can be also studied from the point of view of the pseudospectra of the matrix H_m (see [100] as a reference).

APPENDIX A

Avoiding breakdown in the IDR(s) iteration

In this note we are interested in the study of the breakdown behavior of IDR(s). In [95], the authors classify the breakdown of IDR(s) into two different classes. *Breakdown of type 1* is when it is not possible to create a new residual vector in the subspace $\mathcal{G}_j$. This might happen when the coefficient matrix of the $s \times s$ system of linear equations, that IDR(s) solves during its iterations, is (nearly) singular. *Breakdown of type 2* is when the parameters ω_j are selected too close to zero, then IDR(s) suffers from stagnation.

IDR(s) with biorthogonal residuals [103] solves in large part these two types of breakdowns. The breakdown of type 2 is solved using the "maintaining the convergence" strategy [90] for the selection of the parameters ω_j. For the case of the breakdown of type 1, this IDR(s) variant uses more stable residuals recurrences that in most of cases generate a non-singular coefficient matrix for the inner $s \times s$ system of linear equations. Nevertheless, even IDR(s) with biorthogonal residuals might suffer from the breakdown of type 1 in rare circumstances. First, we analyze the causes of the breakdown of type 1 in biortho IDR(s). Then, we present a computational strategy, that allows IDR(s) to continue its iteration in case of breakdown. This strategy does not rely on restarting the IDR(s) method. On the contrary, it tries to recover the computational work already performed for the upcoming iterations.

A.1 Changing the shadow space during the IDR(s) iteration

Let us explain in more detail the causes of the breakdown type 1. First, we recall the recurrences formulas to create the intermediate residuals $\{\mathbf{r}_k^{(j+1)}\}_{k=1}^s$ in the subspace $\mathcal{G}_{j+1}$ for the biortho IDR(s). We use the notation introduced in chapter 4. The residual is

$$\mathbf{r}_k^{(j+1)} = \mathbf{r}_{k-1}^{(j+1)} - \beta_k^{(j+1)}\mathbf{g}_k^{(j+1)}, \quad \text{for } k = 1,\, 2,\, \ldots,\, s, \tag{A.1}$$

where the scalar $\beta_k^{(j+1)}$ is selected such that

$$\langle \mathbf{r}_k^{(j+1)}, \mathbf{p}_k \rangle = 0. \tag{A.2}$$

The direction vectors are defined as

$$\mathbf{u}_k^{(j+1)} = \hat{\mathbf{u}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)}\mathbf{u}_i^{(j+1)},$$

and

$$\mathbf{g}_k^{(j+1)} = \hat{\mathbf{g}}_k^{(j+1)} - \sum_{i=1}^{k-1} \alpha_i^{(j+1)}\mathbf{g}_i^{(j+1)}. \tag{A.3}$$

The scalars $\{\alpha_i^{(j+1)}\}_{i=1}^{k-1}$ are selected, such that,

$$\langle \mathbf{g}_k^{(j+1)}, \mathbf{p}_i \rangle = 0 \quad \text{for } i = 1, \ldots, k-1, \tag{A.4}$$

and the vectors $\hat{\mathbf{g}}_k^{(j+1)}$ and $\hat{\mathbf{u}}_k^{(j+1)}$ are

$$\hat{\mathbf{g}}_k^{(j+1)} = A\hat{\mathbf{u}}_k^{(j+1)},$$

$$\hat{\mathbf{u}}_k^{(j+1)} = \omega_{j+1}\left(\mathbf{r}_{k-1}^{(j+1)} - \sum_{i=s-k}^{s} \gamma_i^{(j+1)}\mathbf{g}_i^{(j)}\right) + \sum_{i=s-k}^{s} \gamma_i^{(j+1)}\mathbf{u}_i^{(j)},$$

where the scalars $\{\gamma_i^{(j+1)}\}_{i=k}^s$ are selected as

$$\left\langle \mathbf{r}_{k-1}^{(j+1)} - \sum_{i=k}^{s} \gamma_i^{(j+1)}\mathbf{g}_i^{(j)}, \mathbf{p}_\ell \right\rangle = 0 \qquad \text{for } \ell = k,\, k+1,\, \ldots, s. \tag{A.5}$$

Because the condition (A.4), the system of linear equations generated by (A.5) has the following structure

$$\begin{bmatrix} \mu_{k,k} & 0 & \cdots & \cdots & 0 \\ \mu_{k+1,k} & \mu_{k+1,k+1} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ \mu_{s,k} & \mu_{s,k+1} & \cdots & \cdots & \mu_{s,s} \end{bmatrix} \begin{bmatrix} \gamma_k \\ \gamma_{k+1} \\ \vdots \\ \vdots \\ \gamma_s \end{bmatrix} = \begin{bmatrix} \phi_k \\ \phi_{k+1} \\ \vdots \\ \vdots \\ \phi_s \end{bmatrix}, \tag{A.6}$$

where $\mu_{\ell,i} = \langle \mathbf{g}_i, \mathbf{p}_\ell \rangle$, and $\phi_\ell = \langle \mathbf{r}_{k-1}^{(j+1)}, \mathbf{p}_\ell \rangle$ for $k \leq i$, $\ell \leq s$.

Now, we are ready to enunciate the causes of the breakdown of type 1 in the following lemmas.

Lemma 3. *In biortho IDR(s), if the kth direction vector $\mathbf{g}_k^{(j+1)}$ in $\mathcal{G}_{j+1}$ defined in* (A.3) *is orthogonal to the kth vector of the shadow space $\mathbf{p}_k$, and the $(k-1)$th intermediate residual vector in $\mathbf{r}_{k-1}^{(j+1)}$ is not orthogonal to $\mathbf{p}_k$, then a breakdown of type 1 occurs.*

Proof. Given that the kth direction vector $\mathbf{g}_k^{(j+1)}$ is orthogonal to the kth vector of the shadow space $\mathbf{p}_k$, then the entry $\mu_{k,k}$ of the coefficient matrix defined in (A.6) is zero. Then, this matrix is singular. Using the fact that the $\mathbf{r}_{k-1}^{(j+1)}$ is not orthogonal to $\mathbf{p}_k$, then the entry ϕ_k is different from zero. In conclusion, the system of linear equation (A.6) does not have a solution. □

Lemma 4. *In biortho IDR(s), if the $(k-1)$th intermediate residual vector in $\mathcal{G}_{j+1}$, defined in* (A.1)*, is orthogonal to the kth vector of the shadow space $\mathbf{p}_k$, then a breakdown of type 1 occurs.*

Proof. In (A.1), the parameter $\beta_k^{(j+1)}$ is set by imposing the condition (A.2). Due to the fact that the vector $\mathbf{r}_{k-1}^{(j+1)}$ is orthogonal to $\mathbf{p}_k$, the scalar $\beta_k^{(j+1)}$ is zero. Then, IDR(s) makes no progress. □

Corollary 4.1. In biortho IDR(s), the first vector of the shadow space $\mathbf{p}_1$ should be selected to be not orthogonal to the initial residual vector $\mathbf{r}_0$.

Proof. This is a direct consequence of Lemma 4. This is equivalent to the condition imposed on the shadow residual $\hat{\mathbf{r}}_0$ in BiCGStab (see Algorithm 3 line 2). □

As we can see from the Lemmas above, the breakdown of type 1 occurs when the vectors in $\mathcal{G}_{j+1}$ are already orthogonal to the shadow space. The general idea to avoid this breakdown is to change a vector in the shadow space $\mathcal{S}$. More specifically, if we define $\hat{\mathcal{S}}$ as the orthogonal complement of the set

$$\mathcal{P} \equiv span\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{k-1}, \mathbf{p}_{k+1}, \ldots, \mathbf{p}_s\},$$

we can exploit the fact that $\hat{\mathcal{G}}_{j+1} \subset \mathcal{G}_j$, where $\hat{\mathcal{G}}_{j+1} \equiv (I - \omega_{j+1}A)(\mathcal{G}_j \cap \hat{\mathcal{S}})$. By inserting a new vector $\mathbf{p}_k$ in the set $\mathcal{P}$, we do not change the property $\hat{\mathcal{G}}_{j+1} \subset \mathcal{G}_j$. This simply adds an additional constrain to the vectors in the new subspaces $\hat{\mathcal{G}}_{j+1}$, $\hat{\mathcal{G}}_{j+2}, \ldots$ and so on. Algorithm 18 shows the steps to select a new $k-$th vector in the shadow space, and update the other vectors needed to

Algorithm 18 Strategy for recovering from breakdown type 1

```
1: Select a new random vector $\mathbf{p}_k$ in $\mathbb{C}^n$.
2: Set
       $\mathbf{u}_i^{(1)} = \mathbf{u}_i^{(j+1)}$ and $\mathbf{g}_i^{(1)} = \mathbf{g}_i^{(j+1)}$ for $i = 1, \ldots, k-1$,
       $\mathbf{u}_i^{(0)} = \mathbf{u}_i^{(j)}$ and $\mathbf{g}_i^{(0)} = \mathbf{g}_i^{(j)}$ for $i = k, \ldots, s$
   and,
3: Update
       $\mu_{k,i} = \langle \mathbf{g}_i^0, \mathbf{p}_k \rangle$ for $i = 1, \ldots, k$.
       $\phi_k = \langle \mathbf{r}_{k-1}^{(1)}, \mathbf{p}_k \rangle$.
4: Restart the current IDR($s$) iteration.
```

continue the IDR(s) iteration.

Algorithm 18 should be used as breakdown recovery routine in biortho IDR(s) [103] (Algorithm 7 in Chapter 2). This routine should be called inside Algorithm 7 , when the scalar β is (close to) zero (line 22), or in line 21 of the same Algorithm, if the entry $M_{k,k}$ is close to zero.

A.2 Numerical experiment

We consider the following example proposed in [108] by J.-P. Zemke. The coefficient matrix is

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 1 & -1 & -3 & -2 & 0 \\ 1 & 0 & -1 & 1 & 1 & -2 & 1 & 5 & 4 & 0 \\ 0 & 1 & 2 & -1 & 0 & 1 & 0 & -2 & -2 & 1 \\ 0 & 0 & 1 & 0 & -1 & 2 & -1 & -5 & -4 & 0 \\ 0 & 0 & 0 & 1 & 2 & -2 & 1 & 5 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

and the right-hand size vector $\mathbf{b} = \mathbf{e}_1$.

We solve the problem

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

using biortho IDR($s = 2$) taking as initial guess $\mathbf{x}_0 = \mathbf{0}$. To illustrate the influence of the shadow space over the breakdown of type 1 and the use of the Algorithm 18, we run IDR(s) selecting the following shadow spaces

$$P_0 = \text{randn}(10, 2),$$

$$P_1^H = \begin{bmatrix} 0 & -1 & -1 & 1 & 0 & -1 & 0 & 2 & 2 & 0 \\ 1 & 1 & 1 & -1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix},$$

and,

$$P_2^H = \begin{bmatrix} 1 & 1 & 1 & -1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & -1 & -1 & 1 & 0 & -1 & 0 & 2 & 2 & 0 \end{bmatrix}.$$

In case of breakdown, we call the recovery procedure presented as Algorithm 18. Figure A.1 (a) shows the convergence behavior selecting P_0 as shadow space. In this case no breakdown occurs. For P_1, we obtain a breakdown at the first iteration (see Figure A.1 (b)). The breakdown is produced because the initial residual vector is orthogonal to the first vector in the shadow space P_1, this is, $\langle \mathbf{r}_0, \mathbf{p}_1 \rangle = 0$ (see Corollary 4.1). Algorithm 18 is called and ($\mathbf{p}_1$) is substituted. Similar case for the shadow space P_2, IDR(s) creates a second intermediate residual that is orthogonal to the second vector of P_2 which according to Lemma 4 produces a breakdown at the second iteration (see Figure A.1 (c)).

A.3 Conclusions

We have studied the causes of the breakdown type 1 in biortho IDR(s). The main contribution of this note has been the development of a breakdown recovery routine, that allows to continue the IDR(s) iteration in case of this type of breakdown occurs.

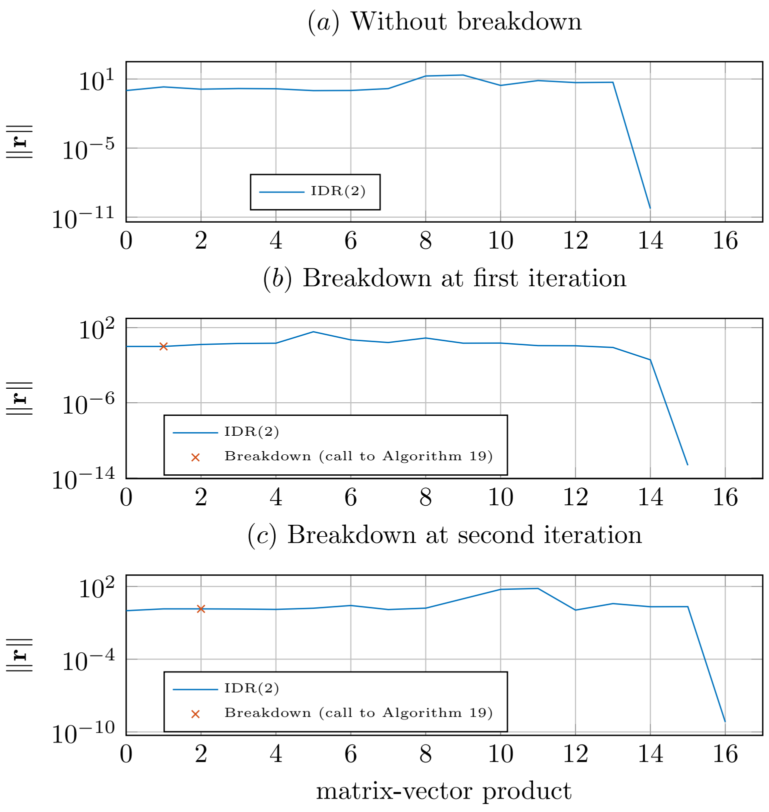

Figure A.1: (a) IDR($s = 2$) using P_0 as shadow space. (b) IDR($s = 2$) using P_1 as shadow space.(c) IDR($s = 2$) using P_2 as shadow space.

Publications associated with this research

Articles published

1. R. Astudillo and M. B. van Gijzen. An Induced Dimension Reduction algorithm to approximate eigenpairs of large nonsymmetric matrices. In *11th International Conference of Numerical Analysis and Applied Mathematics 2013*, volume 1558, AIP Publishing, 2013, pages 2277–2280.

2. R. Astudillo and M. B. van Gijzen. A restarted Induced Dimension Reduction method to approximate eigenpairs of large unsymmetric matrices. *J. Comput. Appl. Math.*, 296:24–35, 2016.

3. R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction method for solving linear matrix equations. *Procedia Comput. Sci.*, 80:222–232, 2016.

4. R. Astudillo and M. B. van Gijzen. The Induced Dimension Reduction method applied to convection-diffusion-reaction problems. In *Numerical Mathematics and Advanced Applications ENUMATH 2015*, B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, and Ö. Uğur, editors, Cham, ZG, Switzerland, 2016, pages 295–303. Springer.

5. R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction Method to solve the quadratic eigenvalue problem. In *Numerical Analysis and its Applications: 6th International Conference, NAA 2016, Lozenetz, Bulgaria, June 15-22, 2016, Revised Selected Papers*, I. Dimov, L. Vulkov, and I. Faragó, editors, Cham, ZG, Switzerland, 2017, pages 203–211. Springer.

Part of the material presented in chapter 5 is a contribution to the article,

6 M. Baumann, R. Astudillo, Y. Qiu, E. Y. M. Ang, M. B. van Gijzen, and R.-É. Plessix. An MSSS-preconditioned matrix equation approach for

the time-harmonic elastic wave equation at multiple frequencies. *Comput. Geosci.*, 22(1):43–61, 2018.

Articles submitted

1. R. Astudillo, J. M. de Gier, and M. B. van Gijzen. Accelerating the Induced Dimension Reduction method using spectral information. Technical Report 17-04, Delft Unversity of Technology, Delft Institute of Applied Mathematics, Delft, ZH, The Netherlands, 2017.

Posters

1. *A restarted Induced Dimension Reduction method to approximate eigenpairs of large nonsymmetric matrices.*Poster in 10th International Workshop on Accurate Solution of Eigenvalue Problems, Dubrovnik, Croatia June 2 5, 2014. (with M. B. van Gijzen).
2. *Induced Dimension Reduction method for solving linear matrix equations.* The Fortieth Woudschoten Conference, Zeist, The Netherlands, 7-9 October, 2015. (with M. B. van Gijzen).

Part of the material in chapter 5 was presented as the poster,

3 *A set of Fortran 90 and Python Routines for solving Linear equations with IDR(s).* SIAM Conference on Applied Linear Algebra, Atlanta, United States of America, 26-30 October, 2015. (with M. Baumann (Presenter) and M. B. van Gijzen).

Curriculum Vitae of the author

Reinaldo Antonio Astudillo Rengifo was born on September 5th, 1981, in Caracas. Venezuela. He graduated as Computer Scientist (BSc.) in 2005 at the Universidad Central de Venezuela (UCV). During 2005 and 2012, Reinaldo worked as a lecturer at the Department of Computer Science UCV, teaching courses on Numerical Analysis, Numerical Linear Algebra, and Discrete Mathematics at undergraduate level. In 2011, he obtained his Master degree (MSc.) in Computer Science from the UCV. In 2012, he joined the Numerical Analysis group of Delft University of Technology in the Netherlands as Ph.D. candidate under the supervision of Dr. Ir. Martin. B. van Gijzen.

Reinaldo published journal papers and conference proceedings as a result of his Ph.D. research (see publication section). Also, he had the opportunity of presenting his work at different conferences as, the 11th International Conference of Numerical Analysis and Applied Mathematics in Rodos (Greece), and the 10th International Workshop on Accurate Solution of Eigenvalue Problems in Dubrovnik (Croatia), among others. Besides his academic activities, Reinaldo co-founded the SIAM Student Chapter of Delft University in 2014.

Reinaldo's research interests are numerical linear algebra in general, with focus on Krylov subspace methods, spectra and pseudospectra computations, solution of large and sparse systems linear equations, and scientific programming.

Publications

1. R. Astudillo and Z. Castillo. Computing pseudospectra using block implicitly restarted Arnoldi iteration. *Math. Comput. Modelling*, 57 (9–10):2149–2157, 2013.

2. R. Astudillo and Z. Castillo. Approximating the weighted pseudospectra of large matrices. *Math. Comput. Modelling*, 57(9–10):2169–2176, 2013.

3. R. Astudillo and M. B. van Gijzen. An Induced Dimension Reduction algorithm to approximate eigenpairs of large nonsymmetric matrices. In *11th International Conference of Numerical Analysis and Applied Mathematics 2013*, volume 1558, AIP Publishing, 2013, pages 2277–2280.

4. R. Astudillo and M. B. van Gijzen. A restarted Induced Dimension Reduction method to approximate eigenpairs of large unsymmetric matrices. *J. Comput. Appl. Math.*, 296:24–35, 2016.

5. R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction method for solving linear matrix equations. *Procedia Comput. Sci.*, 80:222–232, 2016.

6. R. Astudillo and M. B. van Gijzen. The Induced Dimension Reduction method applied to convection-diffusion-reaction problems. In *Numerical Mathematics and Advanced Applications ENUMATH 2015*, B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, and Ö. Uğur, editors, Cham, ZG, Switzerland, 2016, pages 295–303. Springer.

7. R. Astudillo and M. B. van Gijzen. Induced Dimension Reduction Method to solve the quadratic eigenvalue problem. In *Numerical Analysis and its Applications: 6th International Conference, NAA 2016, Lozenetz, Bulgaria, June 15-22, 2016, Revised Selected Papers*, I. Dimov, L. Vulkov, and I. Faragó, editors, Cham, ZG, Switzerland, 2017, pages 203–211. Springer.

8. R. Astudillo, J. M. de Gier, and M. B. van Gijzen. Accelerating the Induced Dimension Reduction method using spectral information. Technical Report 17-04, Delft Unversity of Technology, Delft Institute of Applied Mathematics, Delft, ZH, The Netherlands, 2017.

9. M. Baumann, R. Astudillo, Y. Qiu, E. Y. M. Ang, M. B. van Gijzen, and R.-É. Plessix. An MSSS-preconditioned matrix equation approach for the time-harmonic elastic wave equation at multiple frequencies. *Comput. Geosci.*, 22(1):43–61, 2018.

Bibliography

[1] E. Agullo, L. Giraud, and Y.-F. Jing. Block GMRES method with inexact breakdowns and deflated restarting. *SIAM J. Matrix Anal. Appl.*, 35(4):1625–1651, 2014.

[2] K. Ahuja. *Recycling Krylov subspaces and preconditioners.* PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2011.

[3] H. Anzt, M. Kreutzer, E. Ponce, G. D. Peterson, G. Wellein, and J. Dongarra. Optimization and performance evaluation of the IDR iterative Krylov solver on GPUs. *Int. J. High Perform. Comput. Appl.*, 32(2): 220–230, 2018.

[4] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[5] R. Astudillo and M. B. van Gijzen. A restarted Induced Dimension Reduction method to approximate eigenpairs of large unsymmetric matrices. *J. Comput. Appl. Math.*, 296:24–35, 2016.

[6] R. Astudillo and M. B. van Gijzen. The Induced Dimension Reduction method applied to convection-diffusion-reaction problems. In *Numerical Mathematics and Advanced Applications ENUMATH 2015*, B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, and Ö. Uğur, editors, Cham, ZG, Switzerland, 2016, pages 295–303. Springer.

[7] J. Baglama, D. Calvetti, and L. Reichel. Fast Leja points. *Electron. Trans. Numer. Anal.*, 7:124–140, 1998.

[8] Z. Bai, D. Day, J. Demmel, and J. Dongarra. A test matrix collection for non-Hermitian eigenvalue problems (release 1.0). Technical Report CS-97-355, University of Tennessee, Department of Computer Science, Knoxville, TN, USA, 1997.

[9] Z. Bai, D. Hu, and L. Reichel. A Newton basis GMRES implementation. *IMA J. Numer. Anal.*, 14(4):563–581, 1994.

[10] Z. Bai and Y. Su. SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 26 (3):640–659, 2005.

[11] R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Commun. ACM*, 15(9):820–826, 1972.

[12] M. Baumann, R. Astudillo, Y. Qiu, E. Y. M. Ang, M. B. van Gijzen, and R.-É. Plessix. An MSSS-preconditioned matrix equation approach for the time-harmonic elastic wave equation at multiple frequencies. *Comput. Geosci.*, 22(1):43–61, 2018.

[13] M. Baumann and M. B. van Gijzen. Nested Krylov methods for shifted linear systems. *SIAM J. Sci. Comput.*, 37(5):S90–S112, 2015.

[14] M. Baumann and M. B. van Gijzen. An efficient two-level preconditioner for multi-frequency wave propagation problems. Technical Report 17-03, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, ZH, The Netherlands, 2017.

[15] M. Baumann and M. B. van Gijzen. Efficient iterative methods for multi-frequency wave propagation problems: A comparison study. *Procedia Comput. Sci.*, 108:645–654, 2017.

[16] P. Benner, P. Kürschner, and J. Saak. Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method. *Numer. Algorithms*, 62(2):225–251, 2013.

[17] P. Benner, R.-C. Li, and N. Truhar. On the ADI method for Sylvester equations. *J. Comput. Appl. Math.*, 233(4):1035–1045, 2009.

[18] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. Dongarra. Matrix Market: A web resource for test matrix collections. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement*, R. F. Boisvert, editor, London, UK, 1997, pages 125–137. Chapman & Hall Ltd.

[19] M. Bollhöfer. A robust ILU with pivoting based on monitoring the growth of the inverse factors. *Linear Algebra Appl.*, 338(1–3):201–218, 2001.

[20] M. Bollhöfer. A robust and efficient ILU that incorporates the growth of the inverse triangular factors. *SIAM J. Sci. Comput.*, 25(1):86–103, 2003.

[21] Z. Bujanović and Z. Drmač. A new framework for implicit restarting of the Krylov–Schur algorithm. *Numer. Linear Algebra Appl.*, 22(2): 220–232, 2015.

[22] D. Calvetti, L. Reichel, and D. C. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electron. Trans. Numer. Anal.*, 2:1–21, 1994.

[23] T. P. Collignon and M. B. van Gijzen. Minimizing synchronization in IDR(s). *Numer. Linear Algebra Appl.*, 18(5):805–825, 2011.

[24] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, Ltd, Chichester, WSX, UK, 2009. ISBN 9780470748732.

[25] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM J. Numer. Anal.*, 36(3):864–889, 1999.

[26] V. Druskin and L. Knizhnerman. Extended Krylov subspaces: Approximation of the matrix square root and related functions. *SIAM J. Math. Anal. Appl.*, 19(3):755–771, 1998.

[27] V. Druskin and V. Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Syst. Contr. Lett.*, 60(8):546–560, 2011.

[28] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S.-L. Zhang. A block IDR(s) method for nonsymmetric linear systems with multiple right-hand sides. *J. Comput. Appl. Math.*, 235(14):4095–4106, 2011.

[29] L. Du, T. Sogabe, and S.-L. Zhang. A variant of the IDR(s) method with the quasi-minimal residual strategy. *J. Comput. Appl. Math.*, 236 (5):621–630, 2010.

[30] L. Du, T. Sogabe, and S.-L. Zhang. IDR(s) for solving shifted nonsymmetric linear systems. *J. Comput. Appl. Math.*, 274:35–43, 2015.

[31] G. Ebadi, N. Alipour, and C. Vuik. Deflated and augmented global Krylov subspace methods for the matrix equations. *Appl. Numer. Math.*, 99:137–150, 2016.

[32] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20(2):345–357, 1983.

[33] N. S. Ellner and E. L. Wachspress. New ADI model problem applications. In *ACM '86 Proceedings of 1986 ACM Fall joint Computer Conference*, S. Winkler and H. S. Stone, editors, Los Alamitos, CA, USA, 1986, pages 528–534. IEEE Computer Society Press.

[34] N. S. Ellner and E. L. Wachspress. Alternating direction implicit iteration for systems with complex spectra. *SIAM J. Numer. Anal.*, 28(3):859–870, 1991.

[35] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM J. Sci. Comput.*, 27(4):1471–1492, 2006.

[36] A. Essai. Weighted FOM and GMRES for solving nonsymmetric linear systems. *Numer. Algorithms*, 18(3–4):277–292, 1998.

[37] R. Fletcher. Conjugate gradient methods for indefinite systems. In *Proceedings of the Dundee Conference on Numerical Analysis*, G. A. Watson, editor, Berlin, Heidelberg, New York, 1976, pages 73–89. Springer-Verlag.

[38] J. G. F. Francis. The QR transformation a unitary analogue to the LR transformation—part 1. *Comput. J.*, 4(3):265–271, 1961.

[39] J. G. F. Francis. The QR transformation—part 2. *Comput. J.*, 4(4): 332–345, 1962.

[40] R. W. Freund and N. M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60(1):315–339, 1991.

[41] A. Frommer and U. Glässner. Restarted GMRES for shifted linear systems. *SIAM J. Sci. Comput.*, 19(1):15–26, 1998.

[42] G. H. Golub, S. Nash, and C. F. van Loan. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, 24(6): 909–913, 1979.

[43] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. 4th edition, The Johns Hopkins University Press, Baltimore, MD, USA, 2012. ISBN 9781421407944.

[44] M. H. Gutknecht. IDR explained. *Electron. Trans. Numer. Anal.*, 36: 126–148, 2010.

[45] M. H. Gutknecht and J.-P. M. Zemke. Eigenvalue computations based on IDR. *SIAM J. Matrix Anal. Appl.*, 34(2):283–311, 2013.

[46] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49(6):409–436, 1952.

[47] M. Hochbruck and G. Starke. Preconditioned Krylov subspace methods for Lyapunov matrix equations. *SIAM J. Math. Anal. Appl.*, 16(1):156–171, 1995.

[48] K. Jbilou, A. Messaoudi, and H. Sadok. Global FOM and GMRES algorithms for matrix equations. *Appl. Numer. Math.*, 31(1):49–63, 1999.

[49] Y.-F. Jing, T.-Z. Huang, Y. Duan, and B. Carpentieri. A comparative study of iterative solutions to linear systems arising in quantum mechanics. *J. Comput. Phys.*, 229(11):8511–8520, 2010.

[50] W. D. Joubert and G. F. Carey. Parallelizable restarted iterative methods for nonsymmetric linear systems. part I: Theory. *Intern. J. Computer. Math.*, 44(1–4):243–267, 1992.

[51] W. D. Joubert and G. F. Carey. Parallelizable restarted iterative methods for nonsymmetric linear systems. part II: Parallel implementation. *Intern. J. Computer. Math.*, 44(1–4):269–290, 1992.

[52] D. Kressner and C. Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM J. Matrix Anal. Appl.*, 31(4):1688–1714., 2010.

[53] D. Kressner and C. Tobler. Algorithm 941: htucker—A matlab toolbox for tensors in hierarchical Tucker format. *ACM Trans. Math. Software*, 40(3):22:1–22:22, 2014.

[54] P. Lancaster. Explicit solutions of linear matrix equations. *SIAM Rev.*, 12(4):544–566, 1970.

[55] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.*, 45(4):255–281, 1950.

[56] R. B. Lehoucq. Implicitly restarted Arnoldi methods and subspace iteration. *SIAM J. Matrix Anal. Appl.*, 23(2):551–562, 2001.

[57] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4):789–821, 1996.

[58] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. ISBN 978-0-89871-407-4.

[59] K. Meerbergen. The solution of parametrized symmetric linear systems. *SIAM J. Matrix Anal. Appl.*, 24(4):1038–1059, 2003.

[60] M. Monsalve. Block linear method for large scale Sylvester equations. *Comput. Appl. Math.*, 27(1):47–59, 2008.

[61] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16(4):1154–1171, 1995.

[62] R. B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Math. Comp.*, 65(215):1213–1230, 1996.

[63] R. B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Anal. Appl.*, 21(4): 1112–1135, 2000.

[64] W. A. Mulder and R. E. Plessix. How to choose a subset of frequencies in frequency-domain finite-difference migration. *Geophys. J. Int.*, 158(3): 801–812, 2004.

[65] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix Anal. Appl.*, 13(3):778–795, 1992.

[66] M. P. Neuenhofen. M(s)stab(ℓ): A generalization of IDR(s)stab(ℓ) for sequences of linear systems. arXiv:1604.06043, 2016.

[67] H. V. Nguyen. *Krylov methods for solving a sequence of large systems of linear equations.* PhD thesis, Baylor University, Waco, TX, USA, 2015.

[68] M. L. Park, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[69] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44(169):105–124, 1985.

[70] D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *J. Soc. Indust. Appl. Math.*, 5(1):28–41, 1955.

[71] B. Philippe and L. Reichel. On the generation of Krylov subspace bases. *Appl. Numer. Math.*, 62(9):1171–1186, 2012.

[72] R. E. Plessix and W. A. Mulder. Separation-of-variables as a preconditioner for an iterative Helmholtz solver. *Appl. Numer. Math.*, 44(1): 385–400, 2003.

[73] R. G. Pratt. Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model. *Geophys.*, 64(3): 888–901, 1999.

[74] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.*, 34:269–295, 1980.

[75] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comp.*, 37(155):105–126, 1981.

[76] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 42(166):567–588, 1984.

[77] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm.*, 53(1–3):71–90, 1989.

[78] Y. Saad. Numerical solution of large Lyapunov equations. In *Signal Processing, Scattering, Operator Theory, and Numerical Methods. Proceedings of the International Symposium MTNS-89, vol III*, M. A. Kaashoek, J. H. van Schuppen, and A. S. M. Ran, editors, Boston, MA, USA, 1990, pages 503–511. Birkhäuser Verlag.

[79] Y. Saad. *Iterative Methods for Sparse Linear Systems.* 2nd edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003. ISBN 978-0-89871-534-7.

[80] Y. Saad. *Numerical Methods for Large Eigenvalue Problems: Revised edition.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011. ISBN 978-1-611970-72-2.

[81] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[82] A. K. Saibaba, T. Bakhos, and P. K. Kitanidis. A flexible Krylov solver for shifted systems with application to oscillatory hydraulic tomography. *SIAM J. Sci. Comput.*, 35(6):A3001–A3023, 2013.

[83] S. D. Shank, V. Simoncini, and D. B. Szyld. Efficient low-rank solution of generalized Lyapunov equations. *Numer. Math.*, 134(2):327–342, 2016.

[84] V. Simoncini. Restarted full orthogonalization method for shifted linear systems. *BIT Numer. Math.*, 43(2):459–466, 2003.

[85] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29(3):1268–1288, 2007.

[86] V. Simoncini. Computational methods for linear matrix equations. *SIAM Rev.*, 58(3):377–441, 2016.

[87] V. Simoncini and D. B. Szyld. Interpreting IDR as a Petrov-Galerkin method. *SIAM J. Sci. Comput.*, 32(4):1898–1912, 2010.

[88] G. L. G. Sleijpen and D. R. Fokkema. BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.*, 1:11–32, 1993.

[89] G. L. G. Sleijpen, P. Sonneveld, and M. B. van Gijzen. Bi-CGSTAB as an Induced Dimension Reduction method. *Appl. Numer. Math.*, 60: 1100–1114, 2010.

[90] G. L. G. Sleijpen and H. A. van der Vorst. Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. *Numer. Algorithms*, 10(2):203–223, 1995.

[91] G. L. G. Sleijpen, H. A. van der Vorst, and M. B. van Gijzen. Quadratic eigenproblems are no problem. *SIAM News*, 29(7):8–9, 1996.

[92] G. L. G. Sleijpen and M. B. van Gijzen. Exploiting BiCGstab(ℓ) strategies to Induce Dimension Reduction. *SIAM J. Sci. Comput.*, 32(5):2687–2709, 2010.

[93] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10(1):36–52, 1989.

[94] P. Sonneveld. On the convergence behavior of IDR(s) and related methods. *SIAM J. Sci. Comput.*, 34(5):A2576–A2598, 2012.

[95] P. Sonneveld and M. B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Sci. Comput.*, 31(2):1035–1062, 2008.

[96] K. M. Soodhalter. Block Krylov subspace recycling for shifted systems with unrelated right-hand sides. *SIAM J. Sci. Comput.*, 38(1):A302–A324, 2016.

[97] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[98] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2001.

[99] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Rev.*, 43(2):235–286, 2001.

[100] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Bbehavior of Nonnormal Matrices and Operator*. Princeton University Press, Princeton, NJ, USA, 2005. ISBN 9780691119465.

[101] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

[102] M. B. van Gijzen, G. L. G. Sleijpen, and J.-P. M. Zemke. Flexible and multi-shift Induced Dimension Reduction algorithms for solving large sparse linear systems. *Numer. Linear Algebra Appl.*, 22(1):1–25, 2015.

[103] M. B. van Gijzen and P. Sonneveld. Algorithm 913: An elegant IDR(s) variant that efficiently exploits biorthogonality properties. *ACM Trans. Math. Software*, 38(1):5:1–5:19, 2011.

[104] M. B. van Gijzen, C. B. Vreugdenhil, and H. Oksuzoglu. The finite element discretization for stream-function problems on multiply connected domains. *J. Comput. Phys.*, 140(1):30–46, 1998.

[105] P. Wesseling and P. Sonneveld. Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In *Approximation Methods for Navier-Stokes Problems*, R. Rautmann, editor, Berlin, Heidelberg, New York, 1980, pages 543–562. Springer-Verlag.

[106] M.-C. Yeung and T. F. Chan. ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple Lanczos starting vectors. *SIAM J. Sci. Comput.*, 21 (4):1263–1290, 1999.

[107] J.-P. M. Zemke. On structured pencils arising in Sonneveld methods. Technical Report 186, Technische Universität Hamburg-Harburg, Institut für Mathematik, Hamburg, Germany, 2014.

[108] J.-P. M. Zemke. Variants of IDR with partial orthonormalization. *Electron. Trans. Numer. Anal.*, 46:245–272, 2017.