

Gérard Meurant
Jurjen Duintjer Tebbens

Krylov Methods for Nonsymmetric Linear Systems

From Theory to Computations

Springer Series in Computational Mathematics

Volume 57

Series Editors

Randolph E. Bank, Department of Mathematics, University of California,
San Diego, La Jolla, CA, USA

Ronald L. Graham, (1935–2020), Department of Computer Science & Engineering,
University of California, San Diego, La Jolla, CA, USA

Wolfgang Hackbusch, Max-Planck-Institut für Mathematik in den
Naturwissenschaften, Leipzig, Germany

Josef Stoer, Institut für Mathematik, University of Würzburg, Würzburg, Germany

Richard S. Varga, Kent State University, Kent, OH, USA

Harry Yserentant, Institut für Mathematik, Technische Universität Berlin,
Berlin, Germany

This is basically a numerical analysis series in which high-level monographs are published. We develop this series aiming at having more publications in it which are closer to applications. There are several volumes in the series which are linked to some mathematical software. This is a list of all titles published in this series.

More information about this series at <http://www.springer.com/series/797>

Gérard Meurant · Jurjen Duintjer Tebbens

Krylov Methods for Nonsymmetric Linear Systems

From Theory to Computations



Springer

Gérard Meurant
Paris, France

Jurjen Duintjer Tebbens
Institute of Computer Science
Czech Academy of Sciences
Praha, Czech Republic

Faculty of Pharmacy
Charles University
Hradec Králové, Czech Republic

ISSN 0179-3632

ISSN 2198-3712 (electronic)

Springer Series in Computational Mathematics

ISBN 978-3-030-55250-3

ISBN 978-3-030-55251-0 (eBook)

<https://doi.org/10.1007/978-3-030-55251-0>

Mathematics Subject Classification: 65F10, 65F08, 65F15, 65F18

© Springer Nature Switzerland AG 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland



Alexei Nikolaevich Krylov (1863–1945)

Acknowledgements

The authors thank Claude Brezinski, Erin Carson and Petr Tichý for reading all or parts of earlier versions of the manuscript as well as J.-P.M. Zemke for interesting information about IDR methods.

Many thanks to Zdeněk Strakoš for initiating our research on some of the topics of this book and for his lasting friendship.

Many thanks also to Elizabeth Loew from Springer and to a few anonymous reviewers who made interesting suggestions.

The mystery of creation is like the darkness of night—it is great. Delusions of knowledge are like the fog of the morning.

Rabindranath Tagore (1861–1941), *Stray Birds*, The MacMillan Company, 1916

La science ne va pas sans supercherie: pour résoudre un problème il suffit de l'avoir bien posé.

Science does not come without trickery: To solve a problem it is enough to have it well posed.

Georges Braque (1882–1963), *Le Jour et la Nuit, Carnets 1917–1952*, Gallimard, 1952

L'univers est une ensemble dissymétrique, et je suis persuadé que la vie, telle qu'elle se manifeste à nous, est fonction de la dissymétrie de l'univers ou des conséquences qu'elle entraîne.

The universe is asymmetric and I am persuaded that life, as it is known to us, is a direct result of the asymmetry of the universe or of its indirect consequences.

Louis Pasteur (1822–1895), *Observations sur les formes dissymétriques*, Comptes rendus de l'Académie des Sciences, June 1st, 1874, LXXVII, pages 1515–1518.

Translated as in “The harvest of a quiet eye”, a selection of scientific quotations by Alan L. MacKay, The Institute of Physics, Bristol, 1977.

Contents

1	Introduction	1
2	Notation, definitions and tools	13
2.1	Notation, basic matrix types and properties	13
2.2	Eigenvalues and eigenvectors	16
2.3	The Jordan canonical form	18
2.4	Minimal polynomials	19
2.5	Singular values	20
2.6	The companion matrix	20
2.7	Hessenberg matrices	22
2.8	Tridiagonal matrices	31
2.9	Unitary transformations	33
2.10	Krylov subspaces and their basic properties	40
2.11	Finite precision arithmetic and backward stability	43
2.12	Parallel computing	46
2.13	Preconditioning	47
2.14	Stopping criterion	47
2.15	Historical notes	50
3	Q-OR and Q-MR methods	53
3.1	Definition	53
3.2	Relations between Q-OR and the associated Q-MR process	58
3.3	Residual vectors and residual polynomials	63
3.4	The inverse of U_k and the residual norms	73
3.5	Prescribing the convergence curve	77
3.6	Stagnation of Q-MR	82
3.7	Residual bounds	83
3.8	Residual norms in terms of spectral quantities	85
3.9	Relations between Q-OR/Q-MR methods with different bases	89

3.10	Generalized Q-OR/Q-MR methods	92
3.11	Historical notes	94
4	Bases for Krylov subspaces	99
4.1	Orthonormal basis	99
4.2	Hessenberg basis	109
4.3	Biorthogonal basis	112
4.4	The generalized Hessenberg process	118
4.5	Q-OR optimal basis	119
4.6	Newton and Chebyshev bases	133
4.7	Truncated bases	137
4.8	Parallel computing	146
4.9	Finite precision arithmetic and numerical experiments	149
4.9.1	Orthogonal basis	149
4.9.2	Hessenberg basis	167
4.9.3	Biorthogonal basis	167
4.9.4	Q-OR opt basis	176
4.9.5	Newton and Chebyshev bases	180
4.9.6	Truncated Q-OR basis	181
4.10	Historical notes	188
5	FOM/GMRES and variants	193
5.1	Introduction to FOM and GMRES	193
5.2	Convergence of FOM and GMRES	197
5.3	Prescribed convergence	205
5.4	Stagnation of GMRES	216
5.5	Residual norms	223
5.6	The residual polynomials	231
5.7	Study of convergence using unitary matrices	241
5.8	Estimates of the norm of the error	251
5.9	Other implementations of GMRES	263
5.10	Parallel computing	269
5.11	Finite precision arithmetic	273
5.12	Numerical examples	278
5.12.1	FOM and GMRES	278
5.12.2	Variants of GMRES	281
5.12.3	Prescribed convergence and stagnation	288
5.12.4	Residual norm bounds	290
5.12.5	Error norm estimates	292
5.13	Historical notes	292
6	Methods equivalent to FOM or GMRES	303
6.1	GCR, Orthomin, Orthodir and Axelsson's method	303
6.2	Simpler, residual-based simpler and adaptive simpler GMRES	309

6.3	Orthores	319
6.4	Q-OR Optinv	320
6.5	Parallel computing	323
6.6	Finite precision arithmetic	324
6.7	Numerical examples	327
6.7.1	GCR, Orthodir and Axelsson	327
6.7.2	ASGMRES	330
6.7.3	FOM and Orthores	334
6.7.4	Q-OR Optinv	335
6.8	Historical notes	338
7	Hessenberg/CMRH	341
7.1	Derivation of the methods	341
7.2	Comparison with GMRES and convergence of CMRH	346
7.3	Prescribed convergence	347
7.4	Stagnation of CMRH	348
7.5	Residual norms	348
7.6	Parallel computing	348
7.7	Finite precision arithmetic	349
7.8	Numerical experiments	349
7.9	Historical notes	352
8	BiCG/QMR and Lanczos algorithms	355
8.1	The Lanczos algorithms	355
8.2	Derivation of BiCG	358
8.3	QMR	368
8.4	Breakdowns and look-ahead in BiCG/QMR	373
8.5	Comparison of QMR and GMRES	378
8.6	Prescribed convergence in BiCG and QMR	379
8.7	Residual norms	382
8.8	Lanczos algorithms with look-ahead and formal orthogonal polynomials	383
8.9	Parallel computing	389
8.10	Finite precision arithmetic	391
8.11	Numerical examples	394
8.11.1	BiCG	394
8.11.2	QMR	396
8.11.3	Comparisons with GMRES	402
8.11.4	Methods with look-ahead	403
8.11.5	Ai/Bj	404
8.12	Historical notes	405
9	Transpose-free Lanczos methods	411
9.1	CGS	411
9.2	BiCGStab and extensions	421

9.3	Other product-type methods	430
9.4	Look-ahead for transpose-free methods	433
9.5	Parallel computing	434
9.6	Finite precision arithmetic	435
9.7	Numerical experiments	436
9.7.1	BiCGS	436
9.7.2	BiCGStab and variants	437
9.7.3	QMR-like methods	445
9.8	Historical notes	448
10	The IDR family	455
10.1	The primitive IDR algorithm	455
10.2	The first IDR algorithm	456
10.3	IDR(s) and variants	460
10.4	Other IDR algorithms	466
10.5	Using higher order stabilizing polynomials	471
10.6	Residual norms and convergence	475
10.7	A predecessor of IDR: ML(k)BiCGStab	476
10.8	Parallel computing	477
10.9	Finite precision arithmetic	478
10.10	Numerical examples	478
10.10.1	IDR	478
10.10.2	Variants of IDR and IDR-QMR	482
10.10.3	ML(k)BiCGStab	485
10.11	Historical notes	488
11	Restart, deflation and truncation	497
11.1	Restarted FOM and GMRES	498
11.2	Prescribed convergence	505
11.3	Restarting techniques for FOM and GMRES	515
11.4	Restarted and augmented CMRH and Q-OR Opt	535
11.5	Truncated FOM and GMRES	535
11.6	Truncated Q-OR	538
11.7	Parallel computing	539
11.8	Finite precision arithmetic	539
11.9	Numerical experiments	539
11.9.1	Restarted GMRES without preconditioning	539
11.9.2	Restarted GMRES with preconditioning	543
11.9.3	Restarting with deflation and augmentation without preconditioning	545
11.9.4	Restarting with deflation and augmentation with preconditioning	562
11.9.5	Truncated Q-OR	571
11.10	Historical notes	575

12	Related topics	579
12.1	FGMRES	579
12.1.1	Definition	579
12.1.2	Historical notes	580
12.2	GCRO	581
12.2.1	Definition	581
12.2.2	Historical notes	582
12.3	Relaxation for matrix–vector products	583
12.4	Systems with multiple right-hand sides	583
12.4.1	Definition	583
12.4.2	Historical notes	586
12.5	Shifted linear systems	588
12.5.1	Definition	588
12.5.2	Historical notes	588
12.6	Singular systems	589
12.6.1	Definition	589
12.6.2	Historical notes	589
12.7	Least squares problems	590
12.7.1	Definition	590
12.7.2	Historical notes	593
12.8	Ill-posed problems	593
12.8.1	Definition	593
12.8.2	Historical notes	594
12.9	Eigenvalue computations and rational Krylov methods	595
12.9.1	Definition	595
12.9.2	Historical notes	595
12.10	Functions of matrices	596
12.10.1	Definition	596
12.10.2	Historical notes	597
12.11	Minimum error methods	598
12.11.1	Definition	598
12.11.2	Historical notes	598
12.12	Residual replacement techniques	598
12.13	Residual smoothing techniques	600
12.13.1	Definition	600
12.13.2	Historical notes	600
12.14	Hybrid methods	600
12.15	CGNE and CGNR	601
12.16	USYMLQ and USYMQR	602

13 Numerical comparisons of methods	603
13.1 Introduction	603
13.2 Small matrices	604
13.3 Larger matrices	616
Appendix A: Test matrices and short biographical notices	629
References	639
Index	683

Chapter 1

Introduction



Solving systems of algebraic linear equations is among the most frequent problems in scientific computing. It appears in many areas like physics, engineering, chemistry, biology and many others.

When one wants to solve a real-world problem, usually the first phase is to set up a mathematical model of the problem leading to equations whose solution should give the quantities (also known as variables or unknowns) that are sought, or at least useful approximations. The model can be discrete (posed in a finite-dimensional space) or continuous (posed in an infinite-dimensional space). It can be linear or nonlinear in the sought variables. Many continuous problems lead to systems of ordinary differential or partial differential equations. Another important source of computational tasks is optimization. In these problems one attempts to maximize or minimize some quantities as functions of the given variables.

If the problem is posed in an infinite-dimensional space, the next phase of the computational process is to discretize the equations of the model. It gives some discrete equations in a finite-dimensional space, which may be linear or nonlinear. One in general has to use an iterative method to solve the nonlinear equations. In many cases this leads to solving sequences of linear systems.

The third phase is to solve at least one and possibly many linear systems. Many other computational tasks like optimization problems lead to solving several linear systems as well. These systems can be solved by a variety of different numerical methods. Most of the time the goal is to solve the linear system accurately using a computer and as fast as possible. The chosen numerical method for solving the linear system has then to be coded as efficiently as possible and to be used on a computer with computations done in finite precision arithmetic.

Various complications can arise in the day-to-day work with this type of scientific computing. For instance, the model may not be sufficiently accurate, we may not have as many equations as unknowns or too many, the discretization method may

not be accurate enough and so on. Hence, quite often one has to loop over the steps modeling–discretization–linearization–coding–solve to improve the solution.

Many numerical methods for solving linear systems are classified as either *direct* or *iterative*. Mathematically, direct methods, based on a useful, mostly triangular factorization of the system matrix (see, e.g., [256, 286]), must deliver the exact solution of the system after a known number of elementary operations (that is, additions, multiplications, divisions and square roots). On the contrary, iterative methods produce approximations of the solution by repeating some sequences of operations. Hopefully, if the method is well designed and the given problem has favorable properties, the approximations will converge to the exact solution of the given linear system. Nevertheless, the iterations have to be stopped at some point, and the number of operations to obtain a meaningful approximation of the solution (whatever it means) is not known beforehand.

Of course, things are not that simple in actual computations, and the differences between the two types of methods are not crystal clear. First of all, the elementary operations, like addition and multiplication, are not done exactly with the digital computers we have today. Therefore, except for contrived examples, a direct method cannot produce the exact solution, even though in many cases the computed solution is close to the exact one. Secondly, some of the iterative methods we will study in this book are, mathematically, direct methods but they are designed such that, at each step, they produce an approximation of the solution which is improving as the iterations proceed. Even though they can be mathematically considered as being direct methods, they are used as iterative methods. Moreover, some direct methods can, by neglecting some entries, produce an approximation of the matrix of the linear system or of its inverse that can be used to speed up the convergence of the iterative methods.

Since the beginning of the use of computers after World War II, there has been a (friendly) competition between researchers working on direct methods and those working on iterative methods. The advantage of direct methods, like modern variants of Gaussian elimination, is that they usually produce a solution close to the exact one, even though pivoting techniques have to be used to maintain some form of (weak) stability, like insensitivity to rounding errors. One important drawback is that during factorization, the factors of a sparse matrix can become filled with nonzero entries, leading to storage problems when solving very large problems. Moreover, the users do not always need a very accurate solution of the linear systems. This is the case, for instance, when the linear solve is an inner step in an algorithm for solving a nonlinear system.

On the contrary, iterative methods are not limited by storage problems (if some of them are restarted), and the class of Krylov subspace methods, the focus of this book, requires only the matrix (and sometimes its transpose or adjoint) in matrix–vector product form. Hence, they can even be used to solve systems when the matrix is not readily available as long as we are able to compute the product of the system matrix with a given vector. Iterative methods have the obvious advantage that the user can stop the iterations when satisfied with the approximate solution, but they may have stability problems and may converge poorly or not at all.

Another important class of methods to solve linear systems is *multigrid methods*. Multigrid methods were first developed for solving discretized partial differential equations. They use multiple discretization levels and combine several techniques on different levels (see, e.g., [822]). For some important classes of problems, their complexity is asymptotically optimal, but they can also be sensitive to changes in the problem [309]. The concept was extended for solving general linear systems yielding what is known as *algebraic multigrid methods* [780]. Some books devoted to multigrid are [150, 927]. As with direct and iterative methods, better results can sometimes be obtained by a combination with an iterative stationary or Krylov method or with a direct method. We do not consider multigrid methods in this book.

Krylov methods appear to be particularly suited to solve linear systems when the matrices are sparse (that is, with many zero entries) and very large. Of course, what is considered to be a large system depends on the related period of history. Without going back to the ancient Chinese 2000 years B.C., linear systems have been solved for quite a long time, long before the concept of matrix was introduced by Arthur Cayley and James J. Sylvester in the second half of the 19th century. For instance, Carl F. Gauss [403] was solving by hand and by elimination methods small linear systems of order 6 arising from least squares problems from astronomy in 1809. In 1910 André-Louis Cholesky reported solving systems of order 10 with a mechanical calculator in 4 to 5 hours; see [149]. He also solved systems of order 30 and 56 arising from least squares problems in geodesy. After World War II, the concept of what is a “large” system has evolved together with the progresses in computing speed and amount of memory of digital computers. It went from a few tens or hundreds of unknowns to hundred of millions or even billions today. Note that the fastest computers are now parallel machines and this has a strong influence on the development of algorithms today.

In this book we are concerned with solving linear systems $Ax = b$. The matrix A of order n with real or complex entries is square nonsingular and nonsymmetric (non-Hermitian in the complex case) and b is a real or complex vector. Our aim is to give an overview of the state of the art of Krylov subspace iterative methods and to study their mathematical properties. During the last 60 years, Krylov methods have progressively emerged as the most efficient iterative methods we have for solving large linear systems and they may be expected to remain so, independent of progress in modern computer-related fields like parallel and high-performance computing. Krylov subspaces are defined precisely in Chapter 2, but let us say for the moment that the primary interesting feature is that they are based on repeated multiplication of A with a vector related to b . The (Krylov) subspaces thus constructed may reveal dominant properties of the linear system even if they are still of relatively small dimension. The iterates are obtained through solving projections of the linear system to small Krylov subspaces, whose dimensions grow with the iteration number. Convergence analysis, i.e., analysis of the question of how the iterates approach the desired solution, remains challenging for solving nonsymmetric systems with Krylov methods, in spite of several decades of research efforts. Whereas for symmetric systems it is clear that convergence is dominated by the distribution of the eigenvalues of A and further influenced by components of b in an eigenvector basis, simple quan-

ties that dictate convergence behavior for general nonsymmetric systems seem not to exist, or at least have not been so far discovered. As we said, we will describe and analyze the mathematical properties of the methods but also their behavior in finite precision arithmetic. This is important because it turns out that some methods which are mathematically equivalent do not have the same stability properties and behavior when used on a computer. We also consider their implementations and coding as Matlab¹-like functions. The given codes have to be seen as templates but they can be easily translated to some other high-level programming languages provided a function for the matrix–vector product is available. To obtain practical codes, the inputs have to be tested and some breakdown conditions have to be checked.

There already exist excellent books describing Krylov methods; see [45, 349, 455, 498, 640, 673, 798, 941]. However, the present book is somehow different from those. First, we consider some methods which became popular quite recently after these books were written. Secondly, rather than studying each method and its properties one after the other, we consider them, as far as possible, in the general framework of Q-OR/Q-MR methods. As we will see in Chapter 3, Q-OR means quasi-orthogonal and Q-MR means quasi-minimum residual. We try to prove as many mathematical results as possible in this general framework, even though it has to be slightly generalized to cover some methods. This way of proceeding was inspired by a very nice paper [296] by Michael Eiermann and Oliver Ernst in which they show that many properties of Krylov methods can be proved in this quite general framework. Moreover, we prove results showing that, for many of these methods, one can construct matrices A and right-hand sides b such that A has prescribed eigenvalues and the method has a prescribed residual norm convergence curve. This is a (partial) answer to the problem of knowing which properties of the matrix and right-hand side most influenced the convergence of the method. We also give exact expressions for the residual norms as functions of the eigenvalues, the eigenvectors and the right-hand side. This precisely describes the often very complicated way how convergence depends on these quantities; see also [640].

Apart from differences in their implementations, the main difference between the many Krylov methods for nonsymmetric systems is in the kind of projections to the Krylov subspace they are using and in the type of generated basis for that Krylov subspace. Hence, a chapter is devoted to explain the construction of the most interesting bases. After the presentation of the general framework, each chapter is concerned with a particular class of methods, studying variants of the methods, applying the general results, proving some results that cannot be obtained from the general framework and considering their implementations in a Matlab-like language. Hence, this book does not only present theoretical results but it also discusses the computational aspects of the methods. The properties of the methods are illustrated by numerical examples. Since we wrote above that iterative methods are of interest for solving large linear systems, the reader might be surprised by the fact that most of our numerical examples use small matrices. However, the main goal of our examples is to illustrate some mathematical or numerical properties and differences in behavior

¹Matlab® is a trademark of the MathWorks

of the methods and to do so it is not necessary to use very large examples. Every chapter ends with some historical comments about the bibliography. Note that not all the references are cited in the text, but they are all of interest for a reader who wants to learn more about Krylov methods.

Let us now briefly summarize the contents of each chapter.

In Chapter 2, we first introduce the notation we use and recall some definitions. We consider eigenvalues and eigenvectors in Section 2.2, the Jordan canonical form and the singular value decomposition in Section 2.3. Since Krylov methods are closely linked to polynomials, some useful results about minimal polynomials of a matrix and a vector are recalled in Section 2.4. Krylov methods are also related, explicitly or implicitly, to Hessenberg matrices whose main properties are described in Section 2.7. A result which is particularly important for our purposes is that an unreduced Hessenberg matrix H can be factored as $H = UCU^{-1}$ where U is an upper triangular matrix and C is the companion matrix corresponding to the spectrum of H . We also recall results about inverses of upper Hessenberg matrices. In some methods, we will have to solve (small) linear systems with Hessenberg matrices. For reasons of stability, this is done using Householder or Givens unitary transformations whose definitions are recalled in Section 2.9. Then, in Section 2.10, we define Krylov subspaces and matrices and discuss their properties. Since our goal is to solve linear systems using computers, we recall some basic facts about finite precision arithmetic in Section 2.11. Today, all computers that are used to solve large problems are parallel machines. Even though parallel methods are not our main concern in this book, Section 2.12 briefly discusses the relations of Krylov methods with parallelism. Preconditioning of the linear system is generally used to speed up the convergence of Krylov methods. This issue is considered in Section 2.13. Stopping criteria for the iterative methods described in this book are discussed in Section 2.14.

All Krylov methods use, explicitly or implicitly, bases of the Krylov subspaces. Chapter 3 defines abstract Q-OR and Q-MR methods and studies their properties regardless of the particular type of basis which is used. After the definition given in Section 3.1, we study the relations between the iterates of a general Q-OR method and the corresponding Q-MR method using the same basis (Section 3.2), as well as the relations between the norm of the Q-OR residual and the norm of the Q-MR quasi-residual. The Q-OR and Q-MR residual vectors are given by polynomials in A applied to the initial residual vector. In Section 3.3, we study their coefficients and their zeroes. Then, in Section 3.4 expressions for the Q-OR residual norms and Q-MR quasi-residual norms are given. An important result is that these norms can be related to the first row of the matrix U^{-1} in the decomposition $H = UCU^{-1}$ that we introduced in Chapter 2. It is shown in Section 3.4 how to construct matrices with a prescribed spectrum and right-hand sides such that Q-OR and Q-MR methods have a prescribed residual norm convergence curve. It implies that there is no simple relationship between Q-OR/Q-MR convergence and the spectral properties of the matrix A . This also provides results about the possible stagnation of Q-MR methods described in Section 3.6. Quite often the convergence of specific Q-OR/Q-MR methods is studied by obtaining bounds for the residual norms. Therefore, we show

how to derive such bounds in Section 3.7. In some cases these bounds can be useful to provide insights about the convergence, even though we also show in Section 3.8 that it is possible to exactly express the (quasi-) residual norms as functions of the eigenvalues and eigenvectors of A and the right-hand side. This is shown for the case when A is diagonalizable but can be done for defective matrices as well. In Section 3.9 the relationships between two pairs of abstract Q-OR/Q-MR methods that use a different type of Krylov subspace basis are explored. As we wrote above, some Krylov methods do not fit exactly in the Q-OR/Q-MR framework defined at the beginning of this chapter. Therefore, we slightly generalize it in Section 3.10.

Since our general framework of Chapter 3 does not refer to a particular basis of the Krylov subspaces, Chapter 4 is devoted to the construction of some useful bases. The first basis that we consider in Section 4.1 is orthonormal. This is the best conditioned basis one can think of. Mathematically, it corresponds to the factor Q in a QR factorization of the Krylov matrix. However, in practice, it is computed by the Arnoldi process for which there exist several implementations whose relative merits are discussed. Section 4.2 is devoted to the Hessenberg basis which corresponds to the lower triangular factor L in an implicit LU factorization with partial pivoting of the Krylov matrix. The interest of this basis is that, contrary to the orthonormal basis, dot products are not needed for its computation. The processes for computing the orthonormal and Hessenberg bases use long recurrences, the computation of a new basis vector requiring all the previous basis vectors. The cost of the computation of a basis vector increases as we proceed, and the needed storage is increasing too. Therefore, it can be useful to have ways to compute bases with short recurrences. As described in Section 4.3, this can be done by generating a biorthogonal basis. However, this requires the computation of matrix–vector products with A^T (or the adjoint matrix in the complex case). The previous bases can be seen (with some variations) as the results of particular cases of the generalized Hessenberg process which is briefly recalled in Section 4.4. In Section 4.5 we study a basis called the Q-OR optimal basis. It is non-orthogonal but optimal in the sense that the residual norms are minimized when using the corresponding Q-OR method. The existence of this basis shows that the distinction between Q-OR and Q-MR methods is not as clear as one might think. Section 4.6 is devoted to what is known as a Newton basis. This type of basis was introduced because the construction of orthonormal bases using the Arnoldi process is not easily parallelizable. In a Newton basis, the basis vectors are constructed with a recurrence $v_{k+1} = \alpha_k(A - \xi_k I)v_k$ and then orthogonalized using parallel variants of the QR algorithm. We discuss how to choose the shifts ξ_k to obtain a practical algorithm. We also discuss the Chebyshev basis which is based on polynomials satisfying a three-term recurrence. Section 4.7 presents other ways to construct bases based on truncation of the long recurrences. Finally, in Section 4.9, we discuss the computation of the bases in finite precision arithmetic and give numerical examples. Note that, in some methods to be studied in the next chapters, the basis is constructed implicitly and, at first, it may not be so obvious to the reader that these methods are Q-OR or Q-MR methods.

In Chapter 5 we specialize the results obtained with the Q-OR/Q-MR framework to the pair FOM/GMRES. FOM means *full orthogonalization method* and GMRES

means *generalized minimal residual*. They both use an orthonormal basis constructed with the Arnoldi process. Section 5.1 recalls their definitions and discusses several implementations using the classical or modified Gram–Schmidt versions of the Arnoldi process as well as one using Householder transformations. In Section 5.2 the convergence of FOM and GMRES is studied by providing bounds for the residual norms in terms of polynomials, and different possibilities are explored and discussed: the field of values, the polynomial numerical hull and the pseudospectrum. How to construct matrices and right-hand sides with a prescribed FOM or GMRES residual norm convergence curve is considered in Section 5.3. For this pair of methods not only the eigenvalues of the matrix can be prescribed but also the Ritz and the harmonic Ritz values at each iteration. This shows that, generally, convergence does not depend on the convergence of the (harmonic) Ritz values to the eigenvalues of A . In Section 5.4 we characterize the linear systems for which GMRES residual norms stagnate partially or completely. Section 5.5 specializes the exact expressions for the residual norms as functions of the eigenvalues and eigenvectors obtained in Chapter 3 using the fact that the basis is orthonormal. In the GMRES case these expressions simplify when the matrix A is normal. In Section 5.6 we give exact expressions for the coefficients of the residual polynomials of FOM and GMRES. GMRES convergence using unitary matrices is studied in Section 5.7. In Section 5.8 we show how to compute estimates of the norm of the error during the FOM or GMRES iterations. This is a generalization of what was done for the conjugate gradient method in [672, 674, 689] for the symmetric case. Another implementation of the second phase of GMRES, that is, the solution of the least squares problem and the computation of the norm of the residual [49], is discussed in Section 5.9. Section 5.10 considers parallel variants of GMRES. Section 5.11 is devoted to the behavior of FOM and GMRES in finite precision arithmetic with a discussion of reorthogonalization and backward stability. The main advantage of the modified Gram–Schmidt variant of GMRES is that it has been proved to be backward stable. The results of this chapter are illustrated by numerical examples in Section 5.12.

Chapter 6 is a continuation of Chapter 5 since it is devoted to methods which are mathematically equivalent to FOM or GMRES. Of course, their behavior in finite precision arithmetic might be different, and some of them are (much) less stable than FOM or GMRES. In Section 6.1 we consider the generalized conjugate residual (GCR) method, Orthomin, Orthodir and Axelsson's method. These four methods are mathematically equivalent to GMRES in the sense that they minimize the residual norm. Note that this is true for Orthomin only if it is not truncated in which case it is similar to GCR. Its truncated version was very popular in the 1980s in the oil industry. Simpler GMRES, considered in Section 6.2, was proposed to avoid the reduction to triangular form of the upper Hessenberg matrix in GMRES. However, it was shown to be much less stable than GMRES. We also discuss the modifications that were introduced to improve this method. Orthores which is described in Section 6.3 is a method mathematically equivalent to FOM. Section 6.4 describes a method, mathematically equivalent to GMRES, using the non-orthogonal Q-OR optimal basis described in Chapter 4. Section 6.5 is devoted to parallel computing. Section 6.6

recalls results about the methods of this chapter in finite precision arithmetic. This is complemented by numerical examples in Section 6.7.

Chapter 7 is mainly devoted to CMRH (an acronym for Changing Minimal Residual method based on the Hessenberg process) which is a Q-MR method using the Hessenberg basis described in Chapter 4. CMRH is an interesting method because, contrary to many other Krylov methods, there are no dot products involved, except maybe for the stopping criterion. As GMRES, it uses Givens rotations to solve (small) least squares problems involving upper Hessenberg matrices. The Hessenberg method is the corresponding Q-OR method. The derivation of CMRH is recalled in Section 7.1, and its computer implementation is discussed. Section 7.2 compares the residual norms of CMRH with those of GMRES. In Section 7.3, we show how to construct linear systems with a prescribed CMRH (quasi-) residual norm convergence curve. Stagnation of CMRH is considered in Section 7.4. Section 7.5 gives an upper bound for the residual norms. Sections 7.6 and 7.7 are devoted to brief comments on parallel versions of CMRH and to finite precision arithmetic. Section 7.8 presents a few numerical experiments comparing CMRH to GMRES. For many problems the values of the CMRH residual norms are not far from those of GMRES.

In Chapter 8 we consider the Biconjugate Gradient (BiCG) and the QMR algorithms. To distinguish the latter method from the abstract class of Q-MR methods of the general framework in this book, we denote it without a dash, see [379]. These two methods use a biorthogonal basis and are based on Lanczos nonsymmetric algorithms which are described in Section 8.1. The main interest of these methods is that they are using short recurrences and the storage does not grow with the iteration number. However, this advantage is mitigated by the fact that they use matrix–vector products with the transpose of the matrix (or its adjoint), and they may possibly break down with a division by zero (or by a tiny number). In Section 8.2 we give a detailed derivation of BiCG from the Lanczos algorithm. Section 8.3 describes the QMR method which minimizes the quasi-residual norm. It gives much smoother residual norm convergence curves than BiCG but it is more difficult to code. Possible breakdowns of BiCG and QMR are discussed in Section 8.4. Some of these breakdowns or near-breakdowns can be cured by look-ahead techniques, relaxing the biorthogonality conditions to block biorthogonality. Section 8.5 compares the residual norms obtained with QMR and GMRES. In Section 8.6 we discuss how to construct linear systems with prescribed eigenvalues and residual norms in BiCG and QMR. This is less straightforward than what we have done for FOM/GMRES. Section 8.7 describes upper bounds for the residual norms. In Section 8.8 we consider other Lanczos algorithms that are defined using formal orthogonal polynomials. They also use look-ahead techniques. Parallel computing versions of BiCG and QMR are considered in Section 8.9. Section 8.10 discusses the analysis of BiCG in finite precision arithmetic. In this respect, the situation of algorithms based on biorthogonal basis and short recurrences is much less satisfactory than, for instance, for GMRES. The biorthogonality of the basis vectors can be lost quite rapidly, and this leads to a delay in the convergence of the methods. These methods are difficult to analyze in finite precision because of the possible (near-) breakdowns. Section 8.11 shows numerical examples that illustrate the problems we may have with the methods described in

this chapter for solving difficult examples of linear systems. We are also warning the reader and discuss the problems one can have when numerically comparing the convergence of methods with different characteristics; here, we choose BiCG, QMR and GMRES as an example.

As we have seen in Chapter 8, a shortcoming of the methods based on the Lanczos algorithms and the biorthogonal basis is that they use matrix–vector products with A^T . In Chapter 9 we describe methods which were designed to avoid using A^T but still use short recurrences. These methods are known as transpose-free Lanczos or product-type methods. They are mainly based on the simple fact that $(A^T x, y) = (x, Ay)$ and most of them use polynomials which are the product of the BiCG residual polynomial with some other polynomials. In Section 9.1 we consider the conjugate gradient squared method (CGS, not to be confused with the acronym for the classical Gram–Schmidt algorithm). CGS is, so to speak, a squared version of BiCG. At first, it is not obvious that this method fits in our general framework. However, we show that it can be considered as a generalized Q-OR method. When BiCG residual norms oscillate, which is often the case, those of CGS oscillate even more. BiCGStab, described in Section 9.2, was designed to improve upon CGS, using a local minimization of the residual norm. We also show that BiCGStab is a generalized Q-OR method. BiCGStab is perhaps the most popular Krylov method for nonsymmetric linear systems; however, surprisingly little convergence results exist which are tailored to it. We also consider extensions of BiCGStab, like BiCGStab2 and BiCGStab(ℓ) that were introduced to improve convergence. Section 9.3 briefly reviews many other product-type methods that were proposed after BiCGStab. All these methods avoid using A^T but they can still suffer from breakdowns. Look-ahead techniques are considered in Section 9.4. Parallel versions of the algorithms in this chapter are described in Section 9.5. Unfortunately, not much is known about the theory of transpose-free methods in finite precision arithmetic; see Section 9.6. In Section 9.7 we compare the methods of this chapter on some numerical examples.

In Chapter 10 we consider a family of algorithms known as IDR which means *induced dimension reduction*. We present them in chronological order. We start in Section 10.1 with what can be called the primitive IDR algorithm designed at the end of the 1970s. This algorithm was never published probably since it does not always converge in finite precision arithmetic and it is unstable for some problems. Results were much better with the algorithm described in Section 10.2 which was based on what is now known as the IDR theorem. This algorithm computes residual vectors which belong to subspaces of shrinking dimensions. It uses only one so-called shadow vector. In Section 10.3 we study IDR(s) an algorithm using s shadow vectors. We discuss the relationships of the subspaces used in this algorithm with Krylov subspaces. Like some product-type methods, it can also be considered as a generalized Q-OR method. It uses a first-order polynomial in A to locally minimize the residual norms. Other IDR algorithms using local orthogonalization are described in Section 10.4. They can be used to develop Q-MR variants of IDR algorithms. In Section 10.5 we consider IDR(s)stab(ℓ) which uses higher order stabilizing polynomials. Section 10.6 shows how to obtain upper bounds for the residual norms. In Section 10.7 we describe ML(k)BiCGStab, a method using several shadow vectors,

which can be considered as a predecessor of IDR algorithms. Section 10.8 comments about IDR algorithms and parallel computing. As for product-type methods, not much is known theoretically about IDR algorithms in finite precision arithmetic, see Section 10.9. Section 10.10 compares different IDR algorithms on some examples.

Methods using long recurrences like FOM or GMRES cannot be used to solve large practical problems because the cost per iteration and the storage grow with the number of iterations. Chapter 11 discusses the remedies that have been proposed for this problem. They are known as *restarting* and *truncation*. With restarting the algorithm is stopped after a given number of iterations and restarted with the current approximation. The other possibility is to truncate the long recurrences, using only a few of the previous basis vectors. Unfortunately, these methods do not possess the same nice properties as their counterparts using long recurrences. In Section 11.1 we consider the restarted versions of FOM and GMRES named FOM(m) and GMRES(m) where m is the number of iterations in each cycle. The main problem with a method like GMRES(m) is that it may not converge at all, stagnating in each cycle. We also show that if GMRES(m) stagnates for some iterations before the restart, it stagnates for the same number of iterations after the restart. Then, we derive bounds for the residual norms and consider the residual polynomials. In Section 11.2 we show that, modulo certain restrictions concerning a possible stagnation, one can construct linear systems with a prescribed residual norm convergence curve inside each cycle of restarted FOM and GMRES. Ritz and harmonic Ritz values can be prescribed as well.

In Section 11.3 we study a few restarting techniques for GMRES. The first idea we consider is varying the restart parameter m for one cycle to the next. Unfortunately, increasing m (if it is possible considering storage problems) may not always improve the convergence, and these techniques are not always efficient. Another technique is *augmentation*. It amounts to considering a subspace $\mathcal{K}_m(A, r_0) + \mathcal{W}$ instead of the usual Krylov subspace, that is, some vectors are appended to the basis. There are so far two kinds of vectors to append, approximate eigenvectors of A or rough approximations of the error vector. Ritz (resp. harmonic) approximate eigenvectors are used for FOM (resp. GMRES). We describe several ways to append the new vectors to the basis. The idea of appending approximate eigenvectors comes from the wish to remove the influence of some eigenvalues (generally those with small modulus) on the convergence. This is why these techniques are also called *deflation*. However, we have seen in Chapter 5 that for non-normal matrices convergence may not depend too much on the eigenvalues. Therefore, deflation cannot always be useful. Another technique is to construct preconditioners that remove some eigenvalues from the spectrum. Restarted versions of CMRH and Q-OR Opt are briefly considered in Section 11.4.

Section 11.5 is devoted to truncated FOM and GMRES. We show how to introduce new vectors p_k computed from truncated recurrences leading to a simple update formula for the Q-OR or Q-MR approximate solutions. Roughly speaking, we just need to store $2m$ vectors. In Section 11.6 we introduce truncated Q-OR methods using the bases defined in Chapter 4. On some examples these methods are more

efficient than truncated GMRES. Sections 11.7 and 11.8 briefly consider parallel computing and finite precision computations. Section 11.9 illustrates the results of this chapter with numerical experiments.

In Chapter 12 we give pointers to the literature concerning topics that we have not covered in the previous chapters. They are as follows:

- Section 12.1: FGMRES, a GMRES-like method that can use variable preconditioners.
- Section 12.2: GCRO, a method with nested iterations based on GCR and GMRES.
- Section 12.3: Inexact matrix–vector products, that is, when the products Av are computed approximately.
- Section 12.4: Systems with multiple right-hand sides.
- Section 12.5: Shifted linear systems with matrices $A - \mu I$.
- Section 12.6: Singular systems.
- Section 12.7: Least squares problems.
- Section 12.8: Ill-posed problems with noisy right-hand sides.
- Section 12.9: Eigenvalue computations and rational Krylov methods.
- Section 12.10: Functions of matrices, $f(A)b$ or $u^T f(A)v$.
- Section 12.11: Minimum error methods which minimize the error norm.
- Section 12.12: Residual replacement techniques to improve the maximum attainable accuracy.
- Section 12.13: Residual smoothing techniques.
- Section 12.14: Hybrid methods which combine Krylov methods with some other methods like Chebyshev iterations.
- Section 12.15: CGNE and CGNR which use the conjugate gradient method applied to a symmetrized system like $A^T Ax = A^T b$ or $AA^T y = b$.
- Section 12.16: USYMLQ and USYMQR, two methods based on a tridiagonalization algorithm.

In Chapter 13 we present some more numerical experiments comparing different Krylov methods. We use two sets of matrices, one with matrices of small order and another one with larger matrices. As we can have guessed, there is no clear winner in this comparison. What is the best method depends on what is our definition of “best” (smallest residual or error norm, or smallest computing time) and it is, unfortunately, problem-dependent. There is no clear answer to the question: “Which Krylov method should I use to solve my nonsymmetric linear system?”.

Appendix A describes the linear systems which are used as examples in the previous chapters. In Appendix B we give short biographical notices for some of the mathematicians cited in the text.

Most algorithms considered in this book are described in a Matlab-like language. Then, we can code the matrix–vector product of A with v as $\mathbf{Av}=\mathbf{A}^*\mathbf{v}$ without paying attention to the data structure which is used to store A when the matrix is sparse, that is, when many entries are zero. The authors do not claim that these codes are fully optimized. In fact, in some cases we trade efficiency for clarity. But they can be used as templates for the development of more sophisticated implementations.

There exist several available software packages (written in C++ or Fortran) implementing some of the methods described in this book:

- PETSc [65, 66] (<https://www.mcs.anl.gov/petsc/>),
- Hypre [341] (<https://computation.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods>),
- Trilinos [79, 511] (<https://trilinos.org/>),
- Itsol, Parms, Sparskit (<https://www-users.cs.umn.edu/~saad/software/>).

The authors of this book do not guarantee that the Internet addresses that are cited in the text will be correct and available forever.

This book can be useful for both practitioners and for readers who are more interested in theory. Together with a review of the state of the art, it presents a number of recent theoretical results of the authors, some of them unpublished, as well as a couple of original algorithms. Some of the derived formulas might be useful for the design of possible new methods or for future analysis. For the more applied user, the book gives an up-to-date overview of the majority of the available Krylov methods for nonsymmetric linear systems, including well-known convergence properties and, as we said above, template codes that can serve as the base for more individualized and elaborate implementations. A number of numerical examples demonstrate the properties and the behavior of the described methods.

Chapter 2

Notation, definitions and tools



In this chapter we set up the notation to be used, we recall some basic notions from linear algebra that are useful for our purposes and we introduce *Krylov subspaces* and some of their properties.

2.1 Notation, basic matrix types and properties

As it is customary, matrices are denoted by capital letters like A and their entries by $a_{i,j}$, $i, j = 1, \dots, n$ or, sometimes, by capital Greek letters. The leading principal submatrices of A , for $i, j \leq k$, are denoted as A_k , $k = 1, \dots, n - 1$. $A_{n,k}$, $k < n$ refers to the k first columns of A and $A_{n,n}$ is abbreviated as A . $A_{i:j,k:\ell}$ denotes the submatrix formed from rows i till j and columns k till ℓ of a matrix A ; $A_{i:j,:}$ denotes rows i till j of A , and $A_{:,k:\ell}$ denotes columns k till ℓ . The underline notation \underline{A}_k denotes the $(k + 1) \times k$ matrix that results from appending the k first entries of the $k + 1$ st row of A at the bottom of A_k . The letter I represents the identity matrix of appropriate order. However, when it is needed to explicitly display the order k we will use I_k . Vectors are generally denoted by lowercase letters like x, y, v and so on. We will also use sometimes lowercase Greek letters. Since we are interested in iterative methods, we denote vectors related to some iteration k as y_k with a subindex and its j th component as $[y_k]_j$ to avoid any confusion. A vector e_j is the j th column of an identity matrix of appropriate order. Generally, lowercase Greek letters like α, β, \dots represent scalars. An exception for scalars will be the sines s_i and cosines c_i in rotation matrices as well as some constants in inequalities.

An overbar represents the conjugate of a complex number. Hence, $\bar{\alpha}$ is the conjugate of $\alpha \in \mathbb{C}$. The vector x^* (resp. the matrix A^*) is the conjugate transpose of x (resp. A). We write x^T or A^T for the transpose of a vector or a matrix. The matrix A^{-*} is $[A^{-1}]^*$. A matrix A satisfying $A^* = A$ is *Hermitian* and a real matrix A satisfying $A^T = A$ is *symmetric*. A matrix A satisfying $A^* = -A$ is *skew-Hermitian*, and a real

matrix A satisfying $A^T = -A$ is *skew-symmetric*. A matrix A satisfying $A^*A = I$ is *unitary*, and a real matrix A satisfying $A^TA = I$ is *orthogonal* or *orthonormal* depending on the context.

For vectors x and y of the same size, the scalar $(x, y) \equiv y^*x$ defines a scalar product also referred to as inner product or dot product. The quantity $\|x\| = \sqrt{x^*x}$ denotes the associated vector norm, called *the ℓ_2 norm (or Euclidean norm or simply norm)* of the vector x , that is, the square root of the sum of the moduli of its components squared. If $y^*x = 0$, the vectors x and y are orthogonal to each other. The columns of unitary and orthogonal matrices are orthogonal to each other (in fact, they are *orthonormal* because they have in addition a unit norm).

For a matrix A , $\|A\|$ is the matrix norm induced by the Euclidean vector norm; see, for instance, [438]. It is called the *ℓ_2 norm*. $\|A\|_F$, the *Frobenius norm* is the Euclidean norm of the vector of all entries of A . The Euclidean norm of a vector is invariant under multiplication with a unitary matrix, i.e., for a unitary matrix Q and vector x of corresponding size, $\|Qx\| = \sqrt{x^*Q^*Qx} = \|x\|$. It can be shown that the matrix ℓ_2 norm and the Frobenius norm are invariant as well under multiplication with a unitary matrix (from the left and from the right).

The 1-norm and the ∞ -norm of a matrix A are defined as

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{i,j}|, \quad \|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{i,j}|.$$

For a square nonsingular matrix A , these norms satisfy

$$\frac{1}{\sqrt{n}} \|A\|_\alpha \leq \|A\| \leq \sqrt{n} \|A\|_\alpha, \quad \text{with } \alpha = 1, \infty.$$

The 1-norm (resp. the ∞ -norm) of a vector is the sum (resp. maximum) of the moduli of its components.

For a nonsingular matrix A and an arbitrary matrix norm $\|\cdot\|_\alpha$, the corresponding condition number is given by $\kappa(A) \equiv \|A\|_\alpha \cdot \|A^{-1}\|_\alpha$. When needed it is also denoted as $\kappa_\alpha(A)$.

A Hermitian matrix A is called *positive definite* if $x^*Ax > 0$ for all nonzero vectors x ; it is called *positive semi-definite* if $x^*Ax \geq 0$ for all vectors x . A square matrix A is called *positive real* if the real part $\operatorname{Re}(x^*Ax)$ of x^*Ax is positive for all nonzero vectors x .

A *lower triangular* square matrix L is such that $l_{i,j} = 0$ for $j > i$ and an *upper triangular* square matrix U is such that $u_{i,j} = 0$ for $j < i$. An *LU factorization* of a square matrix A is a factorization of the form $A = LU$, where L is lower triangular with ones on the main diagonal and U is upper triangular. A *Cholesky factorization* of a Hermitian positive definite matrix is a factorization of the form $A = LL^*$, where L is lower triangular with real positive entries on the main diagonal. A *QR factorization* of a rectangular matrix A of size $n \times m$ is a factorization of the form $A = QR$, where $Q^*Q = I$ and the leading square principal submatrix of R is upper triangular. More precisely, if $n \geq m$, then either Q is of size $n \times m$ with $Q^*Q = I_m$

and R is square upper triangular (the “economy size” QR decomposition), or Q is square unitary and R of size $n \times m$ has an upper triangular leading $m \times m$ principal submatrix. If $n < m$, Q is square unitary and R of size $n \times m$ has an upper triangular leading $n \times n$ principal submatrix.

An upper (resp. lower) bidiagonal matrix is an upper (resp. lower) triangular matrix with nonzero entries only on the main and on the first upper (resp. lower) diagonal.

The inverse of a nonsingular upper triangular matrix can be constructed column by column. Let U_{k+1} be such a nonsingular matrix of order $k + 1$. Then,

$$U_{k+1}^{-1} = \begin{pmatrix} U_k^{-1} - \frac{1}{u_{k+1,k+1}} U_k^{-1} U_{1:k,k+1} \\ \frac{1}{u_{k+1,k+1}} \end{pmatrix}. \quad (2.1)$$

The analog result holds for lower triangular matrices. For a general square matrix M with block partitioning

$$M = \begin{pmatrix} M_k & M_{1:k,k+1:n} \\ M_{k+1:n,1:k} & M_{k+1:n,k+1:n} \end{pmatrix} \equiv \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

where $M_k = A$ is nonsingular, the *Schur complement of A in M* is the matrix $S_M(A) \equiv D - CA^{-1}B$. Similarly, if $M_{k+1:n,k+1:n} = D$ is nonsingular, the *Schur complement of D in M* is the matrix defined as $S_M(D) \equiv A - BD^{-1}C$. The matrix M is nonsingular if and only if the Schur complement of D in M is and in that case the Schur complement of A in M is nonsingular and

$$M^{-1} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} S_M(D)^{-1} & -S_M(D)^{-1}BD^{-1} \\ -D^{-1}CS_M(D)^{-1} & S_M(A)^{-1} \end{pmatrix}. \quad (2.2)$$

There holds that if M is Hermitian positive definite, then so is $S_M(D)$.

The (column) *rank* of a matrix A is the number of linearly independent columns of A , which is equal to the number of linearly independent rows. The *Sherman–Morrison formula* (see [75, 823]) shows what is the inverse of a rank-one modification of a nonsingular square matrix A of order n . Let u and v be two vectors with n components. The matrix $A + uv^*$ is nonsingular if and only if $1 + v^*A^{-1}u \neq 0$ and

$$(A + uv^*)^{-1} = A^{-1} - \frac{A^{-1}uv^*A^{-1}}{1 + v^*A^{-1}u}. \quad (2.3)$$

This has been generalized to the *Sherman–Morrison–Woodbury formula* (see [985]). Let U, V be two matrices of arbitrary rank r , $r \leq n$, and let C be nonsingular such that the product UCV exists and is square of order n . Then,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \quad (2.4)$$

An assertion equivalent with A being nonsingular is that its determinant is nonzero. For the definition of and formulae to compute the determinant we refer

to [348, 690]. We mention here only a formula for the determinant of the product of matrices which is important later in the book. The *Cauchy–Binet formula* (see, for instance, [396]) is a relation giving the determinant of the product of two rectangular matrices of compatible dimension. Let $k \leq n$, A be $k \times n$ and B be $n \times k$. Then,

$$\det(AB) = \sum_{\mathcal{I}_k} \det(A_{:, \mathcal{I}_k}) \det(B_{\mathcal{I}_k,:}), \quad (2.5)$$

where the summation is over all the ordered subsets \mathcal{I}_k of k integers, $1 \leq i_1 < i_2 < \dots < i_k \leq n$. Note that if A is the conjugate transpose of B we have

$$\det(B^*B) = \sum_{\mathcal{I}_k} |\det(B_{\mathcal{I}_k,:})|^2.$$

Principal minors of a matrix are determinants of submatrices of the form $A_{\mathcal{I}, \mathcal{I}}$ for some subset \mathcal{I} of the row and column index sets, i.e., the indices of the selected rows and columns coincide. For a nonsingular matrix A , *Cramer's rule* states that the solution x of the linear system $Ax = b$ for a right-hand side vector b has the entries

$$x_j = \frac{\det(\{A(b)\}_j)}{\det(A)}, \quad j = 1, \dots, n,$$

where $\{A(b)\}_j$ denotes the matrix A with its j th column replaced with b .

A matrix A where all entries on the same diagonal are equal, i.e., $a_{i,j} = a_{k,\ell}$ if $i - j = k - \ell$, is a *Toeplitz matrix*. A matrix A where all entries on the same anti-diagonal are equal, i.e., $a_{i,j} = a_{k,\ell}$ if $i + j = k + \ell$, is a *Hankel matrix*.

For complex numbers λ_i , $i = 1, \dots, n$, the $n \times k$ matrix

$$\mathcal{V}_{n,k} = \begin{pmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{k-1} \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_n & \lambda_n^2 & \cdots & \lambda_n^{k-1} \end{pmatrix} \quad (2.6)$$

is called a *Vandermonde matrix*. The determinant of a square Vandermonde matrix (with $k = n$) is

$$\det(\mathcal{V}_n) = \prod_{1 \leq i < j \leq n} (\lambda_j - \lambda_i). \quad (2.7)$$

The matrix \mathcal{V}_n is nonsingular if and only if the λ_i are distinct.

2.2 Eigenvalues and eigenvectors

An *eigenvector* x is a nonzero vector (with real or complex components) whose direction does not change when multiplied by a square matrix A . Therefore we have

$$Ax = \lambda x, \quad x \neq 0. \quad (2.8)$$

Usually the eigenvectors are normalized to be of unit ℓ_2 norm. The scalar λ is called an *eigenvalue* of A .

Since, to have a nonzero eigenvector, the matrix $A - \lambda I$ must be singular, the eigenvalues of A satisfy the equation

$$\det(\lambda I - A) = 0. \quad (2.9)$$

By expanding the determinant, we see that the left-hand side of equation (2.9) is a monic polynomial p_C (that is, with a leading coefficient equal to 1) of degree n in λ , and the eigenvalues are the roots of this polynomial. Hence, there is always at least one eigenvalue (and corresponding eigenvector), and there are at most n distinct eigenvalues λ_i , $i = 1, \dots, n$; they can be real or complex. If the matrix A is real, the complex eigenvalues appear in complex conjugate pairs; if λ is an eigenvalue, so is $\bar{\lambda}$. Moreover, the corresponding eigenvectors are complex conjugates. If the matrix A is real and the order n is odd, there is at least one real eigenvalue. But a real matrix of even order may have only complex eigenvalues in $n/2$ complex conjugate pairs. A pair (λ, x) satisfying equation (2.8) is called an *eigenpair* of A . The set of all the eigenvalues of A is the *spectrum* of A that we denote sometimes as $\sigma(A)$. The *spectral radius* $\rho(A)$ of A is the modulus of its eigenvalue of largest modulus.

The polynomial p_C defined above is called the *characteristic polynomial*. It can be factored as

$$p_C(\lambda) = \prod_{i=1}^n (\lambda - \lambda_i). \quad (2.10)$$

Let $\lambda_1, \dots, \lambda_\ell$ with $\ell \leq n$ be the distinct eigenvalues of A . Then, we can group the common factors in the right-hand side of relation (2.10) and write the characteristic polynomial as

$$p_C(\lambda) = (\lambda - \lambda_1)^{m_1} \cdots (\lambda - \lambda_\ell)^{m_\ell}.$$

The integer m_i is called the *algebraic multiplicity* of the eigenvalue λ_i , that is, its multiplicity as a multiple root of the characteristic polynomial. Of course, we have

$$1 \leq m_i \leq n, \quad i = 1, \dots, \ell, \quad \sum_{i=1}^{\ell} m_i = n.$$

There may be more than one eigenvector associated to an eigenvalue λ_i . The number of linearly independent eigenvectors associated with λ_i is the *geometric multiplicity* g_i of the eigenvalue. Therefore, g_i is the dimension of $\text{Ker}(\lambda_i I - A)$ where Ker is the kernel of the matrix, that is, the set of vectors y such that $(\lambda_i I - A)y = 0$. The geometric multiplicity is smaller than or equal to the algebraic multiplicity. An eigenvalue λ_i of algebraic multiplicity $m_i = 1$ is said to be *simple*. If $m_i > 1$ and there are m_i associated independent eigenvectors, it is said to be *semi-simple*.

2.3 The Jordan canonical form

For the material in this section we refer to [531, Section 3.1]. It is often useful to reduce a general matrix to a simpler form, for instance, through LU or QR factorizations introduced earlier. There exist several so-called canonical forms. We will consider two of them in this section and other forms and factorizations later on.

For every square matrix with complex entries there exists a nonsingular matrix X such that $X^{-1}AX = J$ where J is a block diagonal matrix with blocks of a simple bidiagonal structure. The diagonal blocks are of the form

$$J^{(i,j)} = \begin{pmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & \\ & & & 1 \\ & & & \lambda_i \end{pmatrix}, \quad i = 1, \dots, \ell, \quad j = 1, \dots, g_i, \quad (2.11)$$

where ℓ is the number of distinct eigenvalues of A and g_i is the geometric multiplicity of λ_i . The matrices $J^{(i,j)}$ are called *Jordan blocks* and have only two nonzero constant diagonals. The factorization $A = XJX^{-1}$ is called the *Jordan canonical form* of A .

Let $d_{i,j}$ be the order of $J^{(i,j)}$. Then, the characteristic (and irreducible) polynomial of $J^{(i,j)}$ is $(\lambda - \lambda_i)^{d_{i,j}}$. There may be several Jordan blocks concerning one eigenvalue and the block diagonal matrix whose diagonal blocks are all Jordan blocks $J^{(i,j)}$, $j = 1, \dots, g_i$ associated with the eigenvalue λ_i is called a *Jordan box* and denoted as $B^{(i)}$ when the concerned Jordan blocks are numbered sequentially. The matrix $B^{(i)}$ is of order m_i and contains g_i blocks. The Jordan canonical form is unique up to the order of the diagonal Jordan blocks.

The Jordan canonical form is linked to the decomposition of the space \mathbb{C}^n in cyclic subspaces. A subspace \mathcal{S} is A -invariant if $A\mathcal{S} \subseteq \mathcal{S}$ and is called a *cyclic subspace*. Let m be the number of Jordan blocks of A . Then, there exist vectors v_1, \dots, v_m and integers $d_1 \geq \dots \geq d_m \geq 1$, such that

$$\mathbb{C}^n = \bigoplus_{j=1}^m \mathcal{K}_{d_j}(A, v_j),$$

where $\mathcal{K}_{d_j}(A, v_j)$ is a cyclic subspace and a Krylov subspace; see Section 2.10. For a proof, see the book [640].

If A has only simple or semi-simple eigenvalues or equivalently if there exists a basis of \mathbb{C}^n consisting of eigenvectors, the matrix is *diagonalizable*. There exists a nonsingular matrix X such that

$$X^{-1}AX = \Lambda, \quad (2.12)$$

where Λ is a diagonal matrix whose diagonal entries are the eigenvalues λ_i of A . The factorization $A = X\Lambda X^{-1}$ is called the *spectral factorization* of A . If A is not diagonalizable, it is said to be *defective* and one has to use the Jordan canonical form.

A matrix A is *normal* if $A^*A = AA^*$. It is known that A is normal if and only if it can be diagonalized by a unitary matrix. In other words, the matrix X in (2.12) can be chosen to be unitary and the spectral factorization of A is $A = X\Lambda X^*$. Examples of normal matrices are Hermitian, symmetric and skew-symmetric matrices.

2.4 Minimal polynomials

For the material in this section we refer to [531, Section 3.3]. The Cayley–Hamilton theorem says that the matrix A satisfies its characteristic polynomial, that is, $p_C(A) = 0$. The *minimal polynomial* p_A of A is the monic polynomial of smallest degree such that $p_A(A) = 0$. It can be shown that p_A divides the characteristic polynomial p_C . The *minimal polynomial of a vector v with respect to A* , denoted as $p_{A,v}$, is the monic polynomial of smallest degree such that $p_{A,v}(A)v = 0$. Its degree $d = d(A, v)$ is called the *grade of v with respect to A* . It can be shown that $p_{A,v}$ divides p_A .

If A has ℓ distinct eigenvalues $\lambda_1, \dots, \lambda_\ell$ there exist ℓ Jordan boxes in the Jordan canonical form of A , and the minimal polynomial is

$$p_A(\lambda) = \prod_{i=1}^{\ell} (\lambda - \lambda_i)^{l_i},$$

where l_i are the indices of the eigenvalues; see, for instance, [396]. The *index* l_i of λ_i is the order of the largest Jordan block associated with λ_i . If A is diagonalizable with ℓ distinct eigenvalues, then $p_A(\lambda) = \prod_{i=1}^{\ell} (\lambda - \lambda_i)$, and the degree of $p_{A,v}$ is smaller than or equal to ℓ for all vectors v .

The minimal polynomial of v with respect to A can be expressed using the Jordan canonical form of A . Let the vector $c = X^{-1}v$ be partitioned according to the orders of the Jordan boxes and blocks,

$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_\ell \end{pmatrix}, \quad c_i = \begin{pmatrix} c_{i,1} \\ \vdots \\ c_{i,g_i} \end{pmatrix}, \quad i = 1, \dots, \ell.$$

Then, from [923], we have

$$p_{A,v}(\lambda) = \prod_{i=1}^{\ell} (\lambda - \lambda_i)^{\tilde{l}_i},$$

with

$$\tilde{l}_i = \max_{j=1, \dots, g_i} \xi_{i,j}, \quad \xi_{i,j} = \max_{k=1, \dots, d_{i,j}} \{k \mid [c_{i,j}]_k \neq 0\},$$

where $d_{i,j}$ is the order of the Jordan block $J^{(i,j)}$. The grade of v with respect to A is $d = \tilde{l}_1 + \dots + \tilde{l}_p$.

A matrix A for which the characteristic and the minimal polynomial are equal (up to a multiplying factor ± 1) is said to be *nondiagonalizable*. In the Jordan canonical form there is only one Jordan block for each distinct eigenvalue or equivalently the eigenvalues are of geometric multiplicity one.

2.5 Singular values

The *singular values* σ_i of a rectangular matrix $A \in \mathbb{C}^{n \times m}$ are the square roots of the nonzero eigenvalues of A^*A . It can be shown that these are also the nonzero eigenvalues of AA^* . If the rank of A is r , there are r nonzero eigenvalues of A^*A and of AA^* and thus r singular values which are usually ordered according to $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$.

The *right singular vectors* are the eigenvectors of A^*A . Let us collect them in the columns of a unitary matrix $V \in \mathbb{C}^{m \times m}$. The *left singular vectors* are the eigenvectors of AA^* . Let us collect them in the columns of a unitary matrix $W \in \mathbb{C}^{n \times n}$. Then the matrix A can be factorized into its *singular value decomposition*

$$A = W\Sigma V^*,$$

where $\Sigma \in \mathbb{R}^{n \times m}$ is zero except for its leading principal submatrix of size r which is diagonal and contains the singular values σ_i in descending order.

It can be shown that the ℓ_2 matrix norm of A is the square root of the spectral radius of A^*A . Hence, this is precisely the largest singular value, i.e., $\|A\| = \sigma_1$. Sometimes we will also use the notation σ_{\max} and σ_{\min} for the largest and smallest singular value, respectively. The ℓ_2 condition number of A is therefore $\sigma_{\max}(A)/\sigma_{\min}(A)$. This is also taken as the definition of the condition number when A is a rectangular matrix.

2.6 The companion matrix

Let p be a monic polynomial of degree n ,

$$p(\lambda) = \lambda^n + \alpha_{n-1}\lambda^{n-1} + \dots + \alpha_1\lambda + \alpha_0.$$

Associated with p is the *companion matrix*,

$$C = \begin{pmatrix} 0 & 0 & \dots & 0 & -\alpha_0 \\ 1 & 0 & \dots & 0 & -\alpha_1 \\ 0 & 1 & \dots & 0 & -\alpha_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 1 & -\alpha_{n-1} \end{pmatrix}. \quad (2.13)$$

The matrix C is square of order n and nonderogatory. By computing the determinant of $\lambda I - C$ (that is, the characteristic polynomial of C) which is equal to $p(\lambda)$ we see that the roots of p are the eigenvalues of C . Nowadays, computing the eigenvalues of the companion matrix is a standard way of computing roots of polynomials from the coefficients; see [294]. The companion matrix associated to a matrix A is such that the polynomial p defined above is the characteristic polynomial of A , that is, its roots are the eigenvalues of A .

Let $\hat{\alpha} = (\alpha_1 \cdots \alpha_{n-1})^T$. If $\alpha_0 \neq 0$, the inverse of C is

$$C^{-1} = \begin{pmatrix} -\hat{\alpha}/\alpha_0 & I_{n-1} \\ -1/\alpha_0 & 0 \end{pmatrix}. \quad (2.14)$$

Only the first column of the inverse depends on the coefficients of the polynomial p . Properties of companion matrices were studied in [120, 121]; see also [187, 399, 400]. The Jordan canonical form of a companion matrix is considered in [80]. The singular values and vectors are described by the following result; see also [598].

Theorem 2.1 *The companion matrix C in (2.13) has two (not necessarily distinct) singular values σ_{\pm} satisfying*

$$\sigma_{\pm} = \frac{1}{\sqrt{2}} \left(\|\alpha\|^2 + 1 \pm ((\|\alpha\|^2 + 1)^2 - 4|\alpha_0|^2)^{\frac{1}{2}} \right)^{\frac{1}{2}},$$

where $\alpha = (\alpha_0 \ \alpha_1 \ \cdots \ \alpha_{n-1})^T$. The right singular vector corresponding to σ_{\pm} has the form

$$\gamma \begin{pmatrix} \frac{\hat{\alpha}}{1-\sigma_{\pm}^2} \\ 1 \end{pmatrix}, \quad \gamma = \left(\frac{\|\hat{\alpha}\|^2}{(1-\sigma_{\pm}^2)^2} + 1 \right)^{-\frac{1}{2}},$$

where $\hat{\alpha} = (\alpha_1 \cdots \alpha_{n-1})^T$. The left singular vector corresponding to σ_{\pm} has the form

$$\delta \begin{pmatrix} 1 \\ \frac{\sigma_{\pm}^2 \hat{\alpha}}{\alpha_0(\sigma_{\pm}^2 - 1)} \end{pmatrix}, \quad \delta = \left(\frac{\|\hat{\alpha}\|^2 \sigma_{\pm}^4}{\alpha_0^2 (1 - \sigma_{\pm}^2)^2} + 1 \right)^{-\frac{1}{2}}.$$

All the other singular values have the value one with right singular vectors of the form $(v^T \ 0)^T$ and left singular vectors of the form $(0 \ v^T)^T$ for orthonormal basis vectors v for the subspace orthogonal to $\hat{\alpha}$.

Proof The proof is obtained by computing C^*C and its eigenvalues. □

For more on singular values of companion matrices, see [593]. We also observe that one can use the singular values of C to obtain bounds for the modulus of the roots

of the associated polynomial; see [7, 598–600]. It yields bounds for the modulus of the eigenvalues of a square matrix if we consider its characteristic polynomial.

We have a relation between the matrix A , the companion matrix C associated with the eigenvalues of A and the matrix K defined as

$$K = (b \ A b \ A^2 b \cdots A^{n-1} b),$$

for a given vector b , namely,

$$AK = KC. \quad (2.15)$$

This relation is obvious for columns 1 to $n - 1$. The equality of the last columns is a straightforward consequence of the Cayley–Hamilton theorem since the last column of C contains the negatives of the coefficients of the characteristic polynomial of A . If the matrix K is nonsingular, $A = KCK^{-1}$ is similar to its companion matrix C and nonderogatory; such a matrix is called *cyclic*. Not all square matrices are similar to a companion matrix, but all of them are similar to a block diagonal matrix whose diagonal blocks are companion matrices. This is called the *Frobenius (or rational canonical) form* of the matrix.

2.7 Hessenberg matrices

An upper Hessenberg matrix H is such that $h_{i,j} = 0$ for $i > 2$, $j = 1, \dots, i - 2$, that is, only the entries of the upper triangular part of H are (possibly) nonzero as well as the entries on the first subdiagonal. A lower Hessenberg matrix is a transposed upper Hessenberg matrix. We have already seen an example of an upper Hessenberg matrix when we considered the companion matrix of a polynomial.

The matrix H of order n is *unreduced* (or *irreducible*) if $h_{j+1,j} \neq 0$, $j = 1, \dots, n - 1$, that is, the subdiagonal entries are nonzero. It implies that the matrix H is nonderogatory; see, for instance, [324] for a simple proof.

We will often be interested in solving linear systems with unreduced upper Hessenberg matrices. This can, of course, be done in several ways. We next consider the use of an LU factorization for a permuted unreduced upper Hessenberg matrix.

Let P be the permutation matrix of order n ,

$$P = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

We multiply H from the left with the matrix P . The permuted matrix is

$$PH = \begin{pmatrix} h_{2,1} & h_{2,2} & \cdots & h_{2,n} \\ & h_{3,2} & \ddots & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{n,n-1} & h_{n,n} \\ h_{1,1} & h_{1,2} & \cdots & h_{1,n-1} & h_{1,n} \end{pmatrix} = \begin{pmatrix} \hat{H}_{n-1} & w^{(n-1)} \\ (\ell^{(n-1)})^* & h_{1,n} \end{pmatrix}, \quad (2.16)$$

where the last equation merely defines the notation used in the following. We observe that \hat{H}_{n-1} is upper triangular.

Lemma 2.1 *An LU factorization of PH is given by the triangular matrices written in block form as*

$$L = \begin{pmatrix} I & 0 \\ (\ell^{(n-1)})^* & 1 \end{pmatrix}, \quad U = \begin{pmatrix} \hat{H}_{n-1} & w^{(n-1)} \\ 0 & \alpha \end{pmatrix},$$

where $\ell^{(n-1)} = \hat{H}_{n-1}^{-*} h^{(n-1)}$ and $\alpha = h_{1,n} - (w^{(n-1)}, \ell^{(n-1)})$.

Proof We remark that this factorization is in fact the result of partial pivoting for the first column (if $h_{1,1} < h_{2,1}$). The upper triangular matrix \hat{H}_{n-1} is nonsingular because H is unreduced. From the structure of $PH = LU$, we see that there is no fill-in in the lower triangular factor L . So, we look for L of the form

$$L = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ \ell_1 & \ell_2 & \cdots & \ell_{n-1} & 1 \end{pmatrix}.$$

We write L and U in block form as

$$L = \begin{pmatrix} I & 0 \\ (\ell^{(n-1)})^* & 1 \end{pmatrix}, \quad U = \begin{pmatrix} \hat{U}_{n-1} & u^{(n-1)} \\ 0 & \alpha \end{pmatrix},$$

where \hat{U}_{n-1} of order $n - 1$ is upper triangular. Then, the product LU is

$$PH = \begin{pmatrix} \hat{U}_{n-1} & u^{(n-1)} \\ (\ell^{(n-1)})^* \hat{U}_{n-1} & (\ell^{(n-1)})^* u^{(n-1)} + \alpha \end{pmatrix}.$$

This shows that $\hat{U}_{n-1} = \hat{H}_{n-1}$ and that the vector $u^{(n-1)}$ is equal to the first $n - 1$ elements of the last column of H , that is, $w^{(n-1)}$. Hence $u_{i,j} = h_{i+1,j}$, $i = 1, \dots, n - 1$, $j = 1, \dots, n$. It remains to compute $\ell^{(n-1)}$ and α . By identification

$$(\ell^{(n-1)})^* \hat{U}_{n-1} = (\ell^{(n-1)})^* \hat{H}_{n-1} = (h^{(n-1)})^*.$$

The vector $\ell^{(n-1)}$ is found by solving a lower triangular linear system

$$\hat{H}_{n-1}^* \ell^{(n-1)} = h^{(n-1)}.$$

This gives $\ell^{(n-1)} = \hat{H}_{n-1}^{-*} h^{(n-1)}$. Finally, we have $(\ell^{(n-1)})^* u^{(n-1)} + \alpha = h_{1,n}$ and using the expression of $\ell^{(n-1)}$, we obtain

$$\alpha = h_{1,n} - (\ell^{(n-1)})^* u^{(n-1)} = h_{1,n} - (\hat{H}_{n-1}^{-1} w^{(n-1)}, h^{(n-1)}).$$

Note that the scalar α is a Schur complement. \square

We must not have problems when computing this factorization since we have to solve one triangular system with \hat{H}_{n-1}^* , and this solve is backward stable (this notion will be explained in Section 2.11) as shown in [521], Chapter 8. The only problem could be severe cancellations in the computation of α , but the dot product could be computed using compensated summation; see [521], pages 83–88. This method for solving the linear system is related to what is known as Hyman’s method for computing the determinant of an upper Hessenberg matrix; see [521], page 280. The determinant of PH is $(-1)^{n-1}$ times the determinant of H . In fact,

$$\det(H) = (-1)^{n-1} (h_{1,n} - (\hat{H}_{n-1}^{-1} w^{(n-1)}, h^{(n-1)})) \det(\hat{H}_{n-1}).$$

This is obtained from the determinant of U which is $\alpha \det(\hat{U}_{n-1})$. Therefore,

$$\det(H) = (-1)^{n-1} (h_{1,n} - (\hat{H}_{n-1}^{-1} w^{(n-1)}, h^{(n-1)})) \prod_{j=1}^{n-1} h_{j+1,j}.$$

It is shown in [521] that this is a very stable method to compute the determinant of an Hessenberg matrix. It is also well known that the determinants of the leading principal submatrices of H satisfy a recurrence starting with $\det(H_0) = h_{1,1}$,

$$\det(H_k) = \sum_{j=1}^k (-1)^{j-1} \left(\prod_{i=1}^{j-1} h_{k-i+1,k-i} \right) h_{k-j+1,k} \det(H_{k-j}).$$

To solve a linear system with H we could have tried to compute directly an LU factorization of H since the matrix L has also a very simple bidiagonal nonzero structure but LU factorization without pivoting could be unstable. However, partial pivoting is quite simple for upper Hessenberg matrices and does not change the structure. At step k it is enough to look at rows k and $k+1$ since all the other potential pivots are zero. So, we may just have to interchange these two rows. We will see later that, usually, it is preferred to use unitary transformations to bring H to upper triangular form since they are guaranteed to be stable. They are more expensive, but this is not an important issue in the situations we treat later.

In the next chapters, the most frequently solved Hessenberg linear systems are of the form $Hz = \beta e_1$, i.e., they compute a multiple of the first column of H^{-1} . The columns of H^{-1} are as follows using some submatrices of H .

Theorem 2.2 *The first column of the inverse of H is, with the notation introduced in Lemma 2.1,*

$$H^{-1} e_1 = \frac{1}{\alpha} \begin{pmatrix} -\hat{H}_{n-1}^{-1} w^{(n-1)} \\ 1 \end{pmatrix}.$$

The other columns $H^{-1} e_j$, for $j = 2, \dots, n$, have the entries

$$\begin{pmatrix} e_1^T H^{-1} e_j \\ \vdots \\ e_{n-1}^T H^{-1} e_j \end{pmatrix} = \hat{H}_{n-1}^{-1} e_{j-1} - \frac{e_{j-1}^T \ell^{(n-1)}}{\alpha} \hat{H}_{n-1}^{-1} w^{(n-1)},$$

$$e_n^T H^{-1} e_j = -\frac{e_{j-1}^T \ell^{(n-1)}}{\alpha}.$$

Proof For the first column of the inverse of H , multiplying by the permutation matrix P , we obtain

$$PHz = LUz = Pe_1 = e_n.$$

We first solve $Ly = e_n$, which yields $y = e_n$. Then, $Uz = y = e_n$ has the solution

$$z = H^{-1} e_1 = U^{-1} e_n.$$

Hence, the first column of the inverse of H_k is given by the last column of the inverse of U . From the results above, we obtain the solution of the linear system,

$$\begin{pmatrix} z_1 \\ \vdots \\ z_{n-1} \end{pmatrix} = -\frac{1}{\alpha} \hat{H}_{n-1}^{-1} w^{(n-1)}, \quad z_n = \frac{1}{\alpha},$$

which gives, using (2.1), the first column of the inverse of H .

The other columns of the inverse of H can be computed by solving the corresponding permuted linear systems. Since $Pe_j = e_{j-1}$, $j = 2, \dots, n$ for the j th column of the inverse, we first solve $Ly = e_{j-1}$ which yields

$$y = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ -\ell_{j-1} \end{pmatrix},$$

the 1 being the $j - 1$ st component. If by an abuse of notation we denote again by e_{j-1} the vector of the first $n - 1$ components of y , the j th column of H^{-1} is given by the second claim. \square

Expressions for all the entries of H^{-1} are derived in [664–666]. They involve the determinants of principal leading and trailing submatrices of H . We will see another characterization of H^{-1} in Theorem 2.4.

Explicit relations between eigenvalues, entries of the eigenvectors and matrix entries of unreduced Hessenberg matrices are derived in [1019]; see also [325]. The Jordan canonical form of H is studied in [749]. Normal Hessenberg matrices and their relations to moment matrices are considered in [750].

Note that multiplying an upper Hessenberg matrix from the left or from the right by an upper triangular matrix yields an upper Hessenberg matrix. A factorization of an unreduced upper Hessenberg matrix that will be important in this book is the following.

Theorem 2.3 *An unreduced upper Hessenberg matrix H of order n can be factored as*

$$H = UCU^{-1}, \quad (2.17)$$

where U is a nonsingular upper triangular matrix of order n with $u_{1,1} = 1$ and diagonal entries $u_{k,k} = \prod_{j=1}^{k-1} h_{j+1,j}$ for $k > 1$ and C is the companion matrix of H .

Proof Define $U = (e_1 \ He_1 \ H^2 e_1 \ \dots \ H^{n-1} e_1)$. Then, U is upper triangular and nonsingular since H is unreduced and $u_{k,k} = \prod_{j=1}^{k-1} h_{j+1,j}$. Moreover, we have $HU = UC$ because of the Cayley–Hamilton theorem, see relation (2.15). Therefore, $H = UCU^{-1}$. \square

We will sometimes, for lack of a better name, refer to the decomposition (2.17) as the *triangular Hessenberg decomposition*. We remark that the decomposition of H is unique: The companion matrix C is uniquely determined by its characteristic polynomial and with $Ue_1 = e_1$ the other columns of U follow from successively equating columns $1, \dots, n - 1$ in $HU = UC$, yielding $U = (e_1 \ He_1 \ H^2 e_1 \ \dots \ H^{n-1} e_1)$. The result of Theorem 2.3 is mentioned in [749] without a proof. The author referred to chapter 6 of [535] and chapter 5 of [982].

Inverses of unreduced upper Hessenberg matrices, which are dense matrices, possess interesting properties. In [540] and [340] it was proved that the entries of the lower triangular part of H^{-1} are the product of components of two vectors say x and y , $(H^{-1})_{i,j} = x_j y_i$; see also [38, 72]. In other words, the lower triangular part of H^{-1} is the lower triangular part of a rank-one matrix. An algorithm for computing the vectors x and y is given in [540]. This result about H^{-1} can be proved from the factorization (2.17) as follows.

Theorem 2.4 Let $p(\lambda) = \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1\lambda + \alpha_0$ be the characteristic polynomial of the unreduced Hessenberg matrix H factored as $H = UCU^{-1}$ as in Theorem 2.3. We define $\beta_1 = -\alpha_1/\alpha_0$,

$$\hat{\beta} = -(-\alpha_2/\alpha_0 \cdots \alpha_{n-1}/\alpha_0 \ 1/\alpha_0)^T,$$

and we partition the upper triangular matrix U^{-1} as

$$U^{-1} = \begin{pmatrix} 1 & \hat{\beta}^T \\ 0 & \hat{U}^{-1} \end{pmatrix}.$$

Then, the inverse of H can be written as

$$H^{-1} = \begin{pmatrix} \beta_1 - \hat{\beta}^T \hat{U} \hat{\beta} & (\beta_1 - \hat{\beta}^T \hat{U} \hat{\beta}) \hat{\beta}^T + (e_1^T - \hat{\beta}^T \hat{U} F) \hat{U}^{-1} \\ \hat{U} \hat{\beta} & \hat{U} \hat{\beta} \hat{\beta}^T + \hat{U} F \hat{U}^{-1} \end{pmatrix}, \quad (2.18)$$

where F is the zero matrix except for the entries on the first upper diagonal which are equal to 1.

Proof We have obviously $H^{-1} = UC^{-1}U^{-1}$. We have seen in Section 2.6, relation (2.14), what is the inverse of C . With our notation it can be written as

$$C^{-1} = \begin{pmatrix} \beta_1 & e_1^T \\ \hat{\beta} & F \end{pmatrix}.$$

The inverse of the matrix U^{-1} is

$$U = \begin{pmatrix} 1 & -\hat{\beta}^T \hat{U} \\ 0 & \hat{U} \end{pmatrix}.$$

We obtain the result by the multiplication $UC^{-1}U^{-1}$. □

The matrix $\hat{U}F\hat{U}^{-1}$ is strictly upper triangular which implies that the lower triangular part of the principal trailing block of H^{-1} is the lower triangular part of $\hat{U}\hat{\beta}\hat{\beta}^T$. Hence, we easily see that the lower triangular part of H^{-1} is the lower triangular part of a rank-one matrix. We should also mention that we partitioned the matrix U^{-1} in this way because we will see in the next chapters that the vector $\hat{\beta}$ is related to the residual norms of Q-OR Krylov methods.

Let us denote

$$C = \begin{pmatrix} 0 & -\alpha_0[e^{(n-1)}]^T \\ e_1 & \hat{C} \end{pmatrix}, \quad z^T = -\alpha_0[e^{(n-1)}]^T - \hat{\beta}^T \hat{U} \hat{C},$$

with $[e^{(n-1)}]^T = (0 \cdots 0 \ 1)$ with $n-1$ components. Then, the matrix H can be written in block form as

$$H = \begin{pmatrix} -\hat{\theta}^T \hat{U} e_1 & -(\hat{\theta}^T \hat{U} e_1) \hat{\theta}^T + z^T \hat{U}^{-1} \\ \hat{U} e_1 & (\hat{U} e_1) \hat{\theta}^T + \hat{U} \hat{C} \hat{U}^{-1} \end{pmatrix}.$$

Note that $\hat{U} e_1 = u_{2,2} e_1 = h_{2,1} e_1$, only the first row of $(\hat{U} e_1) \hat{\theta}^T$ is nonzero and $\hat{U} \hat{C} \hat{U}^{-1}$ is upper Hessenberg. This shows how the entries of H are related to the coefficients of the characteristic polynomial.

From the factorization (2.17) of H we can also obtain the factorizations of its leading principal submatrices H_k .

Theorem 2.5 *Let $H = UCU^{-1}$ where U is nonsingular with $u_{1,1} = 1$ and upper triangular and C is a companion matrix. For $k < n$ the principal submatrix H_k can be written as $H_k = U_k C^{(k)} U_k^{-1}$, U_k being the leading principal submatrix of order k of U and the companion matrix of H_k is $C^{(k)} = E_k + (0 \ U_k^{-1} U_{1:k,k+1})$ where E_k is a square down-shift matrix of order k ,*

$$E_k = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}.$$

Proof Clearly

$$H_k = (I_k \ 0) H \begin{pmatrix} I_k \\ 0 \end{pmatrix} = (I_k \ 0) U C U^{-1} \begin{pmatrix} I_k \\ 0 \end{pmatrix},$$

where I_k is the identity matrix of order k . Let

$$(I_k \ 0) U = (U_k \ Z), \quad U^{-1} \begin{pmatrix} I_k \\ 0 \end{pmatrix} = \begin{pmatrix} U_k^{-1} \\ 0 \end{pmatrix}.$$

Since $C = E_n + (0 \cdots 0 \ v)$ where v is a given vector, we have

$$H_k = (U_k \ Z) \begin{pmatrix} E_k U_k^{-1} \\ e_1 e_k^T U_k^{-1} \end{pmatrix} = U_k E_k U_k^{-1} + Z e_1 e_k^T U_k^{-1}.$$

The vector $Z e_1$ of length k is made of the k first components of the $k+1$ st column of U , that is, $U_{1:k,k+1}$. Factoring U_k on the left and U_k^{-1} on the right gives the result. \square

Corollary 2.1 *The coefficients of the characteristic polynomial $\lambda^k + \alpha_{k-1}^{(k)} \lambda^{k-1} + \cdots + \alpha_0^{(k)}$ of H_k are given by*

$$\begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} = u_{k+1,k+1} [U_{k+1}^{-1}]_{1:k,k+1}. \quad (2.19)$$

Proof From the form of $C^{(k)}$ in Theorem 2.5 we see that the coefficients of the characteristic polynomial $\lambda^k + \alpha_{k-1}^{(k)}\lambda^{k-1} + \cdots + \alpha_0^{(k)}$ of H_k are

$$\begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} = -U_k^{-1} U_{1:k,k+1} = u_{k+1,k+1} [U_{k+1}^{-1}]_{1:k,k+1},$$

where the last equality follows from the formula for the inverse of U_{k+1} , see (2.1). \square

The corollary shows that the coefficients of the characteristic polynomial of H_k are not only, obviously, given in the last column of $C^{(k)}$, they also move, in the decomposition $U_{k+1} C^{(k+1)} U_{k+1}^{-1}$ for the next Hessenberg matrix H_{k+1} , to the last column of U_{k+1}^{-1} (when multiplied with $u_{k+1,k+1}$). Each column of U^{-1} contains, up to a scaling factor, the coefficients of the characteristic polynomial of a submatrix H_k . Therefore, when we know H , we can construct the matrix U^{-1} by computing the coefficients of the characteristic polynomials of the matrices H_k from their eigenvalues.

We will sometimes need as well the following relation for the coefficients of the characteristic polynomial of H_k .

Corollary 2.2 Let $M_j = U_j^* U_j$, $j = 1, \dots, k+1$ where U_j was defined in Theorem 2.5. The vector of the coefficients of the characteristic polynomial of H_k denoted as $(\alpha_0^{(k)} \dots \alpha_{k-1}^{(k)})^T$ is the solution of the linear system,

$$M_k \begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} = -M_{1:k,k+1}. \quad (2.20)$$

Proof The proof is straightforward. We have

$$U_k \begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} = -U_{1:k,k+1}.$$

Multiplying by U_k^* we obtain

$$M_k \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{k-1} \end{pmatrix} = -U_k^* U_{1:k,k+1}.$$

Clearly $U_k^* U_{1:k,k+1} = M_{1:k,k+1}$. \square

We also have a recurrence formula for the entries on the first row of the inverses U_k^{-1} using the entries of H that will be useful later.

Lemma 2.2 *The entries $\vartheta_{1,j}$ of the first row of the inverse of U in $H = UCU^{-1}$ are related to the entries $h_{i,j}$ of H by the relation*

$$\vartheta_{1,k+1} = -\frac{1}{h_{k+1,k}} \sum_{j=1}^k \vartheta_{1,j} h_{j,k}, \quad k = 1, \dots, n-1. \quad (2.21)$$

Proof We have seen that $H = UCU^{-1}$ where C is the companion matrix defined in (2.13). Multiplying on the left by U^{-1} , we have $U^{-1}H = CU^{-1}$. From the structure of C and the fact that U^{-1} is upper triangular, the entries of the first row of CU^{-1} are zero except for the last one in position $(1, n)$. Writing the entry $(1, k)$ of $U^{-1}H$ for $k < n$, we obtain

$$\sum_{j=1}^{k+1} \vartheta_{1,j} h_{j,k} = 0 \Rightarrow \vartheta_{1,k+1} = -\frac{1}{h_{k+1,k}} \sum_{j=1}^k \vartheta_{1,j} h_{j,k},$$

which proves the claim. \square

We note that this formula can also be obtained straightforwardly from [712], formula (4.4), page 807. Another formula can be obtained for the entries of the first row of the matrices U_k^{-1} .

Lemma 2.3 *The entries $\vartheta_{1,j}$ of the first row of the inverse of U in $H = UCU^{-1}$ are related to the entries $h_{i,j}$ of H by the relation*

$$\vartheta_{1,k+1} = (-1)^k \frac{\det(H_k)}{\prod_{j=1}^k h_{j+1,j}}, \quad k = 1, \dots, n-1. \quad (2.22)$$

Proof From Corollary 2.1 we have

$$\vartheta_{1,k+1} = \frac{\alpha_0^{(k)}}{u_{k+1,k+1}}.$$

Since $\alpha_0^{(k)}$ is the constant term of the characteristic polynomial of $C^{(k)}$ we have

$$\alpha_0^{(k)} = (-1)^k \det(C^{(k)}) = (-1)^k \det(H_k).$$

Moreover, $u_{k+1,k+1} = \prod_{j=1}^k h_{j+1,j}$. \square

The relation (2.22) can also be proved by induction from (2.21). The interest of the recurrence relation (2.21) is that only one column of H is involved in the

computation of $\vartheta_{1,k+1}$. We observe that the last two results yield another formula for the determinant of H_k ,

$$\det(H_k) = (-1)^{k-1} \prod_{j=1}^{k-1} h_{j+1,j} \sum_{j=1}^k \vartheta_{1,j} h_{j,k}.$$

Corollary 2.3 *Using the notation of Lemma 2.2, the matrix H_k is singular if and only if $\vartheta_{1,k+1} = 0$.*

Proof This is straightforward from Lemma 2.3. □

For completeness we remark that

$$\vartheta_{1,k+1} = -\frac{1}{u_{k+1,k+1}} \sum_{j=1}^k \vartheta_{1,j} u_{j,k+1}.$$

This relation is proved by considering the first row of the equality

$$U_{k+1}^{-1} = \begin{pmatrix} U_k^{-1} & \frac{1}{u_{k+1,k+1}} U_k^{-1} U_{1:k,k+1} \\ & \frac{1}{u_{k+1,k+1}} \end{pmatrix}.$$

2.8 Tridiagonal matrices

A symmetric upper (or lower) Hessenberg matrix is tridiagonal. In this section we consider the inverse of a real nonsingular symmetric tridiagonal matrix of order k ,

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \ddots & & \\ & & & & \alpha_{k-1} & \beta_k \\ & & & & \beta_k & \alpha_k \end{pmatrix}, \quad (2.23)$$

where the values β_j , $j = 2, \dots, k$ are assumed to be nonzero. From [70] or [671] and the references therein, it is known that there exist two sequences of nonzero numbers $\{u_i\}$, $\{v_i\}$, $i = 1, \dots, k$ such that

$$T_k^{-1} = \begin{pmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 & \dots & u_1 v_k \\ u_1 v_2 & u_2 v_2 & u_2 v_3 & \dots & u_2 v_k \\ u_1 v_3 & u_2 v_3 & u_3 v_3 & \dots & u_3 v_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_1 v_k & u_2 v_k & u_3 v_k & \dots & u_k v_k \end{pmatrix}. \quad (2.24)$$

Moreover, u_1 can be chosen arbitrarily. Please note that, in this section, u_i and v_i , $i = 1, \dots, k$ are real numbers. For more details on inverses of tridiagonal matrices, in particular the computation of the u_i 's and v_i 's, see [432]. Here, we are interested in recovering the coefficients α_i and β_i from the u_i 's and v_i 's.

Proposition 2.1 *Knowing the nonzero values u_i, v_i , $i = 1, \dots, k$ in (2.24), the nonzero coefficients of the matrix T_k defined in (2.23) are given by*

$$\alpha_i = -\frac{u_{i+1}}{d_i} - \beta_i \frac{u_{i-1}}{u_i}, \quad \beta_{i+1} = \frac{u_i}{d_i}, \quad i = 1, \dots, k-1,$$

with $d_i = u_i(u_i v_{i+1} - u_{i+1} v_i)$ and assuming $\beta_1 = 0$. The last diagonal coefficient is

$$\alpha_k = \frac{1 - u_{k-1} v_k \beta_k}{u_k v_k}.$$

Proof Considering the i th row and columns i and $i + 1$ of the relation $T_k T_k^{-1} = I$ we obtain the two equations

$$\begin{aligned} u_{i-1} v_i \beta_i + u_i v_i \alpha_i + u_i v_{i+1} \beta_{i+1} &= 1, \\ u_{i-1} \beta_i + u_i \alpha_i + u_{i+1} \beta_{i+1} &= 0. \end{aligned}$$

It yields the linear system,

$$\begin{pmatrix} u_i v_i & u_i v_{i+1} \\ u_i & u_{i+1} \end{pmatrix} \begin{pmatrix} \alpha_i \\ \beta_{i+1} \end{pmatrix} = \begin{pmatrix} 1 - \beta_i u_{i-1} v_i \\ -\beta_i u_{i-1} \end{pmatrix}.$$

Using Cramer's rule we obtain the solution given above, d_i being the negative of the determinant of the matrix. For the last diagonal entry, we have

$$u_{k-1} v_k \beta_k + u_k v_k \alpha_k = 1,$$

which gives the result. \square

We will also be interested in the LU factorization of a nonsymmetric tridiagonal matrix T_k written as

$$T_k = \begin{pmatrix} \alpha_1 & \gamma_2 & & & \\ \beta_2 & \alpha_2 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_{k-1} & \gamma_k \\ & & & \beta_k & \alpha_k \end{pmatrix}. \quad (2.25)$$

We factorize this matrix as $T_k = L_k D_k^{-1} U_k$ where D_k is a diagonal matrix with δ_j , $j = 1, \dots, k$ on the diagonal and L_k, U_k are bidiagonal matrices

$$L_k = \begin{pmatrix} \delta_1 & & & \\ \beta_2 & \delta_2 & & \\ & \ddots & \ddots & \\ & & \beta_{k-1} & \delta_{k-1} \\ & & & \beta_k & \delta_k \end{pmatrix}, \quad U_k = \begin{pmatrix} \delta_1 & \gamma_2 & & \\ & \delta_2 & \gamma_3 & \\ & & \ddots & \ddots \\ & & & \delta_{k-1} & \gamma_k \\ & & & & \delta_k \end{pmatrix}.$$

By identification, we find

$$\delta_1 = \alpha_1, \quad \delta_j = \alpha_j - \frac{\beta_j \gamma_j}{\delta_{j-1}}, \quad j = 2, \dots, k.$$

Of course, since we are doing a factorization without pivoting, it can be done only if $\delta_j \neq 0$, $j = 1, \dots, k-1$. Even if the pivots are different from zero, the factorization can be numerically unstable.

2.9 Unitary transformations

In this section we are interested in constructing unitary transformations to annihilate one or several components of a given vector. This is very useful for the stable computation of a number of factorizations, e.g., to reduce Hessenberg matrices to triangular form. A first way to do this is to use Householder reflections [533]. A Householder matrix or transformation is a unitary matrix of the form

$$P = I - \beta v v^*, \quad (2.26)$$

where v is a vector and β is a scalar determined to zero some components of a vector x . If the matrix P is real, multiplication by P represents geometrically a reflection with respect to the hyperplane orthogonal to v . For the computation of these transformations see, for instance, [438], section 5.1 page 208 or [624].

A code inspired by [438] to zero the components m to n of a real vector x with n components is the following. The tests are done to prevent some possible cancellations.

```
function [v,beta,P] = householder(x,m);
%
% The result is x - beta (v' x) v
nx = length(x);
x = x(m-1:nx);
n = length(x);
sig = x(2:n)' * x(2:n);
v = [1; x(2:n)];
if sig == 0
    beta = 0;
else
```

```

mu = sqrt(x(1)^2 + sig);
if x(1) <= 0
    v(1) = x(1) - mu;
else
    v(1) = -sig/(x(1) + mu);
end % if x(1)
beta = 2 * v(1)^2 / (sig + v(1)^2);
v = v / v(1);
end % if sig
v = [zeros(m-2,1); v];
if nargout == 3
    P = speye(nx,nx) - beta * (v * v');
end % if nargout

```

In many cases the matrix P is not needed. To apply the transformation to a vector x only v and β are needed since $Px = x - \beta(v^*x)v$. The code above does not work for complex vectors x . But this can be done as described in [624]. A code to compute P such that $Px = \|x\|e_1$ when x is complex is the following.

```

function [v,beta,P] = housec(x);
%
nx = length(x);
xi = x(1);
alpha = norm(x);
if alpha == 0 || xi == alpha
    v = zeros(nx,1);
    beta = 0;
    P = eye(nx,nx);
    return
end % if alpha
v = (x - alpha * eye(nx,1)) / (xi - alpha);
beta = conj((alpha - xi) / alpha);
if nargout == 3
    P = eye(nx,nx) - beta * (v * v');
end % if nargout

```

If we want to zero the components m to n of x we could use the following function for which m has to be larger than 2 (otherwise we could use `housec`).

```

function [v,beta,P] = householderc(x,m);
%
nx = length(x);
[vv,beta] = housec(x(m-1:nx));
v = [zeros(m-2,1); vv];
if nargout == 3
    P = eye(nx,nx) - beta * (v * v');
end % if nargout

```

Zeroing components in a given vector can also be done with so-called Givens rotations [426]. This is done component by component. Assume that we have a vector with two components x, y and that we want to zero the second component y by multiplying the vector with a unitary 2×2 matrix. We define this unitary matrix (which is called a rotation matrix) as

$$\begin{pmatrix} c & \bar{s} \\ -s & c \end{pmatrix}.$$

If x, y are real, s, c can be chosen to be real as well and the matrix is orthogonal. In this case, s and c correspond to the sine and cosine of a rotation angle. Even if this is not the case in complex arithmetic, we will call s and c the sines and cosines of the Givens rotation.

To improve the stability of the computation s and c can be chosen in the following way. Let μ and τ be chosen as follows:

$$\text{if } |x| \leq |y|, \mu = \frac{x}{y}, \tau = \frac{\bar{\mu}}{|\mu|}, \text{ else } \mu = \frac{y}{x}, \tau = \frac{\mu}{|\mu|}. \quad (2.27)$$

Then,

$$c = \frac{|x|}{\sqrt{|x|^2 + |y|^2}}, \quad s = \frac{|y|\tau}{\sqrt{|x|^2 + |y|^2}}.$$

Note that $|\tau| = 1$ and c is real. This reduction is implemented in the following code:

```
function [c,s,rot] = givens(x,y);
%
if y == 0
    c = 1;
    s = 0;
elseif x == 0
    c = 0;
    s = 1;
else
    cs = sqrt(abs(y)^2 + abs(x)^2);
    if abs(x) < abs(y)
        mu = x / y;
        tau = conj(mu) / abs(mu);
    else
        mu = y / x;
        tau = mu / abs(mu);
    end % if abs
    c = abs(x) / cs; % cosine
    s = abs(y) * tau / cs; % sine
end % if y
%
if nargout == 3
```

```

rot = [c conj(s); -s c];
else
    rot = [];
end % if nargout

```

To zero the $(j+1)$ st component of a given vector v with n components, we multiply the vector v by the matrix

$$G_j = \begin{pmatrix} I_{j-1} & & \\ & \begin{pmatrix} c_j & \bar{s}_j \\ -s_j & c_j \end{pmatrix} & \\ & & I_{n-j-1} \end{pmatrix},$$

where s_j and c_j are computed as above with $x = v_j$ and $y = v_{j+1}$. The multiplication with G_j modifies only the components j and $j+1$ of the vector v . For the computation of Givens rotations, see [96, 624]. The stability of Householder reflections and Givens rotations is considered in [521].

Let H_k be a nonsingular upper Hessenberg matrix of order k with real and strictly positive subdiagonal entries. In Section 2.7 we have already seen a way of solving linear systems of the form

$$H_k y = \beta e_1, \quad (2.28)$$

where β is nonzero. We will now do so through reducing H_k to upper triangular form with unitary transformations. We will also be interested in solving a least squares problem

$$\min_y \|\beta e_1 - \underline{H}_k y\|, \quad \underline{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix}. \quad (2.29)$$

The matrix \underline{H}_k is of size $(k+1) \times k$.

To reduce H_k and \underline{H}_k to upper triangular form it is convenient to use Givens rotations. We have to zero the entries on the first subdiagonal of the upper Hessenberg matrix. Givens rotations will do that one entry at a time. Moreover, in the applications we have in mind, the matrix \underline{H}_k is constructed one column at a time and, when adding a new column, we would just have to compute one more rotation matrix. Our goal is to construct step by step a unitary matrix Q_k and an upper triangular matrix R_k such that $Q_k^* H_k = R_k$.

To zero the entry $h_{2,1}$ we multiply H_k from the left by a unitary matrix

$$G_1 = \begin{pmatrix} (c_1 \bar{s}_1) & 0 \\ (-s_1 c_1) & 0 \\ 0 & I_{k-2} \end{pmatrix},$$

where s_1 and c_1 (such that $c_1^2 + |s_1|^2 = 1$) are chosen as above to zero the $(2,1)$ entry of H_k . Multiplying by G_1 modifies the first two rows of H_k , and the leading principal submatrix of order 2 is now upper triangular. The first row will not anymore be modified in the next steps of the algorithm, and therefore it is the first row of R_k .

Let us assume that we have done $j - 1$ steps and we now want to zero the entry $h_{j+1,j}$. We multiply the matrix we have obtained so far by the unitary matrix

$$G_j = \begin{pmatrix} I_{j-1} & & \\ & \begin{pmatrix} c_j & \bar{s}_j \\ -s_j & c_j \end{pmatrix} & \\ & & I_{k-j-1} \end{pmatrix}.$$

The previous steps have given us the final values of rows 1 to $j - 1$ of the matrix R . Let $\tilde{r}_{j,j}$ be the current value of the diagonal entry in position (j, j) . The scalars s_j and c_j are computed as in (2.27) to zero the entry $h_{j+1,j}$:

$$\text{if } |\tilde{r}_{j,j}| \leq h_{j+1,j}, \mu = \frac{\tilde{r}_{j,j}}{h_{j+1,j}}, \tau = \frac{\bar{\mu}}{|\mu|}, \text{ else } \mu = \frac{h_{j+1,j}}{|\tilde{r}_{j,j}|}, \tau = \frac{\mu}{|\mu|}.$$

Then, the cosine and sine are

$$c_j = \frac{|\tilde{r}_{j,j}|}{\sqrt{|\tilde{r}_{j,j}|^2 + h_{j+1,j}^2}}, \quad s_j = \frac{|h_{j+1,j}| \tau}{\sqrt{|\tilde{r}_{j,j}|^2 + h_{j+1,j}^2}}. \quad (2.30)$$

We now have to compute the final values of the entries of row j of R_k . In particular, $r_{j,j} = c_j \tilde{r}_{j,j} + \bar{s}_j h_{j+1,j}$. If the matrix is real it yields

$$r_{j,j} = \text{sgn}(\tau) \sqrt{\tilde{r}_{j,j}^2 + h_{j+1,j}^2},$$

where sgn is the sign function. Applying $k - 1$ rotation matrices to the left of H_k we obtain a triangular matrix,

$$G_{k-1} \cdots G_1 H_k = Q_k^* H_k = R_k.$$

To compute the solution of $H_k y = \beta e_1$ we have to solve a triangular system,

$$R_k y = \beta q,$$

with $q = G_{k-1} \cdots G_1 e_1$. Hence, we just have to apply the rotation matrices to the right-hand side. At each step only two components of the right-hand side vector are modified. Of course, the matrices G_j are never formed; we just compute the sines and cosines defining the rotations and discard them as soon as the rotation has been applied. However, in the applications we are interested in this book the matrices H_k are constructed step by step, adding a new column at each step. In this case the sines and cosines of the rotations have to be stored since, when we add a new column, we have to apply all the previous rotations to it.

Now we consider the least squares problem (2.29). The last row of the matrix \underline{H}_k has only one nonzero entry in position $(k + 1, k)$. To reduce the matrix to upper triangular form the first $k - 1$ steps are the same as for H_k and the k th step is to apply

a rotation to zero the entry $h_{k+1,k}$. In the end we obtain

$$G_k G_{k-1} \cdots G_1 \underline{H}_k = Q_{k+1}^* \underline{H}_k = \begin{pmatrix} \hat{R}_k \\ 0 \cdots 0 \end{pmatrix}.$$

Note that this is an abuse of notation since the matrices G_j , $j = 1, \dots, k-1$ do not have the same dimension as when reducing H_k . But only the sines and cosines do matter and those are the same as before. The norm we want to minimize can be written as

$$\|\beta e_1 - \underline{H}_k y\| = \|Q_{k+1}^*(\beta e_1 - \underline{H}_k y)\| = \left\| \beta Q_{k+1}^* e_1 - \begin{pmatrix} \hat{R}_k \\ 0 \end{pmatrix} y \right\|.$$

Let $\hat{q} = Q_{k+1}^* e_1$. Then, the unique solution of the least squares problem is given by solving the upper triangular system,

$$\hat{R}_k y = \beta \hat{q}_{1:k}.$$

Inserting the solution y in the norm we find that the norm of the residual is

$$\|\beta e_1 - \underline{H}_k y\| = \beta |[\hat{q}]_{k+1}|.$$

Note that \hat{R}_k is the same as R_k except for the bottom right entry. For our purposes later on it is useful to compute the last component $[\hat{q}]_{k+1}$.

Proposition 2.2 *Using the notation above, the residual norm of the least squares problem (2.29) is*

$$\|\beta e_1 - \underline{H}_k y\| = \beta |s_1 \cdots s_k|. \quad (2.31)$$

The last component of the solution of equation (2.28) is

$$y_k = (-1)^{k-1} \beta \frac{s_1 \cdots s_{k-1}}{(R_k)_{k,k}}. \quad (2.32)$$

Proof Let us proceed by induction to compute the last entry of the first column of Q_{k+1}^* . We first consider the product of the leading principal submatrices of order 3 of G_2 and G_1 ,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_2 & \bar{s}_2 \\ 0 & -s_2 & c_2 \end{pmatrix} \begin{pmatrix} c_1 & \bar{s}_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_1 & \bar{s}_1 & 0 \\ -s_1 c_2 & c_1 c_2 & \bar{s}_2 \\ s_1 s_2 & -s_2 c_1 & c_2 \end{pmatrix}.$$

Hence, the entry we are interested in is $s_1 s_2$. Let us assume that the entry $(j, 1)$ (the last nonzero entry in the first column) in the product $G_{j-1} \cdots G_1$ is $(-1)^{j-1} s_1 \cdots s_{j-1}$. This gives the second claim. In the matrix G_j , the first nonzero entry in row $j+1$ is in position $(j+1, j)$ with a value $-s_j$. Hence, the entry $(j+1, 1)$ in $G_j \cdots G_1$ is $(-1)^j s_1 \cdots s_j$. This yields an expression for $|[\hat{q}]_{k+1}|$ and thus the first claim. \square

It is also interesting to relate the solutions of (2.28) and (2.29). To distinguish between the two problems, let us denote the solution of the least squares problem by $y^{(k)}$ where the index k refers to the dimension of the problem. We observe that q differs only from $\hat{q}_{1:k}$ by the last entry.

Proposition 2.3 *The vector y solution of (2.28) is given by*

$$y = \begin{pmatrix} y^{(k-1)} \\ 0 \end{pmatrix} + \beta[q]_k \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k}, \quad (2.33)$$

where $y^{(k-1)}$ is the solution of the least squares problem of order $k - 1$ and $[q]_k$ is the last component of q in $R_k y = \beta q$.

Proof To prove the claim we have to relate the inverses of the matrices R_k and \hat{R}_k . As we said above, they differ only by the bottom right entry (k, k) . Let $z^{(k-1)}$ be the vector of the $k - 1$ first components of the last column of \hat{R}_k which is therefore $(z^{(k-1)} \hat{r}_k)^T$. Using what we have seen at the end of Section 2.7 for the inverse of an upper triangular matrix, the last columns of \hat{R}_k^{-1} and R_k^{-1} are, respectively,

$$\frac{1}{\hat{r}_{k,k}} \begin{pmatrix} -\hat{R}_{k-1}^{-1} z^{(k-1)} \\ 1 \end{pmatrix}, \quad \frac{1}{r_{k,k}} \begin{pmatrix} -R_{k-1}^{-1} z^{(k-1)} \\ 1 \end{pmatrix}.$$

But $\hat{R}_{k-1}^{-1} = R_{k-1}^{-1}$. Therefore, the two last columns differ only by a multiplying factor,

$$[R_k^{-1}]_{:,k} = \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k}.$$

The right-hand side βq of the problem (2.28) and the k first components $\beta \hat{q}_{1:k}$ of the problem (2.29) differ only by the last component. Then,

$$\begin{aligned} y &= \beta \left(\begin{bmatrix} R_{k-1}^{-1} \\ 0 \end{bmatrix} [R_k^{-1}]_{:,k} \right) \begin{pmatrix} q_{1:k-1} \\ [q]_k \end{pmatrix}, \\ &= \beta \left(\begin{bmatrix} \hat{R}_{k-1}^{-1} \\ 0 \end{bmatrix} \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k} \right) \begin{pmatrix} \hat{q}_{1:k-1} \\ [q]_k \end{pmatrix}, \\ &= \begin{pmatrix} \beta \hat{R}_{k-1}^{-1} \hat{q}_{1:k-1} \\ 0 \end{pmatrix} + \beta[q]_k \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k}. \end{aligned}$$

The $k - 1$ first components of the first vector on the right-hand side is the solution of the least squares problem of order $k - 1$. \square

2.10 Krylov subspaces and their basic properties

Let A be a nonsingular square matrix of order n , b a vector with n components and $k \geq 1$ be an integer. The *Krylov subspace* $\mathcal{K}_k(A, b)$ is defined as

$$\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2b, \dots, A^{k-1}b\}. \quad (2.34)$$

Krylov methods are based on projection of the given problem onto Krylov subspaces of growing dimension. In the implicit construction of Krylov subspaces needed in these methods, we can exploit cheap multiplication of vectors with A , e.g., when A is sparse or when the matrix–vector product can be computed in a matrix-free way without storing A .

What is the dimension of a given Krylov subspace $\mathcal{K}_k(A, b)$? To answer this question we have to consider the minimal polynomial $p_{A,b}$ of b with respect to A that we have introduced in Section 2.4. Its degree $d = d(A, b)$ is the grade of b with respect to A . By definition we have $p_{A,b}(A)b = 0$. Let

$$p_{A,b}(\lambda) = \lambda^d + \gamma_{d-1}\lambda^{d-1} + \dots + \gamma_1\lambda + \gamma_0.$$

It yields

$$A^d b = - \sum_{j=0}^{d-1} \gamma_j A^j b.$$

Hence, if $A^d b \neq 0$, it is a linear combination of the vectors in $\mathcal{K}_d(A, b)$. Since $p_{A,b}$ is the monic polynomial of smallest degree satisfying $p_{A,b}(A)b = 0$, the dimension of the Krylov subspaces $\mathcal{K}_j(A, b)$, $j = 1, \dots, d$ is j , and the dimension of $\mathcal{K}_j(A, b)$ for $j > d$ is d . The grade of b with respect to A is the maximum subspace dimension in the nested sequence of Krylov subspaces

$$\mathcal{K}_1(A, b) \subset \dots \subset \mathcal{K}_j(A, b) \subset \dots \subset \mathcal{K}_d(A, b) = \mathcal{K}_{d+1}(A, b) = \dots$$

Note that d is smaller than or equal to the degree of the minimal polynomial of A which is smaller than or equal to the degree of the characteristic polynomial of A . If A is diagonalizable and has ℓ distinct eigenvalues, we have $d \leq \ell$.

Let us assume that $\gamma_0 \neq 0$. Since we have

$$A^d b + \gamma_{d-1} A^{d-1} b + \dots + \gamma_1 A b + \gamma_0 b = 0,$$

multiplying by A^{-1} and dividing by γ_0 yields

$$x = A^{-1} b = - \left\{ \frac{1}{\gamma_0} A^{d-1} b + \frac{\gamma_{d-1}}{\gamma_0} A^{d-2} b + \dots + \frac{\gamma_1}{\gamma_0} b \right\}.$$

The solution of the linear system $Ax = b$ belongs to a Krylov subspace $\mathcal{K}_d(A, b)$ where d is the grade of b with respect to A . Even though d might be large, it makes sense to consider Krylov subspaces to compute approximations of the solution x .

The idea is that through repeated multiplication with A , dominant properties might be discovered at an early stage of the iterative process.

Let

$$K_{n,k} = (b \ A b \ A^2 b \ \cdots \ A^{k-1} b), \ k = 1, \dots, n-1 \quad (2.35)$$

be the *Krylov matrices* corresponding to the Krylov subspaces (see also (2.15)). To obtain expressions for several residual norms in Krylov methods, it is useful to introduce the following factorization of Krylov matrices.

Lemma 2.4 *Let A be a diagonalizable matrix with $A = X\Lambda X^{-1}$ where the matrix Λ is diagonal and let $c = X^{-1}b$. Then, the Krylov matrices $K_{n,k}$ defined in (2.35) can be factorized as*

$$K_{n,k} = XD_c \mathcal{V}_{n,k}, \quad (2.36)$$

where D_c is a diagonal matrix whose diagonal entries are c_i , $i = 1, \dots, n$ and $\mathcal{V}_{n,k}$ is an $n \times k$ Vandermonde matrix constructed with the eigenvalues $\lambda_1, \dots, \lambda_n$ of A ; see (2.6).

Proof We have

$$\begin{aligned} K_{n,k} &= (b \ A b \ \cdots \ A^{k-1} b), \\ &= (XX^{-1}b \ X\Lambda X^{-1}b \ \cdots \ X\Lambda^{k-1}X^{-1}b), \\ &= X(c \ \Lambda c \ \cdots \ \Lambda^{k-1}c). \end{aligned}$$

But the matrices Λ^j , $j = 1, \dots, k-1$ are diagonal and

$$\Lambda^j c = \begin{pmatrix} \lambda_1^j c_1 \\ \vdots \\ \lambda_n^j c_n \end{pmatrix} = D_c \begin{pmatrix} \lambda_1^j \\ \vdots \\ \lambda_n^j \end{pmatrix}.$$

Therefore, $K_{n,k} = XD_c \mathcal{V}_{n,k}$. □

This factorization was used in [548]; see also [642, 1017]. When the matrix A is not diagonalizable we can obtain a generalized factorization by using the Jordan canonical form $A = XJX^{-1}$ but the powers of the Jordan blocks are upper triangular with the powers of the eigenvalues on the diagonals and the factorization is not as useful as it is for diagonalizable matrices.

Krylov matrices have at least two bad numerical properties. First, when k becomes large they may not be of full numerical rank. This is understandable since their columns are obtained by repeated multiplication with the matrix A . Under certain conditions the vectors $A^j b$ tend to a vector which is a multiple of the eigenvector of A corresponding to the eigenvalue of largest modulus. Therefore, the columns of $K_{n,k}$ tend to be in the same direction, and they may lose their linear independence in finite precision arithmetic.

Secondly, Krylov matrices may have a large condition number. We have

$$K_{n,k}^* K_{n,k} = \mathcal{V}_{n,k}^* D_c^* X^* X D_c \mathcal{V}_{n,k}.$$

Let us assume that A is a normal matrix and therefore $X^* X = I$ and that all the components of c are equal to 1. Then, $K_{n,k}^* K_{n,k} = \mathcal{V}_{n,k}^* \mathcal{V}_{n,k}$. Note that $K_{n,k}^* K_{n,k}$ is then a positive definite Hankel matrix. It is known that square Hankel matrices are badly conditioned when the eigenvalues λ_j are real; see [408]. However, the situation can sometimes be different when the eigenvalues are complex. For results on the condition numbers of Krylov matrices, see [58], [931] for real matrices and [82] for Hermitian matrices. Depending on the eigenvalues of A , one observes in many cases that the condition number of the Krylov matrix grows exponentially with k .

As an example, we generated a random normal matrix A of order 30 with real and complex eigenvalues. Its condition number was 16.48. The vector b had all components equal to 1. The numerical rank of K_{30} was 26, and its condition number was $3.02 \cdot 10^{16}$. The smallest singular value was $4.1247 \cdot 10^{-4}$, and the largest one was $1.2457 \cdot 10^{13}$.

Another example, inspired by [186], is a tridiagonal matrix of order 10 with constant upper and lower first diagonals equal, respectively, to 20 and -1 . The main diagonal is zero except $a_{1,1} = -1$. The right-hand side is $b = e_1$. Even though the rank of $K_{10}(A, b)$ is 10, its condition number is $7.6234 \cdot 10^{12}$. If we construct the corresponding Krylov matrix of order 20, its condition number is $2.4136 \cdot 10^{29}$ and its numerical rank is 8. Note that these matrices have only pairs of complex conjugate eigenvalues. These small experiments show that, in finite precision arithmetic, the columns of the matrices $K_{n,k}$ cannot be used as a basis of the Krylov subspaces even for moderate values of k . For a study of perturbations of Krylov subspaces and bases, see [186] and [611].

Krylov subspaces and Hessenberg matrices are intimately linked. To see this we consider the (economy size) QR factorization of Krylov matrices. Let

$$K_{n,k+1} = (b \ A K_{n,k}) = V_{n,k+1} R_{k+1},$$

where $V_{n,k+1}^* V_{n,k+1} = I$ and R_{k+1} is upper triangular of order $k + 1$. Using also the QR factorization of $K_{n,k}$ we have

$$(b \ A V_{n,k} R_k) = (b \ A V_{n,k}) \begin{pmatrix} 1 & 0 \\ 0 & R_k \end{pmatrix} = (b \ A K_{n,k}) = V_{n,k+1} R_{k+1},$$

which yields

$$(b \ A V_{n,k}) = V_{n,k+1} R_{k+1} \begin{pmatrix} 1 & 0 \\ 0 & R_k^{-1} \end{pmatrix}.$$

From the last k columns of the previous relation, we obtain

$$AV_{n,k} = V_{n,k+1} R_{k+1} \begin{pmatrix} 0 \\ R_k^{-1} \end{pmatrix} = V_{n,k+1} \underline{H}_k. \quad (2.37)$$

The first k columns of $V_{n,k+1}$ are equal to the columns of $V_{n,k}$, and we see that the product of the last two matrices on the first right-hand side is a $(k+1) \times k$ upper Hessenberg matrix \underline{H}_k . If we consider the first k rows H_k of this Hessenberg matrix, then

$$H_k = (I_k \ 0) R_{k+1} \begin{pmatrix} 0 \\ R_k^{-1} \end{pmatrix} = R_k (I_k \ R_k^{-1} R_{1:k,k+1}) \begin{pmatrix} 0 \\ I_k \end{pmatrix} R_k^{-1},$$

which can be related to Theorem 2.5 and shows that R_k can be identified with U_k in that theorem. Note that this matrix describes the change of basis of the Krylov subspace from the natural (or monomial) basis to an orthonormal basis given by the columns of $V_{n,k}$. We also remark that we did not use the property of $V_{n,k}$ being unitary. Therefore, the previous construction is valid for any basis of the Krylov subspace, provided that R_k is upper triangular and nonsingular.

The definition of a Krylov subspace can be extended by using more than one generating vector b . Let B be an $n \times s$ matrix with $n \geq s \geq 1$ of rank s with columns b_i . The block Krylov subspace is defined as

$$\mathcal{K}_k(A, B) = \text{span} \left\{ \sum_{j=0}^{k-1} (A)^j B c_j \mid c_j \in \mathbb{C}^s \right\} = \sum_{j=1}^s \mathcal{K}_k(A, b_j), \quad k \geq 1. \quad (2.38)$$

Note that there exist other definitions of block Krylov subspaces where the vectors c_j are replaced by $s \times s$ matrices C_j . They are used in methods for solving linear systems with s right-hand sides; see Chapter 12. In this case the block Krylov subspace is the Cartesian product of subspaces defined by (2.38). In this book we are interested in linear systems with one right-hand side and the definition (2.38) is the one we will use in Chapter 10 about IDR methods. For the grade of block Krylov subspaces, see [492].

2.11 Finite precision arithmetic and backward stability

In this book we are interested in the mathematical theory of Krylov methods but also in practical computations. Unfortunately, computations are not done exactly on a digital computer, so we have to deal with rounding errors in finite precision arithmetic. For interesting information about IEEE floating-point arithmetic standards, see the small but enlightening book [729] and also [200, 201, 427, 521, 709].

In the standard model corresponding to IEEE arithmetic, for any of the four basic operations $(+, -, *, /)$ denoted by op and two floating-point numbers x and y , the floating-point result $fl(x op y)$ of the operation applied to x and y satisfies

$$fl(x op y) = (x op y)(1 + \delta), \quad |\delta| \leq u,$$

u being the unit roundoff which is $(1/2)\beta^{1-t}$ where β is the base of the arithmetic and t is the number of digits in the mantissa of floating-point numbers. This bound

is obtained when rounding to the nearest floating-point number is used, but this is generally the case. Otherwise, u is twice this value. In IEEE double precision $\beta = 2, t = 53$ and

$$u = 1.110223024625157 \cdot 10^{-16}.$$

It is half of the machine epsilon $\varepsilon = \beta^{1-t}$ which is the distance from 1 to the next larger floating-point number.

The operations we are interested in for Krylov methods are dot products, ratios of dot products, matrix–vector products and additions of scalar multiples of vectors. We just summarize the results; for more details, see [521] and Section 4.1 of [676]. If x and y are real vectors of length n , we have

$$fl(x^T y) = (x + \Delta x)^T y = x^T(y + \Delta y),$$

where the perturbation terms are bounded as

$$|\Delta x| \leq \gamma_n |x|, \quad |\Delta y| \leq \gamma_n |y|, \quad \gamma_n = \frac{nu}{1 - nu}, \quad nu < 1.$$

This can be written as

$$fl(x^T y) = x^T y + cnu + O(u^2),$$

with $|c| \leq |x|^T |y|$.

For ratios of dot products (with $w^T z \neq 0$) we have

$$fl\left(\frac{x^T y}{w^T z}\right) = \frac{x^T y}{w^T z} + cu + O(u^2),$$

with

$$|c| \leq \frac{|x|^T |y|}{|w^T z|} + \frac{n}{1 - nu} \left(\frac{|x|^T |y|}{|w^T z|} + \frac{|w|^T |z| |x^T y|}{(w^T z)^2} \right).$$

For a matrix–vector product $y = Ax$ where A is a square sparse matrix with at most m nonzero entries per row, we obtain

$$fl(y) = (A + \Delta A)x, \quad |\Delta A| \leq \gamma_m |A|,$$

where γ_m has been defined above.

We now consider $\alpha x + \beta y$ where α and β are scalars and x and y are two vectors. We have

$$|fl(\alpha x + \beta y) - (\alpha x + \beta y)| \leq 2u(|\alpha| |x| + |\beta| |y|) + O(u^2),$$

and

$$\|fl(\alpha x + \beta y) - (\alpha x + \beta y)\| \leq 2u(|\alpha| \|x\| + |\beta| \|y\|) + O(u^2).$$

What we would like for our computations with Krylov methods for solving linear systems is that the behavior of the algorithm in finite precision arithmetic is “not

too far" from what we would have mathematically. Hence we are concerned with the concept of *stability*. Intuitively, an algorithm is stable if it is not too sensitive to perturbations. Perturbations may arise from uncertainties in the data, that is, the matrix A and/or the right-hand side b , or from rounding errors. A particularly interesting way to consider stability is to introduce the *backward error*. Assume that we have an approximate solution \tilde{x} of $Ax = b$. Then, the (normwise) relative backward error is defined as

$$\varepsilon_B(\tilde{x}) = \min\{\omega \mid (A + \Delta A)\tilde{x} = b + \Delta b, \|\Delta A\|/\|A\| \leq \omega, \|\Delta b\|/\|b\| \leq \omega\}.$$

That is, we ask for the smallest relative perturbations such that the approximation \tilde{x} is the exact solution of $(A + \Delta A)\tilde{x} = b + \Delta b$. It has been shown (see, for instance, [770] and [521, p. 120]) that

$$\varepsilon_B(\tilde{x}) = \frac{\|b - A\tilde{x}\|}{\|A\| \|\tilde{x}\| + \|b\|}.$$

An algorithm is (normwise) backward stable if it produces an approximate solution \tilde{x} such that $\varepsilon_B(\tilde{x})$ is proportional to the machine epsilon ε . If it is the case, it means that the approximate solution that we have computed is the exact solution of a nearby problem. It must be noted that stability does not only depend on the algorithm but also on the problem to be solved. Some algorithms may look stable for some problems and unstable for others. The ideal situation is to have algorithms which are stable for very large classes of problems.

As an example related to the contents of this chapter, let us consider Givens rotations. We annihilate the second components of 1000 random vectors $(x, y)^T$ drawn for a standard normal distribution. For each pair, let

$$Q_k = \begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix}.$$

We first consider $\|I_2 - Q_k^T Q_k\|$ for $k = 1, \dots, 1000$. The minimum is $1.9439 \cdot 10^{-20}$, the maximum is $4.4734 \cdot 10^{-16}$, the mean is $1.1988 \cdot 10^{-16}$ and the standard deviation is $1.1988 \cdot 10^{-16}$. Hence, the rotation matrices are orthonormal up to working precision but this does not come as a surprise considering their definition. Let us now look at the results for the second component of $Q_k(x, y)^T$ which, mathematically, must be zero. The minimum is 0, the maximum is $4.4409 \cdot 10^{-16}$, that is, 2ε , the mean is $1.9923 \cdot 10^{-17}$ and the standard deviation is $4.2911 \cdot 10^{-17}$. For the first component we do not know the exact value of the result; therefore we compare the result in double precision to the same computation done in extended precision with 32 decimal digits. The minimum of the absolute error is 0, the maximum is $8.8818 \cdot 10^{-16}$, that is, 4ε , the mean is $5.1251 \cdot 10^{-17}$ and the standard deviation is $1.0949 \cdot 10^{-16}$. Hence, the maximum absolute error for both components is a small multiple of the machine epsilon and we have a perfectly nice computation. But, of course, the problems where we want to use Givens rotations are not random.

As another example, consider, given a computed vector \hat{x} , the computation of the residual $r = b - A\hat{x}$. Let $\hat{r} = fl(b - A\hat{x})$ be the floating-point result of the computation. The vector \hat{r} is often called the *true* residual. Of course, it depends on what is the meaning we associate with the word “true”, but, for sure, in finite precision arithmetic this is generally not the exact residual. It is shown in [521] that

$$\hat{r} = r + \Delta r, \quad |\Delta r| \leq \gamma_{n+1} (|A| |\hat{x}| + |b|),$$

where $|\cdot|$ is used to denote componentwise bounds. We can obtain a normwise bound by replacing the moduli by norms. The upper bound can be much larger than the machine epsilon if $\|A\|$ and/or $\|b\|$ are large, but note that Δr is an absolute difference.

2.12 Parallel computing

Nowadays large problems are solved using parallel computers. Even though there exist different architectures for parallel computers, many of them are made of nodes which are linked by a communication network. Nodes are made of several or many processors which are made of several or many cores which are the computing units and, sometimes, accelerators known as GPUs. Generally, the memory is distributed in the nodes in which sometimes the memory is shared between the processors. Except for problems and numerical methods which are said to be “embarrassingly parallel”, data has to be exchanged between nodes or processors. Usually, the goal of a computer user is to solve his/her problem in the smallest amount of time, provided the solution is satisfactory, that is, sufficiently accurate. Communication is generally a limiting factor for the performance of an algorithm. Krylov methods for solving linear systems need at least one matrix–vector product per iteration. Moreover, many of these methods have also to compute dot products of vectors of length n . For solving large problems, the matrix and the vectors in the algorithms have to be distributed in local memories. Hence, computing matrix–vector products and dot products involve global communications which slow down the computation. Note that a dot product is a reduction operation computing only one scalar value from two vectors. Even though, depending on the distribution of the vectors in the local memories, the computation can be done partly in parallel, if the scalar result of the dot product is needed by all the processing units, it has to be broadcasted through the communication network.

We will see in the next chapters that the algorithms which are mathematically the most efficient ones are not very well suited for parallel computation. Therefore, along the years, we have seen tentatives for developing new algorithms which are more parallel or for developing modifications of known algorithms to increase their parallelism by minimizing, as far as possible, the communications. Recently, this class of algorithms has been named *communication-avoiding*, even though one cannot completely avoid communications for solving linear systems.

Our goal in this book is to study mathematically the main Krylov methods and to give (small) examples to illustrate the theory. Therefore, parallel computing is not our main objective. However, we will indicate what are the limitations of the studied algorithms concerning parallel computing and some of the modifications that have been proposed for increasing their parallelism.

An important issue for these variants of Krylov methods with improved parallelism is stability. Quite often these methods trade stability for increased parallelism. Roughly speaking, these two goals, stability and parallelism, are antagonistic; for a discussion of this issue for symmetric linear systems, see [194].

2.13 Preconditioning

In this book the algorithms are described and discussed without preconditioning. However, for many if not all problems, the use of preconditioners is crucial to obtain a fast convergence of Krylov methods. Introducing a preconditioner in Krylov methods is relatively easy for nonsymmetric linear systems since we just have to modify the matrix–vector products with the matrix A .

Let M be the nonsingular preconditioning matrix. Left preconditioning is defined by solving $M^{-1}Ax = M^{-1}b$. Hence, after modifying the right-hand side b in the initialization phase, we replace in the iterations a matrix–vector $z = Av$ by $z = M^{-1}Av$. It amounts to solving

$$Mz = Av.$$

Most preconditioners are constructed in such a way that a solve with the matrix M is easy and cheap.

Right preconditioning is replacing $Ax = b$ by $AM^{-1}y = b$ and $x = M^{-1}y$. The benefit of left preconditioning is that we directly obtain the approximations of the exact solution x . But the drawback is that what is computed in many Krylov methods is the preconditioned residual $M^{-1}(b - Ax)$. Using the norm of this vector to stop the iterations does not tell what is the norm of the residual $b - Ax$. The benefits and drawbacks of right preconditioning are the opposite of those of left preconditioning. The residual is the one we would like to see, but the algorithms deliver approximations y_k to y and not to x . After convergence we have to solve $Mx_k = y_k$ from the last iterate.

We observe that for parallel computing the solves $Mz = y$ have to be parallelized too. Otherwise, they could be the bottleneck of the computation.

2.14 Stopping criterion

An important question is when to stop an iterative method. There is no definitive answer to this question because it depends on what the user is looking for. For

instance, when the linear solve is within a Newton-like method for solving a nonlinear system of equations, the user might want to obtain a small residual norm, even though this may be not necessary during the first iterations; see, for instance, [267, 913]. However, in many other problems, the user would like to obtain a solution x_k “close” to the exact solution x . It turns out that there are Krylov methods for which approximations of the norms of the error $x - x_k$ can be obtained during the iterations. This is the case for the A -norm of the error in the conjugate gradient method for the symmetric case [431, 435, 672, 674, 689, 893] and for the ℓ_2 norm of the error in GMRES, see [677] and Section 5.8.

The fact that residual-based criteria are not always appropriate for stopping the iterations was already mentioned in 1952 in the paper [515] introducing the conjugate gradient (CG) algorithm for positive definite symmetric linear systems; see page 410 and page 432. This remark was based on the fact that one can construct examples for which the CG residual norms are increasing at all iterations whence the A -norm of the error is decreasing.

At iteration k the error $\varepsilon_k = x - x_k$ and the residual $r_k = b - Ax_k$ are linked by the relation $A\varepsilon_k = r_k$, and we have

$$\frac{1}{\|A^{-1}\|} \|\varepsilon_k\| \leq \|r_k\| \leq \|A\| \|\varepsilon_k\|.$$

If $\|A\|$ is large, a small error norm does not necessarily imply a small residual norm and, if $\|A^{-1}\|$ is large, a small residual norm does not necessarily imply a small error norm. It also holds that

$$\frac{\|\varepsilon_k\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|r_k\|}{\|b\|}.$$

The product $\|A\| \|A^{-1}\|$ is the condition number of A . If the condition number is large the relative norm of the error can be large, even if $\|r_k\|/\|b\|$ is small.

As an example let us take the matrix `fs_183_6` of order $n = 183$ whose properties are given in Appendix A. Its norm is $1.1808 \cdot 10^9$, and its condition number is $1.7368 \cdot 10^{11}$. The norm of the right-hand side b is $1.1808 \cdot 10^9$. The norm of the “exact” solution given by a direct method is 13.53 with a residual norm $5.9852 \cdot 10^{-8}$. We do 40 iterations of GMRES (see Chapter 5) with $x_0 = 0$. The norm of the error is $1.0956 \cdot 10^{-2}$, the norm of the residual is $7.9730 \cdot 10^{-4}$ and

$$\frac{\|\varepsilon_k\|}{\|x\|} = 8.0987 \cdot 10^{-4}, \quad \frac{\|r_k\|}{\|b\|} = 6.7520 \cdot 10^{-13}.$$

The ratio of these two values is of the order of 10^9 . This is an example for which stopping using residual norms is not satisfactory since the relative residual norm is small but the error norm is still large.

In [27] it is proposed to use variants of the backward error for the stopping criterion $\omega_j \leq \epsilon$ where ϵ is a user-defined threshold with several possibilities for ω_j at iteration k ,

$$\omega_1 = \max_i \frac{[r_k]_i}{[\|A\| |x_k| + \|b\|]_i}, \quad \omega_2 = \frac{\|r_k\|_\infty}{\|A\|_\infty \|x_k\|_1 + \|b\|_\infty}, \quad \omega_3 = \frac{\|r_k\|_\infty}{\|b\|_\infty}.$$

We have $\omega_2 \leq \omega_1$ and $\omega_2 \leq \omega_3$. The definitions for ω_1 and ω_2 were used to avoid the use of $\|A\|$ which is generally unknown and difficult to compute in the definition of the backward error.

For our example above the backward error computed with the Euclidean norms is $4.6477 \cdot 10^{-14}$ and

$$\omega_1 = 1.3178 \cdot 10^{-4}, \quad \omega_2 = 2.2822 \cdot 10^{-15}, \quad \omega_3 = 4.1992 \cdot 10^{-13}.$$

Hence, in this case, only ω_1 is satisfactory but its computation needs one more matrix–vector product and this may be too costly.

If we take a different starting vector x_s defined as $10^4 \text{randn}(183, 1)$ we have an initial residual whose norm is $4.1359 \cdot 10^{12}$ and, after 40 iterations,

$$\omega_1 = 8.8955 \cdot 10^{-1}, \quad \omega_2 = 3.0590 \cdot 10^{-11}, \quad \omega_3 = 1.5930 \cdot 10^{-8}.$$

The bad scaling of the initial vector can be corrected by what is known as Hegedüs' trick, see [640]. Let

$$\xi = \frac{b^T A x_s}{\|A x_s\|^2}.$$

Then, we define a new starting vector as ξx_s . For our example, this new nonzero starting vector yields $2.5908 \cdot 10^7$ for the norm of the initial residual vector and

$$\omega_1 = 1.0525 \cdot 10^{-3}, \quad \omega_2 = 2.0282 \cdot 10^{-14}, \quad \omega_3 = 3.7325 \cdot 10^{-12},$$

which is an improvement over using x_s .

Unfortunately, many codes implementing Krylov methods use

$$\|r_k\| \leq \epsilon \|r_0\|$$

as a stopping criterion. In our example it is the same as $\|r_k\| \leq \epsilon \|b\|$ since $x_0 = 0$. The value of $\|r_k\|/\|b\|$ is $6.7520 \cdot 10^{-13}$ with $x_0 = 0$ and stopping with this criterion is misleading. Anyway, it is always better to replace $\|r_0\|$ by $\|b\|$.

Let us consider an example with a well-conditioned matrix `pde225` of order 225. Its condition number is 39.06. After 40 GMRES iterations the error norm is $5.0041 \cdot 10^{-3}$ and the norm of the residual is $7.3010 \cdot 10^{-3}$ and

$$\frac{\|\varepsilon_k\|}{\|x\|} = 4.9773 \cdot 10^{-4}, \quad \frac{\|r_k\|}{\|b\|} = 4.5189 \cdot 10^{-4}.$$

Moreover,

$$\omega_1 = 7.7306 \cdot 10^{-4}, \quad \omega_2 = 1.0155 \cdot 10^{-6}, \quad \omega_3 = 6.1092 \cdot 10^{-4}.$$

Hence, for well-conditioned matrices, $\|r_k\| \leq \epsilon \|b\|$ is an acceptable stopping criterion.

Note that, in some Krylov methods, the residual vector and the approximate solution may not be available during the iterations. We also observe that, when using a preconditioner, it is not clear which backward error must be considered, the one for the preconditioned system or the one for the original system. Moreover, computing any norm of $M^{-1}A$ or AM^{-1} might not be possible.

In fact, when solving problems arising from (systems of) partial differential equations, the stopping criterion of the linear solver must be related to the discretization error of the numerical method which is used. For a discussion of this issue, see [640]; for self-adjoint PDEs, see also [22, 27, 29–33, 578, 744–746].

2.15 Historical notes

The word “eigenvalue” comes from the German word “eigen” which means “own” or “proper”. Eigenvalues were considered in the 18th century long before matrices appeared in the field of applied mathematics. In the early days they were called proper, latent or characteristic values. This last naming is still found in the characteristic polynomial $\det(\lambda I - A)$ whose roots are the eigenvalues. The term “latent root” was coined by James J. Sylvester in 1883. He wrote *It will be convenient to introduce here a notion (which plays a conspicuous part in my new theory of multiple algebra), namely that of the latent roots of a matrix, latent in a somewhat similar sense as vapour may be said to be latent in water or smoke in a tobacco-leaf.* Sylvester also coined the term “matrix” in 1850.

The Jordan canonical form is named after Camille Jordan (1838–1922). His paper was published in 1870 but equivalent results were published in 1868 by Karl Weierstrass (1815–1897). For details on the feud that followed in which Leopold Kronecker (1823–1891) was involved see [640], page 188.

The minimal polynomial was introduced by Ferdinand Georg Frobenius (1849–1917) in 1878. Frobenius was one of the first to prove the general form of the Cayley–Hamilton theorem which was proved only for 2×2 and 3×3 matrices by Arthur Cayley in 1858 and, indirectly, by William Rowan Hamilton for $n = 4$ using quaternions at the beginning of the 1860s. However, Hamilton claimed that he already knew this result in 1846.

Singular values originated from works of C. Jordan and Eugenio Beltrami (1835–1900) on bilinear forms in 1873–1874. James J. Sylvester (1814–1897) worked also on the SVD in 1889 as well as Léon César Autonne (1859–1916) in 1915. Apparently the term “singular values” was coined (in French “valeurs singulières”) by Emile Picard (1856–1941) in 1910.

Companion matrices appeared as diagonal blocks in the Frobenius canonical form of a matrix. This is related to a direct sum decomposition of a space into cyclic subspaces. The relation $AK = KC$ has been known for quite a long time and used to compute the characteristic polynomial of matrices; see [438], page 348.

Hessenberg matrices are named after Karl Hessenberg (1904–1959). Hessenberg matrices do not appear in Hessenberg's thesis but in the report [512] in 1940, page 23, equation (58). This paper, with handwritten equations, is available at the Web site <http://www.hessenberg.de> from where this information was obtained. The characterization of the lower triangular part of inverses of Hessenberg matrices was published by Yasuhiko Ikebe [540] in 1979 and Dmitry K. Faddeev [340] in 1981 in Russian and in 1984 in English. The result for the inverse of a symmetric tridiagonal matrix was published in French in 1971 by Jacques Baranger and Marc Duc-Jacquet; see [70].

Unitary transformations that bear his name were introduced by Alston Scott Householder (1904–1993) in the short paper [533] in 1958. The goal was to obtain a QR factorization of a matrix. However, related transformations appeared in a book by Herbert W. Turnbull and Alexander C. Aitken [930] in 1932.

James Wallace Givens (1910–1993) advocated the use of rotation matrices to reduce a matrix to upper triangular form. We remark that rotations were used by Carl Gustav Jacob Jacobi (1804–1851) in 1846.

Vandermonde matrices are named after Alexandre-Théophile Vandermonde (1735–1796) even though, apparently, the determinant of these matrices does not appear explicitly in his publications. However, he made important contributions to the theory of determinants.

The Cauchy–Binet formula was first proposed in 1812 independently by Augustin-Louis Cauchy (1789–1857) and Jacques Binet (1786–1856).

The Sherman–Morrison formula was published in 1950 in the four-page paper [823] by two statisticians, Jack Sherman and Winifred J. Morrison. They were interested in the change of the inverse of a matrix when one of its entries is modified. It appears in the form given above, with two general vectors u and v , in a paper by Maurice S. Bartlett [75] in 1951. The extension to matrices instead of vectors appeared in a 1950 report by Max A. Woodbury [985] although, according to William W. Hager [500], it has been published before in some disguises in papers by William J. Duncan (1944) and Louis Guttman (1946).

Alexei Nikolaevich Krylov (1863–1945) was a Russian mathematician, physicist and naval engineer. For information about his life, see his memoirs [609] or the translation into English by Laura N. Meyerovich [691] in 2014.

Krylov's paper [608] in 1931 whose title can be translated as “On the numerical solution of the equation by which, in technical matters, frequencies of small oscillations of material systems are determined” introduced what is now called Krylov vectors. He was interested in the coefficients of the characteristic polynomial of a matrix. His method was described in matrix notation by Felix Gantmacher (1908–1964) in 1934; see [396]. Let us assume that we have a vector v of grade n with respect to A and we are looking for the coefficients of the characteristic polynomial p_c . We can write

$$\begin{pmatrix} v & Av & \cdots & A^{n-1}v & A^n v \\ 1 & \lambda & \cdots & \lambda^{n-1} & \lambda^n - p_c(\lambda) \end{pmatrix} \begin{pmatrix} -\alpha_0 \\ \vdots \\ -\alpha_{n-1} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

The first row of this matrix relation comes from the Cayley–Hamilton theorem and the second row from the definition of the characteristic polynomial. To obtain a nonzero solution the determinant of the matrix on the left must be zero and with a few manipulations we get

$$p_C(\lambda) = \frac{\det \begin{pmatrix} v & Av & \cdots & A^{n-1}v & A^n v \\ 1 & \lambda & \cdots & \lambda^{n-1} & \lambda^n \end{pmatrix}}{\det(v \ A v \ \cdots \ A^{n-1} v)}.$$

The interest of this formulation is that λ appears only in the last row of the matrix in the numerator rather than in every row as in $\lambda I - A$. However, using this method cannot be recommended numerically since we have seen that Krylov matrices can be badly conditioned and even rank-deficient. But we observe that we can use the same trick using a basis different from the monomial one to write down the polynomial. For a description of Krylov’s method, see also Alston S. Householder’s book [535], page 148.

However, Krylov was not the only mathematician to use a sequence v, Av, A^2v, \dots , see, for instance, H.W. Turnbull and A.C. Aitken [930] in 1932, pages 43, 46–51. They used these vectors for theoretical purposes to obtain the Frobenius canonical form and the Jordan canonical form of a matrix.

The term “Krylov sequence” appears in a paper by A.S. Householder and Friedrich L. Bauer [536, p. 33] in 1959. We observe that Walter Arnoldi’s paper [35] in 1951 which, essentially, is concerned with computing an orthonormal basis of a Krylov subspace does not mention Krylov. It only refers to the 1950 paper of Cornelius Lanczos [615]. In Lanczos’ paper Krylov is only mentioned in a footnote on the first page (page 255).

The concept of backward error appeared in disguise in early papers by John von Neumann and Herman H. Goldstine [956] in 1947 and also by Alan Turing [929] in 1948. It was used in a paper by Wallace Givens [426] in 1958 and even earlier in a 1954 report. But backward error analysis was essentially developed and popularized by James H. Wilkinson (1919–1986) starting in the 1950s; see his famous book [982] from 1965. For more details on rounding error analysis, see Nicholas Higham’s book [521].

Chapter 3

Q-OR and Q-MR methods



In the previous chapter we briefly introduced Krylov subspaces. Krylov methods are based on reduction of the problem to small Krylov subspaces whose dimension grows with the iteration number. Most popular Krylov methods can be classified as either a quasi-orthogonal residual (Q-OR) method or a quasi-minimal residual (Q-MR) method, with most Q-OR methods having Q-MR analogs; see [296]. Well-known Q-OR/Q-MR pairs of methods are the FOM/GMRES pair or the BiCG/QMR pair. These methods will be described in later chapters. The existing Q-OR/Q-MR methods mainly differ in the type of basis used for the Krylov subspace. The next chapter addresses various types of bases in detail. In this chapter we study the mathematical properties of abstract Q-OR and Q-MR methods regardless of the basis which is chosen.

3.1 Definition

Usually there is no good reason to start a Q-OR or Q-MR method with an initial guess different from zero. However, in practice, several of these methods are restarted to avoid too large storage demands and, necessarily, the vector from which we restart is different from zero; see Chapter 11. Therefore, in the following, we will use a general starting vector x_0 . Let $r_0 = b - Ax_0$ be the corresponding initial residual vector.

Since the Krylov subspaces $\mathcal{K}_k(A, r_0)$ are nested, we are interested in ascending bases. This means that, if v_1, \dots, v_k are the basis vectors for $\mathcal{K}_k(A, r_0)$, then v_1, \dots, v_k, v_{k+1} are the basis vectors for $\mathcal{K}_{k+1}(A, r_0)$ as long as $k + 1 \leq d$ where d is the grade of r_0 with respect to A . We choose $v_1 = r_0/\|r_0\|$. Moreover, when $k + 1 \leq d$, Av_k is in $\mathcal{K}_{k+1}(A, r_0)$ and we can write it in the basis v_1, \dots, v_k, v_{k+1} as

$$Av_k = \sum_{j=1}^{k+1} h_{j,k} v_j. \quad (3.1)$$

The scalars $h_{j,k}$ are such that the basis vectors have the properties required for the given Krylov method. They can, for instance, be chosen such that v_{j+1} is the orthogonal projection of Av_j onto $\mathcal{K}_j(A, r_0)$. We assume that the basis vectors are of unit norm and then write

$$h_{k+1,k} v_{k+1} = Av_k - \sum_{j=1}^k h_{j,k} v_j,$$

with $h_{k+1,k} = \|Av_k - \sum_{j=1}^k h_{j,k} v_j\|$ real and positive. By induction we see that the vector v_{k+1} can be expressed as a polynomial of degree k in A applied to v_1 or r_0 . Because of (3.1) the basis vectors satisfy

$$AV_{n,k} = V_{n,k} H_k + h_{k+1,k} v_{k+1} e_k^T = V_{n,k+1} \underline{H}_k, \quad (3.2)$$

where H_k is upper Hessenberg and the columns of $V_{n,k}$ are the basis vectors v_1, \dots, v_k . The matrix H_k is the leading principal submatrix of order k of the larger upper Hessenberg matrix H (with real and positive subdiagonal entries) in the decomposition

$$AV = VH \quad (3.3)$$

valid when $k = n$ in (3.2). The matrix \underline{H}_k is H_k appended with the k first entries of the $(k+1)$ st row of H . Relation (3.2) was in fact derived already differently, see relation (2.37). It is called an *Arnoldi-like relation* for reasons that will become obvious later. It is also sometimes called a *Hessenberg decomposition* or relation.

The subsequent iterates x_k , $k \geq 1$ in Q-OR and Q-MR methods are of the form

$$x_k = x_0 + V_{n,k} y_k, \quad (3.4)$$

for some uniquely defined vector $y_k \in \mathbb{R}^k$ or \mathbb{C}^k if the data is complex. This means that we look for x_k in $x_0 + \mathcal{K}_k(A, r_0)$. Since $v_1 = r_0 / \|r_0\|$, the *residual vector* r_k , defined as $r_k = b - Ax_k$, can be written as

$$\begin{aligned} r_k &= b - Ax_k = b - Ax_0 - AV_{n,k} y_k \\ &= \|r_0\| V_{n,k} e_1 - AV_{n,k} y_k \\ &= V_{n,k} (\|r_0\| e_1 - H_k y_k) - h_{k+1,k} [y_k]_k v_{k+1}. \end{aligned} \quad (3.5)$$

In a Q-OR method, the k th iterate x_k^O is defined (provided that H_k is nonsingular) by computing y_k in (3.4) as the solution of the linear system

$$H_k y_k = \|r_0\| e_1. \quad (3.6)$$

This annihilates the term within parenthesis in the rightmost expression of (3.5). We will denote the solution of equation (3.6) by y_k^O . Thus, the iterates of the Q-OR method are $x_k^O = x_0 + \|r_0\| V_{n,k} H_k^{-1} e_1$, the residual vector, which we denote as r_k^O , is proportional to v_{k+1} and

$$\|r_k^O\| = h_{k+1,k} |[y_k^O]_k|.$$

In case H_k is singular and x_k^O is not defined, we will define the residual norm as being infinite, $\|r_k^O\| = \infty$. The vector y_k^O is usually computed using Givens rotations to annihilate the subdiagonal entries of H_k and then solving an upper triangular system, as we have seen in Section 2.9. This is more convenient and stable than using an LU factorization.

The residual vector in (3.5) can also be written as

$$r_k = V_{n,k+1}(\|r_0\| e_1 - \underline{H}_k y_k). \quad (3.7)$$

Instead of annihilating the term within parenthesis in the rightmost expression of (3.5), it is even more desirable to minimize the norm of the residual itself. But this is costly if the columns of the matrix $V_{n,k+1}$ are not orthonormal. We can write down the corresponding normal equations,

$$(V_{n,k+1} \underline{H}_k)^*(V_{n,k+1} \underline{H}_k) y_k = \|r_0\| \underline{H}_k^* V_{n,k+1}^* V_{n,k+1} e_1,$$

and compute incrementally the Cholesky factorization $L_{k+1} L_{k+1}^*$ of $V_{n,k+1}^* V_{n,k+1}$. The matrix $\tilde{H}_k = L_{k+1}^* \underline{H}_k$ is $(k+1) \times k$ and upper Hessenberg and minimization of the norm of r_k in (3.7) is equivalent to minimization of the norm

$$\| \|r_0\| L_{k+1}^* e_1 - \tilde{H}_k y_k \| = \| \|r_0\| \overline{\ell}_{1,1} e_1 - \underline{H}_k y_k \|.$$

The QR factorization of \underline{H}_k can be computed by an update algorithm. But this algorithm is complicated and expensive and, up to our knowledge, it has not been proposed in the literature.

However, we can bound the residual norm,

$$\|r_k\| \leq \|V_{n,k+1}\| \| \|r_0\| e_1 - \underline{H}_k y_k \|.$$

Therefore, considering the right-hand side of this inequality, in a Q-MR method where the basis may not be orthogonal, the vector y_k is computed as the solution of the least squares problem

$$\min_y \| \|r_0\| e_1 - \underline{H}_k y \|.$$
 (3.8)

The solution of the problem (3.8), which we denote as y_k^M , is usually computed, just like the solution of (3.6), using Givens rotations to zero the subdiagonal entries of \underline{H}_k and then solving an upper triangular system; see Section 2.9. Note that if

$\|V_{n,k+1}\| \neq 1$, y_k^M does not minimize the norm of the residual but the norm of what is called the *quasi-residual* defined as

$$z_k^M = \|r_0\| e_1 - \underline{H}_k y_k^M. \quad (3.9)$$

However, since the basis vectors, which are the columns of $V_{n,k}$, are of unit norm, we have

$$\|V_{n,k+1}\| \leq \|V_{n,k+1}\|_F = \sqrt{k+1}.$$

This indicates that the difference between the norms of the residual and quasi-residual vectors does not grow too fast with the number of iterations. In case where the basis is orthonormal, one sometimes speaks of a true minimal residual (MR) method inside the class of Q-MR methods. An example is the GMRES method. Similarly, with an orthonormal basis for the Krylov subspaces, the corresponding Q-OR method produces truly orthogonal residual vectors, where orthogonality is meant with respect to the current Krylov subspace. This can be verified by multiplying (3.5) with $V_{n,k}^*$ from the left.

We denote by x_k^M and r_k^M the iterates and residual vectors of a general Q-MR method. It is interesting that the Q-MR iterates are always defined contrary to the Q-OR iterates. This chapter is mainly concerned with relations between Q-OR and Q-MR iterates and residual vectors. They are often expressed in terms of the current Hessenberg matrix H_k (or \underline{H}_k) and sometimes in terms of the sines and cosines in the Givens rotations used to bring H_k to upper triangular form, see Section 2.9. For other relations we will use the entries of the upper triangular matrix U (or its inverse) in the triangular Hessenberg decomposition $H = UCU^{-1}$ introduced in Theorem 2.3. Even if its relation to Krylov spaces has already been mentioned, we summarize it in the following result.

Theorem 3.1 *Let C be the companion matrix corresponding to the eigenvalues of the nonderogatory matrix A , let $r_0 = \|r_0\| Ve_1$ for a nonsingular matrix V and let the Krylov matrix $K = (r_0 \ A r_0 \ \cdots \ A^{n-1} r_0)$ be nonsingular. K can be decomposed as*

$$K = \|r_0\| VU, \quad (3.10)$$

for a nonsingular upper triangular matrix U if and only if $Ue_1 = e_1$ and

$$AV = VH, \quad (3.11)$$

for an unreduced upper Hessenberg matrix H satisfying $H = UCU^{-1}$.

Proof Let (3.10) hold, hence necessarily $Ue_1 = e_1$. We have

$$\begin{aligned} \|r_0\| AV &= AKU^{-1} = (K, A^n r_0) \begin{pmatrix} 0 \\ I_n \end{pmatrix} U^{-1} = (\|r_0\| VU, A^n r_0) \begin{pmatrix} 0 \\ I_n \end{pmatrix} U^{-1} \\ &= \|r_0\| VU (I_n, (\|r_0\| VU)^{-1} A^n r_0) \begin{pmatrix} 0 \\ I_n \end{pmatrix} U^{-1}, \end{aligned}$$

hence

$$A\|r_0\|VU = \|r_0\|VU \left(I_n, (\|r_0\|VU)^{-1} A^n r_0 \right) \begin{pmatrix} 0 \\ I_n \end{pmatrix}.$$

The rightmost matrix product is a companion matrix and by comparison with (2.15) we see that it equals C . Thus the unreduced upper Hessenberg matrix

$$H = U \left(I_n, (\|r_0\|VU)^{-1} A^n r_0 \right) \begin{pmatrix} 0 \\ I_n \end{pmatrix} U^{-1}$$

satisfies $H = UCU^{-1}$. If on the other hand, $AV = VH = VUCU^{-1}$, then

$$A\|r_0\|VU = \|r_0\|VUC.$$

Because $r_0 = \|r_0\|Ve_1$ and $Ue_1 = e_1$, comparing again with (2.15) gives $\|r_0\|VU = K$. \square

Hence the matrix U in the triangular Hessenberg decomposition of H represents the change of basis from V to the Krylov matrix K . Decomposition (3.10) is in general not a QR decomposition as V needs not be orthogonal; we assumed only that it has unit norm columns. In the following, we will use as well the matrix $M = U^*U$ which is equal to

$$M = \frac{1}{\|r_0\|^2} K^* V^{-*} V^{-1} K.$$

It is the moment matrix with entries $m_{i,j} = (V^{-1}A^i r_0)^* V^{-1} A^j r_0 / \|r_0\|^2$.

From relation (3.5) we see that if there is an index k for which $h_{k+1,k} = 0$ the Q-OR residual vector is $r_k^O = 0$ and x_k^O is the solution of the linear system. This is also what is obtained with the Q-MR method since the last row of the matrix \underline{H}_k is then zero. This index cannot be smaller than $d(A, r_0)$ since, otherwise, we would obtain an annihilating polynomial $p(A)r_0 = 0$ of degree smaller than $d(A, r_0)$. It means that Q-OR/Q-MR methods are direct methods mathematically providing the exact solution after a finite number of steps. However, they are used as iterative methods, and we hope to obtain a good approximation of the solution for $k \ll d(A, r_0)$. Of course, in finite precision arithmetic we never obtain $h_{k+1,k} = 0$ exactly except in small contrived examples. In the following we assume that $h_{j+1,j} > 0$, $j = 1, \dots, k$. Unless explicitly stated otherwise, we will also assume that H_k is nonsingular. We also assume that all basis vectors v_j for Krylov subspaces have unit norm.

3.2 Relations between Q-OR and the associated Q-MR process

In this section we study basic relations between Q-OR and corresponding Q-MR iterations, i.e., we compare the iterates $x_k^O = x_0 + V_{n,k}y_k^O$ and $x_k^M = x_0 + V_{n,k}y_k^M$, the vectors of coefficients y_k^O and y_k^M , the errors and the (quasi-) residual norms. We denote the errors by ε_k^O and ε_k^M . This notation does not agree with our conventions but note that e_k has already been used for the columns of the identity matrix. As we use the same starting vector for both methods, we have $r_0^O = r_0^M = r_0 = b - Ax_0$.

Theorem 3.2 Assume that $h_{k+1,k} \neq 0$. The vectors of coefficients y_k^O and y_k^M in a Q-OR and a corresponding Q-MR method satisfy

$$y_k^M = y_k^O - \gamma [y_k^O]_k H_k^{-1} H_k^{-*} e_k,$$

with

$$\gamma = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}.$$

For the iterates and the errors we have

$$x_k^M = x_k^O - \gamma [y_k^O]_k V_{n,k} H_k^{-1} H_k^{-*} e_k \quad (3.12)$$

and

$$\varepsilon_k^M = \varepsilon_k^O - \gamma [y_k^O]_k V_{n,k} H_k^{-1} H_k^{-*} e_k.$$

For the quasi-residual vector we have

$$z_k^M = \gamma [y_k^O]_k \begin{pmatrix} H_k^{-*} e_k \\ -\frac{1}{h_{k+1,k}} \end{pmatrix}. \quad (3.13)$$

Proof The vector y_k^M of coefficients for the Q-MR method is obtained mathematically by solving the normal equations corresponding to the least squares problem (3.8),

$$\underline{H}_k^* \underline{H}_k y_k^M = \|r_0\| \underline{H}_k^* e_1 = \|r_0\| H_k^* e_1.$$

This is a slight abuse of notation since the vectors e_1 in the last two terms do not have the same dimension. The matrix of this linear system is

$$\underline{H}_k^* \underline{H}_k = H_k^* H_k + h_{k+1,k}^2 e_k e_k^T.$$

With the definition of γ in the claim and applying the Sherman–Morrison formula (see relation (2.3)), we have

$$(H_k^* H_k + h_{k+1,k}^2 e_k e_k^T)^{-1} = (H_k^* H_k)^{-1} - \gamma (H_k^* H_k)^{-1} e_k e_k^T (H_k^* H_k)^{-1},$$

and

$$\begin{aligned} y_k^M &= \|r_0\| (H_k^* H_k)^{-1} H_k^* e_1, \\ &= \|r_0\| [H_k^{-1} e_1 - \gamma (e_k^T H_k^{-1} e_1) (H_k^* H_k)^{-1} e_k], \\ &= y_k^O - \gamma [y_k^O]_k (H_k^* H_k)^{-1} e_k, \end{aligned}$$

proving the first claim. The second and third claims follow immediately. Let us consider the quasi-residual. From (3.9),

$$\begin{aligned} z_k^M &= \|r_0\| e_1 - \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix} y_k^O + \gamma [y_k^O]_k \underline{H}_k H_k^{-1} H_k^{-*} e_k, \\ &= - \begin{pmatrix} 0 \\ \vdots \\ 0 \\ h_{k+1,k} [y_k^O]_k \end{pmatrix} + \gamma [y_k^O]_k \begin{pmatrix} H_k^{-*} e_k \\ h_{k+1,k} \|H_k^{-*} e_k\|^2 \end{pmatrix}. \end{aligned}$$

Finally,

$$z_k^M = [y_k^O]_k \begin{pmatrix} \gamma H_k^{-*} e_k \\ -h_{k+1,k} + \gamma h_{k+1,k} \|H_k^{-*} e_k\|^2 \end{pmatrix}.$$

But, using the expression for γ , we have

$$-h_{k+1,k} + \gamma h_{k+1,k} \|H_k^{-*} e_k\|^2 = -\frac{h_{k+1,k}}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2} = -\frac{\gamma}{h_{k+1,k}}.$$

This proves the last claim. \square

From Theorem 3.2 we obtain a relation between the norm of the Q-OR residual and the norm of the Q-MR quasi-residual.

Proposition 3.1 *The norm of the Q-MR quasi-residual z_k^M is linked to the norm of the Q-OR residual by*

$$\|z_k^M\|^2 = \frac{1}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2} \|r_k^O\|^2. \quad (3.14)$$

Proof From (3.13) the norm of z_k^M is

$$\begin{aligned}
\|z_k^M\|^2 &= |[y_k^O]_k|^2 \left[\gamma^2 \|H_k^{-*} e_k\|^2 + h_{k+1,k}^2 (-1 + \gamma \|H_k^{-*} e_k\|^2)^2 \right], \\
&= h_{k+1,k}^2 |[y_k^O]_k|^2 \left[\frac{h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}{(1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2)^2} + \left(-1 + \frac{h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2} \right)^2 \right], \\
&= h_{k+1,k}^2 |[y_k^O]_k|^2 \frac{1}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}, \\
&= \gamma |[y_k^O]_k|^2.
\end{aligned}$$

This proves the result since $\|r_k^O\| = h_{k+1,k} |[y_k^O]_k|$. \square

We will see later (relation (3.18)) what is the value of $1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2$. Proposition 3.1 shows that the norm of the quasi-residual of a Q-MR method is smaller than the corresponding norm of the Q-OR residual. These norms can also be related to the sines and cosines in the Givens rotations used to bring H_k to upper triangular form, see Section 2.9.

Theorem 3.3 *Let s_j and c_j be the sines and cosines in the Givens rotations used to bring H_k to upper triangular form. Then the norms of the Q-MR quasi-residuals are*

$$\|z_k^M\| = \|r_0\| |s_1 s_2 \cdots s_k|. \quad (3.15)$$

We have the following relations between the residual norms of the Q-OR method and the quasi-residual norms of the associated Q-MR method,

$$\|z_k^M\| = |c_k| \|r_k^O\|, \quad (3.16)$$

$$\frac{1}{\|r_k^O\|^2} = \frac{1}{\|z_k^M\|^2} - \frac{1}{\|z_{k-1}^M\|^2}. \quad (3.17)$$

Proof The first result (3.15) follows directly from Proposition 2.2. For the second result, we start from relation (3.14) in Proposition 3.1, showing that

$$\frac{\|r_k^O\|^2}{\|z_k^M\|^2} = 1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2.$$

Using the rotation matrices, we have

$$H_k^{-*} e_k = G_1^* \cdots G_{k-1}^* R_k^{-*} e_k = \frac{1}{|[R_k]_{k,k}|} G_1^* \cdots G_{k-1}^* e_k.$$

It yields

$$1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2 = \frac{|[R_k]_{k,k}|^2 + h_{k+1,k}^2}{|[R_k]_{k,k}|^2} = \frac{1}{1 - |s_k|^2},$$

where the last equality follows from (2.30). Hence,

$$\|z_k^M\|^2 = (1 - |s_k|^2) \|r_k^O\|^2 = |c_k|^2 \|r_k^O\|^2,$$

proving relation (3.16). For the third result we remark that

$$\begin{aligned} \frac{1}{\|z_k^M\|^2} - \frac{1}{\|z_{k-1}^M\|^2} &= \frac{1}{\|r_0\|^2} \left[\frac{1}{|s_1 \cdots s_k|^2} - \frac{1}{|s_1 \cdots s_{k-1}|^2} \right], \\ &= \frac{1}{\|r_0\|^2} \frac{1 - |s_k|^2}{|s_1 \cdots s_k|^2} = \frac{1}{\|r_0\|^2} \frac{|c_k|^2}{|s_1 \cdots s_k|^2}, \end{aligned}$$

which proves (3.17). \square

The results of Theorem 3.3 can also be obtained in the same way as Proposition 4.1 in [379] where some of them are proved for a particular Q-MR method in which H_k is block tridiagonal; see also [245, 296] and [152, 153].

Equation (3.17) is responsible for the well-known peak–plateau phenomenon; see [152, 245]. If the quasi-residual norms of the Q-MR method stagnate at iterations $k-1$ and k , the iterate of the corresponding Q-OR method is not defined, that is, H_k is singular. More generally, if we have near stagnation of the Q-MR method we have peaks for the Q-OR residual norm.

From the proof of Theorem 3.3 we have

$$1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2 = \frac{1}{|c_k|^2} = \frac{\|r_k^O\|^2}{\|z_k^M\|^2}. \quad (3.18)$$

Hence the coefficient γ in Theorem 3.2 can also be expressed as $\gamma = h_{k+1,k}^2 |c_k|^2$.

We can also relate the iterates of the Q-OR and Q-MR methods at different iterations. After transformation to upper triangular form with the Givens rotations, the upper Hessenberg matrices H_k and \underline{H}_k give rise to matrices R_k and \hat{R}_k , see Section 2.9.

Theorem 3.4 *We have*

$$x_k^O = x_{k-1}^M + \|r_0\| [q_k]_k \frac{\hat{r}_{k,k}}{r_{k,k}} V_{n,k} [\hat{R}_k^{-1}]_{:,k}, \quad (3.19)$$

the vector q_k being the right-hand side when solving $R_k y_k^O = G_{k-1} \cdots G_1 e_1$.

The error vectors satisfy a similar relation, replacing x_k^O (resp. x_{k-1}^M) by ε_k^O (resp. ε_{k-1}^M).

Proof From Proposition 2.3 we have

$$y_k^O = \begin{pmatrix} y_{k-1}^M \\ 0 \end{pmatrix} + \|r_0\| [q_k]_k \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k}.$$

Since $x_k^O = x_0 + V_{n,k} y_k^O$ and $x_{k-1}^M = x_0 + V_{n,k-1} y_{k-1}^M$, we obtain the result. \square

For our purposes later in this book it is interesting to see how the Q-MR iterates are changing with k .

Theorem 3.5 *We have the relation*

$$x_k^M = x_{k-1}^M + \|r_0\| [\hat{q}_k]_k V_{n,k} [\hat{R}_k^{-1}]_{:,k}, \quad (3.20)$$

where \hat{q}_k is the vector obtained by applying successively the rotations $1, \dots, k$ to e_1 .

Proof We have seen that, when using Givens rotations, the right-hand side of the least squares problem is computed gradually. From what we had at iteration $k-1$, we add a new component and apply the rotation G_k which modifies the last two components. Therefore,

$$\hat{q}_k = \begin{pmatrix} \hat{q}_{k-1} \\ [\hat{q}_k]_k \end{pmatrix}.$$

Then,

$$y_k^M = \|r_0\| \begin{pmatrix} \hat{R}_{k-1}^{-1} - \frac{1}{\hat{r}_{k,k}} \hat{R}_{k-1}^{-1} [\hat{R}_k]_{1:k-1,k} \\ 0 \end{pmatrix} \begin{pmatrix} \hat{q}_{k-1} \\ [\hat{q}_k]_k \end{pmatrix}.$$

It yields

$$x_k^M = x_0 + \|r_0\| V_{n,k} \left\{ \begin{pmatrix} \hat{R}_{k-1}^{-1} \hat{q}_{k-1} \\ 0 \end{pmatrix} + [\hat{q}_k]_k [\hat{R}_k^{-1}]_{:,k} \right\},$$

and

$$x_k^M = x_0 + \|r_0\| \{ V_{n,k-1} \hat{R}_{k-1}^{-1} \hat{q}_{k-1} + [\hat{q}_k]_k V_{n,k} [\hat{R}_k^{-1}]_{:,k} \},$$

which proves the claim. \square

Computationally, relation (3.20) is not so interesting for two reasons. First, in practical algorithms we do not compute the matrix \hat{R}_k^{-1} and second, computing the second vector in the right-hand side is as costly as computing $x_0 + V_{n,k} y_k^M$. Moreover, generally the approximate solution is computed only when we have decided that convergence has occurred. However, Theorems 3.4 and 3.5 will be helpful when we will consider bases obtained by truncation; see Chapter 11.

3.3 Residual vectors and residual polynomials

Let us first show how we can express the Q-OR and Q-MR residual vectors in the basis given by the columns of $V_{n,k}$. From the definition of the Q-OR method, the residual vector is

$$r_k^O = -h_{k+1,k} [y_k^O]_k v_{k+1} = -\|r_0\| h_{k+1,k} (e_k^T H_k^{-1} e_1) v_{k+1},$$

since from equation (3.6) we have $H_k y_k^O = \|r_0\| e_1$. From Chapter 2, we have $H_k = U_k C^{(k)} U_k^{-1}$. From Theorems 2.4 and 2.5 it yields

$$e_k^T H_k^{-1} e_1 = -\frac{u_{k,k}}{\alpha_0^{(k)}},$$

where $\alpha_0^{(k)}$ is the constant coefficient of the characteristic polynomial of H_k . From Corollary 2.1, $\alpha_0^{(k)} = u_{k+1,k+1} \vartheta_{1,k+1}$ where the $\vartheta_{1,j}$'s are the entries of the first row of U_{k+1}^{-1} . Since $u_{k,k}$ is the product of the subdiagonal entries of H from 1 to $k-1$, we obtain

$$e_k^T H_k^{-1} e_1 = -\frac{1}{h_{k+1,k} \vartheta_{1,k+1}}, \quad [y_k^O]_k = -\frac{\|r_0\|}{h_{k+1,k} \vartheta_{1,k+1}},$$

and the Q-OR residual vector is

$$r_k^O = \frac{\|r_0\|}{\vartheta_{1,k+1}} v_{k+1}.$$

Since v_{k+1} is of unit norm, this shows that

$$|\vartheta_{1,k+1}| = \frac{\|r_0\|}{\|r_k^O\|},$$

see also Theorem 3.12. For the Q-MR method, we have the following result which shows how the residual vector depends on all the previous basis vectors.

Theorem 3.6 *Let $\vartheta_{1,j}$'s be the entries of the first row of U_{k+1}^{-1} and $z_k^M = \|r_0\| e_1 - H_k y_k^M$ be the quasi-residual vector. Then, the Q-MR residual vector is*

$$r_k^M = \frac{\|z_k^M\|^2}{\|r_0\|} V_{n,k+1} \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,k+1} \end{pmatrix}. \quad (3.21)$$

Proof The residual vector is given by

$$r_k^M = V_{n,k+1}(\|r_0\|e_1 - \underline{H}_k y_k^M).$$

Let us assume that H_k is nonsingular. From Theorem 3.2 we have

$$z_k^M = [y_k^O]_k \gamma \begin{pmatrix} H_k^{-*} e_k \\ -\frac{1}{h_{k+1,k}} \end{pmatrix},$$

with

$$\gamma = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}, \quad [y_k^O]_k = -\frac{\|r_0\|}{h_{k+1,k} \bar{\vartheta}_{1,k+1}}.$$

Let us consider $H_k^{-*} e_k$. From the decomposition of Theorem 2.5 we have

$$\begin{aligned} H_k^{-*} e_k &= U_k^{-*} [C^{(k)}]^{-*} U_k^* e_k, \\ &= u_{k,k} U_k^{-*} [C^{(k)}]^{-*} e_k, \\ &= -\frac{u_{k,k}}{\bar{\alpha}_0^{(k)}} U_k^{-*} e_1. \end{aligned}$$

We have already seen that

$$\frac{u_{k,k}}{\bar{\alpha}_0^{(k)}} = \frac{u_{k,k}}{u_{k+1,k+1} \bar{\vartheta}_{1,k+1}} = \frac{1}{h_{k+1,k} \bar{\vartheta}_{1,k+1}},$$

and we have

$$U_k^{-*} e_1 = \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,k} \end{pmatrix}.$$

Therefore,

$$H_k^{-*} e_k = -\frac{1}{h_{k+1,k} \bar{\vartheta}_{1,k+1}} \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,k} \end{pmatrix}.$$

Even though this does not fit in our general notation rules, to simplify the notation let $\bar{\vartheta}$ be the vector $(\bar{\vartheta}_{1,1} \dots \bar{\vartheta}_{1,k+1})^T$. Then,

$$\begin{aligned}
z_k^M &= -[y_k^O]_k \gamma \frac{1}{h_{k+1,k} \bar{\vartheta}_{1,k+1}} \bar{\vartheta}, \\
&= \frac{\|r_k^O\|^2}{\|r_0\|} \frac{\gamma}{h_{k+1,k}^2} \bar{\vartheta}, \\
&= \frac{1}{\|r_0\|} \|z_k^M\|^2 \bar{\vartheta},
\end{aligned}$$

since $\gamma/h_{k+1,k}^2 = \|z_k^M\|^2/\|r_k^O\|^2$. \square

Hence, up to a multiplying factor, the Q-MR residual vectors are linear combinations of the basis vectors with the conjugates of the $\vartheta_{1,j}$'s as coefficients. In the real case we have $\vartheta_{1,j} = \pm \|r_0\|/\|r_{j-1}^O\|$. The Q-OR residual norms are, in general, not monotonically decreasing. They can even become large. From Theorem 3.6 we see that the basis vectors which are the most important ones for the Q-MR residual vector are those for which the corresponding Q-OR residual norm is small. If the Q-OR residual norms are decreasing, the most important basis vectors are the last ones. If a Q-OR residual norm is large, the corresponding basis vector “almost disappear” from the basis.

Note that the result of Theorem 3.6 is still valid if H_k is singular because then $\|z_k^M\| = \|z_{k-1}^M\|$ and $\vartheta_{1,k+1} = 0$ which implies $r_k^M = r_{k-1}^M$, that is, stagnation of the Q-MR method.

We will see later in this chapter (Section 3.8, Corollary 3.5) that $\|z_k^M\|^2$ can be written as a function of the eigenvalues and eigenvectors of A as well as the right-hand side b .

We now consider the so-called *residual polynomials*. The relation (3.1) shows that the basis vectors v_j can be expressed as polynomials in A applied to the initial residual vector r_0 . It implies that the residual vectors r_k^O in Q-OR and r_k^M in Q-MR can be expressed as polynomials of degree k in A applied to the initial residual r_0 ,

$$r_k^O = p_k^O(A)r_0, \quad r_k^M = p_k^M(A)r_0.$$

These polynomials have the value 1 at 0 by definition of the residual vector. To study the residual polynomials we first proceed as in [440].

Proposition 3.2 *For any polynomial s_k of exact degree k with leading coefficient γ_k we have*

$$s_k(A)r_0 = \|r_0\| [V_{n,k} s_k(H_k)e_1 + \zeta_k \gamma_k v_{k+1}], \quad (3.22)$$

where $\zeta_k = h_{k+1,k} \cdots h_{2,1}$.

Proof We prove the result by induction using the Arnoldi-like relation (3.2). Since $e_k^T e_1 = 0$, we have

$$AV_{n,k}e_1 = V_{n,k}H_k e_1.$$

Assume $A^{j-1}V_{n,k}e_1 = V_{n,k}H_k^{j-1}e_1$, $j < k$. Then,

$$\begin{aligned} A^j V_{n,k} e_1 &= A(A^{j-1}V_{n,k}e_1), \\ &= AV_{n,k}H_k^{j-1}e_1, \\ &= V_{n,k}H_k^j e_1 + h_{k+1,k}v_{k+1}e_k^T H_k^{j-1}e_1, \\ &= V_{n,k}H_k^j e_1, \end{aligned}$$

since $H_k^{j-1}e_1$ is zero below the j th position. This does not hold for $j = k$. But the k th entry of $H_k^{k-1}e_1$ is $\prod_{j=1}^{k-1} h_{j+1,j}$, hence

$$A^k V_{n,k} e_1 = V_{n,k}H_k^k e_1 + \left[\prod_{j=1}^k h_{j+1,j} \right] v_{k+1}.$$

The fact that $V_{n,k}e_1 = v_1 = r_0/\|r_0\|$ yields the result. \square

By choosing appropriately the polynomial s_k in Proposition 3.2 we obtain a characterization of the Q-OR residual polynomial.

Theorem 3.7 *The residual polynomial p_k^O of the Q-OR method is proportional to the characteristic polynomial of H_k .*

Proof We have seen that $r_k^O = -h_{k+1,k}[y_k^O]_k v_{k+1}$. Hence, from relation (3.22) a necessary and sufficient condition for s_k to be proportional to the residual polynomial is to satisfy $s_k(H_k)e_1 = 0$. From Theorem 2.5 H_k can be factored as $H_k = U_k C^{(k)} U_k^{-1}$. Then, since $U_k^{-1}e_1 = e_1$,

$$s_k(H_k)e_1 = U_k s_k(C^{(k)}) U_k^{-1} e_1 = U_k s_k(C^{(k)}) e_1 = 0.$$

The first column of a power of the companion matrix $C^{(k)}$ is $[C^{(k)}]^j e_1 = e_{j+1}$ for $0 \leq j < k$, and the vector $[C^{(k)}]^k e_1$ is the negative of the vector of the coefficients $a_j^{(k)}$ of the characteristic polynomial of H_k . Let γ_j be the coefficients of s_k . Since $s_k(C^{(k)})e_1 = 0$, we have

$$\begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{k-1} \end{pmatrix} - \gamma_k \begin{pmatrix} a_0^{(k)} \\ a_1^{(k)} \\ \vdots \\ a_{k-1}^{(k)} \end{pmatrix} = 0.$$

This shows that the Q-OR residual polynomial is proportional to the characteristic polynomial of H_k since their coefficients are proportional. We have to divide the latter by its constant term to obtain the residual polynomial, which must have the value 1 at the origin. \square

Theorem 3.7 shows that the roots of the k th Q-OR residual polynomial are the eigenvalues of H_k .

Let us now consider the Q-MR residual polynomial p_k^M . The vector of coefficients y_k^M for the k th Q-MR iterate is obtained, see (3.9), by solving

$$\underline{H}_k^* H_k y_k^M = (H_k^* H_k + h_{k+1}^2 e_k e_k^T) y_k^M = \|r_0\| H_k^* e_1.$$

Since we assume that H_k is nonsingular, we can define

$$\hat{H}_k = H_k + h_{k+1,k}^2 H_k^{-*} e_k e_k^T. \quad (3.23)$$

The second term in the right-hand side is only nonzero in the last column; hence, the matrix \hat{H}_k is upper Hessenberg and y_k^M can also be obtained as the solution of $\hat{H}_k y_k^M = \|r_0\| e_1$. For properties of the Q-MR residual polynomial p_k^M we first study some properties of \hat{H}_k .

Lemma 3.1 *The matrix \hat{H}_k defined in (3.23) satisfies the Arnoldi-like relation*

$$AV_{n,k} = V_{n,k} \hat{H}_k + h_{k+1,k} V_{n,k+1} \begin{pmatrix} -h_{k+1,k} H_k^{-*} e_k \\ 1 \end{pmatrix} e_k^T. \quad (3.24)$$

Proof Inserting $H_k = \hat{H}_k - h_{k+1,k}^2 H_k^{-*} e_k e_k^T$ in the relation (3.2) yields the result. \square

We can relate the second term in the right-hand side of relation (3.24) to the residual vector r_k^M .

Theorem 3.8 *Let*

$$\omega_k = \frac{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}{[y_k^O]_k}.$$

The matrix \hat{H}_k satisfies the relation

$$AV_{n,k} = V_{n,k} \hat{H}_k - \omega_k r_k^M e_k^T. \quad (3.25)$$

Proof With (3.13) we have

$$z_k^M = \gamma [y_k^O]_k \begin{pmatrix} H_k^{-*} e_k \\ -\frac{1}{h_{k+1,k}} \end{pmatrix},$$

where

$$\gamma = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2}.$$

Let us factor out $-\gamma/h_{k+1,k}$ and use $h_{k+1,k}^2/\gamma = 1 + h_{k+1,k}^2 \|H_k^{-*} e_k\|^2$ to obtain

$$z_k^M = -\frac{[y_k^O]_k \gamma}{h_{k+1,k}} \begin{pmatrix} -h_{k+1,k} H_k^{-*} e_k \\ 1 \end{pmatrix}.$$

We conclude by using relation (3.24) and noticing that $r_k^M = V_{n,k+1} z_k^M$. \square

Now we have the analog of Proposition 3.2.

Proposition 3.3 *For any polynomial s_k of exact degree k with leading coefficient γ_k , we have*

$$s_k(A)r_0 = \|r_0\| [V_{n,k} s_k(\hat{H}_k) e_1 - \xi_k \gamma_k r_k^M], \quad (3.26)$$

where $\xi_k = \omega_k h_{k,k-1} \cdots h_{2,1}$ and ω_k was defined in Theorem 3.8.

Proof The proof is similar to the proof of Proposition 3.2. We first remark that, because of the nonzero structure of H_k and \hat{H}_k , they differ only in their last column. Therefore $A^j V_{n,k} e_1 = V_{n,k} \hat{H}_k^j e_1$, $j = 1, \dots, k-1$, can be proved exactly as in Proposition 3.2 and

$$\begin{aligned} A^k V_{n,k} e_1 &= A \cdot A^{k-1} V_{n,k} e_1 = A(V_{n,k} \hat{H}_k^{k-1} e_1) \\ &= (V_{n,k} \hat{H}_k - \omega_k r_k^M e_k^T) \hat{H}_k^{k-1} e_1, \\ &= V_{n,k} \hat{H}_k^k e_1 - (e_k^T \hat{H}_k^{k-1} e_1) \omega_k r_k^M. \end{aligned}$$

Using that the k th entry of $\hat{H}_k^{k-1} e_1$ is $\prod_{j=1}^{k-1} h_{j+1,j}$ and using relation (3.25) ends the proof. \square

From the last Proposition, which holds for any polynomial of degree k , we obtain the following result for the Q-MR residual polynomial.

Theorem 3.9 *The residual polynomial p_k^M of the Q-MR method is proportional to the characteristic polynomial of \hat{H}_k .*

Proof Let us choose the polynomial s_k in relation (3.26) of Proposition 3.3 such that $s_k(\hat{H}_k) e_1 = 0$, then

$$s_k(A)r_0 = -\|r_0\| \xi_k \gamma_k r_k^M.$$

For the same reasons as in the proof of Theorem 3.7, this shows that the Q-MR residual polynomial is proportional to the characteristic polynomial of \hat{H}_k . \square

Theorem 3.9 shows that the roots of the k th Q-MR residual polynomial are the eigenvalues of \hat{H}_k . Our proof is completely different from the one in [440] where

this type of result was proved for GMRES. As we see it holds for any Q-MR method, as long as H_k is nonsingular.

We can relate the characteristic polynomial of \hat{H}_k to the characteristic polynomial of H_k . First, we consider the factorization of \hat{H}_k . This result was first proved in [684].

Theorem 3.10 *The matrix \hat{H}_k can be written as $\hat{H}_k = U_k \hat{C}^{(k)} U_k^{-1}$, U_k being upper triangular and*

$$\hat{C}^{(k)} = C^{(k)} - \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)}} U_k^{-1} U_k^{-*} e_1 e_k^T,$$

being a companion matrix where $C^{(k)}$ is the companion matrix in the decomposition $H_k = U_k C^{(k)} U_k^{-1}$ in Theorem 2.5 and $\alpha_0^{(k)}$ is the constant term in the characteristic polynomial of H_k . Moreover, if $(\beta_0^{(k)} \dots \beta_{k-1}^{(k)})^T$ are the coefficients of the characteristic polynomial of \hat{H}_k , then we have

$$M_k \begin{pmatrix} \beta_0^{(k)} \\ \vdots \\ \beta_{k-1}^{(k)} \end{pmatrix} = -M_{1:k,k+1} + \delta_{k+1} e_1, \quad (3.27)$$

where $M_k = U_k^* U_k$ and

$$\delta_{k+1} = \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)}} = \frac{1}{(M_{k+1}^{-1})_{k+1,1}}. \quad (3.28)$$

Proof Let us first consider $H_k^{-*} e_k$. We have $H_k^{-*} = U_k^{-*} [C^{(k)}]^{-*} U_k^*$. Using the inverse of $C^{(k)}$ (refer to relation (2.14)) we see that

$$H_k^{-*} e_k = -\frac{u_{k,k}}{\alpha_0^{(k)}} U_k^{-*} e_1. \quad (3.29)$$

On the other hand, we have $h_{k+1,k} = u_{k+1,k+1}/u_{k,k}$, see Theorem 2.3. Since $u_{1,1} = 1$ and $h_{k+1,k}$ is real and positive, this implies that all the diagonal entries of U_j are real and positive for all indices j as we have seen above. Then, using the definition of \hat{H}_k ,

$$\hat{H}_k = U_k C^{(k)} U_k^{-1} - \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)} u_{k,k}} U_k^{-*} e_1 e_k^T.$$

Let us factor U_k on the left and U_k^{-1} on the right. We obtain

$$\hat{H}_k = U_k [C^{(k)} - \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)} u_{k,k}} U_k^{-1} U_k^{-*} e_1 e_k^T U_k] U_k^{-1}.$$

We remark that $e_k^T U_k = u_{k,k} e_k^T$. Hence \hat{H}_k is similar to the matrix

$$\hat{C}^{(k)} = C^{(k)} - \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)}} U_k^{-1} U_k^{-*} e_1 e_k^T.$$

The second term on the right-hand side modifies only the last column. Therefore, $\hat{C}^{(k)}$ is a companion matrix, and the coefficients of the characteristic polynomial of \hat{H}_k are given by the negative of the last column of $\hat{C}^{(k)}$. We remark that

$$u_{k+1,k+1}^2 = \frac{1}{(M_{k+1}^{-1})_{k+1,k+1}}.$$

From Theorem 2.5 we have

$$\begin{pmatrix} \beta_0^{(k)} \\ \vdots \\ \beta_{k-1}^{(k)} \end{pmatrix} = -U_k^{-1} U_{1:k,k+1} + \delta_{k+1} U_k^{-1} U_k^{-*} e_1, \quad \delta_{k+1} = \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)}}.$$

We have, see (2.19),

$$\delta_{k+1} = \frac{u_{k+1,k+1}^2}{\alpha_0^{(k)}} = \frac{u_{k+1,k+1}^2}{u_{k+1,k+1} e_1^T U_{k+1}^{-1} e_{k+1}} = \frac{1}{(M_{k+1}^{-1})_{k+1,1}}.$$

Multiplying both sides by $M_k = U_k^* U_k$ we obtain

$$M_k \begin{pmatrix} \beta_0^{(k)} \\ \vdots \\ \beta_{k-1}^{(k)} \end{pmatrix} = -M_{1:k,k+1} + \delta_{k+1} e_1,$$

which proves relation (3.27). \square

Note that if A is real, the coefficients $\alpha_0^{(k)}$ and δ_{k+1} are real. We observe that, compared to the linear system (2.20) for the characteristic polynomial of H_k , only the first entry of the right-hand side is modified.

Corollary 3.1 *The coefficients $\beta_j^{(k)}$ of the characteristic polynomial of \hat{H}_k are related to the coefficients $\alpha_j^{(k)}$ of the characteristic polynomial of H_k by*

$$\begin{pmatrix} \beta_0^{(k)} \\ \vdots \\ \beta_{k-1}^{(k)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} + \delta_{k+1} M_k^{-1} e_1. \quad (3.30)$$

Proof The proof is straightforward from Theorem 3.10. \square

We remark that the decomposition in Theorem 2.5 can also be used for an expression of the entry $[y_k^O]_k$ which played an important role in the previous sections: From (3.6) we have that

$$y_k^O = \|r_0\| H_k^{-1} e_1 = \|r_0\| U_k [C^{(k)}]^{-1} U_k^{-1} e_1 = \|r_0\| U_k [C^{(k)}]^{-1} e_1.$$

By considering the inverse of the companion matrix $C^{(k)}$ (relation (2.14)), we obtain

$$[y_k^O]_k = -\frac{\|r_0\| u_{k,k}}{\alpha_0^{(k)}}, \quad (3.31)$$

and since $\alpha_0^{(k)}$ is the trailing coefficient in the characteristic polynomial of H_k ,

$$[y_k^O]_k = -\frac{\|r_0\| \prod_{j=1}^{k-1} h_{j+1,j}}{\det(H_k)}.$$

Note that we can relate this result to relation (2.22) in Lemma 2.3.

Now, we show how the coefficients of the residual and characteristic polynomials arising in Q-OR and Q-MR methods can be expressed using the moment matrices $U_j^* U_j$.

Theorem 3.11 *Let $M_j = U_j^* U_j$, $j = 1, \dots, k+1$. The coefficients of the characteristic polynomial of H_k (in ascending order of powers) are given by*

$$\alpha_{j-1}^{(k)} = \frac{(M_{k+1}^{-1})_{j,1} - (M_k^{-1})_{j,1}}{(M_{k+1}^{-1})_{k+1,1}}, \quad j = 1, \dots, k, \quad \alpha_k^{(k)} = 1. \quad (3.32)$$

Moreover, the coefficients of the Q-OR residual polynomial (in ascending order of powers) at iteration k are

$$\frac{\alpha_{j-1}^{(k)}}{\alpha_0^{(k)}} = \frac{(M_{k+1}^{-1})_{j,1} - (M_k^{-1})_{j,1}}{(M_{k+1}^{-1})_{1,1} - (M_k^{-1})_{1,1}}, \quad j = 1, \dots, k, \quad \frac{1}{\alpha_0^{(k)}} = \frac{(M_{k+1}^{-1})_{k+1,1}}{(M_{k+1}^{-1})_{1,1} - (M_k^{-1})_{1,1}}.$$

The coefficients of the characteristic polynomial of \hat{H}_k (in ascending order of powers) are given by

$$\beta_{j-1}^{(k)} = \frac{(M_{k+1}^{-1})_{j,1}}{(M_{k+1}^{-1})_{k+1,1}}, \quad j = 1, \dots, k, \quad \beta_k^{(k)} = 1. \quad (3.33)$$

The coefficients of the Q-MR residual polynomial (in ascending order of powers) at iteration k are

$$\frac{\beta_{j-1}^{(k)}}{\beta_0^{(k)}} = \frac{(M_{k+1}^{-1})_{j,1}}{(M_{k+1}^{-1})_{1,1}}, \quad j = 1, \dots, k, \quad \frac{1}{\beta_0^{(k)}} = \frac{(M_{k+1}^{-1})_{k+1,1}}{(M_{k+1}^{-1})_{1,1}}.$$

Proof Let us consider the first column of the inverse of M_{k+1} , given by the solution of the linear system,

$$M_{k+1}x = \begin{pmatrix} M_k & M_{1:k,k+1} \\ (M_{1:k,k+1})^* & m_{k+1,k+1} \end{pmatrix} \begin{pmatrix} \tilde{x} \\ x_{k+1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

It yields the two equations

$$\begin{aligned} M_k \tilde{x} + x_{k+1} M_{1:k,k+1} &= e_1, \\ (M_{1:k,k+1})^* \tilde{x} + x_{k+1} m_{k+1,k+1} &= 0. \end{aligned}$$

Solving the first equation for \tilde{x} we have

$$\tilde{x} = M_k^{-1}(e_1 - x_{k+1} M_{1:k,k+1})$$

and multiplying by e_j^T , $j = 1, \dots, k$ we obtain, using (2.20),

$$\begin{aligned} e_j^T \tilde{x} &= (M_{k+1}^{-1})_{j,1} = (M_k^{-1})_{j,1} - (M_{k+1}^{-1})_{k+1,1} e_j^T M_k^{-1} M_{1:k,k+1}, \\ &= (M_k^{-1})_{j,1} + (M_{k+1}^{-1})_{k+1,1} \alpha_{j-1}^{(k)}, \end{aligned}$$

since $x_{k+1} = (M_{k+1}^{-1})_{k+1,1}$. The last relation proves (3.32) and the values for $\alpha_{j-1}^{(k)} \neq 0$ after division with $\alpha_0^{(k)}$ (we assume H_k is nonsingular, thus $\alpha_0^{(k)} \neq 0$). Finally, we have, using (3.30) and (3.28),

$$\begin{aligned} \beta_{j-1}^{(k)} &= \alpha_{j-1}^{(k)} + \delta_{k+1}(M_k^{-1})_{j,1}, \\ &= \frac{(M_{k+1}^{-1})_{j,1} - (M_k^{-1})_{j,1}}{(M_{k+1}^{-1})_{k+1,1}} + \frac{(M_k^{-1})_{j,1}}{(M_{k+1}^{-1})_{k+1,1}}, \\ &= \frac{(M_{k+1}^{-1})_{j,1}}{(M_{k+1}^{-1})_{k+1,1}}, \end{aligned}$$

giving all the relations for the $\beta_j^{(k)}$'s in the claim. \square

3.4 The inverse of U_k and the residual norms

From Chapter 2, Theorem 2.5, we know that each of the upper Hessenberg matrices H_k can be factored as $H_k = U_k C^{(k)} U_k^{-1}$ where U_k is an upper triangular matrix and $C^{(k)}$ a companion matrix. We now show that the Q-OR residual norms and the Q-MR quasi-residual norms are related to the entries of the inverse of U_k . In the following, let $\vartheta_{i,j}$ be the entries of the inverse of the matrix U_k .

Theorem 3.12 *Assume that the basis vectors v_j are of unit norm. The relative residual norms of the Q-OR method are*

$$\frac{\|r_k^O\|}{\|r_0\|} = \frac{1}{|\vartheta_{1,k+1}|}. \quad (3.34)$$

The relative quasi-residual norms of the Q-MR method are

$$\frac{\|z_k^M\|}{\|r_0\|} = \frac{1}{\left[\sum_{j=1}^{k+1} |\vartheta_{1,j}|^2\right]^{\frac{1}{2}}}. \quad (3.35)$$

Proof We have already proved the first claim in Section 3.3. But let us consider it again. With Corollary 2.1 the coefficients of the characteristic polynomial $\lambda^k + \alpha_{k-1}^{(k)} \lambda^{k-1} + \cdots + \alpha_0^{(k)}$ of H_k are

$$\begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \end{pmatrix} = u_{k+1,k+1} [U_{k+1}^{-1}]_{1:k,k+1} = u_{k+1,k+1} \begin{pmatrix} \vartheta_{1,k+1} \\ \vdots \\ \vartheta_{k,k+1} \end{pmatrix}.$$

Then, from (3.31),

$$[y_k^O]_k = -\frac{\|r_0\| u_{k,k}}{\alpha_0^{(k)}} = -\|r_0\| \frac{u_{k,k}}{u_{k+1,k+1} \vartheta_{1,k+1}} = -\|r_0\| \frac{1}{\vartheta_{1,k+1} h_{k+1,k}}.$$

Here we assume that H_k is nonsingular and therefore, that the k th Q-OR residual exists. Hence, since $\|r_k^O\| = h_{k+1,k} |[y_k^O]_k|$, we have for the iterations k where the Q-OR iterates exist that

$$\frac{\|r_k^O\|}{\|r_0\|} = \frac{1}{|\vartheta_{1,k+1}|}.$$

If H_k is singular, $\alpha_0^{(k)} = 0$, $\|r_k^O\| = \infty$ and with $\vartheta_{1,k+1} = 0$ the result is still true. For the relation (3.35) we use (3.17) for all the iterations,

$$\begin{aligned} \frac{|\vartheta_{1,k+1}|^2}{\|r_0\|^2} &= \frac{1}{\|z_k^M\|^2} - \frac{1}{\|z_{k-1}^M\|^2}, \\ \frac{|\vartheta_{1,k}|^2}{\|r_0\|^2} &= \frac{1}{\|z_{k-1}^M\|^2} - \frac{1}{\|z_{k-2}^M\|^2}, \\ &\vdots = \vdots \\ \frac{|\vartheta_{1,2}|^2}{\|r_0\|^2} &= \frac{1}{\|z_1^M\|^2} - \frac{1}{\|z_0^M\|^2}. \end{aligned}$$

Summing up all these relations we obtain

$$\frac{1}{\|z_k^M\|^2} - \frac{1}{\|z_0^M\|^2} = \frac{1}{\|r_0\|^2} [|\vartheta_{1,k+1}|^2 + \dots + |\vartheta_{1,2}|^2].$$

But noticing that $\|z_0^M\| = \|r_0\|$ and $\vartheta_{1,1} = 1$ gives the result. \square

The first result of Theorem 3.12 was proved with a more complicated proof in [289]. This is a remarkable result which is true for any Q-OR or Q-MR method as long as the basis vectors are of unit norm. It shows once again why we have the peak-plateau phenomenon, if $\vartheta_{1,k+1} = 0$ and H_k is singular, the Q-MR quasi-residual norm stagnates. This result also shows that without stagnation the quasi-residual norm is strictly decreasing.

From Theorem 3.12 we immediately obtain the residual norms as functions of the moment matrices M_j . It is a generalization of a well-known result for GMRES; see [291] and the references therein.

Corollary 3.2 *Assume that the basis vectors v_j are of unit norm. Let M_j be defined as $M_j = U_j^* U_j$. Then the norms of the Q-OR residual and of the Q-MR quasi-residual vectors are given by*

$$\frac{\|r_k^O\|^2}{\|r_0\|^2} = \frac{1}{(M_{k+1}^{-1})_{1,1} - (M_k^{-1})_{1,1}}, \quad \frac{\|z_k^M\|^2}{\|r_0\|^2} = \frac{1}{(M_{k+1}^{-1})_{1,1}}. \quad (3.36)$$

Proof Since $M_{k+1}^{-1} = U_{k+1}^{-1} U_{k+1}^{-*}$ is the product of an upper triangular matrix and a lower triangular matrix, the $(1, 1)$ entry of $(M_{k+1}^{-1})_{1,1}$ is

$$(M_{k+1}^{-1})_{1,1} = \sum_{j=1}^{k+1} |\vartheta_{1,j}|^2 = \frac{\|r_0\|^2}{\|z_k^M\|^2}.$$

For the other relation we remark that

$$|\vartheta_{1,k+1}|^2 = \sum_{j=1}^{k+1} |\vartheta_{1,j}|^2 - \sum_{j=1}^k |\vartheta_{1,j}|^2$$

and use (3.34). \square

Theorem 3.12 allows to obtain a factorization of the matrices U_k^{-1} involving the Q-OR residual norms and the coefficients of the residual polynomial.

Theorem 3.13 Assume that the basis vectors v_j are of unit norm and $\vartheta_{1,k+1} \neq 0$. The matrix U_{k+1}^{-1} can be scaled as

$$U_{k+1}^{-1} = \|r_0\| \hat{U}_{k+1}^{-1} D_{k+1}, \quad (3.37)$$

with the diagonal matrix

$$D_{k+1} = \begin{pmatrix} \frac{1}{\|r_0\|} & & & \\ & \frac{e^{i\phi_2}}{\|r_1^O\|} & & \\ & & \ddots & \\ & & & \frac{e^{i\phi_{k+1}}}{\|r_k^O\|} \end{pmatrix},$$

where ϕ_j are the arguments of the (possibly) complex numbers $\vartheta_{1,j}$ and the columns of the upper triangular matrix \hat{U}_{k+1}^{-1} are the coefficients of the residual polynomials from 0 up to k , that is, for instance,

$$[\hat{U}_{k+1}^{-1}]_{1:k+1,k+1} = \begin{pmatrix} 1 \\ \alpha_1^{(k)}/\alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)}/\alpha_0^{(k)} \\ 1/\alpha_0^{(k)} \end{pmatrix},$$

where $\alpha_j^{(k)}$, $j = 0, \dots, k-1$ are the coefficients of the characteristic polynomial of H_k .

Proof We have from (2.19) that

$$\begin{pmatrix} \alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)} \\ 1 \end{pmatrix} = u_{k+1,k+1} \begin{pmatrix} \vartheta_{1,k+1} \\ \vdots \\ \vartheta_{k,k+1} \\ \vartheta_{k+1,k+1} \end{pmatrix}.$$

It yields

$$\begin{pmatrix} 1 \\ \alpha_1^{(k)}/\alpha_0^{(k)} \\ \vdots \\ \alpha_{k-1}^{(k)}/\alpha_0^{(k)} \\ 1/\alpha_0^{(k)} \end{pmatrix} = \frac{1}{\vartheta_{1,k+1}} \begin{pmatrix} \vartheta_{1,k+1} \\ \vdots \\ \vartheta_{k,k+1} \\ \vartheta_{k+1,k+1} \end{pmatrix}.$$

The matrix in the right-hand side is the $k + 1$ st column of U_{k+1}^{-1} and we can write

$$\frac{1}{\vartheta_{1,k+1}} = \frac{1}{|\vartheta_{1,k+1}| e^{i\phi_{k+1}}} = \frac{\|r_k^O\|}{\|r_0\|} e^{-i\phi_{k+1}}.$$

This gives the result, and the nonzero entries of the columns of \hat{U}_{k+1}^{-1} are the coefficients of the successive residual polynomials whose roots are the eigenvalues of the matrices H_j , $j = 1, \dots, k$. \square

We remark that if the matrix A , the right-hand side and the starting vector are real, the coefficients $e^{i\phi_j}$ are just ± 1 .

The entries of the inverse of U_k can also be used for an alternative to the relation between iterates (3.12) and to the analog relation between errors.

Theorem 3.14 *The iterates of the Q-OR and Q-MR methods are related by*

$$x_k^M = x_k^O - \frac{\|z_k^M\|^2}{\|r_0\|} V_{n,k} H_k^{-1} \begin{pmatrix} \frac{1}{\vartheta_{1,2}} \\ \vdots \\ \frac{1}{\vartheta_{1,k}} \end{pmatrix}. \quad (3.38)$$

The error vectors ε_k^M and ε_k^O satisfy the same relation if they, respectively, replace x_k^M and x_k^O .

Proof With relation (3.12) we have

$$x_k^M = x_k^O - \gamma [y_k^O]_k V_{n,k} H_k^{-1} H_k^{-*} e_k.$$

But, using (3.29) and (3.31),

$$H_k^{-*} e_k = -\frac{u_{k,k}}{\alpha_0^{(k)}} U_k^{-*} e_1 = \frac{\overline{[y_k^O]_k}}{\|r_0\|} U_k^{-*} e_1.$$

We have from the proof of Proposition 3.1 that $\|z_k^M\|^2 = \gamma \| [y_k^O]_k \|^2$. Noticing that the entries in the first column of U_k^{-*} are $\overline{\vartheta_{1,j}}$, $j = 1, \dots, k$ we obtain the result. For the error vectors we just subtract the exact solution from both sides of (3.38). \square

3.5 Prescribing the convergence curve

Hundreds of papers have been written about the convergence of Krylov methods for nonsymmetric matrices. Most of them are concerned with upper bounds of the norm of the residual and an important number attempts to relate residual norm bounds to spectral properties of the matrix, which explain convergence speed well when the matrix is normal. However, in this section and in the next chapters, we will show that any convergence curve is possible for the (relative) residual norms of Q-OR methods and any non-increasing convergence curve is possible for the (relative) quasi-residual norms of Q-MR methods. Moreover, as long as A does not have to satisfy other constraints (like being symmetric), the eigenvalues of the matrix A can be prescribed for any convergence curve as well as the eigenvalues of the matrices H_k and \hat{H}_k in all iteration numbers k .

For simplicity we consider convergence curves where the residual norms are nonzero for $k = 1, \dots, n - 1$. This implies that the method does not stop before iteration n . For a Q-OR method with orthonormal bases for the Krylov subspaces (a true OR method) it is not difficult to prove the following theorem using results of previous sections.

Theorem 3.15 *Let f_j , $j = 0, \dots, n - 1$ be n nonzero positive numbers, λ_j , $j = 1, \dots, n$ be n nonzero complex numbers and $\theta_i^{(j)}$, $j = 1, \dots, n - 1$ and $i = 1, \dots, j$ be given nonzero complex numbers. For any unitary matrix V we can define the right-hand side $b = f_0 V e_1$ and a matrix A such that when applying an OR method for solving $Ax = b$ with $x_0 = 0$, we have*

$$\frac{\|r_k^O\|}{\|r_0\|} = f_k, \quad k = 1, \dots, n - 1,$$

the eigenvalues of A are λ_j , $j = 1, \dots, n$ and the eigenvalues of H_k , $k = 1, \dots, n - 1$ are $\theta_i^{(k)}$, $i = 1, \dots, k$.

Proof Let V be any unitary matrix and $b = f_0 V e_1$. For the construction of the matrix A we use the results of Theorem 3.12 and Theorem 3.13. We first construct the upper triangular matrix $U^{-1} = f_0 \hat{U}^{-1} D$. The diagonal entries of the diagonal matrix D are chosen as $d_{1,1} = 1/f_0$ and $d_{j,j} = 1/f_{j-1}$. Let $[\hat{U}^{-1}]_{1,1} = 1$. For each $j = 1, \dots, n - 1$ we construct the coefficients of the polynomial whose value at the origin is 1 and the roots are the given values $\theta_i^{(j)}$, $i = 1, \dots, j$. These coefficients (starting with the constant term 1 for the first row) are the entries of the j th column of \hat{U}^{-1} . Let C be the companion matrix corresponding to a monic polynomial with λ_j , $j = 1, \dots, n$ as roots. Then, we define the unreduced upper Hessenberg matrix

$$H = U C U^{-1}.$$

Finally, the matrix A is defined as $A = V H V^*$. Clearly, we have $AV = VH$ and this yields Arnoldi relations (3.2). Note that the constructed matrix A is non-derogatory. \square

Instead of the OR residual norms we can prescribe the MR residual norms by using relation (3.17) and Theorem 3.15. But, of course, the prescribed values have to be decreasing. In Theorem 3.15 we use a unitary matrix V but it provides as well a general framework for Q-OR/Q-MR methods using non-orthonormal bases. We will see later how to choose the matrix V when we consider particular instances of Q-OR and Q-MR methods with non-orthonormal bases. We postpone this presentation because we will have to be particularly careful when the matrix H must have more properties than just being upper Hessenberg. For instance, in some methods, H is required to be tridiagonal. Hence, the construction of A cannot be completely general.

Results like Theorem 3.15 show that everything can happen with Krylov methods for nonsymmetric matrices. We can construct examples with very good or very bad convergence of the residual norms, and this can be done independently of what are the chosen eigenvalues of the matrix. This type of independence on eigenvalues does not hold for the standard Krylov methods for a symmetric or normal matrix A . Moreover, when the sought matrix is constructed, the prescribed convergence curve occurs only for one particular right-hand side. This suggests that results on convergence obtained without considering the right-hand side cannot be very general.

In addition, Theorem 3.15 shows that in an OR method, the roots of the residual polynomials can be prescribed in all iterations, because they are the eigenvalues of the submatrices H_k . It may seem a little counterintuitive that the roots of p_k^O where $r_k^O = p_k^O(A)r_0$ can be prescribed independent of $\|r_k^O\|$. But this becomes clear when we realize that the k th column of U_k^{-1} contains the coefficients of the residual polynomial and that they can be scaled (without changing the corresponding roots) in order to fix the first entry of that column, which determines $\|r_k^O\|$. A natural question is whether the roots of the residual polynomials can be prescribed for a Q-MR method as well. The relation between the submatrices U_k^{-1} and the coefficients of a Q-MR residual polynomial was given implicitly in Corollary 3.1 and can be used to show that we can fix the columns of U_k^{-1} to obtain prescribed roots of Q-MR residual polynomials.

Lemma 3.2 *Let the entries $\vartheta_{j,k+1}$ of the $k + 1$ st column of U_{k+1}^{-1} satisfy, for given complex numbers $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$,*

$$\begin{aligned}\vartheta_{k+1,k+1} &\geq \frac{2\|U_k^{-*}e_1\|}{|\beta_0^{(k)}|}, \\ |\vartheta_{1,k+1}|^2 - \vartheta_{k+1,k+1}|\beta_0^{(k)}| |\vartheta_{1,k+1}| + \|U_k^{-*}e_1\|^2 &= 0, \\ \frac{\vartheta_{1,k+1}}{|\vartheta_{1,k+1}|} &= \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|}, \\ \vartheta_{j,k+1} &= \beta_{j-1}^{(k)}\vartheta_{k+1,k+1} - \frac{e_j^T U_k^{-1} U_k^{-*} e_1}{\vartheta_{1,k+1}}, \quad j = 2, \dots, k.\end{aligned}$$

Then if a Q-MR method generates in the $k + 1$ st iteration the submatrix U_{k+1}^{-1} in the triangular Hessenberg decomposition $H_{k+1} = U_{k+1} C^{(k+1)} U_{k+1}^{-1}$, the roots of the k th Q-MR residual polynomial are those of the monic polynomial with coefficients $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$ (in ascending order of powers).

Proof The results of Corollary 3.1 can be written, using Theorem 3.10, as

$$\begin{pmatrix} \beta_0^{(k)} \\ \vdots \\ \beta_{k-1}^{(k)} \end{pmatrix} = \frac{1}{\vartheta_{k+1,k+1}} \begin{pmatrix} \vartheta_{1,k+1} \\ \vdots \\ \vartheta_{k,k+1} \end{pmatrix} + \frac{1}{\vartheta_{k+1,k+1} \vartheta_{1,k+1}} (U_k^* U_k)^{-1} e_1. \quad (3.39)$$

Remember that $\vartheta_{k+1,k+1}$ is real. If we equate the first row, we obtain

$$\beta_0^{(k)} = \frac{\vartheta_{1,k+1}}{\vartheta_{k+1,k+1}} + \frac{\|U_k^{-*} e_1\|^2}{\vartheta_{k+1,k+1} \vartheta_{1,k+1}} = \vartheta_{1,k+1} \left(\frac{1}{\vartheta_{k+1,k+1}} + \frac{\|U_k^{-*} e_1\|^2}{\vartheta_{k+1,k+1} |\vartheta_{1,k+1}|^2} \right),$$

and the factor between brackets on the right-hand side being real, $\beta_0^{(k)}$ and $\vartheta_{1,k+1}$ must have the same phase angle, i.e.,

$$\frac{\vartheta_{1,k+1}}{|\vartheta_{1,k+1}|} = \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|}.$$

As for the length of $\vartheta_{1,k+1}$, taking absolute values leads to the quadratic equation

$$|\vartheta_{1,k+1}|^2 - \vartheta_{k+1,k+1} |\beta_0^{(k)}| |\vartheta_{1,k+1}| + \|U_k^{-*} e_1\|^2 = 0,$$

which can be satisfied for positive real values $|\vartheta_{1,k+1}|$ if

$$|\beta_0^{(k)}|^2 \geq \frac{4 \|U_k^{-*} e_1\|^2}{\vartheta_{k+1,k+1}^2}.$$

With these choices of $\vartheta_{1,k+1}$ and $\vartheta_{k+1,k+1}$, the last claim follows by equating rows two till k in (3.39). \square

The lemma shows that there is some freedom in the choice of the entries of U_{k+1}^{-1} if we want to force the roots of the Q-MR residual polynomial. The positive real diagonal entry $\vartheta_{k+1,k+1}$ must satisfy a constraint (the first claim) such that the roots of the involved quadratic equation (in the second claim) are real. The size of $\vartheta_{1,k+1}$ can then be chosen either as the smallest or as the largest root of that equation. We can use this freedom to prescribe residual norms in the corresponding Q-OR method.

Corollary 3.3 Let the entries $\vartheta_{j,k+1}$ of the $k + 1$ st column of U_{k+1}^{-1} satisfy, for a given real positive number f_k and complex numbers $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$,

$$\begin{aligned}\vartheta_{k+1,k+1} &= \frac{1 + f_k^2 \|U_k^{-*} e_1\|^2}{f_k |\beta_0^{(k)}|}, \\ |\vartheta_{1,k+1}| &= 1/f_k, \\ \frac{\vartheta_{1,k+1}}{|\vartheta_{1,k+1}|} &= \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|}, \\ \vartheta_{j,k+1} &= \beta_{j-1}^{(k)} \vartheta_{k+1,k+1} - \frac{e_j^T U_k^{-1} U_k^{-*} e_1}{\vartheta_{1,k+1}}, \quad j = 2, \dots, k.\end{aligned}$$

Then if a Q-OR method generates in the $k+1$ st iteration the submatrix U_{k+1}^{-1} in the triangular Hessenberg decomposition $H_{k+1} = U_{k+1} C^{(k+1)} U_{k+1}^{-1}$, it generates the k th residual norm satisfying $\|r_k^Q\| = f_k$. The roots of the k th Q-MR residual polynomial for the corresponding Q-MR method are those of the monic polynomial with coefficients $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$ (in ascending order of powers).

Proof To have the roots of the k th Q-MR residual polynomial equal to those of the polynomial with coefficients $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$, the entries of U_{k+1}^{-1} must satisfy the conditions in Lemma 3.2. In particular, $|\vartheta_{1,k+1}|$ must be one of the two roots

$$|\vartheta_{1,k+1}| = \frac{\vartheta_{k+1,k+1} |\beta_0^{(k)}| \pm \sqrt{\vartheta_{k+1,k+1}^2 |\beta_0^{(k)}|^2 - 4 \|U_k^{-*} e_1\|^2}}{2}$$

of the corresponding quadratic equation. If, in addition, we wish to force the prescribed residual norm f_k for the corresponding Q-OR method, the entries on the first row of U_{k+1}^{-1} must satisfy the first condition in Theorem 3.12. For the k th residual norm,

$$|\vartheta_{1,k+1}| = 1/f_k.$$

Combining with the previous equation gives

$$2/f_k - \vartheta_{k+1,k+1} |\beta_0^{(k)}| = \pm \sqrt{\vartheta_{k+1,k+1}^2 |\beta_0^{(k)}|^2 - 4 \|U_k^{-*} e_1\|^2},$$

whence

$$\begin{aligned}\frac{4}{f_k^2} - \frac{4 \vartheta_{k+1,k+1} |\beta_0^{(k)}|}{f_k} + \vartheta_{k+1,k+1}^2 |\beta_0^{(k)}|^2 &= \vartheta_{k+1,k+1}^2 |\beta_0^{(k)}|^2 - 4 \|U_k^{-*} e_1\|^2, \\ \frac{1}{f_k^2} + \|U_k^{-*} e_1\|^2 &= \frac{|\beta_0^{(k)}|}{f_k} \vartheta_{k+1,k+1}, \\ \vartheta_{k+1,k+1} &= \frac{1 + f_k^2 \|U_k^{-*} e_1\|^2}{f_k |\beta_0^{(k)}|}.\end{aligned}$$

□

This value of $\vartheta_{k+1,k+1}$ satisfies the first condition in Lemma 3.2 because

$$\frac{1 + f_k^2 \|U_k^{-*} e_1\|^2}{f_k} \geq 2 \|U_k^{-*} e_1\|.$$

With this corollary we can formulate an analog of Theorem 3.15 involving the matrix \hat{H}_k defined in (3.23) (whose eigenvalues are the roots of the corresponding MR residual polynomial) instead of the matrix H_k (whose eigenvalues are roots of the OR residual polynomial).

Theorem 3.16 *Let f_j , $j = 0, \dots, n-1$ be n nonzero positive numbers, λ_j , $j = 1, \dots, n$ be n nonzero complex numbers and $\zeta_i^{(j)}$, $j = 1, \dots, n-1$ and $i = 1, \dots, j$ be given nonzero complex numbers. For any unitary matrix V we can define the right-hand side $b = f_0 V e_1$ and a matrix A such that when applying an OR method for solving $Ax = b$ with $x_0 = 0$, we have*

$$\frac{\|r_k^O\|}{\|r_0\|} = f_k, \quad k = 1, \dots, n-1,$$

the eigenvalues of A are λ_j , $j = 1, \dots, n$ and the eigenvalues of the matrices \hat{H}_k , $k = 1, \dots, n-1$ defined in (3.23) are $\zeta_i^{(k)}$, $i = 1, \dots, k$.

Proof We will use the same type of construction as in Theorem 3.15. Let V be any unitary matrix and $b = f_0 V e_1$. If we have constructed the appropriate upper triangular matrix U^{-1} ensuring that the prescribed relative residual norms and eigenvalues of \hat{H}_k are obtained, then we let C be the companion matrix corresponding to the monic polynomial with λ_j , $j = 1, \dots, n$ as roots. We define the unreduced upper Hessenberg matrix

$$H = U C U^{-1}$$

and the matrix A is defined as $A = V H V^*$.

Let us denote the coefficients of the monic polynomial whose roots are the given values $\zeta_i^{(k)}$, $i = 1, \dots, k$ with $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$. Then we can generate an upper Hessenberg matrices \hat{H}_k whose eigenvalues are $\zeta_i^{(k)}$, $i = 1, \dots, k$ by choosing the entries of U_{k+1}^{-1} according to Corollary 3.3. At the same time, this will ensure that the k th relative residual norm is f_k . Doing this consecutively for $k = 1, \dots, n-1$, we can construct $U = U_n$. \square

As for Theorem 3.15, in the previous theorem we can prescribe the MR residual norms instead of the OR residual norms by using relation (3.17). This means that just as for OR methods, for MR methods the roots of the residual polynomials p_k^M where $r_k^M = p_k^M(A)r_0$ can be prescribed independent of $\|r_k^M\|$.

We did not consider in this section the special situation of prescribing infinite OR residual norms or, equivalently, prescribing stagnating MR residual norms. Stagnating general Q-MR residual norms are addressed in the next section.

3.6 Stagnation of Q-MR

For simplicity let us write $z_k = z_k^M$. Partial stagnation for p iterations of a Q-MR method is defined as having $\|z_k\| < \|z_{k-1}\|$, $k = 1, \dots, m$, $\|z_k\| = \|z_{k-1}\|$, $k = m + 1, \dots, m + p - 1$, and $\|z_k\| < \|z_{k-1}\|$, $k = m + p, \dots, n$. If $m = 0$ we have $\|z_k\| = \|r_0\|$, $k = 1, \dots, p - 1$ and $\|z_p\| < \|z_{p-1}\|$. Hence, in all cases, the norms of the quasi-residuals stay the same for p iterations starting from $k = m$. Complete stagnation corresponds to $m = 0$ and $p = n$. Thus $\|z_k\| = \|r_0\|$, $k = 1, \dots, n - 1$, and $\|z_n\| = 0$. Since $\|z_{n-1}\| \neq 0$ implies that the degree of the minimal polynomial of A is equal to n , in this case, we have to assume that the matrix A is nonderogatory.

Complete stagnation of GMRES (which is a true MR method) has been studied in [1016, 1017]; see also [829, 838] where conditions for non-stagnation are studied as well as [642] for the study of worst-case GMRES for normal matrices. Necessary and sufficient conditions for complete or partial stagnation of GMRES have been given in [682]. But we have seen in the previous sections that any Q-MR method can possibly stagnate.

Obviously, stagnation should be part of prescribed residual norm behavior, and it is relatively easy to construct an upper Hessenberg matrix H with prescribed eigenvalues for which we have stagnation of Q-MR with a right-hand side $b = e_1$ by using Theorems 3.3 and 3.12. As an illustration, let us consider complete stagnation. We construct a nonsingular upper triangular matrix that we denote by U^{-1} with all the entries on the first row equal to 0 except $[U^{-1}]_{1,1} = 1$. With the prescribed eigenvalues we compute the coefficients of the monic polynomial with those prescribed values as roots and then the corresponding companion matrix C . Finally, we set $H = UCU^{-1}$. If we use GMRES to solve $Hx = e_1$ we observe complete stagnation from iteration 1 to $n - 1$. To obtain a general linear system with complete stagnation we define $A = VHV^{-1}$ and $b = Ve_1$ but the choice of the nonsingular matrix V depends on the Krylov method; for instance, for GMRES we can choose V unitary but this is different for other methods. Moreover, for some methods the matrix H has to have a particular nonzero structure. We will discuss these issues in more detail in the next chapters. In this chapter we have assumed that Hessenberg matrices H_k are nonsingular unless explicitly stated otherwise. This is equivalent with assuming non-stagnation in the k th iteration. Many results of the previous sections do not hold anymore when H_k is singular or, equivalently, when $\alpha_0^{(k)}$ is zero.

3.7 Residual bounds

As we said above, many papers have been written providing upper bounds depending on the eigenvalues of A for the residual norms of particular Krylov methods. The aim of these works was often to have reliable a priori information on convergence speed since, in a number of applications, a priori information is available in the form of eigenvalues, or at least the condition number.

Let us derive bounds for general Q-MR and Q-OR methods. For simplicity, we consider the case where the Q-MR algorithm does not stop before iteration n . We first use the same technique as in [379].

Theorem 3.17 *Assume the Q-MR algorithm with unit norm basis vectors terminates at iteration n . Let H be such that $AV = VH$ and assumed to be diagonalizable as $H = X_H \Lambda X_H^{-1}$. Then, the Q-MR residual norm at iteration k is bounded as*

$$\|r_k^M\| \leq \|r_0\| \|X_H\| \|X_H^{-1}\| \sqrt{k+1} \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \quad (3.40)$$

where π_k is the set of polynomials of degree k .

Proof We can write the quasi-residual vector z_k^M as

$$\begin{aligned} z_k^M &= \|r_0\| e_1 - \underline{H}_k y, \\ &= \|r_0\| (e_1 - \underline{H}_k z), \\ &= \|r_0\| \left(e_1 - H \begin{pmatrix} z \\ 0 \end{pmatrix} \right), \end{aligned}$$

with $z \in \mathbb{C}^k$ and with an abuse of notation since e_1 changes length in these equalities. We are interested in the norm of the term within parenthesis. The matrix H is upper Hessenberg with nonzero subdiagonal entries. If we look at the first columns of the powers of H , only the first $j+1$ components of the vector $H^j e_1$ are nonzero. Therefore, given z , there is a polynomial q of degree $k-1$ such that

$$\begin{pmatrix} z \\ 0 \end{pmatrix} = q(H) e_1,$$

and we can write

$$e_1 - H \begin{pmatrix} z \\ 0 \end{pmatrix} = p(H) e_1,$$

where p is a polynomial of degree k with a value 1 at the origin. Hence,

$$\min \|z_k^M\| \leq \|r_0\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(H)e_1\|.$$

The bound (3.40) is obtained by bounding $\|p(H)e_1\| = \|X_H p(\Lambda) X_H^{-1} e_1\|$ using the fact that the eigenvalues of H are the eigenvalues of A and noticing that, since the basis vectors are of unit norm, $\|V_{n,k+1}\| \leq \sqrt{k+1}$. \square

We observe that if the matrix V of the basis vectors is unitary we can replace X_H in the bound (3.40) by the matrix X of the eigenvectors of A . Moreover, if the matrix A is normal then the term $\kappa(X) = \|X\| \|X^{-1}\|$ can be replaced by 1. Using Theorem 3.17 and the relations between the Q-MR and Q-OR methods we obtain an upper bound for the norm of the Q-OR residual.

Corollary 3.4 *Using the same notation as in Theorem 3.17, the Q-OR residual norm is bounded as*

$$\|r_k^O\| \leq \frac{\|r_0\|}{|c_k|} \|X_H\| \|X_H^{-1}\| \sqrt{k+1} \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \quad (3.41)$$

where c_k is the cosine of the k th Givens rotation used to triangularize the upper Hessenberg matrices.

Proof We use the bound of Theorem 3.17 and relation (3.16) of Theorem 3.3. \square

Unfortunately we could have $c_k = 0$ (or very small in finite precision arithmetic) and the upper bound goes to infinity. This reflects the fact that at some iterations the matrices H_k might be (close to being) singular.

The solution of the problem $\min_{p \in \pi_k, p(0)=1} \max_{\lambda \in \sigma(A)} |p(\lambda)|$ is not known explicitly and moreover, in many cases, we do not know the eigenvalues of A . Bounds have been given for the value of the minimum when A has some particular eigenvalue distributions; see, for instance, [789, 801]. We will consider this in more detail in Chapter 5. If the upper bounds (3.40) and (3.41) converges to zero, they can provide insights into the convergence of the numerical method. However, if the condition number $\|X_H\| \|X_H^{-1}\|$ is large, they may not explain the behavior of the method. Moreover, these bounds separate the effect of the matrix and the initial residual vector. We have seen in Section 3.5 that the influence of the right-hand side may be important.

Other bounds can be obtained by considering the residual polynomials.

Theorem 3.18 *Assume that the matrix A is diagonalizable as $A = X\Lambda X^{-1}$. The residual norms of the Q-MR and Q-OR methods are bounded by*

$$\|r_k^M\| \leq \|r_0\| \|X\| \|X^{-1}\| \max_{\lambda \in \sigma(A)} \frac{\prod_{j=1}^k |\lambda - \zeta_j^{(k)}|}{\prod_{j=1}^k |\zeta_j^{(k)}|}, \quad (3.42)$$

$$\|r_k^O\| \leq \|r_0\| \|X\| \|X^{-1}\| \max_{\lambda \in \sigma(A)} \frac{\prod_{j=1}^k |\lambda - \theta_j^{(k)}|}{\prod_{j=1}^k |\theta_j^{(k)}|}, \quad (3.43)$$

where $\zeta_j^{(k)}, j = 1, \dots, k$ (resp. $\theta_j^{(k)}$) are the eigenvalues of the matrix \hat{H}_k (resp. H_k).

Proof Let p_k^M be the Q-MR residual polynomial. Then,

$$r_k^M = p_k^M(A)r_0 = Xp_k^M(\Lambda)X^{-1}r_0.$$

We have seen in Theorem 3.9 that the residual polynomial p_k^M is proportional to the characteristic polynomial of \hat{H}_k . It is obtained by dividing the characteristic polynomial by the constant term which is $\det(\hat{H}_k)$. This immediately gives the upper bound (3.42). The proof of the other bound (3.43) is similar but we note that $\det(H_k)$ may be zero and the bound infinite. \square

The bounds could become small if the eigenvalues of \hat{H}_k and H_k converge in some sense to the eigenvalues of A but we remark that if one $|\zeta_j^{(k)}|$ or $|\theta_j^{(k)}|$ is small the bounds may be far from being sharp.

Note also that, in the bounds of this section, it may be better to keep $\|X^{-1}r_0\|$ and not to bound it as $\|X^{-1}\| \|r_0\|$ as it is usually done.

3.8 Residual norms in terms of spectral quantities

The result in Section 3.5 (and analog results in later chapters) shows that there are sets of matrices with arbitrary eigenvalue distributions and right-hand sides giving prescribed residual norms for Q-OR methods or prescribed quasi-residual norms for Q-MR methods. This seems surprising because with normal matrices eigenvalues govern the convergence behavior. However, Theorem 3.15 does not imply that the behavior of these methods is not influenced by the eigenvalues at all. It just means that if we modify the eigenvalues, then to have the same (quasi-)residual norms, the eigenvectors and/or the right-hand side must be modified.

In this section we give closed-form expressions for (quasi)-residual norms in terms of eigenvalues, eigenvectors, the right-hand side and basis vectors. For simplicity, we will not consider the case of defective matrices; it is addressed in detail in [686]. The main ingredient to write the (quasi)-residual norms as a function of the eigenvalues and eigenvectors of A is Corollary 3.2, which is based on the moment matrix,

$$M = \frac{1}{\|r_0\|^2} K^* V^{-*} V^{-1} K,$$

and its leading principal submatrices M_k , see (3.10). With the spectral factorization of A we can write the Krylov matrix K in terms of eigenvalues, eigenvectors and the right-hand side as it was done in Lemma 2.4. It is assumed that K is nonsingular, but the following result is essentially the same if A does not have distinct eigenvalues or when some components of the right-hand side in the eigenvector basis are zero.

Theorem 3.19 Let A be a diagonalizable matrix with a spectral factorization $X\Lambda X^{-1}$ where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ contains the distinct eigenvalues, let b be the right-hand side and $r_0 = b - Ax_0$ such that $c = X^{-1}r_0$ has no zero entries and let $Z = V^{-1}X$ with V the matrix of the basis vectors of Krylov subspaces. Then for $0 < k < n$,

$$(M_{k+1}^{-1})_{1,1} = \|r_0\|^2 \mu_k^N / \mu_{k+1}^D,$$

where $\mu_1^N = \sum_{i=1}^n \left| \sum_{j=1}^n Z_{i,j} c_j \lambda_j \right|^2$, for $k \geq 2$

$$\mu_k^N = \sum_{\mathcal{I}_k} \left| \sum_{\mathcal{J}_k} \det(Z_{\mathcal{I}_k, \mathcal{J}_k}) c_{j_1} \cdots c_{j_k} \lambda_{j_1} \cdots \lambda_{j_k} \prod_{j_\ell < j_p \in \mathcal{J}_k} (\lambda_{j_p} - \lambda_{j_\ell}) \right|^2, \text{ and for } k \geq 1$$

$$\mu_{k+1}^D = \sum_{\mathcal{I}_{k+1}} \left| \sum_{\mathcal{J}_{k+1}} \det(Z_{\mathcal{I}_{k+1}, \mathcal{J}_{k+1}}) c_{j_1} \cdots c_{j_{k+1}} \prod_{j_\ell < j_p \in \mathcal{J}_{k+1}} (\lambda_{j_p} - \lambda_{j_\ell}) \right|^2.$$

The summations are over all sets of indices $\mathcal{I}_{k+1}, \mathcal{J}_{k+1}, \mathcal{I}_k, \mathcal{J}_k$ defined as \mathcal{I}_ℓ (or \mathcal{J}_ℓ) to be a set of ℓ indices $(i_1, i_2, \dots, i_\ell)$ such that $1 \leq i_1 < \dots < i_\ell \leq n$ and $Z_{\mathcal{I}_\ell, \mathcal{J}_\ell}$ is the submatrix of Z whose row and column indices of entries are defined, respectively, by \mathcal{I}_ℓ and \mathcal{J}_ℓ .

Proof We have $M = U^*U = (1/\|r_0\|^2)K^*V^{-*}V^{-1}K$, and the matrices M_j , $j = 1, \dots, n-1$ are the leading principal submatrices of M . The matrix A has been assumed to be diagonalizable with $A = X\Lambda X^{-1}$. Then, as we have seen in Chapter 2, $K = X(c \Lambda c \cdots \Lambda^{n-1}c)$ with $c = X^{-1}r_0$ and $M = (1/\|r_0\|^2)\tilde{M}$ with

$$\tilde{M} = (c \Lambda c \cdots \Lambda^{n-1}c)^* X^*V^{-*}V^{-1}X (c \Lambda c \cdots \Lambda^{n-1}c).$$

Let D_c be the diagonal matrix with diagonal entries c_i , and let $\mathcal{V}_{n,k+1}$ denote the first $k+1$ columns of the square Vandermonde matrix for $\lambda_1, \dots, \lambda_n$

$$\mathcal{V}_{n,k+1} = \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^k \\ 1 & \lambda_2 & \cdots & \lambda_2^k \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_n & \cdots & \lambda_n^k \end{pmatrix}.$$

Then the matrix M_{k+1} can be written as $(1/\|r_0\|^2)\tilde{M}_{k+1}$ with

$$\tilde{M}_{k+1} = \mathcal{V}_{n,k+1}^* D_c^* X^* V^{-*} V^{-1} X D_c \mathcal{V}_{n,k+1}.$$

With the $n \times (k+1)$ matrix $F = ZD_c\mathcal{V}_{n,k+1}$, it is the product F^*F of two rectangular matrices. Now, as in [291], we apply Cramer's rule and the Cauchy–Binet formula

(see Chapter 2) to compute the $(1, 1)$ entry of the inverse of \tilde{M}_{k+1} . From Cramer's rule we have

$$(\tilde{M}_{k+1}^{-1})_{1,1} = \frac{1}{\det(\tilde{M}_{k+1})} \det \begin{pmatrix} 1 & \tilde{m}_{1,2} & \cdots & \tilde{m}_{1,k} \\ 0 & & & \\ \vdots & \tilde{M}_{2:k+1,2:k+1} & & \\ 0 & & & \end{pmatrix} = \frac{\det(\tilde{M}_{2:k+1,2:k+1})}{\det(\tilde{M}_{k+1})}.$$

Let us first consider the determinant of \tilde{M}_{k+1} . Let \mathcal{I}_{k+1} be a set of $k + 1$ row indices $(i_1, i_1, \dots, i_{k+1})$ such that $1 \leq i_1 < \dots < i_{k+1} \leq n$ and $F_{\mathcal{I}_{k+1},:}$ the submatrix of F whose row indices belong to \mathcal{I}_{k+1} . Then, by the Cauchy–Binet formula,

$$\det(\tilde{M}_{k+1}) = \sum_{\mathcal{I}_{k+1}} |\det(F_{\mathcal{I}_{k+1},:})|^2,$$

where the summation is over all such possible ordered sets of $k + 1$ indices.

Now we have to compute the determinant of $F_{\mathcal{I}_{k+1},:}$, a matrix which consists of rows i_1, \dots, i_{k+1} of $ZD_c\mathcal{V}_{n,k+1}$. It is the product of a $(k + 1) \times n$ matrix that we can write as $(ZD_c)_{\mathcal{I}_{k+1},:}$ by the $n \times (k + 1)$ matrix $\mathcal{V}_{n,k+1}$. Once again we can use the Cauchy–Binet formula. Let \mathcal{J}_{k+1} denote any ordered set of $k + 1$ column indices, and let $Z_{\mathcal{I}_{k+1},\mathcal{J}_{k+1}}$ be the submatrix of Z whose rows and columns belong, respectively, to \mathcal{I}_{k+1} and \mathcal{J}_{k+1} and

$$\mathcal{V}(\lambda_{j_1}, \dots, \lambda_{j_{k+1}}) = \begin{pmatrix} 1 & \lambda_{j_1} & \cdots & \lambda_{j_1}^k \\ 1 & \lambda_{j_2} & \cdots & \lambda_{j_2}^k \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_{j_{k+1}} & \cdots & \lambda_{j_{k+1}}^k \end{pmatrix}$$

is a square Vandermonde matrix of order $k + 1$. Then,

$$\det(F_{\mathcal{I}_{k+1},:}) = \sum_{\mathcal{J}_{k+1}} \det(Z_{\mathcal{I}_{k+1},\mathcal{J}_{k+1}}) c_{j_1} \cdots c_{j_{k+1}} \det(\mathcal{V}(\lambda_{j_1}, \dots, \lambda_{j_{k+1}})),$$

where the summation is over all such possible ordered sets of $k + 1$ indices. Moreover, we have

$$\det(\mathcal{V}(\lambda_{j_1}, \dots, \lambda_{j_{k+1}})) = \prod_{j_1 \leq j_l < j_p \leq j_{k+1}} (\lambda_{j_p} - \lambda_{j_l}),$$

the indices j_l and j_p belonging to the set \mathcal{J}_{k+1} . Finally, the determinant of \tilde{M}_{k+1} is

$$\mu_{k+1}^D = \sum_{\mathcal{I}_{k+1}} \left| \sum_{\mathcal{J}_{k+1}} \det(Z_{\mathcal{I}_{k+1},\mathcal{J}_{k+1}}) c_{j_1} \cdots c_{j_{k+1}} \prod_{j_1 \leq j_l < j_p \leq j_{k+1}} (\lambda_{j_p} - \lambda_{j_l}) \right|^2,$$

where the summation is over all such possible ordered sets of $k + 1$ indices, \mathcal{I}_{k+1} and \mathcal{J}_{k+1} .

Let us now consider the determinant of $\tilde{M}_{2:k+1,2:k+1}$ which is a matrix of order k . The computation is essentially the same, except that we have to consider rows and columns 2 to $k + 1$. Therefore, it is not $\mathcal{V}_{n,k}$ which is involved but a matrix that can be written as $\Lambda \mathcal{V}_{n,k}$. We have

$$\tilde{M}_{2:k+1,2:k+1} = \mathcal{V}_{n,k}^* \bar{\Lambda} D_c^* Z^* Z D_c \Lambda \mathcal{V}_{n,k}.$$

Then, we have some additional factors arising from the diagonal matrix Λ and we have to consider only sets of k indices \mathcal{I}_k and \mathcal{J}_k . The determinant of $\tilde{M}_{2:k+1,2:k+1}$ is obtained as

$$\mu_k^N = \sum_{\mathcal{I}_k} \left| \sum_{\mathcal{J}_k} \det(Z_{\mathcal{I}_k, \mathcal{J}_k}) c_{j_1} \cdots c_{j_k} \lambda_{j_1} \cdots \lambda_{j_k} \prod_{j_1 \leq j_l < j_p \leq j_k} (\lambda_{j_p} - \lambda_{j_l}) \right|^2,$$

where the summation is over all such possible ordered sets of k indices, \mathcal{I}_k and \mathcal{J}_k .

Finally we obtain

$$(M_{k+1}^{-1})_{1,1} = \|r_0\|^2 \frac{\mu_k^N}{\mu_{k+1}^D},$$

which concludes the proof. \square

The (quasi)-residual norms can be written as follows.

Corollary 3.5 *Using the notation of Theorem 3.19 we have for $0 < k < n$*

$$\|z_k^M\|^2 = \frac{\mu_{k+1}^D}{\mu_k^N}, \quad \|r_k^O\|^2 = \left(\frac{\mu_k^N}{\mu_{k+1}^D} - \frac{\mu_{k-1}^N}{\mu_k^D} \right)^{-1},$$

where μ_0^N is defined as $\mu_0^N = \mu_1^D$.

Proof We have, using Corollary 3.2,

$$\frac{\|z_k^M\|^2}{\|r_0\|^2} = \frac{1}{(M_{k+1}^{-1})_{1,1}} = \frac{1}{\|r_0\|^2} \frac{\mu_k^D}{\mu_k^N}$$

and similarly for $\|r_k^O\|^2$. We set $\mu_0^N = \mu_1^D$ because $M_{1,1} = 1$. \square

The last two results show that the (quasi)-residual norms in Q-OR/Q-MR methods depend upon the following three types of objects: Eigenvalues, components of the right-hand side in the eigenvector basis and determinants of submatrices of Z , i.e., of the eigenvector matrix multiplied with the inverse of the generated basis for

the Krylov subspaces. Even if A is normal, i.e., $X^*X = I$, this needs not imply that the determinants in μ_k^N , μ_{k+1}^D can be simplified. However, if V is unitary, like in the GMRES method and in most Krylov methods for symmetric matrices, and if the matrix A is normal, we have $X^*X = I$ and the sums over \mathcal{J}_k and \mathcal{J}_{k+1} reduce to only one term ($\mathcal{J}_k = \mathcal{I}_k$, respectively, $\mathcal{J}_{k+1} = \mathcal{I}_{k+1}$). We then recover the formula in [291] and in an unpublished report from M. Bellalij and H. Sadok (A new approach to GMRES convergence, 2011).

Theorem 3.19 suggests that, in general, eigenvalues, eigenvectors, the right-hand side and the generated Krylov subspace basis can play a similarly important role. The formulas in Theorem 3.19 are quite complicated closed-form expressions but they describe exactly the dependence of (quasi)-residual norms on eigenvalues and eigenvectors. However, we can obtain simpler bounds by using the following result which was proved in [86].

Lemma 3.3 *Let E and F be two matrices of sizes $n \times (k + 1)$ and $n \times n$, respectively, $k \leq n - 1$. If the matrix E is of full rank*

$$\frac{\sigma_{\min}(F)^2}{e_1^T(E^*E)^{-1}e_1} \leq \frac{1}{e_1^T(E^*(F^*F)E)^{-1}e_1} \leq \frac{\sigma_{\max}(F)^2}{e_1^T(E^*E)^{-1}e_1}. \quad (3.44)$$

It yields lower and upper bounds for the norm of the quasi-residual of the Q-MR method.

Theorem 3.20 *Let A be a diagonalizable matrix, $A = X\Lambda X^{-1}$, $c = X^{-1}r_0$ and*

$$\mu_k = \frac{\sum_{1 \leq i_1 < \dots < i_{k+1} \leq n} \left(\prod_{j=1}^{k+1} |c_{i_j}|^2 \right) \prod_{i_1 \leq i_\ell < i_p \leq i_{k+1}} |\lambda_{i_p} - \lambda_{i_\ell}|^2}{\sum_{1 \leq i_1 < \dots < i_k \leq n} \left(\prod_{j=1}^k |c_{i_j}|^2 |\lambda_{i_j}|^2 \right) \prod_{i_1 \leq i_\ell < i_p \leq i_k} |\lambda_{i_p} - \lambda_{i_\ell}|^2}.$$

Then,

$$\mu_k [\sigma_{\min}(V^{-1}X)]^2 \leq \|z_k^M\|^2 \leq \mu_k \|V^{-1}X\|^2.$$

Proof Using Lemma 3.3 with $E = D_c \mathcal{V}_{n,k+1}$ and $F = V^{-1}X$ in the proof of Theorem 3.19 we can eliminate the factor $X^*V^{-*}V^{-1}X$ in the expression $\mathcal{V}_{n,k+1}^* D_c^* X^*V^{-*}V^{-1}X D_c \mathcal{V}_{n,k+1}$. Then, the derivation is done as if we would have $Z = I$, and it is enough to use the Cauchy–Binet formula only once. \square

3.9 Relations between Q-OR/Q-MR methods with different bases

Our goal in this section is to compare Q-OR and Q-MR methods using different bases of the same Krylov subspaces corresponding to a matrix A and an initial

residual vector r_0 . Let V, U and \tilde{V}, \tilde{U} be the matrices arising with two different bases for the Krylov subspaces, for instance, in two different Q-OR Krylov methods. In other words, the columns of V and \tilde{V} contain these bases, and U and \tilde{U} are the upper triangular matrices in the triangular Hessenberg decomposition (2.17) of the two generated Hessenberg matrices. We assume these matrices to be nonsingular, i.e., the methods do not terminate prematurely. Since these matrices are related to the same Krylov subspace, we have the relation (see (3.11))

$$K/\|r_0\| = VU = \tilde{V}\tilde{U}.$$

Let us define

$$W \equiv \tilde{V}^{-1}V = \tilde{U}U^{-1}. \quad (3.45)$$

The matrix W can be seen as exhibiting the departure of one method to the other. Note that the last equality shows that W is upper triangular with real positive diagonal entries. We will see below that the farther W is from the identity matrix, the less the first method behaves like the second one.

The Hessenberg matrix H of the first method can be transformed with W to the Hessenberg matrix \tilde{H} generated in the second method. Using $\tilde{H} = \tilde{U}C\tilde{U}^{-1}$, we have

$$H = UCU^{-1} = V^{-1}\tilde{V}\tilde{U}C\tilde{U}^{-1}\tilde{V}^{-1}V = W^{-1}\tilde{H}W.$$

Of course the two matrices H and \tilde{H} are similar since they are both similar to A . For the submatrices \underline{H}_k and H_k of H , which define, with the computed bases for the Krylov subspaces, respectively, the Q-MR and Q-OR iterates, we have the following results which first appeared in [289].

Theorem 3.21 *Let W_k denote the leading principal submatrix of order k of W . Then,*

$$\tilde{H}_k = W_{k+1}\underline{H}_k W_k^{-1},$$

and

$$\tilde{H}_k = W_k H_k W_k^{-1} + [0 \ \frac{u_{k+1,k+1}}{\tilde{u}_{k,k}} W_{1:k,k+1}].$$

Proof Using definition (3.45) we obtain $W_k = \tilde{U}_k U_k^{-1}$. Considering $AV_{n,k} = V_{n,k+1}\underline{H}_k$ we can write

$$A\tilde{V}_{n,k}\tilde{U}_k U_k^{-1} = \tilde{V}_{n,k+1}\tilde{U}_{k+1} U_{k+1}^{-1}\underline{H}_k.$$

Hence, $A\tilde{V}_{n,k} = \tilde{V}_{n,k+1}\tilde{U}_{k+1} U_{k+1}^{-1}\underline{H}_k U_k \tilde{U}_k^{-1}$, and

$$\tilde{H}_k = \tilde{U}_{k+1} U_{k+1}^{-1}\underline{H}_k U_k \tilde{U}_k^{-1} = W_{k+1}\underline{H}_k W_k^{-1}.$$

Using a partitioning of W_{k+1} we have

$$\underline{H}_k = \begin{pmatrix} W_k & W_{1:k,k+1} \\ 0 & \omega_k \end{pmatrix} \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix} W_k^{-1}.$$

Therefore,

$$\tilde{H}_k = W_k H_k W_k^{-1} + W_{1:k,k+1} h_{k+1,k} e_k^T W_k^{-1}. \quad (3.46)$$

Because W_k is upper triangular,

$$\tilde{H}_k = W_k H_k W_k^{-1} + h_{k+1,k} [W_k^{-1}]_{k,k} W_{1:k,k+1} e_k^T.$$

But with Theorem 2.3 and (3.45),

$$h_{k+1,k} = \frac{u_{k+1,k+1}}{u_{k,k}}, \quad [W_k^{-1}]_{k,k} = \frac{u_{k,k}}{\tilde{u}_{k,k}},$$

and we obtain the result. \square

Thus \underline{H}_k is transformed to \tilde{H}_k with W_k and W_{k+1} , and the matrix \tilde{H}_k is a rank-one modification of H_k , transformed with W_k . The transformations and the rank-one modification can have a large impact on the iterates and on convergence behavior.

We can express the relation between the first row of U^{-1} and the first row of \tilde{U}^{-1} , which determine the corresponding Q-OR residual norms, by the matrix W : Using $U^{-1} = \tilde{U}^{-1}W$, see (3.45), we immediately obtain $e_1^T U^{-1} = e_1^T \tilde{U}^{-1}W$, or, equivalently,

$$\begin{pmatrix} 1 \\ \vartheta_{1,2} \\ \vdots \\ \vartheta_{1,n} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ w_{1,2} & w_{2,2} & & \\ \vdots & \ddots & \ddots & \\ w_{1,n} & \dots & w_{n-1,n} & w_{n,n} \end{pmatrix} \begin{pmatrix} 1 \\ \tilde{\vartheta}_{1,2} \\ \vdots \\ \tilde{\vartheta}_{1,n} \end{pmatrix}. \quad (3.47)$$

Since the moduli of the $\vartheta_{1,j}$'s are the reciprocals of the relative Q-OR residual norms, depending on the entries of W , the residual norms of a Q-OR method can be arbitrarily different from those of another Q-OR method and similarly for Q-MR methods. The next theorem expresses the differences between the convergence of two Q-OR methods using W .

Theorem 3.22 *If $\|\tilde{r}_k^O\| \neq \infty$ (i.e., $\tilde{\vartheta}_{1,k+1} \neq 0$), then for $k = 1, \dots, n-1$,*

$$\|r_k^O\| = \left[w_{1,k+1} + \frac{1}{\|\tilde{r}_1^O\|} w_{2,k+1} + \dots + \frac{1}{\|\tilde{r}_{k-1}^O\|} w_{k,k+1} + \frac{1}{\|\tilde{r}_k^O\|} w_{k+1,k+1} \right]^{-1}.$$

Proof From the $(k + 1)$ st row in (3.47) we obtain

$$\vartheta_{1,k+1} = w_{1,k+1} + \tilde{\vartheta}_{1,2}w_{2,k+1} + \cdots + \tilde{\vartheta}_{1,k}w_{k,k+1} + \tilde{\vartheta}_{1,k+1}w_{k+1,k+1}.$$

Using Theorem 3.12 we obtain the claim. \square

The following theorem relates the norms of the quasi-residual vectors in two Q-MR methods.

Theorem 3.23

$$\|\tilde{z}_k^M\| \leq \|W_k\| \|z_k^M\|. \quad (3.48)$$

Proof Taking the norms of the first k rows of relation (3.47) we obtain

$$\left(\sum_{j=1}^k |\vartheta_{1,j}|^2 \right)^{\frac{1}{2}} \leq \|W_k\| \left(\sum_{j=1}^k |\tilde{\vartheta}_{1,j}|^2 \right)^{\frac{1}{2}}.$$

We obtain the result from Theorem 3.12. \square

Note that if we interchange z_k^M and \tilde{z}_k^M , W_k has to be changed to W_k^{-1} .

3.10 Generalized Q-OR/Q-MR methods

Some well-known Krylov methods like (Bi)CGS, BiCGStab and IDR do not exactly fit in our general Q-OR/Q-MR framework. The basis vectors that are (implicitly) generated by the iterations do not satisfy an Arnoldi-like relation (3.2), but a relation

$$AG_{n,k}R_k = G_{n,k+1}\underline{H}_k, \quad (3.49)$$

where R_k is upper triangular with a unit diagonal and $G_{n,k}e_1 = r_0/\|r_0\|$. If the diagonal entries of R_k are not equal to 1, we can scale it and modify the columns of \underline{H}_k accordingly. We could have multiplied both sides of relation (3.49) by R_k^{-1} but, in these methods, the columns of $V_{n,k} = G_{n,k}R_k$ are generally used as basis vectors and the *generalized Q-OR method* is defined by

$$x_k = x_0 + V_{n,k}y_k, \quad H_k y_k = \|r_0\|e_1.$$

The basis vectors in $V_{n,k} = G_{n,k}R_k$ are not of unit norm but we can scale them by modifying the entries of the matrix \underline{H}_k . Hence, to study the mathematical properties of the methods we can assume that the columns of $V_{n,k}$ are of unit norm and the upper Hessenberg matrix has been scaled accordingly.

Relation (3.49) yields

$$\begin{aligned} r_k &= r_0 - AV_{n,k}y_k \\ &= G_{n,k}(\|r_0\|e_1 - H_k y_k) - h_{k+1,k}[y_k]_k g_{k+1} e_k^T \\ &= -h_{k+1,k}[y_k]_k g_{k+1} e_k^T. \end{aligned}$$

Hence, the residual vector r_k is proportional to g_{k+1} .

We can also define a *generalized Q-MR method* by minimizing the norm of $\|r_0\|e_1 - \underline{H}_k y_k$ to compute y_k .

For these generalized methods, the question is: What are the results proved in this chapter that still hold? Rather than restating all the results, we just list below what is still true and what has to be modified.

- Theorem 3.2 holds because the proofs depend only on \underline{H}_k . Proposition 3.1 holds also if g_{k+1} is of unit norm. Otherwise we have to divide $\|r_k^O\|$ by the norm of g_{k+1} .
- Theorem 3.3 holds for the same reasons. Note that the matrix R_k in the proof of that theorem is not the same as the matrix in our generalized Hessenberg relation.
- If $g_1 = r_0/\|r_0\|$, the columns of $G_{n,k}$ can be expressed as polynomials in A applied to r_0 and so are the residual vectors.
- Proposition 3.2 and Theorem 3.7 have to be modified. We notice that we have

$$AV_{n,k} = V_{n,k+1}R_{k+1}^{-1}\underline{H}_k,$$

and the matrix $R_k^{-1}\underline{H}_k$ is upper Hessenberg. Consequently, the generalized Q-OR residual polynomial is proportional to the characteristic polynomial of $R_k^{-1}\underline{H}_k$. The roots of the residual polynomial are the eigenvalues of the pencil (H_k, R_k) ; see also [494, Theorem 2.2].

- The relation of Lemma 3.1 changes to

$$AG_{n,k} = G_{n,k}R_k^{-1}\hat{H}_k + h_{k+1,k}G_{n,k+1} \begin{pmatrix} -h_{k+1,k}H_k^{-*}e_k \\ 1 \end{pmatrix} e_k^T,$$

with the same definition of \hat{H}_k as in the standard case (with $R_k = I$).

- For Theorem 3.8 we have a relation

$$AG_{n,k}R_k = G_{n,k}\hat{H}_k - \omega_k G_{n,k+1}z_k^M e_k^T, \quad (3.50)$$

for some ω_k . But, from the definition of r_k , we have $r_k^M = G_{n,k+1}(\|r_0\|e_1 - \underline{H}_k y_k)$ and therefore $r_k^M = G_{n,k+1}z_k^M$. The last term of (3.50) is proportional to $r_k^M e_k^T$ and we have the analog of Proposition 3.3.

- Using the previous results, the analog of Theorem 3.9 holds and the residual polynomial is proportional to the characteristic polynomial of $R_k^{-1}\hat{H}_k$. When H_k is nonsingular, the roots of the residual polynomial are the eigenvalues of the pencil (\hat{H}_k, R_k) .

- Theorem 3.10 still holds since the proof depends only on \underline{H}_k .
- Theorem 3.12 is true (as long as the vectors g_k are of unit norm) with H_k (and not $H_k R_k^{-1}$) because y_k^O is obtained from H_k . The following corollary also holds. But the factorization of Theorem 3.13 does not hold because the characteristic polynomial of H_k is not proportional to the residual polynomial.

3.11 Historical notes

Even though we have seen in Chapter 2 that Krylov vectors started to be used in the 1930s, Krylov methods for solving linear systems became to be developed after World War II. In fact, Krylov vectors also appeared on page 3 of the report [512] by Karl Hessenberg in 1940.

The first significant earlier works on what was not called Krylov methods at that time are those of Cornelius Lanczos [615, 616] in 1950–1952 and of Magnus R. Hestenes and Eduard Stiefel [515] in 1952. For interesting information about Lanczos' life and his work, see [410] and the book [411].

Hestenes and Stiefel proposed the Conjugate Gradient (CG) method for solving symmetric positive definite linear systems. Lanczos was first interested in the computation of eigenvalues but also in solving linear systems. Moreover, he also proposed an algorithm for nonsymmetric linear systems as we will see in Chapter 8.

Although it is clear from relation (4.8b) page 415 of [515] that the CG residual vectors are polynomials in A applied to the initial residual, there is no mention of Krylov in this paper. As we have already seen in Chapter 2, Krylov is only mentioned in a footnote of [615].

The paper [35] by Walter E. Arnoldi in 1951 proposed what is essentially the algorithm we use today to compute an orthonormal basis for Krylov subspaces related to a general matrix A . However, he saw his algorithm to be a variation of the nonsymmetric Lanczos method that he first described at the beginning of his paper. Our familiar upper Hessenberg matrices appeared in relation (23) on page 26. There is no mention of Krylov in Arnoldi's paper.

The real expansion of the Krylov methods started in the 1970s together with the need for solving larger problems which was possible with the increase in the speed of computers. Since the early 1980s, there has been considerable interest in developing generalizations of CG and the symmetric Lanczos process for non-Hermitian matrix computations.

Many algorithms were proposed, the most noticeable ones being: BiCG [355] by Roger Fletcher in 1976, FOM [789] by Yousef Saad in 1981, GMRES [801] by Y. Saad and Martin H. Schultz in 1986, CGS [861] by Peter Sonneveld in 1989 with a report in 1984, and BiCGStab [938] by Henk van der Vorst in 1990.

Some of these methods use short recurrences to compute the approximate solutions and the residual vectors. This is a nice feature because it limits the number of vectors to be stored but, on many problems, the norms of the residuals are heavily oscillating. The need to have short recurrences but smooth residual norms motivated

Roland W. Freund and Noël M. Nachtigal to propose the quasi-minimal residual (QMR) method [379] in 1991 with some reports dating from 1990. This method uses a non-orthogonal basis computed using the nonsymmetric Lanczos algorithm.

A very nice paper was published in 2001 by Michael Eiermann and Oliver G. Ernst [296]. Rather than studying the Krylov methods one by one, they show that most (if not all) methods can be studied in a general abstract framework. In their own words,

We show that essentially any Krylov subspace method for solving a linear system can be classified as either an MR or OR method by appropriate choice of the inner product.

This corresponds to the Q-OR/Q-MR methods we study in this book. In fact, this paper was our motivation for considering a framework as general as possible to study the properties of Krylov methods.

Some of the relations between general Q-OR and Q-MR methods that we described in Section 3.2 were proved before for particular methods. The relation between the singularity of H_k in the Arnoldi method (FOM) and the stagnation of GMRES was studied by Peter N. Brown [152] in 1991. He also derived the relations between the residual norms in the two methods.

These relations and particularly the peak/plateau phenomenon were more clearly stated and generalized in the papers [242, 245] by Jane Cullum and Anne Greenbaum in 1995–1996; see also [296]. The results in Section 3.2 were published in 2016 by J. Duintjer Tebbens and G. Meurant [289].

The polynomials associated with the QMR method of Freund and Nachtigal were studied under the name of quasi-kernel polynomials by R. Freund [367] in 1992. A study about the roots of the polynomials associated with Krylov methods was done by Thomas Manteuffel and James Otto [658] in 1994. In 1999 Serge Goosens and Dirk Roose [440] proved that the roots of the GMRES residual polynomials are the so-called harmonic Ritz values. The coefficients of the FOM and GMRES polynomials were exactly expressed in terms of the eigenvalues and eigenvectors of A by G. Meurant [684] in 2017; see Chapter 5.

The relation between the entries of the first row of the inverse of U_k and the Q-OR and Q-MR residual norms was proved by J. Duintjer Tebbens and G. Meurant [289] in 2016 following some previous results for FOM and GMRES.

The relation between the eigenvalues of A and the convergence of Krylov methods has been a topic of research for quite a long time. In the CG method for symmetric linear systems, the convergence behavior is dictated by the distribution of the eigenvalues of the matrix. So, the question arose if it was the same for methods like FOM and GMRES since a dependence of convergence in eigenvalues was observed in many numerical experiments. This question was answered in a series of papers by Zdeněk Strakoš and his co-authors in the 1990s.

A. Greenbaum and Z. Strakoš [463] proved in 1994 that if a residual norm convergence curve is generated by GMRES, the same curve can be obtained with a matrix having prescribed nonzero eigenvalues. In 1996 A. Greenbaum, Vlastimil Pták and Z. Strakoš [460] complemented that result by proving that any non-increasing sequence of residual norms can be given by GMRES. Finally, in 1998, in the paper [34] a complete parametrization was given by Mario Arioli, V. Pták and

Z. Strakoš of all pairs $\{A, b\}$ generating a prescribed GMRES residual norm convergence curve and such that A has a prescribed spectrum. The results in these papers show that the GMRES residual norm convergence is not, in general, depending on the eigenvalues of A alone. Further details were given in [681] by G. Meurant in 2012.

These results generated some misunderstandings. Many papers appeared with sentences like “GMRES convergence does not depend on the eigenvalues”. This was not the conclusion of Strakoš and his co-workers. Their results prove that GMRES convergence, particularly for non-normal matrices, does not depend on the eigenvalues *alone*. Nevertheless, despite all the results in these nice papers, there are still some people believing that GMRES convergence depends only on the eigenvalue distribution, *errare humanum est, perseverare diabolicum*.

These results were complemented in 2012 by J. Duintjer Tebbens and G. Meurant [287] who showed that GMRES convergence curves do not even depend on the Ritz values (that is, the roots of the FOM residual polynomials) generated during all iterations of the GMRES process; see also [686] in 2015. In 2017 these authors and Kui Du [281] proved that any admissible harmonic Ritz value set (that is, the roots of the GMRES residual polynomials of all iterations) is possible for any GMRES convergence curve.

Prescribing the residual norms of GMRES with early termination was addressed in the last section of [34], in the Ph.D. thesis of Joerg Liesen and in [288]. Prescribing the residual norms of restarted GMRES (in a weak sense, that is, at the end of each cycle) was considered by Eugene Vecharynski and Julien Langou [951, 952] in 2010-2011. Prescribing the residual norms of restarted FOM was studied by Marcel Schweitzer [819] in 2016. In [296] the singular values of Hessenberg matrices are prescribed.

We note that in 1996, J. Cullum proved that, given the results of a BiCG computation on $Ax = b$, one can obtain a matrix B with the same eigenvalues as A and a vector c such that the residual norms generated by FOM on $Bx = c$ are identical to those generated by BiCG.

Complete stagnation of GMRES was studied by Ilya Zavorin, Dianne P. O’Leary and Howard C. Elman [1017] in 2003. In 2008, Valeria Simoncini and Daniel B. Szyld [838] gave conditions involving the symmetric (or Hermitian) part and the skew-symmetric parts of A implying that GMRES is not stagnating; see also [829]. The complete stagnation of GMRES was studied for $n \leq 4$ [679] in 2012. Necessary and sufficient conditions for GMRES complete and partial stagnation were given in the paper [682] by G. Meurant in 2014.

Many upper bounds for the residual norms of Krylov methods have been published over the years. We will describe some of them in the next chapters. Residual bounds for QMR were derived in the paper [379] by R.W. Freund and N. Nachtigal in 1991.

Following ideas from Hassane Sadok, closed-form expressions for the GMRES residual norms were obtained for normal matrices in the paper [291] by J. Duintjer Tebbens, G. Meurant, H. Sadok and Z. Strakoš in 2014. These results were extended to non-normal matrices [686] by G. Meurant and J. Duintjer Tebbens in 2015 and generalized to any Q-MR method [289] in 2016. They show exactly how

the (quasi)-residual norms in Q-OR/Q-MR methods depend upon the eigenvalues, the eigenvectors, the components of the right-hand side in the eigenvector basis and the basis vectors.

The relation between the convergence of methods with different types of bases was studied in [289].

Chapter 4

Bases for Krylov subspaces



In this chapter we consider the computation of bases for the Krylov subspace $\mathcal{K}_k(A, r_0)$. We could have used another notation for the vector on which the Krylov subspace is constructed, but in the next chapters, the Krylov subspaces we have to deal with will be based on the initial residual vector $r_0 = b - Ax_0$.

We have seen in Chapter 2 that what can be called the natural basis (or sometimes the monomial basis or the Krylov basis), defined by the column of the matrices

$$K_{n,k} = (r_0 \ A r_0 \ A^2 r_0 \ \cdots \ A^{k-1} r_0), \quad (4.1)$$

can be badly conditioned. In many practical examples, the basis vectors may lose their linear independence quite fast. Hence, this basis cannot be used reliably for Q-OR and Q-MR methods. There are many ways by which one can construct other bases of the Krylov subspace. We will only describe what seem the most useful ones. For the iterative methods we would like to use later, we need a basis for which the basis vectors are numerically strongly linearly independent, ideally obtained with low computational costs.

For simplicity we assume in this chapter that the matrix A is real. However, most of the constructions described below can be generalized to complex matrices by replacing A^T by A^* and by conjugating the corresponding coefficients.

4.1 Orthonormal basis

Let us assume we have a unit norm basis stored in the columns of $V_{n,k}$ satisfying the Arnoldi-like relation (3.2). The basis which is (at least mathematically) the best conditioned one is an orthonormal basis, since all the singular values of $V_{n,k}$ are then equal to 1. In theory it can be obtained by computing the QR factorization

$K_{n,k} = V_{n,k} U_k$ of the matrix $K_{n,k}$. In other words, Theorem 3.1 would apply with (3.10) being a classical QR factorization.

However, we cannot use the QR factorization in this way since we do not want to compute the columns of $K_{n,k}$. A more stable method to compute an orthonormal basis is to use the Gram–Schmidt process in a particular way. When it is tailored to the columns of the Krylov matrix it is called the *Arnoldi process* [35]. Note that we distinguish the Arnoldi process that computes orthonormal basis vectors from the Arnoldi algorithm that uses them to compute eigenvalues or to solve linear systems.

Let $v_1 = r_0/\|r_0\|$. Let us assume that we have already computed orthonormal basis vectors v_2, \dots, v_j of unit norms. To obtain the next basis vector which has to be orthogonal to all the previous ones and has to belong to $\mathcal{K}_{j+1}(A, r_0)$, we orthogonalize Av_j against the previous basis vectors by removing the components of Av_j on these vectors. Let

$$\tilde{v}_j = Av_j - \sum_{i=1}^j (Av_j, v_i)v_i. \quad (4.2)$$

Clearly,

$$(\tilde{v}_j, v_\ell) = (Av_j, v_\ell) - \sum_{i=1}^j (Av_j, v_i)(v_i, v_\ell) = 0, \quad \ell = 1, \dots, j,$$

since in the sum only the term with $i = \ell$ is nonzero. The vector \tilde{v}_j is orthogonal to the previous basis vectors. To obtain the next basis vector v_{j+1} we just have to normalize it as $v_{j+1} = \tilde{v}_j/\|\tilde{v}_j\|$. Denoting,

$$h_{i,j} = (Av_j, v_i), \quad i = 1, \dots, j, \quad h_{j+1,j} = \|\tilde{v}_j\|,$$

in matrix form, using $j = 1, \dots, k$, we have what is known as the *Arnoldi relation*,

$$AV_{n,k} = V_{n,k} H_k + h_{k+1,k} v_{k+1} e_k^T, \quad (4.3)$$

where $V_{n,k}$ is an orthonormal matrix with columns v_j , $j = 1, \dots, k$ and H_k is upper Hessenberg with real positive entries on the first subdiagonal. This construction can be continued as long as $h_{k+1,k} \neq 0$, that is, when k is less than the grade of r_0 with respect to A .

This algorithm is the classical Gram–Schmidt (CGS) variant of the Arnoldi process. From relation (4.2) we see that, in matrix form, we have

$$v_{j+1} = \frac{1}{h_{j+1,j}}(I - V_{n,j} V_{n,j}^T)Av_j.$$

The matrix $I - V_{n,j} V_{n,j}^T$ represents the projection on the orthogonal complement of the subspace spanned by v_1, \dots, v_j , that is, $\mathcal{K}_j(A, r_0)$. A Matlab-like code for computing the basis using CGS, where u is the starting vector, is the following.

```

function [V,H] = Arnoldi_CGS(A,u,nitmax)
%
n = size(A,1);
V = zeros(n,nitmax);
H = zeros(nitmax,nitmax);
u = u / norm(u);
V(:,1) = u;
for j = 1:nitmax % number of iterations
    Av = A * V(:,k); % matrix vector product
    w = Av;
    for i = 1:j
        vAv = V(:,i)' * Av;
        H(i,j) = vAv;
        w = w - vAv * V(:,i);
    end % for i
    nw = norm(w);
    if nw == 0; return; end;
    if j < n
        H(j+1,j) = nw;
        V(:,j+1) = w / nw; % next basis vector
    end % if j
end % for j

```

Note that the loop over i can be more efficiently coded as

$$\begin{aligned} H(1:j,j) &= V(:,1:j)' * Av; \\ w &= w - V(:,1:j) * H(1:j,j); \end{aligned}$$

We coded it using a `for` loop to show the difference with the algorithm we will describe below. The condition $nw = 0$ corresponds to $h_{j+1,j} = 0$ which means that $AV_{n,j} = V_{n,j}H_j$ and that we have generated a basis for an invariant subspace for A . In a practical implementation the condition can be replaced with $\text{abs}(nw) < tol$ for an appropriately chosen tolerance value tol .

Note that in our Gram–Schmidt-like algorithm we chose v_j in $\mathcal{K}_j(A, v_1)$, multiplied it with A and orthogonalized the result against the previous vectors. But we may have chosen any other vector from $\mathcal{K}_j(A, v_1) \setminus \mathcal{K}_{j-1}(A, v_1)$.

In the CGS algorithm all the dot products (Av_j, v_i) are independent of each other and can be computed in parallel. This seems nice. However, for reasons of stability we will discuss later, the algorithm which is the most used in practice is the modified Gram–Schmidt (MGS) variant of the Arnoldi process. Mathematically the two algorithms are equivalent, but their behaviors in finite precision arithmetic are different and MGS is more stable than CGS. In MGS, instead of computing all the dot products with Av_j , we compute the dot products with the last known vector which is the result of the subtraction of the components of Av_j . Hence, the code is almost the same as for CGS except for one line:

```

function [V,H] = Arnoldi_MGS(A,u,nitmax)
%
n = size(A,1);
V = zeros(n,nitmax);
H = zeros(nitmax,nitmax);
u = u / norm(u);
V(:,1) = u;
for j = 1:nitmax % number of iterations
    Av = A * V(:,k); % matrix vector product
    w = Av;
    for i = 1:j
        vAv = V(:,i)' * w;
        H(i,j) = vAv;
        w = w - vAv * V(:,i);
    end % for i
    nw = norm(w);
    if nw == 0; return; end;
    if j < n
        H(j+1,j) = nw;
        V(:,j+1) = w / nw; % next basis vector
    end % if j
end % for j

```

We remark that contrary to the Arnoldi-CGS algorithm the loop over i cannot be easily parallelized since we always need the last vector w to compute the dot product. In the MGS version we have

$$v_{j+1} = \frac{1}{h_{j+1,j}}(I - v_j v_j^T) \cdots (I - v_1 v_1^T) A v_j.$$

Another way of computing an orthonormal basis, proposed in [964, 965], is to use Householder reflections, which are usually exploited in the QR algorithm; see, for instance, [438]. In fact, one orthogonalizes against each other the vectors v_1, Av_1, \dots, Av_j where v_1, v_2, \dots are the consecutively generated basis vectors, similarly in spirit as in the Arnoldi process. But the orthogonalization is based on QR factorization instead of the Gram–Schmidt process.

The first basis vector is, as usual, $v_1 = r_0 / \|r_0\|$ and the first Householder transformation P_1 (an $n \times n$ orthonormal matrix) is constructed such that $P_1 v_1 = e_1$. Therefore, $v_1 = P_1^T e_1 = P_1 e_1$. The second Householder transformation P_2 is constructed such that $P_2 P_1 [v_1, Av_1] = [P_2 e_1, P_2 P_1 Av_1]$ has an upper triangular top 2×2 matrix which we denote by R_2 and elsewhere is zero. The Householder transformation P_2 zeroes out the entries $3, \dots, n$ of $P_2 P_1 Av_1$, therefore, the corresponding Householder vector (the vector v in (2.26)) has zero first entry and $P_2 e_1 = e_1$. Then v_2 is defined as $v_2 = P_1 P_2 e_2$ and v_1 and v_2 can be collected in

$$V_{n,2} = P_1 P_2 (e_1 \ e_2),$$

giving the QR factorization

$$(v_1 \ A v_1) = V_{n,2} R_2.$$

In general, we will have

$$v_{j+1} = P_1 \cdots P_{j+1} e_{j+1}.$$

Assume that we already know v_1, \dots, v_j and P_1, \dots, P_j such that $v_j = P_1 \cdots P_j e_j$. The Householder transformation P_{j+1} is chosen such that

$$P_{j+1} \cdots P_1 (v_1 \ A v_1 \ \dots \ A v_j)$$

has an upper triangular top $(j+1) \times (j+1)$ matrix denoted R_{j+1} and is zero elsewhere. Note that for $\ell = 2, \dots, j$ the first $\ell - 1$ components of the Householder vector corresponding to P_ℓ are zero. Therefore,

$$P_{j+1} \cdots P_1 (v_1 \ A v_1 \ \dots \ A v_j) = (P_1 v_1 \ P_2 P_1 A v_1 \ \dots \ P_{j+1} \cdots P_1 A v_j)$$

and P_{j+1} merely needs to zero out the desired components of $P_j \cdots P_1 A v_j$. Let us define $V_{n,j+1} = P_1 \dots P_{j+1} (e_1 \ \dots \ e_{j+1})$. Since $P_\ell = P_\ell^T = P_\ell^{-1}$, $\ell = 1, \dots, j+1$, the matrix $V_{n,j+1}$ is orthonormal. It can be proved by induction that the columns of $V_{n,j+1}$ span $\mathcal{K}_{n,j+1}(A, r_0)$; see [965].

From discarding the first column in

$$(v_1 \ A v_1 \ \dots \ A v_j) = V_{n,j+1} R_{j+1}$$

we obtain the Arnoldi-like relation

$$AV_{n,j} = V_{n,j+1} \underline{H}_j,$$

the size $(j+1) \times j$ upper Hessenberg matrix \underline{H}_j being R_{j+1} without first column. Thus the j th column of the upper Hessenberg matrix H can be obtained by computing $y = P_{j+1} \cdots P_1 A v_j$ and $H_{1:j+1,j} = y_{1:j+1}$. A basic implementation of this algorithm is the following.

```
function [V,H] = Arnoldi_Householder_basic(A,u,nitmax)
%
n = size(A,1);
V = zeros(n,nitmax);
H = zeros(nitmax,nitmax);
mi = norm(u);
u = u / mi;
[w,beta,P] = householder(u,2);
V(:,1) = P(:,1);
%
```

```

for j = 1:nitmax
    Av = A * V(:,j); % matrix vector product
    rr = P' * Av;
    if j < n
        mn = norm(rr(j+1:n));
    else
        mn = 0;
    end % if j
    if mn ~= 0
        if j+1 <= n
            [w,beta,PP] = householder(rr,j+2);
            rr = PP * rr;
        end % if j+1
    else
        w = zeros(n,1);
    end % if mn
    if j <= n-1
        H(1:j+1,j) = rr(1:j+1);
    else
        H(1:n,n) = rr(1:n);
    end % if j
    if norm(w) ~= 0
        P = P * PP;
    end % if norm
    if j < n
        V(:,j+1) = P(:,j+1);
    end % if j
end % for j

```

In practice it is not necessary to compute the Householder matrices P_ℓ . We can just compute the action of the transformation on a given vector. If $P_\ell = I - 2zz^T$ and y is a given vector, we have

$$P_\ell y = y - 2(z^T y)z.$$

This is what is used in the following implementation. Here for clarity we store the Householder vectors and the squares of their norms but this is not necessary. They can be re-computed when needed.

```

function [V,H] = Arnoldi_Householder(A,u,nitmax)
%
n = size(A,1);
m = nitmax;
V = zeros(n,m+1); % basis vectors
W = zeros(n,m+1); % Householder vectors
ww = zeros(1,m+1); % Householder vector norms
H = zeros(m+1,m);

```

```
mi = norm(u);
u = u / mi;
w = eye(n,1);
w(1) = 1;
sir = sign(u(1));
if sir == 0
    sir = 1;
end
if mi ~= 0
    beta = u(1) + sir * mi;
    w(2:n) = u(2:n) ./ beta;
end
W(:,1) = w;
wtw = w' * w;
ww(1) = wtw;
e1 = eye(n,1);
vv = e1 - 2 * (w(1) / wtw) * w;
V(:,1) = vv;
for j = 1:m
    Av = A * V(:,j); % matrix-vector product
    rAv = Av;
    for jj = 1:k
        wx = W(:,jj);
        wtAv = wx' * rAv;
        rAv = rAv - 2 * (wtAv / ww(jj)) * wx;
    end % for jj
    rr = rAv;
    if j < n
        mn = norm(rr(j+1:n));
    else
        mn = 0;
    end % if j
    if mn ~= 0
        if j+1 <= n
            w(1:j) = zeros(j,1);
            w(kj+1:n) = rr(j+1:n);
            mi = norm(w);
            if mi ~= 0 && j+1 < n
                beta = w(j+1) + sign(w(j+1)) * mi;
                if abs(beta) <= eps
                    beta = w(j+1) - sign(w(j+1)) * mi;
                end % if abs
                if abs(beta) <= eps % sign(0) = 0 in Matlab
                    beta = -mi;
                end % if abs
            end
        end
    end
end
```

```

w(j+2:n) = w(j+2:n) ./ beta;
end % if mi
w(j+1) = 1;
wtr = w' * rr;
rr = rr - 2 * (wtr / (w' * w)) * w;
end % if j+1
else
w = zeros(n,1);
end % if mn
W(:,j+1) = w;
ww(j+1) = w' * w;
if j <= n-1
H(1:j+1,j) = rr(1:j+1);
else
H(1:m,m) = rr(1:m);
end % if j
if j < n
if ww(j+1) > 0
ej1 = zeros(n,1);
ej1(j+1) = 1;
we = ej1 - 2 * (w(k+1) / ww(k+1)) * w;
for jj = kj:-1:1 % apply the preceding Householder
reflections
wx = W(:,jj);
wte = wx' * we;
we = we - 2 * (wte / ww(jj)) * wx;
end % for jj
else
% error
end % if ww
V(:,j+1) = we;
end % if j
end % for j

```

We next show that orthogonal bases for Krylov subspaces are, besides their desirable numerical properties, as well bases that lead to small Q-OR residual norms in a specific sense. From Chapter 3, Theorem 3.12, we know that the inverses of the Q-OR residual norms are given by the absolute values of the entries on the first row of U^{-1} which is defined by $K = VU$, see Theorem 3.1, where we assume that $\|r_0\| = 1$. It would be interesting to choose the entries of U to have these inverses as large as possible. The inverses of the matrices U_k are given recursively by

$$U_k^{-1} = \begin{pmatrix} U_{k-1}^{-1} - \frac{1}{u_{k,k}} U_{k-1}^{-1} U_{1:k-1,k} \\ 0 & \frac{1}{u_{k,k}} \end{pmatrix},$$

see (2.1). The entries of U^{-1} have been denoted as $\vartheta_{i,j}$. At step k assume we already know the entries of U_{k-1}^{-1} . Ideally we would like to choose the entries $\vartheta_{j,k}$ of column k of U^{-1} to maximize the $(1, k)$ entry of U_k^{-1} , that is, to maximize

$$\frac{1}{u_{k,k}} \left| \sum_{j=1}^{k-1} \vartheta_{1,j} u_{j,k} \right|.$$

Writing the k th columns of $K = VU$ (and assuming w.l.o.g. that $\|r_0\| = 1$) we obtain

$$A^{k-1}v_1 = \sum_{j=1}^k u_{j,k} v_j.$$

But

$$A^{k-1}v_1 = A(A^{k-2}v_1) = \sum_{j=1}^{k-1} u_{j,k-1} Av_j.$$

The columns of V have to be of unit norm, and we must have

$$u_{k,k} = \left\| \sum_{j=1}^{k-1} u_{j,k-1} Av_j - \sum_{j=1}^{k-1} u_{j,k} v_j \right\|, \quad (4.4)$$

as well as

$$v_k = \frac{1}{u_{k,k}} \sum_{j=1}^{k-1} (u_{j,k-1} Av_j - u_{j,k} v_j). \quad (4.5)$$

At step k the first sum in (4.5) is known. However, the function to maximize is non-differentiable and quite complicated. We can, however, use (4.4) and (4.5) to do something simpler and approximately obtain large entries on the first row of the inverse of U , namely, through forcing small diagonal entries $u_{k,k}$. Let us see how this works to construct a basis of $\mathcal{K}_k(A, v_1)$. The entry $u_{1,1}$ is equal to one.

We have $u_{2,2} = \|Av_1 - u_{1,2}v_1\|$. So now, we would like to pick $u_{1,2}$ to minimize $u_{2,2}$. This is a least squares problem. Obviously the solution is given by

$$u_{1,2} = \frac{v_1^T Av_1}{v_1^T v_1} = v_1^T Av_1.$$

It yields $v_2 = \frac{1}{u_{2,2}} [Av_1 - (v_1^T Av_1)v_1]$. It follows that

$$v_1^T v_2 = \frac{1}{u_{2,2}} v_1^T [Av_1 - (v_1^T Av_1)v_1] = 0.$$

Therefore, with this choice which minimizes the norm of the diagonal entry $u_{2,2}$ we have v_2 orthogonal to v_1 .

At the next step we would like to choose $u_{1,3}$ and $u_{2,3}$ to minimize $\|u_{2,2}Av_2 + u_{1,2}Av_1 - u_{2,3}v_2 - u_{1,3}v_1\|$. This is a least squares problem $\min \|c - By\|$ with

$$c = u_{2,2}Av_2 + u_{1,2}Av_1, \quad B = \begin{pmatrix} v_1 & v_2 \end{pmatrix}, \quad y = \begin{pmatrix} u_{1,3} \\ u_{2,3} \end{pmatrix}.$$

We can solve this problem using the normal equations,

$$B^T By = \begin{pmatrix} 1 & v_1^T v_2 \\ v_2^T v_1 & 1 \end{pmatrix} y = B^T c = \begin{pmatrix} v_1^T [u_{2,2}Av_2 + u_{1,2}Av_1] \\ v_2^T [u_{2,2}Av_2 + u_{1,2}Av_1] \end{pmatrix}.$$

Since the matrix $B^T B$ is equal to the identity because v_2 is orthogonal to v_1 , we obtain $y = B^T c$. But

$$v_3 = \frac{1}{u_{3,3}}[c - By] = \frac{1}{u_{3,3}}[c - BB^T c].$$

Let us consider $v_1^T v_3$. We have $v_1^T B = [1 \ 0]$; therefore

$$v_1^T BB^T c = v_1^T [u_{2,2}Av_2 + u_{1,2}Av_1] = v_1^T c.$$

Hence, $v_1^T v_3 = 0$. The same thing happens for $v_2^T v_3 = 0$. In fact, $I - BB^T$ is the projector on the orthogonal to the range of B . The new vector is orthogonal to the two previous ones. Now that we have understood the mechanism, we can prove the following result.

Theorem 4.1 *The basis obtained by minimizing the diagonal entries of U in $K = VU$ is orthonormal.*

Proof Assume that the basis vectors $v_j, j = 1, \dots, k-1$ are orthonormal. From (4.4), the least squares problem is $\min \|c - By\|$ with

$$B = V_{n,k-1}, \quad c = \sum_{j=1}^{k-1} u_{j,k-1} Av_j, \quad y = \begin{pmatrix} u_{1,k} \\ \vdots \\ u_{k-1,k} \end{pmatrix}.$$

The solution is $y = V_{n,k-1}^T c$, and the next basis vector is given by (4.5). Since $I - BB^T$ is the projector on the orthogonal of the space spanned by the vectors v_1, \dots, v_{k-1} , the vector v_k is orthogonal to all the previous basis vectors. \square

Therefore, when we minimize the diagonal entries of U we are constructing an orthonormal basis of the Krylov subspace. Since we use v_1 as a starting vector and because of the implicit Q-theorem [438, Chapter 7], this method is mathematically equivalent to the Arnoldi process, even though we are constructing U and not H . The matrix H can be recovered from U using the relation $H = UCU^{-1}$: Starting with $He_1 = Ue_2$, we have the recurrence

$$He_j = \frac{Ue_{j+1} - \sum_{i=1}^{j-1} u_{ij}He_i}{u_{jj}}, \quad j = 2, \dots, n-1$$

and for the last column of H (which is rarely needed in practice) there holds

$$He_n = \frac{\sum_{i=1}^n c_{in} Ue_i - \sum_{i=1}^{n-1} u_{ij}He_i}{u_{nn}}.$$

Note that the previous algorithm is more costly than the Arnoldi process in number of operations and storage since we have to store the vectors Av_j . Moreover, it is not very efficient numerically since orthogonality is lost sooner than with Arnoldi-CGS.

4.2 Hessenberg basis

Let us consider a rectangular LU factorization of the $n \times k$ Krylov matrix $K_{n,k}$ defined in (4.1) of the form $K_{n,k} = L_{n,k}R_k$ where $L_{n,k}$ is a lower trapezoidal $n \times k$ matrix (i.e., its top $k \times k$ submatrix is lower triangular) and R_k is an upper triangular matrix of order k with ones on the main diagonal. This factorization exists if the leading principal minors (the determinants corresponding to leading principal submatrices) of $K_{n,k}$ are nonzero. The following result was proved in [805].

Theorem 4.2 *Let $K_{n,k}$ be the Krylov matrix corresponding to $\mathcal{K}_k(A, r_0)$. Let us assume that the principal minors of $K_{n,k}$ are nonzero. Then, the columns ℓ_j of the lower trapezoidal matrix $L_{n,k}$, such that $K_{n,k} = L_{n,k}R_k$ with R_k upper triangular and ones on the main diagonal, are given by $\ell_1 = r_0$ and*

$$\ell_j = A\ell_{j-1} - L_{n,j-1}L_{j-1,n}^L A\ell_{j-1}, \quad j = 2, \dots, k, \quad (4.6)$$

where $L_{j-1,n}^L = ([L_{j-1}^{(1)}]^{-1} \ 0)$, $L_{j-1}^{(1)}$ being the top $(j-1) \times (j-1)$ submatrix of $L_{n,j-1}$. Moreover, ℓ_1, \dots, ℓ_k span the Krylov subspace $\mathcal{K}_k(A, r_0)$ and

$$AL_{n,k-1} = L_{n,k}\tilde{H}_{k-1}, \quad (4.7)$$

where \tilde{H}_{k-1} is a $k \times (k-1)$ upper Hessenberg matrix with ones on the subdiagonal, the other nonzero entries of the j th column being $L_{j,n}^L A\ell_j$.

Proof Let k_j , $j = 1, \dots, k$ be the columns of $K_{n,k}$. From [535] we obtain that

$$\ell_j = k_j - K_{n,j-1}K_{j-1,n}^L k_j, \quad K_{j-1,n}^L = ([K_{j-1}^{(1)}]^{-1} \ 0).$$

Since $k_j = Ak_{j-1}$ we have

$$k_j = A\ell_{j-1} + AK_{n,j-2}K_{j-2,n}^L k_{j-1} = A\ell_{j-1} + K_{n,j-1}g_j,$$

with the first component of the vector g_j being zero. Using $K_{n,j-1}K_{j-1,n}^L K_{n,j-1} = K_{n,j-1}$ we have

$$\begin{aligned}
\ell_j &= k_j - K_{n,j-1} K_{j-1,n}^L k_j, \\
&= A\ell_{j-1} + K_{n,j-1} g_j - K_{n,j-1} K_{j-1,n}^L (A\ell_{j-1} + K_{n,j-1} g_j), \\
&= A\ell_{j-1} - K_{n,j-1} K_{j-1,n}^L A\ell_{j-1}.
\end{aligned}$$

But $K_{j-1,n}^L = \left([K_{j-1}^{(1)}]^{-1} \ 0 \right)$ and $K_{j-1}^{(1)} = L_{j-1}^{(1)} R_j$. It yields $K_{n,j-1} K_{j-1,n}^L = L_{n,j-1} L_{j-1,n}^L$ which proves relation (4.6).

From the factorization of $K_{n,k}$ it is obvious that the columns of $L_{n,k}$ span the Krylov subspace $\mathcal{K}_k(A, r_0)$. We prove relation (4.7) by induction. We have

$$\ell_2 = A\ell_1 - \tilde{h}_{1,1}\ell_1, \quad \tilde{h}_{1,1} = \frac{[A\ell_1]_1}{[\ell_1]_1}.$$

It yields

$$A\ell_1 = (\ell_1 \ \ell_2) \begin{pmatrix} \tilde{h}_{1,1} \\ 1 \end{pmatrix} = L_{n,2} \tilde{H}_1.$$

Assume the relation holds for $j - 1$. Then,

$$\begin{aligned}
AL_{n,j} &= (AL_{n,j-1} \ A\ell_j), \\
&= \left(AL_{n,j-1} \ \ell_{j+1} + L_{n,j} L_{j,n}^L A\ell_j \right), \\
&= \left(L_{n,j} \tilde{H}_{j-1} \ \ell_{j+1} + L_{n,j} L_{j,n}^L A\ell_j \right), \\
&= \left(L_{n,j} \ \ell_{j+1} \right) \begin{pmatrix} \tilde{H}_{j-1} & L_{j,n}^L A\ell_j \\ 0 & 1 \end{pmatrix}, \\
&= L_{n,j+1} \tilde{H}_j.
\end{aligned}$$

This holds for $j = 1, \dots, k - 1$. □

There are two things we have to modify to obtain a practical algorithm. First, the factorization can break down if $[\ell_j]_j = 0$ for some j and if we have a small $[\ell_j]_j$ we may have stability problems. This was solved in [805] by using a pivoting strategy as in Gaussian elimination. The matrix $L_{n,k}$ is no longer trapezoidal but there exists a permutation matrix P such that $PL_{n,k}$ is trapezoidal. Second, we have to normalize the vectors ℓ_j in some way.

We introduce a permutation vector s to represent P . Assume that s_1, \dots, s_j are known, and we would like to determine s_{j+1} . We compute $w = A\ell_j$, and we subtract multiples of the vectors ℓ_1, \dots, ℓ_j to annihilate the components s_1, \dots, s_j of w . This yields a new vector w , and we look for the index i giving the maximum of the moduli of the components of w . Then, we set $s_{j+1} = i$ and we normalize w by the maximum. Note that because of the permutations, the subdiagonal entries of H are no longer equal to 1. This algorithm is summarized in the following code.

```

function [L,H,s] = Hessenberg(A,u,nitmax)
%
n = size(A,1);
H = zeros(nitmax,nitmax);
L = zeros(n,nitmax);
s = [1:n]';
i0 = index(u);
L(:,1) = u / u(i0);
s = swap(s,1,i0);
for j = 1:nitmax
    w = A * L(:,j);
    for i = 1:j
        c = w(s(i));
        H(i,j) = c;
        w(s(i)) = 0;
        si = s(i+1:n);
        w(si) = w(si) - c * L(si,i);
    end % for i
    if j < nitmax
        sj = s(j+1:n);
        i1 = index(w(sj));
        i0 = i1 + j;
        s = swap(s,j+1,i0);
        H(j+1,j) = w(sj(i1));
        L(:,j+1) = w / H(j+1,j);
    end % if
end % for i
end

function s = swap(s,i,j);
ss = s(i);
s(i) = s(j);
s(j) = ss;
end

function i0 = index(u);
[y,I] = max(abs(u));
i0 = I(1);
end

```

We remark that there is no dot product in this algorithm. However, since we have to pivot for stability, we have to look for the maximum of the components of a vector, and this is also a reduction operation on parallel computers. The previous code is simple but very inefficient computationally because of the indirect addressing in $w(si) = w(si) - c * L(si,i)$. A more efficient, but more complicated, code will be described in Chapter 7.

4.3 Biorthogonal basis

The main problem concerning the bases we have constructed in the previous sections is that to compute a new basis vector we need to have access to all the previous ones. When the order of the matrix A is large, this can lead to storage problems in practical computations. Moreover, the computational costs of each step increase when we are adding a new vector to the basis.

When the matrix A is symmetric it is known that the Arnoldi process reduces to the Lanczos algorithm (which was, in fact, introduced earlier). Using the symmetry property this algorithm needs only the last two previous vectors to compute the next basis vector; see, for instance, [676]. When A is nonsymmetric it is in general not possible to construct an orthogonal basis with a three-term recurrence using only matrix–vector products with A ; see [334, 335, 338, 640]. However, a three-term recurrence algorithm for nonsymmetric matrices was introduced by Cornelius Lanczos in [615]; see also [616]. It uses the transpose matrix A^T (or the conjugate transpose A^* in the complex case).

We construct two sets of vectors v_j and w_j for $j = 1, \dots, k$ as follows. Let v_1 and w_1 be given such that $(v_1, w_1) \neq 0$. Then, assuming that v_1, \dots, v_j and w_1, \dots, w_j are known and biorthogonal, that is,

$$(v_i, w_j) = 0, \quad i \neq j, \quad (v_i, w_i) \neq 0, \quad i = 1, \dots, j,$$

we define the vectors

$$\begin{aligned} \tilde{v}_j &= Av_j - \sum_{i=1}^j h_{i,j} v_i, \\ \tilde{w}_j &= A^T w_j - \sum_{i=1}^j \hat{h}_{i,j} w_i, \end{aligned}$$

which are to become, after the normalizations $v_{j+1} = \tilde{v}_j / h_{j+1,1}$ and $w_{j+1} = \tilde{w}_j / \hat{h}_{j+1,1}$, the next basis vectors v_{j+1} and w_{j+1} . We will assume that $(\tilde{v}_j, \tilde{w}_j)$ is nonzero; this guarantees that the process does not break down at the j th iteration.

To compute the coefficients $h_{i,j}$ and $\hat{h}_{i,j}$ we require that \tilde{v}_j is orthogonal to w_i , $i = 1, \dots, j$ and that \tilde{w}_j is orthogonal to v_i , $i = 1, \dots, j$. This property is called *biorthogonality*. Multiplying the first relation by w_i^T and the second by v_i^T , $i \leq j$, we obtain

$$h_{i,j} = \frac{(Av_j, w_i)}{(w_i, v_i)}, \quad \hat{h}_{i,j} = \frac{(A^T w_j, v_i)}{(w_i, v_i)}, \quad i = 1, \dots, j.$$

In matrix form, the two relations for \tilde{v}_j and \tilde{w}_j give for the j th iteration of the biorthogonalization process

$$AV_{n,j} = V_{n,j}H_j + \tilde{v}_j e_j^T, \quad A^T W_{n,j} = W_{n,j}\hat{H}_j + \tilde{w}_j e_j^T, \quad (4.8)$$

with upper Hessenberg matrices H_j and \hat{H}_j . The biorthogonality of the previously defined basis vectors v_1, \dots, v_j and w_1, \dots, w_j implies that

$$h_{i,\ell} = \frac{(Av_\ell, w_i)}{(w_i, v_i)}, \quad \hat{h}_{i,\ell} = \frac{(A^T w_\ell, v_i)}{(w_i, v_i)}, \quad i = 1, \dots, \ell, \quad \ell = 1, \dots, j-1$$

and, as is easily verified, we also have $h_{\ell+1,\ell} = \frac{(Av_\ell, w_{\ell+1})}{(w_{\ell+1}, v_{\ell+1})}$, $\hat{h}_{\ell+1,\ell} = \frac{(A^T w_\ell, v_{\ell+1})}{(w_{\ell+1}, v_{\ell+1})}$ for $\ell = 1, \dots, j-1$.

Let D_j be the diagonal matrix $W_{n,j}^T V_{n,j} = V_{n,j}^T W_{n,j}$ with diagonal entries $(w_1, v_1), (w_2, v_2), \dots, (w_j, v_j)$. Then from (4.8) we have

$$D_j \hat{H}_j = V_{n,j}^T A^T W_{n,j} = (W_{n,j}^T A V_{n,j})^T = (D_j H_j)^T = H_j^T D_j.$$

We see that H_j^T must be upper Hessenberg, because so is \hat{H}_j . Therefore, the matrix H_j is a (nonsymmetric) tridiagonal matrix and so is \hat{H}_j . For this reason we change the notation for H_j to T_j , i.e.,

$$T_j \equiv H_j.$$

It follows that $\hat{H}_j = D_j^{-1} T_j^T D_j$. The previous matrix equations (4.8) can then be written as

$$AV_{n,j} = V_{n,j} T_j + \tilde{v}_j e_j^T, \quad A^T W_{n,j} = W_{n,j} D_j^{-1} T_j^T D_j + \tilde{w}_j e_j^T. \quad (4.9)$$

Moreover, we have

$$W_{n,j}^T A V_{n,j} = W_{n,j}^T V_{n,j} T_j = D_j T_j.$$

Let us introduce the notation

$$\alpha_i \equiv t_{i,i} = \frac{(Av_i, w_i)}{(w_i, v_i)}, \quad i = 1, \dots, j \quad (4.10)$$

$$\beta_i \equiv t_{i-1,i} = \frac{(Av_i, w_{i-1})}{(w_{i-1}, v_{i-1})}, \quad i = 2, \dots, j \quad (4.11)$$

$$\delta_i \equiv t_{i,i-1} \frac{d_{i,i}}{d_{i-1,i-1}} = \frac{(Av_{i-1}, w_i)}{(w_{i-1}, v_{i-1})}, \quad i = 2, \dots, j. \quad (4.12)$$

Then the relations for \tilde{v}_j and \tilde{w}_j can be written as

$$\tilde{v}_j = Av_j - \alpha_j v_j - \beta_j v_{j-1}, \quad (4.13)$$

$$\tilde{w}_j = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}, \quad (4.14)$$

and the matrix T_j in (4.9) is

$$T_j = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 \frac{d_{1,1}}{d_{2,2}} \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{j-1} \frac{d_{j-2,j-2}}{d_{j-1,j-1}} \alpha_{j-1} & \beta_j & \\ & & & \delta_j \frac{d_{j-1,j-1}}{d_{j,j}} \alpha_j & \end{pmatrix},$$

whereas the matrix $D_j^{-1}T_j^T D_j$ in (4.9) is

$$D_j^{-1}T_j^T D_j = \begin{pmatrix} \alpha_1 & \delta_2 & & & \\ \beta_2 \frac{d_{1,1}}{d_{2,2}} \alpha_2 & \delta_3 & & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{j-1} \frac{d_{j-2,j-2}}{d_{j-1,j-1}} \alpha_{j-1} & \delta_j & \\ & & & \beta_j \frac{d_{j-1,j-1}}{d_{j,j}} \alpha_j & \end{pmatrix}.$$

Lemma 4.1 *The tridiagonal matrix $D_j^{-1}T_j^T D_j$ involved in the second equation of (4.9) is diagonally similar to the tridiagonal matrix T_j involved in the first equation of (4.9).*

Proof The claim holds if

$$T_j^T = \Delta_j T_j \Delta_j^{-1} \quad (4.15)$$

for a nonsingular diagonal matrix Δ_j . Let $\Delta_{i,i}$ denote the entries of that diagonal matrix, and let us define $\Delta_{j,j} = \beta_j$ and $\Delta_{j-1,j-1} = \delta_j \frac{d_{j-1,j-1}}{d_{j,j}}$. Then the positions $(j-1, j)$ and $(j, j-1)$ on both sides of equation (4.15) coincide. Equating the positions $(j-2, j-1)$ and $(j-1, j-2)$ on both sides of (4.15) gives the conditions

$$\frac{\Delta_{j-2,j-2}}{\Delta_{j-1,j-1}} \beta_{j-1} = \delta_{j-1} \frac{d_{j-2,j-2}}{d_{j-1,j-1}}, \quad \frac{\Delta_{j-1,j-1}}{\Delta_{j-2,j-2}} \delta_{j-1} \frac{d_{j-2,j-2}}{d_{j-1,j-1}} = \beta_{j-1},$$

which are satisfied with the choice $\Delta_{j-2,j-2} = \left(\Delta_{j-1,j-1} \delta_{j-1} \frac{d_{j-2,j-2}}{d_{j-1,j-1}} \right) / \beta_{j-1}$. The remaining diagonal entries follow analogously; their values are

$$\Delta_{i-1,i-1} = \left(\Delta_{i,i} \delta_i \frac{d_{i-1,i-1}}{d_{i,i}} \right) / \beta_i, \quad i = j-1, j-2, \dots, 2,$$

and this concludes the proof. \square

The vectors \tilde{v}_j and \tilde{w}_j have to be normalized to obtain v_{j+1} and w_{j+1} . We can rewrite (4.13)–(4.14) as

$$\begin{aligned}\delta_{j+1} \frac{d_{j,j}}{d_{j+1,j+1}} v_{j+1} &= Av_j - \alpha_j v_j - \beta_j v_{j-1}, \\ \beta_{j+1} \frac{d_{j,j}}{d_{j+1,j+1}} w_{j+1} &= A^T w_j - \alpha_j w_j - \delta_j w_{j-1},\end{aligned}$$

and consider the scaling factors $\tilde{\beta}_{j+1} = \beta_{j+1} \frac{d_{j,j}}{d_{j+1,j+1}}$ and $\tilde{\delta}_{j+1} = \delta_{j+1} \frac{d_{j,j}}{d_{j+1,j+1}}$ for \tilde{w}_j and \tilde{v}_j , respectively. In practice, the scaling factors $\tilde{\beta}_{j+1}$ and $\tilde{\delta}_{j+1}$ are chosen, and the coefficients β_{j+1} and δ_{j+1} are computed from them, after the vectors \tilde{v}_j and \tilde{w}_j have been scaled and v_j and w_j have been obtained.

The normalization can be done in many different ways. One possibility is to choose the normalizing factors such that

$$\tilde{\beta}_{j+1} \tilde{\delta}_{j+1} = (\tilde{v}_j, \tilde{w}_j).$$

With this choice, we have $(v_{j+1}, w_{j+1}) = 1$. We can choose $\tilde{\delta}_{j+1} = \sqrt{|(\tilde{v}_j, \tilde{w}_j)|}$ and $\tilde{\beta}_{j+1} = \pm \tilde{\delta}_{j+1}$ depending on the sign of $(\tilde{v}_j, \tilde{w}_j)$. A different normalization, if we want to have basis vectors of unit norm, is to use $\tilde{\delta}_{j+1} = \|\tilde{v}_j\|$, $\tilde{\beta}_{j+1} = \|\tilde{w}_j\|$ in which case obviously $\beta_{j+1} = \tilde{\beta}_{j+1}$, $\delta_{j+1} = \tilde{\delta}_{j+1}$ and

$$v_{j+1} = \frac{\tilde{v}_j}{\delta_{j+1}}, \quad w_{j+1} = \frac{\tilde{w}_j}{\beta_{j+1}}.$$

For other normalizations of the basis vectors, see [482]. The basis vectors can, for instance, in the complex case, be scaled such that the resulting tridiagonal matrix T_j is complex symmetric.

We have a breakdown of the algorithm if there is a step with $(\tilde{v}_j, \tilde{w}_j) = 0$. There are different possible situations. First, one of the vectors \tilde{v}_j or \tilde{w}_j can be zero. In both cases we have found an invariant subspace. If $\tilde{v}_j = 0$ we can compute the solution of the linear system $Ax = b$. If $\tilde{w}_j = 0$, we can compute the solution of a “transposed” linear system $A^T x = c$ if that is desired. This presumes w_1 corresponds to a normalized initial residual for the transposed system. Otherwise, if the solution of the original linear system is the only goal, one can restart the algorithm with another initial vector w_1 . We remark that the influence of the choice of the *shadow vector* w_1 has not been very intensively studied. Papers about this topic are [456, 910].

A more critical situation, which is called a serious breakdown, is when the dot product $(\tilde{v}_j, \tilde{w}_j)$ is equal to zero with $\tilde{v}_j \neq 0$ and $\tilde{w}_j \neq 0$. A way to solve this problem is to use a *look-ahead strategy*, see, for instance, [138, 373, 379]. The various types of breakdowns and look-ahead strategies are treated in detail in Chapter 8 about the BiCG and QMR methods.

Theorem 4.3 *The vectors v_1, \dots, v_k span the Krylov subspace $\mathcal{K}_k(A, v_1)$, and the vectors w_1, \dots, w_k span the Krylov subspace $\mathcal{K}_k(A^T, w_1)$. The basis vectors can be written as*

$$v_k = p_k(A)v_1, \quad w_k = p_k(A^T)w_1,$$

where p_k is a polynomial of degree k .

Proof The result is obvious from the definition of the basis vectors. \square

A code which delivers basis vectors of unit norm is as follows.

```
function [V,W,T] = biortho(A,v,w,nitmax)
v = v / norm(v);
w = w / norm(w);
n = size(A,1);
v1 = zeros(n,1);
w1 = v1;
T = sparse(nitmax,nitmax);
V = zeros(n,nitmax);
W = zeros(n,nitmax);
At = A';
beta = 0;
rho = 1;
zeta = 1;
wv = w' * v;
for k = 1:nitmax
    V(:,k) = v;
    W(:,k) = w;
    Av = A * v;
    Aw = At * w;
    alpha = (w' * Av) / wv;
    vt = Av - alpha * v - beta * v1;
    wt = Aw - alpha * w - (beta * rho / zeta) * w1;
    rho = norm(vt);
    zeta = norm(wt);
    v1 = v;
    w1 = w;
    v = vt / rho;
    w = wt / zeta;
    wvnew = w' * v;
    % we must check the near breakdown, wvnew small
    beta = zeta * wvnew / wv;
    wv = wvnew;
    T(k,k) = alpha;
    T(k,k+1) = beta;
    T(k+1,k) = rho;
end % for k
```

This code may not be very efficient because of the sparse addressing in T. If nitmax is not too large it could be more efficient to initialize as T = zeros(nitmax,nitmax).

The previous algorithms use three-term recurrences which are reminiscent of the classical Gram–Schmidt algorithm or of the recurrence in the Lanczos algorithm. Hence, we can possibly get the same problems, that is, a loss of biorthogonality in some cases.

Since three-term recurrences are numerically prone to instability, it is advisable to consider algorithms which use two-term recurrences. As an example, we choose an algorithm described in [482]. Two new sets of vectors p_k (not to be confused with the polynomials p_k) and q_k are introduced which allow to obtain two-term recurrences. These new vectors are A-biconjugate which means that $(Ap_i, q_j) = 0, i \neq j$. In the implementation below the basis vectors are of unit norm.

```
function [V,W,T] = biortho_2t(A,v,w,nitmax)
v = v / norm(v);
w = w / norm(w);
n = size(A,1);
p = v;
q = w;
T = sparse(nitmax,nitmax);
V = zeros(n,nitmax);
W = zeros(n,nitmax);
At = A';
Ap = A * p;
Atq = At * q;
alpha = v' * w;
delta = Ap' * q;
for k = 1:nitmax
    V(:,k) = v;
    W(:,k) = w;
    phi = delta / alpha;
    vt = Ap - phi * v;
    wt = Atq - phi * w;
    rho = norm(vt);
    zeta = norm(wt);
    v = vt / rho;
    w = wt / zeta;
    alpha = v' * w;
    % we must check the breakdowns
    psi = zeta * alpha / delta;
    psit = rho * alpha / delta;
    p = v - psi * p;
    q = w - psit * q;
    Ap = A * p;
```

```

Atq = At * q;
delta = Ap' * q;
T(k,k) = 1;
T(k,k+1) = 0;
T(k+1,k) = 0;
end % for k

```

We note that in this two-term recurrence algorithm there is one more potential source of breakdown. Namely, the scalar $\delta_k = (Ap_k, q_k)$ can be zero or too small. Hence, this algorithm may break down earlier than the three-term version.

The appealing property of biorthonormal bases is that we have short recurrences to construct the basis vectors. So, unless we need them for some other purposes, when solving linear systems we do not have to store too many recurrence vectors. The drawbacks are that there are additional possibilities of breakdown (or near-breakdown) and that two matrix–vector products per iteration are needed. There are cases where it could be difficult to perform matrix–vector products with A^T . For example, if the matrix A is distributed on the processors of a parallel computer, the data distribution is generally done to optimize the matrix–vector product with A and, therefore, the matrix–vector product with A^T may be far from being efficient. Another example is matrix-free implementations where the matrix–vector product with A is replaced with a difference operator, but it may not be clear how to perform the matrix–vector product with A^T without storing A^T .

Moreover, in finite precision computations the basis vectors may lose their biorthogonality properties. Ways of maintaining these properties have been considered in [258, 259].

4.4 The generalized Hessenberg process

The generalized Hessenberg process can be regarded as a general framework with the previously treated bases representing special choices within that framework; see, for instance, [982].

As before, we compute the ascending basis vectors of $\mathcal{K}_k(A, r_0)$ by choosing $v_1 = r_0$ and with subsequent basis vectors of the form

$$v_{j+1} = Av_j - \sum_{i=1}^j h_{i,j} v_i. \quad (4.16)$$

Let $Y_j = (y_1 \dots y_j)$ be any given matrix with linearly independent columns of size n . In the generalized Hessenberg process, the coefficients $h_{i,j}$ are determined by enforcing the orthogonality of v_{j+1} with respect to the columns of Y_j . In matrix form we will obtain

$$AV_j = V_{j+1} \underline{H}_j, \quad Y_j^T V_j = L_j,$$

where \underline{H}_j is a $(j+1) \times j$ upper Hessenberg matrix with ones on the first subdiagonal and L_j is a lower triangular matrix. To achieve this, first let $\eta_1 = (v_1, y_1)$. Then the entries of \underline{H}_j are computed for $j = 1, \dots$ as

$$\begin{aligned} u &= Av_j, \\ h_{i,j} &= \frac{(y_i, u)}{\eta_i}, \quad u = u - h_{i,j} v_i, \quad i = 1, \dots, j, \\ h_{j+1,j} &= 1, \quad v_{j+1} = u, \quad \eta_{j+1} = (y_{j+1}, v_{j+1}). \end{aligned}$$

Note that the basis vectors are not of unit norm.

If the vectors y_j are the columns of the identity matrix we obtain the Hessenberg basis without permutations described in Section 4.2. However, if there is a breakdown due to $\eta_{j+1} = 0$, the order of the vectors y_j can be changed.

We do not need to choose the vectors y_j a priori. They can be chosen on the fly. For instance, we may choose $y_j = v_j$. Then, we recover the Arnoldi algorithm with a different scaling of the basis vectors. Another possible choice is to choose vectors spanning the Krylov subspaces for A^T and w_1 , for example, if w_1 corresponds to the initial residual of a “transposed” linear system. In the case $w_1 = r_0$, we obtain an algorithm which is close to what is known as the *one-sided Lanczos algorithm*. In the biorthogonalization algorithms in Section 4.3, the matrix $Y_j^T V_j$ is not only lower triangular but even diagonal. In [790] Saad remarked that in the biorthogonalization algorithms we can choose the vectors spanning $\mathcal{K}_k(A^T, r_0)$ almost as we wish. For instance, $y_j = q_j(A^T)y_1$ where q_j is a polynomial of degree j . Hence, this is also a particular case of the generalized Hessenberg process.

4.5 Q-OR optimal basis

In this section we derive an ascending basis which is optimal for Q-OR methods in the sense that, given the current iteration, it defines the next basis vector such that the resulting iterate yields the smallest possible Q-OR residual norm. We recall that in Section 4.1 we already made a preliminary attempt to define optimal bases through minimization of the diagonal entries of the matrix U . This resulted in a standard orthogonal basis (see Theorem 4.1).

Let us assume that we have any ascending basis of the Krylov subspaces defined by the columns of unit norm of a matrix V such that $K = VU$. From Theorem 3.12 we see that, if we want to use this basis in a Q-OR method, it can be beneficial to maximize the absolute values of the entries of the first row of U^{-1} . Moreover, we have seen in Lemma 2.2, relation (2.21), that we have a relation between the first row of the upper triangular matrix U^{-1} with entries denoted $\vartheta_{1,j}$ and the k th column of the upper Hessenberg matrix H ,

$$\vartheta_{1,k+1} = -\frac{1}{h_{k+1,k}} \sum_{j=1}^k \vartheta_{1,j} h_{j,k}, \quad k = 1, \dots, n-1.$$

Suppose that at step k of the algorithm we have already computed $\vartheta_{1,j}$, $j = 1, \dots, k$. We would like to choose $h_{j,k}$, $j = 1, \dots, k+1$ to maximize the absolute value of $\vartheta_{1,k+1}$, knowing that $h_{k+1,k}$ has to be chosen to obtain a vector v_{k+1} of unit norm. From the relation (3.1), the subdiagonal entry $h_{k+1,k}$ is given by the norm of the vector

$$\tilde{v}_k = Av_k - \sum_{j=1}^k h_{j,k} v_j, \quad (4.17)$$

and the next basis vector is $v_{k+1} = \tilde{v}_k / h_{k+1,k}$ with $h_{k+1,k} = \|\tilde{v}_k\|$.

For simplicity, let us introduce the notation

$$\begin{aligned} d &= Av_k, \quad B = V_k = (v_1 \cdots v_k), \\ y &= (h_{1,k} \cdots h_{k,k})^T, \quad \vartheta = (\vartheta_{1,1} \cdots \vartheta_{1,k})^T. \end{aligned}$$

Then, from Lemma 2.2,

$$|\vartheta_{1,k+1}| = \frac{|\vartheta^T y|}{\|d - By\|}.$$

We would like to maximize $|\vartheta_{1,k+1}|^2$. Hence, considering the reciprocal, we have to solve

$$\min_{\substack{y \in \mathbb{R}^k \\ \vartheta^T y \neq 0}} \frac{1}{|\vartheta_{1,k+1}|^2} = \min_{\substack{y \in \mathbb{R}^k \\ \vartheta^T y \neq 0}} \frac{\|d - By\|^2}{(\vartheta^T y)^2}. \quad (4.18)$$

In other words we wish to find the minimum of a rational function of several variables. To solve this problem we need some technical results that were proved in [685].

Lemma 4.2 *Let S be a square real symmetric positive definite matrix of order n with a spectral factorization $S = Q\Lambda Q^T$ with $\Lambda = \text{diag}(\lambda_j)$, λ_j , $j = 1, \dots, n$ being the eigenvalues of S and $Q^T Q = I$. Let $c \in \mathbb{R}^n$ which is not orthogonal to any of the eigenvectors of S and γ be a real positive number. Then, the eigenvalues μ_j , $j = 1, \dots, n$ of the matrix $S - \gamma c c^T$ are given by the solutions of the secular equation*

$$f(\mu) \equiv 1 - \gamma \sum_{j=1}^n \frac{(z_j)^2}{\lambda_j - \mu} = 0, \quad (4.19)$$

with $z = Q^T c$.

Proof See, e.g., [428], [438, Chapter 8.5] or [432]. Let us look for an eigenpair (μ, y) of the rank-one modification of S such that

$$(S - \gamma c c^T)y = \mu y.$$

Since

$$(S - \mu I)y = \gamma c c^T y,$$

taking the inverse of $S - \mu I$ for $\mu \neq \lambda_j$, $j = 1, \dots, n$ and multiplying by c^T , it yields $c^T y = \gamma c^T (S - \mu I)^{-1} c c^T y$. But $c^T y \neq 0$, since otherwise y would be an eigenvector of S and c would be orthogonal to such an eigenvector. Therefore, μ must satisfy the equation

$$1 - \gamma c^T (S - \mu I)^{-1} c = 0.$$

Using the spectral factorization of S and $z = Q^T c$, we obtain the so-called secular equation (4.19). \square

The following corollary allows us to find under which conditions the matrix $S - \gamma c c^T$ is positive definite.

Corollary 4.1 *Using the hypotheses and the notation of Lemma 4.2, let*

$$\gamma_{opt} = \frac{1}{\sum_{j=1}^n \frac{(z_j)^2}{\lambda_j}} = \frac{1}{c^T S^{-1} c}.$$

Then $S - \gamma c c^T$ is positive definite if $0 \leq \gamma < \gamma_{opt}$, positive semi-definite if $\gamma = \gamma_{opt}$ and indefinite if $\gamma > \gamma_{opt}$.

Proof Obviously the rank-one modification of S is positive definite if $\gamma = 0$. So, let us assume $\gamma > 0$. The function f defined in (4.19) has poles at the eigenvalues λ_j of S which are positive. Computing the derivative outside of the poles, we see that the function f is decreasing in between the poles. Hence, since $\gamma > 0$, there is an interlacing between the eigenvalues of S and the eigenvalues of $S - \gamma c c^T$. This is obvious if the eigenvalues of S are distinct. If there is, for instance, a double eigenvalue, two poles of the secular function are the same and this eigenvalue is also an eigenvalue of $S - \gamma c c^T$. Since S is positive definite, only the smallest eigenvalue of $S - \gamma c c^T$ can possibly be negative. The function f tends to 1 from below when $\mu \rightarrow -\infty$. From (4.19) we have

$$f(0) = 1 - \gamma \sum_{j=1}^n \frac{(z_j)^2}{\lambda_j}.$$

This value is positive when $\gamma < \gamma_{opt}$ and $S - \gamma c c^T$ is singular when $\gamma = \gamma_{opt}$. \square

In the next proposition we consider the minimization problem for the rational function formulated in (4.18).

Proposition 4.1 *Let B be an $n \times k$ real matrix with $n > k$ and $d \in \mathbb{R}^n$, $d \neq 0$ such that the matrix $B_d = (B, -d)$ is of full-rank $k + 1$. Let $\vartheta \neq 0$ be a given vector in \mathbb{R}^k such that the vector ϑ appended with a zero at the bottom is not orthogonal to any eigenvector of $B_d^T B_d$. Let*

$$\delta_{opt} = \min_{y \in \mathbb{R}^k, \vartheta^T y \neq 0} \frac{\|d - By\|^2}{(\vartheta^T y)^2}. \quad (4.20)$$

Then, if $B^T d \neq 0$,

$$\delta_{opt} = \frac{\alpha}{\alpha \vartheta^T (B^T B)^{-1} \vartheta + \omega^2}, \quad (4.21)$$

with

$$\alpha = d^T d - d^T B (B^T B)^{-1} B^T d > 0 \quad (4.22)$$

and

$$\omega = d^T B (B^T B)^{-1} \vartheta. \quad (4.23)$$

Moreover, if $\omega \neq 0$, a solution y_{opt} of the minimization problem (4.20) is given by

$$y_{opt} = (B^T B)^{-1} B^T d + \frac{\alpha}{\omega} (B^T B)^{-1} \vartheta. \quad (4.24)$$

If $B^T d = 0$, the minimization problem (4.20) has no finite solution.

Proof Note that $\delta_{opt} \geq 0$, but, with our hypothesis d is not in the range of B and there is no vector y such that $\|d - By\| = 0$. Hence, $\delta_{opt} > 0$.

We use the same technique as in [718] and [572]. We consider the problem

$$\sup_{\delta > 0} \delta \text{ such that } \|d - By\|^2 \geq \delta (\vartheta^T y)^2 \text{ for all } y \in \mathbb{R}^k, \vartheta^T y \neq 0. \quad (4.25)$$

From Theorem 2 in [572] this yields the solution of the minimization problem (4.20). We have

$$\|d - By\|^2 = d^T d - y^T B^T d - d^T B y + y^T B^T B y, \quad (\vartheta^T y)^2 = y^T \vartheta \vartheta^T y.$$

Therefore, the problem (4.25) can be written in matrix form as

$$\sup_{\delta > 0} \delta \text{ such that } (y^T \ 1) C(\delta) \begin{pmatrix} y \\ 1 \end{pmatrix} \geq 0, \text{ for all } y \in \mathbb{R}^k, \vartheta^T y \neq 0, \quad (4.26)$$

where

$$C(\delta) = \begin{pmatrix} B^T B - \delta \vartheta \vartheta^T & -B^T d \\ -d^T B & d^T d \end{pmatrix}.$$

We remark that the matrix $C(\delta)$ can be written as

$$C(\delta) = B_d^T B_d - \delta \begin{pmatrix} \vartheta \\ 0 \end{pmatrix} (\vartheta^T \ 0), \quad B_d = (B \ -d).$$

Hence, $C(\delta)$ is a rank-one perturbation of a symmetric positive definite matrix. We can apply the results of Corollary 4.1. There exists a unique value

$$\delta = \delta_{opt} = \left((\vartheta^T \ 0) (B_d^T B_d)^{-1} \begin{pmatrix} \vartheta \\ 0 \end{pmatrix} \right)^{-1}$$

for which the matrix $C(\delta)$ is positive semi-definite. If $0 \leq \delta < \delta_{opt}$ the matrix $C(\delta)$ is positive definite and if $\delta > \delta_{opt}$ the matrix $C(\delta)$ is indefinite. Therefore, δ_{opt} is the largest value for which the relation (4.26) is true for all vectors y . Since the last component of the vector in the rank-one modification is zero, to compute δ_{opt} we need to find the leading principal block of the inverse of $C(0) = B_d^T B_d$. It is given by the inverse of the Schur complement of $d^T d$ in $B_d^T B_d$ (which is positive definite), see relation (2.2), and thus

$$\delta_{opt} = \frac{1}{\vartheta^T \left(B^T B - \frac{1}{d^T d} B^T d d^T B \right)^{-1} \vartheta}.$$

We use the Sherman–Morrison formula (see Chapter 2, relation (2.3)) to obtain

$$\delta_{opt} = \frac{1}{\vartheta^T \left[(B^T B)^{-1} + (B^T B)^{-1} \frac{B^T d d^T B}{d^T d - d^T B (B^T B)^{-1} B^T d} (B^T B)^{-1} \right] \vartheta}.$$

Setting $\alpha = d^T d - d^T B (B^T B)^{-1} B^T d$ and $\omega = d^T B (B^T B)^{-1} \vartheta$, we obtain the relation (4.21). The value δ_{opt} is the minimum of (4.20) that we were looking for. We remark that the coefficient α is the Schur complement of $B^T B$ in the matrix $C(0)$ which is symmetric positive definite. Therefore, α is positive.

With $\delta = \delta_{opt}$, we have equality in (4.26) if y is an eigenvector corresponding to the zero eigenvalue. In that case, we have equality in (4.25) as well, i.e., $\|d - By\|^2 = \delta_{opt}(\vartheta^T y)^2$; hence, y_{opt} is the eigenvector of $C(\delta_{opt})$ corresponding to its zero eigenvalue.

From equating the leading k rows in $C(\delta_{opt}) \begin{pmatrix} y_{opt} \\ 1 \end{pmatrix} = 0$ we obtain

$$(B^T B - \delta_{opt} \vartheta \vartheta^T) y_{opt} = B^T d. \quad (4.27)$$

Using again the Sherman–Morrison formula to compute the inverse of the rank-one modification of $B^T B$, we have

$$\begin{aligned} y_{opt} &= (B^T B - \delta_{opt} \vartheta \vartheta^T)^{-1} B^T d, \\ &= (B^T B)^{-1} B^T d + (B^T B)^{-1} \frac{\delta_{opt} \vartheta \vartheta^T}{1 - \delta_{opt} \vartheta^T (B^T B)^{-1} \vartheta} (B^T B)^{-1} B^T d, \\ &= (B^T B)^{-1} B^T d + \frac{\delta_{opt} [\vartheta^T (B^T B)^{-1} B^T d]}{1 - \delta_{opt} \vartheta^T (B^T B)^{-1} \vartheta} (B^T B)^{-1} \vartheta. \end{aligned}$$

Now, $\vartheta^T(B^T B)^{-1}B^T d = d^T B(B^T B)^{-1}\vartheta = \omega$. Let $t = (B^T B)^{-1}\vartheta$. We have to consider the multiplying factor

$$\frac{\delta_{opt}\omega}{1 - \delta_{opt}\vartheta^T t}, \quad \text{with } \delta_{opt} = \frac{\alpha}{\alpha\vartheta^T t + \omega^2}.$$

After some algebra it yields

$$\frac{\delta_{opt}\omega}{1 - \delta_{opt}\vartheta^T t} = \frac{\alpha}{\omega},$$

and this proves the result (4.24) for y_{opt} .

Let us now consider the case $B^T d = 0$. Then, $\|d - By\|^2 = d^T d + y^T B^T B y$. Therefore, writing out (4.21), we have

$$y^T(B^T B - \delta_{opt}\vartheta\vartheta^T)y = -d^T d < 0.$$

The matrix within parenthesis is positive semi-definite, and there is no vector y that could satisfy this identity. Hence, there is no finite solution to the minimization problem (4.20). \square

Let us apply Proposition 4.1 to our situation of building a Q-OR optimal basis. When the optimum exists, to obtain α and ω in (4.22), resp. (4.23), it is necessary to solve two linear systems with the matrix $V_{n,k}^T V_{n,k}$,

$$(V_{n,k}^T V_{n,k})t = \vartheta, \quad (V_{n,k}^T V_{n,k})s = V_{n,k}^T A v_k, \quad \vartheta = (1 \ \vartheta_{1,2} \ \cdots \ \vartheta_{1,k})^T, \quad (4.28)$$

giving

$$\omega = (V_{n,k}^T A v_k)^T t, \quad \alpha = \|A v_k\|^2 - (V_{n,k}^T A v_k)^T s$$

and the optimal k leading entries of the k th column of H ,

$$h_{1:k,k} = s + \frac{\alpha}{\omega} t. \quad (4.29)$$

The vector s is the solution of the least squares problem $\min_y \|A v_k - V_{n,k} y\|$. The solution (4.24) of the problem in Proposition 4.1 is the solution of this least squares problem plus a correction depending on ϑ .

Unfortunately, a straightforward implementation of the algorithm is particularly inefficient since we compute far too many dot products. Let us prove some properties of the basis that will help us simplify the implementations of the algorithm.

We will assume that $\vartheta_{1,j} \neq 0$, $j = 1, \dots, k$, which is equivalent with the first k iterates being defined and the Hessenberg matrices being nonsingular. We also assume that the absolute values of the $\vartheta_{1,j}$'s are strictly increasing. The reason for this assumption will become clear later. With the values $h_{j,k}$ defined in (4.29), the vector \tilde{v}_k in (4.17) is

$$\tilde{v}_k = (I - V_{n,k}(V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T) A v_k - \frac{\alpha}{\omega} V_{n,k}(V_{n,k}^T V_{n,k})^{-1} \vartheta. \quad (4.30)$$

The first term of the right-hand side is in the orthogonal complement of the subspace $\mathcal{K}_k(A, b)$, and the second one is in $\mathcal{K}_k(A, b)$. We remark that if we would take $\alpha = 0$ for every iteration we will construct an orthogonal basis.

Looking at the angles between the basis vectors, from (4.30) we have

$$V_{n,k}^T v_{k+1} = \frac{1}{h_{k+1,k}} V_{n,k}^T \tilde{v}_k = -\frac{\alpha}{\omega h_{k+1,k}} \begin{pmatrix} 1 \\ \vartheta_{1,2} \\ \vdots \\ \vartheta_{1,k} \end{pmatrix}. \quad (4.31)$$

The next lemma shows how this expression can be simplified.

Lemma 4.3 *The basis vectors satisfy*

$$V_{n,k+1}^T v_{k+1} = \frac{1}{\vartheta_{1,k+1}} \begin{pmatrix} 1 \\ \vartheta_{1,2} \\ \vdots \\ \vartheta_{1,k} \\ \vartheta_{1,k+1} \end{pmatrix}. \quad (4.32)$$

Proof From relation (4.31) we have

$$V_{n,k+1}^T v_{k+1} = \frac{1}{h_{k+1,k}} \begin{pmatrix} V_{n,k}^T \\ v_{k+1}^T \end{pmatrix} \tilde{v}_k = \begin{pmatrix} -\frac{\alpha}{\omega h_{k+1,k}} \vartheta \\ 1 \end{pmatrix}.$$

We need to consider $\alpha/(\omega h_{k+1,k})$. We first remark that because of $\vartheta^T s = t^T V_{n,k}^T A v_k$, we have

$$\omega = \vartheta^T s. \quad (4.33)$$

Then, we compute $h_{k+1,k}^2 = \|\tilde{v}_k\|^2$ using (4.17),

$$h_{k+1,k}^2 = \|A v_k\|^2 - 2(A v_k, V_{n,k} h_{1:k,k}) + (V_{n,k}^T V_{n,k} h_{1:k,k}, h_{1:k,k}),$$

and, using (4.28),

$$V_{n,k} h_{1:k,k} = V_{n,k} \left(s + \frac{\alpha}{\omega} t \right), \quad V_{n,k}^T V_{n,k} h_{1:k,k} = V_{n,k}^T A v_k + \frac{\alpha}{\omega} \vartheta.$$

It yields

$$\begin{aligned} h_{k+1,k}^2 &= \|Av_k\|^2 \left(Av_k, V_{n,k}(s + \frac{\alpha}{\omega}t) \right) + \left(V_{n,k}^T Av_k + \frac{\alpha}{\omega}v, s + \frac{\alpha}{\omega}t \right) \\ &= \|Av_k\|^2 - (Av_k, V_{n,k}s) - \frac{\alpha}{\omega}[(Av_k, V_{n,k}t) - (\vartheta, s)] + \left(\frac{\alpha}{\omega}\right)^2(\vartheta, t). \end{aligned}$$

But

$$\|Av_k\|^2 - (Av_k, V_{n,k}s) = \alpha, \quad (Av_k, V_{n,k}t) - (\vartheta, s) = 0.$$

Therefore,

$$h_{k+1,k}^2 = \alpha + \left(\frac{\alpha}{\omega}\right)^2(\vartheta, t).$$

Introducing ω in the first term of the right-hand side of this relation and using (4.33), we obtain

$$h_{k+1,k}^2 = \frac{\alpha}{\omega} \left(\vartheta^T s + \frac{\alpha}{\omega} \vartheta^T t \right).$$

Therefore,

$$\frac{\alpha}{h_{k+1,k}\omega} = \frac{\alpha h_{k+1,k}}{h_{k+1,k}^2 \omega} = \frac{h_{k+1,k}}{\vartheta^T s + \frac{\alpha}{\omega} \vartheta^T t} = \frac{h_{k+1,k}}{\vartheta^T h_{1:k,k}} = -\frac{1}{\vartheta_{1,k+1}},$$

where the last equality uses (2.21). This proves the relation (4.32). \square

Note that if two consecutive absolute values are equal, say $|\vartheta_{1,k+1}| = |\vartheta_{1,k}|$, the cosine of the angle between v_k and v_{k+1} is 1 and, consequently, the two vectors are the same and this means that we do not have a basis any longer. This is why we assume that the absolute values of the $\vartheta_{1,j}$, $j = 1, \dots, k$ are strictly increasing.

From Lemma 4.3 the matrix $V_{n,k}^T V_{n,k}$ is

$$V_{n,k}^T V_{n,k} = \begin{pmatrix} 1 & \frac{1}{\vartheta_{1,2}} & \frac{1}{\vartheta_{1,3}} & \cdots & \frac{1}{\vartheta_{1,k}} \\ \frac{1}{\vartheta_{1,2}} & 1 & \frac{\vartheta_{1,2}}{\vartheta_{1,3}} & \cdots & \frac{\vartheta_{1,2}}{\vartheta_{1,k}} \\ \frac{1}{\vartheta_{1,3}} & \frac{\vartheta_{1,2}}{\vartheta_{1,3}} & 1 & \cdots & \frac{\vartheta_{1,3}}{\vartheta_{1,k}} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{1}{\vartheta_{1,k}} & \frac{\vartheta_{1,2}}{\vartheta_{1,k}} & \cdots & \cdots & 1 \end{pmatrix}. \quad (4.34)$$

It means that in this method, from the values of $\vartheta_{1,j}$ we know all the angles between the basis vectors. The next lemma shows that, mathematically, the vector t is zero except for the last component.

Lemma 4.4 *The solution of $V_{n,k}^T V_{n,k} t = \vartheta$ is*

$$t = (V_{n,k}^T V_{n,k})^{-1} \vartheta = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{1,k} \end{pmatrix}. \quad (4.35)$$

Proof Using Lemma 4.3 we have

$$\vartheta_{1,k} V_{n,k}^T v_k = \vartheta.$$

Therefore,

$$\vartheta = \vartheta_{1,k} V_{n,k}^T V_{n,k} e_k = V_{n,k}^T V_{n,k} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{1,k} \end{pmatrix},$$

proving the claim. \square

The next lemma proves that the basis is semi A -orthogonal (or conjugate), that is, $v_i^T A v_k = 0$, $i > k$.

Lemma 4.5 *The matrix $V_{n,k}^T A V_{n,k}$ is upper triangular.*

Proof From the Arnoldi-like relation (3.2) we have

$$A v_k = h_{k+1,k} v_{k+1} + V_{n,k} h_{1:k,k}.$$

It yields

$$v_i^T A v_k = \sum_{j=1}^{k+1} h_{j,k} v_i^T v_j.$$

From relation (4.34) the i th row of $V^T V$ is

$$\frac{1}{\vartheta_{1,i}} \dots \frac{\vartheta_{1,i-1}}{\vartheta_{1,i}} 1 \frac{\vartheta_{1,i}}{\vartheta_{1,i+1}} \dots \frac{\vartheta_{1,i}}{\vartheta_{1,n}}.$$

Then, with $i > k$ and using relation (2.21),

$$\begin{aligned} \sum_{j=1}^{k+1} h_{j,k} v_i^T v_j &= \sum_{j=1}^{k+1} h_{j,k} \frac{\vartheta_{1,j}}{\vartheta_{1,i}}, \\ &= h_{k+1,k} \frac{\vartheta_{1,k+1}}{\vartheta_{1,i}} + \sum_{j=1}^k h_{j,k} \frac{\vartheta_{1,j}}{\vartheta_{1,i}}, \\ &= \frac{1}{\vartheta_{1,i}} [h_{k+1,k} \vartheta_{1,k+1} - h_{k+1,k} \vartheta_{1,k+1}], \\ &= 0. \end{aligned}$$

\square

It is interesting to remark that the inverse of the matrix $V_{n,k}^T V_{n,k}$ in (4.34) (when it exists) has a particular structure.

Lemma 4.6 *Assuming that the $|\vartheta_{1,j}|$'s are strictly increasing, the inverse of the matrix $V_{n,k}^T V_{n,k}$ in (4.34) is tridiagonal.*

Proof This is readily seen from the structure of the matrix in relation (4.34); see Section 2.8 or [671]. But this can also be proved directly as follows. From relation (3.2) we have

$$V_{n,k}^T A V_{n,k} = V_{n,k}^T V_{n,k} H_k + h_{k+1,k} V_{n,k}^T v_{k+1} e_k^T.$$

Let $R_k = V_{n,k}^T A V_{n,k} - h_{k+1,k} V_{n,k}^T v_{k+1} e_k^T$. From Lemma 4.5 the matrix R_k is upper triangular. We have $(V_{n,k}^T V_{n,k})^{-1} = H_k R_k^{-1}$. This shows that $(V_{n,k}^T V_{n,k})^{-1}$ is upper Hessenberg. However, it is a symmetric matrix and, therefore, it is tridiagonal. \square

From Proposition 2.1 we can find exact expressions for the nonzero entries of the inverse of the matrix $V_{n,k}^T V_{n,k}$ in (4.34). Let α_j be the diagonal entries and β_j the subdiagonal entries. We obtain from Proposition 2.1 with $u_i = \vartheta_{1,i}$ and $v_i = 1/\vartheta_{1,i}$,

$$\begin{aligned} \alpha_1 &= \frac{\vartheta_{1,2}^2}{\vartheta_{1,2}^2 - 1}, \quad \beta_2 = -\frac{\vartheta_{1,2}}{\vartheta_{1,2}^2 - 1}, \\ \alpha_i &= \frac{\vartheta_{1,i-1}^2}{\vartheta_{1,i}^2 - \vartheta_{1,i-1}^2} + \frac{\vartheta_{1,i+1}^2}{\vartheta_{1,i+1}^2 - \vartheta_{1,i}^2}, \quad \beta_{i+1} = -\frac{\vartheta_{1,i}\vartheta_{1,i+1}}{\vartheta_{1,i+1}^2 - \vartheta_{1,i}^2}, \quad i = 2, \dots, k-1, \end{aligned}$$

and

$$\alpha_k = \frac{\vartheta_{1,k}^2}{\vartheta_{1,k}^2 - \vartheta_{1,k-1}^2}.$$

From this result we can obtain a lower bound on the singular values of $V_{n,k}$. This will tell us how well behaved is the basis. It depends only on the values of the $\vartheta_{1,j}$'s.

Theorem 4.4 *Assume $|\vartheta_{1,j+1}| \neq |\vartheta_{1,j}|$, $j = 1, \dots, n-1$. Let μ_i , $i = 1, \dots, k$ be defined as*

$$\mu_1 = \frac{|\vartheta_{1,2}|}{\vartheta_{1,2}^2 - 1}, \mu_i = \frac{|\vartheta_{1,i-1}| |\vartheta_{1,i}|}{\vartheta_{1,i}^2 - \vartheta_{1,i-1}^2} + \frac{|\vartheta_{1,i}| |\vartheta_{1,i+1}|}{\vartheta_{1,i+1}^2 - \vartheta_{1,i}^2}, \quad 1 < i < k, \mu_k = \frac{|\vartheta_{1,k-1}| |\vartheta_{1,k}|}{\vartheta_{1,k}^2 - \vartheta_{1,k-1}^2}.$$

The smallest singular value σ_k of $V_{n,k}$ is bounded from below:

$$\sigma_k \geq \frac{1}{(\max_i(\alpha_i + \mu_i))^{\frac{1}{2}}}.$$

Proof Using Gershgorin disks, the eigenvalues of the tridiagonal matrix $(V_{n,k}^T V_{n,k})^{-1}$ are located in the union of intervals $[\alpha_i - \mu_i, \alpha_i + \mu_i]$, $i = 1, \dots, k$ where $\mu_1 = |\beta_2|$, $\mu_i = |\beta_{i+1}| + |\beta_i|$, $i = 2, \dots, k-1$ and $\mu_k = |\beta_k|$. Hence, the smallest eigenvalue of $V_{n,k}^T V_{n,k}$ is bounded from below by $1/\max_i(\alpha_i + \mu_i)$. \square

The values in the denominator of the main result of Theorem 4.4 are

$$\alpha_1 + \mu_1 = \frac{1}{1 - \frac{1}{\vartheta_{1,2}^2}} \left(1 + \frac{1}{|\vartheta_{1,2}|} \right), \quad \alpha_k + \mu_k = \frac{1}{1 - \frac{\vartheta_{1,k-1}^2}{\vartheta_{1,k}^2}} \left(1 + \frac{|\vartheta_{1,k-1}|}{|\vartheta_{1,k}|} \right),$$

and

$$\alpha_i + \mu_i = \frac{1}{\frac{\vartheta_{1,i}^2}{\vartheta_{1,i-1}^2} - 1} \left(1 + \frac{|\vartheta_{1,i}|}{|\vartheta_{1,i-1}|} \right) + \frac{1}{1 - \frac{\vartheta_{1,i}^2}{\vartheta_{1,i+1}^2}} \left(1 + \frac{|\vartheta_{1,i}|}{|\vartheta_{1,i+1}|} \right), \quad i = 2, \dots, k-1.$$

If the values $|\vartheta_{1,i}|$ are strictly increasing all the terms are positive. But we see that if some consecutive values of $|\vartheta_{1,i}|$ are close, the maximum of $\alpha_i + \mu_i$ is large and at least the lower bound of the smallest singular value is small. On the contrary, if the sequence $|\vartheta_{1,i}|$ is increasing rapidly, the lower bound is large and the basis is well behaved.

Some of the mathematical properties of the basis that were proved can be used to simplify its computation. For instance, the computation of the new unnormalized vector \tilde{v}_k in (4.17) can be simplified since from relations (4.29) and (4.35),

$$V_{n,k} h_{1:k,k} = V_{n,k} s + \frac{\alpha}{\omega} \vartheta_{1,k} v_k.$$

But

$$\omega = t^T V_{n,k}^T A v_k = \vartheta_{1,k} v_k^T A v_k.$$

Assuming $\vartheta_{1,k} \neq 0$, it yields

$$\tilde{v}_k = A v_k - V_{n,k} s - \beta v_k, \quad \beta = \frac{\alpha}{v_k^T A v_k}.$$

Moreover, the k first entries of column k of H are given by

$$h_{1:k,k} = s + \beta e_k.$$

We see that we have a breakdown of the algorithm only if $v_k^T A v_k = 0$. It means that this method cannot be used to build bases for skew-symmetric matrices. However, if A is positive real, that is, if $(Av, v) > 0$ for all nonzero vectors v , it is not possible to have a breakdown.

Let us see how we can organize the computation in the algorithm at step k . We have to solve a linear system with the matrix $V_{n,k}^T V_{n,k}$. Even if we know its inverse is tridiagonal, we use for the moment its Cholesky decomposition and would like to compute the Cholesky factorization incrementally. Let us assume that we know the lower triangular matrix L_{k-1} such that $L_{k-1} L_{k-1}^T = V_{n,k-1}^T V_{n,k-1}$. Then, denote

$$L_k L_k^T = \begin{pmatrix} V_{n,k-1}^T V_{k-1} & z_k \\ z_k^T & 1 \end{pmatrix},$$

with $z_k = V_{n,k-1}^T v_k$ and

$$L_k = \begin{pmatrix} L_{k-1} & 0 \\ \ell_k^T & \ell_{k,k} \end{pmatrix}.$$

It is well known that by identification, we find that ℓ_k is obtained by solving $L_{k-1} \ell_k = z_k = V_{n,k-1}^T v_k$ and $\ell_{k,k} = \sqrt{1 - \ell_k^T \ell_k}$.

However, it seems more interesting to construct the inverses of the matrices L_k incrementally. Doing so, we can replace triangular solves by matrix–vector products. We have

$$\tilde{L}_k = L_k^{-1} = \begin{pmatrix} L_{k-1}^{-1} & 0 \\ -\frac{1}{\ell_{k,k}} \ell_k^T L_{k-1}^{-1} & \frac{1}{\ell_{k,k}} \end{pmatrix}.$$

The improved algorithm is the following.

```
function [V,H] = QORopt(A,u,nitmax);
%
% Initialization phase
n = size(A,1);
V = zeros(n,nitmax);
H = zeros(nitmax,nitmax);
Lt = zeros(nitmax,nitmax);
nu = zeros(nitmax,1);
%
V(:,1) = u / norm(u);
Av = A * V(:,1);
Lt(1,1) = 1;
omega = V(:,1)' * Av;
alpha = Av' * Av - omega^2;
H(1,1) = omega + alpha / omega;
vt = Av - H(1,1) * V(:,1);
H(2,1) = norm(vt);
V(:,2) = vt / H(2,1);
nu(1) = 1;
nu(2) = -H(1,1) / H(2,1);
% End of initialization
```

```

%
for j = 2:nitmax
    Av = A * V(:,j);
    vV = V(:,1:j-1)' * V(:,j);
    vtA = V(:,1:j)' * Av;
    lt = Lt(1:j-1,1:j-1) * vV;
    yt = lt' * Lt(1:j-1,1:j-1);
    if lt' * lt < 1
        ljj = sqrt(1 - lt' * lt);
    else
        % error
    end % if lt
    Lt(j,1:j) = [-yt / ljj, 1 / ljj];
    lA = Lt(1:j,1:j) * vtA;
    s = Lt(1:j,1:j)' * lA;
    alpha = Av' * Av - lA' * lA;
    beta = alpha / vtA(j);
    H(1:j-1,j) = s(1:j-1);
    H(j,j) = s(j) + beta;
    if j < n
        vt = Av - V(:,1:j) * H(1:j,j);
        H(j+1,j) = norm(vt);
        nu(j+1) = -(nu(1:j)' * H(1:j,j)) / H(j+1,j);
        V(:,j+1) = vt / H(j+1,j);
    end % if j
end % for j

```

In this code the vector ν contains the entries $\theta_{1,j}$. To have a practical algorithm one must test if $vtA(j)$ is zero or too small before the computation of **beta**. The previous implementation is more costly than Arnoldi MGS since we have to compute almost twice the number of dot products of Arnoldi. Fortunately, these dot products are all independent and occur as dense matrix–vector products.

The algorithm could still be simplified a little more if we use the fact that the inverse of $V_{n,k}^T V_{n,k}$ is tridiagonal. Using the multiplication by a tridiagonal matrix to compute the vector s removes $k - 1$ dot products per iteration. However, we will see later that this simplified version is less stable than the previous one.

```

function [V,H] = QORopt_T(A,u,nitmax);
%
% Initialization phase
n = size(A,1);
V = zeros(n,nitmax);
H = zeros(nitmax,nitmax);
nu = zeros(nitmax,1);
alpk = zeros(nitmax,1);
betk = zeros(nitmax,1);

```

```

%
V(:,1) = u / norm(u);
Av = A * V(:,1);
omega = V(:,1)' * Av;
alpha = Av' * Av - omega^2;
H(1,1) = omega + alpha / omega;
vt = Av - H(1,1) * V(:,1);
H(2,1) = norm(vt);
V(:,2) = vt / H(2,1);
nu(1) = 1;
nu(2) = -H(1,1) / H(2,1);
nu2 = nu(2)^2;
% entries of the tridiagonal matrix (inverse of V^T V)
alpk(1) = nu2 / (nu2 - 1);
alpk(2) = alpk(1);
betk(1) = - nu(2) / (nu2 - 1);
% End of initialization
%
for j = 2:nitmax
    Av = A * V(:,j);
    vtA = V(:,1:j)' * Av;
    % product with a tridiagonal matrix
    s = alpk(1:j) .* vtA;
    s(1:j-1) = s(1:j-1) + betk(1:j-1) .* vtA(2:j);
    s(2:j) = s(2:j) + betk(1:j-1) .* vtA(1:j-1);
    AVks = Av - V(:,1:j) * s;
    alpha = AVks' * AVks;
    beta = alpha / vtA(j);
    H(1:j-1,j) = s(1:j-1);
    H(j,j) = s(j) + beta;
    if j < n
        vt = AVks - beta * V(:,j);
        H(j+1,j) = norm(vt);
        nu(j+1) = -(nu(1:j)' * H(1:j,j)) / H(j+1,j);
        V(:,j+1) = vt / H(j+1,j);
        nuj1 = nu(j-1)^2;
        nuj = nu(j)^2;
        nujp = nu(j+1)^2;
        % entries of the tridiagonal matrix (inverse of V^T V)
        alpk(j) = nuj1 / (nuj - nuj1) + nujp / (nujp - nuj);
        betk(j) = -(nu(j) * nu(j+1)) / (nujp - nuj);
        alpk(j+1) = nujp / (nujp - nuj);
    end % if j
end % for j

```

4.6 Newton and Chebyshev bases

We have seen that the natural (monomial) basis for the Krylov subspace $\mathcal{K}_k(A, r_0)$ can be badly conditioned and, quite often, one uses an orthonormal basis constructed using Arnoldi MGS. However, it is not easy to parallelize the computation of this basis due to many global communications that are required for the dot products. Hence, starting in the 1990s, some researchers have considered other ways to compute orthonormal bases that are more amenable to parallel computing; see [58, 59, 173, 321, 756, 824, 962]. These issues arise also in the so-called communication-avoiding Krylov methods; see [188, 527].

Basis vectors are constructed as follows:

$$v_1 = r_0 / \|r_0\|, \quad v_{i+1} = \frac{1}{\eta_i}(A - \xi_i I)v_i, \quad i = 1, \dots, k-1, \quad (4.36)$$

where η_i are real normalizing factors and the given $k-1$ points ξ_i , $i = 1, \dots, k-1$ in the complex plane are referred to as *shifts*. If

$$V_{n,k} = (v_1 \ v_2 \ \dots \ v_k)$$

is of rank k , we have a basis of the Krylov subspace. Such a basis is called a *Newton basis* because the polynomial giving the basis vectors is in Newton form. The hope is that we can choose the shifts in such a way that the basis with the vectors v_j is better conditioned than the monomial basis. Then, when we have computed the vectors v_1, \dots, v_k using the relation (4.36), they can be orthonormalized to obtain an even better conditioned basis. This can be done with parallel variants of the QR factorization; see [116, 165, 265, 321, 527, 528]. This is why this technique is more suited to parallel computing than the Arnoldi process.

If the data is real and if we use complex shifts the recurrence (4.36) implies that we have to compute with complex numbers. However, we can generate a basis using complex conjugate shifts relying only on real arithmetic. This can be done in the following way. If the shift is real we use (4.36). Otherwise, assume that we would like to apply two shifts ξ and $\bar{\xi}$ successively with normalizing factors η_1 and η_2 to a given vector v . The result must be

$$\frac{1}{\eta_1 \eta_2} (A - \bar{\xi}I)(A - \xi I)v = \frac{1}{\eta_1 \eta_2} [A^2 - 2\operatorname{Re}(\xi)A + |\xi|^2 I]v.$$

This can be done by first adding the vector

$$\tilde{v} = \frac{1}{\eta_1} [Av - \operatorname{Re}(\xi)v],$$

to the basis. Then, we add the vector

$$\frac{1}{\eta_2} \left[A\tilde{v} - \operatorname{Re}(\xi)\tilde{v} + \frac{1}{\eta_1} (\operatorname{Im}(\xi))^2 v \right].$$

Re and Im are the real and imaginary parts of a complex number. One can check that after these two steps we obtain the correct result.

Computing an orthonormal basis is done in three steps. The first step is the choice or the computation of the shifts ξ_i . The second one computes the vectors v_j , $j = 1, \dots, k$, and the third step orthonormalizes these already computed vectors.

The choices of the shifts that have been considered in the literature are often based on heuristics. It is almost obvious that successive shifts have to be different enough from each other. If they are too close, we will get back to the troubles we mentioned for the monomial basis. In [59] the shifts were determined by first running m iterations of the Arnoldi process, computing the distinct eigenvalues of the upper Hessenberg matrix H_m (which are known as Ritz values), ordering them appropriately and taking these complex numbers as shifts. A modified Leja ordering is used in [59]. The modification allows to order consecutively an eigenvalue and its conjugate transpose. The choices in [59] are supported by bounds for the condition number of the Newton basis in particular cases, that is, when A is a normal matrix. These results were obtained by relating the condition number of the basis to problems in polynomial interpolation; see also [763]. Note that with this choice of shifts we can compute at most m basis vectors. A similar algorithm was used in [321, 824]. When the number of basis vectors that are needed is larger than m , the shifts can be applied in a periodic fashion, but this can lead to a poor conditioning of the basis; see [756].

There are other possibilities for the choice of the shifts. In [756] the so-called *spoke sets* are introduced. When we have the m Ritz values as above, we compute their barycenter. The segments joining the Ritz values to the barycenter are the spokes. Then, we compute fast Leja points (see [51]) on each of the spokes. With this algorithm we can compute as many shifts as we need. The number m of initial Arnoldi iterations has just an influence on the goodness of the eigenvalue approximations and on the number of spokes. In [756] these sets are updated as the computation of basis vectors proceeds, but this may not be necessary if the initial approximation of the spectrum of A is good enough. Numerical results are presented in [756]. We observe that we can eventually use other set of points on each of the spoke sets. The points have only to be distributed in such a way that we do not have two points which are too close since the shifts must be different enough to maintain linear independence of the basis vectors.

Another possibility is to consider a region enclosing the Ritz values and to compute appropriate points in this region. For instance, after m initial steps of the Arnoldi process, the Ritz values are computed and a minimal area ellipse containing these points in the complex plane is computed. In fact, if the data is real the axes of the ellipse are aligned with the axes of the complex plane and we can just consider a half ellipse. A set X of $N \geq m$ points are computed in the half ellipse to obtain a small Lebesgue constant; see [687] for points on the unit disk. The N Lagrange polynomials ℓ_j related to the points in X are such that the polynomial ℓ_j is equal to

1 at point ξ_j and zero at the other points. The Lebesgue function is defined as

$$\Lambda_X(\xi) = \sum_{j=1}^N |\ell_j(\xi)|,$$

The Lebesgue constant ℓ_X is the maximum of the Lebesgue function over $\xi \in \Omega$ where Ω is our given (half) ellipse. The goal is to find sets of points ξ_j yielding small Lebesgue constants because this is good for interpolation of functions since the Lebesgue constant tells us how far a polynomial interpolant is from the optimal polynomial in the maximum norm. It also guarantees that the points are not too close to each other which is what we are looking for since, otherwise, the Lebesgue function would be large in the neighborhood of those points. Minimizing the Lebesgue constant is a difficult optimization problem since the Lebesgue function is not differentiable and very oscillatory when the number of points N is large. However, this problem can be solved offline. We can compute interpolation points for a given half ellipse or half disk and then map those points to the half ellipse obtained from the Ritz values.

Rather than using a Newton basis, we can construct a Chebyshev basis. Of course, this is a different technique but it uses the same first and third phases than the construction of the Newton basis. Let us assume that, as above, we have an ellipse of center d (with d real) whose foci are $d - c$ and $d + c$ enclosing the Ritz values. Let C_j be the Chebyshev polynomial of degree j , and let p_j be the shifted and scaled polynomial defined by

$$p_j(\lambda) = C_j\left(\frac{d-\lambda}{c}\right) / C_j\left(\frac{d}{c}\right).$$

Using the recurrence relation of the Chebyshev polynomials, we have

$$\begin{aligned} p_{j+1}(\lambda) &= \frac{1}{C_{j+1}\left(\frac{d}{c}\right)} \left[2\left(\frac{d-\lambda}{c}\right) C_j\left(\frac{d-\lambda}{c}\right) - C_{j-1}\left(\frac{d-\lambda}{c}\right) \right], \\ &= \frac{1}{C_{j+1}\left(\frac{d}{c}\right)} \left[2\left(\frac{d-\lambda}{c}\right) C_j\left(\frac{d}{c}\right) p_j(\lambda) - C_{j-1}\left(\frac{d}{c}\right) p_{j-1}(\lambda) \right]. \end{aligned}$$

Let $\alpha_j = C_j(d/c)$. This coefficient satisfies the 3-term recurrence,

$$\alpha_{j+1} = 2\left(\frac{d}{c}\right) \alpha_j - \alpha_{j-1}, \quad \alpha_1 = \frac{d}{c}, \quad \alpha_0 = 1.$$

Let us define

$$\chi_j = \frac{1}{2\frac{d}{c} - \frac{\alpha_{j-1}}{\alpha_j}}, \quad \xi_j = \frac{1}{2\frac{d}{c} \frac{\alpha_j}{\alpha_{j-1}} - 1}.$$

Then,

$$p_{j+1}(\lambda) = 2 \left(\frac{d - \lambda}{c} \right) \chi_j p_j(\lambda) - \xi_j p_{j-1}(\lambda).$$

We define the basis vectors of unit norm as

$$v_j = \frac{p_j(A)v}{\|p_j(A)v\|}.$$

Let

$$\begin{aligned} \tilde{v}_{j+1} &= 2 \left(\frac{d}{c} \right) \chi_j \|\tilde{v}_j\| v_j - 2 \left(\frac{d}{c} \right) \chi_j \|\tilde{v}_j\| A v_j - \xi_j \|\tilde{v}_{j-1}\| v_{j-1}, \\ \tilde{v}_2 &= v_1 - \frac{1}{d} A v_1. \end{aligned}$$

Then, $v_{j+1} = \tilde{v}_{j+1}/\|\tilde{v}_{j+1}\|$. These relations can be written in matrix form as $AV_{n,k} = V_{n,k+1}\underline{T}_k$ where the $(k+1) \times k$ matrix \underline{T}_k is tridiagonal. The three nonzero entries in column j of this matrix are

$$\begin{pmatrix} -\frac{c\xi_j}{2\chi_j} \frac{\|\tilde{v}_{j-1}\|}{\|\tilde{v}_j\|} \\ d \\ -\frac{c}{2\chi_j} \frac{\|\tilde{v}_{j-1}\|}{\|\tilde{v}_j\|} \end{pmatrix}.$$

Other types of enclosing regions and polynomials have been considered, mainly to develop hybrid methods; see Section 12.14 of Chapter 12. In [313] the union of convex hulls of subsets of the Ritz values was considered as well as an L_∞ optimal approximation polynomial. The authors of [858] enclosed the Ritz values in a polygon and used a least squares polynomial. This was also used in [793] but the least squares polynomial was computed in a different and more stable way using modified moments. All these methods can be used to compute a basis of the Krylov subspace.

When we have computed a basis, either with the Newton polynomial or with any of the other methods we have described in this section, we could use it straightforwardly. But it may be that the basis is not so well conditioned. A possibility is to orthogonalize the set of vectors we have computed. This could be done using different algorithms based on the QR factorization. Some of the papers on this topic use parallel versions of the QR factorization of a set of given vectors. This includes RODDEC in [321], tiled QR factorization algorithms in [116, 165] and the tall and skinny QR (TSQR) factorization in [265, 527, 528].

4.7 Truncated bases

The Arnoldi process constructs an orthonormal basis of the Krylov subspaces $\mathcal{K}_k(A, r_0)$. We have seen that to compute the next basis vector v_{j+1} we have to orthogonalize Av_j against all the previous basis vectors v_i , $i = 1, \dots, j$. This can lead to storage problems when the order of the matrix is large. Moreover, the cost of the orthogonalization is increasing with the index j . In [789] it was proposed to truncate the recurrence for the orthogonalization. We orthogonalize Av_j only against the q previous basis vectors. Therefore, the computation of the next basis vector is done as

$$\tilde{v} = Av_j - \sum_{i_0}^j h_{i,j} v_i, \quad h_{i,j} = (Av_j, v_i), \quad v_{j+1} = \tilde{v}/\|\tilde{v}\|, \quad (4.37)$$

the start of the recurrence is $i_0 = \max(1, j - q + 1)$ and $h_{j+1,j} = \|\tilde{v}\|$. Doing this, the cost of the computation of the next basis vector is fixed (except at the beginning of the iterations) and the matrices H_k are banded upper Hessenberg matrices. Of course, the basis is only locally orthogonal, $V_{n,k}^T V_{n,k} \neq I$ and $H_k \neq V_{n,k}^T A V_{n,k}$. The basis vectors are such that $(v_i, v_j) = \delta_{i,j}$, $|i - j| \leq q$. From [788] we have the following proposition.

Proposition 4.2 *Assume the columns of $V_{n,k}$ to be linearly independent. Let $\hat{H}_k = (V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T A V_{n,k}$. Then,*

$$\hat{H}_k = H_k + t_k e_k^T, \quad (4.38)$$

with $t_k = h_{k+1,k} (V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T v_{k+1}$.

Proof The matrix H_k satisfies an Arnoldi-like relation. Since $V_{n,k}^T V_{n,k}$ is symmetric positive definite, multiplying from the left by $(V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T$ we obtain

$$\hat{H}_k = (V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T A V_{n,k} = H_k + h_{k+1,k} (V_{n,k}^T V_{n,k})^{-1} V_{n,k}^T v_{k+1} e_k^T,$$

which yields the value of t_k . \square

We see that H_k and \hat{H}_k differ only by the last column and therefore, \hat{H}_k is also upper Hessenberg. The matrix \hat{H}_k arises when requiring $r_0 - AV_{n,k}y$ to be orthogonal to $\mathcal{K}_k(A, r_0)$: This gives the equation

$$V_{n,k}^T A V_{n,k} y = V_{n,k}^T r_0 = \|r_0\| V_{n,k}^T V_{n,k} e_1,$$

which is equivalent to the equation $\hat{H}_k y = \|r_0\| e_1$; see [789]. Although the “orthogonality” condition for the residual used here is that of a Q-OR method, we remark that solving $\hat{H}_k y = \|r_0\| e_1$ is different from what we would have using the same basis in a Q-OR method: In a Q-OR method, we would solve $H_k y = \|r_0\| e_1$, see Section 3.1.

The basis obtained by truncation of the Arnoldi process is studied in [565] where the following results are proved.

Theorem 4.5 *Assume $q \geq 2$. For all $|i - j| \geq q + 1$, $i, j \leq k$, we have*

$$|(v_i, v_j)| \leq c_i \|A - A^T\| h_{i,i-1}^{-1},$$

where c_i is a value depending on the entries of H_k . Moreover,

$$\max \{0, 1 - (k - q - 1) c \|A - A^T\|\} \leq \sigma(V_{n,k}^T V_{n,k}) \leq 1 + (k - q - 1) c \|A - A^T\|,$$

for all singular values of $V_{n,k}^T V_{n,k}$ and c depends on the entries of H_k .

The conclusion in [565] is that $V_{n,k}$ might completely deviate from orthogonality even if $q = k - 2$ when A is nonsymmetric (i.e., $\|A - A^T\| \gg 1$). For very nonsymmetric matrices the columns of $V_{n,k}$ may become almost linearly dependent.

Let us consider now a slightly different way of constructing a basis using truncation. We will not only consider vectors from $V_{n,k}$ but also allow some vectors from $AV_{n,k}$ (other than Av_k) to compute the next basis vector.

Let p and q be two positive integers. We denote by $V_{k-q+1:k}$ the matrix whose columns are v_{k-q+1}, \dots, v_k , that is, the last q columns of $V_{n,k}$ as above. We consider that if $q = 0$ the matrix $V_{k+1:k}$ is the empty matrix, that is, the corresponding term will not exist. Similarly we define $AV_{k-p+1:k}$. To generate a basis of the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$ from the basis of $\mathcal{K}_k(A, r_0)$ we need at least one multiplication with A . Hence, we have to take $p \geq 1$ because we need to use Av_k . Then, the new (unnormalized) basis vector is defined as $\tilde{v} = B^{(k)}y$ with

$$B^{(k)} = (AV_{\max(1,k-p+1):k} \ V_{\max(1,k-q+1):k}).$$

Hence, we take the last p vectors in $AV_{n,k}$ and the last q vectors in $V_{n,k}$, assuming that $k \geq p + q$. If $p = 1$ and $q = 0$, we will generate the natural basis of the Krylov subspace which is known to be badly conditioned. We must choose $q > 0$. At the beginning of the iterations, if $k < p + q$, we first reduce the value of q and, if necessary, the value of p to obtain the appropriate number of vectors. If $p = 1$ and $q \geq k$ we use only Av_k and all the previous vectors v_j recovering the Arnoldi process. The vector of coefficients y could be determined by imposing orthogonality of the residual to the k th Krylov subspace, but we will define it differently. This is explained later in this section.

Let us see what kind of matrix relation we obtain with this definition of $B^{(k)}$. Let η_j be the norm of \tilde{v} (this will be denoted as $h_{j+1,j}$ later on). Then

$$\eta_j v_{j+1} = y_1 Av_{j-p+1} + \cdots + y_p Av_j + y_{p+1} v_{j-q+1} + \cdots + y_{p+q} v_j.$$

We have seen that we need to have $y_p \neq 0$. Dividing by y_p , it yields

$$Av_j = \frac{\eta_j}{y_p} v_{j+1} - \frac{y_{p+1}}{y_p} v_{j-q+1} - \cdots - \frac{y_{p+q}}{y_p} v_j - \frac{y_1}{y_p} Av_{j-p+1} - \cdots - \frac{y_{p-1}}{y_p} Av_{j-1}.$$

For the indices which are smaller than 1, the coefficients are assumed to be zero. In this relation y_p is somehow arbitrary and we will take $y_p = 1$. Using this relation for $j = 1, \dots, k$ we obtain in matrix form

$$AV_{n,k} = V_{n,k} S_k - AV_{n,k} R_k + \eta_k v_{k+1} e_k^T,$$

where S_k is a banded upper Hessenberg matrix and R_k is a banded strictly upper triangular matrix of order k . A generic column j of S_j has q nonzero entries coming from the coefficients of the vectors v_i . A generic column j of R_k has $p - 1$ nonzero entries coming from the coefficients of the vectors Av_i and 0 on the diagonal. Then,

$$AV_{n,k} [I + R_k] = V_{n,k} S_k + \eta_k v_{k+1} e_k^T.$$

The matrix $I + R_k$, as well as its inverse, is upper triangular with ones on the diagonal. It yields

$$AV_{n,k} = V_{n,k} S_k [I + R_k]^{-1} + \eta_k v_{k+1} e_k^T [I + R_k]^{-1}.$$

But $e_k^T [I + R_k]^{-1} = e_k^T$ and we finally obtain

$$AV_{n,k} = V_{n,k} H_k + \eta_k v_{k+1} e_k^T,$$

with

$$H_k = S_k [I + R_k]^{-1}, \quad (4.39)$$

an upper Hessenberg matrix. This can also be written as

$$AV_{n,k} = V_{n,k+1} \underline{H}_k,$$

with $\underline{H}_k = \underline{S}_k [I + R_k]^{-1}$ where \underline{S}_k is a banded $(k + 1) \times k$ upper Hessenberg matrix. Hence, we have an Arnoldi-like relation where the matrix H_k is in factored form.

Let us assume that we have computed the ascending basis for the Krylov subspaces with the subsequent basis vectors stored in the columns of V . Then, there exists an upper triangular matrix U such that $K = VU$ where K is the Krylov matrix. From Section 2.7 we know that the matrix H_k can be written as

$$H_k = U_k C^{(k)} U_k^{-1}, \quad (4.40)$$

U_k being upper triangular and the leading principal submatrix of order k of U and $C^{(k)} = E_k + (0 \ U_k^{-1} U_{1:k,k+1})$ where E_k is a square down-shift matrix of order k ,

$$E_k = \begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ \ddots & \ddots & & \\ & & 1 & 0 \end{pmatrix}.$$

Note that $C^{(k)}$ is a companion matrix. Moreover, the subdiagonal entries of H are $h_{j+1,j} = u_{j+1,j+1}/u_{j,j}$, $j = 1, \dots, n-1$. Let us consider a relation that will help us for choosing the vector y later on.

Proposition 4.3 *Let $\vartheta_{i,j}$ be the entries of U^{-1} . We have*

$$\vartheta_{1,k+1} = -\frac{1}{s_{k+1,k}} \sum_{j=1}^k \vartheta_{1,j} s_{j,k}.$$

Proof Using the two relations (4.39) and (4.40) we have for H_j , it yields

$$H_j = S_j(I + R_j)^{-1} = U_j E_j U_j^{-1} + (0 \ U_{1:j,j+1}) U_j^{-1}.$$

Then, multiplying from the left with $e_1^T U_j^{-1}$ we obtain

$$e_1^T U_j^{-1} S_j(I + R_j)^{-1} = e_1^T E_j U_j^{-1} + e_1^T U_j^{-1} (0 \ U_{1:j,j+1}) U_j^{-1}.$$

The first term on the right-hand side is zero. The other term is a row vector with only the last component being nonzero. Writing this for $j = n-1$, we have

$$e_1^T U_{n-1}^{-1} S_{n-1} = e_1^T U_{n-1}^{-1} (0 \ U_{1:n-1,n}) U_{n-1}^{-1} (I + R_{n-1}),$$

and considering the first $k+1$ columns for an index $k+1 < n-1$,

$$(1 \ \vartheta_{1,2} \ \cdots \ \vartheta_{1,k+1}) S_{k+1} = (0 \ \cdots \ 0).$$

When we use the k th column of S_{k+1} , we obtain

$$(1 \ \vartheta_{1,2} \ \cdots \ \vartheta_{1,k+1}) \begin{pmatrix} s_{1,k} \\ \vdots \\ s_{k+1,k} \end{pmatrix} = 0.$$

Computing $\vartheta_{1,k+1}$ from this relation proves the result. \square

We note that in the sum over j some entries $s_{j,k}$ are zero since S_{k+1} is a banded matrix. In a generic column only the last q terms are nonzero.

As in Section 4.5, we will attempt to define the new basis vector through the choice of the k th column of H_k such that $|\vartheta_{1,k+1}|$ is as large as possible with the

constraint that the new basis vector has to be of unit norm. It is easy to see that $h_{k+1,k} = s_{k+1,k} = \eta_k$. The relation defining v_{k+1} can be written as

$$\eta_k v_{k+1} = Av_k - B^{(k)}z,$$

where z is the vector with components $-y_1, \dots, -y_{p-1}, -y_{p+1}, \dots, -y_{p+q}$ and

$$B^{(k)} = (Av_{k-p+1} \cdots Av_{k-1} v_{k-q+1} \cdots v_k).$$

The coefficients of v_{k-q+1}, \dots, v_k define the column of S_k , and the coefficients of $Av_{k-p+1}, \dots, Av_{k-1}$ define minus the column of R_k . Define the size $p+q$ vector

$$\tilde{\vartheta} = (0 \cdots 0 \vartheta_{1,k-q+1} \cdots \vartheta_{1,k})^T.$$

Then, because S_k is banded,

$$\sum_{j=1}^k \vartheta_{1,j} s_{j,k} = \tilde{\vartheta}^T z.$$

Using Proposition 4.3 we have $\vartheta_{k+1} = (\tilde{\vartheta}^T z)/\eta_k$ and, in analogy with what we did in Section 4.5, the maximization of $|\vartheta_{k+1}|$ corresponds to solving for a vector z the minimization problem

$$\min_{\tilde{\vartheta}^T z \neq 0} \frac{\|Av_k - B^{(k)}z\|^2}{(\tilde{\vartheta}^T z)^2}.$$

The solution of the minimization problem which is similar to (4.20) was given in Proposition 4.1. It is written as

$$z_{opt} = ([B^{(k)}]^T B^{(k)})^{-1} [B^{(k)}]^T Av_k + \frac{\alpha}{\omega} ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta}, \quad (4.41)$$

with

$$\begin{aligned} \alpha &= (Av_k)^T Av_k - (Av_k)^T B^{(k)} ([B^{(k)}]^T B^{(k)})^{-1} [B^{(k)}]^T Av_k, \\ \omega &= (Av_k)^T B^{(k)} ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta}. \end{aligned}$$

When we have computed z_{opt} (if this is possible) we have

$$\tilde{v} = Av_k - B^{(k)}z_{opt}, \quad v_{k+1} = \tilde{v}/\|\tilde{v}\|.$$

The entries of S and R are determined by

$$\begin{pmatrix} s_{1,k} \\ \vdots \\ 0 \\ z_p \\ \vdots \\ z_{p+q-1} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_p \\ \vdots \\ z_{p+q-1} \end{pmatrix}, \quad s_{k+1,k} = \|\tilde{v}\|, \quad \begin{pmatrix} r_{1,k} \\ \vdots \\ 0 \\ z_1 \\ \vdots \\ z_{p-1} \end{pmatrix} = -\begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_1 \\ \vdots \\ z_{p-1} \end{pmatrix},$$

and $h_{k+1,k} = s_{k+1,k} = \|\tilde{v}\|$. The entries of the k th column of H are obtained from those of S and R , see (4.39).

As in Section 4.5, we can prove some mathematical properties of the basis vectors which allow to simplify the algorithm. Let $s = ([B^{(k)}]^T B^{(k)})^{-1} [B^{(k)}]^T A v_k$ (not to be confused with the $s_{i,j}$'s) and $t = ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta}$. Then, $z_{opt} = s + (\alpha/\omega)t$. We remark that, as in (4.30),

$$\tilde{v} = [I - B^{(k)}([B^{(k)}]^T B^{(k)})^{-1} [B^{(k)}]^T] A v_k - \frac{\alpha}{\omega} B^{(k)}([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta}. \quad (4.42)$$

Therefore, as in (4.31),

$$[B^{(k)}]^T v_{k+1} = -\frac{\alpha}{\omega h_{k+1,k}} \tilde{\vartheta}. \quad (4.43)$$

We have, as in (4.33),

$$\omega = (A v_k)^T B^{(k)} ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta} = s^T \tilde{\vartheta}.$$

Theorem 4.6 *The vector t is given by*

$$t = ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{1,k} \end{pmatrix}. \quad (4.44)$$

Proof Using the same techniques as in Lemma 4.3 in Section 4.5 we can prove that

$$h_{k+1,k}^2 = \frac{\alpha}{\omega} \left(\tilde{\vartheta}^T s + \frac{\alpha}{\omega} \tilde{\vartheta}^T t \right).$$

It implies that

$$\frac{\alpha}{\omega h_{k+1,k}} = -\frac{1}{\vartheta_{1,k+1}}. \quad (4.45)$$

Hence with (4.43),

$$[B^{(k)}]^T v_{k+1} = \begin{pmatrix} (Av_{k-p+1})^T \\ \vdots \\ (Av_{k-1})^T \\ v_{k-q+1}^T \\ \vdots \\ v_k^T \end{pmatrix} v_{k+1} = \frac{1}{\vartheta_{1,k+1}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{1,k-q+1} \\ \vdots \\ \vartheta_{1,k} \end{pmatrix}. \quad (4.46)$$

It yields

$$\begin{aligned} (Av_j)^T v_{k+1} &= 0, \quad j = k-p+1, \dots, k-1, \\ v_j^T v_{k+1} &= \frac{\vartheta_{1,j}}{\vartheta_{1,k+1}}, \quad j = k-q+1, \dots, k+1. \end{aligned}$$

Writing the previous relation for $v_k = B^{(k)}e_k$ instead of v_{k+1} , we obtain

$$\vartheta_{1,k} [v_{k-q+1}, \dots, v_k]^T B^{(k)} e_k = \begin{pmatrix} \vartheta_{k-q+1} \\ \vdots \\ \vartheta_{1,k} \end{pmatrix}.$$

Moreover, we have from (4.42)

$$\begin{aligned} (Av_k, \tilde{v}) &= (Av_k, Av_k - B^{(k)}s - \frac{\alpha}{\omega} B^{(k)}([B^{(k)}]^T B^{(k)})^{-1} \tilde{v}), \\ &= \|Av_k\|^2 - (Av_k, B^{(k)}s) - \alpha, \\ &= 0. \end{aligned}$$

Using the orthogonality relations in (4.46) between v_k and the Av_j 's and using $(Av_{k-1})^T v_k = 0$, it yields

$$\vartheta_{1,k} [Av_{k-p+1}, \dots, Av_{k-1}, v_{k-q+1}, \dots, v_k]^T B^{(k)} e_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{k-q+1} \\ \vdots \\ \vartheta_{1,k} \end{pmatrix}.$$

This is nothing other than

$$\vartheta_{1,k} [B^{(k)}]^T B^{(k)} e_k = \tilde{\vartheta},$$

which gives

$$t = ([B^{(k)}]^T B^{(k)})^{-1} \tilde{\vartheta} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vartheta_{1,k} \end{pmatrix},$$

and this concludes the proof. \square

Using Theorem 4.6 we also have

$$\omega = t^T [B^{(k)}]^T A v_k = \vartheta_{1,k} v_k^T A v_k. \quad (4.47)$$

Therefore, we can simplify the computation of v_{k+1} since

$$\tilde{v} = A v_k - B^{(k)} z_{opt} = A v_k - B^{(k)} s - \frac{\alpha}{\omega} \vartheta_{1,k} B^{(k)} e_k = A v_k - B^{(k)} s - \frac{\alpha}{v_k^T A v_k} v_k.$$

Moreover, we have, using the definition of s , that

$$s^T [B^{(k)}]^T A v_k = s^T [B^{(k)}]^T B^{(k)} s = \|B^{(k)} s\|^2 \Rightarrow (B^{(k)} s)^T [A v_k - B^{(k)} s] = 0.$$

Hence,

$$\alpha = (A v_k)^T [A v_k - B^{(k)} s] = \|A v_k - B^{(k)} s\|^2.$$

Let us now consider a property of the $\vartheta_{1,j}$'s.

Theorem 4.7 *Let us assume that $\alpha > 0$. Then,*

$$\frac{|\vartheta_{1,k}|}{|\vartheta_{1,k+1}|} < 1.$$

Proof We have from the proof of Theorem 4.6 and (4.45)

$$h_{k+1,k}^2 = \frac{\alpha}{\omega} \left(\tilde{\vartheta}^T s + \frac{\alpha}{\omega} \vartheta_{1,k}^2 \right) = \alpha + \left(\frac{\alpha}{\omega} \right)^2 \vartheta_{1,k}^2 = \alpha + \beta^2,$$

with $\beta = \alpha / v_k^T A v_k$. We have seen in the proof of Theorem 4.6 that $v_k^T v_{k+1} = \vartheta_{1,k} / \vartheta_{1,k+1}$. On the other hand, we have with (4.46) and (4.47) that

$$v_k^T v_{k+1} = -\frac{\alpha \vartheta_{1,k}}{\omega h_{k+1,k}} = -\frac{\beta}{h_{k+1,k}} = -\frac{\beta}{\sqrt{\alpha + \beta^2}}.$$

Therefore,

$$\frac{|\vartheta_{1,k}|}{|\vartheta_{1,k+1}|} = \frac{|\beta|}{\sqrt{\alpha + \beta^2}} < 1,$$

which proves the claim. \square

The main difference of the truncated algorithm with the full algorithm in Section 4.5 is that the matrix $B^{(k)}$ we have to consider changes at every iteration since we remove some columns and add new ones (i.e., the first k columns of $B^{(k+1)}$ are not the k columns of $B^{(k)}$). Hence, we cannot use an incremental Cholesky factorization of the matrix $[B^{(k)}]^T B^{(k)}$ or of its inverse as it was done before. However, except in the first iterations, the order of $B^{(k)}$ is fixed as $n \times (p + q - 1)$ whence it was $n \times k$ in the full algorithm. To compute the solution when it exists we have to solve a linear system with the matrix $[B^{(k)}]^T B^{(k)}$. In fact, we solve

$$[B^{(k)}]^T B^{(k)} s = [B^{(k)}]^T A v_k.$$

Note that s is the solution of a (small) least squares problem. The matrix $[B^{(k)}]^T B^{(k)}$ is a banded matrix because of the orthogonality relations between the $A v_j$'s and the v_j 's. To compute the new matrix $[B^{(k)}]^T B^{(k)}$ from the one at the previous iteration we generally need only $2(p + q)$ dot products of vectors of length n .

The interest of truncating is to reduce the storage and possibly the computational costs of the algorithm. Moreover, the size of the linear system we have to solve at each iteration is fixed (except at the beginning of the iterations) and generally small. The algorithm is the following.

```

function [V,H,L,R] = QRopt_trunc(A,b,nitmax,p,q);
% Initialization phase
n = size(A,1);
V = zeros(n,nitmax+1);
R = eye(nitmax+1,nitmax+1);
S = zeros(nitmax+1,nitmax+1);
AV = zeros(n,nitmax);
nuu = zeros(1,nitmax+1);
nuu(1) = 1;
ni = 0;
bet = norm(b);
v = b / bet;
V(:,1) = v;
% End of initialization
for k = 2:nitmax
    ni = ni + 1;
    Av = A * v; % matrix-vector product
    AV(:,k-1) = Av;
    kp = max(1,k-p); % starting index in AV
    lp = min(p,k-1); % number of vectors in AV
    kq = max(1,k-q); % starting index in V
    lq = min(q,k-1); % number of vectors in V
    while lp + lq > k % reduce the number of vectors
        if lq > 1
            kq = kq + 1;
        end
        % Compute the new matrix [B^{(k)}]^T B^{(k)}
        % ...
    end
end

```

```

lq = lq - 1;
elseif lp > 1
    kp = kp + 1;
    lp = lp - 1;
end % if
end % while
if k == 2
    vAv = v' * Av;
    alpha = Av' * Av - vAv^2;
    omega = vAv;
    z = vAv + alpha / omega;
    v = Av - z * v;
    S(1,1) = z;
else % if k == 2
    Ck = [AV(:,kp:k-2) V(:,kq:k-1)];
    sk = size(Ck,2);
    rsk = max(sk-lq,0);
    nut = [zeros(rsk,1); nuu(k-lq:k-1)'];
    CCk = Ck' * Ck; % matrix-matrix product
    CAv = Ck' * Av; % matrix-vector product
    yy = CCk \ CAv; % linear solve
    alpha = Av' * Av - CAv' * yy;
    yyy = CCk \ nut; % linear solve
    omega = Av' * Ck * yyy;
    z = yy + (alpha / omega) * yyy;
    v = Av - Ck * z; % next basis vector
    S(1:k-1,k-1) = [zeros(k-1-lq,1); z(sk-lq+1:end)];
    R(k-2-lp+2:k-2,k-1) = -z(1:lp-1);
end % if k = 2
nk = norm(v);
v = v / nk;
V(:,k) = v;
S(k,k-1) = nk;
nuu(k) = -(nuu(1:k-1) * S(1:k-1,k-1)) / S(k,k-1);
end % for k
S = S(1:ni,1:ni);
R = R(1:ni,1:ni);
H = S / R;

```

4.8 Parallel computing

Computing bases of Krylov subspaces $\mathcal{K}_k(A, r_0)$ reliably and efficiently on parallel computers is not easy. Using the monomial basis

$$(r_0 \ A r_0 \ A^2 r_0 \ \cdots \ A^{k-1} r_0)$$

would have been fine as parallelism is concerned since all the vectors can be computed simultaneously. Unfortunately, we have seen that this basis is badly conditioned and that, in finite precision arithmetic, the basis vectors may lose their linear independence long before k has reached the grade of r_0 with respect to A . For a study of lower bounds on communication costs and parallel algorithms to build Krylov bases, see [67], section 7.

As we have seen, of the three algorithms we studied to build an orthonormal basis, CGS is the most parallel since, at each step, the k dot products are independent. They can even be seen as a dense $k \times n$ matrix–vector product. On the contrary, the dot products have to be computed sequentially, one after the other, in MGS. Unfortunately again, as we will see in the next section, CGS is much less stable than MGS. To obtain orthogonality up to machine precision CGS has to be iterated.

This mismatch of Gram–Schmidt algorithms with parallelism motivated the development of Newton bases and s -step algorithms. We have seen how to construct Newton bases in Section 4.6. In the s -step techniques, given a basis vector v_j , we generate vectors $(p_0(A)v_j \ p_1(A)v_j \ \cdots \ p_s(A)v_j)$ in parallel with polynomials satisfying

$$p_{j+1}(\xi) = \xi p_j(\xi) - \theta_j p_j(\xi), \quad j = 1, \dots, s, \quad p_0 \equiv 1,$$

where the θ_j are the shifts which are known. They are computed from some Ritz values obtained by running a few iterations of the standard Arnoldi process in the initialization phase. Three-term recurrences are also used in some papers. For details on these techniques, see [58, 59, 173, 263, 321, 596, 597, 756, 824, 962, 963].

Once the $s + 1$ vectors are known, they have to be mutually orthogonalized and the last s ones have to be orthogonalized against the previous basis vectors. In [263] this was done by using MGS but with a partitioning of the set of vectors to be able to overlap computations and communications. Most papers used parallel versions of the QR factorization of a set of given vectors. This includes RODDEC in [321], tiled QR factorization algorithms in [116, 165] and the tall and skinny QR (TSQR) factorization in [265, 527, 528]. The algorithm presented in [527] is known as CA-Arnoldi (*communication-avoiding Arnoldi*). It is also recommended to perform some equilibration of the matrix and scaling of the basis vectors. In [545] it is suggested to use varying values of s in the s -step method. Orthogonalization algorithms minimizing the number of synchronization points per iteration are also described in [901].

The advantage of the Hessenberg basis over the orthogonal basis is that we do not need any dot product to compute the basis vectors. However, for stability, we have to use partial pivoting. Finding the maximum of the components in a vector is a reduction operation but that puts less stress on the communication network than computing k dot products per iteration. So far, it seems that the Hessenberg basis has not been used on parallel computers except for solving dense linear systems; see [292, 293].

An s -step version of the nonsymmetric Lanczos algorithm was developed in [597] using the monomial basis. Unlike the standard nonsymmetric Lanczos algorithm, the s -step version produces a nonsymmetric block tridiagonal matrix with $s \times s$ blocks. The computed basis vectors are not mutually biorthogonal.

A CA-nonsymmetric Lanczos with three-term recurrences was roughly described in [527]. A detailed description of a two-term recurrence CA-Lanczos is provided in [188]; see also [67]. It starts from the following version of nonsymmetric Lanczos in block form:

$$V_{n,k} = P_{n,k}R_k, \quad \tilde{V}_{n,k} = \tilde{P}_{n,k}\tilde{R}_k,$$

$$AP_{n,k} = V_{n,k}L_k + \gamma_k v_{k+1}e_k^T, \quad A^T\tilde{P}_{n,k} = \tilde{V}_{n,k}\tilde{L}_k + \tilde{\gamma}_k \tilde{v}_{k+1}e_k^T.$$

The columns of the matrices $P_{n,k}$ are biconjugate, and the columns of the matrices $V_{n,k}$ are biorthogonal. The matrices L_k and \tilde{L}_k are lower triangular. The s -step version is based on a (local) change of basis and works as follows. Assume we have $p_{sk+1}, \tilde{p}_{sk+1}$ and $v_{sk+1}, \tilde{v}_{sk+1}$. Using, for instance, Newton basis polynomials, we generate the columns of the $n \times s+1$ matrices $\mathcal{P}_{n,(k)}$ and $\tilde{\mathcal{P}}_{n,(k)}$ such that

$$\text{span}(\mathcal{P}_{n,(k)}) = \mathcal{K}_{s+1}(A, p_{sk+1}), \quad \text{span}(\tilde{\mathcal{P}}_{n,(k)}) = \mathcal{K}_{s+1}(A^T, \tilde{p}_{sk+1}),$$

(the index k is within parenthesis because it does not refer to the number of columns) and columns of matrices $\mathcal{V}_{n,(k)}$ and $\tilde{\mathcal{V}}_{n,(k)}$ such that

$$\text{span}(\mathcal{V}_{n,(k)}) = \mathcal{K}_s(A, v_{sk+1}), \quad \text{span}(\tilde{\mathcal{V}}_{n,(k)}) = \mathcal{K}_s(A^T, \tilde{v}_{sk+1}).$$

This is computed using the matrix powers kernel; see [527]. Let $\mathcal{Z}_{n,(k)} = [\mathcal{P}_{n,(k)}, \mathcal{V}_{n,(k)}]$ and $\tilde{\mathcal{Z}}_{n,(k)} = [\tilde{\mathcal{P}}_{n,(k)}, \tilde{\mathcal{V}}_{n,(k)}]$. The vectors of the standard Lanczos algorithm can be written as

$$p_{sk+j} = \mathcal{Z}_{n,(k)}\hat{p}_j, \quad \tilde{p}_{sk+j} = \tilde{\mathcal{Z}}_{n,(k)}\check{p}_j, \quad (4.48)$$

$$v_{sk+j} = \mathcal{Z}_{n,(k)}\hat{v}_j, \quad \tilde{v}_{sk+j} = \tilde{\mathcal{Z}}_{n,(k)}\check{v}_j, \quad (4.49)$$

for $j = 1, \dots, s$. The polynomial relations arising from the Newton basis can be written in matrix form

$$A\underline{\mathcal{P}}_{n,(k)} = \mathcal{P}_{n,(k)}B_{(k)}^{\mathcal{P}}, \quad A^T\underline{\tilde{\mathcal{P}}}_{n,(k)} = \tilde{\mathcal{P}}_{n,(k)}B_{(k)}^{\tilde{\mathcal{P}}},$$

$$A\underline{\mathcal{V}}_{n,(k)} = \mathcal{V}_{n,(k)}B_{(k)}^{\mathcal{V}}, \quad A^T\underline{\tilde{\mathcal{V}}}_{n,(k)} = \tilde{\mathcal{V}}_{n,(k)}B_{(k)}^{\tilde{\mathcal{V}}}.$$

where $\underline{\mathcal{P}}_{n,(k)}$ is the same as $\mathcal{P}_{n,(k)}$ but with the last column set to zero, and the same for the other matrices. Note that this differs from our usual underline notation. We observe that $B_{(k)}^{\mathcal{P}}$ is of order $s+1$ as well as $B_{(k)}^{\tilde{\mathcal{P}}}$ and $B_{(k)}^{\mathcal{V}}$ is of order s as well as $B_{(k)}^{\tilde{\mathcal{V}}}$. It yields

$$A\underline{\mathcal{Z}}_{n,(k)} = \mathcal{Z}_{n,(k)} B_{(k)}^{\mathcal{Z}}, \quad A^T \tilde{\underline{\mathcal{Z}}}_{n,(k)} = \tilde{\mathcal{Z}}_{n,(k)} B_{(k)}^{\tilde{\mathcal{Z}}},$$

with block diagonal matrices,

$$B_{(k)}^{\mathcal{Z}} = \begin{pmatrix} B_{(k)}^{\mathcal{P}} & \\ & B_{(k)}^{\mathcal{V}} \end{pmatrix}, \quad B_{(k)}^{\tilde{\mathcal{Z}}} = \begin{pmatrix} B_{(k)}^{\tilde{\mathcal{P}}} & \\ & B_{(k)}^{\tilde{\mathcal{V}}} \end{pmatrix},$$

and $\underline{\mathcal{Z}}_{n,(k)}$ (resp. $\tilde{\underline{\mathcal{Z}}}_{n,(k)}$) is constructed from $\underline{\mathcal{P}}_{n,(k)}$ and $\underline{\mathcal{V}}_{n,(k)}$ (resp. $\tilde{\underline{\mathcal{P}}}_{n,(k)}$ and $\tilde{\underline{\mathcal{V}}}_{n,(k)}$). Then,

$$Ap_{sk+j} = A\mathcal{Z}_{n,(k)}\hat{p}_j = A\underline{\mathcal{Z}}_{n,(k)}\hat{p}_j = \mathcal{Z}_{n,(k)}B_{(k)}^{\mathcal{Z}}\hat{p}_j,$$

$$A^T\tilde{p}_{sk+j} = A^T\tilde{\mathcal{Z}}_{n,(k)}\check{p}_j = A^T\tilde{\underline{\mathcal{Z}}}_{n,(k)}\check{p}_j = \tilde{\mathcal{Z}}_{n,(k)}B_{(k)}^{\tilde{\mathcal{Z}}}\check{p}_j.$$

Substituting these relations in those of the nonsymmetric Lanczos algorithm yields relations for computing the vectors of coefficients \hat{p}_j , \check{p}_j , \hat{v}_j and \check{v}_j . However, some scalar coefficients have to be computed to use these recurrences. Let $G_{(k)} = \tilde{\mathcal{Z}}_{n,(k)}^T \mathcal{Z}_{n,(k)}$ of order $2s + 1$. The scalars are computed by

$$(\tilde{v}_{sk+j+1}, v_{sk+j+1}) = (\hat{v}_{j+1}, G_{(k)}\check{v}_{j+1}), \quad (\tilde{p}_{sk+j}, Ap_{sk+j}) = (\hat{p}_j, G_{(k)}B_{(k)}^{\mathcal{Z}}\check{p}_j).$$

The vectors corresponding to the standard algorithm are recovered from relations (4.48) and (4.49). It remains to compute the normalizing coefficients, for details see [188], page 37. Per s steps, communications occur only when computing $\mathcal{Z}_{n,(k)}$, $\tilde{\mathcal{Z}}_{n,(k)}$ and $G_{(k)}$. There is no communication in the loop over $j = 1, \dots, s$. Mathematically, this CA-Lanczos algorithm is equivalent to the sequential algorithm. However, this may not be the case in finite precision arithmetic. Look-ahead techniques for taking care of the possible (near-) breakdowns are discussed in [188], page 204.

So far, no parallel version of the construction of the Q-OR optimal basis has been proposed.

4.9 Finite precision arithmetic and numerical experiments

4.9.1 Orthogonal basis

Gram–Schmidt orthogonalization algorithms in finite precision arithmetic were first studied in the context of orthogonalizing the columns of a given rectangular matrix B to compute a QR factorization $B = QR$ with R upper triangular and Q orthogonal; see, for instance, [98, 99]. The main application was solving least squares problems;

see [100, 620]. Due to the rounding errors, there is in general a gradual loss of orthogonality in the computed columns of the matrix Q . The differences between CGS and MGS were first shown experimentally in [769] where it was observed that orthogonality is better preserved in MGS than in CGS.

The reason for the loss of orthogonality can be seen with only two real vectors, as it is explained in [99]. Let q_1 of unit norm and b_2 be two vectors. We orthogonalize b_2 against q_1 by

$$\hat{q}_2 = b_2 - r_{1,2} q_1, \quad r_{1,2} = q_1^T b_2, \quad q_2 = \frac{\hat{q}_2}{\|\hat{q}_2\|}.$$

Let us assume that q_1 and b_2 are known exactly and that the norms are computed exactly. In finite precision arithmetic we compute $fl(\hat{q}_2)$ and it can be shown that

$$\|fl(\hat{q}_2) - \hat{q}_2\| \leq cu\|b_2\|,$$

where c is equal to the length of the vectors times a small constant and u is the unit roundoff. Then,

$$I - \left(q_1 \frac{fl(\hat{q}_2)}{\|fl(\hat{q}_2)\|} \right)^T \left(q_1 \frac{fl(\hat{q}_2)}{\|fl(\hat{q}_2)\|} \right) = - \begin{pmatrix} 0 & \frac{q_1^T fl(\hat{q}_2)}{\|fl(\hat{q}_2)\|} \\ \frac{q_1^T fl(\hat{q}_2)}{\|fl(\hat{q}_2)\|} & 0 \end{pmatrix}.$$

Therefore, the level of orthogonality, which is the norm of this matrix, is

$$\frac{|q_1^T fl(\hat{q}_2)|}{\|fl(\hat{q}_2)\|}.$$

Now, since q_1 and \hat{q}_2 are exactly orthogonal,

$$|q_1^T fl(\hat{q}_2)| = |q_1^T (fl(\hat{q}_2) - \hat{q}_2)| \leq \|fl(\hat{q}_2) - \hat{q}_2\| \leq cu\|b_2\|.$$

Hence, the bound on the level of orthogonality is proportional to

$$u \frac{\|b_2\|}{\|fl(\hat{q}_2)\|}.$$

This bound can be large if $\|fl(\hat{q}_2)\|$ is small. This happens if there is cancellation in the computation of \hat{q}_2 , that is, if the angle between q_1 and b_2 is small.

Let us consider a small example,

$$B = \begin{pmatrix} 0.7071067812 & 0.7071067883 \\ 0.7071067812 & 0.7071067741 \end{pmatrix}.$$

This matrix is of rank 2. The second column is equal to the first column slightly rotated. With CGS and MGS the computed matrix Q is

$$Q = \begin{pmatrix} 0.7071067812 & 0.7071067923 \\ 0.7071067812 & -0.7071067701 \end{pmatrix},$$

and

$$Q^T Q = \begin{pmatrix} 1 & 1.570092452 \cdot 10^{-8} \\ 1.570092452 \cdot 10^{-8} & 1 \end{pmatrix}$$

which is far from the identity matrix. If we orthogonalize a second time we obtain a matrix Q_2 such that

$$Q_2^T Q_2 = \begin{pmatrix} 1 & -8.865115929 \cdot 10^{-17} \\ -8.865115929 \cdot 10^{-17} & 1 \end{pmatrix}$$

which is the identity matrix up to working precision.

The loss of orthogonality in CGS was studied in [425] (see also [424]) where it is proved that the computed quantities satisfy

$$\|I - Q^T Q\| \leq \frac{c_1}{1 - c_2 u \kappa^2(B)} u \kappa^2(B),$$

when $c_2 u \kappa^2(B) < 1$ and c_1 depends on the dimension of B . This condition corresponds to $B^T B$ being numerically nonsingular.

In [98] it was proved that, with MGS, the computed matrix Q satisfies

$$\|I - Q^T Q\| \leq \frac{c_1}{1 - c_2 \kappa(B) u} u \kappa(B),$$

if $c_2 u \kappa(B) < 1$ where c_1 and c_2 are constants depending on the number of columns of B . For MGS the bound depends on $\kappa(B)$ whence for CGS it depends on $\kappa^2(B)$. The rank of the computed Q in MGS was studied in [420] where, under some restrictions on the condition number of B and the arithmetic, it was shown that $\kappa(Q) \leq 1.3$.

However, we are not interested in any matrix but in the construction of a basis for the Krylov subspaces. In theory, as it was stated in [461], the Arnoldi process using any of the three variants obtains the QR factorization of $B = (v_1 \ A V_{n,k-1})$ where k is the iteration number. In fact, we have to deal with $(v_1 \ f l(AV_{n,k-1}))$ but the effects of rounding errors in the matrix–vector products can be included in the bounds and one can work instead with $B = (v_1 \ A V_{n,k-1})$ where the multiplication is done exactly.

For the Householder and MGS implementations, the computed basis vectors satisfy a perturbed Arnoldi relation,

$$AV_{n,k} = V_{n,k+1} \underline{H}_k + F_{n,k},$$

where the perturbation term, due to rounding errors, is bounded by

$$\|F_{n,k}\| \leq c_1 \eta \varepsilon \|A\|, \quad \eta = k^{\frac{3}{2}} n + k^{\frac{1}{2}} \ell n^{\frac{1}{2}},$$

with c_1 a constant independent of the problem, ℓ the maximum number of nonzero entries per row of A and ε is the machine epsilon; see [279, 774]. Expressions for the basis vectors generated by perturbed Arnoldi relations are given in [1018, 1020].

For the Householder implementation we have (see [279])

$$\|I - V_{n,k}^T V_{n,k}\| \leq c_2 k^{\frac{3}{2}} n \varepsilon.$$

This inequality is almost the best we can hope for, that is, orthogonality to working precision if k and n are not too large. Unfortunately, as we have seen, the Householder variant is more expensive than MGS.

For MGS we have

$$\|I - V_{n,k}^T V_{n,k}\| \leq c_3 \eta \varepsilon \kappa([v_1 \ A V_{n,k}]),$$

where c_3 is a constant independent of the problem. This result was further extended in [461]. In this paper it is assumed that $\|A\| = 1$ and that the basis vectors are exactly of unit norm, but this is only a change of scaling which does not restrict the generality of the results. An upper bound was obtained for $\kappa([v_1 \ A V_{n,k}])$. This result is based on the study of the change of the smallest singular value of a matrix when we append a new column. Let V be an $n \times k$ matrix of rank k with $n > k$ and h be a vector of length n . Then,

$$\sigma_{\min}(V) \geq \sigma_{\min}([V \ h]) \geq \frac{\tau}{(\sigma_{\min}^2(V) + \tau^2 + \|h\|^2)^{\frac{1}{2}}} \sigma_{\min}(V),$$

where $\tau = \min_y \|h - Vy\|$. This result yields

$$\sigma_{\min}([v_1 \ A V_{n,k}]) \geq \min_y \|v_1 - AV_{n,k}\| \frac{\sigma_{\min}(AV_{n,k})}{(2 + \sigma_{\min}^2(AV_{n,k}))^{\frac{1}{2}}}.$$

Finally,

$$\kappa([v_1 \ A V_{n,k}]) \leq c_4 \frac{\kappa(A)}{\min_y \|v_1 - AV_{n,k}\|},$$

under the conditions

$$\|A\| = 1, \quad c_3 n^{\frac{3}{2}} \varepsilon \leq \frac{1}{2}, \quad \min_y \|v_1 - AV_{n,k}\| > c_5 k n^{\frac{3}{2}} \kappa(A) \varepsilon.$$

Therefore, as long as the minimum is not too small, the level of orthogonality is bounded by a quantity which is of the order of $\varepsilon \kappa(A)$. But, if the minimum of the least squares problem is small, the level of orthogonality can be large independently

of the condition number. Moreover, the level of orthogonality and the minimum of the least squares problem are almost inversely proportional.

To bound the condition number of $V_{n,k}$ one cannot use the results of [420] since, as it was remarked in [736], to compute the constants and the restriction on the condition number, [420] used results of Wilkinson corresponding to dot products computed in quadruple precision. This was corrected in [736] where it is shown that when using MGS on a matrix B ,

$$k\gamma_n \kappa_F(B_{n,k}) < \frac{1}{8} \Rightarrow \kappa(V_{n,k}) < \frac{4}{3},$$

where $\gamma_n = cnu/(1 - cnu)$ with c a small integer and

$$\kappa_F(B) = \min_{\text{diagonal } D > 0} \|BD\|_F / \sigma_{\min}(BD).$$

In practice, this gives a restriction on the condition number of A which is $\sigma_{\min}(A) \gg n^2 u \|A\|_F$. Note that this may not be satisfied with some matrices in our set of examples having a large condition number. But the factor n^2 is an overestimate and might be replaced by n ; see [736]. We will see in the numerical experiments below that $\kappa(V_{n,k}) \leq 4/3$ and the columns of $V_{n,k}$ are linearly independent until the minimum of $\|v_1 - AV_{n,k}y\|$ has not reached its minimum value over k .

Let us first compare on some examples the three variants of the Arnoldi process to compute an orthonormal basis of the Krylov subspace. Mathematically they give the same result but, as we have seen above, this may not be the case in finite precision arithmetic.

We first consider the matrix `fs_183_6` of order 183; see Appendix A. It has 1000 nonzero entries, its norm is $1.180839 \cdot 10^9$ and its condition number is $1.736781 \cdot 10^{11}$. This badly conditioned matrix has a very large real eigenvalue whose value is $8.7314 \cdot 10^8$, many (almost) multiple eigenvalues and only 26 complex eigenvalues.

Figure 4.1 shows the level of orthogonality of the basis vectors measured by $\|I - V_{n,k}^T V_{n,k}\|$ for $k = 1, \dots, 140$. The number of iterations was chosen to have the norm of the residual $b - Ax_k$ in GMRES (to be described in Chapter 5) reaching the maximum attainable accuracy. We observe that CGS and MGS start losing orthogonality from the beginning but CGS is much worse than MGS. The level of orthogonality of the Householder variant stays at the machine epsilon level.

Figure 4.2 displays the minimum singular value σ_{\min} of $V_{n,k}$ as a function of k . For CGS σ_{\min} starts to decrease rapidly at iteration 20 reaching the machine epsilon at iteration 40. For MGS σ_{\min} starts to decrease only after iteration 50 and it decreases more slowly to a level around 10^{-12} .

As we can see in Figure 4.3, the matrices $V_{n,k}$ are of full-rank for MGS and Householder but not for CGS for which the basis vectors become rapidly linearly dependent. However, if we use reorthogonalization in CGS as we can observe in Figure 4.4 we obtain a level of orthogonality close to the machine precision and the matrices $V_{n,k}$ are of full-rank but this is twice as costly as using CGS or MGS.

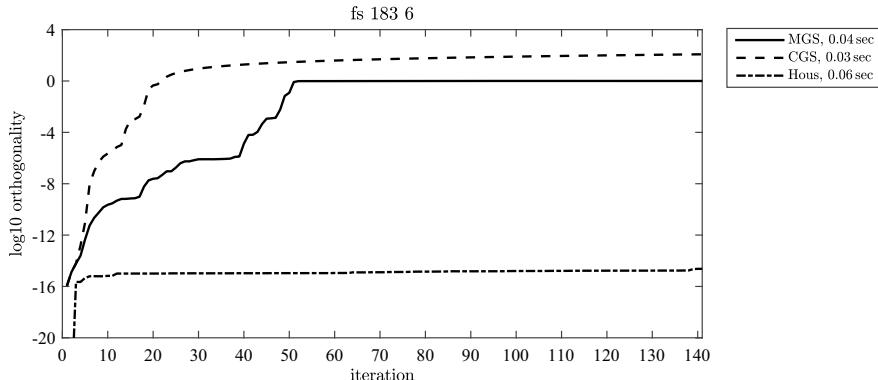


Fig. 4.1 fs 183 6, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

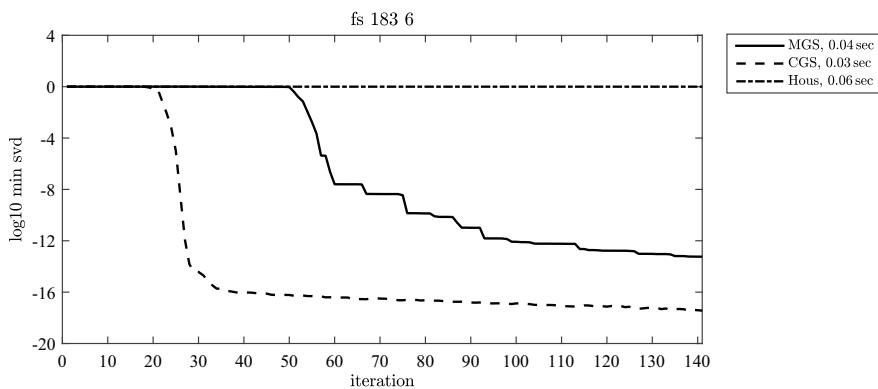


Fig. 4.2 fs 183 6, \log_{10} of minimum singular value of $V_{n,k}$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

However, we note that CGS with reorthogonalization is easier to parallelize than MGS or Householder.

Let us illustrate numerically some of the results we have stated before about the bounds for the level of orthogonality and the condition number of $V_{n,k}$. Figure 4.5 shows the \log_{10} of the level of orthogonality and of the condition number of $V_{n,k}$ as a function of k . The vertical line corresponds to the first iteration where $\text{cond}(V_{n,k}) > 4/3$. We observe that it corresponds to the maximum of the level of orthogonality.

Figure 4.6 displays the \log_{10} of the level of orthogonality and of $\min_y \|v_1 - AV_{n,k}y\|$ as a function of k . We observe that there are almost symmetrical with respect to a parallel to the x -axis at the level 10^{-8} . Moreover, the minimum of the least squares problems reaches its minimum value at iteration 51, that is, when for the first time $\text{cond}(V_{n,k}) > 4/3$.

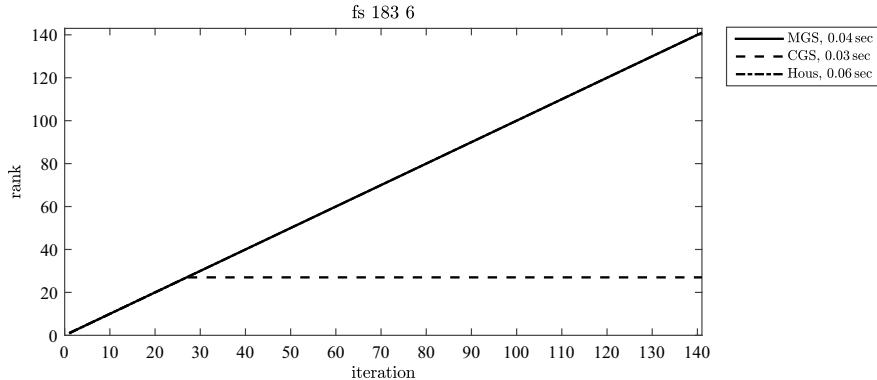


Fig. 4.3 fs 183 6, numerical rank of $V_{n,k}$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

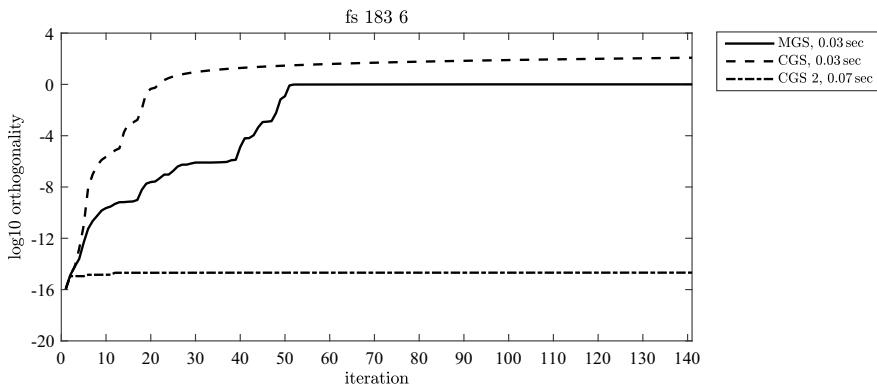


Fig. 4.4 fs 183 6, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$, Arnoldi MGS (plain), CGS (dashed), CGS with reorthogonalization (dot-dashed)

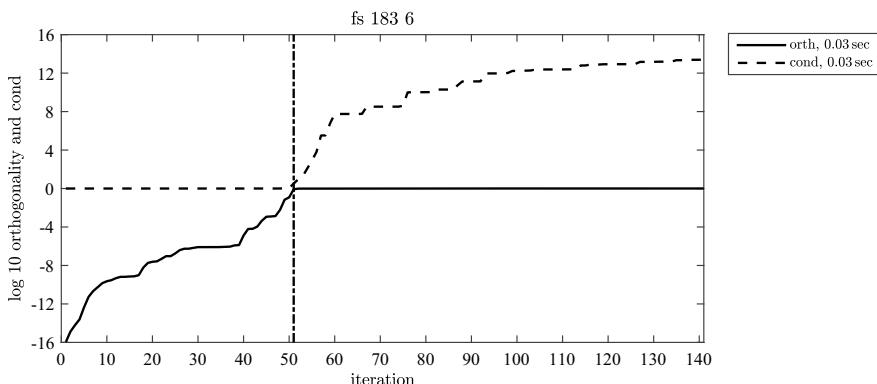


Fig. 4.5 fs 183 6, MGS, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$ (plain) and \log_{10} of $\text{cond}(V_{n,k})$ (dashed)

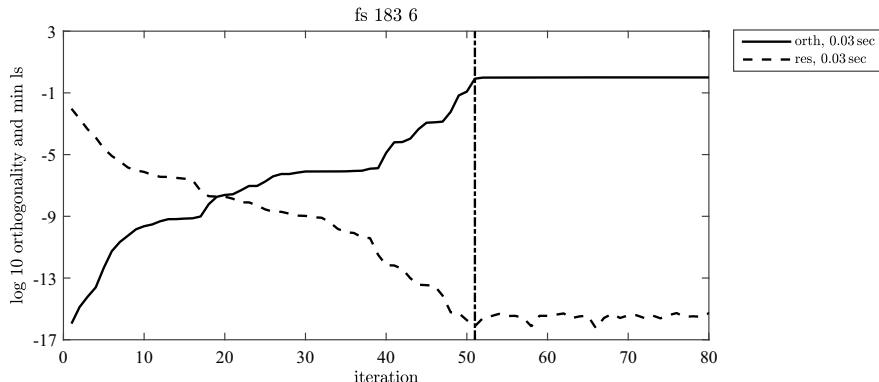


Fig. 4.6 fs 183 6, MGS, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$ (plain) and \log_{10} of $\min_y \|v_1 - AV_{n,k}y\|$

Things are not always so bad for CGS as in the previous example with the matrix **fs 183 6** which has a very large condition number. Figures 4.7, 4.8 and 4.9 display the results for a matrix **fs 680 1c** which is the matrix **fs 680** left multiplied by the inverse of its diagonal. Its order is 680, and its condition number is 8694.4. The level of orthogonality is not much different for CGS and MGS. The smallest CGS singular value starts to decrease at iteration 90 with a faster decrease for CGS than for MGS. It explains the loss of linear independence of the basis vectors after iteration 100. For MGS the condition number of $V_{n,k}$ becomes larger than $4/3$ at iteration 98 which is when the level of orthogonality approximately reaches its maximum value.

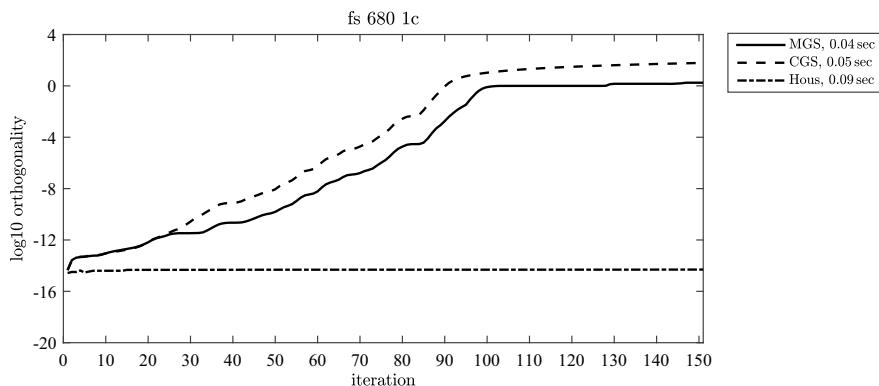


Fig. 4.7 fs 680 1c, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

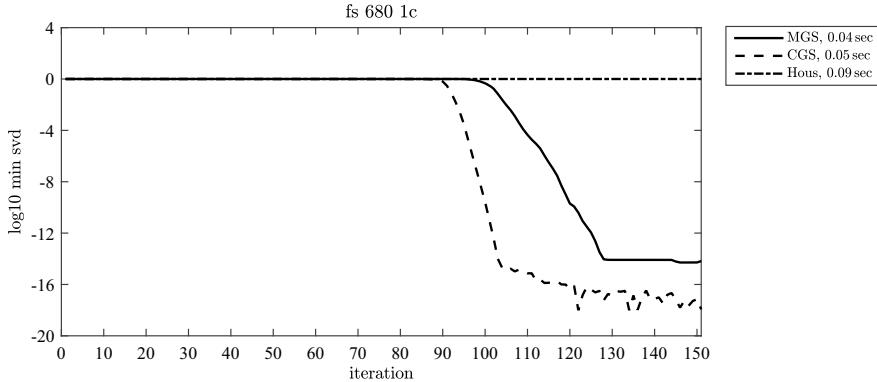


Fig. 4.8 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

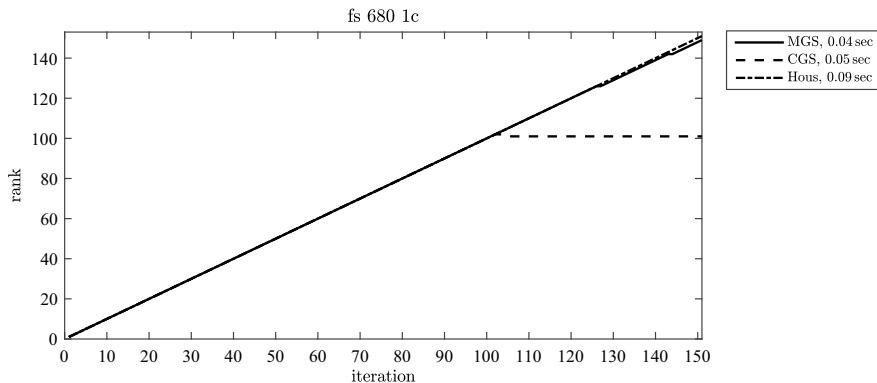


Fig. 4.9 fs 680 1c, numerical rank of $V_{n,k}$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

Everything is even better if we consider a dense random matrix of order 100 with entries taken from a normal distribution; see Figures 4.10 and 4.11. Its condition number is 221.1. The growth of the level of orthogonality for CGS and MGS is quite limited, and the basis vectors stay linearly independent for the three Arnoldi variants. For MGS the condition number of $V_{n,k}$ stays equal to 1 to working precision.

Tables 4.1, 4.3 and 4.5 show the level of orthogonality for small matrices whose characteristics are given in Appendix A. The matrices are ordered by increasing condition number. The value k_{max} was generally chosen to have the norm of the residual $b - Ax_k$ in GMRES reaching the maximum attainable accuracy. Then, we computed $\|I - V_{n,k}^T V_{n,k}\|$ for $k = k_{max}/2, 3 k_{max}/4, k_{max}$. Tables 4.2, 4.4 and 4.6 tell us if $V_{n,k}$ is of full-rank (value 1) or not (value 0).

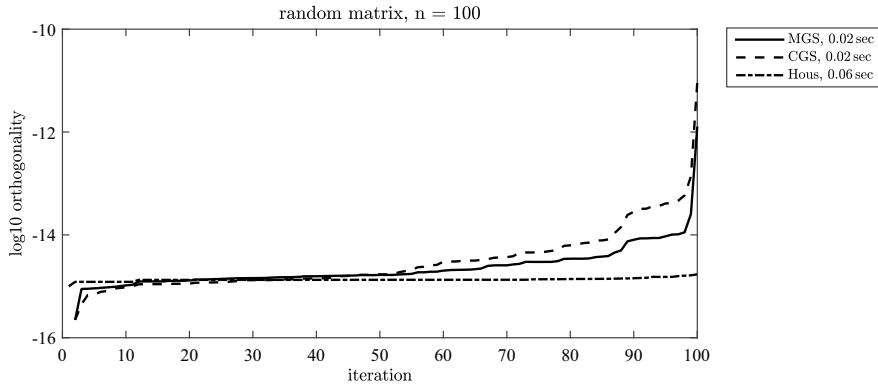


Fig. 4.10 random matrix $n = 100$, $\log_{10} \|I - V_{n,k}^T V_{n,k}\|$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

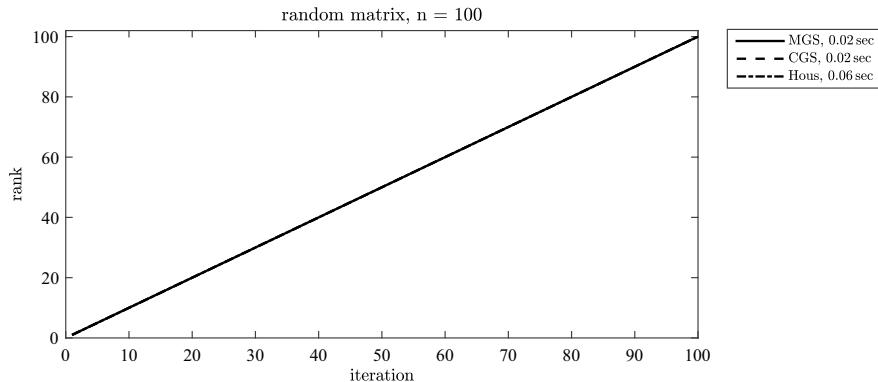


Fig. 4.11 random matrix $n = 100$, numerical rank of $V_{n,k}$, Arnoldi MGS (plain), CGS (dashed), Householder (dot-dashed)

The Householder variant always provides full-rank matrices $V_{n,k}$. In most cases linear independence is lost with MGS only at the end of the computation whence with CGS, linear independence is lost sooner. From these results we see that, as it is well known, MGS is much more reliable than CGS. The Householder variant always gives a basis orthogonal to working precision but it is much more expensive than MGS.

Tables 4.7–4.8 show results for some larger matrices whose characteristics are described in Appendix A. The conclusions are the same as for the smaller matrices even though we are doing a large number of iterations.

Table 4.1 Orthogonal basis, level of orthogonality $\|I - V_{n,k}^T V_{n,k}\|$

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
pde225	150	MGS	8.87268e-06	1.00000e+00	1.04105e+00
		CGS	1.02309e-05	2.01364e+01	5.81332e+01
		Hous	1.49221e-15	1.69951e-15	1.53115e-15
gre 343	300	MGS	5.13710e-11	1.64483e-08	1.00000e+00
		CGS	6.04734e-11	1.95206e-08	1.82692e+01
		Hous	6.90902e-15	6.91382e-15	7.12645e-15
jphw 991	200	MGS	9.99884e-01	1.01354e+00	1.58572e+00
		CGS	1.28283e+01	6.28239e+01	1.13823e+02
		Hous	3.11032e-15	3.11065e-15	1.77636e-15
pde2961	320	MGS	9.98643e-11	1.20878e-04	1.00000e+00
		CGS	3.72243e-09	8.37528e-02	7.19384e+01
		Hous	1.30343e-15	1.31035e-15	1.42505e-15
jagmesh1	300	MGS	2.56071e-11	1.05117e-02	9.99978e-01
		CGS	2.46037e-11	9.98410e-03	6.23556e+01
		Hous	9.10403e-15	9.10403e-15	8.99301e-15
bfwa782	350	MGS	2.56554e-10	3.40441e-06	9.99999e-01
		CGS	4.02303e-06	4.35283e-01	7.03859e+01
		Hous	2.67729e-15	2.67943e-15	2.23915e-15
dw2048	1100	MGS	9.72802e-10	3.79597e-08	1.23845e+00
		CGS	1.23013e-09	5.81286e-08	1.11669e+02
		Hous	3.99922e-15	3.99927e-15	3.70655e-15
jagmesh2	800	MGS	5.04555e-10	5.99434e-09	1.00000e+00
		CGS	5.10730e-10	6.16361e-09	6.39863e+01
		Hous	7.54981e-15	7.54981e-15	9.77019e-15
raefsky2	450	MGS	7.65051e-12	5.37674e-07	9.99990e-01
		CGS	1.14645e-08	3.08352e-03	8.24263e+01
		Hous	1.20189e-15	1.28575e-15	1.63044e-15
fs 680 1c	150	MGS	1.03158e-06	1.00000e+00	1.76365e+00
		CGS	1.40037e-04	2.36480e+01	6.16462e+01
		Hous	4.84144e-15	4.85895e-15	4.68821e-15
add20	600	MGS	6.70889e-05	2.08236e-01	9.99993e-01
		CGS	1.15854e-04	4.38106e+01	1.94805e+02
		Hous	2.80932e-15	3.05958e-15	3.17504e-15
raefsky1	350	MGS	2.10633e-09	2.20035e-04	1.00000e+00
		CGS	4.99345e-04	4.17342e+01	1.29733e+02
		Hous	1.65301e-15	1.71249e-15	1.95837e-15
jagmesh4	700	MGS	7.71645e-10	4.19945e-09	1.00000e+00
		CGS	7.75091e-10	4.21131e-09	5.27035e+01
		Hous	9.69737e-15	9.85340e-15	7.09608e-15

Table 4.2 Orthogonal basis, full-rank?

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
pde225	150	MGS	1	1	0
		CGS	1	0	0
		Hous	1	1	1
gre 343	300	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
jpwh 991	200	MGS	1	0	0
		CGS	1	0	0
		Hous	1	1	1
pde2961	320	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
jagmesh1	300	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
bfwa782	350	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
dw2048	1100	MGS	1	1	0
		CGS	1	1	0
		Hous	1	1	1
jagmesh2	800	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
raefsky2	450	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
fs 680 1c	150	MGS	1	1	0
		CGS	1	0	0
		Hous	1	1	1
add20	600	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
raefsky1	350	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
jagmesh4	700	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1

Table 4.3 Orthogonal basis, level of orthogonality $\|I - V_{n,k}^T V_{n,k}\|$

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 680 1	200	MGS	6.77390e-05	1.00022e+00	1.00031e+00
		CGS	2.79658e+01	7.79633e+01	1.28963e+02
		Hous	1.53747e-15	1.77852e-15	2.02702e-15
sherman1	500	MGS	2.70845e-10	1.26492e-05	1.00000e+00
		CGS	2.26463e-09	1.07433e-04	7.62024e+01
		Hous	1.98982e-15	2.07737e-15	2.47558e-15
nos3	320	MGS	1.20557e-10	2.76875e-06	1.00000e+00
		CGS	4.36592e-10	1.45541e-05	4.00054e+01
		Hous	1.40815e-15	1.67583e-15	1.81767e-15
sherman5	1200	MGS	8.42376e-11	1.20163e-06	1.00000e+00
		CGS	1.31342e+02	4.31332e+02	7.32331e+02
		Hous	2.34924e-15	3.09819e-15	3.94926e-15
cavity05	600	MGS	2.65855e-09	1.88779e-04	1.00000e+00
		CGS	5.98539e+01	2.09854e+02	3.60854e+02
		Hous	1.93201e-15	2.36063e-15	2.84076e-15
e05r0500	260	MGS	1.30545e-12	1.23258e-11	1.00000e+00
		CGS	2.07901e-10	2.49630e+01	6.59620e+01
		Hous	1.99227e-15	2.08694e-15	2.40684e-15
comsol	300	MGS	7.35558e-11	1.11730e-04	9.99894e-01
		CGS	1.02401e-01	6.61955e+01	1.42194e+02
		Hous	1.97194e-15	2.03769e-15	2.14028e-15
olm1000	600	MGS	1.21041e-10	1.94756e-08	1.00002e+00
		CGS	1.10633e-07	3.46162e-05	9.39512e+01
		Hous	9.32592e-15	9.32592e-15	6.21500e-15
cavity10	900	MGS	3.69730e-09	1.69909e-04	9.99990e-01
		CGS	7.97982e+01	3.04783e+02	5.30782e+02
		Hous	1.56174e-15	1.80295e-15	2.29916e-15
steam2	200	MGS	3.16985e-05	2.97633e-02	9.99998e-01
		CGS	6.93494e+00	5.69116e+01	1.07910e+02
		Hous	1.63269e-15	2.10438e-15	2.71183e-15
1138bus	700	MGS	4.66556e-10	4.36917e-06	9.99990e-01
		CGS	3.94684e-09	5.94643e-05	8.01764e+01
		Hous	2.88885e-15	2.88893e-15	2.89164e-15
steam1	250	MGS	2.16062e-09	4.42644e-04	1.00000e+00
		CGS	2.87627e-05	4.26646e+01	9.56639e+01
		Hous	2.00946e-15	2.17903e-15	2.77963e-15
bcstk26	1000	MGS	3.51248e-09	1.97226e-08	1.35347e-07
		CGS	2.51641e-08	1.73838e-07	2.14991e-06
		Hous	2.71555e-15	3.52105e-15	3.66456e-15

(continued)

Table 4.3 (continued)

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
nos7	500	MGS	4.33362e-06	3.97429e-03	1.00000e+00
		CGS	8.46358e+00	1.33341e+02	2.59341e+02
		Hous	1.94902e-15	2.35107e-15	2.90346e-15
watt1	350	MGS	6.04305e-06	4.88901e-03	9.97121e-01
		CGS	7.17213e+01	1.59587e+02	2.47548e+02
		Hous	2.51821e-15	2.53828e-15	2.76226e-15
bcsstk14	1000	MGS	2.18847e-10	7.81651e-10	4.07960e-09
		CGS	4.35543e-09	1.79805e-08	1.16017e-07
		Hous	2.76062e-15	3.21571e-15	3.80704e-15

Table 4.4 Orthogonal basis, full-rank?

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 680 1	200	MGS	1	0	0
		CGS	0	0	0
		Hous	1	1	1
sherman1	500	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
nos3	320	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
sherman5	1200	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
cavity05	600	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
e05r0500	260	MGS	1	1	0
		CGS	1	0	0
		Hous	1	1	1
comsol	300	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
olm1000	600	MGS	1	1	0
		CGS	1	1	0
		Hous	1	1	1
cavity10	900	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1

(continued)

Table 4.4 (continued)

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
steam2	200	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
1138bus	700	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
steam1	250	MGS	1	1	0
		CGS	1	0	0
		Hous	1	1	1
bcsstk26	1000	MGS	1	1	1
		CGS	1	1	1
		Hous	1	1	1
nos7	500	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
watt1	350	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
bcsstk14	1000	MGS	1	1	1
		CGS	1	1	1
		Hous	1	1	1

Table 4.5 Orthogonal basis, level of orthogonality $\|I - V_{n,k}^T V_{n,k}\|$

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 183 6	60	MGS	8.04143e-07	1.16067e-03	1.00000e+00
		CGS	9.10694e+00	2.40978e+01	4.00956e+01
		Hous	1.05275e-15	1.07161e-15	1.11871e-15
bcsstk20	500	MGS	1.37254e-07	1.09236e-06	1.00000e+00
		CGS	1.94453e-04	2.77081e-01	3.88859e+01
		Hous	2.44373e-15	2.68352e-15	3.33799e-15
mcfe	820	MGS	3.61548e-09	9.51558e-08	1.00000e+00
		CGS	2.77459e+02	4.82459e+02	6.32459e+02
		Hous	2.45425e-15	3.43660e-15	4.03524e-15
nnc	250	MGS	5.21281e-11	7.15228e-08	1.00000e+00
		CGS	4.42729e-05	1.09073e+01	6.75230e+01
		Hous	1.31923e-15	1.77931e-15	2.60803e-15
lnsp	420	MGS	4.64988e-04	1.26153e-03	1.00000e+00
		CGS	2.04000e+02	3.09000e+02	4.15000e+02
		Hous	2.60896e-15	2.86824e-15	3.11344e-15

Table 4.6 Orthogonal basis, full-rank?

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 183 6	60	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
bcstk20	500	MGS	1	1	0
		CGS	1	1	0
		Hous	1	1	1
mcfe	820	MGS	1	1	0
		CGS	0	0	0
		Hous	1	1	1
nnc	250	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
lisp	420	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1

Table 4.7 Orthogonal basis, level of orthogonality $\|I - V_{n,k}^T V_{n,k}\|$

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
add32	200	MGS	3.10819e-03	9.99991e-01	1.00000e+00
		CGS	1.39133e-03	2.47693e+01	7.57658e+01
		Hous	2.13793e-15	2.19081e-15	2.51120e-15
supg 0001	120	MGS	4.45235e-13	2.10302e-08	1.00000e+00
		CGS	3.96113e-13	1.81957e-08	1.34168e+01
		Hous	1.22888e-15	1.22888e-15	1.57842e-15
supg 00001	220	MGS	1.91751e-11	1.89860e-07	1.00000e+00
		CGS	1.92612e-11	2.28379e-07	2.29456e+01
		Hous	1.64019e-15	2.07781e-15	1.66577e-15
ex37	150	MGS	3.27771e-05	9.91130e-01	1.00000e+00
		CGS	8.77015e-05	1.35284e+01	5.15234e+01
		Hous	3.34951e-15	4.22634e-15	3.74571e-15
supg 001	150	MGS	1.52140e-12	4.20265e-06	1.00000e+00
		CGS	3.17955e-12	8.62691e-06	2.25818e+01
		Hous	1.37764e-15	1.46162e-15	1.51709e-15
supg 000001	240	MGS	1.80847e-12	6.57180e-08	1.00000e+00
		CGS	1.74893e-12	6.49281e-08	2.59817e+01
		Hous	1.44660e-15	1.51999e-15	1.52530e-15
supg 01	500	MGS	1.15063e-10	9.62993e-01	1.00000e+00
		CGS	9.14382e-10	1.97478e+01	1.45744e+02
		Hous	3.55521e-15	3.55956e-15	1.95669e-15

(continued)

Table 4.7 (continued)

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
supg 1	1100	MGS	1.03296e-08	4.98544e-06	1.00000e+00
		CGS	1.32156e-07	6.41814e-05	1.44281e+02
		Hous	1.68535e-15	1.84849e-15	1.87925e-15
supg 100	1000	MGS	1.37925e-05	9.72074e-03	9.90547e-01
		CGS	9.36987e-05	2.84146e+00	8.24447e+01
		Hous	1.77810e-15	1.88000e-15	1.88800e-15
wang4	700	MGS	3.96381e-09	4.27821e-04	9.99999e-01
		CGS	1.59301e+02	3.34301e+02	5.10301e+02
		Hous	3.71635e-15	3.73801e-15	3.75841e-15
memplus	900	MGS	6.02615e-05	1.35445e-02	9.83414e-01
		CGS	8.55692e-04	4.70760e+01	2.73072e+02
		Hous	5.01452e-15	5.32946e-15	5.14741e-15
convdiff xu 5000	400	MGS	4.99947e-11	1.28082e-02	9.99280e-01
		CGS	2.07740e-04	6.73998e+01	1.68399e+02
		Hous	1.76539e-14	1.76539e-14	1.69879e-14
convdiff xu 1000	700	MGS	3.31018e-05	1.85644e-03	1.00000e+00
		CGS	1.46945e+02	3.21944e+02	4.97944e+02
		Hous	2.08730e-14	2.08730e-14	1.59882e-14
sherman3	800	MGS	9.65353e-09	1.88123e-04	9.99999e-01
		CGS	1.90760e-07	2.03378e-01	1.20114e+02
		Hous	2.16486e-15	2.24186e-15	2.32287e-15
convdiff xu 500	1100	MGS	1.38315e-04	8.70403e-01	1.01955e+00
		CGS	3.38976e+02	6.13976e+02	8.89975e+02
		Hous	2.92005e-14	2.92005e-14	2.12075e-14

Table 4.8 Orthogonal basis, full-rank?

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
add32	200	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
supg 0001	120	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
supg 00001	220	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1

(continued)

Table 4.8 (continued)

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
ex37	150	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
supg 001	150	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
supg 000001	240	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
supg 01	500	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
supg 1	1100	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
supg 100	1000	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
wang4	700	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
memplus	900	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
convdiff xu 5000	400	MGS	1	1	1
		CGS	1	0	0
		Hous	1	1	1
convdiff xu 1000	700	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1
sherman3	800	MGS	1	1	1
		CGS	1	1	0
		Hous	1	1	1
convdiff xu 500	1100	MGS	1	1	1
		CGS	0	0	0
		Hous	1	1	1

4.9.2 Hessenberg basis

Let us now consider the computation of the Hessenberg basis. We show the results for the computation with pivoting (P) and without pivoting (NP). To compute the basis we use a column-oriented variant of Gaussian elimination. Therefore, we can anticipate that, without pivoting, we will have some problems. Even with partial pivoting Gaussian elimination is not backward stable; see [521, 982].

We start by considering the matrix `fs 183 6`. We look at the minimum singular value and the rank of $V_{n,k}$ as a function of k . We observe in Figure 4.12 that, without pivoting, the minimum singular value is very oscillating. It is even 0 at iteration 28. With pivoting the minimum singular value is bounded from below. In Figure 4.13 we see that the matrix $V_{n,k}$ stays of full-rank whence linear independence is lost very early without pivoting. We note that the maximum singular value is becoming very large, already $2.4931 \cdot 10^{28}$ at iteration 10. It demonstrates that the algorithm without pivoting is very unreliable.

Figure 4.14 shows that things are a little better for `fs 680 1c` and the non-pivoting algorithm at the beginning of the iterations, but the basis loses full-rank rapidly. With pivoting the matrix of the basis vectors is of full-rank up to the end.

Tables 4.9, 4.10 and 4.11 show the minimum singular value for our set of small matrices. We observe that, with pivoting, the minimum singular values are bounded from below. With pivoting the matrices $V_{n,k}$ are all of full-rank for these values of k whence, without pivoting, for most cases the basis vectors are linearly dependent. The same conclusions apply for the set of large matrices. Hence, we do not show the results.

4.9.3 Biorthogonal basis

The Lanczos biorthogonalization process in finite precision arithmetic was studied in [55]. The analysis is comparable to what was done by Chris Paige for the symmetric Lanczos algorithm; see [730]. The normalization is such that $\omega_k = (w_k, v_k) = 1$. It is shown that the computed quantities satisfy

$$\begin{aligned} AV_{n,k} &= V_{n,k}T_k + \rho_{k+1}v_{k+1}e_k^T + F_k, \\ AW_{n,k} &= W_{n,k}T_k^T + \zeta_{k+1}w_{k+1}e_k^T + G_k. \end{aligned}$$

The perturbation terms are bounded by

$$\begin{aligned} |F_k| &\leq (3 + \ell)\varepsilon|A||V_{n,k}| + 4\varepsilon|V_{n,k}||T_k| + O(\varepsilon^2), \\ |G_k| &\leq (3 + \ell)\varepsilon|A|^T|W_{n,k}| + 4\varepsilon|W_{n,k}||T_k|^T + O(\varepsilon^2), \end{aligned}$$

where ℓ is the maximum number of nonzero entries in a row of A and ε is the machine precision. Under the assumption that local biorthogonality is satisfied up to

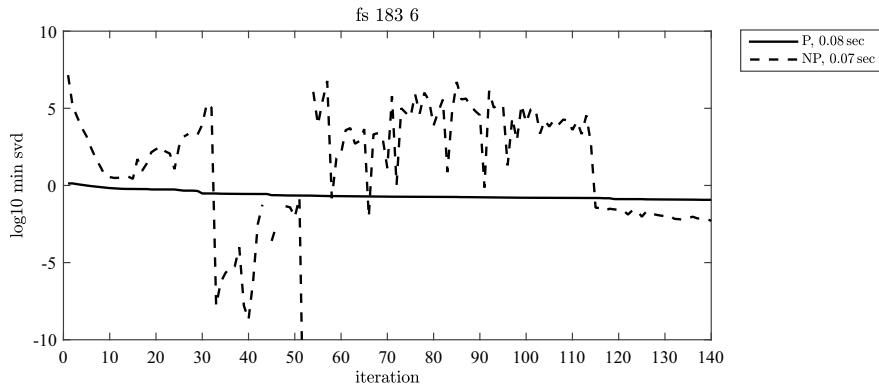


Fig. 4.12 fs 183 6, \log_{10} of minimum singular value of $V_{n,k}$, Hessenberg with pivoting (plain), without pivoting (dashed)

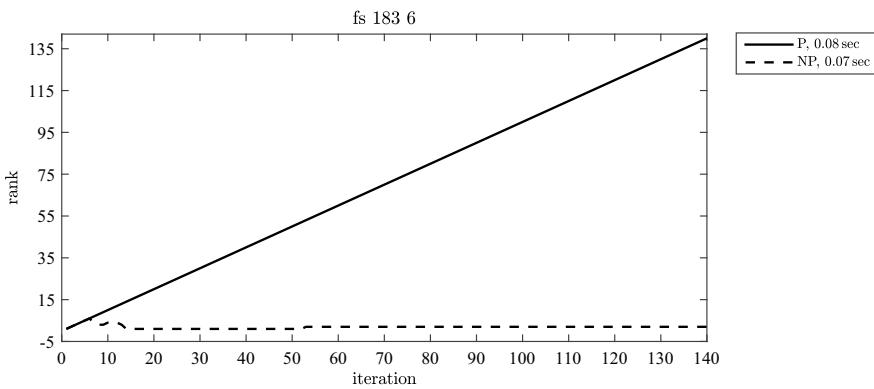


Fig. 4.13 fs 183 6, numerical rank of $V_{n,k}$, Hessenberg with pivoting (plain), without pivoting (dashed)

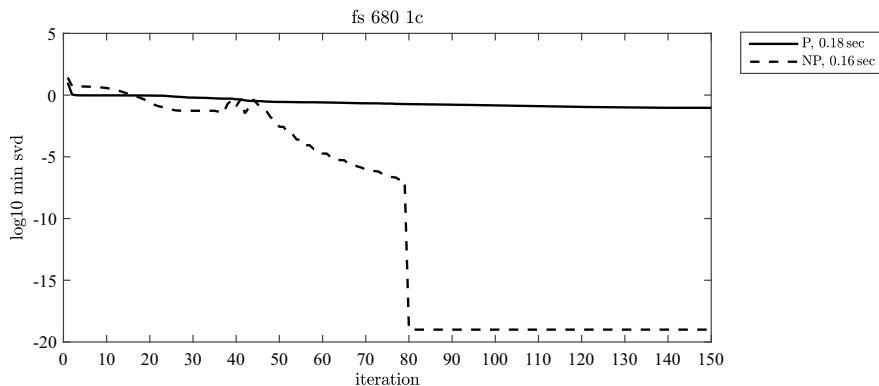


Fig. 4.14 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, Hessenberg with pivoting (plain), without pivoting (dashed)

Table 4.9 Hessenberg basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
pde225	150	P	2.22840e-01	1.01854e-01	6.17571e-02
		NP	2.49479e-03	8.79447e-04	7.47398e-05
gre 343	300	P	8.75137e-02	6.09448e-02	4.13582e-02
		NP	—	—	—
add32	200	P	3.08347e-01	2.63313e-01	2.42096e-01
		NP	1.55559e-05	5.32163e-06	3.50857e-09
jpwh 991	200	P	3.35382e-01	1.63988e-01	1.40246e-01
		NP	—	—	—
pde2961	320	P	2.37268e-01	1.82221e-01	1.55421e-01
		NP	2.54093e-01	1.92032e-01	3.43075e-02
jagmesh1	300	P	1.83640e-01	1.36888e-01	1.09117e-01
		NP	4.99729e-06	1.36883e-08	9.88263e-09
bfwa782	350	P	1.06732e-01	5.80222e-02	4.84146e-02
		NP	4.13806e-07	2.14405e-08	1.68462e-08
dw2048	1100	P	4.95667e-02	3.32355e-02	2.54709e-02
		NP	2.97240e-05	4.92081e-05	5.74488e-06
jagmesh2	800	P	7.18214e-02	3.35785e-02	2.44671e-02
		NP	—	—	—
raefsky2	450	P	1.94285e-01	1.33687e-01	9.12973e-02
		NP	9.07332e-07	7.59378e-08	2.14162e-08
fs 680 1c	150	P	2.04604e-01	1.24504e-01	9.27945e-02
		NP	—	—	—
add20	600	P	1.55200e-01	1.33089e-01	1.27907e-01
		NP	1.17078e-08	5.89796e-10	9.90852e-11
raefsky1	350	P	2.14104e-01	1.53321e-01	1.08543e-01
		NP	3.12939e-08	1.12467e-08	4.56326e-08
jagmesh4	700	P	8.41320e-02	5.68680e-02	4.57803e-02
		NP	1.88317e-07	1.66663e-07	1.12184e-07

machine precision and that no (near-)breakdown has occurred, it is shown that the loss of biorthogonality is linked to the convergence of the eigenvalues of T_k to the eigenvalues of A as it is the case for symmetric problems.

The nonsymmetric Lanczos process was also considered in [734]. It is proved that running k steps of the Lanczos process on A in the presence of perturbations is equivalent to running k steps of an exact Lanczos process on a perturbation of the block diagonal augmented matrix $\text{diag}(T_k, A)$. Contrary to what can be done for the symmetric case, the perturbation terms cannot be bounded a priori because the perturbation terms depend on $V_{n,k}$ and $W_{n,k}$ whose column norms can grow

Table 4.10 Hessenberg basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 680 1	200	P	1.26128e-01	9.32812e-02	7.91468e-02
		NP	–	–	–
sherman1	500	P	8.30463e-02	4.98510e-02	3.38501e-02
		NP	–	–	–
nos3	320	P	1.89690e-01	1.06246e-01	8.45968e-02
		NP	1.43542e-04	1.23364e-04	2.43043e-05
sherman5	1200	P	3.89632e-02	3.13096e-02	2.77782e-02
		NP	–	–	–
cavity05	600	P	8.09481e-02	4.66004e-02	3.40827e-02
		NP	–	–	–
e05r0500	260	P	9.13603e-02	5.68226e-02	5.67900e-02
		NP	1.78005e-05	1.39065e-10	1.26306e-11
comsol	300	P	2.04996e-01	1.19924e-01	9.28624e-02
		NP	–	–	–
olm1000	600	P	1.04194e-01	5.02717e-02	2.16228e-02
		NP	7.37348e-04	1.42654e-03	3.17965e-04
cavity10	900	P	8.14912e-02	5.46122e-02	3.53022e-02
		NP	–	–	–
steam2	200	P	2.10729e-01	1.94717e-01	1.92796e-01
		NP	1.03309e-01	1.28573e-02	6.34272e-08
1138bus	700	P	6.94044e-02	3.74032e-02	3.16676e-02
		NP	1.99170e-06	9.53902e-05	1.63423e-04
steam1	250	P	2.40412e-01	2.31217e-01	1.86220e-01
		NP	2.99934e-13	2.43912e-15	1.18234e-19
bcstk26	1000	P	4.37985e-02	4.29303e-02	4.18375e-02
		NP	5.63599e-04	1.28757e-03	9.12003e-05
nos7	500	P	8.98427e-02	4.35874e-02	3.62241e-02
		NP	2.09311e-14	1.99236e-16	1.78921e-17
watt1	350	P	1.69862e-01	1.58398e-01	1.47533e-01
		NP	–	–	–
bcstk14	1000	P	3.93454e-02	2.05086e-02	1.97626e-02
		NP	4.50091e-06	7.40841e-07	1.30916e-07

unboundedly. The problems come from the fact that the nonsymmetric algorithm may break down.

Let us now consider some numerical examples. We define the level of (bi)-orthogonality as

$$\|D_k - V_{n,k}^T W_{n,k}\|, \quad D_k = \text{diag}(V_{n,k}^T W_{n,k}).$$

Table 4.11 Hessenberg basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 183 6	60	P	3.02968e-01	2.34437e-01	2.04048e-01
		NP	8.76373e+03	2.54327e-04	1.21382e+02
bcsstk20	500	P	5.00671e-02	5.00616e-02	3.25388e-02
		NP	—	—	—
mcfe	820	P	6.76867e-02	3.99123e-02	3.91402e-02
		NP	—	—	—
nnc	250	P	1.25911e-01	6.98524e-02	5.72514e-02
		NP	1.93228e-13	6.68771e-14	9.20619e-15
linsp	420	P	8.86135e-02	8.74352e-02	7.64189e-02
		NP	—	—	—

The biorthogonalization algorithm with basis vectors of unit norm is denoted as *biortho* below. The variant with $(v_j, w_j) = 1$ (that is, $D_k = I$) is denoted as *biortho b* and the variant with two-term recurrences as *biortho 2t*. Figure 4.15 displays the level of biorthogonality for the matrix `fs 183 6`. We plot the level of biorthogonality as a function of the number of matrix–vector products. We see that biorthogonality is lost very rapidly.

Figure 4.16 shows that the smallest singular value of $V_{n,k}$ decreases quite fast to the machine precision level. Consequently, we see in Figure 4.17 that the basis vectors become to be linearly dependent. It may look strange that the rank is increasing after that point, but remember that what we plot is the numerical rank, that is, the number of singular values of $V_{n,k}$ that are larger than $n \epsilon \|V_{n,k}\|$. The three variants give almost the same results for this example.

Figures 4.18, 4.19 and 4.20 show that, as we can guess, things are better for the matrix `fs 680 1c`. In Figure 4.18 we observe that the level of biorthogonality is slowly increasing, the algorithm *biortho b* being worse than the two others.

Figure 4.19 shows that the smallest singular values of $V_{n,k}$ stay bounded from below. Consequently the matrices $V_{n,k}$ are of full-rank for all iterations: see Figure 4.20.

Then we would like to see if the basis vectors are linearly independent for our set of small matrices. This is shown in Tables 4.12, 4.13 and 4.14. There we define $\tilde{k}_{\max} = k_{\max}/2$ where the value of k_{\max} was given in Tables 4.1, 4.3 and 4.5 because we have two matrix–vector products per iteration. A “b” means that we had a near-breakdown during the computation. In many cases linear independence is lost very early in the computation. It seems that *biortho* is slightly better than *biortho b* but there is not much difference between the three variants on this set of examples.

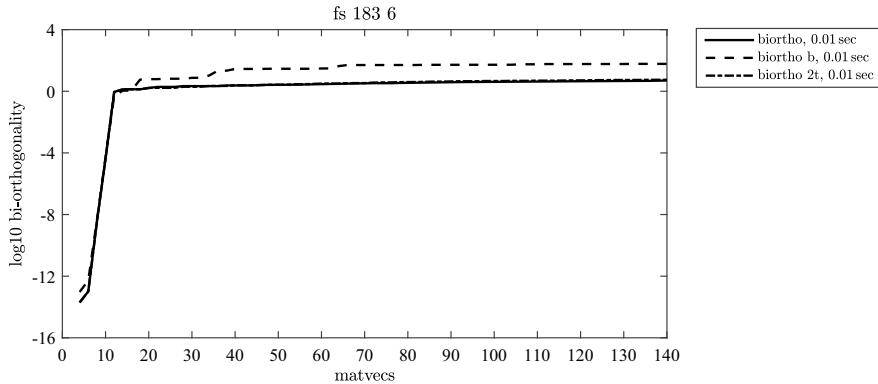


Fig. 4.15 fs 183 6, \log_{10} of the level of biorthogonality, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

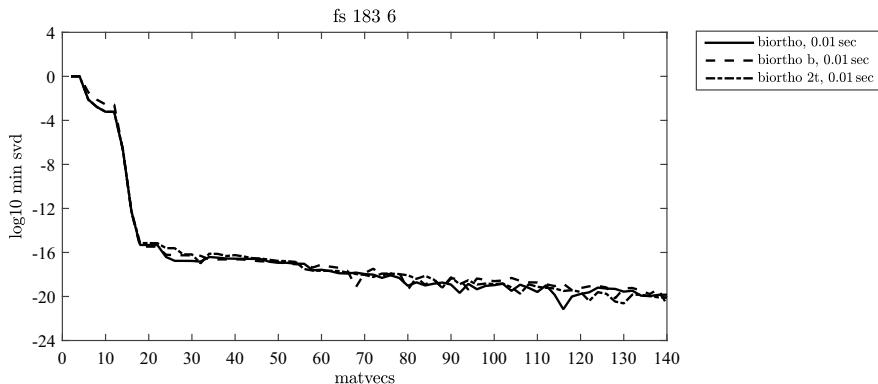


Fig. 4.16 fs 183 6, \log_{10} of minimum singular value of $V_{n,k}$, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

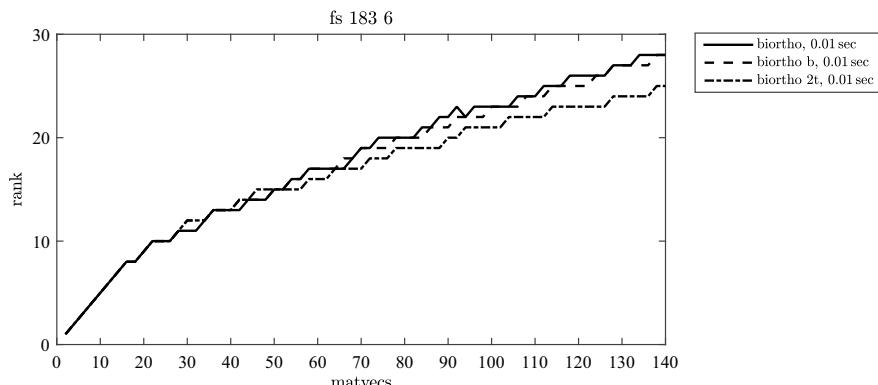


Fig. 4.17 fs 183 6, numerical rank of $V_{n,k}$, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

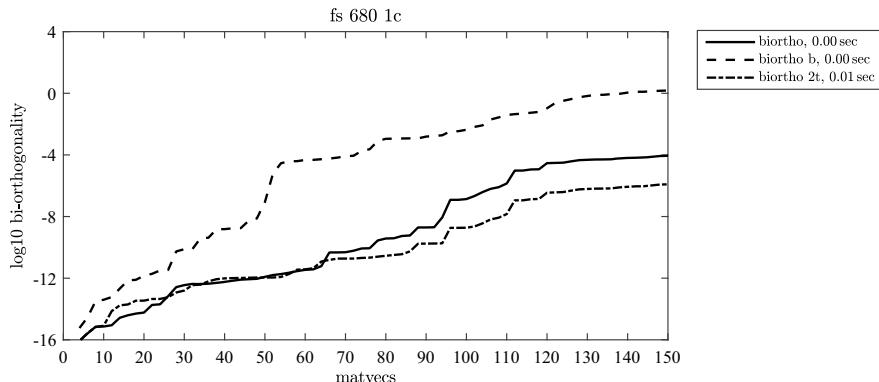


Fig. 4.18 fs 680 1c, \log_{10} of the level of biorthogonality, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

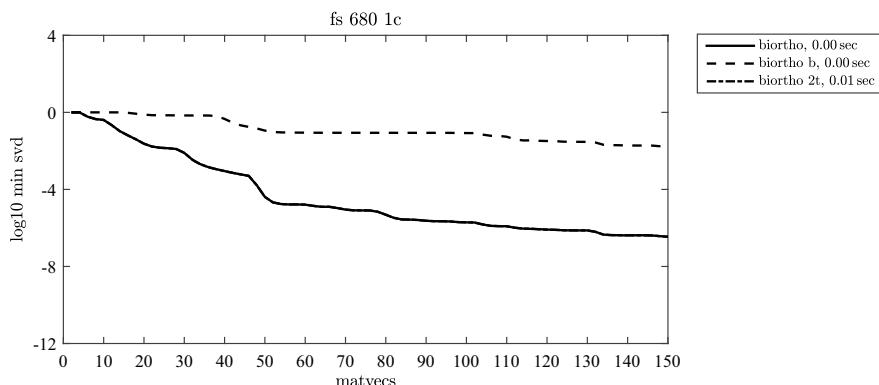


Fig. 4.19 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

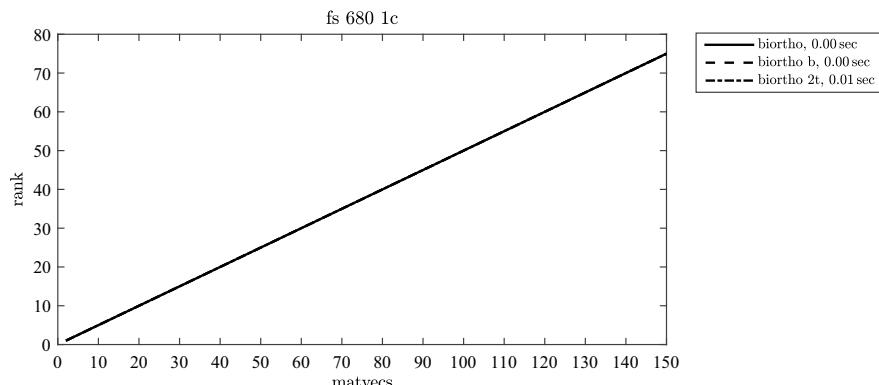


Fig. 4.20 fs 680 1c, numerical rank of $V_{n,k}$, biortho (plain), biortho bis (dashed), biortho 2t (dot-dashed)

Table 4.12 Biorthogonal basis, full-rank?

matrix	\tilde{k}_{\max}	meth	$\tilde{k}_{\max}/2$	$3\tilde{k}_{\max}/4$	\tilde{k}_{\max}
pde225	75	biortho	1	1	1
		biortho b	1	1	1
		biortho 2t	1	1	1
gre 343	150	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0
jphw 991	100	biortho	0	0	0
		biortho b	b	b	b
		biortho 2t	0	0	0
pde2961	160	biortho	1	1	0
		biortho b	1	0	0
		biortho 2t	1	1	0
jagmesh1	150	biortho	1	1	0
		biortho b	1	1	0
		biortho 2t	1	1	0
bfwa782	175	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0
dw2048	550	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
jagmesh2	400	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
raefsky2	225	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	1	0
fs 680 1c	75	biortho	1	1	1
		biortho b	1	1	1
		biortho 2t	1	1	1
add20	300	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
raefsky1	175	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0
jagmesh4	350	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0

Table 4.13 Biorthogonal basis, full-rank?

matrix	\tilde{k}_{\max}	meth	$\tilde{k}_{\max}/2$	$3\tilde{k}_{\max}/4$	\tilde{k}_{\max}
fs 680 1	100	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
sherman1	250	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
nos3	160	biortho	1	1	0
		biortho b	1	1	0
		biortho 2t	1	1	0
sherman5	600	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
cavity05	300	biortho	1	1	0
		biortho b	1	1	0
		biortho 2t	1	1	0
e05r0500	130	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0
comsol	150	biortho	1	1	1
		biortho b	1	1	1
		biortho 2t	1	1	1
olm1000	300	biortho	1	1	1
		biortho b	1	1	1
		biortho 2t	1	1	1
cavity10	450	biortho	1	1	0
		biortho b	1	1	1
		biortho 2t	1	1	1
steam2	100	biortho	1	1	1
		biortho b	1	1	1
		biortho 2t	1	1	1
1138bus	350	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
steam1	125	biortho	1	0	0
		biortho b	1	0	0
		biortho 2t	1	0	0
bcsstk26	500	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0

(continued)

Table 4.13 (continued)

matrix	\tilde{k}_{\max}	meth	$\tilde{k}_{\max}/2$	$3\tilde{k}_{\max}/4$	\tilde{k}_{\max}
nos7	250	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
watt1	175	biortho	1	1	0
		biortho b	b	b	b
		biortho 2t	1	0	0
bcsstk14	500	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0

Table 4.14 Biorthogonal basis, full-rank?

matrix	\tilde{k}_{\max}	meth	$\tilde{k}_{\max}/2$	$3\tilde{k}_{\max}/4$	\tilde{k}_{\max}
fs 183 6	30	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
bcsstk20	250	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
mcfe	410	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
nnc	125	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0
lisp	210	biortho	0	0	0
		biortho b	0	0	0
		biortho 2t	0	0	0

4.9.4 Q-OR opt basis

So far there is no theoretical study of the Q-OR opt algorithms in finite precision arithmetic. Hence, we have to rely only on numerical experiments.

The Q-OR construction of a basis does not work well for the matrix `fs 183 6`. As one can see in Figure 4.21 the minimum singular value decreases quite fast before iteration 45 and the matrices $V_{n,k}$ loose linear independence; see Figure 4.22. However, it turns out that this problem can be solved by scaling the matrix to have a unit diagonal. The variant Q-OR t using the tridiagonal matrix gives worse results than Q-OR, linear independence is lost sooner.

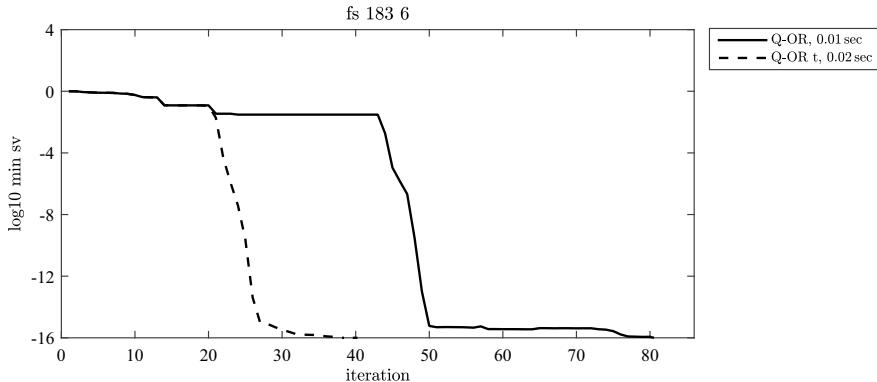


Fig. 4.21 fs 183 6, \log_{10} of minimum singular value of $V_{n,k}$, Q-OR (plain), Q-OR t (dashed)

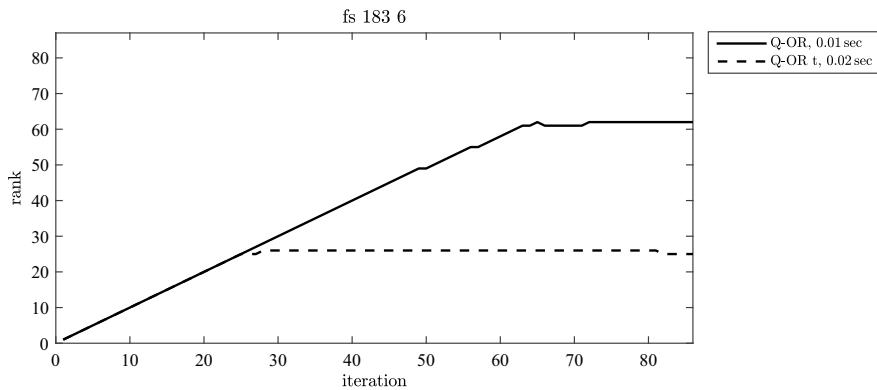


Fig. 4.22 fs 183 6, numerical rank of $V_{n,k}$, Q-OR (plain), Q-OR t (dashed)

Nevertheless things are far from being always so bad. In Figure 4.23 we see that for the matrix `fs 680 1c` and Q-OR the minimum singular value stays almost constant after a small decrease at the beginning of the iterations. For Q-OR the minimum singular value starts to decrease after iteration 90. For Q-OR all the matrices $V_{n,k}$ are of full-rank; see Figure 4.24.

Figure 4.25 shows the evolution of the smallest singular value of $V_{n,k}$ and the lower bound that was derived in Theorem 4.4. For this example, we obtain a very sharp bound. Note that the bound is cheap to compute from the coefficients $\vartheta_{1,j}$ and α_j .

Tables 4.15, 4.16 and 4.17 display the minimum singular values for our set of small matrices. In most cases the minimum singular values for Q-OR stay almost constant up to the end of the computation, and the matrices $V_{n,k}$ are of full-rank. A noticeable counterexample is the matrix `watt1` for which the three singular values are close to the machine precision.

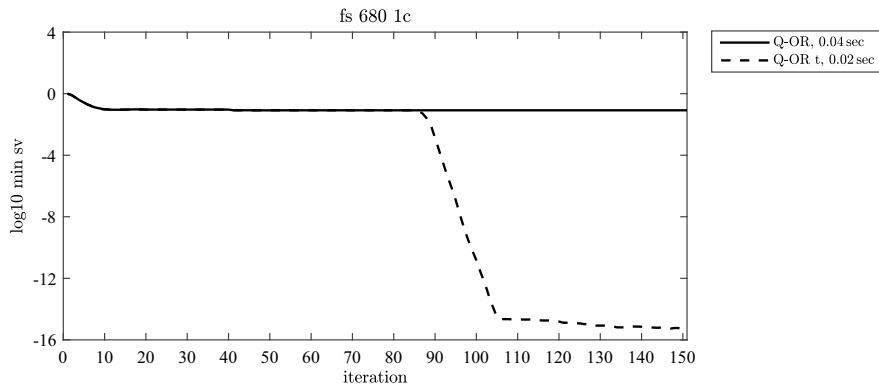


Fig. 4.23 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, Q-OR (plain), Q-OR t (dashed)

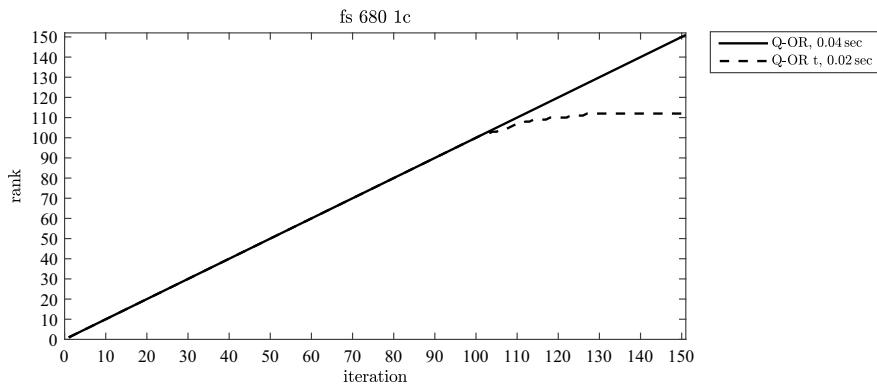


Fig. 4.24 fs 680 1c, numerical rank of $V_{n,k}$, Q-OR (plain), Q-OR t (dashed)

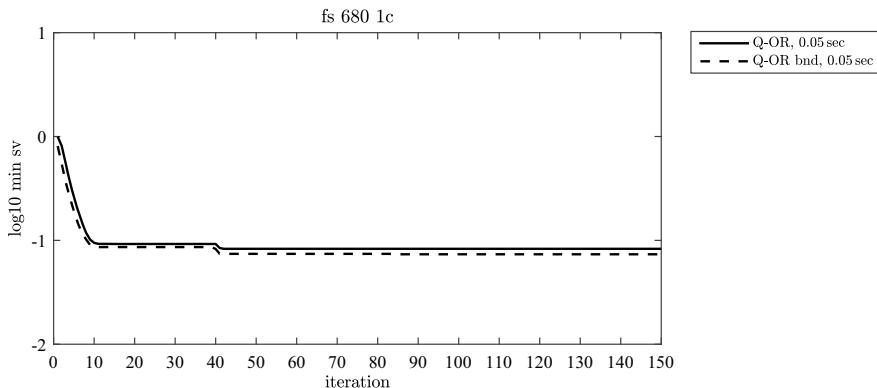


Fig. 4.25 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$ (plain) and lower bound (dashed)

Table 4.15 Q-OR basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
pde225	150	Q-OR	1.55883e-01	1.55883e-01	1.55883e-01
		Q-OR t	1.55883e-01	1.08871e-13	1.09989e-16
gre 343	300	Q-OR	1.24815e-03	1.24815e-03	8.99862e-05
		Q-OR t	b	b	b
jpw 991	200	Q-OR	1.95489e-01	1.95489e-01	1.95489e-01
		Q-OR t	9.34034e-06	3.21767e-16	2.67627e-16
pde2961	320	Q-OR	8.02894e-02	8.02894e-02	8.02894e-02
		Q-OR t	8.02894e-02	8.02894e-02	2.46649e-16
jagmesh1	300	Q-OR	7.44718e-04	7.44718e-04	7.44718e-04
		Q-OR t	5.58896e-04	2.28406e-16	1.53662e-16
bfwa782	350	Q-OR	1.22772e-03	1.22772e-03	1.22772e-03
		Q-OR t	b	b	b
dw2048	1100	Q-OR	9.85321e-04	4.87273e-04	4.87273e-04
		Q-OR t	b	b	b
jagmesh2	800	Q-OR	1.24481e-03	1.24481e-03	1.24481e-03
		Q-OR t	b	b	b
raefsky2	450	Q-OR	3.41664e-02	3.41664e-02	3.41664e-02
		Q-OR t	3.60599e-16	2.82953e-16	2.57456e-16
fs 680 1c	150	Q-OR	8.29405e-02	8.29405e-02	8.29405e-02
		Q-OR t	8.29405e-02	2.10465e-15	5.92565e-16
add20	600	Q-OR	3.15505e-02	3.15505e-02	3.15505e-02
		Q-OR t	6.72326e-13	1.62587e-16	8.40083e-17
raefsky1	350	Q-OR	3.03776e-02	3.03776e-02	3.03776e-02
		Q-OR t	3.56975e-16	3.96356e-16	3.66755e-16
jagmesh4	700	Q-OR	2.36739e-04	2.36739e-04	2.36739e-04
		Q-OR t	2.39015e-16	b	b

Unfortunately, the variant Q-OR t, which is cheaper, is much less reliable. There are decreases of the smallest singular values and losses of linear independence and even some breakdowns (denoted by “b”). We will see in Chapter 11 that this method can be used anyway with restart when things go wrong.

Table 4.18 displays the minimum singular values for our set of larger matrices. Again the minimum singular values for Q-OR stay almost constant up to the end of the computation, and the matrices $V_{n,k}$ are of full-rank. The variant Q-OR t is less reliable.

Table 4.16 Q-OR basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 680 1	200	Q-OR	1.53173e-01	1.53173e-01	1.53173e-01
		Q-OR t	1.53168e-01	4.08576e-16	2.28053e-16
sherman1	500	Q-OR	6.66388e-02	6.66388e-02	6.66388e-02
		Q-OR t	6.66388e-02	6.66388e-02	2.23228e-16
nos3	320	Q-OR	3.25842e-02	3.25842e-02	3.25842e-02
		Q-OR t	3.25842e-02	3.25842e-02	1.25284e-15
sherman5	1200	Q-OR	1.43999e-04	1.43999e-04	1.43999e-04
		Q-OR t	b	b	b
cavity05	600	Q-OR	4.71853e-03	4.71853e-03	4.71853e-03
		Q-OR t	1.82461e-16	1.42636e-16	1.17619e-16
e05r0500	260	Q-OR	4.70464e-04	3.79337e-05	3.79331e-05
		Q-OR t	b	b	b
comsol	300	Q-OR	7.78147e-04	7.78147e-04	7.78147e-04
		Q-OR t	1.63886e-16	1.16455e-16	9.76493e-17
olm1000	600	Q-OR	2.71568e-04	2.71568e-04	2.71568e-04
		Q-OR t	1.96225e-16	1.41080e-16	9.59407e-17
cavity10	900	Q-OR	3.17021e-03	3.17021e-03	3.17021e-03
		Q-OR t	1.44120e-16	1.59093e-16	1.16448e-16
steam2	200	Q-OR	2.73170e-04	2.73170e-04	2.73170e-04
		Q-OR t	1.07339e-16	6.69418e-17	5.36375e-17
1138bus	700	Q-OR	2.41130e-02	2.41130e-02	2.41130e-02
		Q-OR t	2.41130e-02	2.41130e-02	2.26715e-16
steam1	250	Q-OR	2.03455e-03	2.03455e-03	2.03455e-03
		Q-OR t	4.25634e-11	1.03441e-17	3.37502e-20
bcstk26	1000	Q-OR	4.79697e-02	4.59041e-02	4.59041e-02
		Q-OR t	4.79697e-02	4.59041e-02	4.59041e-02
nos7	500	Q-OR	5.40056e-04	5.40056e-04	5.40056e-04
		Q-OR t	1.72997e-16	1.36280e-16	7.63043e-17
watt1	350	Q-OR	4.20973e-15	3.15187e-16	2.97811e-16
		Q-OR t	1.41027e-16	1.50820e-16	1.40416e-16
bcstk14	1000	Q-OR	1.42173e-02	1.00896e-02	1.71194e-03
		Q-OR t	1.42173e-02	1.00896e-02	1.71194e-03

4.9.5 Newton and Chebyshev bases

We consider the Newton basis computed with the spoke sets proposed in [756] (Newt RS) as well as the basis computed using Chebyshev polynomials (that we inappropriately name Newt C). We first do 10 iterations of the Arnoldi process to obtain the Ritz values needed to compute the spoke sets and the best ellipse enclosing these Ritz values. Then, we compute the Newton or Chebyshev bases. We note that

Table 4.17 Q-OR basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 183 6	60	Q-OR	3.04675e-02	1.09905e-05	3.64184e-16
		Q-OR t	3.32576e-16	8.52280e-17	4.97508e-17
bcsstk20	500	Q-OR	4.41265e-03	4.41265e-03	4.41265e-03
		Q-OR t	4.41253e-03	7.76738e-17	3.24529e-20
mcfe	820	Q-OR	6.40895e-04	1.94158e-16	5.33835e-18
		Q-OR t	6.91592e-17	4.83322e-17	7.54380e-21
nnc	250	Q-OR	8.11927e-04	8.11927e-04	1.29464e-17
		Q-OR t	3.94692e-17	1.95496e-17	b
linsp 511	420	Q-OR	3.47128e-16	1.39180e-16	9.22760e-17
		Q-OR t	8.61530e-17	5.53993e-17	b

the Newton basis was considered to increase the parallelism for restarted Krylov methods. Therefore, we do not hope to obtain an independent set of basis vectors for a large number of iterations.

Figures 4.26 and 4.27 show the decrease of the minimum singular values for the Newton and Chebyshev bases compared to what is obtained with the Arnoldi process. For `fs 183 6` the results are not good since the decrease in the minimum singular value is very fast. For `fs 680 1c` the decrease starts early but it is not too fast and the basis vectors are linearly independent until iteration 60.

Tables 4.19, 4.20 and 4.21 give the numerical rank that is obtained with this technique. We observe that in many cases we obtain linearly independent basis vectors for several tens of iterations. It means that the Newton basis vectors (at least with these choices of shifts) can be used in restarted methods. There are a few exceptions like `watt1` and `fs 183 6` for which linear independence is lost very quickly. The results are approximately the same for the two methods. However, for a large number of iterations, the cost of the computation of the spoke sets is higher than the cost of the computation of the Chebyshev polynomials.

4.9.6 Truncated Q-OR basis

We consider the bases obtained by truncation as described in Section 4.7. We choose two positive integers p and q . The basis is constructed by using the columns of $V_{n,k-q+1:k}$ and those of $AV_{n,k-p+1:k}$ to compute v_{k+1} .

Figure 4.28 displays the \log_{10} of the smallest singular values of $V_{n,k}$ for different couples (p, q) and the matrix `fs 680 1c`. Truncating with $p = 1$ is not very efficient as shown by the results for $(1, 10)$, $(1, 20)$ and $(1, 30)$. Using the same number of vectors, it is more efficient to use $(10, 10)$ than $(1, 20)$, $(15, 15)$ than $(1, 30)$ and $(20, 20)$ than $(1, 40)$. When $p > 1$ the decrease of the smallest singular values starts

Table 4.18 Q-OR basis, min svd

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
add32	200	Q-OR	2.17378e-01	2.17378e-01	2.17378e-01
		Q-OR t	2.17378e-01	5.42966e-16	3.13413e-16
ex37	150	Q-OR	2.14998e-01	2.14998e-01	2.14998e-01
		Q-OR t	2.14998e-01	2.32505e-03	6.00217e-16
memplus	900	Q-OR	1.57823e-02	1.57823e-02	1.57823e-02
		Q-OR t	3.41510e-16	2.75202e-16	3.92356e-16
sherman3	800	Q-OR	3.28456e-02	3.28456e-02	3.28456e-02
		Q-OR t	3.28456e-02	3.28456e-02	2.63877e-16
wang4	700	Q-OR	1.80524e-02	1.80524e-02	1.80524e-02
		Q-OR t	1.02060e-16	9.35450e-17	7.01411e-17
supg 100	1000	Q-OR	8.94879e-02	8.94879e-02	8.94879e-02
		Q-OR t	8.94879e-02	8.94879e-02	2.25903e-16
supg 1	1100	Q-OR	7.89416e-02	7.89416e-02	7.89416e-02
		Q-OR t	7.89416e-02	7.89416e-02	4.00352e-16
supg 01	500	Q-OR	6.89313e-02	6.89313e-02	6.89313e-02
		Q-OR t	6.89313e-02	6.89301e-02	3.44400e-16
supg 001	150	Q-OR	1.13489e-01	1.13489e-01	1.13489e-01
		Q-OR t	1.13489e-01	1.13489e-01	4.01641e-13
supg 0001	120	Q-OR	1.18052e-01	1.18052e-01	1.18052e-01
		Q-OR t	1.18052e-01	1.18052e-01	1.72482e-08
supg 00001	220	Q-OR	9.89371e-02	9.89371e-02	9.89371e-02
		Q-OR t	9.89371e-02	9.89371e-02	4.23142e-13
supg 000001	240	Q-OR	6.05181e-02	6.05181e-02	6.05181e-02
		Q-OR t	6.05181e-02	6.05181e-02	1.82545e-16
convdiff xu 500	1100	Q-OR	4.94595e-06	4.94475e-06	4.94475e-06
		Q-OR t	b	b	b
convdiff xu 1000	700	Q-OR	2.50652e-06	2.50648e-06	2.50577e-06
		Q-OR t	1.93158e-16	1.93554e-16	1.68894e-16
convdiff xu 5000	400	Q-OR	3.26141e-04	3.26141e-04	3.26141e-04
		Q-OR t	b	b	b

later. Moreover, as shown from Figure 4.29 the numerical rank of $V_{n,k}$ is larger. The plain curve corresponds to the full Q-OR opt basis.

The results are even better for the matrix supg 001 as one can see in Figures 4.30 and 4.31. Using $p > 1$ the smallest singular values are bounded from below and the matrices $V_{n,k}$ are of full numerical rank whence linear independence is lost when truncating with $p = 1$.

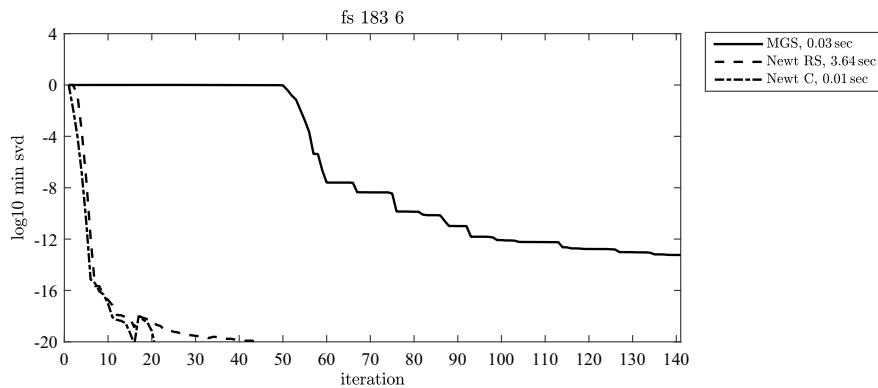


Fig. 4.26 fs 183 6, \log_{10} of minimum singular value of $V_{n,k}$, Arnoldi (plain), Newton RS (dashed), Chebyshev (dot-dashed)

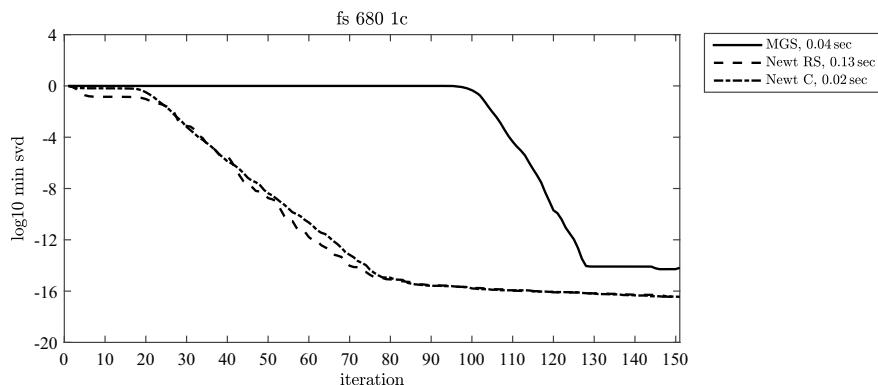


Fig. 4.27 fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, Arnoldi (plain), Newton RS (dashed), Chebyshev (dot-dashed)

Table 4.19 Newton and Chebyshev bases, numerical rank

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
pde225	150	Newt RS	67	75	74
		Cheby	70	76	76
gre 343	300	Newt RS	55	55	54
		Cheby	56	56	55
jpw 991	200	Newt RS	42	42	42
		Cheby	42	42	42
pde2961	320	Newt RS	128	134	136
		Cheby	106	140	146
jagmesh1	300	Newt RS	82	82	82
		Cheby	79	79	79
bfwa782	350	Newt RS	126	128	126
		Cheby	124	126	125
dw2048	1100	Newt RS	56	56	56
		Cheby	55	55	1
jagmesh2	800	Newt RS	107	107	107
		Cheby	104	104	104
raefsky2	450	Newt RS	99	103	105
		Cheby	127	131	133
fs 680 1c	150	Newt RS	67	68	68
		Cheby	68	70	71
add20	600	Newt RS	72	73	73
		Cheby	71	72	72
raefsky1	350	Newt RS	98	113	124
		Cheby	97	112	123
jagmesh4	700	Newt RS	103	103	104
		Cheby	100	100	100

Table 4.20 Newton and Chebyshev bases, numerical rank

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 680 1	200	Newt RS	39	42	45
		Cheby	33	38	41
sherman1	500	Newt RS	106	106	107
		Cheby	104	103	103
nos3	320	Newt RS	85	85	84
		Cheby	85	85	83
sherman5	1200	Newt RS	85	85	83
		Cheby	78	78	76
cavity05	600	Newt RS	114	115	116
		Cheby	112	114	115
e05r0500	260	Newt RS	80	95	100
		Cheby	79	91	95
comsol	300	Newt RS	109	109	109
		Cheby	110	110	110
olm1000	600	Newt RS	136	141	141
		Cheby	135	138	140
cavity10	900	Newt RS	127	128	129
		Cheby	125	126	127
steam2	200	Newt RS	51	64	69
		Cheby	51	62	68
1138bus	700	Newt RS	102	106	108
		Cheby	100	103	104
steam1	250	Newt RS	75	83	88
		Cheby	75	82	87
bcstk26	1000	Newt RS	76	79	81
		Cheby	75	78	80
nos7	500	Newt RS	23	22	22
		Cheby	22	22	21
watt1	350	Newt RS	7	7	8
		Cheby	5	6	6
bcstk14	1000	Newt RS	215	218	220
		Cheby	210	213	215

Table 4.21 Newton and Chebyshev bases, numerical rank

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
fs 183 6	60	Newt RS	7	7	8
		Cheby	6	7	7
bcsstk20	500	Newt RS	41	48	54
		Cheby	39	47	53
mcfe 765	820	Newt RS	70	80	87
		Cheby	70	80	87
nnc 261	250	Newt RS	68	70	70
		Cheby	65	65	65
linsp 511	420	Newt RS	47	46	46
		Cheby	49	48	48

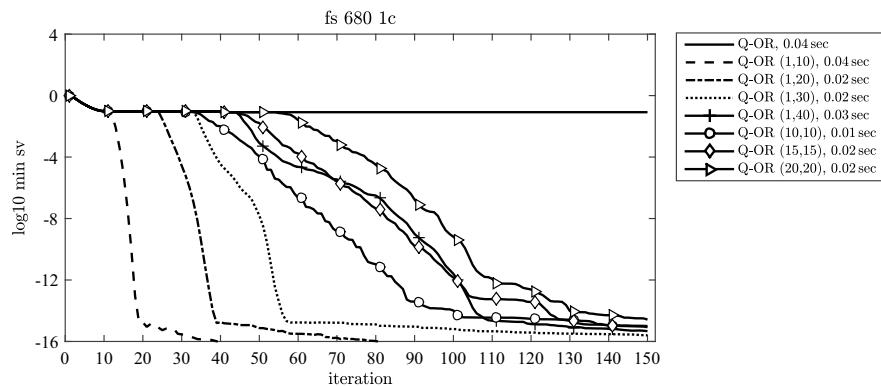
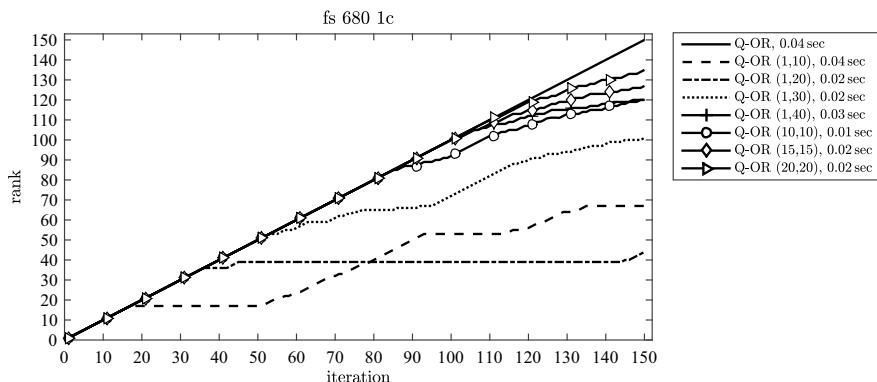
**Fig. 4.28** fs 680 1c, \log_{10} of minimum singular value of $V_{n,k}$, different values of (p, q) **Fig. 4.29** fs 680 1c, numerical rank of $V_{n,k}$, different values of (p, q)

Table 4.22 Newton and Chebyshev bases, numerical rank

matrix	k_{\max}	meth	$k_{\max}/2$	$3k_{\max}/4$	k_{\max}
add32	200	Newt RS	81	86	88
		Cheby	81	84	86
ex37	150	Newt RS	61	77	79
		Cheby	61	76	78
memplus	900	Newt RS	159	196	208
		Cheby	157	188	194
sherman3	800	Newt RS	95	118	139
		Cheby	94	118	138
wang4	700	Newt RS	122	125	126
		Cheby	121	123	125
supg 100	1000	Newt RS	126	129	131
		Cheby	124	127	129
supg 1	1100	Newt RS	131	135	139
		Cheby	130	134	137
supg 01	500	Newt RS	147	161	165
		Cheby	146	160	163
supg 001	150	Newt RS	75	111	122
		Cheby	75	106	115
supg 0001	120	Newt RS	60	85	90
		Cheby	60	83	93
supg 00001	220	Newt RS	94	97	97
		Cheby	98	109	118
supg 000001	240	Newt RS	95	97	97
		Cheby	99	110	118
convdiff xu 500	1100	Newt RS	156	156	156
		Cheby	147	147	142
convdiff xu 1000	700	Newt RS	128	127	127
		Cheby	130	128	128
convdiff xu 5000	400	Newt RS	105	131	132
		Cheby	92	102	100

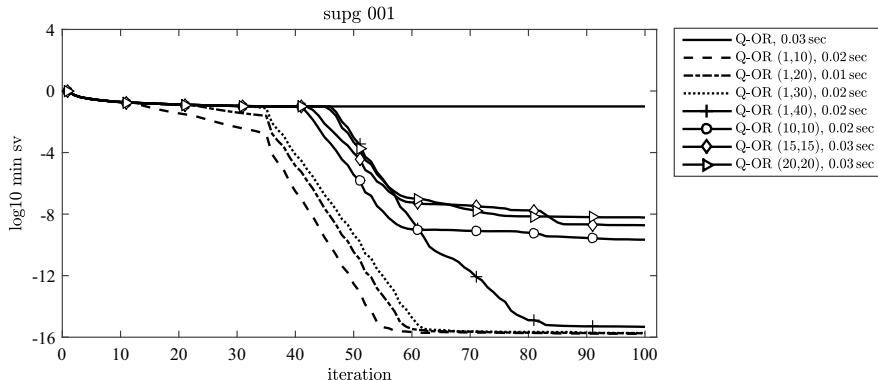


Fig. 4.30 supg 001, \log_{10} of minimum singular value of $V_{n,k}$, different values of (p, q)

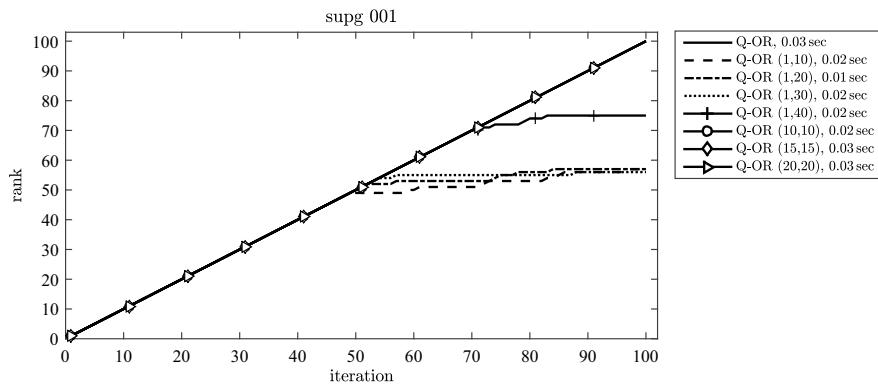


Fig. 4.31 supg 001, numerical rank of $V_{n,k}$, different values of (p, q)

4.10 Historical notes

The methods for computing an orthonormal basis of a Krylov subspace originated in the techniques used to orthogonalize a given set of vectors or functions. This goes back to the 19th century. Pierre Simon de Laplace (1749–1827) used orthogonalization in a disguised form in a paper about a problem in Astronomy in 1820; see the translation of this paper by Julien Langou [619]. Laplace wanted to estimate the masses of Jupiter and Saturn from astronomical data of six planets. He used what is now known as the modified Gram–Schmidt process. Augustin-Louis Cauchy (1789–1857) also used related ideas. The modern history starts with the works of Jorgen Pedersen Gram (1850–1916) a Danish actuary and mathematician and Erhard Schmidt (1876–1959) a German mathematician. Gram published a paper in 1883 where he orthogonalized a set of linearly independent vectors. This paper was cited in Schmidt's works in 1905 and 1907. Schmidt orthogonalized a set of

functions with respect to a dot product defined by a Riemann integral. He used what is now known as the classical Gram–Schmidt process. For an early exposition of the Gram–Schmidt algorithm for least squares problems, see the paper [984] by Y.K. Wong in 1935.

The paper [35] describing the Arnoldi process was received in May 1950. Walter Edwin Arnoldi (1917–1995) was an American engineer. As one can see in “The annals of the computation laboratory of Harvard University, Proceedings of a second symposium on large-scale digital calculating machinery” organized on September 13–16, 1949 by Howard H. Aiken, Arnoldi attended this symposium where Cornelius Lanczos gave a talk whose title is “An iteration method for the solution of the eigenvalue problem of linear differential and integral operators”. Arnoldi was registered as working for Hamilton Standard Propellers, Wethersfield (Connecticut). Lanczos’ paper in these proceedings (pages 164–206) is very similar to his paper [615] published in October 1950 but written in September 1949. Arnoldi’s paper starts with recalling Lanczos biorthogonalization algorithm but this is written in a more matrix-oriented way than in Lanczos’ paper, even though he used the strange notation u for the matrix. Arnoldi described the tridiagonal structure of the resulting matrix in Lanczos’ algorithm. It is not before Section 6 that what we call today the Arnoldi process is described, although the basis vectors are not normalized. The upper Hessenberg structure of the matrix is shown in relation (23). The algorithm is fully stated in Section 8. Curiously it is written in the conclusion that “The Galerkin variation appears to be of greatest advantage in the eigenvalue problem, requiring substantially only half the number of matrix–column products involved in the Lanczos procedure” since there is only one matrix–vector product per iteration. There is no comment on the fact that the method uses long recurrences and needs the storage of all basis vectors.

There were not many references to Arnoldi’s paper in the 1960s and 70s, a few tenths, even though it was cited and commented in James H. Wilkinson’s book [982] in 1965, pages 382–385. Wilkinson wrote negative comments about the loss of orthogonality in a 2×2 example. However, when run on this example in IEEE double precision, the Arnoldi algorithm delivers the eigenvalues with full accuracy. There were hundreds of references to Arnoldi’s paper in the next twenty years and tens of thousands of references since 1990.

Probably the first paper in the literature on orthonormalizing a set of vectors using a computer is by Philip J. Davis and Philip Rabinowitz [255]. This paper used CGS. An Algol implementation named ORTHO by Philip J. Walsh [971] in 1962 included reorthogonalization and was much used.

The normwise backward stability of Householder reflections to compute a QR factorization was proved by J.H. Wilkinson in 1965.

In 1966 John R. Rice [769] showed experimentally that the two different versions of the Gram–Schmidt orthogonalization, CGS and MGS, have very different properties when executed in finite precision arithmetic.

In 1967 Åke Björck [98] gave error bounds for the computed orthogonal factor in MGS. In 1968 he published two Algol subroutines for the solution of linear least squares problems based on MGS.

Heinz Rutishauser [787] introduced a criterion for selective reorthogonalization of the basis vectors computed by the Gram–Schmidt algorithms in 1967.

In 1968 Charles Sheffield (1935–2002) made the surprising observation that the MGS QR factorization of A is equivalent to the Householder QR algorithm applied to A with a square matrix of zeros on top. The equivalence holds also in finite precision arithmetic. In 1992 Å. Björck and Christopher C. Paige used this equivalence to derive backward stable MGS least squares algorithms.

Walter Hoffmann considered iterative algorithms for Gram–Schmidt orthogonalization in [530].

Interesting papers about Gram–Schmidt orthogonalization are those of Axel Ruhe about the numerical aspects of Gram–Schmidt orthogonalization [781] and Lyle Pursell and S.Y. Trimble on using Gaussian elimination to do Gram–Schmidt orthogonalization [759].

The potential loss of orthogonality of the Gram–Schmidt algorithms in finite precision arithmetic has been much studied; see Å. Björck and C.C. Paige [103] and Å. Björck [99]. If A has full numerical column rank, then one reorthogonalization step suffices for CGS and MGS to achieve orthogonality to roundoff levels. Therefore, Gram–Schmidt with reorthogonalization is frequently called the “twice-is-enough” algorithm. This term was coined by William Kahan in connection with orthogonalizing a Krylov sequence; see Beresford N. Parlett’s book [751].

These issues have been studied extensively by Luc Giraud, J. Langou and some collaborators in a series of papers by L. Giraud and J. Langou [419–421], L. Giraud, J. Langou and Miroslav Rozložník [423, 424], L. Giraud, J. Langou, M. Rozložník and Jesper van den Eshof [425]. These results were slightly improved in 2006 in the context of GMRES by C.C. Paige, M. Rozložník and Zdeněk Strakoš [736] as we will see in Chapter 5.

Gram–Schmidt orthogonalization is discussed in the books by Å. Björck [100], Gene H. Golub and Charles Van Loan [438] and Gilbert W. Stewart [880]. An interesting paper summarizing the variants of Gram–Schmidt algorithms is by G.W. Stewart [884]. For an history of Gram–Schmidt orthogonalization, see [627].

The construction of the Hessenberg basis originated in the work of Karl Hessenberg [512]; see the historical note in Chapter 2. This basis was considered by J.H. Wilkinson [982] page 377 as a particular case of the generalized Hessenberg process (as well as the Arnoldi process). It was used by Hassane Sadok to derive the CMRH method [805]; see Chapter 7.

The biorthogonal basis was introduced by C. Lanczos (1893–1974) in 1950 [615]; see also [616]. For a good introduction to these methods, see Martin H. Gutknecht [482]. Petr Tichý considered the choice of the shadow vector in the Lanczos biorthogonalization method [910]. David M. Day studied how to maintain biorthogonality in the nonsymmetric Lanczos algorithm, see [258, 259]. The biorthogonal basis was used by Roland W. Freund and Noël M. Nachtigal to derive the QMR iterative method, see [379, 382, 383].

The Q-OR optimal basis was introduced in [685].

Newton bases were considered by Zhaojun Bai, Dan T. Hu and Lothar Reichel in 1992 to introduce more parallelism into the Arnoldi process, see [58, 59, 173].

The difficult problem with this approach is the choice of the shifts to obtain a well-conditioned basis. One possibility is to use sets of points obtained from approximate eigenvalues and to order them as (fast) Leja points; for their computation, see James Baglama, Daniela Calvetti and L. Reichel [51]. Another issue is to have a parallel algorithm for the QR factorization of the basis vectors. These problems were also addressed by Jocelyne Erhel [321, 962] as well as Roger B. Sidje [824]. A more recent work on Newton bases is by Bernard Philippe and L. Reichel [756].

A way to compute shifts is to consider a region in the complex plane enclosing the eigenvalues of H_m (the Ritz values) for a moderate value of m . One such region is the smallest ellipse containing the Ritz values. For comments on the problems that may happen with this choice, see N.M. Nachtigal, L. Reichel and Lloyd N. Trefethen [712], Section 2 page 800.

An early reference about truncating long recurrences for solving linear systems is Paul K.W. Vinsome [953] in 1976. Truncating the Arnoldi recurrence relation was considered by Y. Saad in 1980 in the incomplete orthogonalization method [788] Section 3.2 for computing approximations of the eigenvalues of A . The same basis was used for the approximation of the solution of a linear system in 1981 [789]. The properties of the basis obtained by truncation were studied by Zhongxiao Jia in 1996 [565]. A strategy was suggested to choose the number of vectors that are kept in the recurrence.

In [835] Valeria Simoncini and Daniel B. Szyld called the non-orthogonal bases of the Krylov subspaces “non-optimal”. They studied how non-orthogonality influences the Q-OR and Q-MR methods based on these bases.

Concerning parallelism, the use of s -step Krylov methods for parallel computers started at the end of the 1980s with the conjugate gradient method for symmetric positive definite systems; see, for instance, [224] by Anthony T. Chronopoulos and Charles W. Gear; see also [222]. This was extended to the nonsymmetric case by Sun Kyung Kim and A.T. Chronopoulos [596, 597] in 1991–1992. The study of communication-avoiding algorithms started at the beginning of the 21st century; see [265] by Jim Demmel, Laura Grigori, Mark Hoemmen and J. Langou in 2012. The CA-Lanczos algorithm was described in the Ph.D. thesis of Erin Carson [188] in 2015.

Chapter 5

FOM/GMRES and variants



5.1 Introduction to FOM and GMRES

FOM (Full Orthogonalization Method) and GMRES (Generalized Minimal RESidual) is a pair of Q-OR/Q-MR methods using an orthonormal basis for the Krylov subspaces. In fact, as we have seen in Chapter 3, the FOM residual vectors are proportional to the basis vectors. Thus, FOM is a Q-OR method for which the residual vectors are orthogonal to each other. It is a true OR (Orthogonal Residual) method. In GMRES, since the basis is orthonormal, the norm of the quasi-residual is equal to the norm of the residual. Therefore GMRES is a true MR (Minimum Residual) method. The minimization of the norm of the residual implies that in GMRES the residual norms are decreasing which is a most wanted property for iterative methods. However, as we will see, the GMRES norms may stagnate.

Regarding minimum residual methods it is interesting to remark the following property.

Theorem 5.1 *The norm of the residual vector r_k is minimized if and only if r_k is orthogonal to $A\mathcal{K}_k(A, r_0)$.*

Proof We can write

$$\begin{aligned}\min_{x_k \in x_0 + \mathcal{K}_k(A, r_0)} \|r_k\| &= \min_{x_k \in x_0 + \mathcal{K}_k(A, r_0)} \|b - Ax_k\|, \\ &= \min_{y \in \mathcal{K}_k(A, r_0)} \|r_0 - Ay\|, \\ &= \min_{w \in A\mathcal{K}_k(A, r_0)} \|r_0 - w\|.\end{aligned}$$

The solution to the last minimization problem is given by the orthogonal projection of r_0 onto $A\mathcal{K}_k(A, r_0)$. Let P_k be this orthogonal projector. Then, the solution is $w = P_k r_0$ and $r_k = (I - P_k)r_0$. But $I - P_k$ is the orthogonal projector onto the orthogonal complement of $A\mathcal{K}_k(A, r_0)$. Hence, $r_k \perp A\mathcal{K}_k(A, r_0)$. \square

This result, which holds for GMRES, is used to construct the iterates and the residuals in some methods equivalent to GMRES in the sense that, mathematically, they produce the same residual norms as GMRES. We will study these methods in Chapter 6.

In FOM and GMRES we construct the orthonormal basis vectors v_k as well as the upper Hessenberg matrices H_k and \underline{H}_k . The construction of the basis is done by one of the variants of the Arnoldi process; see Chapter 4. This yields several variants of FOM and GMRES. For instance, we have GMRES-CGS using the classical Gram–Schmidt orthonormalization, GMRES-MGS using the modified Gram–Schmidt algorithm and GMRES-H using Householder reflections to orthogonalize the basis vectors [964, 965].

Once we have basis vectors, as we have seen in equations (3.6) and (3.8), we need to solve problems involving H_k or \underline{H}_k . This could lead also to several variants of the algorithms. Generally, as in [792] and [801], for the sake of stability, the matrices H_k and \underline{H}_k are transformed to upper triangular form by using Givens rotations; see Chapter 2. This can be done in place, without any additional storage. In these implementations of the algorithms the computation of the basis vectors and the transformation of the upper Hessenberg matrix H to upper triangular form are intertwined. This allows to compute the norm of the residual vector without having to compute the approximate solution x_k and the residual vector r_k .

In FOM the norm of the residual at iteration k is $h_{k+1,k} |[y_k^F]_k|$ where y_k^F is the solution of $H_k y_k^F = \|r_0\| e_1$. Since we reduce H_k to upper triangular form using $Q_k H_k = R_k$, where Q_k is the product of Givens rotation matrices ($Q_k = G_{k-1} \cdots G_1$, see Chapter 2, Section 2.9 but note that this was denoted there as $Q_k^* H_k$), we have

$$[y_k^F]_k = \frac{\|r_0\| [Q_k e_1]_k}{[R_k]_{k,k}}.$$

The numerator is obtained by applying successively (that is, as soon as they are computed) the Givens rotations to the initial right-hand side $\|r_0\| e_1$.

In GMRES the norm of the residual obtained from solving the least squares problem

$$\min \| \|r_0\| e_1 - \underline{H}_k y^G \|,$$

is simply $\|r_0\| |[\underline{Q}_k e_1]_{k+1}|$ where $\underline{Q}_k = G_k \cdots G_1$. In both cases the iterations can be stopped when the norm of the residual is smaller than $\epsilon \|b\|$ where ϵ is a user-given threshold. This criterion can be supplemented by the computation of estimates of the norm of the error as we will see in Section 5.8.

A template for GMRES using the modified Gram–Schmidt implementation (see Section 4.1) and Givens rotations is as follows.

```
function [x,ni,resn] = GMRES_MGS(A,b,x0,epsi,nitmax);
%
n = size(A,1);
rhs = zeros(nitmax+1,1);
```

```
V = zeros(n,nitmax);
H = zeros(nitmax+1,nitmax);
rot = zeros(2, nitmax); % init Givens rotations
resn = zeros(1,nitmax);
x = x0;
r = b - A * x;
ni = 0;
nb = norm(b);
bet = norm(r);
resn(1) = bet;
rhs(1) = bet;
v = r / bet;
V(:,1) = v;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    w = A * v;
    for j = 1:k % modified Gram-Schmidt
        vj = V(:,j);
        vw = vj' * w;
        H(j,k) = vw;
        w = w - vw * vj;
    end % for j
    nw = norm(w);
    v = w / nw;
    H(k+1,k) = nw;
    V(:,k+1) = v; % next basis vector
    nw1 = nw;
    % apply the preceding Givens rotations to the last column
    for kk = 1:k-1
        h1 = H(kk,k);
        h2 = H(kk+1,k);
        H(kk+1,k) = -rot(2,kk) * h1 + rot(1,kk) * h2;
        H(kk,k) = rot(1,kk) * h1 + conj(rot(2,kk)) * h2;
    end % for kk
    % compute, store and apply a new rotation to zero
    % the last term in kth column
    nw = H(k,k);
    cs = sqrt(abs(nw1)^2 + abs(nw)^2);
    if abs(nw) < abs(nw1)
        mu = nw / nw1;
        tau = conj(mu) / abs(mu);
    else
        mu = nw1 / nw;
        tau = mu / abs(mu);
    end
end
```

```

end % if abs
% store the rotation for the next columns
rot(1,k) = abs(nw) / cs; % cosine
rot(2,k) = abs(nw1) * tau / cs; % sine
% modify the diagonal entry and the right-hand side
H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
c = rhs(k);
rhs(k) = rot(1,k) * c;
rhs(k+1) = -rot(2,k) * c;
nresidu = abs(rhs(k+1)); % estimate of the residual norm
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k
%
% computation of the solution
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + V(:,1:k) * y;
resn = resn(1:ni+1);

```

In this code, orthogonalizing Av_k against the previous vectors yields the next basis vector v_{k+1} and the k -th column of the matrix H . Then, we apply the previous Givens rotations to the column just computed and a new rotation is computed to zero the entry $h_{k+1,k}$ and applied to the k -th column and to the right-hand side. The sine and cosine of the new rotation are stored to be able to apply the rotation for the next iterations. The (computed) norm of the residual is simply the modulus of the $k+1$ st component of the modified right-hand side. Note that this can be different from the norm of $b - Ax_k$ which is not computed in this code, particularly when we have reached the maximum attainable accuracy.

The CGS variant (see Section 4.1) is obtained by changing $w = A * v$; to $Av = A * v$; $w = Av$; and $vw = vj' * w$; to $vw = vj' * Av$.

The code for FOM is almost the same except that we do not want to apply the last rotation G_k if we wish to compute the iterate x_k at iteration k and we have to compute the residual norm differently. Hence, the following lines in the code above,

```

H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
c = rhs(k);
rhs(k) = rot(1,k) * c;
rhs(k+1) = -rot(2,k) * c;
nresidu = abs(rhs(k+1)); % estimate of the residual norm
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu

```

have to be replaced by

```
% estimate of the residual norm
nresidu = H(k+1,k) * abs(rhs(k) / H(k,k));
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
if k < nitmax
    H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
    c = rhs(k);
    rhs(k) = rot(1,k) * c;
    rhs(k+1) = -rot(2,k) * c;
end % if k
```

Of course, since FOM and GMRES use the same basis of the Krylov subspace and GMRES minimizes the norm of the residual, the residual norms given by FOM are larger than or equal to those given by GMRES. Moreover, if GMRES residual norms exactly stagnate for some iterations, the corresponding matrices H_k are singular and the FOM iterates are not defined (see Theorem 3.3), but this does not prevent to continue iterating.

5.2 Convergence of FOM and GMRES

In this section we study some techniques that have been used to assess GMRES convergence. They are based on bounds of residual norms using the residual polynomials.

Mathematically, FOM and GMRES terminate with the exact solution of the linear system at iteration $m \leq n$ if $h_{m+1,m} = 0$. This iteration number m is the grade of r_0 with respect to A , that is, the degree of the minimal polynomial of the vector r_0 with respect to A denoted as p_{A,r_0} in Chapter 2. However, this is not too interesting for at least two reasons. First, in real-life problems m is usually quite large and it is not practical to run FOM or GMRES for so many iterations. These algorithms are used as iterative methods and the user might want to stop the iterations long before having $\|r_m\| = 0$. Secondly, in finite precision arithmetic, it is likely that we will never get $h_{m+1,m} = 0$ exactly for any m , except in contrived examples. Therefore it is more interesting to look at how and why the residual norms are decreasing (or not) as a function of the iteration number.

Hundreds of papers have been written about GMRES convergence. Most of these works are based on the fact, as we have seen in Chapter 3, that the residual vectors are polynomials in A times the initial residual vector. At iteration k we have $r_k^G = p_k^G(A)r_0$ where p_k^G is a polynomial of degree k whose value is one at the origin. It is the solution of the minimization problem

$$\|r_k^G\| = \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)r_0\|. \quad (5.1)$$

Then, we have the following result which appeared in [308] Theorem 5.4, [301] Theorem 3.3 and [801] Proposition 4.

Theorem 5.2 *Let A be a diagonalizable matrix $A = X\Lambda X^{-1}$ with a spectrum $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$ and π_k be the set of polynomials of degree k . The norm of the residual vector at iteration $k < n$ of GMRES with the initial residual $r_0 = b - Ax_0$ satisfies*

$$\|r_k^G\| \leq \|r_0\| \|X\| \|X^{-1}\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|. \quad (5.2)$$

If the matrix A is normal we have

$$\|r_k^G\| \leq \|r_0\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|. \quad (5.3)$$

Proof Since, at iteration k , GMRES minimizes the Euclidean norm of the residual over the Krylov subspace, the residual norm satisfies (5.1). The matrix A is assumed to be diagonalizable and p in (5.1) is a polynomial, therefore,

$$\|p(A)r_0\| = \|Xp(\Lambda)X^{-1}r_0\| \leq \|r_0\| \|Xp(\Lambda)X^{-1}\| \leq \|r_0\| \|X\| \|X^{-1}\| \|p(\Lambda)\|.$$

The matrix $p(\Lambda)$ being diagonal we obtain $\|p(\Lambda)\| = \max_{\lambda \in \sigma(A)} |p(\lambda)|$.

When the matrix A is normal, the matrix X of the eigenvectors is unitary and $\|X\| = \|X^{-1}\| = 1$. \square

With this result it seems that all we have to do is to find expressions or upper bounds for

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\| \text{ or } \max_{\lambda \in \sigma(A)} |p(\lambda)|.$$

Several techniques have been considered to bound these terms. People have been looking for compact sets $\Omega \subset \mathbb{C}$ containing the eigenvalues of A that will give an upper bound when replacing $\sigma(A)$ by Ω in the inequalities above. One such approach is based on the field of values of A which is defined as

$$\mathcal{F}(A) = \{v^*Av \mid v^*v = 1, v \in \mathbb{C}^n\}.$$

The field of values is a convex set containing $\sigma(A)$. If the matrix A is normal the field of values is the convex hull of the eigenvalues. For the study of Krylov methods it is important to know if the origin is in the field of values. Unit vectors such that $v^*Av = 0$ are called *isotropic vectors*. Algorithms to compute them are described in [181, 680, 933]. If λ is an eigenvalue of A , we have $\lambda = v^*Av$ for the eigenvector v and the localization of the eigenvalues is related to subsets of the field of values; see [182].

Assuming that the origin is not in $\mathcal{F}(A)$, define

$$\mu(\mathcal{F}(A)) = \min_{z \in \mathcal{F}(A)} |z|,$$

the distance of $\mathcal{F}(A)$ from the origin. Then, we have

$$\|r_k^G\| \leq \|r_0\| \left(1 - \mu(\mathcal{F}(A)) \mu(\mathcal{F}(A^{-1}))\right)^{k/2}.$$

A refined bound is proved in [296], Theorem 6.1; see also [295]. A simpler bound was conjectured in [317],

$$\|r_k^G\| \leq 2 \|r_0\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{z \in \mathcal{F}(A)} |p(z)|.$$

Its proof depends on the proof (not yet obtained at the time of this writing) of the Crouzeix's conjecture [237–240]. With the results of [240] it can be proved that the preceding inequality holds with 2 replaced by $1 + \sqrt{2}$. For the relations of GMRES convergence with the field of values, see also [547].

For the case where the Hermitian part $M = (A + A^*)/2$ of A (whose eigenvalues are real) is positive definite, we show how to prove the bound

$$\|r_k^G\| \leq \|r_0\| \left(1 - \frac{\lambda_{\min}(M)^2}{\lambda_{\max}(A^*A)}\right)^{k/2}. \quad (5.4)$$

In this case the GMRES residual norms decrease strictly monotonically. The last inequality was already proved in [308], Theorem 5.4 by using a first-order polynomial $p_1 = 1 + \alpha z$ iterated k times,

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\| \leq \| [p_1(A)]^k \| \leq \|p_1(A)\|^k.$$

By definition

$$\|p_1(A)\|^2 = \max_{x \neq 0} \frac{((I + \alpha A)x, (I + \alpha A)x)}{(x, x)}.$$

Bounding the term in the right-hand side, the bound is minimized by choosing

$$\alpha = - \frac{\lambda_{\min}(M)}{\lambda_{\max}(A^*A)},$$

and this gives the bound for the norm of the residual.

This bound was improved in [84]. Let $\beta \in (0, \pi/2)$ such that $\cos(\beta) = \mu(\mathcal{F}(A))/\|A\|$. Then,

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \left(2 + \frac{2}{\sqrt{3}}\right)(2 + \gamma_\beta)\gamma_\beta^k,$$

where

$$\gamma_\beta = 2 \sin \left(\frac{\beta}{4 - 2\beta/\pi} \right) < \sin(\beta).$$

A generalization of the field of values is the polynomial numerical hull introduced in [715], for any k , as

$$\mathcal{H}_k(A) = \{z \in \mathbb{C} \mid \|p(A)\| \geq |p(z)|, \forall p \in \pi_k\}.$$

It has been shown that $\mathcal{H}_1(A) = \mathcal{F}(A)$. The polynomial numerical hull provides a lower bound on the quantity involved in the upper bound of $\|r_k^G\|$,

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{z \in \mathcal{H}_k(A)} |p(z)| \leq \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\|.$$

The polynomial numerical hull has been studied in [161, 162, 332, 457–459] for different types of matrices, for instance, Jordan blocks and triangular Toeplitz matrices. However, the sets $\mathcal{H}_k(A)$, when they are not known analytically, are most of the time difficult to compute.

Another kind of set that has been suggested is the ϵ -pseudospectrum of A which is defined as

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} \mid \|(zI - A)^{-1}\| \geq \epsilon^{-1}\}.$$

For properties and applications of the ϵ -pseudospectrum, see [926]. As in [924] let L_ϵ be the arc length of the boundary Γ_ϵ of $\Lambda_\epsilon(A)$. Then, using the Cauchy integral formula,

$$p(A) = \frac{1}{2\pi i} \int_{\Gamma_\epsilon} (zI - A)^{-1} p(z) dz,$$

where i is the square root of -1 and taking norms, we obtain the bound,

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\| \leq \frac{L_\epsilon}{2\pi\epsilon} \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{z \in \Lambda_\epsilon(A)} |p(z)|.$$

The problem is, of course, the choice of the parameter ϵ . It has to be chosen for the multiplicative factor to be of moderate size but small enough for the set Λ_ϵ to be not too large. Note that a “good” value of ϵ may not be the same for all iterations k .

In [317] there is a discussion of the relative merits and failures of the bound (5.2) based on eigenvalues and those based on the field of values and the pseudospectrum. Unfortunately, depending on the chosen example, it is not always the same bound which is descriptive of the behavior of the residual norms. Moreover, there are examples for which none of the bounds is satisfactory. For the bound (5.2) a large

condition number of the eigenvector matrix X could make the bound useless. This can happen if some eigenvectors are nearly aligned. A problem with the field of values bound is that the origin has to be outside of $\mathcal{F}(A)$ and this is not the case for many practical problems. Moreover, the field of values can be quite large due to non-normality or to outlying eigenvalues. The main difficulty with the pseudospectrum is the choice of the parameter ϵ .

When A is diagonalizable, using (5.2), we have to bound

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|.$$

This can be done by using well chosen polynomials. If we assume that the matrix A is real, its eigenvalues are real or pairs of complex conjugate numbers. If the eigenvalues are contained in a disk of center $c \in \mathbb{R}$ and radius $r > 0$ that excludes the origin, we can choose $p(\lambda) = (1 - \lambda/c)^k$. It yields

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)| \leq \left| \frac{r}{c} \right|^k.$$

Even though this situation is not very common in real-life problems, this tells us that if the disk is far from the origin with a radius not too large, convergence must be fast if $\|X\| \|X^{-1}\|$ is not large. A refined result can be obtained if, say, $m < n$ eigenvalues are located outside of the circle; see [177, 789, 801].

If we assume that the eigenvalues are within an ellipse with center $c \in \mathbb{R}$, focal distance $d > 0$, major semi-axis $a > 0$ and if the origin is outside the ellipse, then

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)| \leq \frac{C_k(a/d)}{|C_k(c/d)|},$$

where C_k is a complex Chebyshev polynomial. The exact expression for the right-hand side of this inequality is given in [798] (see also [789]) but it is approximately equal to

$$\left(\frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}} \right)^k.$$

In [789] Theorem 4.2, it is stated that there exist $k + 1$ eigenvalues of A that we can label as $\lambda_1, \dots, \lambda_{k+1}$ such that

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)| \leq \left(\sum_{j=1}^{k+1} \prod_{\substack{i=1 \\ i \neq j}}^{k+1} \frac{|\lambda_i|}{|\lambda_i - \lambda_j|} \right)^{-1}.$$

However, the proof in [791] is only valid if the eigenvalues are real. In that case, we even know that there exist $k + 1$ eigenvalues such that equality holds. Unfortunately, the eigenvalues $\lambda_1, \dots, \lambda_{k+1}$ were not specified. For a discussion of this bound, see [642]. In that paper it was conjectured that, in case of complex eigenvalues, the previous bound holds with a factor $4/\pi$ in front of the right-hand side.

The main problem with some of the previous bounds is that the origin has to be outside of the field of values. Another formulation was chosen in [829] where it is proved that, for a diagonalizable matrix,

$$\|r_{2k}^G\| \leq \|r_0\| \|X\| \|X^{-1}\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(q_2(\lambda))|,$$

for any polynomial q_2 of degree 2 such that $q_2(0) = 0$. This bound shows that if $q_2(A)$ has a positive definite symmetric part (but not necessarily A), we expect the norm of the residual after $2k$ iterations to be effectively bounded by the solution of the min-max problem associated with a polynomial p of degree k and not $2k$ as in the standard bound. When the eigenvalues are contained in a circle this yields bounds that look like

$$\|r_{2k}^G\| \leq c \|r_0\| \left| \frac{r}{c} \right|^k,$$

where c is independent of k .

Let us now discuss the different ways we have for bounding the residual norm. From what we have seen above we have,

$$\frac{\|r_k^G\|}{\|r_0\|} = \min_{\substack{p \in \pi_k \\ p(0)=1}} \frac{\|p(A)r_0\|}{\|r_0\|}, \quad (\text{GMRES}) \quad (5.5)$$

$$\leq \max_{\|v\|=1} \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)v\|, \quad (\text{worst-case GMRES}) \quad (5.6)$$

$$\leq \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\|. \quad (\text{ideal GMRES}) \quad (5.7)$$

The bound (5.6) is attainable (for some iteration numbers) and is the best bound independent of the initial residual. It describes the worst possible situation when varying the initial residual and, in that sense, it is of interest. Let us denote it by $\psi_k(A)$ as in [337]. The bound (5.7) which is denoted as $\varphi_k(A)$ and called the ideal GMRES bound was introduced in [464]. The motivation of the authors was described (in our notation) in the sentence: “*By passing from (5.5) to (5.7) we disentangle the matrix essence of the process from the distracting effects of the initial vector and end up with a pair of elegant mathematical problems*”. However, it turns out that the

initial vector is not always “a distracting effect”. We will see in the next section that we can easily construct matrices and right-hand sides such that GMRES exhibits complete stagnation that is $\|r_k^G\| = \|r_0\|$, $k = 1, \dots, n - 1$. But, if we change the right-hand side and the initial vector such that r_0 is an eigenvector of A , GMRES converges in one iteration. Convergence is also fast if r_0 is (approximately) in a subspace spanned by a few of the eigenvectors. So, the initial residual does matter sometimes. For instance, a real-life problem where the initial residual matters is described in [641]. Moreover, we will see that we can write explicitly the coefficients of the residual polynomial p_k^G as well as the norm of the residual r_k^G as functions of the eigenvalues and eigenvectors of A when the matrix is diagonalizable. Hence, replacing (5.5) by (5.7) is replacing a problem that we can solve exactly by a problem whose solution is, most of the time, not known. Nevertheless, (5.7) is an interesting mathematical problem by itself. Note that in (5.6) we have to minimize the norm of a vector whence in (5.7) we have to minimize the norm of a matrix.

To justify the use of the bound (5.7) some researchers formulated the question: Is (5.7) equal to (5.6) or does there exist matrices for which the inequality is strict? It is known that the equality $\varphi_k(A) = \psi_k(A)$ holds for normal matrices and for $k = 1$ for any nonsingular matrix. Moreover, we have $\varphi_k(A) = \psi_k(A) = 0$ when k is larger than the degree of the minimal polynomial of A . In [915] and [333] it was shown that there exist matrices A and iteration numbers k for which $\psi_k(A) < \varphi_k(A)$. The 4×4 example in [915] was extended to larger examples in [337]. It must be noticed that these examples considered real polynomials, matrices and vectors. Whether the results are the same for complex polynomials and vectors is studied in [337].

Bounds that depend on the initial residual vector were considered in [914]. The idea is to keep the residual vector together with the polynomial and to write

$$\|p(A)r_0\| = \|Xp(\Lambda)X^{-1}r_0\| \leq \|X\| \|p(\Lambda)X^{-1}r_0\|.$$

Let $c = X^{-1}r_0/\|r_0\|$. Then,

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \|X\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda)c\| = \|X\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \left(\sum_{j=1}^n |c_j|^2 |p(\lambda_j)|^2 \right)^{\frac{1}{2}}.$$

Note that $\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda)c\|$ is the residual norm at iteration k when using GMRES with $x_0 = 0$ for solving $\Lambda x = c$. We will see in Section 5.5, Theorem 5.17, that we can obtain an exact expression for this residual norm as a function of the eigenvalues in the diagonal of Λ . Following [914], the GMRES bound can be related to ideal GMRES on a rank-one modification of Λ since

$$\frac{\|r_k^G\|}{\|r_0\|} \leq (1 + \|X\|) \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda + z_j e_j^T)\|,$$

where, for any index $j \in [1, n]$ such that $c_j \neq 0$, the vector z_j is $z_j = (\Lambda - \lambda_j I)c$ and e_j is a vector in \mathbb{R}^n . The matrix $\Lambda + z_j e_j^T$ is diagonal except for the j th column. Numerical experiments in [914] show that these bounds can be effective in describing GMRES convergence.

Computable upper bounds for the GMRES residual norms depending on the initial residual are also considered in [633]. These bounds are based on the construction of so-called unitary GMRES-equivalent matrices. We will come back to the study of GMRES convergence using GMRES-equivalent matrices in Section 5.7.

Bounds of the residual norm for normal matrices involving differences of some eigenvalues are given in [548]. Upper bounds on GMRES residual norms for Toeplitz tridiagonal matrices are described in [637] and [630, 631].

FOM convergence has been less studied than GMRES convergence. The reason is probably that there can be iterations for which H_k is singular and the FOM iterates are said to be infinite. Therefore bounds of the residual norm must include a factor that can be potentially very large, if not infinite. In fact, we can use results in Theorem 3.3, relation (3.16) which yields

$$\|r_k^F\| = \frac{1}{|c_k|} \|r_k^G\|,$$

where c_k is the cosine of the Givens rotation. When GMRES stagnates ($\|r_k^G\| = \|r_{k-1}^G\|$) the sine s_k of the rotation is equal to 1 and $c_k = 0$. In any case we can write

$$|c_k| = \left(1 - \frac{\|r_k^G\|}{\|r_{k-1}^G\|} \right)^{\frac{1}{2}}.$$

Therefore, the residual norms of FOM and GMRES are linked by the relation,

$$\|r_k^F\| = \frac{\|r_{k-1}^G\|}{(\|r_{k-1}^G\|^2 - \|r_k^G\|^2)^{\frac{1}{2}}} \|r_k^G\|.$$

We observe that if $\|r_{k-1}^G\| \gg \|r_k^G\|$, that is, when GMRES converges fast, the multiplying factor is close to 1 and the FOM and GMRES residual norms are almost the same. When GMRES almost stagnates, the FOM residual norms are much larger than those of GMRES. Of course, we can bound $\|r_{k-1}^G\|$ and $\|r_k^G\|$ keeping the factor

$$\frac{1}{(\|r_{k-1}^G\|^2 - \|r_k^G\|^2)^{\frac{1}{2}}},$$

in the upper bound to obtain bounds for the FOM residual norms.

5.3 Prescribed convergence

In this section we describe how to construct matrices and right-hand sides such that FOM and GMRES yield prescribed residual norm convergence curves. We will see that we can also prescribe the eigenvalues of the matrix as well as Ritz or harmonic Ritz values at all iterations. More general results on prescribed convergence were discussed earlier in Chapter 3 for abstract Q-OR and Q-MR methods.

These constructions were proposed to try answering the question:

What are the properties of the matrix A determining the convergence of GMRES residual norms?

Unfortunately, the bounds of the residual norms described in the previous section do not allow to answer this question. The fact that GMRES convergence can be prescribed as well as the eigenvalues shows that everything can happen for the GMRES residual norms as long as they are decreasing and their behavior cannot be only linked to the eigenvalues of A , at least for non-normal matrices. The possibility of construction of linear systems with a prescribed convergence curve leads some people to write that “*for non-normal matrices GMRES convergence does not depend on the eigenvalues*”. We will see in Section 5.5 that this is not completely true. Things are more complicated than what is said by this simple statement.

To prescribe the residual norms in FOM and GMRES we use the general construction in Theorem 3.15.

Theorem 5.3 *Let f_j , $j = 0, \dots, n - 1$, $f_0 = 1$ be n nonzero, finite positive real numbers, λ_j , $j = 1, \dots, n$ be n nonzero complex numbers and $\theta_j^{(k)}$, $k = 1, \dots, n - 1$ and $j = 1, \dots, k$ be given nonzero complex numbers and α a positive real number. There exist matrices A and right-hand sides b such that when applying the FOM method for solving $Ax = b$ starting with a given vector x_0 , the norms of the residual vectors satisfy*

$$\frac{\|r_k^F\|}{\|r_0\|} = f_k, \quad k = 0, \dots, n - 1,$$

the eigenvalues of A are λ_j , $j = 1, \dots, n$, the eigenvalues of H_k , $k = 1, \dots, n - 1$ (the Ritz values) are $\theta_j^{(k)}$, $j = 1, \dots, k$ and $\|r_0\| = \alpha$. The matrices are of the form $A = VUCU^{-1}V^$ and the right-hand sides are of the form $Ax_0 + \alpha Ve_1$ for a unitary matrix V , for the companion matrix C corresponding to the eigenvalues λ_j , $j = 1, \dots, n$ and for an upper triangular matrix U such that the entries $\vartheta_{i,j}$ of its inverse satisfy $\vartheta_{1,1} = 1$ and for $k = 1, \dots, n - 1$,*

$$\begin{aligned} |\vartheta_{1,k+1}| &= 1/f_k, \\ \vartheta_{j,k+1} &= \frac{\alpha_{j-1}^{(k)}}{\alpha_0^{(k)} f_k}, \quad j = 2, \dots, k, \\ \vartheta_{k+1,k+1} &= \frac{1}{\alpha_0^{(k)} f_k}, \end{aligned}$$

where the coefficients of the monic polynomial with roots $\theta_j^{(k)}$, $j = 1, \dots, k$ are $\alpha_0^{(k)}, \dots, \alpha_{k-1}^{(k)}$ (in ascending order of powers).

Proof Let V be any nonsingular unitary matrix and $b = Ax_0 + \alpha Ve_1$. We first construct the upper triangular matrix $U^{-1} = \hat{U}^{-1}D$ according to Theorem 3.13. The diagonal entries of the diagonal matrix D are chosen as $d_{k+1,k+1} = 1/f_k$, $k = 0, \dots, n-1$. Let $[\hat{U}^{-1}]_{1,1} = 1$. For each $j = 1, \dots, n-1$ we construct the coefficients of the polynomial whose value at the origin is 1 and the roots are the given values $\theta_i^{(j)}$, $i = 1, \dots, j$. This can be done by dividing the $\alpha_{j-1}^{(k)}$ by $\alpha_0^{(k)}$; $\alpha_0^{(k)}$ is nonzero as otherwise, H_k would be singular and f_k infinite. These coefficients (starting with the constant term 1 for the first row) are the entries of the j th column of \hat{U}^{-1} . Consequently, the matrix U^{-1} has the entries as defined in the claim. Let C be the companion matrix corresponding to the monic polynomial with λ_j , $j = 1, \dots, n$ as roots. Then, we set

$$H = UCU^{-1}.$$

Finally, the matrix A is $A = VHV^*$. Clearly, we have $AV = VH$ and this yields Arnoldi-like relations (3.2). \square

The converse of this theorem is also true, i.e., any pair of matrix and right-hand side generating given prescribed FOM residual norms, Ritz values and spectrum are of the described form. We observe that with the scaling we have chosen here allowing to use a nonzero starting vector, the Krylov matrix K satisfies $K = \|r_0\|VU$.

We also note that the constructed matrix A is nonderogatory. As noted in [34] a classical matrix theory result states that a matrix is nonderogatory if and only if it is similar to its companion matrix (see for instance [531], Theorem 3.3.15). The previous theorem describes this similarity transformation explicitly.

To prescribe the residual norms in GMRES (that must be decreasing) we choose a strictly decreasing sequence g_j , $j = 0, \dots, n-1$. We compute the corresponding prescribed FOM residual norms by using the following result which is following from relation (3.17),

$$\frac{1}{\|r_k^F\|^2} = \frac{1}{\|r_k^G\|^2} - \frac{1}{\|r_{k-1}^G\|^2}. \quad (5.8)$$

Therefore,

$$\frac{1}{f_k^2} = \frac{1}{g_k^2} - \frac{1}{g_{k-1}^2},$$

and we construct the matrix A using these values of f_k . Note that we cannot prescribe stagnation of the GMRES residual norm in this way.

If we do not want to prescribe the Ritz values, the rows from 2 to n of the upper triangular matrix \hat{U}^{-1} can be chosen arbitrarily with nonzero entries on the diagonal. The only freedom in prescribing GMRES residual norms and the Ritz values of all n iterations using the previous construction is in the unitary matrix V which represents a unitary basis for the Krylov space $\mathcal{K}_n(A, r_0)$.

The code below shows how to construct the matrix A and the right-hand side b . The notation of the inputs corresponds to what we have seen above. The variable `alpha` contains the prescribed norm of r_0 , the variable `lamb` contains the prescribed eigenvalues of A and the columns of the array `Ritz` contain the prescribed Ritz values. In the outputs, `Uh` is the matrix \hat{U}^{-1} . To obtain a more practical code, we would have to test that we have enough values in the inputs.

```

function[A,b,D,Uh,C]=presc_convGMRES(g,x0,alpha,lamb,V,Ritz);
%
% coefficients of the FOM residual polynomials
% from the Ritz values
n = length(g);
Xi = zeros(n,n-1);
for j = 1:n-1
    y = transpose(polynome(Ritz(1:j,j)));
    y = y / y(j+1);
    Xi(1:j+1,j) = y(j+1:-1:1);
end
Uh(2:n,2:n) = Xi(2:n,1:n-1);
Uh(1,1:n) = ones(1,n);
% inverses of FOM residual norms
rF = zeros(1,n);
rF(1) = 1 / g(1);
for j = 2:n
    rF(j) = sqrt(1 / g(j)^2 - 1 / g(j-1)^2);
end
D = diag(1 ./ rF);
Di = diag(rF);
% companion matrix of the eigenvalues of A
C = companionmat(lamb);
Y = (Uh \ C) * Uh;
Y = D * Y * Di;
A = (V * Y) / V;
b = alpha * V(:,1) + A * x0;
end
%
function C=companionmat(lamb);
%
```

```

n = length(lamb);
p = polynome(lamb);
C = zeros(n,n);
C(:,n) = -p(n+1:-1:2);
C(2:n,1:n-1) = eye(n-1);
end
%
function c=polynome(x);
%
n = length(x);
c = [1 zeros(1,n)];
for j=1:n
    c(2:(j+1)) = c(2:(j+1)) - x(j).*c(1:j);
end
%the result should be real if the roots are complex conjugate
if isequal(sort(x(imag(x)>0)),sort(conj(x(imag(x)<0))))
    c = real(c);
end
end

```

In the preceding construction we assumed that FOM or GMRES does not terminate before iteration n which is the order of the matrix. However, as in [288], we can easily prescribe early termination at iteration m with $m < n$. We choose f_j , $j = 0, \dots, m-1$, $f_0 = 1$ to be m nonzero positive numbers (or the corresponding g_j , $j = 0, \dots, m-1$), λ_j , $j = 1, \dots, m$ to be m nonzero complex numbers and $\theta_i^{(j)}$, $j = 1, \dots, m-1$ and $i = 1, \dots, j$ to be given nonzero complex numbers. Then, we construct H_m in the same way as we constructed H above. The matrix A and the right-hand side b are

$$A = V \begin{pmatrix} H_m & B \\ 0 & D \end{pmatrix} V^*, \quad b = Ax_0 + \alpha Ve_1.$$

The matrices B and D can be chosen arbitrarily. The matrix D can even be chosen singular, in which case A is singular. Applying FOM or GMRES with these matrix and right-hand side, the iterations will terminate at iteration m since $h_{m+1,m} = 0$. When terminating at iteration m we prescribe only m eigenvalues of A . However, the other eigenvalues can also be prescribed by choosing D appropriately.

In practice we cannot construct very large matrices A because the matrix U^{-1} can become badly conditioned for large n and it becomes difficult to compute its inverse. So, this construction is more of theoretical than practical interest.

It is surprising that the Ritz values of all iterations can in the GMRES method be prescribed independent from the residual norms, because for close to normal situations it has been shown that residual norm convergence depends on the Ritz value distribution (see, e.g., [943]). However, Ritz values are not the roots of GMRES

polynomials; the roots of GMRES polynomials are harmonic Ritz values, i.e., the eigenvalues of the matrices \hat{H}_k defined in (3.23). We can use the general results in Section 3.5, in particular Corollary 3.3 and Theorem 3.16, to show that even harmonic Ritz values can in the GMRES method be prescribed independent from the residual norms.

Theorem 5.4 *Assume we are given n positive numbers*

$$g_0 > g_1 > \dots > g_{n-1} > 0, \quad g_0 = 1,$$

n nonzero complex numbers λ_j , $j = 1, \dots, n$, nonzero complex numbers $\zeta_j^{(k)}$ for $j = 1, \dots, k$ and $k = 1, \dots, n - 1$ and a positive real number α . There exist matrices A and right-hand sides b such that when applying the GMRES method for solving $Ax = b$ starting with a given vector x_0 , the norms of the residual vectors satisfy

$$\frac{\|r_k^G\|}{\|r_0\|} = g_k, \quad k = 0, \dots, n - 1,$$

the eigenvalues of A are λ_j , $j = 1, \dots, n$, the eigenvalues of the matrices \hat{H}_k , $k = 1, \dots, n - 1$ defined in (3.23) (the harmonic Ritz values) are $\zeta_j^{(k)}$, $j = 1, \dots, k$ and $\|r_0\| = \alpha$. The matrices are of the form $A = VUCU^{-1}V^$ and the right-hand sides are of the form $Ax_0 + \alpha Ve_1$ for a unitary matrix V , for the companion matrix C corresponding to the eigenvalues λ_j , $j = 1, \dots, n$ and for an upper triangular matrix U such that the entries $\vartheta_{i,j}$ of its inverse satisfy $\vartheta_{1,1} = 1$ and for $k = 1, \dots, n - 1$,*

$$\begin{aligned} \vartheta_{k+1,k+1} &= \frac{1}{g_k^2 |\beta_0^{(k)}| \sqrt{1/g_k^2 - 1/g_{k-1}^2}}, \\ \vartheta_{1,k+1} &= \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|} \sqrt{1/g_k^2 - 1/g_{k-1}^2}, \\ \vartheta_{j,k+1} &= \beta_{j-1}^{(k)} \vartheta_{k+1,k+1} - \frac{e_j^T U_k^{-1} U_k^{-*} e_1}{\vartheta_{1,k+1}}, \quad j = 2, \dots, k. \end{aligned}$$

where the coefficients of the monic polynomial with roots $\zeta_j^{(k)}$, $j = 1, \dots, k$ are $\beta_0^{(k)}, \dots, \beta_{k-1}^{(k)}$ (in ascending order of powers).

Proof We can use Theorem 3.16 and modify it for the GMRES method using (5.8). To generate prescribed harmonic Ritz values in addition to FOM residual norms, the entries $\vartheta_{j,k+1}$ corresponding to the k th iteration have to satisfy the equalities in Corollary 3.3.

From

$$|\vartheta_{1,k+1}| = 1/f_k,$$

$$\frac{\vartheta_{1,k+1}}{|\vartheta_{1,k+1}|} = \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|},$$

we obtain, using (5.8), that

$$\vartheta_{1,k+1} = \frac{\beta_0^{(k)}}{|\beta_0^{(k)}|} \sqrt{1/g_k^2 - 1/g_{k-1}^2}.$$

Furthermore, we have

$$\begin{aligned} \vartheta_{k+1,k+1} &= \frac{1 + f_k^2 \|U_k^{-*} e_1\|^2}{f_k |\beta_0^{(k)}|} = \frac{1 + f_k^2 \sum_{j=1}^k |\vartheta_{1,j}|^2}{f_k |\beta_0^{(k)}|} \\ &= \frac{1 + f_k^2 \sum_{j=1}^k 1/f_{j-1}^2}{f_k |\beta_0^{(k)}|} = \frac{\sum_{j=1}^{k+1} 1/f_{j-1}^2}{|\beta_0^{(k)}|/f_k} = \frac{1/g_k^2}{|\beta_0^{(k)}| \sqrt{1/g_k^2 - 1/g_{k-1}^2}}, \end{aligned}$$

where for the last equality we used again (5.8). This gives the expression for $\vartheta_{k+1,k+1}$ in the claim. The rest of the claim is obtained using the same construction as in the proof of Theorem 3.16. \square

Prescribing the GMRES residual norms can also be done by using an orthonormal basis of the subspace $A\mathcal{K}_k(A, r_0)$ as it was done in [34]; see also [640]. For completeness, let us describe this construction.

Theorem 5.5 *Assume we are given n positive numbers*

$$g_0 \geq g_1 \geq \cdots \geq g_{n-1} > 0, \quad g_0 = 1,$$

n complex numbers $\lambda_1, \dots, \lambda_n$ all different from 0 and α real positive. Let A be a square nonsingular matrix of order n and b an n -dimensional vector. The following assertions are equivalent:

- 1- *The spectrum of A is $\{\lambda_1, \dots, \lambda_n\}$ and GMRES applied to A and b starting from x_0 yields residuals r_j^G , $j = 0, \dots, n-1$ such that $\|r_0\| = \alpha$ and*

$$\frac{\|r_j^G\|}{\|r_0\|} = g_j, \quad j = 0, \dots, n-1.$$

- 2- *The matrix A is of the form $A = WYCY^{-1}W^*$ and $b = Ax_0 + \alpha Wh$, where W is a unitary matrix, Y is given by*

$$Y = \begin{pmatrix} h & R \\ 0 & 0 \end{pmatrix},$$

R being any nonsingular upper triangular matrix of order $n - 1$, h a vector such that

$$h = (\eta_1, \dots, \eta_n)^T, \quad \eta_j = (g_{j-1}^2 - g_j^2)^{\frac{1}{2}}$$

and C is the companion matrix corresponding to the eigenvalues $\lambda_1, \dots, \lambda_n$.

In reference to the three authors of the paper [34] we call the parametrization, $A = WCY^{-1}W^*$, the APS parametrization. In this parametrization, the prescribed residual norm convergence curve is implicitly contained in the vector h which is the first column of Y . The unitary matrix W represents a unitary basis of the Krylov space $AK_n(A, r_0)$. In [34] it was proven that the Krylov matrix K is equal to the matrix WY . With our different scaling we have $K = \|r_0\| WY$. The structure of Y can be understood considering that the first column of K is r_0 and the other columns are the $n - 1$ first columns of AK . So, the first column of Y has to be nonzero and the other columns must be those of an upper triangular matrix appended with a zero last row. For more details on the APS parametrization, see also [640].

Let us consider the relations between the two parametrizations as well as some other properties of the involved matrices.

First, we have relations between the $\vartheta_{1,j}$'s which are the entries of the first row of U^{-1} and the η_j 's. We remark that $g_j^2 = \eta_{j+1}^2 + \dots + \eta_n^2$ and

$$|\vartheta_{1,j+1}|^2 = \frac{1}{f_j^2} = \frac{1}{g_j^2} - \frac{1}{g_{j-1}^2}.$$

It yields

$$|\vartheta_{1,j+1}|^2 = \frac{\eta_j^2}{(\eta_{j+1}^2 + \dots + \eta_n^2)(\eta_j^2 + \dots + \eta_n^2)}.$$

On the other hand, from relation (3.35), we have $g_j^2 = 1/(|\vartheta_{1,1}|^2 + \dots + |\vartheta_{1,j+1}|^2)$ and

$$\eta_j^2 = \frac{|\vartheta_{1,j+1}|^2}{(|\vartheta_{1,1}|^2 + \dots + |\vartheta_{1,j}|^2)(|\vartheta_{1,1}|^2 + \dots + |\vartheta_{1,j+1}|^2)}.$$

Theorem 5.6 *The matrix U is the upper triangular Cholesky factor of the matrix Y^*Y . A QR factorization of the matrix H is $H = QR$ where $Q = V^*W$ is upper Hessenberg unitary and R is upper triangular. Moreover, we have*

$$Q = UY^{-1} = U^{-*}Y^*, \quad R = YCU^{-1}.$$

The matrices Q and R are also related to the APS parametrization by $RQ = YCY^{-1}$.

Proof We have $WY = VU$ and

$$Y^*W^*WY = Y^*Y = U^*V^*VU = U^*U.$$

The matrix U being upper triangular with positive diagonal entries, U^* is the Cholesky factor of Y^*Y . Moreover, $Y = W^*VU$. The columns of the matrix W in the APS parametrization give an ascending basis of the Krylov space $A\mathcal{K}_n(A, r_0)$, and we have $AK = W\tilde{R}$ where the matrix \tilde{R} is upper triangular and

$$\begin{aligned} W\tilde{R} &= AK, \\ &= \|r_0\| AVU, \\ &= \|r_0\| VHU, \\ &= \|r_0\| AWY, \\ &= \|r_0\| WYC. \end{aligned}$$

Therefore, $\tilde{R} = \|r_0\| YC$. Moreover, $VHU = WYC$ and

$$H = V^*WYC U^{-1} = Q\mathcal{R},$$

where $Q = V^*W$ is unitary and $\mathcal{R} = YCU^{-1}$ is upper triangular since

$$YC = \frac{1}{\|r_0\|} \tilde{R}.$$

We have a QR factorization of H . It gives the relation between both orthonormal bases since $W = VQ$. Using the other relation linking V and W (that is, $WY = VU$) we obtain

$$Q = UY^{-1} = U^{-*}Y^*.$$

This relation implies $Y = Q^*U$, which is a QR factorization of Y . The unitary factor Q^* is just the conjugate transpose of that of H .

We remark that since $H = Q\mathcal{R} = V^*AV$, we have two factorizations of the matrix A as

$$A = VQRV^* = VQ(\mathcal{R}Q)Q^*V^* = W\mathcal{H}W^*.$$

The matrix

$$\mathcal{H} = \mathcal{R}Q = YCY^{-1}$$

is also upper Hessenberg. This is an RQ factorization of \mathcal{H} . Since

$$H = UCU^{-1} = QYCY^{-1}Q^*,$$

we obtain the relation between H and \mathcal{H} , $H = Q\mathcal{H}Q^*$. The matrix Q transforms the upper Hessenberg matrix \mathcal{H} to the upper Hessenberg matrix H . \square

We would like to characterize the unitary matrix Q that was introduced as a factor of the factorization $H = QR$. The entries of this matrix can be written explicitly as functions of the sines and cosines of the Givens rotations used to reduce H to upper triangular form; see [153], relation (2.26) and [152], relation (4.3) page 65. However, it is also interesting to relate the entries of Q to the η_j 's defined in Theorem 5.5 that describe GMRES convergence as well as to the GMRES residual norms. Note that the scaling of Q and \mathcal{R} in [152] is different from ours. The matrix Q is given by $Q = UY^{-1}$. Therefore we first compute U and the inverse of Y .

Lemma 5.1 *Using the notation of Theorem 5.5, let \hat{h} be the vector of the first $n - 1$ components of h . The inverse of the matrix Y defined in Theorem 5.5 is*

$$Y^{-1} = \begin{pmatrix} 0 & \cdots & 0 & 1/\eta_n \\ R^{-1} & & -R^{-1}\hat{h}/\eta_n \end{pmatrix}. \quad (5.9)$$

Let \hat{L} be the lower triangular Cholesky factor of $R^*R - R^*\hat{h}\hat{h}^T R$. Then

$$U = \begin{pmatrix} 1 & \hat{h}^T R \\ 0 & \vdots \\ \hat{L}^* & 0 \end{pmatrix}. \quad (5.10)$$

Proof One can easily check that the matrix in (5.9) is the inverse of the matrix Y . The matrix U^* is the lower triangular Cholesky factor of the Hermitian positive definite matrix Y^*Y . From the definition of Y and $\|h\| = 1$, we have

$$Y^*Y = \begin{pmatrix} 1 & \hat{h}^T R \\ R^*\hat{h} & R^*R \end{pmatrix}.$$

The Cholesky factor L of the matrix Y^*Y is

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ R^*\hat{h} & \hat{L} \end{pmatrix}$$

if $\hat{L}\hat{L}^* = R^*R - R^*\hat{h}\hat{h}^T R = R^*(I - \hat{h}\hat{h}^T)R$ is the Cholesky factorization of the matrix in the right-hand side. This matrix is Hermitian positive definite because the eigenvalues of the symmetric matrix $I - \hat{h}\hat{h}^T$ are $n - 1$ times 1 and η_n^2 . Therefore, \hat{L} has real and positive diagonal entries and the factorization is unique. Since $U = L^*$ this proves the result for the matrix U . \square

Since we know the matrices U and Y^{-1} from Lemma 5.1, we can immediately write down $Q = UY^{-1}$ as

$$Q = \begin{pmatrix} \hat{h}^T & \frac{1}{\eta_n} - \frac{\|\hat{h}\|^2}{\eta_n} \\ \hat{L}^* R^{-1} & -\frac{\hat{L}^* R^{-1} \hat{h}}{\eta_n} \end{pmatrix}. \quad (5.11)$$

This expression can be considerably simplified as shown in the following theorem.

Theorem 5.7 *Let \check{L} be the lower triangular Cholesky factor of the matrix $I - \hat{h}\hat{h}^T$, where \hat{h} was defined in Lemma 5.1 and Theorem 5.5, and S be a diagonal matrix whose diagonal entries are plus or minus times $e^{-\phi_i}$ where ϕ_i is the phase of $(R)_{i,i}$. Then, the Q factor in $H = QR$ is*

$$Q = \begin{pmatrix} \hat{h}^T & \eta_n \\ S\check{L}^T & -\frac{S\check{L}^T \hat{h}}{\eta_n} \end{pmatrix}. \quad (5.12)$$

Moreover, the entries of the upper triangular matrix \check{L}^T appearing in (5.12) are for $j > i$,

$$\begin{aligned} (\check{L}^T)_{i,j} &= -\frac{\eta_i \eta_j}{\sqrt{\eta_{i+1}^2 + \dots + \eta_n^2} \sqrt{\eta_i^2 + \dots + \eta_n^2}}, \\ &= -\left(\frac{1}{\|r_i^G\|^2} - \frac{1}{\|r_{i-1}^G\|^2}\right)^{\frac{1}{2}} \left(\|r_{j-1}^G\|^2 - \|r_j^G\|^2\right)^{\frac{1}{2}}, \end{aligned}$$

and the diagonal entries are

$$(\check{L}^T)_{i,i} = \frac{\sqrt{\eta_{i+1}^2 + \dots + \eta_n^2}}{\sqrt{\eta_i^2 + \dots + \eta_n^2}} = \frac{\|r_i^G\|}{\|r_{i-1}^G\|}.$$

Proof We first remark that since $\|h\|^2 = \|\hat{h}\|^2 + \eta_n^2 = 1$, then

$$\frac{1}{\eta_n} - \frac{\|\hat{h}\|^2}{\eta_n} = \eta_n.$$

From Lemma 5.1 we have $\hat{L}\hat{L}^* = R^*R - R^*\hat{h}\hat{h}^T R$. This implies

$$(R^{-*}\hat{L})(\hat{L}^*R^{-1}) = I - \hat{h}\hat{h}^T = \check{L}\check{L}^T.$$

However we cannot directly identify \hat{L}^*R^{-1} , which appears in Q , with \check{L}^T . This last matrix is the upper triangular Cholesky factor of $I - \hat{h}\hat{h}^T$ and has positive diagonal entries. The matrices \check{L}^T and R^{-1} being both upper triangular, the diagonal entries of the upper triangular matrix \hat{L}^*R^{-1} are

$$(\hat{L}^* R^{-1})_{i,i} = \frac{(\hat{L}^*)_{i,i}}{(R)_{i,i}}.$$

The diagonal entries of \hat{L}^* are real and positive and $R_{i,i}$ may be complex. Let \mathcal{S} be a diagonal matrix whose diagonal entry $(\mathcal{S})_{i,i}$ is plus or minus times $e^{-i\phi_i}$ where ϕ_i is the phase of $(R)_{i,i}$. Then we have $\hat{L}^* R^{-1} = \mathcal{S} \check{L}^T$. This together with (5.11) proves the result.

Moreover, since \check{L} is the lower triangular Cholesky factor of a rank-one modification of the identity matrix, its entries can be computed. This can be done using the results of [416], where the authors considered an $\mathcal{L}\mathcal{D}\mathcal{L}^T$ factorization with \mathcal{L} lower triangular and \mathcal{D} diagonal. The matrix \mathcal{L} has a very special structure with ones on the diagonal and

$$(\mathcal{L})_{i,j} = \eta_i \gamma_j, \quad i = 2, \dots, n, \quad j = 1, \dots, n-1.$$

The values γ_j and the diagonal elements d_j of the diagonal matrix \mathcal{D} are computed by recurrences

$$\delta_1 = -1,$$

and for $j = 1, \dots, n$,

$$d_j = 1 + \delta_j (\eta_j)^2, \quad \gamma_j = \delta_j \frac{\eta_j}{d_j}, \quad \delta_{j+1} = \frac{\delta_j}{d_j}.$$

It is easy to see that

$$d_j = \frac{\eta_{j+1}^2 + \dots + \eta_n^2}{\eta_j^2 + \dots + \eta_n^2} = \frac{\|r_j^G\|^2}{\|r_{j-1}^G\|^2}, \quad \delta_j = -\frac{1}{\eta_j^2 + \dots + \eta_n^2} = -\frac{\|r_0\|^2}{\|r_{j-1}^G\|^2},$$

and

$$\gamma_j = -\frac{\eta_j}{\eta_{j+1}^2 + \dots + \eta_n^2} = -\frac{\eta_j \|r_0\|^2}{\|r_j^G\|^2}.$$

We have to take the square roots of the elements d_j to obtain the Cholesky factor $\check{L} = \mathcal{L}\sqrt{\mathcal{D}}$. The multiplying factor for the column j below the diagonal element is

$$\gamma_j \sqrt{d_j} = -\frac{\eta_j}{\sqrt{\eta_{j+1}^2 + \dots + \eta_n^2} \sqrt{\eta_j^2 + \dots + \eta_n^2}} = -\frac{\eta_j \|r_0\|^2}{\|r_j^G\| \|r_{j-1}^G\|},$$

and therefore $(\check{L})_{i,j} = \eta_i \gamma_j \sqrt{d_j}$. By transposing this we obtain the entries we need for the matrix Q . The diagonal element of the column j is

$$(\check{L})_{j,j} = \sqrt{d_j} = \frac{\sqrt{\eta_{j+1}^2 + \dots + \eta_n^2}}{\sqrt{\eta_1^2 + \dots + \eta_n^2}} = \frac{\|r_j^G\|}{\|r_{j-1}^G\|},$$

and this concludes the proof. \square

In the QR factorization of Theorem 5.7, the upper triangular matrix \mathcal{R} is scaled such that the entries on the first row of Q are real and positive.

We observe that the result of Theorem 5.7 can be used for any upper Hessenberg matrix H with real positive subdiagonal entries. Then, the residual norms involved in Q are those obtained by applying GMRES to the matrix H itself with e_1 as a right-hand side and a zero starting vector.

5.4 Stagnation of GMRES

In this section we characterize the matrices and right-hand sides that yield complete or partial stagnation of GMRES. Recall that, when GMRES stagnates in some iteration, the corresponding Hessenberg matrix is singular and the iterate for FOM is not defined.

Let us assume for simplicity that GMRES does not stop before iteration n . As in Chapter 3 we define partial stagnation for a given matrix A and right-hand side b as having

$$\|r_k^G\| < \|r_{k-1}^G\|, \quad k = 1, \dots, m, \quad \|r_k^G\| = \|r_{k-1}^G\|, \quad k = m+1, \dots, m+p-1,$$

and

$$\|r_k^G\| < \|r_{k-1}^G\|, \quad k = m+p, \dots, n.$$

Hence the norms of the residual stay the same for p iterations starting from $k = m$. Complete stagnation corresponds to $m = 0$ and $p = n$. Thus

$$\|r_k^G\| = \|r_0\|, \quad k = 1, \dots, n-1, \quad \text{and} \quad \|r_n^G\| = 0.$$

Complete stagnation of GMRES was studied in [1016, 1017]; see also [829, 838] where conditions for non-stagnation were studied, [642] where the worst-case convergence of GMRES for normal matrices was considered and [23]. A study of necessary and sufficient conditions for complete stagnation when $n \leq 4$ is given in [679]. A complete characterization of matrices yielding complete or partial stagnation was given in [682]. Note that the definition of partial stagnation in [1017] is different from ours. The definition in that paper corresponds to $m = 0$; that is, stagnation at the beginning of GMRES iterations.

The paper [679] studied the existence of stagnation vectors for a given real matrix. We have the following result which was proved in [1017].

Theorem 5.8 Let $x_0 = 0$ and A be a real matrix. We have complete stagnation of GMRES if and only if the right-hand side b of the linear system $Ax = b$ satisfies

$$(b, A^j b) = 0, \quad j = 1, \dots, n - 1. \quad (5.13)$$

If it exists, such a vector b is called a stagnation vector.

The characterization of Theorem 5.8 shows that complete stagnation is not possible if the origin is outside the field of values of any of the matrices A^j , $j = 1, \dots, n - 1$, which are powers of A . This is the case if the symmetric part of any of these matrices is definite. We add the condition $\|b\| = 1$ to (5.13) since if b is a solution then ab is also a solution of (5.13). Note that if b is a solution, then $-b$ is also a solution. Hence, the number of stagnation vectors is even. We now have n nonlinear equations in n unknowns, (the components of b), and the question is to know if there exists at least a real solution and, if so, how many. Since we are interested in real vectors b , the system defined by (5.13) and $b^T b = 1$ is a polynomial system. Theorem 5.8 can be rephrased as follows.

Theorem 5.9 A necessary condition to have a stagnation vector is that the origin is in the field of values of A^j , $j = 1, \dots, n - 1$.

Proof Clearly a solution b has to be in the intersection of the inverse images of 0 for the functions $b \mapsto (A^j b, b)$, $j = 1, \dots, n - 1$. If at least one of these sets is empty, there is no solution to the nonlinear system. \square

The converse of this theorem is false for $n \geq 3$ and real vectors b . Counter-examples are given in [679]. There exist real matrices A with 0 in the fields of values of A^j for $j = 1, \dots, n - 1$, and no real stagnation vector. The nonlinear system (5.13) can be transformed into a problem with symmetric matrices since with a real matrix B and a real vector b , we have the equivalence

$$b^T B b = 0 \Leftrightarrow b^T (B + B^T) b = 0.$$

Therefore, when A is real and if we are looking for real vectors b , we can consider the polynomial system

$$b^T (A^j + (A^j)^T) b = b^T A_{(j)} b = 0, \quad j = 1, \dots, n - 1, \quad b^T b = 1, \quad (5.14)$$

with symmetric matrices $A_{(j)}$. The polynomial system (5.14) corresponds to the simultaneous annealing of $n - 1$ quadratic forms defined by symmetric matrices and a nonzero vector. We are only interested in real solutions since complex solutions of (5.14) do not provide a solution of the real stagnation problem. The following straightforward theorem gives a sufficient condition for the non-existence of real stagnation vectors.

Theorem 5.10 Let A be a real matrix of order n . A sufficient condition for the non-existence of unit real stagnation vectors b is that there exist a vector μ with real components μ_j , $j = 1, \dots, n - 1$ such that the matrix

$$A(\mu) = \sum_{j=1}^{n-1} \mu_j A_{(j)}$$

is (positive or negative) definite.

Proof Let us assume that the matrix $A(\mu)$ is definite for a given choice of coefficients. If there is a real vector b satisfying equation (5.14), multiply $A(\mu)$ by b from the right and b^T from the left. Then

$$b^T A(\mu) b = b^T \left(\sum_{j=1}^{n-1} \mu_j A_{(j)} \right) b = \sum_{j=1}^{n-1} \mu_j b^T A_{(j)} b = 0,$$

but since the matrix $A(\mu)$ is definite, this gives $b = 0$ which is impossible since $b^T b = 1$. \square

Therefore, to have at least one real stagnation vector, the matrix $A(\mu)$ must be indefinite for any choice of the real numbers μ_j . Of course, as we already know, there is no stagnation vector if any of the matrices $A_{(j)}$ is definite. The solution of these polynomial systems were studied in [679] using results from algebraic geometry. Unfortunately, only small systems of order $n \leq 4$ can be studied in this way.

We now return to the problem of characterizing linear systems for which we have stagnation. Let us study the stagnation phenomenon using the results established for the convergence curve of the residual norms since stagnation is a particular case of prescribed convergence. We consider partial stagnation since complete stagnation is just a particular case. In the notation of the previous section it is equivalent to

$$g_k = g_{k-1}, \quad k = m + 1, \dots, m + p - 1.$$

From the relation between the FOM and the GMRES residual norms it gives $|\vartheta_{1,k}| = 0$, $k = m + 2, \dots, m + p$. We cannot use the matrix D in the proof of Theorem 5.3 any longer but we can still construct the first row of the matrix U^{-1} whose entries are the $\vartheta_{1,j}$. If we are just interested in constructing matrices yielding stagnation in given iterations we can choose the other entries of U^{-1} as we wish (this is not true if in addition, we prescribe Ritz values or harmonic Ritz values, which moreover satisfy special conditions in the stagnation case; see [281, 287]). Then, we take $A = VUCU^{-1}V^*$ and $b = Ax_0 + \alpha Ve_1$, V being any unitary matrix. Hence, partial stagnation corresponds to matrices A for which the matrix U^{-1} has zero entries on the corresponding positions of its first row.

Stagnation conditions can also be studied as in [682] using the unitary matrix Q of Section 5.3.

Lemma 5.2 *In case of partial stagnation of GMRES the columns $j = m + 1, \dots, m + p - 1$, of Q are zero except for the subdiagonal entries $(j+1, j)$ which are $\pm e^{-i\phi_j}$. The rows $i = m + 2, \dots, m + p$ are zero for columns i to n .*

Proof Partial stagnation corresponds to having some values η_j equal to zero. We apply Theorem 5.7. We have seen that $Q_{i,j}$ is proportional to $\eta_{i+1}\eta_j$ for $1 < i \leq j < n$. Therefore, the columns $m + 1$ to $m + p - 1$ are zero except for the subdiagonal entries which are $\pm e^{-i\phi_i}$ depending on $(R)_{i,i}$. The rows i from $m + 2$ to $m + p$ are zero for columns i to $n - 1$. The entries in the last column for these rows are zero because of the structure of \check{L}^T and \hat{h} . \square

Then we can characterize the matrices and right-hand sides leading to partial stagnation. Note that we do not scale the initial residual as before since this is not necessary to study stagnation. If we want to normalize we would have to slightly change the definitions of the right-hand sides.

Theorem 5.11 *We have partial stagnation in GMRES for iteration $m \geq 0$ to iteration $m + p - 1$ if and only if the matrix A and the right-hand side b can be written as*

$$A = W\mathcal{R}QW^*, \quad b = Ax_0 + WQ^*e_1,$$

where W is unitary, \mathcal{R} is upper triangular, Q is unitary with the sparsity structure defined in Lemma 5.2. We have also the same partial stagnation if and only if $A = VQ\mathcal{R}V^*$ and $b = Ax_0 + Ve_1$ with V unitary and Q and \mathcal{R} as before.

Proof Assume that we have partial stagnation as defined above. From Theorems 5.5 and 5.6,

$$A = WYCY^{-1}W^* = W\mathcal{R}QW^*.$$

We have seen from Theorem 5.7 and Lemma 5.2 that Q has the required structure and values. Moreover $b = Ax_0 + Wh = Ax_0 + WQ^*e_1$ in the APS parametrization. For the proof of the converse it is enough to show that if we have such a factorization of the matrix A , the columns of W are a basis of $A\mathcal{K}_n(A, r_0)$. This will be done below for complete stagnation in Theorem 5.13. The proof of the other assertion follows from the relation between W and V in the APS parametrization. \square

Of course, one can easily extend this theorem to the case where we have several sequences of stagnation separated with phases of strict decrease of the residual norm. Complete stagnation is a very particular case of partial stagnation. It implies that the vector \hat{h} of the first $n - 1$ components of h is identically zero and $\check{L} = I$ and the proof of the following result is straightforward.

Theorem 5.12 *In case of complete stagnation, the orthogonal matrix Q in $H = QR$ is*

$$Q = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ \mathcal{S} & & & 0 \end{pmatrix}, \quad (5.15)$$

where \mathcal{S} is a diagonal matrix with $\pm e^{-i\phi_i}$ as diagonal entries according to the diagonal entries of R .

Note that Theorem 5.12 implies that $H = QR$ is an upper Hessenberg matrix with a first row proportional to $(e_n)^T$; that is, all the elements of the first row are zero, except the last one.

Theorem 5.13 *We have complete stagnation in GMRES if and only if the matrix A can be written as $A = W\mathcal{U}P\mathcal{W}^*$, where W is unitary, \mathcal{U} is upper triangular and nonsingular and*

$$P = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ & I & & 0 \end{pmatrix}, \quad (5.16)$$

is a permutation matrix. A right-hand side giving stagnation is $b = Ax_0 + We_n$.

Proof Assume that we have complete stagnation. From Theorems 5.5 and 5.6

$$\begin{aligned} A &= W\mathcal{U}CY^{-1}\mathcal{W}^* \\ &= W\mathcal{R}\mathcal{Q}\mathcal{W}^*. \end{aligned}$$

The matrix Q has the structure given in (5.15). Such a matrix can be written as $Q = \check{\mathcal{S}}P$ with P defined in (5.16) and $\check{\mathcal{S}}$ is a diagonal matrix with $(\check{\mathcal{S}})_{1,1} = 1$ and $(\check{\mathcal{S}})_{j,j} = (\mathcal{S})_{j-1,j-1}$, $j = 2, \dots, n$. The matrix $\check{\mathcal{S}}$ can be absorbed in the upper triangular matrix by defining $\mathcal{U} = \mathcal{R}\check{\mathcal{S}}$. Moreover $b = Ax_0 + Wh = Ax_0 + We_n$ in the APS parametrization of Theorem 5.5.

Conversely, let us assume that $A = W\mathcal{U}P\mathcal{W}^*$ and $b = Ax_0 + We_n$. From [460], section 2 page 466, we just need to show that the columns of W are a basis of $A\mathcal{K}_n$. We proceed by induction. Since $W^*r_0 = e_n$ and $Pe_n = e_1$ we have

$$Ar_0 = W\mathcal{U}P\mathcal{W}^*r_0 = W\mathcal{U}e_1.$$

But $\mathcal{U}e_1$ is a vector proportional to e_1 . Hence Ar_0 is proportional to w_1 , the first column of W . More generally, assume that $A^{j-1}r_0 = Wq$ where q is a vector with only the first $j-1$ components nonzero. Then,

$$A^j r_0 = A(A^{j-1}r_0) = W\mathcal{U}Pq.$$

The j first components of the vector Pq are $0, q_1, \dots, q_{j-1}$. Multiplying the vector Pq by the nonsingular upper triangular matrix \mathcal{U} , we obtain a vector with the first j components nonzero showing that $A^j r_0$ is in the span of the j first columns of W . \square

The result of Theorem 5.13 could have also been obtained from Theorem 2.2 in [34] page 639. In case of complete stagnation the matrix denoted \hat{H} in [34] is the same as P . We note that $\mathcal{U}P$ is an upper Hessenberg matrix whose last column is proportional to e_1 . Other equivalent characterizations can be obtained as shown in the following theorem.

Theorem 5.14 *We have complete stagnation in GMRES if and only if the matrix A and the right-hand side b can be written as in one of the four cases:*

- (1) $A = W\mathcal{U}P\mathcal{W}^*$, $b = Ax_0 + We_n$,
- (2) $A = WP^T\mathcal{L}\mathcal{W}^*$, $b = Ax_0 + We_n$,
- (3) $A = VP\mathcal{U}V^*$, $b = Ax_0 + Ve_1$,
- (4) $A = V\mathcal{L}P^TV^*$, $b = Ax_0 + Ve_1$.

where W and V are unitary matrices, \mathcal{U} is a nonsingular upper triangular matrix, \mathcal{L} is a nonsingular lower triangular matrix and P is the permutation matrix defined in (5.16).

Proof Case (1) has been proved in Theorem 5.13. Let us consider case (2). If we have complete stagnation for A and b , then we have also complete stagnation for A^* and b . This was shown in [1017] and follows from

$$r_0^* A^j r_0 = r_0^* (A^j)^* r_0 = r_0^* (A^*)^j r_0 = 0, \quad j = 1, \dots, n-1.$$

The conjugate transpose of A is written as $WP^T\mathcal{U}^*W^T$. The matrix P^T is such that $P^T e_1 = e_n$ and $\mathcal{L} = \mathcal{U}^*$ is lower triangular.

Another characterization of the matrices leading to complete stagnation is obtained by using the orthogonal matrix V in the Arnoldi process. From $A = WRQW^*$ and $W = VQ$, we have $A = VQRV^*$ as well as $b = Ax_0 + We_n = Ax_0 + Ve_1$. The matrix Q can be written as $Q = P\check{S}$ where P is defined in (5.16) and \check{S} is a diagonal matrix with $(\check{S})_{j,j} = (\mathcal{S})_{j,j}$, $j = 1, \dots, n-1$ and $(\check{S})_{n,n} = 1$. The matrix \check{S} can be absorbed in the upper triangular matrix by defining $\mathcal{U} = \check{S}\mathcal{R}$. It yields $A = VP\mathcal{U}V^*$ and proves case (3). Case (4) is proved by considering as before the transpose matrix $A^* = V\mathcal{U}^*P^TV^*$ with $\mathcal{L} = \mathcal{U}^*$. \square

Corollary 5.1 *We have complete stagnation in GMRES if and only if the matrix A is $A = VBV^*$ with V unitary, the matrix B being one of the four matrices below and the right-hand sides b are chosen properly,*

- (1) B is an upper Hessenberg matrix with the last column proportional to e_1 and $b = Ax_0 + Ve_n$,
- (2) B is a lower Hessenberg matrix with the last row proportional to e_1^T and $b = Ax_0 + Ve_n$,

- (3) B is an upper Hessenberg matrix with the first row proportional to e_n^T and $b = Ax_0 + Ve_1$,
- (4) B is a lower Hessenberg matrix with the first column proportional to e_n and $b = Ax_0 + Ve_1$.

Proof These results are readily obtained from Theorem 5.14. For instance in case (1) of Theorem 5.14, the multiplication of the upper triangular matrix \mathcal{U} to the right by the matrix P of the form (5.16) gives an upper Hessenberg matrix whose last column is zero except for the first entry. \square

Let us now consider the residual vectors in case of stagnation. To do this we go back to the scaling of the right-hand side we have used before. Since $r_k^G \in r_0 + A\mathcal{K}_k(A, r_0)$ we can write $r_k^G = r_0 + W_{n,k}t_k$. The following result characterizes t_k and the residual vector, which is valid independent of whether GMRES stagnates or not.

Theorem 5.15 *We have*

$$t_k = -\|r_0\| \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_k \end{pmatrix},$$

and

$$r_k^G = \|r_0\| W \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \eta_{k+1} \\ \vdots \\ \eta_n \end{pmatrix}. \quad (5.17)$$

Proof We follow [640], page 297. We can expand r_0 in the basis defined by the columns w_j of W as

$$r_0 = \sum_{j=1}^n (r_0, w_j) w_j.$$

But

$$(r_0, w_j) = \|r_0\| (Wh, We_j) = \|r_0\| (h, W^*We_j) = \|r_0\| \eta_j.$$

Using the orthogonality property of r_k we have

$$0 = W_{n,k}^* r_k^G = W_{n,k}^* (r_0 + W_{n,k}t_k) = W_{n,k}^* r_0 + t_k.$$

This proves the result for t_k . Using this in $r_k^G = r_0 + W_{n,k}t_k$, we obtain

$$r_k^G = \|r_0\| W \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_k \\ \eta_{k+1} \\ \vdots \\ \eta_n \end{pmatrix} - \|r_0\| W \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \|r_0\| W \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \eta_{k+1} \\ \vdots \\ \eta_n \end{pmatrix},$$

and this concludes the proof. \square

As we have seen above, since,

$$\eta_{k+1}^2 = \frac{1}{\|r_0\|^2} (\|r_k^G\|^2 - \|r_{k+1}^G\|^2),$$

having $\|r_k^G\| = \|r_{k+1}^G\|$ implies $\eta_{k+1} = 0$. From Theorem 5.15 it yields $r_{k+1}^G = r_k^G$. Hence, stagnation of the residual norms implies that the residual vectors stay the same. This is also the case for the iterates and the error vectors.

In practice, when solving real-life problems, exact stagnation is a rather rare phenomenon. However, what is more likely to happen are phases of quasi-stagnation where the residual norms still decrease but very slowly. This leads to large values of the FOM residual norms for these iterations.

5.5 Residual norms

In this section we derive exact expressions for the GMRES residual norms as functions of the eigenvalues, eigenvectors (or principal vectors) and right-hand sides. These expressions can be obtained from the general result of Theorem 3.19 using the fact that, for GMRES, the basis vectors are orthonormal.

We will use the notation for index sets introduced in (2.5), which we briefly recall here: \mathcal{I}_ℓ (or \mathcal{J}_ℓ) is defined to be a set of ℓ ordered indices $(i_1, i_2, \dots, i_\ell)$ such that $1 \leq i_1 < \dots < i_\ell \leq n$.

Theorem 5.16 *Let A be a diagonalizable matrix with a spectral factorization $X\Lambda X^{-1}$ where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ contains distinct eigenvalues, let b be the right-hand side and $r_0 = b - Ax_0$ such that $c = X^{-1}r_0$ has no zero components. Then for $k < n$,*

$$\|r_k^G\|^2 = \mu_{k+1}^N / \mu_k^D,$$

where $\mu_1^D = \sum_{i=1}^n \left| \sum_{j=1}^n X_{i,j} c_j \lambda_j \right|^2$, for $k \geq 2$

$$\mu_k^D = \sum_{\mathcal{I}_k} \left| \sum_{\mathcal{J}_k} \det(X_{\mathcal{I}_k, \mathcal{J}_k}) c_{j_1} \cdots c_{j_k} \lambda_{j_1} \cdots \lambda_{j_k} \prod_{j_\ell < j_p \in \mathcal{J}_k} (\lambda_{j_p} - \lambda_{j_\ell}) \right|^2, \quad \text{and}$$

$$\mu_{k+1}^N = \sum_{\mathcal{I}_{k+1}} \left| \sum_{\mathcal{J}_{k+1}} \det(X_{\mathcal{I}_{k+1}, \mathcal{J}_{k+1}}) c_{j_1} \cdots c_{j_{k+1}} \prod_{j_\ell < j_p \in \mathcal{J}_{k+1}} (\lambda_{j_p} - \lambda_{j_\ell}) \right|^2.$$

The summations are over all sets of indices $\mathcal{I}_{k+1}, \mathcal{J}_{k+1}, \mathcal{I}_k, \mathcal{J}_k$ and $X_{\mathcal{I}_\ell, \mathcal{J}_\ell}$ is the submatrix of X whose row and column indices of entries are defined, respectively, by \mathcal{I}_ℓ and \mathcal{J}_ℓ .

Proof The result is obtained straightforwardly from Theorem 3.19 and relation (3.36) noticing that $V^*V = I$. \square

In the case of GMRES, the formula for the residual norm simplifies when the matrix A is normal. The notation is the same as in Theorem 5.16.

Theorem 5.17 *Let A be a normal matrix with a spectral factorization $X\Lambda X^*$ with Λ diagonal (with distinct diagonal entries λ_i) and X unitary and let r_0 be the initial residual vector such that $c = X^*r_0$ has no zero components. Applying GMRES to (A, r_0) , the residual norms are given by*

$$\|r_1^G\|^2 = \frac{\sum_{\mathcal{I}_2} |c_{i_1}|^2 |c_{i_2}| \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_2 \\ i_\ell, i_j \in \mathcal{I}_2}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}{\sum_{i=1}^n |c_i|^2 |\lambda_i|^2}, \quad (5.18)$$

$$\|r_k^G\|^2 = \frac{\sum_{\mathcal{I}_{k+1}} \left[\prod_{j=1}^{k+1} |c_{i_j}|^2 \right] \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_{k+1} \\ i_\ell, i_j \in \mathcal{I}_{k+1}}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}{\sum_{\mathcal{I}_k} \left[\prod_{j=1}^k |c_{i_j}|^2 |\lambda_{i_j}|^2 \right] \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_k \\ i_\ell, i_j \in \mathcal{I}_k}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}. \quad (5.19)$$

Proof Since $X^*X = I$, in the proof of Theorem 3.19 it is enough to use the Cauchy–Binet formula only once. \square

We observe that in the normal case the residual norms depend on the eigenvectors of A only through the projections of the initial residual on the eigenvectors, whereas in the non-normal case they also depend on determinants of submatrices of X . Hence, generally, the dependence on the eigenvectors is stronger in the non-normal case. But, in any case, Theorems 5.16 and 5.17 show exactly how the residual norms depend on the eigenvalues of A .

Formula (5.19) for the GMRES residual norm with a normal matrix, which holds at every iteration step, offers an insight to the fact that outlying eigenvalues can often be associated with an initial stage of slow GMRES convergence (see, for instance, the

example in [638]). From (5.19) we see that if there is one tight cluster of eigenvalues and, say, m other eigenvalues well separated from this cluster, then after $m + 1$ iterations there will be at least one small factor $|\lambda_{i_j} - \lambda_{i_\ell}|^2$ in every summation term of the numerator because $m + 2$ eigenvalues are involved and at least one pair of eigenvalues will belong to the cluster. If the weights are of similar size, one can therefore expect acceleration of convergence after the m initial steps. An analogous argument can be used in the presence of multiple clusters; with ℓ well-separated clusters and another m well-separated single eigenvalues, acceleration might be expected after $m + \ell$ steps. The sharpness of the start of the acceleration as well as the approximate slope of the convergence curve depend on the tightness of the clusters in comparison with the mutual distance of the well-separated eigenvalues.

It should be pointed out that another expression for $\|r_k\|$ was given in [548], Theorem 4.1, using a minimization problem over $k + 1$ eigenvalues. Unfortunately, this expression contains constants which are not specified even though they are bounded from above and from below. In [642], Theorem 2.1, the formula (5.19) was derived for $k = n - 1$.

We observe that when $x_0 = 0$ and $\|b\| = 1$ which can always be obtained by scaling, the numerators in the expressions for the residual norms are always smaller than the denominators.

When the matrix A is only diagonalizable and not normal, the formulas in Theorem 5.16 are quite complicated but we can obtain simpler bounds by using Theorem 3.20. For GMRES we obtain bounds for the residual norm.

Theorem 5.18 *Let A be a diagonalizable matrix, $A = X\Lambda X^{-1}$, with Λ diagonal (with distinct diagonal entries λ_i), the initial residual r_0 such that $c = X^{-1}r_0$ has no zero components and*

$$\mu_k = \frac{\sum_{1 \leq i_1 < \dots < i_{k+1} \leq n} \left[\prod_{j=1}^{k+1} |c_{i_j}|^2 \right] \prod_{i_1 \leq i_\ell < i_p \leq i_{k+1}} |\lambda_{i_p} - \lambda_{i_\ell}|^2}{\sum_{1 \leq i_1 < \dots < i_k \leq n} \left[\prod_{j=1}^k |c_{i_j}|^2 |\lambda_{i_j}|^2 \right] \prod_{i_1 \leq i_\ell < i_p \leq i_k} |\lambda_{i_p} - \lambda_{i_\ell}|^2}.$$

Then,

$$\mu_k [\sigma_{\min}(X)]^2 \leq \|r_k^G\|^2 \leq \mu_k \|X\|^2.$$

We observe that μ_k is the same as the result for normal matrices, except for the definition of c . Note that in these bounds the right-hand side is contained in μ_k through $c = X^{-1}r_0$ and there is no need for $\|X^{-1}\|$ as in many other bounds in the literature. Moreover, if the matrix is “almost” normal in the sense that $\sigma_{\min}(X)$ is not much different from $\|X\|$ then the dependence on the eigenvalues is “almost” the same as for a normal matrix.

The upper bound of Theorem 5.18 is equivalent to the one in [914] which is

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \|X\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda)c\|,$$

with $c = X^{-1}r_0$. Since Λ is diagonal and therefore normal, we can apply the result of Theorem 5.17. The value μ_k defined in Theorem 5.18 is an explicit expression for the minimum $\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda)c\|$.

The expressions for the residual norms in theorems 5.16 and 5.17 are quite intricate since the number of terms in a sum over an index set \mathcal{I}_k is $\binom{n}{k}$. Their main merit is they are exact.

We would like to see how Theorem 5.16 can be extended to the non-diagonalizable case by considering the situation where the Jordan canonical form of A has one Jordan block only. Let $A = XJX^{-1}$ with $J = \text{bidiag}(\lambda, 1)$ for a nonzero eigenvalue λ and let r_0 be the initial residual such that the components of $c = X^{-1}r_0$ are nonzero (otherwise GMRES terminates before the n th iteration). Then the moment matrix is $M = (1/\|r_0\|^2)\tilde{M}$ with

$$\tilde{M} = K^*K = (c \ Jc \ \dots \ J^{n-1}c)^* X^*X (c \ Jc \ \dots \ J^{n-1}c).$$

In contrast with the Krylov matrix $(c \ \Lambda c \ \dots \ \Lambda^{n-1}c) = D_c \mathcal{V}_n$, the Krylov matrix $(c \ Jc \ \dots \ J^{n-1}c)$ cannot be written as the product of a diagonal matrix containing the entries of c with a Vandermonde matrix. Instead, it can be factorized as

$$(c \ Jc \ \dots \ J^{n-1}c) = CE \equiv \quad (5.20)$$

$$\begin{pmatrix} c_1 & c_2 & \dots & \dots & c_n \\ c_2 & c_3 & \dots & \dots & c_n \\ c_3 & \dots & c_n \\ \vdots & c_n \\ c_n \end{pmatrix} \begin{pmatrix} 1 & \lambda & \lambda^2 & \dots & \lambda^{n-1} \\ 0 & 1 & 2\lambda & \dots & \binom{n-1}{1} \lambda^{n-2} \\ 0 & 0 & 1 & \dots & \binom{n-1}{2} \lambda^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix},$$

where the matrix C is a Hankel “anti upper triangular” matrix defined by $c_1, \dots, c_n, 0, \dots, 0$. Here is a small example for illustration: Let $n = 5$ and let all entries of $c = X^{-1}r_0$ be nonzero. Then the Krylov matrix $(c \ Jc \ \dots \ J^4c)$ is

$$\begin{pmatrix} c_1 & \lambda c_1 + c_2 & \lambda^2 c_1 + 2\lambda c_2 + c_3 & \lambda^3 c_1 + 3\lambda^2 c_2 + 3\lambda c_3 + c_4 & \lambda^4 c_1 + 4\lambda^3 c_2 + 6\lambda^2 c_3 + 4\lambda c_4 + c_5 \\ c_2 & \lambda c_2 + c_3 & \lambda^2 c_2 + 2\lambda c_3 + c_4 & \lambda^3 c_2 + 3\lambda^2 c_3 + 3\lambda c_4 + c_5 & \lambda^4 c_2 + 4\lambda^3 c_3 + 6\lambda^2 c_4 + 4\lambda c_5 \\ c_3 & \lambda c_3 + c_4 & \lambda^2 c_3 + 2\lambda c_4 + c_5 & \lambda^3 c_3 + 3\lambda^2 c_4 + 3\lambda c_5 & \lambda^4 c_3 + 4\lambda^3 c_4 + 6\lambda^2 c_5 \\ c_4 & \lambda c_4 + c_5 & \lambda^2 c_4 + 2\lambda c_5 & \lambda^3 c_4 + 3\lambda^2 c_5 & \lambda^4 c_4 + 4\lambda^3 c_5 \\ c_5 & \lambda c_5 & \lambda^2 c_5 & \lambda^3 c_5 & \lambda^4 c_5 \end{pmatrix}$$

with the factorization

$$(c \ Jc \ \cdots \ J^4c) = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \\ c_2 & c_3 & c_4 & c_5 \\ c_3 & c_4 & c_5 \\ c_4 & c_5 \\ c_5 \end{pmatrix} \begin{pmatrix} 1 & \lambda & \lambda^2 & \lambda^3 & \lambda^4 \\ & 1 & 2\lambda & 3\lambda^2 & 4\lambda^3 \\ & & 1 & 3\lambda & 6\lambda^2 \\ & & & 1 & 4\lambda \\ & & & & 1 \end{pmatrix}.$$

The $(k+1)$ st leading principal submatrix \tilde{M}_{k+1} of \tilde{M} is given by

$$\tilde{M}_{k+1} = (c \ Jc \ \cdots \ J^k c)^* X^* X (c \ Jc \ \cdots \ J^k c).$$

With (5.20) and defining

$$Y \equiv XC,$$

we have

$$\tilde{M}_{k+1} = (E_{:,1:k+1})^* (XC)^* XCE_{:,1:k+1} = (E_{:,1:k+1})^* Y^* YE_{:,1:k+1},$$

which can be written as the product $\tilde{M}_{k+1} = F^* F$ of two rectangular matrices where $F \equiv YE_{:,1:k+1}$. The matrix $E_{:,1:k+1}$ depends only on the eigenvalue, the matrix Y contains all information from the principal vectors and the right-hand side. Using exactly the same proof technique as for Theorem 5.16, we obtain for a single Jordan block the following result.

Theorem 5.19 *Let A be a nonsingular matrix with a single eigenvalue λ and with Jordan form XJX^{-1} where $J = \text{bidiag}(\lambda, 1)$. Let r_0 be the initial residual such that the last entry of $c = X^{-1}r_0$ is nonzero, let E be the eigenvalue matrix defined by (5.20) and let $Y = XC$, where C is the Hankel matrix defined in (5.20). When solving $Ax = b$ the GMRES residual norm at iteration $k < n$ satisfies*

$$\|r_k^G\|^2 = \frac{\sum_{\mathcal{I}_{k+1}} |\sum_{\mathcal{J}_{k+1}} \det(Y_{\mathcal{I}_{k+1}, \mathcal{J}_{k+1}}) \det(E_{\mathcal{J}_{k+1}, 1:k+1})|^2}{\sum_{\mathcal{I}_k} |\sum_{\mathcal{J}_k} \det(Y_{\mathcal{I}_k, \mathcal{J}_k}) \det(E_{\mathcal{J}_k, 2:k+1})|^2}. \quad (5.21)$$

The numerator of $\|r_k^G\|^2$ contains the determinants of $E_{\mathcal{J}_{k+1}, 1:k+1}$ for all index sets \mathcal{J}_{k+1} . Their values are given in the following result.

Lemma 5.3 *For all the sets of $k+1$ indices \mathcal{J}_{k+1} in the numerator of (5.21), the only determinant of $E_{\mathcal{J}_{k+1}, 1:k+1}$ which is nonzero is $\det(E_{1:k+1, 1:k+1}) = 1$.*

Proof We have to consider all the sets of indices j_ℓ such that $1 \leq j_1 < \cdots < j_{k+1} \leq n$. Since E is upper triangular, all the determinants involving a row of index larger than $k+1$ are zero. The only set of indices \mathcal{J}_{k+1} without a row of index larger than $k+1$ is $\{1, 2, \dots, k+1\}$. The corresponding submatrix is triangular with ones on the diagonal. \square

From Lemma 5.3 there is only one term for the sum over \mathcal{J}_{k+1} in the numerator μ_k^N in (5.21) and

$$\mu_k^N = \sum_{\mathcal{I}_{k+1}} |\det(Y_{\mathcal{I}_{k+1}, 1:k+1})|^2.$$

We remark that in this case the numerator does not depend on the eigenvalue. For the denominator in (5.21) we are interested in the determinants of $E_{\mathcal{J}_k, 2:k+1}$. They are characterized in the following lemma.

Lemma 5.4 *The $k + 1$ nonzero determinants of $E_{\mathcal{J}_k, 2:k+1}$ are obtained for the sets of indices \mathcal{J}_k not containing an index strictly larger than $k + 1$. Enumerating those sets in lexicographic order, the determinants are, respectively, $\lambda^k, \lambda^{k-1}, \dots, \lambda, 1$. Moreover, the denominator μ_k^D for $\|r_k\|^2$ in (5.21) is*

$$\mu_k^D = \sum_{\mathcal{I}_k} \left| \lambda^k \det(Y_{\mathcal{I}_k, \hat{\mathcal{I}}_1}) + \cdots + \lambda \det(Y_{\mathcal{I}_k, \hat{\mathcal{I}}_k}) + \det(Y_{\mathcal{I}_k, \hat{\mathcal{I}}_{k+1}}) \right|^2,$$

where $\hat{\mathcal{I}}_j$, $j = 1, \dots, k + 1$, are the sets of indices with k elements in the ordered combinations of $k + 1$ elements enumerated in lexicographic ordering.

Proof The first claim is obvious since if there is a row index strictly larger than $k + 1$ in \mathcal{J}_k then there is a zero row in the matrix $E_{\mathcal{J}_k, 2:k+1}$ and the determinant is zero. The proof of the second claim is by induction on k . For $k = 1$ the only nonzero determinants of $E_{1,2}$ are, in lexicographical order, $\det(E_{1,2}) = E_{1,2} = \lambda$ and $\det(E_{2,2}) = E_{2,2} = 1$. Let us assume that the claim is true for $k - 1$. We have to consider the determinants of submatrices of order k of the $n \times k$ matrix

$$E_{:,2:k+1} = \begin{pmatrix} \lambda & \lambda^2 & \cdots & \lambda^{k-1} & \lambda^k \\ 1 & 2\lambda & \cdots & \binom{k-1}{1} \lambda^{k-2} & \binom{k}{1} \lambda^{k-1} \\ 0 & 1 & \cdots & \binom{k-1}{2} \lambda^{k-3} & \binom{k}{2} \lambda^{k-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \binom{k}{k-1} \lambda \\ 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

In lexicographic order the first set of indices \mathcal{J}_k is $\{1, 2, \dots, k\}$. We have to consider the determinant of the matrix $E^{(k)}$ obtained from the first k rows of $E_{:,2:k+1}$. Let us compute this determinant using the last column. It is equal to

$$\begin{aligned} (-1)^{k+1} [\lambda^k \det(E_{-1,1:k-1}^{(k)}) - \binom{k}{1} \lambda^{k-1} \det(E_{-2,1:k-1}^{(k)}) \\ - \cdots + (-1)^{k-1} \binom{k}{k-1} \lambda \det(E_{-k,1:k-1}^{(k)})], \end{aligned}$$

where $\det(E_{-j,1:k-1}^{(k)})$ denotes the determinant of the square submatrix of order $k-1$ of $E^{(k)}$ from columns 1 to $k-1$ with row j removed. Those determinants are given by our induction hypothesis (in reverse order); they are $1, \lambda, \dots, \lambda^{k-1}$. Therefore we can factor λ^k in the expression displayed above and we obtain

$$(-1)^{k+1} \lambda^k [1 - \binom{k}{1} + \binom{k}{2} - \cdots + (-1)^{k-1} \binom{k}{k-1}].$$

We observe that the sum within brackets is equal to $(-1)^{k+1}$ and thus the determinant we were looking for is λ^k . The proof for the other sets of indices \mathcal{J}_k is along the same lines. \square

Combining Lemmas 5.4 and 5.3 with Theorem 5.19, we obtain the next result.

Theorem 5.20 *Let A be a nonsingular matrix with a single eigenvalue λ and with Jordan canonical XJX^{-1} where $J = \text{bidiag}(\lambda, 1)$. Let r_0 be the initial residual such that the last entry of $c = X^{-1}r_0$ is nonzero, let E be the eigenvalue matrix defined by (5.20) and let $Y = XC$, where C is as defined in (5.20). When solving $Ax = b$ the GMRES residual norm at iteration $k < n$ satisfies*

$$\|r_k\|^2 = \frac{\sum_{\mathcal{I}_{k+1}} |\det(Y_{\mathcal{I}_{k+1},1:k+1})|^2}{\sum_{\mathcal{I}_k} |\lambda^k \det(Y_{\mathcal{I}_k,\hat{\mathcal{I}}_1}) + \cdots + \lambda \det(Y_{\mathcal{I}_k,\hat{\mathcal{I}}_k}) + \det(Y_{\mathcal{I}_k,\hat{\mathcal{I}}_{k+1}})|^2}, \quad (5.22)$$

where $\hat{\mathcal{I}}_j$, $j = 1, \dots, k+1$ are the sets of indices with k elements in the ordered combinations of $k+1$ elements enumerated in lexicographic ordering.

A result for the residual norms generated by GMRES applied to a Jordan block is given in [548]. The expression in that paper contains constants whose values are generally unknown.

We observe from Theorem 5.20 an interesting, slightly enhanced independence from the spectrum in comparison with diagonalizable matrices: The numerator is fully independent from the eigenvalue and so are the summands $\det(Y_{\mathcal{I}_k,\hat{\mathcal{I}}_{k+1}})$ in the denominator.

The generalization of the previous results to multiple Jordan blocks is straightforward. Let A be a matrix with the Jordan canonical form XJX^{-1} and m ($m \leq n$) distinct eigenvalues denoted as $\lambda_1, \lambda_2, \dots, \lambda_m$. We assume that A is nonderogatory because we consider GMRES processes that do not terminate before iteration n . Let the size of the Jordan block J_i corresponding to λ_i be n_i , i.e., $\sum_{i=1}^m n_i = n$, and let us

denote by s_i , $i = 1, \dots, m$ the index of the row where the block J_i starts, to which we add $s_{m+1} = n + 1$. The block J_i goes from row s_i to row $s_{i+1} - 1$. To avoid early termination, we also assume that the initial residual r_0 is such that the entries on positions $s_{i+1} - 1$, $1 \leq i \leq m$, of $c = X^{-1}r_0$ are nonzero.

As above, we have

$$\tilde{M} = K^*K = (c \ Jc \ \cdots \ J^{n-1}c)^* X^*X (c \ Jc \ \cdots \ J^{n-1}c).$$

For multiple Jordan blocks, the factorization (5.20) has to be modified as follows. If we define the rows s_i to $s_{i+1} - 1$ of E corresponding to the eigenvalue λ_i as

$$E_{s_i:s_{i+1}-1,:} \equiv \begin{pmatrix} 1 & \lambda_i & \lambda_i^2 & \cdots & \lambda_i^{n_i-1} & \cdots & \lambda_i^{n-1} \\ 0 & 1 & 2\lambda_i & \cdots & \binom{n_i-1}{1} \lambda_i^{n_i-2} & \cdots & \binom{n-1}{1} \lambda_i^{n-2} \\ 0 & 0 & 1 & \cdots & \binom{n_i-1}{2} \lambda_i^{n_i-3} & \cdots & \binom{n-2}{2} \lambda_i^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & \binom{n-1}{n_i-1} \lambda_i^{n-n_i} \end{pmatrix}$$

and the corresponding diagonal block of C as

$$C_{s_i:s_{i+1}-1,s_i:s_{i+1}-1} \equiv \begin{pmatrix} c_{s_i} & c_{s_i+1} & \cdots & \cdots & c_{s_{i+1}-1} \\ c_{s_i+1} & c_{s_i+2} & \cdots & & c_{s_{i+1}-1} \\ c_{s_i+2} & \cdots & c_{s_{i+1}-1} \\ \vdots & & c_{s_{i+1}-1} \\ c_{s_{i+1}-1} & & & & \end{pmatrix},$$

then

$$(c \ Jc \ \cdots \ J^{n-1}c) = CE.$$

The matrix C is block diagonal with Hankel anti-upper triangular diagonal blocks of order n_i . We again give an example to illustrate this.

Consider a matrix $A = XJX^{-1}$ of order 5 with J defined as

$$J = \begin{pmatrix} \lambda & 1 & & & \\ & \lambda & 1 & & \\ & & \lambda & & \\ & & & \mu & 1 \\ & & & & \mu \end{pmatrix}, \quad (5.23)$$

where λ and μ ($\lambda \neq \mu$) are given complex numbers different from 0. Let $c = X^{-1}r_0$ and let c have no zero entries. Then the Krylov matrix $(c J c \cdots J^{n-1} c)$ is

$$\begin{pmatrix} c_1 & \lambda c_1 + c_2 & \lambda^2 c_1 + 2\lambda c_2 + c_3 & \lambda^3 c_1 + 3\lambda^2 c_2 + 3\lambda c_3 & \lambda^4 c_1 + 4\lambda^3 c_2 + 6\lambda^2 c_3 \\ c_2 & \lambda c_2 + c_3 & \lambda^2 c_2 + 2\lambda c_3 & \lambda^3 c_2 + 3\lambda^2 c_3 & \lambda^4 c_2 + 4\lambda^3 c_3 \\ c_3 & \lambda c_3 & \lambda^2 c_3 & \lambda^3 c_3 & \lambda^4 c_3 \\ c_4 & \mu c_4 + c_5 & \mu^2 c_4 + 2\mu c_5 & \mu^3 c_4 + 3\mu^2 c_5 & \mu^4 c_4 + 4\mu^3 c_5 \\ c_5 & \mu c_5 & \mu^2 c_5 & \mu^3 c_5 & \mu^4 c_5 \end{pmatrix}$$

and can be factorized as

$$(c J c \cdots J^{n-1} c) = \begin{pmatrix} c_1 & c_2 & c_3 & 0 & 0 \\ c_2 & c_3 & 0 & 0 & 0 \\ c_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_4 & c_5 \\ 0 & 0 & 0 & c_5 & 0 \end{pmatrix} \begin{pmatrix} 1 & \lambda & \lambda^2 & \lambda^3 & \lambda^4 \\ 0 & 1 & 2\lambda & 3\lambda^2 & 4\lambda^3 \\ 0 & 0 & 1 & 3\lambda & 6\lambda^2 \\ 1 & \mu & \mu^2 & \mu^3 & \mu^4 \\ 0 & 1 & 2\mu & 3\mu^2 & 4\mu^3 \end{pmatrix},$$

with a block diagonal matrix C .

Let, as above,

$$Y \equiv XC.$$

Then if the $(k+1)$ st leading principal submatrix \tilde{M}_{k+1} of \tilde{M} is written as

$$\begin{aligned} \tilde{M}_{k+1} &= (c J c \cdots J^k c)^* X^* X (c J c \cdots J^k c), \\ &= E_{:,1:k+1}^* C^* X^* X C E_{:,1:k+1}, \\ &= (YE_{:,1:k+1})^* YE_{:,1:k+1}, \end{aligned}$$

we obtain the formula

$$\|r_k^G\|^2 = \frac{\sum_{\mathcal{I}_{k+1}} |\sum_{\mathcal{J}_{k+1}} \det(Y_{\mathcal{I}_{k+1}, \mathcal{J}_{k+1}}) \det(E_{\mathcal{J}_{k+1}, 1:k+1})|^2}{\sum_{\mathcal{I}_k} |\sum_{\mathcal{J}_k} \det(Y_{\mathcal{I}_k, \mathcal{J}_k}) \det(E_{\mathcal{J}_k, 2:k+1})|^2}. \quad (5.24)$$

This formula is the same as the one presented in Theorem 5.19, but of course, here Y and E are generalizations of the Y and E in Theorem 5.19. The matrix E represents all the influence of the eigenvalues and Y all the influence of the eigenvectors, principal vectors and right-hand side. A difference is that the interplay between the distinct eigenvalues will play a role. The determinants of $E_{\mathcal{J}_{k+1}, 1:k+1}$ and $E_{\mathcal{J}_k, 2:k+1}$ may contain eigenvalue differences.

5.6 The residual polynomials

In this section we derive exact expressions for the coefficients of the FOM and GMRES residual polynomials for diagonalizable matrices as functions of the eigen-

values and eigenvectors (and the initial residual vector). This will give us explicit expressions for the solution of the minimization problem (5.1). Expressions for the value of the minimum were given in the previous section.

The residual vectors r_k^F in FOM and r_k^G in GMRES at iteration k can be expressed as polynomials of degree k in A applied to the initial residual r_0 ,

$$r_k^F = p_k^F(A)r_0, \quad r_k^G = p_k^G(A)r_0.$$

These polynomials are so-called *residual polynomials* and they have a value 1 at 0. The roots of p_k^F (the so-called Ritz values) are the eigenvalues of the upper Hessenberg matrix H_k . When the matrix H_k is nonsingular, the roots of p_k^G (the so-called harmonic Ritz values) are the eigenvalues of the upper Hessenberg matrix $\hat{H}_k = H_k + h_{k+1,k}H_k^{-*}e_k e_k^T$ defined in Chapter 3.

Let A be diagonalizable with $A = X\Lambda X^{-1}$. We have seen from relation (2.36) of Lemma 2.4 that the matrix $K_{n,k}$ can be factorized as

$$K_{n,k} = (r_0, Ar_0, \dots, A^{k-1}r_0) = X(c, \Lambda c, \dots, \Lambda^{k-1}c) = XD_c\mathcal{V}_{n,k},$$

where D_c is a diagonal matrix with the components of $c = X^{-1}r_0$ on the diagonal and $\mathcal{V}_{n,k}$ is an $n \times k$ Vandermonde matrix.

Let us first assume that A is normal with $A = X\Lambda X^*$. Then $c = X^*r_0$. The eigenvalues $\theta_j^{(k)}$ of H_k are the roots of its characteristic polynomial. We have $K_{n,k} = \|r_0\| V_{n,k} U_k$, but we can absorb the multiplying factor in U_k . Therefore we will write $K_{n,k} = V_{n,k} U_k$ and we have

$$\begin{aligned} 0 &= \det(H_k - \theta I) = \det(V_{n,k}^* A V_{n,k} - \theta I), \\ &= \det(U_k^{-*} K_{n,k}^* A K_{n,k} U_k^{-1} - \theta I), \\ &= \det(U_k^{-*} [K_{n,k}^* A K_{n,k} - \theta U_k^* U_k] U_k^{-1}), \\ &= \det(M_k^{-1}) \det(K_{n,k}^* X \Lambda X^* K_{n,k} - \theta M_k), \\ &= \frac{1}{\det(M_k)} \det(\mathcal{V}_{n,k}^* D_c^* \Lambda D_c \mathcal{V}_{n,k} - \theta \mathcal{V}_{n,k}^* D_c^* D_c \mathcal{V}_{n,k}), \end{aligned}$$

with $M_k = U_k^* U_k$. Clearly $p_k(\theta) = \det(\mathcal{V}_{n,k}^* D_c^* \Lambda D_c \mathcal{V}_{n,k} - \theta \mathcal{V}_{n,k}^* D_c^* D_c \mathcal{V}_{n,k})$ is a polynomial of degree k in θ . We observe that the two polynomials on the left- and right-hand sides are the same up to a constant factor. The matrix M_k can only be singular when at least one diagonal entry of U_k is zero, but this corresponds to H_k being reducible.

The following theorem gives a closed-form expression of the polynomial p_k .

Theorem 5.21 *Let A be a normal matrix with the spectral factorization $A = X\Lambda X^*$, where Λ has distinct diagonal entries and the initial residual vector r_0 is such that $c = X^*r_0$ has no zero components. Then, the polynomial p_k whose roots are the Ritz values at iteration k is given by*

$$p_k(\theta) = \sum_{\mathcal{I}_k} (\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2. \quad (5.25)$$

The characteristic polynomial \tilde{p}_k^F of H_k (which is monic) is obtained by dividing p_k by

$$(-1)^k \sum_{\mathcal{I}_k} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2 = (-1)^k \det(M_k).$$

The residual polynomial p_k^F for FOM is obtained by dividing p_k by

$$\sum_{\mathcal{I}_k} \lambda_{i_1} \cdots \lambda_{i_k} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

if this quantity is different from zero. If it is zero, then $p_k(0) = 0$, the matrix H_k is singular and the FOM iterate is not defined.

Proof To simplify the notation, let $B = D_c \mathcal{V}_{n,k}$. Then, the Ritz values are solutions of the equation

$$\det(B^* \Lambda B - \theta B^* B) = \det(B^*(\Lambda - \theta I)B) = 0.$$

Let $G = (\Lambda - \theta I)B$, we have $\det(B^* G) = 0$. We need to compute the determinant of the product of two rectangular matrices which are, respectively, $k \times n$ and $n \times k$. We use the Cauchy–Binet formula (see Chapter 2) to get

$$\det(B^* G) = \sum_{\mathcal{I}_k} \overline{\det(B_{\mathcal{I}_k,:})} \det(G_{\mathcal{I}_k,:}),$$

where \mathcal{I}_k is defined above and $B_{\mathcal{I}_k,:}$ (resp. $G_{\mathcal{I}_k,:}$) is the submatrix of B (resp. G) constructed with the rows in \mathcal{I}_k and all the k columns. We have

$$\det(B_{\mathcal{I}_k,:}) = c_{i_1} \cdots c_{i_k} \prod_{i_1 \leq i_p < i_q \leq i_k} (\lambda_{i_q} - \lambda_{i_p}),$$

and, since $\Lambda - \theta I$ is diagonal,

$$\det(G_{\mathcal{I}_k,:}) = (\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta) c_{i_1} \cdots c_{i_k} \prod_{i_1 \leq i_p < i_q \leq i_k} (\lambda_{i_q} - \lambda_{i_p}). \quad (5.26)$$

Therefore, we have the new polynomial,

$$\det(B^* G) = \sum_{\mathcal{I}_k} (\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{i_1 \leq i_p < i_q \leq i_k} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

whose roots are the Ritz values at iteration k . We remark that the product $(\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta)$ is a polynomial of degree k in θ whose roots are k of the eigenvalues of A . Hence, p_k is a sum of polynomials whose roots are eigenvalues whose indices are given by all possible ordered subsets of k elements of $\{1, \dots, n\}$. We have

$$(\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta) = \sum_{j=0}^k (-1)^{k-j} e_{(j)}(\lambda_{i_1}, \dots, \lambda_{i_k}) \theta^{k-j},$$

where $e_{(j)}$ is a symmetric elementary polynomial that is

$$e_{(j)}(\mu_1, \dots, \mu_k) = \sum_{1 \leq j_1 < j_2 < \dots < j_j \leq k} \mu_{j_1} \cdots \mu_{j_j}, \quad e_{(0)} \equiv 1.$$

The polynomial p_k is not monic nor with a value 1 at 0, but it is a constant multiple of the characteristic polynomial of H_k . To obtain a monic polynomial (the characteristic polynomial of H_k) we have to divide p_k by the coefficient of θ^k which is

$$(-1)^k \sum_{\mathcal{I}_k} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2 = (-1)^k \det(M_k).$$

To obtain a value 1 at the origin (the FOM residual polynomial p_k^F) we have to divide p_k by the constant coefficient which is

$$\sum_{\mathcal{I}_k} \lambda_{i_1} \cdots \lambda_{i_k} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

provided that it is nonzero. The other coefficients of p_k involve the symmetric elementary polynomials; the coefficient of θ^{k-j} , $j = 0, \dots, k$ is

$$(-1)^{k-j} \sum_{\mathcal{I}_k} e_{(j)}(\lambda_{i_1}, \dots, \lambda_{i_k}) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

and this concludes the proof. \square

According to Theorem 5.21 the FOM residual vector at iteration k is

$$\begin{aligned} r_k^F &= p_k^F(A)r_0, \\ &= Xp_k^F(\Lambda)X^*r_0, \\ &= \sum_{j=1}^n c_j p_k^F(\lambda_j) x^{(j)}, \end{aligned}$$

where $x^{(j)}$ is the column of X corresponding to the eigenvalue λ_j and $c = X^*r_0$ is the vector whose components are the projections of r_0 on the eigenvectors. By orthogonality of the eigenvectors we also have $(x^{(j)})^T r_k^F = p_k^F(\lambda_j)c_j$. Let

$$\mu_0 = \sum_{\mathcal{I}_k} \lambda_{i_1} \cdots \lambda_{i_k} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2 \neq 0.$$

The value of the polynomial p_k^F at the eigenvalue λ_j is

$$p_k^F(\lambda_j) = \frac{1}{\mu_0} \sum_{\tilde{\mathcal{I}}_k} (\lambda_{i_1} - \lambda_j) \cdots (\lambda_{i_k} - \lambda_j) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \tilde{\mathcal{I}}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

where the sum is over all ordered set of indices (i_1, i_2, \dots, i_k) with $i_\ell \neq j$, $\ell = 1, \dots, k$. Hence, $p_k^F(\lambda_j)$ depends on the distances of λ_j to all the other eigenvalues and also on the pairwise distances of these other eigenvalues. Note that the knowledge of the coefficients of the polynomial p_k^F gives the decomposition of the residual vector r_k^F on the natural basis of the Krylov subspace $\mathcal{K}_k(A, r_0)$.

Let us see how to generalize these results to the case A diagonalizable with $A = X\Lambda X^{-1}$. The only difference is that we have to use the Cauchy–Binet formula twice. But unfortunately the formulas become much more intricate.

Theorem 5.22 *Let A be a diagonalizable matrix with the spectral factorization $A = X\Lambda X^{-1}$, where Λ has distinct diagonal entries and the initial residual vector r_0 is such that $c = X^{-1}r_0$ has no zero components. Then, the polynomial p_k whose roots are the Ritz values at iteration k is given by*

$$p_k(\theta) = \sum_{\mathcal{I}_k} (\lambda_{i_1} - \theta) \cdots (\lambda_{i_k} - \theta) c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}.$$

The characteristic polynomial \tilde{p}_k^F of H_k is obtained by dividing p_k by

$$(-1)^k \sum_{\mathcal{I}_k} c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}.$$

The residual polynomial p_k^F for FOM is obtained by dividing p_k by

$$\begin{aligned} & \sum_{\mathcal{I}_k} c_{i_1} \cdots c_{i_k} \lambda_{i_1} \cdots \lambda_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ & \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^* X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}, \end{aligned}$$

if this quantity is different from zero.

Proof The proof is essentially the same as for Theorem 5.21; we just outline the differences. Now, the equation for the Ritz values is

$$\begin{aligned} 0 = \det(H_k - \theta I) &= \det(M_k^{-1}) \det(K_{n,k}^* X \Lambda X^{-1} K_{n,k} - \theta M_k), \\ &= \frac{1}{\det(M_k)} \det(\mathcal{V}_{n,k}^* D_c^*(X^* X) \Lambda D_c \mathcal{V}_{n,k} - \theta \mathcal{V}_{n,k}^* D_c^*(X^* X) D_c \mathcal{V}_{n,k}). \end{aligned}$$

Using the same notation as above the equation reads

$$\det(B^*(X^* X) \Lambda B - \theta B^*(X^* X) B) = \det(B^*(X^* X)(\Lambda - \theta I)B) = 0.$$

Let us define $G = (\Lambda - \theta I)B$. Then we have $\det(B^*(X^* X)G) = 0$. Once again we can use the Cauchy–Binet formula to obtain

$$\det(B^*(X^* X)G) = \sum_{\mathcal{I}_k} \overline{\det((X^* X B)_{\mathcal{I}_k, :})} \det(G_{\mathcal{I}_k, :}).$$

As before, $\det(G_{\mathcal{I}_k, :})$ is given by (5.26). For $(X^* X B)_{\mathcal{I}_k, :}$ we use again the Cauchy–Binet formula. It yields

$$\det((X^* X B)_{\mathcal{I}_k, :}) = \sum_{\mathcal{J}_k} \det((X^* X)_{\mathcal{I}_k, \mathcal{J}_k}) c_{j_1} \cdots c_{j_k} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\lambda_{j_q} - \lambda_{j_p}),$$

and this concludes the proof. \square

Let $\mu_0 = p_k(0) \neq 0$. Then, the residual vector is $r_k^F = \sum_{j=1}^n c_j p_k^F(\lambda_j) x^{(j)}$ with $c = X^{-1} r_0$ and

$$p_k^F(\lambda_j) = \frac{1}{\mu_0} \sum_{\mathcal{I}_k} (\lambda_{i_1} - \lambda_j) \cdots (\lambda_{i_k} - \lambda_j) c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p})$$

$$\times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^* X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}.$$

We now consider the polynomials related to GMRES. They are the characteristic polynomials of

$$\hat{H}_k = H_k + h_{k+1,k}^2 H_k^{-*} e_k e_k^T,$$

if H_k is nonsingular. From the definition of the harmonic Ritz values and vectors, its roots are given by the solutions of the generalized eigenproblem

$$\underline{H}_k^* \underline{H}_k x = \zeta H_k^* x,$$

for ζ and x .

We have $H_k = V_{n,k}^* A V_{n,k}$ and $\underline{H}_k = V_{n,k+1}^* A V_{n,k}$. It yields

$$V_{n,k}^* A^* (V_{n,k+1} V_{n,k+1}^*) A V_{n,k} x = \zeta V_{n,k}^* A^* V_{n,k} x.$$

But, $V_{n,k+1} V_{n,k+1}^*$ is the orthogonal projector on the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$. Since $A V_{n,k} x$ is in this subspace, we have $(V_{n,k+1} V_{n,k+1}^*) A V_{n,k} x = A V_{n,k} x$ and

$$V_{n,k}^* A^* A V_{n,k} x = \zeta V_{n,k}^* A^* V_{n,k} x.$$

When the matrix A is normal this leads to the following result.

Theorem 5.23 *Let A be a normal matrix with the spectral factorization $A = X \Lambda X^*$, where Λ has distinct diagonal entries and the initial residual vector r_0 is such that $c = X^* r_0$ has no zero components. Then, the polynomial q_k whose roots are the harmonic Ritz values at iteration k is given by*

$$\sum_{\mathcal{I}_k} (|\lambda_{i_1}|^2 - \zeta \overline{\lambda_{i_1}}) \cdots (|\lambda_{i_k}|^2 - \zeta \overline{\lambda_{i_k}}) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2. \quad (5.27)$$

The characteristic polynomial \tilde{p}_k^G of \hat{H}_k is obtained by dividing q_k by

$$(-1)^k \sum_{\mathcal{I}_k} \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2.$$

The residual polynomial p_k^G for GMRES is obtained by dividing q_k by

$$\sum_{\mathcal{I}_k} |\lambda_{i_1}|^2 \cdots |\lambda_{i_k}|^2 |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2.$$

Proof With the same notation as in Theorem 5.21 the equation for the harmonic Ritz values is

$$\det(B^* \Lambda^* \Lambda B - \zeta B^* \Lambda^* B) = \det(B^*(\Lambda^* \Lambda - \zeta \Lambda^*)B) = 0.$$

Let $G = (\Lambda^* \Lambda - \zeta \Lambda^*)B$, we have $\det(B^* G) = 0$. Therefore, the polynomial q_k is

$$\det(B^* G) = \sum_{\mathcal{I}_k} (|\lambda_{i_1}|^2 - \zeta \overline{\lambda_{i_1}}) \cdots (|\lambda_{i_k}|^2 - \zeta \overline{\lambda_{i_k}}) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \notin \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2.$$

We have

$$(|\lambda_{i_1}|^2 - \zeta \overline{\lambda_{i_1}}) \cdots (|\lambda_{i_k}|^2 - \zeta \overline{\lambda_{i_k}}) = \sum_{j=0}^k (-1)^{k-j} e_{(j)}(\lambda_{i_1}, \dots, \lambda_{i_k}) \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} \zeta^{k-j}.$$

The coefficient of ζ^{k-j} , $j = 0, \dots, k$ in the polynomial q_k is

$$(-1)^{k-j} \sum_{\mathcal{I}_k} e_{(j)}(\lambda_{i_1}, \dots, \lambda_{i_k}) \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2.$$

From this we can obtain the characteristic polynomial of \hat{H}_k and the GMRES residual polynomial dividing by the appropriate coefficient. \square

The residual vector at iteration k is $r_k^G = \sum_{j=1}^n c_j p_k^G(\lambda_j) x^{(j)}$ with $c = X^* r_0$ and

$$p_k^G(\lambda_j) = \frac{1}{\gamma_0} \sum_{\hat{\mathcal{I}}_k} (|\lambda_{i_1}|^2 - \lambda_j \overline{\lambda_{i_1}}) \cdots (|\lambda_{i_k}|^2 - \lambda_j \overline{\lambda_{i_k}}) |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

with

$$\gamma_0 = \sum_{\mathcal{I}_k} |\lambda_{i_1}|^2 \cdots |\lambda_{i_k}|^2 |c_{i_1}|^2 \cdots |c_{i_k}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2.$$

Let us now assume that A is diagonalizable. Then we have the following result. The proof is essentially the same as what we did for the FOM residual polynomial.

Theorem 5.24 Let A be a diagonalizable matrix with the spectral factorization $A = X\Lambda X^{-1}$, where Λ has distinct diagonal entries and the initial residual vector r_0 is such that $c = X^{-1}r_0$ has no zero components. Then, the polynomial q_k whose roots are the harmonic Ritz values at iteration k is given by

$$q_k(\zeta) = \sum_{\mathcal{I}_k} (\lambda_{i_1} - \zeta) \cdots (\lambda_{i_k} - \zeta) \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}.$$

The characteristic polynomial \tilde{p}_k^G of \hat{H}_k is obtained by dividing q_k by

$$(-1)^k \sum_{\mathcal{I}_k} \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\},$$

The residual polynomial p_k^G for GMRES is obtained by dividing q_k by

$$\sum_{\mathcal{I}_k} |\lambda_{i_1}|^2 \cdots |\lambda_{i_k}|^2 c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p}) \\ \times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}.$$

Proof The equation for the harmonic Ritz values is

$$\det(B^*(X^*X)\Lambda^*\Lambda B - \zeta B^*(X^*X)\Lambda^*B) = \det(B^*(X^*X)(\Lambda^*\Lambda - \zeta\Lambda^*)B) = 0.$$

If we choose $G = (\Lambda^*\Lambda - \zeta\Lambda^*)B$, we have $\det(B^*(X^*X)G) = 0$. Using twice the Cauchy–Binet formula, we obtain

$$\det(B^*(X^*X)G) = \sum_{\mathcal{I}_k} (\lambda_{i_1} - \zeta) \cdots (\lambda_{i_k} - \zeta) \overline{\lambda_{i_1}} \cdots \overline{\lambda_{i_k}} c_{i_1} \cdots c_{i_k} \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} (\lambda_{i_q} - \lambda_{i_p})$$

$$\times \left\{ \sum_{\mathcal{J}_k} \overline{\det((X^*X)_{\mathcal{I}_k, \mathcal{J}_k})} \overline{c_{j_1}} \cdots \overline{c_{j_k}} \prod_{\substack{j_1 \leq j_p < j_q \leq j_k \\ j_p, j_q \in \mathcal{J}_k}} (\overline{\lambda_{j_q}} - \overline{\lambda_{j_p}}) \right\}. \quad \square$$

The residual vector at iteration k is $r_k^G = \sum_{j=1}^n c_j p_k^G(\lambda_j) x^{(j)}$ but with $c = X^{-1} r_0$. The formula for $p_k^G(\lambda_j)$ is similar to what we have for normal matrices except for the additional sum over \mathcal{J}_k .

Theorems 5.23 and 5.24 give the exact solution of the minimization problem

$$\min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)r_0\|,$$

for matrices A that are normal or diagonalizable. It is worth noticing that the right-hand side (or the initial residual vector when $x_0 \neq 0$) plays an important role in the coefficients of the residual polynomials. The results of this section have been used to obtain bounds of the distances between Ritz values and the eigenvalues of A by using known bounds for roots of polynomials; see [684].

Since we know the coefficients of the polynomials associated with FOM and GMRES, we can write explicitly the Ritz and harmonic Ritz values for iteration numbers $k \leq 4$ since they are the roots of these polynomials. Let us assume for simplicity that A is normal and $k = 2$. We consider the polynomial p_2 whose roots are the two Ritz values θ_1 and θ_2 ,

$$p_2(\theta) = \delta_2 \theta^2 + \delta_1 \theta + \delta_0,$$

whose coefficients are from Theorem 5.21,

$$\begin{aligned} \delta_2 &= \sum_{\mathcal{I}_2} |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2, \\ \delta_1 &= \sum_{\mathcal{I}_2} (\lambda_{i_1} + \lambda_{i_2}) |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2, \\ \delta_0 &= \sum_{\mathcal{I}_2} \lambda_{i_1} \lambda_{i_2} |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2, \end{aligned}$$

where the summations are over all the ordered pairs of indices (i_1, i_2) such that $1 \leq i_1 < i_2 \leq n$. The two Ritz values at the second iteration are given by the roots of p_2 ,

$$\theta_{1,2} = \frac{-\delta_1 \pm \sqrt{\delta_1^2 - 4\delta_2\delta_0}}{2\delta_2}.$$

The harmonic Ritz values are the roots of the polynomial

$$q_2(\zeta) = \varsigma_2 \zeta^2 + \varsigma_1 \zeta + \varsigma_0,$$

where

$$\begin{aligned}\varsigma_2 &= \sum_{I_2} \overline{\lambda_{i_1} \lambda_{i_2}} |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2, \\ \varsigma_1 &= \sum_{I_2} (\lambda_{i_1} + \lambda_{i_2}) \overline{\lambda_{i_1} \lambda_{i_2}} |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2, \\ \varsigma_0 &= \sum_{I_2} |\lambda_{i_1}|^2 |\lambda_{i_2}|^2 |c_{i_1}|^2 |c_{i_2}|^2 |\lambda_{i_2} - \lambda_{i_1}|^2.\end{aligned}$$

Then, the two harmonic Ritz values are

$$\zeta_{1,2} = \frac{-\varsigma_1 \pm \sqrt{\varsigma_1^2 - 4\varsigma_2\varsigma_0}}{2\varsigma_2}.$$

Note that when A is real all the coefficients δ_j , ς_j are real.

5.7 Study of convergence using unitary matrices

In this section we consider another way of studying the convergence of GMRES using unitary GMRES-equivalent matrices. In [463] the authors studied the matrices B that generate the same Krylov residual space as the one given by the pair (A, r_0) , that is,

$$B\mathcal{K}_k(B, r_0) = A\mathcal{K}_k(A, r_0), \quad k = 1, 2, \dots, n.$$

Then GMRES applied to (B, r_0) yields the same residual norm convergence history as GMRES applied to (A, r_0) . Matrices with this property will be called $\text{GMRES}(A, r_0)$ -equivalent matrices.

The interest of studying the convergence of GMRES applied to unitary GMRES-equivalent matrices is that for unitary matrices, convergence is mainly governed by the eigenvalue distribution, as opposed to the situation with non-normal matrices. These matrices have eigenvalues which are located on the unit circle in the complex plane. Therefore, instead of having to possibly consider the distribution of the eigenvalues of A for general normal matrices, which can be located anywhere in the complex plane, we will have, so to speak, to deal with eigenvalues on a one-dimensional variety. In the following, we will relate GMRES convergence to the eigenvalues of a unitary matrix obtained from the orthonormal ascending bases of $\mathcal{K}_k(A, r_0)$ and $A\mathcal{K}_k(A, r_0)$. We mainly rely on the results in [291].

Let W be a unitary matrix whose columns give an ascending basis of $A\mathcal{K}_n(A, r_0)$ and \mathcal{H} an unreduced upper Hessenberg matrix such that $AW = W\mathcal{H}$. Then, it is

known from [463] that the matrix

$$B = W\tilde{R}\mathcal{H}W^* \quad (5.28)$$

where \tilde{R} is any nonsingular upper triangular matrix is GMRES(A, r_0)-equivalent.

The two papers [604, 633] analyzed the eigenvalues of particular unitary GMRES(A, r_0)-equivalent matrices and studied in detail the relation with GMRES convergence. In [633] QR and RQ factorizations of the matrix \mathcal{H} were used to obtain bounds for the residual norms in terms of the largest gap in the spectrum of the unitary Q factors on the unit circle. It was shown that a large maximum gap in the spectrum of the Q factor in an RQ factorization $\mathcal{H} = RQ$ implies fast GMRES convergence. In [604] an inverse result was shown, namely that fast GMRES convergence implies a large gap in the spectrum of Q in a certain RQ factorization of \mathcal{H} . It was also shown that the entries of this particular Q can be expressed in terms of the residual norms only.

We would like to explore how and to what extent GMRES convergence can be explained with unitary GMRES(A, r_0)-equivalent pairs. With a unitary GMRES(A, r_0)-equivalent pair (B, c) we mean a matrix with right-hand side c which generate the same convergence history as (A, r_0) where B is unitary and c is not necessarily equal to r_0 . We start with the study of unitary GMRES(A, r_0)-equivalent matrices B . Throughout the section, we assume GMRES for A and r_0 does not terminate before the n th iteration.

Proposition 5.1 *Let W be a unitary matrix whose columns give an ascending basis of $A\mathcal{K}_n(A, r_0)$, let \mathcal{H} be an unreduced upper Hessenberg matrix such that $AW = W\mathcal{H}$ and let $\mathcal{H} = RQ$ be an RQ factorization of \mathcal{H} . Then, the following assertions are equivalent:*

- 1- B is unitary and GMRES(A, r_0)-equivalent,
- 2- $B = WD_1QW^*$, where D_1 is a diagonal unitary matrix.

Proof The matrix B in (5.28) is unitary if and only if $\tilde{R}\mathcal{H}$ is unitary for some nonsingular upper triangular \tilde{R} , that is, $\tilde{R}\mathcal{H} = Q$ for a unitary Q . This Q must then be the Q factor of an RQ factorization $\mathcal{H} = RQ$ of \mathcal{H} and if we define $\tilde{R} \equiv R^{-1}$ then $B = WQW^*$ is unitary and GMRES(A, r_0)-equivalent. Because all other RQ factorizations of \mathcal{H} have the form $\mathcal{H} = (RD_1^{-1})(D_1Q)$, the same holds for $B = WD_1QW^*$. \square

Thus all unitary GMRES(A, r_0)-equivalent matrices are of the form $B = WQW^*$ where Q is the unitary factor of an RQ decomposition of \mathcal{H} . The RQ factorization of \mathcal{H} was addressed in [463], Section 3.1 and the spectra of corresponding Q factors were investigated in [604, 633] in relation to the residual norms generated by (A, r_0) . Note that the Hessenberg matrix \mathcal{H} is not the Hessenberg matrix generated in a standard implementation of GMRES where an orthogonal basis of $\mathcal{K}_n(A, r_0)$ is build. The matrix \mathcal{H} results from building an orthogonal basis of $A\mathcal{K}_n(A, r_0)$ by starting the Arnoldi process with the vector $Ar_0/\|Ar_0\|$.

In the standard implementation of GMRES, we construct the unique unitary matrix \hat{V} whose columns span the Krylov space $\mathcal{K}_n(A, r_0)$ and which is the result of the Arnoldi process applied to (A, r_0) . The unitary \hat{V} satisfies

$$A\hat{V} = \hat{V}H_+, \quad \hat{V}e_1 = r_0/\|r_0\|, \quad (5.29)$$

for an unreduced upper Hessenberg matrix H_+ with real positive subdiagonal entries. Let us define the unique QR decomposition

$$H_+ = Q^+R \quad (5.30)$$

to be the factorization such that Q^+ is a unitary upper Hessenberg matrix with positive real entries in the first row.

Lemma 5.5 *Let W be a unitary matrix whose columns give an ascending basis of $A\mathcal{K}_n(A, r_0)$ and let \hat{V} be the unitary matrix satisfying (5.29). If Q^+ is the unitary factor in the QR factorization (5.30) of H_+ , then*

$$Q^+ = \hat{V}^*WD_2,$$

where D_2 is a diagonal unitary matrix.

Proof Because the columns of W form an ascending basis of $A\mathcal{K}_n(A, r_0)$, we can write

$$A\hat{V} = W\hat{R}$$

for some nonsingular upper triangular matrix \hat{R} . Then from $A\hat{V} = \hat{V}H_+ = W\hat{R}$ we have the QR decomposition

$$H_+ = (\hat{V}^*W)\hat{R}.$$

Hence, for the properly chosen diagonal unitary matrix D_2 , $Q^+ = \hat{V}^*WD_2$ has its first row real and positive. \square

We remark that the moduli of the entries of the matrix Q^+ coincide with those of Q , which are described in [604]. The next corollary shows that Q^+ and Q are identical up to unitary row and column scaling.

Corollary 5.2 *Let Q^+ be the unitary factor in the QR factorization (5.30) of H_+ and let Q be the unitary factor of an RQ decomposition of \mathcal{H} . Then*

$$Q = D_3^*Q^+D_2^*$$

where D_2 is the matrix of Lemma 5.5 and D_3 is a diagonal unitary matrix.

Proof From (5.29) and Lemma 5.5 we have

$$AWD_2(Q^+)^* = WD_2(Q^+)^*H_+,$$

which implies

$$W^*AW = \mathcal{H} = D_2(Q^+)^*H_+Q^+D_2^*.$$

Hence we have the RQ decomposition

$$\mathcal{H} = D_2(Q^+)^*(Q^+\mathcal{R})Q^+D_2^* = (D_2\mathcal{R})(Q^+D_2^*).$$

Therefore, Q is of the form $Q = D_3^*Q^+D_2^*$ for some diagonal unitary matrix D_3 . \square

Lemma 5.5 allows to characterize unitary GMRES (A, r_0) -equivalent matrices in a way which is simpler than what was done in Proposition 5.1.

Theorem 5.25 *The following assertions are equivalent:*

- 1- B is unitary and GMRES (A, r_0) -equivalent,
- 2- $B = WV^*$, where V is a unitary matrix whose columns give an ascending basis of $\mathcal{K}_n(A, r_0)$ and W is a unitary matrix whose columns give an ascending basis of $A\mathcal{K}_n(A, r_0)$.

Proof Because of Proposition 5.1, if B is unitary and GMRES (A, r_0) -equivalent, then B is of the form

$$B = \hat{W}D_1Q\hat{W}^*,$$

where the columns of \hat{W} are an ascending orthonormal basis of $A\mathcal{K}_n(A, r_0)$. Using Lemma 5.5 and Corollary 5.2, we obtain

$$B = \hat{W}D_1Q\hat{W}^* = \hat{W}D_1D_3^*Q^+D_2^*\hat{W}^* = \hat{W}D_1D_3^*\hat{V}^*\hat{W}\hat{W}^* = \hat{W}D_1D_3^*\hat{V}^*.$$

Putting $V = \hat{V}D_3$ and $W = \hat{W}D_1$ gives the first implication. Now let $B = WV^*$. Then with Lemma 5.5,

$$B = W(V^*W)W^* = W(D_1\hat{V}^*W)W^* = W(D_1Q^+D_2^*)W^*$$

and with Corollary 5.2,

$$B = W(D_1Q^+D_2^*)W^* = W(D_1D_3QD_2^*)W^* = W(D_1D_3Q)W^*.$$

This yields the second implication if we use Proposition 5.1. \square

Theorem 5.25 shows how unitary GMRES (A, r_0) -equivalent matrices are closely related to the Krylov subspaces $\mathcal{K}_k(A, r_0)$ and the Krylov residual subspaces

$A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$. These subspaces, and therefore also the matrices V and W , depend strongly on the interplay between A and r_0 . Linking the properties of unitary GMRES-(A, r_0)-equivalent matrices (like spectral properties) to some simple properties of A only is therefore rather complicated.

Now we are interested in unitary matrices that give the same residual norm convergence curve as (A, r_0) with a right-hand side possibly different from r_0 . Our goal is to characterize the set of all pairs (B, c) with these properties. We will exploit the relation between unitary upper Hessenberg matrices with real positive subdiagonal entries and so-called *Schur parameters*. This relation is briefly outlined below.

Any unitary upper Hessenberg matrix Q_+ of order n with positive subdiagonal entries can be uniquely parameterized by n complex parameters γ_k such that $|\gamma_k| < 1$, $k = 1, \dots, n-1$ and $|\gamma_n| = 1$, see [17, 18, 160, 344, 442]. The γ_k 's are called Schur parameters. They are also known as partial correlation coefficients in statistics and reflection coefficients in signal processing. It is useful to introduce the so-called complementary Schur parameters σ_k , $k = 1, \dots, n-1$ (not to be confused with singular values) which are real and positive such that $\sigma_k = \sqrt{1 - |\gamma_k|^2}$.

The unitary matrix Q_+ can be written as the product

$$Q_+ = F_1(\gamma_1) F_2(\gamma_2) \cdots F_{n-1}(\gamma_{n-1}) \tilde{F}_n(\gamma_n),$$

where

$$F_k(\gamma_k) = \text{diag}\left(I_{k-1}, \begin{pmatrix} -\gamma_k & \sigma_k \\ \sigma_k & \overline{\gamma_k} \end{pmatrix}, I_{n-k-1}\right), \quad \tilde{F}_n(\gamma_n) = \text{diag}(I_{n-1}, -\gamma_n).$$

By identification, the nonzero entries of Q_+ are given by

$$q_{i+1,i} = \sigma_i, \quad q_{i,j} = -\overline{\gamma_{i-1}} \sigma_i \sigma_{i+1} \cdots \sigma_{j-1} \gamma_j, \quad 1 \leq i \leq j. \quad (5.31)$$

It means that the matrix Q_+ has the following form (see [344]),

$$Q_+ = \begin{pmatrix} -\gamma_1 & -\sigma_1 \gamma_2 & \cdots & -\sigma_1 \cdots \sigma_{k-1} \gamma_k & \cdots & -\sigma_1 \cdots \sigma_{n-1} \gamma_n \\ \sigma_1 & -\overline{\gamma_1} \gamma_2 & \cdots & -\overline{\gamma_1} \sigma_2 \cdots \sigma_{k-1} \gamma_k & \cdots & -\overline{\gamma_1} \sigma_2 \cdots \sigma_{n-1} \gamma_n \\ & \ddots & \vdots & & \ddots & -\overline{\gamma_2} \sigma_3 \cdots \sigma_{n-1} \gamma_n \\ & & \ddots & & & \vdots \\ & & & -\overline{\gamma_{k-1}} \gamma_k & \cdots & -\overline{\gamma_{k-1}} \sigma_k \cdots \sigma_{n-1} \gamma_n \\ & & & \sigma_k & \vdots & \vdots \\ & & & & \ddots & -\overline{\gamma_{n-1}} \gamma_n \end{pmatrix}.$$

Conversely, if we know Q_+ , the Schur parameters are given by

$$\sigma_i = q_{i+1,i}, \quad 1 \leq i < n, \quad \gamma_i = -\frac{q_{1,i}}{\sigma_1 \cdots \sigma_{i-1}}, \quad 1 \leq i \leq n.$$

It turns out that the Schur parameters for the Hessenberg matrices generated by the Arnoldi process are closely related to the corresponding GMRES residual norms.

Theorem 5.26 *Let $\|r_0\|, \|r_1^G\|, \dots, \|r_{n-1}^G\|$ be the residual norms when GMRES is applied to (A, r_0) , and let the unitary matrix B with the vector c generate the unitary upper Hessenberg matrix Q_+ when the Arnoldi process is applied. The following assertions are equivalent:*

- 1- (B, c) is GMRES (A, r_0) -equivalent,
- 2- Q_+ is a matrix with Schur parameters satisfying

$$|\gamma_j| = \|r_j^G\| \sqrt{\frac{1}{\|r_j^G\|^2} - \frac{1}{\|r_{j-1}^G\|^2}}, \quad j = 1, \dots, n-1, \quad |\gamma_n| = 1,$$

and with complementary Schur parameters

$$\sigma_j = \frac{\|r_j^G\|}{\|r_{j-1}^G\|}, \quad j = 2, \dots, n-1.$$

If the phase angle of $\gamma_k = |\gamma_k| e^{i\phi_k}$ is ϕ_k for $k = 1, \dots, n$, then the nonzero entries of Q_+ are

$$\begin{aligned} q_{1,k} &= -e^{i\phi_k} \left(\|r_{k-1}^G\|^2 - \|r_k^G\|^2 \right)^{\frac{1}{2}}, \quad k = 1, \dots, n, \\ q_{j,k} &= -e^{i(\phi_{j-1} + \phi_k)} \left(\frac{1}{\|r_{j-1}^G\|^2} - \frac{1}{\|r_{j-2}^G\|^2} \right)^{\frac{1}{2}} \left(\|r_{k-1}^G\|^2 - \|r_k^G\|^2 \right)^{\frac{1}{2}}, \quad j \leq k, \\ q_{k+1,k} &= \|r_k^G\| / \|r_{k-1}^G\|. \end{aligned}$$

Proof Consider GMRES applied to the pair (A, r_0) . In the standard GMRES implementation, one computes a QR decomposition of the upper Hessenberg matrix H_+ in (5.30) with the help of Givens rotations. One multiplies from the left with a cumulation \hat{Q} of Givens rotations, i.e., $\hat{Q}^* H = \hat{R}$ where \hat{R} is upper triangular and where

$$\hat{Q}^* = F_{n-1} \cdots F_1, \quad F_k = \text{diag} \left(I_{k-1}, \begin{pmatrix} c_k & -s_k \\ \bar{s}_k & \bar{c}_k \end{pmatrix}, I_{n-k-1} \right),$$

that is,

$$\hat{Q} = F_1^{-1} \cdots F_{n-1}^{-1}, \quad F_k^{-1} = \text{diag} \left(I_{k-1}, \begin{pmatrix} \bar{c}_k & s_k \\ -\bar{s}_k & c_k \end{pmatrix}, I_{n-k-1} \right).$$

Note that the scaling here is slightly different from what we have used above, but this does not influence the results. The subdiagonal entries of \hat{Q} are $-\bar{s}_k$, which are not necessarily positive. But in the special case where the system matrix B is unitary, the

Arnoldi process directly generates the unitary upper Hessenberg matrix Q_+ , which has real positive subdiagonal entries. Then, by identification, we have

$$|c_k| = |\gamma_k|, \quad s_k = \sigma_k, \quad k = 1, \dots, n-1.$$

Moreover, we have seen that,

$$\|r_k\| = \|r_0\| \prod_{j=1}^k |s_j|, \quad k = 0, 1, \dots, n-1. \quad (5.32)$$

Hence we can relate the residual norms and the Schur parameters. A straightforward inductive argument yields

$$\sigma_j = \frac{\|r_j^G\|}{\|r_{j-1}^G\|}, \quad j = 2, \dots, n-1.$$

If the phase angle of $\gamma_j = |\gamma_j|e^{i\phi_j}$ is ϕ_j , then as a function of the residual norms we have

$$\gamma_j = e^{i\phi_j} \|r_j^G\| \sqrt{\frac{1}{\|r_j^G\|^2} - \frac{1}{\|r_{j-1}^G\|^2}}.$$

We can also write the entries of Q_+ as functions of the residual norms. Let us consider the column k of the matrix Q_+ . The entry in the first row is

$$q_{1,k} = -\sigma_1 \cdots \sigma_{k-1} \gamma_k = -e^{i\phi_k} \left(\|r_{k-1}^G\|^2 - \|r_k^G\|^2 \right)^{\frac{1}{2}}.$$

The entry in row $j < k$ is

$$\begin{aligned} q_{j,k} &= -\gamma_{j-1} \sigma_j \cdots \sigma_{k-1} \gamma_k \\ &= -e^{i(\phi_{j-1} + \phi_k)} \left(\frac{1}{\|r_{j-1}^G\|^2} - \frac{1}{\|r_{j-2}^G\|^2} \right)^{\frac{1}{2}} \left(\|r_{k-1}^G\|^2 - \|r_k^G\|^2 \right)^{\frac{1}{2}}. \end{aligned}$$

For row k , we get

$$q_{k,k} = -\gamma_{k-1} \gamma_k = -e^{i(\phi_{k-1} + \phi_k)} \left(\frac{1}{\|r_{k-1}^G\|^2} - \frac{1}{\|r_{k-2}^G\|^2} \right)^{\frac{1}{2}} \left(\|r_{k-1}^G\|^2 - \|r_k^G\|^2 \right)^{\frac{1}{2}}.$$

Finally, as we already know, $q_{k+1,k} = \sigma_k = \|r_k^G\|/\|r_{k-1}^G\|$. \square

The previous theorem shows that for the upper Hessenberg matrix generated by the Arnoldi process applied to a unitary GMRES(A, r_0)-equivalent pair, its Schur

parameters give the residual norms and, except for the phase angles, the residual norms determine the Schur parameters. It is interesting to compare the results of the last theorem with those of Theorem 5.7.

Theorem 5.26 gives immediately the following characterization.

Corollary 5.3 *The following assertions are equivalent:*

- 1- (B, c) is a GMRES(A, r_0)-equivalent pair and B is unitary,
- 2- The matrix B and the vector c are of the form

$$B = ZV^*WZ^*, \quad c = \|r_0\|Ze_1,$$

where Z is any unitary matrix, V is a unitary matrix whose first k columns give a basis for $\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$ and W is a unitary matrix whose first k columns give a basis for $A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$.

Proof With Theorem 5.26, the pair (B, c) is GMRES(A, r_0)-equivalent and B is unitary if and only if the matrix B and the vector c are of the form

$$B = ZQ_+Z^*, \quad c = \|r_0\|Ze_1,$$

where Z is unitary and Q_+ is a unitary upper Hessenberg matrix with real positive subdiagonal entries whose Schur parameters are such that

$$|\gamma_k| = \|r_k^G\| \sqrt{\frac{1}{\|r_k^G\|^2} - \frac{1}{\|r_{k-1}^G\|^2}}, \quad k = 1, \dots, n-1, \quad |\gamma_n| = 1.$$

All unitary Hessenberg matrices generated by unitary GMRES(A, r_0)-equivalent pairs are diagonal unitary row and column scalings of each other. For example, denoting $\gamma_k = |\gamma_k|e^{i\phi_k}$, Q_+ is a diagonal unitary row and column scaling

$$Q_+ = D_5^* Q_{++} D_6, \quad D_5^* = \text{diag}(1, e^{-i\phi_1}, \dots, e^{-i\phi_{n-1}}), \quad D_6 = \text{diag}(e^{i\phi_1}, \dots, e^{i\phi_n})$$

of the upper Hessenberg matrix Q_{++} where all the Schur parameters are real and positive.

A particular unitary GMRES(A, r_0)-equivalent pair is (S, e_1) with $S = \hat{V}^* \hat{W}$ where \hat{V} is a unitary matrix whose first k columns give a basis of $\mathcal{K}_k(A, r_0)$ and \hat{W} is a unitary matrix whose first k columns give a basis of $A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$. Note that because of Lemma 5.5, S is a unitary row and column scaling of Q_+ in that lemma and is, in particular, upper Hessenberg. Therefore the Arnoldi process for the pair (S, e_1) generates a unitary upper Hessenberg matrix which is a diagonal unitary scaling of $\hat{V}^* \hat{W}$ and the upper Hessenberg matrix Q_+ generated by any unitary GMRES(A, r_0)-equivalent pair can be written as $Q_+ = D_7^* S D_8 = (\hat{V} D_7)^* \hat{W} D_8$ for appropriate diagonal unitary matrices D_7 and D_8 . \square

We see that like unitary GMRES(A, r_0)-equivalent matrices, the unitary GMRES- (A, r_0) -equivalent pairs are determined (here up to unitary equivalence expressed by Z in Corollary 5.3), by orthonormal bases for the Krylov subspaces $\mathcal{K}_k(A, r_0)$ and Krylov residual subspaces $A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$.

It is clear from the previous discussion that there exist unitary GMRES(A, r_0)-equivalent matrices with different spectra: With Proposition 5.1 the same convergence curve can be generated with the spectrum of WQW^* and with the spectrum of WD_1QW^* where D_1 represents any unit diagonal scaling. Similarly, there exist unitary system matrices of GMRES(A, r_0)-equivalent pairs with different spectra. We can also prove the following result.

Theorem 5.27 *Consider GMRES applied to an unreduced unitary Hessenberg matrix Q with right-hand side e_1 and zero initial guess. The following assertions are equivalent:*

- 1- \tilde{Q} is unitary and GMRES(Q, e_1)-equivalent,
- 2- $\tilde{Q} = D_1QD_2$, where D_i , $i = 1, 2$ are diagonal unitary matrices.

Proof An ascending orthogonal basis for $\mathcal{K}_k(Q, e_1)$ is given by the unit vectors e_1, \dots, e_n and an ascending orthogonal basis for

$$Q\mathcal{K}_n(Q, e_1) = \text{span}\{Qe_1, Q^2e_1, \dots, Q^n e_1\}$$

is given by the columns of Q . Therefore, with Theorem 5.25, $\tilde{Q} = WV^*$ where W is a unitary diagonal column scaling of Q and V is a unitary diagonal column scaling of the identity matrix I . \square

Clearly, the spectra of the unitary matrices Q and D_1QD_2 in the previous theorem can differ strongly (one does not need to be a rotation of the other and they do not interlace in general). In fact, infinitely many spectra seem to be able to generate the same behavior as the pair (Q, e_1) .

As mentioned, if the spectrum of a unitary matrix A has a large maximum gap, it was shown in [633] that we have fast GMRES convergence. On the other hand, if we have fast GMRES convergence, then there does not need to be a large gap in the spectrum of the unitary matrix A ; the fast convergence can be assured by a particular decomposition of r_0 in the invariant subspaces of A . If Q_+ is the corresponding upper Hessenberg matrix resulting from the Arnoldi process applied to (A, r_0) , then (Q_+, e_1) is a unitary GMRES(A, r_0)-equivalent pair where Q_+ has the same spectrum as the unitary matrix A . Since the right-hand side e_1 for the pair (Q_+, e_1) is independent of A and r_0 , the interplay of A and r_0 has been in some sense wrapped into the entries of Q_+ . Convergence still depends on how rich e_1 is in the various eigenvectors of Q_+ . However, if we scale Q_+ with unitary diagonal matrices D_1 and D_2 such that $Q_{++} = D_1Q_+D_2$ has only real positive Schur parameters, then, according to [604], fast convergence corresponds to a large gap in the spectrum of the Hessenberg matrix Q_{++} . The pair (Q_{++}, e_1) is another unitary GMRES(A, r_0)-equivalent pair. This pair

does not result from the Arnoldi process for (A, r_0) . It results from the Arnoldi process for a pair (\tilde{A}, \tilde{r}_0) where the eigenvalues of the unitary matrix \tilde{A} must contain the same large gap as the spectrum of Q_{++} . GMRES applied to this matrix \tilde{A} with an arbitrary right-hand side will exhibit fast convergence, see [633].

Using our previous results on unitary GMRES(A, r_0)-equivalent matrices and pairs, we have that the residual norms of GMRES applied to (A, r_0) are those of GMRES applied to a unitary matrix whose eigenvalues λ_i are the eigenvalues in the generalized eigenvalue problem

$$Wx = \lambda Vx.$$

For this unitary matrix, we can apply the bound of Theorem 5.2 with $\|X\| = \|X^{-1}\| = 1$ and obtain the following corollary.

Corollary 5.4 *The GMRES residual norms for the pair (A, r_0) are bounded as*

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{i=1,\dots,n} |p_k(\mu_i)|,$$

with μ_i , $i = 1, \dots, n$ being the eigenvalues of the generalized eigenvalue problem

$$V^* x = \mu W^* x, \quad (5.33)$$

where V is a unitary matrix whose first k columns give a basis of the Krylov space $\mathcal{K}_k(A, r_0)$ and W is a unitary matrix whose first k columns give a basis of the Krylov residual space $A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$.

Also, we can use the results of Theorem 5.17 specialized to the case of unitary matrices.

Theorem 5.28 *Let $X\Lambda X^*$ be the spectral factorization of a unitary matrix with Λ diagonal (with distinct diagonal entries λ_i of unit moduli), X unitary and r_0 be the initial residual vector such that $c = X^* r_0$ has no zero entries. Applying GMRES to this matrix with r_0 as the right-hand side, the residual norms are given by*

$$\|r_1^G\|^2 = \frac{\sum_{I_2} |c_{i_1}|^2 |c_{i_2}|^2 \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_2 \\ i_\ell, i_j \in I_2}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}{\sum_{i=1}^n |c_i|^2}, \quad (5.34)$$

$$\|r_k^G\|^2 = \frac{\sum_{I_{k+1}} \left[\prod_{j=1}^{k+1} |c_{i_j}|^2 \right] \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_{k+1} \\ i_\ell, i_j \in I_{k+1}}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}{\sum_{I_k} \left[\prod_{j=1}^k |c_{i_j}|^2 \right] \prod_{\substack{i_1 \leq i_\ell < i_j \leq i_k \\ i_\ell, i_j \in I_k}} |\lambda_{i_j} - \lambda_{i_\ell}|^2}. \quad (5.35)$$

Theorem 5.29 Let V be a unitary matrix whose first k columns give a basis of the Krylov space $\mathcal{K}_k(A, r_0)$ and W be a unitary matrix whose first k columns give a basis of the Krylov residual space $A\mathcal{K}_k(A, r_0)$ for $1 \leq k \leq n$. The residual norms of GMRES applied to (A, r_0) satisfy the relations (5.34) and (5.35) where λ_i are the eigenvalues in the generalized eigenvalue problem

$$Wx = \lambda Vx,$$

and the c_i are the first components of the corresponding eigenvectors.

Note that, if we denote by θ_j the angle of $\lambda_j = e^{i\theta_j}$ of modulus 1, we have

$$|\lambda_p - \lambda_q|^2 = 2(1 - \cos(\theta_p - \theta_q)).$$

We observe that the GMRES convergence for unitary matrices depends strongly on the angles between pairs of eigenvalues on the unit circle. As it is obvious, the residual norms stay the same when all eigenvalues are rotated by a given angle (without modifying the eigenvectors). A bad configuration for convergence is when the eigenvalues are almost regularly distributed on the unit circle and the weights are all of similar size. It has been proved that complete stagnation is possible only with a unitary spectra that represents a rotation of the roots of unity; see, [1017], or, for a shorter proof, [291].

5.8 Estimates of the norm of the error

In this section we provide formulas for the norm of the error when using FOM or GMRES as well as relations between the error norm and the residual norm. From these formulas we are able to compute estimates of the norm of the error during the iterations. As we said above, since stopping criteria based on the norm of the residual may sometimes be misleading, such estimates could lead to a more reliable way to stop the iterations.

First we give formulas for the norm of the error $\varepsilon_k = x - x_k$. Formulas for general Q-OR or Q-MR methods were also given in Chapter 3. Clearly, the relation between the error and the residual is $A\varepsilon_k = r_k$.

We assume for simplicity that the nonsingular matrix A is real and the matrices H_k , $1 \leq k \leq n$ are nonsingular. Therefore all the FOM iterates exist and the algorithms do not stop before $k = n$. Whatever is the algorithm, FOM or GMRES, the residual is equal to $r_k = r_0 - AV_{n,k}y_k$ and we have

$$\|\varepsilon_k\|^2 = (A^{-1}r_0, A^{-1}r_0) - 2(A^{-1}r_0, V_{n,k}y_k) + (V_{n,k}y_k, V_{n,k}y_k). \quad (5.36)$$

When we consider FOM we have the following result.

Theorem 5.30 In FOM the square of the norm of the error $\|\varepsilon_k^F\|^2$, $1 \leq k \leq n-1$, is given by

$$\begin{aligned}\|\varepsilon_k^F\|^2 &= \|r_0\|^2[(H_n^{-1}e^1, H_n^{-1}e_1) - (H_k^{-1}e_1, H_k^{-1}e_1) \\ &\quad + 2h_{k+1,k}(H_k^{-1}e_1, e_k)([H_n^{-1}e_{k+1}]_{1:k}, H_k^{-1}e_1)],\end{aligned}$$

where $[H_n^{-1}e_{k+1}]_{1:k}$ denotes the k first components of the $k+1$ st column of the inverse of H_n .

Proof In FOM the vector of coefficients y_k^F in the orthogonal basis is given by $H_k y_k^F = \|r_0\|e_1$ and then the k th iterate is $x_k^F = x_0 + V_{n,k}y_k^F$. The first term on the right-hand side of relation (5.36) is $(A^{-1}r_0, A^{-1}r_0)$. We write $r_0 = \|r_0\|v_1 = \|r_0\|V_n e_1$. Since with our hypothesis $AV_n = V_n H_n$ and $H_n = V_n^T A V_n$ is assumed to be nonsingular, we have $V_n H_n^{-1} = A^{-1} V_n$. Therefore,

$$(A^{-1}r_0, A^{-1}r_0) = \|r_0\|^2(H_n^{-1}e_1, H_n^{-1}e_1).$$

Note that this result uses the global orthogonality of the basis vectors. Also by orthogonality of the matrices $V_{n,k}$ the third term in equation (5.36) is (y_k^F, y_k^F) . Hence,

$$(V_{n,k}y_k^F, V_{n,k}y_k^F) = \|r_0\|^2(H_k^{-1}e_1, H_k^{-1}e_1).$$

It is more difficult to find an expression for the middle term $(A^{-1}r_0, V_{n,k}y_k^F)$ in relation (5.36). We write

$$(A^{-1}r_0, V_{n,k}y_k^F) = \|r_0\|(e_1, V_{n,k}^T A^{-T} V_{n,k}y_k^F),$$

and we remark that from multiplication with A^{-1} and transposition in the Arnoldi relation (4.3),

$$V_{n,k}^T A^{-T} V_{n,k} = H_k^{-T} - h_{k+1,k} H_k^{-T} e_k (v_{k+1})^T A^{-T} V_{n,k}.$$

Therefore,

$$(A^{-1}r_0, V_{n,k}y_k^F) = \|r_0\|[(H_k^{-1}e_1, y_k^F) - h_{k+1,k}(H_k^{-1}e_1, e_k)(A^{-1}v_{k+1}, V_{n,k}y_k^F)].$$

We have

$$\begin{aligned}(A^{-1}v_{k+1}, V_{n,k}y_k^F) &= \|r_0\|(A^{-1}V_n e_{k+1}, V_{n,k}H_k^{-1}e_1), \\ &= \|r_0\|(V_n H_n^{-1}e_{k+1}, V_{n,k}H_k^{-1}e_1).\end{aligned}$$

But

$$(V_n H_n^{-1}e_{k+1}, V_{n,k}H_k^{-1}e_1) = ([H_n^{-1}e_{k+1}]_{1:k}, H_k^{-1}e_1),$$

where $[H_n^{-1}e_{k+1}]_{1:k}$ denotes the vector of the k first components of $H_n^{-1}e_{k+1}$. Hence,

$$\begin{aligned}(A^{-1}r_0, V_{n,k}y_k^F) &= \|r_0\|^2[(H_k^{-1}e_1, H_k^{-1}e_1) \\ &\quad - h_{k+1,k}(H_k^{-1}e_1, e_k)([H_n^{-1}e_{k+1}]_{1:k}, H_k^{-1}e_1)].\end{aligned}$$

We have a factor -2 in front of this term. The first term on the right-hand side can be regrouped with (y_k^F, y_k^F) to obtain the result. \square

Note that if the Arnoldi process stops at iteration m with $h_{m+1,m} = 0$ we can replace n by m in the previous theorem. Of course, at iteration k we do not know H_n , so the norm of the error cannot be directly computed by the formula of Theorem 5.30. The fact that this formula depends on quantities that are not yet known is related to the fact that, mathematically, the algorithm must deliver the exact solution at iteration m with $m \leq n$. FOM is, in this sense, a direct method that we use as an iterative method. The formula for the norm of the error is somehow similar to what was obtained for the conjugate gradient algorithm in the symmetric case in [675], Theorem 2.1. We will see in a moment how to compute an approximation of the norm of the error.

We do not know the signs of all the terms in the formula of Theorem 5.30 and therefore, even if we would know H_n , it may not be appropriate to compute differences like

$$(H_n^{-1}e_1, H_n^{-1}e_1) - (H_k^{-1}e_1, H_k^{-1}e_1)$$

in finite precision arithmetic; see [435, 672, 676, 893] for a summary. It is thus interesting to try to express the first term of the right-hand side of the formula of Theorem 5.30 as a function of the second one to avoid computing the difference. To achieve this we write H_n blockwise as

$$H_n = \begin{pmatrix} H_k & W_k \\ Y_k^T & \check{H}_k \end{pmatrix}, \quad (5.37)$$

where $k < n$. Since H_n is upper Hessenberg and H_k is square we note that Y_k^T has just one nonzero element $h_{k+1,k}$ in the top right corner. Thus $Y_k^T = h_{k+1,k}e_1(e_k)^T$. The matrices H_k of order k and \check{H}_k of order $n - k$ are square upper Hessenberg and W_k is generally a full rectangular matrix (this is not consistent with our general notation but we use it to keep the formulas as simple as possible). Let $S_k = H_k - W_k \check{H}_k^{-1} Y_k^T$ be the Schur complement; the inverse of H_n can be written as

$$H_n^{-1} = \begin{pmatrix} S_k^{-1} & -S_k^{-1} W_k \check{H}_k^{-1} \\ -\check{H}_k^{-1} Y_k^T S_k^{-1} & \check{H}_k^{-1} + \check{H}_k^{-1} Y_k^T S_k^{-1} W_k \check{H}_k^{-1} \end{pmatrix},$$

see (2.2). Because of the structure of Y_k^T we have

$$S_k = H_k - h_{k+1,k}(W_k \check{H}_k^{-1} e_1)(e_k)^T.$$

Hence, S_k is a rank-one modification of H_k . By using the Sherman–Morrison formula (see Chapter 2) we obtain

$$S_k^{-1} = H_k^{-1} + \frac{h_{k+1,k}}{1 - h_{k+1,k}(e_k, H_k^{-1}W_k\check{H}_k^{-1}e_1)} H_k^{-1}W_k\check{H}_k^{-1}e_1(e_k)^T H_k^{-1}.$$

Note that if $h_{k+1,k}(e_k, H_k^{-1}W_k\check{H}_k^{-1}e_1) = 1$, then S_k is singular. Since we have assumed that H_k is nonsingular, this would imply that H_n is singular, contrary to our hypothesis. The difference $(H_n^{-1}e_1, H_n^{-1}e_1) - (H_k^{-1}e_1, H_k^{-1}e_1)$ can now be expressed in the following way.

Lemma 5.6 *Assume the block partitioning of H_n as in (5.37), let $w_k = W_k\check{H}_k^{-1}e_1$, $S_k = H_k - W_k\check{H}_k^{-1}Y_k^T$ and*

$$\gamma_k = \frac{h_{k+1,k}(e_k, H_k^{-1}e_1)}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)}.$$

Then,

$$S_k^{-1}e_1 = H_k^{-1}e_1 + \gamma_k H_k^{-1}w_k \quad (5.38)$$

and

$$\begin{aligned} (H_n^{-1}e_1, H_n^{-1}e_1) - (H_k^{-1}e_1, H_k^{-1}e_1) &= (h_{k+1,k}(e_k, S_k^{-1}e_1))^2(\check{H}_k^{-1}e_1, \check{H}_k^{-1}e_1) \\ &\quad + 2\gamma_k(H_k^{-1}e_1, H_k^{-1}w_k) + \gamma_k^2(H_k^{-1}w_k, H_k^{-1}w_k). \end{aligned}$$

Proof We are interested in the first column of H_n^{-1} and, in particular, in $S_k^{-1}e_1$ for which we have

$$S_k^{-1}e_1 = H_k^{-1}e_1 + \gamma_k H_k^{-1}W_k\check{H}_k^{-1}e_1 = H_k^{-1}e_1 + \gamma_k H_k^{-1}w_k.$$

The remaining part of the first column of H_n^{-1} which is $-\check{H}_k^{-1}Y_k^TS_k^{-1}e_1$ can be written as

$$-h_{k+1,k}(e_k, S_k^{-1}e_1)\check{H}_k^{-1}e_1.$$

Hence,

$$(H_n^{-1}e_1, H_n^{-1}e_1) = (S_k^{-1}e_1, S_k^{-1}e_1) + (h_{k+1,k}(e_k, S_k^{-1}e_1))^2(\check{H}_k^{-1}e_1, \check{H}_k^{-1}e_1),$$

and

$$(S_k^{-1}e_1, S_k^{-1}e_1) = (H_k^{-1}e_1, H_k^{-1}e_1) + 2\gamma_k(H_k^{-1}e_1, H_k^{-1}w_k) + \gamma_k^2(H_k^{-1}w_k, H_k^{-1}w_k),$$

with $w_k = W_k\check{H}_k^{-1}e_1$. □

Then, for the other part of the formula of Theorem 5.30, we are interested in computing the k first elements of $H_n^{-1}e_{k+1}$ (which are denoted as $[H_n^{-1}e_{k+1}]_{1:k}$) where e_{k+1} is here the $k+1$ st column of the identity matrix of order n .

Lemma 5.7 *Using the notation of Lemma 5.6, we have*

$$[H_n^{-1}e_{k+1}]_{1:k} = -\frac{\gamma_k}{h_{k+1,k}(e_k, H_k^{-1}e_1)} H_k^{-1} w_k.$$

Proof By construction the vector $[H_n^{-1}e_{k+1}]_{1:k}$ is the first column of the top right block of the inverse of H_n , that is, $-S_k^{-1}W_k\check{H}_k^{-1}e_1$. Using the results for S_k^{-1} , we obtain

$$\begin{aligned} & -S_k^{-1}W_k\check{H}_k^{-1}e_1 = \\ & - \left[H_k^{-1} + \frac{h_{k+1,k}}{1 - h_{k+1,k}(e_k, H_k^{-1}W_k\check{H}_k^{-1}e_1)} H_k^{-1}W_k\check{H}_k^{-1}e_1(e^k)^T H_k^{-1} \right] W_k\check{H}_k^{-1}e_1. \end{aligned}$$

Hence, since $(e_k)^T H_k^{-1}W_k\check{H}_k^{-1}e_1$ is a scalar, this is

$$\begin{aligned} & - \left[1 + \frac{h_{k+1,k}(e_k, H_k^{-1}W_k\check{H}_k^{-1}e_1)}{1 - h_{k+1,k}(e_k, H_k^{-1}W_k\check{H}_k^{-1}e_1)} \right] H_k^{-1}W_k\check{H}_k^{-1}e_1 = \\ & - \frac{1}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)} H_k^{-1}w_k. \end{aligned}$$

This last expression can be rewritten using γ_k defined in Lemma 5.6 to obtain the result. \square

Finally, we have the following expression for the square of the norm of the error in FOM.

Theorem 5.31 *Assume the block partitioning of H_n as in relation (5.37) and denote $w_k = W_k\check{H}_k^{-1}e_1$ and*

$$\gamma_k = \frac{h_{k+1,k}(e_k, H_k^{-1}e_1)}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)}.$$

Then,

$$\|\varepsilon_k^F\|^2 / \|r_0\|^2 = \gamma_k^2 \{ \|\check{H}_k^{-1}e_1\|^2 + \|H_k^{-1}w_k\|^2 \}. \quad (5.39)$$

Proof We use the results of Lemmas 5.6 and 5.7. The third term in the right-hand side of the formula of Theorem 5.30 is given by

$$2h_{k+1,k}(H_k^{-1}e_1, e_k)([H_n^{-1}e_{k+1}]_{1:k}, H_k^{-1}e_1) = -2\gamma_k(H_k^{-1}w_k, H_k^{-1}e_1).$$

We see that this term cancels with the same one but of opposite sign in the formula of Lemma 5.6. Therefore, we obtain, using (5.38),

$$\frac{\|\varepsilon_k^F\|^2}{\|r_0\|^2} = \left\{ h_{k+1,k} [(e_k, H_k^{-1} e_1) + \gamma_k (e_k, H_k^{-1} w_k)] \right\}^2 \|\check{H}_k^{-1} e_1\|^2 + \gamma_k^2 \|H_k^{-1} w_k\|^2.$$

Expressing everything in terms of γ_k , the norm of the error squared can be written even more simply as

$$\frac{\|\varepsilon_k^F\|^2}{\|r_0\|^2} = \gamma_k^2 \{ \|\check{H}_k^{-1} e_1\|^2 + \|H_k^{-1} w_k\|^2 \}.$$

We observe that we have only positive terms in the right-hand side. \square

We remark that

$$\frac{1}{1 - h_{k+1,k}(e_k, H_k^{-1} w_k)} = (H_n^{-1})_{k,k}.$$

The vectors which are involved in the expression for $\|\varepsilon_k^F\|^2$ are $\check{H}_k^{-1} e_1$, w_k , $H_k^{-1} w_k$ and $H_k^{-1} e_1$. Of course, at FOM iteration k we do not know \check{H}_k and w_k yet.

Let us relate the norm of the residual of the FOM method to the norm of the error in a different way. The following result was proved in [789].

Lemma 5.8 *The norm of the residual in FOM is*

$$\|r_k^F\| = \|r_0\| h_{k+1,k} |(H_k^{-1} e_1, e_k)|. \quad (5.40)$$

Proof This is obvious since we have seen that, as in any Q-OR method, $\|r_k\| = h_{k+1,k} [\|y_k^F\|]_k$ and, since we have assumed that H_k is nonsingular, $y_k^F = \|r_0\| H_k^{-1} e_1$. \square

The norm of the residual can be used in the expressions for the norm of the error. The next theorem shows that we obtain a simple and elegant formula for the norm of the error in terms of the norm of the residual.

Theorem 5.32 *Assume the block partitioning of H_n as in relation (5.37) and denote $w_k = W_k \check{H}_k^{-1} e_1$. Then,*

$$\begin{aligned} \|\varepsilon_k^F\|^2 &= \|r_k^F\|^2 \frac{\|\check{H}_k^{-1} e_1\|^2 + \|H_k^{-1} w_k\|^2}{[1 - h_{k+1,k}(e_k, H_k^{-1} w_k)]^2} \\ &= \|r_k^F\|^2 (H_n^{-1})_{k,k}^2 [\|\check{H}_k^{-1} e_1\|^2 + \|H_k^{-1} w_k\|^2]. \end{aligned} \quad (5.41)$$

Proof This result is obvious using the definition of γ_k , relation (5.39) and Lemma 5.8. \square

From Theorem 5.32 we see that the (squares of the) norms of the error and the residual are close if and only if the multiplying factor in relation (5.41) is close to 1. This factor depends on iterations $k + 1$ to n through \check{H}_k and w_k .

To approximate the norm of the error in FOM we use the same technique as in [435, 672], introducing a delay d which is a strictly positive integer. At iteration k ($k > d$) of FOM we approximate $\|\varepsilon_{k-d}^F\|^2$ by replacing H_k by H_{k-d} and H_n by H_k in the formula of Theorem 5.31. We now use the partitioning

$$H_k = \begin{pmatrix} H_{k-d} & W_{k-d} \\ Y_{k-d}^T & \check{H}_{k-d} \end{pmatrix}.$$

Note that the notation is different from the previous one since H_k is of order k and \check{H}_{k-d} of order d . Then, at iteration k we approximate the square $\|\varepsilon_{k-d}^F\|^2$ of the norm of the error at iteration $k - d$ using the result of Theorem 5.31 by

$$\chi_{k-d} = \|r_0\|^2 \gamma_{k-d}^2 \left\{ \|\check{H}_{k-d}^{-1} e_1\|^2 + \|H_{k-d}^{-1} w_{k-d}\|^2 \right\},$$

with $w_{k-d} = W_{k-d} \check{H}_{k-d}^{-1} e_1$ and

$$\gamma_{k-d} = \frac{h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1} e_1)}{1 - h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1} w_{k-d})}.$$

Note that χ_{k-d} is the sum of two positive quantities. A rationale for the choice of this approximation is given in the following Proposition.

Proposition 5.2 *The difference of the squares of the error norms at iterations $k - d$ and k is*

$$\begin{aligned} \|\varepsilon_{k-d}^F\|^2 - \|\varepsilon_k^F\|^2 &= \chi_{k-d} - 2\|r_0\|^2 \{ h_{k+1,k}(H_k^{-1} e_1, e_k)([H_n^{-1} e_{k+1}]_{1:k}, H_k^{-1} e_1) \\ &\quad - h_{k-d+1,k-d}(H_{k-d}^{-1} e_1, e_{k-d})([H_n^{-1} e_{k-d+1}]_{1:k-d} - [H_k^{-1} e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1} e_1) \}. \end{aligned}$$

Proof Let us write the difference of the squares of the error norms at iterations $k - d$ and k using the formula of Theorem 5.30. Some terms cancel and we obtain

$$\begin{aligned} \|\varepsilon_{k-d}^F\|^2 - \|\varepsilon_k^F\|^2 &= \|r_0\|^2 \{ (H_k^{-1} e_1, H_k^{-1} e_1) - (H_{k-d}^{-1} e_1, H_{k-d}^{-1} e_1) \\ &\quad + 2h_{k-d+1,k-d}(H_{k-d}^{-1} e_1, e_{k-d})([H_n^{-1} e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1} e_1) \\ &\quad - 2h_{k+1,k}(H_k^{-1} e_1, e_k)([H_n^{-1} e_{k+1}]_{1:k}, H_k^{-1} e_1) \}. \end{aligned}$$

Adding and subtracting the term

$$2\|r_0\|^2 h_{k-d+1,k-d}(H_{k-d}^{-1} e_1, e_{k-d})([H_k^{-1} e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1} e_1)$$

and rearranging terms, we obtain

$$\begin{aligned}
\|\varepsilon_{k-d}^F\|^2 - \|\varepsilon_k^F\|^2 &= 2\|r_0\|^2 \{-h_{k+1,k}(H_k^{-1}e_1, e_k)([H_n^{-1}e_{k+1}]_{1:k}, H_k^{-1}e_1) \\
&\quad + h_{k-d+1,k-d}(H_{k-d}^{-1}e_1, e_{k-d})([H_n^{-1}e_{k-d+1}]_{1:k-d} - [H_k^{-1}e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1}e_1)\} \\
&\quad + \|r_0\|^2 \{(H_k^{-1}e_1, H_k^{-1}e_1) - (H_{k-d}^{-1}e_1, H_{k-d}^{-1}e_1) \\
&\quad + 2h_{k-d+1,k-d}(H_{k-d}^{-1}e_1, e_{k-d})([H_k^{-1}e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1}e_1)\}.
\end{aligned}$$

For the last two terms we have, with Lemma 5.6,

$$\begin{aligned}
&\|H_k^{-1}e_1\|^2 - \|H_{k-d}^{-1}e_1\|^2 \\
&\quad + 2h_{k-d+1,k-d}(H_{k-d}^{-1}e_1, e_{k-d})([H_k^{-1}e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1}e_1) \\
&\quad = (h_{k-d+1,k-d}(e_{k-d}, S_{k-d}^{-1}e_1))^2 \|\check{H}_{k-d}^{-1}e_1\|^2 + 2\gamma_{k-d}(H_{k-d}^{-1}e_1, H_{k-d}^{-1}w_k) \\
&\quad + \gamma_{k-d}^2 \|H_{k-d}^{-1}w_{k-d}\|^2 \\
&\quad + 2h_{k-d+1,k-d}(H_{k-d}^{-1}e_1, e_{k-d})([H_k^{-1}e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1}e_1).
\end{aligned}$$

With Lemma 5.7 we have

$$[H_k^{-1}e_{k-d+1}]_{1:k-d} = -\frac{\gamma_{k-d}}{h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}e_1)} H_{k-d}^{-1}w_{k-d}$$

and using in addition (5.38) gives

$$\begin{aligned}
&\|H_k^{-1}e_1\|^2 - \|H_{k-d}^{-1}e_1\|^2 \\
&\quad + 2h_{k-d+1,k-d}(H_{k-d}^{-1}e_1, e_{k-d})([H_k^{-1}e_{k-d+1}]_{1:k-d}, H_{k-d}^{-1}e_1) \\
&\quad = \gamma_{k-d}^2 \|H_{k-d}^{-1}w_{k-d}\|^2 + (h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}e_1 + \gamma_{k-d}H_{k-d}^{-1}w_{k-d}))^2 \|\check{H}_{k-d}^{-1}e_1\|^2 \\
&\quad + 2\gamma_{k-d}(H_{k-d}^{-1}e_1, H_{k-d}^{-1}w_k) - 2\gamma_{k-d}(H_{k-d}^{-1}e_1, H_{k-d}^{-1}w_k) \\
&\quad = \gamma_{k-d}^2 \|H_{k-d}^{-1}w_{k-d}\|^2 + h_{k-d+1,k-d}^2((e_{k-d}, H_{k-d}^{-1}e_1) \\
&\quad + \gamma_{k-d}(H_{k-d}^{-1}w_{k-d}, e_{k-d}))^2 \|\check{H}_{k-d}^{-1}e_1\|^2.
\end{aligned}$$

However, by the definition of γ_{k-d} ,

$$\begin{aligned}
&h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}e_1) + \frac{h_{k-d+1,k-d}^2(e_{k-d}, H_{k-d}^{-1}e_1)}{1 - h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}w_{k-d})} (H_{k-d}^{-1}w_{k-d}, e_{k-d}) \\
&= \frac{h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}e_1)}{1 - h_{k-d+1,k-d}(e_{k-d}, H_{k-d}^{-1}w_{k-d})} = \gamma_{k-d},
\end{aligned}$$

and this concludes the proof. \square

Thus when FOM starts to converge we often have $\|\varepsilon_k^F\|^2 \ll \|\varepsilon_{k-d}^F\|^2$ and, additionally, the sum of the last two terms is small, meaning that χ_{k-d} gives a reasonable approximation of $\|\varepsilon_{k-d}^F\|^2$. Of course, we can anticipate that if FOM converges slowly we will obtain a poor approximation of the norm of the error.

Concerning the implementation, at iteration k we have to compute the last element of $H_{k-d}^{-1}e_1$, $\check{H}_{k-d}^{-1}e_1$, w_{k-d} and $H_{k-d}^{-1}w_{k-d}$. As for computing the iterate x_k^F we may use Givens rotations. The last element of $H_{k-d}^{-1}e_1$ is already known since this has been computed to obtain the iterate x_{k-d}^F or at least $\|r_{k-d}^F\|$ if the solution is computed only at the end of the iterations. Computing $\check{H}_{k-d}^{-1}e_1$ can be done with $d - 1$ rotations. This is not expensive since usually d will be small. The most computationally expensive part is obtaining $H_{k-d}^{-1}w_{k-d}$. This is done using the rotations computed in the Arnoldi process from step 1 to step $k - d - 1$. Moreover we have to apply the rotations to the columns of W_{k-d} . But this has already been done during FOM iterations $k - d$ to k . The result is a part of the triangular matrix R_k that is used to compute the solution at iteration k . We then have to take a linear combination of the columns of W_{k-d} and to solve a triangular system of order $k - d$ to obtain $H_{k-d}^{-1}w_{k-d}$.

To compute an approximation of the GMRES error norm we first relate the GMRES error norm to the FOM error norm. We have already seen in Chapter 3 relations between the Q-OR and Q-MR iterates. This could be applied to FOM and GMRES. We reformulate Theorem 3.2 using a different notation.

Let t_k be the last column of $(H_k^T H_k)^{-1}$ and $[t_k]_k$ its last element, that is,

$$[t_k]_k = (e_k, (H_k^T H_k)^{-1} e_k) = \|H_k^{-T} e_k\|^2. \quad (5.42)$$

Theorem 5.33 *Let x_k^G (resp. x_k^F) be the GMRES (resp. FOM) iterates and*

$$\delta_{k+1} = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 [t_k]_k},$$

and $u_k = \delta_{k+1} t_k$. Then,

$$x_k^G = x_k^F - [y_k^F]_k V_{n,k} u_k,$$

$$\varepsilon_k^G = \varepsilon_k^F + [y_k^F]_k V_{n,k} u_k,$$

where y_k^F is the coordinate vector of the FOM method, $y_k^F = \|r_0\| H_k^{-1} e_1$.

We use the relations between the errors in Theorem 5.33 to obtain an expression for the error norm in GMRES.

Theorem 5.34 *Assume the block partitioning of H_n as in relation (5.37). Denote $w_k = W_k \check{H}_k^{-1} e_1$ and*

$$\gamma_k = \frac{h_{k+1,k}(e_k, H_k^{-1} e_1)}{1 - h_{k+1,k}(e_k, H_k^{-1} w_k)}.$$

The vector t_k being the last column of $(H_k^T H_k)^{-1}$, $[t_k]_k$ its last element defined in relation (5.42),

$$\delta_{k+1} = \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2 [t_k]_k},$$

and $u_k = \delta_{k+1} t_k$, we have

$$\|\varepsilon_k^G\|^2 = \|\varepsilon_k^F\|^2 + \|r_0\|^2 (e_k, H_k^{-1} e_1) [2\gamma_k (H_k^{-1} w_k, u_k) + (e_k, H_k^{-1} e_1) \|u_k\|^2]. \quad (5.43)$$

Proof From Theorem 5.33 and denoting $\omega_k = [y_k^F]_k$ for simplicity we have

$$\varepsilon_k^G = \varepsilon_k^F + \omega_k V_{n,k} u_k.$$

The norm of ε_k^G is

$$\|\varepsilon_k^G\|^2 = \|\varepsilon_k^F\|^2 + 2\omega_k (\varepsilon_k^F, V_{n,k} u_k) + \omega_k^2 (V_{n,k} u_k, V_{n,k} u_k).$$

Since $(V_{n,k} u_k, V_{n,k} u_k) = (u_k, u_k)$ we are left with computing $(\varepsilon_k^F, V_{n,k} u_k)$. We have

$$\varepsilon_k^F = A^{-1} r_k^F = A^{-1} r_0 - V_{n,k} y_k^F.$$

But $r_0 = \|r_0\| V_n e_1$ and $A^{-1} V_n = V_n H_n^{-1}$. Therefore

$$\varepsilon_k^F = \|r_0\| V_n H_n^{-1} e_1 - V_{n,k} y_k^F.$$

This gives the decomposition of the error in FOM over the vectors v_j of the orthonormal basis of the Krylov subspace. Multiplying by $V_{n,k}^T$ we obtain

$$V_{n,k}^T \varepsilon_k^F = \|r_0\| [H_n^{-1} e_1]_{1:k} - y_k^F,$$

where, once again, $[H_n^{-1} e_1]_{1:k}$ denotes the k first components of the first column of the inverse of H_n . Then,

$$(\varepsilon_k^F, V_{n,k} u_k) = \|r_0\| ([H_n^{-1} e_1]_{1:k}, u_k) - (y_k^F, u_k) = \|r_0\| ([H_n^{-1} e_1]_{1:k}, u_k) - (H_k^{-1} e_1, u_k).$$

But because of (5.38), $[H_n^{-1} e_1]_{1:k} = H_k^{-1} e_1 + \gamma_k H_k^{-1} w_k$. Therefore

$$(\varepsilon_k^F, V_{n,k} u_k) = \|r_0\| \gamma_k (H_k^{-1} w_k, u_k),$$

and this concludes the proof. \square

Corollary 5.5 *Using the notation of Theorem 5.34 we have*

$$\|\varepsilon_k^G\|^2 = \|r_0\|^2 [\gamma_k^2 \|\check{H}_k^{-1} e_1\|^2 + \|\gamma_k H_k^{-1} w_k + (e_k, H_k^{-1} e_1) u_k\|^2]. \quad (5.44)$$

Proof We have, using Theorem 5.31,

$$\begin{aligned} & \|r_0\|^2 [\gamma_k^2 \|\check{H}_k^{-1} e_1\|^2 + \gamma_k^2 \|H_k^{-1} w_k\|^2 + 2\gamma_k (e_k, H_k^{-1} e_1) (H_k^{-1} w_k, u_k) \\ & \quad + (e_k, H_k^{-1} e_1)^2 \|u_k\|^2] \\ &= \|r_0\|^2 [\gamma_k (\|\check{H}_k^{-1} e_1\|^2 + \|H_k^{-1} w_k\|^2) + (e_k, H_k^{-1} e_1) ((e_k, H_k^{-1} e_1) \|u_k\|^2 \\ & \quad + 2\gamma_k (H_k^{-1} w_k, u_k))] \\ &= \|\varepsilon_k^F\|^2 + \|r_0\|^2 (e_k, H_k^{-1} e_1) \left((e_k, H_k^{-1} e_1) \|u_k\|^2 + 2\gamma_k (H_k^{-1} w_k, u_k) \right), \end{aligned}$$

which, according to (5.43), equals $\|\varepsilon_k^G\|^2$. \square

To estimate the GMRES error norm we use relation (5.43) and the estimate for the FOM error norm defined above. Using an integer delay d , we have

$$\begin{aligned} \|\varepsilon_{k-d}^G\|^2 &= \|\varepsilon_{k-d}^F\|^2 + \\ & \|r_0\|^2 (H_{k-d}^{-1} e_1, e_{k-d}) [2\gamma_{k-d} (H_{k-d}^{-1} w_{k-d}, u_{k-d}) + (H_{k-d}^{-1} e_1, e_{k-d}) \|u_{k-d}\|^2], \end{aligned}$$

where the definitions are similar to those in Theorem 5.34. It means, with Corollary 5.5, that our approximation to the square of the GMRES norm of the error can be written as

$$\chi_{k-d} = \|r_0\|^2 [\gamma_{k-d}^2 \|\check{H}_{k-d}^{-1} e_1\|^2 + \|\gamma_{k-d} H_{k-d}^{-1} w_{k-d} + (H_{k-d}^{-1} e_1, e_{k-d}) u_{k-d}\|^2],$$

and we have the sum of two positive quantities.

It has been suggested in [555] to use instead $\chi_{k-d} - \|r_0\|^2 \|(e_k, H_k^{-1} e_1) u_k\|^2$. However, this difference may not always be positive and something different has to be done in this case, like taking the absolute value or returning to χ_{k-d} only. Note also that, for the added term, we need to compute $(e_k, H_k^{-1} e_1)$ and u_k , even though they can be reused at some later iterations.

It is also interesting to write the GMRES error norm as a function of the residual norms.

Theorem 5.35 Assume the block partitioning of H_n as in relation (5.37). Denote $w_k = W_k \check{H}_k^{-1} e_1$ and let t_k be the last column of $(H_k^T H_k)^{-1}$ and $[t_k]_k$ its last element defined in relation (5.42). Then,

$$\begin{aligned} \|\varepsilon_k^G\|^2 &= \|\varepsilon_k^F\|^2 + 2 \|r_k^G\|^2 \frac{h_{k+1,k}}{1 - h_{k+1,k} (e_k, H_k^{-1} w_k)} (H_k^{-1} w_k, t_k) \\ & \quad + \|r_k^G\|^2 \frac{\|t_k\|^2}{1 + h_{k+1,k}^2 [t_k]_k}. \end{aligned}$$

Proof We already know that $\|\varepsilon_k^F\|^2$ can be written in terms of $\|r_k^F\|^2$. Let us concentrate on the last two terms in (5.43). Using the definition of γ_k and u_k we have that $2\|r_0\|^2\gamma_k(H_k^{-1}e_1, e_k)(H_k^{-1}w_k, u_k)$ is equal to

$$\begin{aligned} & 2\|r_0\|^2 \frac{h_{k+1,k}(H_k^{-1}e_1, e_k)^2}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)} \frac{h_{k+1,k}^2}{1 + h_{k+1,k}^2[t_k]_k} (H_k^{-1}w_k, t_k), \\ &= 2 \frac{\|r_k^F\|^2}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)} \frac{h_{k+1,k}}{1 + h_{k+1,k}^2[t_k]_k} (H_k^{-1}w_k, t_k), \end{aligned}$$

where we used Lemma 5.8. Further, because of Proposition 3.1 specialized to FOM and GMRES,

$$\begin{aligned} & 2 \frac{\|r_k^F\|^2}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)} \frac{h_{k+1,k}}{1 + h_{k+1,k}^2[t_k]_k} (H_k^{-1}w_k, t_k) \\ &= 2\|r_k^G\|^2 \frac{h_{k+1,k}}{1 - h_{k+1,k}(e_k, H_k^{-1}w_k)} (H_k^{-1}w_k, t_k). \end{aligned}$$

The other term is

$$\begin{aligned} \|r_0\|^2(H_k^{-1}e_1, e_k)^2\|u_k\|^2 &= \|r_0\|^2\delta_{k+1}^2(H_k^{-1}e_1, e_k)^2\|t_k\|^2, \\ &= \|r_k^F\|^2 \frac{\|t_k\|^2}{(1 + h_{k+1,k}^2[t_k]_k)^2}, \\ &= \|r_k^G\|^2 \frac{\|t_k\|^2}{1 + h_{k+1,k}^2[t_k]_k}. \end{aligned} \quad \square$$

The last term in the formula of Theorem 5.35 is positive but unfortunately we do not know the sign of the middle term. Therefore we cannot tell if $\|\varepsilon_k^G\|$ is smaller or larger than $\|\varepsilon_k^F\|$. However, we can express everything in terms of $\|r_k^G\|$ as in the following corollary.

Corollary 5.6 *The square of the norm of the error in GMRES, $\|\varepsilon_k^G\|^2$, is equal to $\pi_k\|r_k^G\|^2$ where*

$$\pi_k = \frac{n_k}{d_k},$$

with

$$\begin{aligned} n_k &= \|(1 + h_{k+1,k}^2[t_k]_k)H_k^{-1}w_k + (1 - h_{k+1,k}(e_k, H_k^{-1}w_k))t_k\|^2 \\ &\quad + (1 + h_{k+1,k}^2[t_k]_k)^2\|\check{H}_k^{-1}e_1\|^2, \end{aligned}$$

$$d_k = (1 - h_{k+1,k}(e_k, H_k^{-1}w_k))^2 (1 + h_{k+1,k}^2[t_k]_k).$$

Proof From relation (5.41) we have

$$\|\varepsilon_k^F\|^2 = \|r_k^F\|^2 \frac{\|\check{H}_k^{-1}e_1\|^2 + \|H_k^{-1}w_k\|^2}{[1 - h_{k+1,k}(e_k, H_k^{-1}w_k)]^2}.$$

Taking the term

$$\|r_k^F\|^2 \frac{\|H_k^{-1}w_k\|^2}{[1 - h_{k+1,k}(e_k, H_k^{-1}w_k)]^2} = \|r_k^G\|^2 (1 + h_{k+1,k}^2[t_k]_k) \frac{\|H_k^{-1}w_k\|^2}{[1 - h_{k+1,k}(e_k, H_k^{-1}w_k)]^2},$$

and adding it to the two last terms of Theorem 5.35, the numerator is

$$(1 + h_{k+1,k}^2[t_k]_k)^2 \|H_k^{-1}w_k\|^2 + 2(1 + h_{k+1,k}^2[t_k]_k)(1 - h_{k+1,k}(e_k, H_k^{-1}w_k))(H_k^{-1}w_k, t_k) + (1 - h_{k+1,k}(e_k, H_k^{-1}w_k))^2 \|t_k\|^2,$$

and the denominator is

$$(1 - h_{k+1,k}(e_k, H_k^{-1}w_k))^2 (1 + h_{k+1,k}^2[t_k]_k).$$

This ratio is to be multiplied by $\|r_k^G\|^2$. The remaining term is

$$\|r_k^G\|^2 \frac{1 + h_{k+1,k}^2[t_k]_k}{(1 - h_{k+1,k}(e_k, H_k^{-1}w_k))^2} \|\check{H}_k^{-1}e_1\|^2,$$

proving the claim. \square

Of course, $\pi_k > 0$. However, it seems difficult to know a priori if it is smaller or larger than 1.

5.9 Other implementations of GMRES

The implementation of GMRES proposed in [49] by E.H. Ayachour is concerned with the second phase of the algorithm, that is, the solution of the least squares problem and the computation of the norm of the residual. In the first phase which is the computation of the basis vectors and the columns of the matrix H the MGS version of the Arnoldi process is used. The derivation of the method in [49] is a little bit complicated but, as it is mentioned there, we can also use the Sherman–Morrison formula (see Chapter 2). This is what we do below. The Hessenberg matrix at iteration k is partitioned as

$$\underline{H}_k = \begin{pmatrix} (h^{(k)})^* \\ \tilde{H}_k \end{pmatrix}. \quad (5.45)$$

The row vector $(h^{(k)})^*$ is $(h_{1,1} \ h_{1,2} \ \cdots \ h_{1,k})$ and the matrix \tilde{H}_k is upper triangular. Note that this is, without the permutation, what we did in Chapter 2, relation (2.16), to obtain an LU factorization. Let us assume that $h_{k+1,k} \neq 0$. Then the matrix \tilde{H}_k is nonsingular. We solve the GMRES least squares problem using the normal equations,

$$\underline{H}_k^* \underline{H}_k y = \|r_0\| \underline{H}_k^* e_1. \quad (5.46)$$

We have the following result for the solution of the least squares problem.

Theorem 5.36 Assume $h_{k+1,k} \neq 0$ and let $u = \tilde{H}_k^{-*} h^{(k)}$. Then, the solution y of equation (5.46) is

$$y = \|r_0\| \frac{1}{1 + \|u\|^2} \tilde{H}_k^{-1} u. \quad (5.47)$$

Using this solution, the norm of the residual at iteration k is

$$\|r_k^A\| = \frac{\|r_0\|}{(1 + \|u\|^2)^{\frac{1}{2}}}. \quad (5.48)$$

Proof The matrix in equation (5.46) is

$$\underline{H}_k^* \underline{H}_k = \tilde{H}_k^* \tilde{H}_k + h^{(k)} (h^{(k)})^*.$$

This is a rank-one modification of $\tilde{H}_k^* \tilde{H}_k$. As we have done already several times, to compute the inverse we use the Sherman–Morrison formula (see Chapter 2) which yields

$$\begin{aligned} (\underline{H}_k^* \underline{H}_k)^{-1} &= (\tilde{H}_k^* \tilde{H}_k)^{-1} - (\tilde{H}_k^* \tilde{H}_k)^{-1} \frac{h^{(k)} (h^{(k)})^*}{1 + (h^{(k)})^* (\tilde{H}_k^* \tilde{H}_k)^{-1} h^{(k)}} (\tilde{H}_k^* \tilde{H}_k)^{-1}, \\ &= (\tilde{H}_k^* \tilde{H}_k)^{-1} - \frac{1}{1 + \|u\|^2} \tilde{H}_k^{-1} u u^* \tilde{H}_k^{-*}. \end{aligned}$$

We have to apply this inverse to the right-hand side of equation (5.46) which is

$$\|r_0\| \underline{H}_k^* e_1 = \|r_0\| h^{(k)}.$$

It yields the solution

$$\begin{aligned} y &= \|r_0\| \left(\tilde{H}_k^{-1} u - \frac{\|u\|^2}{1 + \|u\|^2} \tilde{H}_k^{-1} u \right), \\ &= \frac{\|r_0\|}{1 + \|u\|^2} \tilde{H}_k^{-1} u. \end{aligned}$$

The residual norm is

$$\|r_k^A\| = \|r_0\| \left\| e_1 - \frac{1}{1 + \|u\|^2} \begin{pmatrix} (h^{(k)})^* \\ \tilde{H}_k \end{pmatrix} \tilde{H}_k^{-1} u \right\|.$$

After some straightforward computations this is the expression in (5.48). \square

Mathematically $\|r_k^A\| = \|r_k^G\|$. Note that the result about the residual norm in Theorem 5.36 was also proved in [678], Theorem 5.1. From the expression of the residual norm we obtain

$$\|r_k^G\|^2 = \frac{1}{\frac{1}{\|r_{k-1}^G\|^2} + \frac{|u_k|^2}{\|r_0\|^2}}.$$

From Theorem 3.2 in [678] the term $|u_k|/\|r_0\|$ is the inverse of the FOM residual norm. Note that, if $\beta_k = \|r_k^G\|^2/\|r_0\|^2$, then we have the recurrence relation,

$$\beta_k = \beta_{k-1} \frac{1}{1 + \beta_{k-1}|u_k|^2}.$$

A few comments are in order about Theorem 5.36. First, the matrix \tilde{H}_k^* is lower triangular. Hence, when solving $\tilde{H}_k^* u = h^{(k)}$, since the right-hand side $h^{(k)}$ is $h^{(k-1)}$ appended with $h_{1,k}$ at the bottom, the solution u is what we have obtained at iteration $k-1$ except for the last component u_k which is

$$u_k = \frac{1}{h_{k+1,k}} \left(h_{1,k} - \sum_{j=1}^{k-1} \bar{h}_{j+1,k} u_j \right).$$

At iteration k we just have to compute u_k since we already know the previous components. Note that this is enough to compute the residual norm without solving for x_k . To compute the vector y we need to solve the linear system $\tilde{H}_k y = \beta u$ with an upper triangular matrix. However, solving numerically a least squares problem using the normal equation is, in general, not recommended.

Second, we have assumed that $h_{k+1,k} \neq 0$. If this would not happen we have to compute the solution which would be the exact solution of the linear system. But, in this case, the matrix \tilde{H}_k is singular, its last diagonal entry being zero. This was considered in [49]. Let $\check{H}_k = \tilde{H}_k + e_k e_k^T$, that is, \check{H}_k is the same as \tilde{H}_k except for the entry in position (k, k) which is equal to 1. Doing this, we can write $\underline{H}_k^* \underline{H}_k$ as

$\check{H}_k^* \check{H}_k$ plus a matrix of rank 2 and we can use the Sherman–Morrison–Woodbury formula to obtain the inverse. This was done differently in [49] where u is defined as $u = \check{H}_k^{-*} h^{(k)}$. The solution y is given by

$$y = \frac{\|r_0\|}{\bar{u}_k} \check{H}_k^{-1} e_k,$$

and $x_k = x_0 + V_{n,k}y$. Of course, in finite precision arithmetic we will never get $h_{k+1,k} = 0$ exactly but we may encounter small values of $h_{k+1,k}$.

From the formula (5.48) we remark that we have stagnation, that is, $\|b - Ax_k\|/\|b\|$ if and only if $u_k = 0$ which is equivalent to

$$h_{1,k} = \sum_{j=1}^{k-1} \bar{h}_{j+1,k} u_j.$$

If we already have stagnation from the beginning the components u_j , $j = 1, \dots, k-1$ are zero and stagnation at iteration k is equivalent to $h_{1,k} = 0$ which corresponds to $v_1^* Av_k = 0$, that is, Av_k is orthogonal to r_0 .

The following code implements Ayachour's variant of GMRES (without testing for $h_{k+1,k}$ small).

```
function [x,ni,resn]=GMRES_MGS_A(A,b,x0,epsi,nitmax);
%
n = size(A,1);
V = zeros(n,nitmax+1);
H = zeros(nitmax+1,nitmax);
u = zeros(nitmax,1);
resn = zeros(1,nitmax);
x = x0;
r = b - A * x;
ni = 0;
r0 = norm(b);
bet = norm(r);
resn(1) = bet;
v = r / bet;
V(:,1) = v;
alpha = 1;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    w = A * v;
    for j = 1:k % modified Gram-Schmidt
        vj = V(:,j);
        vw = vj' * w;
```

```

H(j,k) = vw;
w = w - vw * vj;
end % for j
nw = norm(w);
v = w / nw;
H(k+1,k) = nw;
V(:,k+1) = v; % next basis vector
if k > 1
    s = H(2:k,k)' * u(1:k-1);
else
    s = 0;
end % if k
ukk = conj(H(1,k)) - s;
gamma = 1 / sqrt(H(k+1,k)^2 + (abs(ukkan) * alpha)^2);
sk = H(k+1,k) * gamma;
alpha_old = alpha;
alpha = alpha * sk;
u(k) = ukk / H(k+1,k);
nresidu = bet * alpha; % estimate of the residual norm
resn(ni+1) = nresidu;
if nresidu < (epsi * r0) || ni >= nitmax ...
    || abs((alpha_old - alpha) / alpha) < 1e-10
    break % get out of the k loop
end % if nresidu
end % for k
%
rhs = (bet / (1 + norm(u(1:k))^2)) * u(1:k);
y = H(2:k+1,1:k) \ rhs;
x = x0 + V(:,1:k) * y;
resn = resn(1:ni+1);

```

Let us now consider another possibility for the implementation of GMRES. From Theorem 2.5 we know that the upper Hessenberg matrices H_k can be factorized as $H_k = U_k C^{(k)} U_k^{-1}$ where U_k is an upper triangular matrix and $C^{(k)}$ is a companion matrix. Let $\vartheta_{i,j}$ be the entries of U_k . We have $\vartheta_{1,1} = 1$ and it was shown in Theorem 3.12 that

$$\|r_{k-1}^G\| = \frac{\|r_0\|}{\left[\sum_{i=1}^k |\vartheta_{1,i}|^2\right]^{\frac{1}{2}}}, \quad k = 1, \dots, n, \quad (5.49)$$

where r_k^G are the residual vectors of the GMRES method.

From Lemma 2.2, when we know H_k , the entries $\vartheta_{1,j}$ can be computed by

$$\vartheta_{1,j+1} = -\frac{1}{h_{j+1,j}} \sum_{i=1}^j \vartheta_{1,i} h_{i,j}, \quad j = 1, \dots, k, \quad \vartheta_{1,1} = 1. \quad (5.50)$$

Relations (5.49) and (5.50) show that we can compute the GMRES residual norm without reducing the matrix H_k to upper triangular form as it is done in the standard implementation of GMRES. Hence, we can implement GMRES by decoupling the computation of the basis vectors from the computation of the solution of the least squares problem. The iterations can be stopped using relation (5.49).

We observe that using relation (5.50) is equivalent to solving by back substitution a linear system $L_k \vartheta^{(k+1)} = e_1$ where $\vartheta^{(k+1)} = (\vartheta_{1,1} \dots \vartheta_{1,k+1})^T$, e_1 is the first column of the identity matrix of order $k+1$ and L_k is a lower triangular matrix of order $k+1$,

$$L_k = \begin{pmatrix} 1 & & & \\ h_{1,1} & h_{2,1} & & \\ \vdots & \vdots & \ddots & \\ h_{1,k} & h_{2,k} & \dots & h_{k+1,k} \end{pmatrix}. \quad (5.51)$$

The diagonal entries of L_{k+1} are real and positive and there is no error in the right-hand side. From [521] we have the backward stability result,

$$(L_k + \Delta L_k) \hat{\vartheta}^{(k+1)} = e_1, \quad |\Delta L_k| \leq \gamma_{k+1} |L_k|,$$

where $\gamma_k = ku/(1 - ku)$, u being the unit roundoff and $\hat{\vartheta}^{(k+1)}$ is the computed result and assuming that $ku < 1$. In fact, the results in Lemma 8.4 of [521] give

$$h_{j+1,j} \hat{\vartheta}_{1,j+1} (1 + \theta_j^{(0)}) = - \sum_{i=1}^j \vartheta_{1,i} h_{i,j} (1 + \theta_j^{(i)}), \quad |\theta_j^{(i)}| \leq \gamma_j, \quad \forall i.$$

This means that the computed result is the exact result for slightly perturbed values of the entries of the j -th column of H .

The forward analysis in [521] shows that

$$\frac{\|\vartheta^{(k+1)} - \hat{\vartheta}^{(k+1)}\|_\infty}{\|\vartheta^{(k+1)}\|_\infty} \leq \frac{\text{cond}(L_k, \vartheta^{(k+1)}) \gamma_{k+1}}{1 - \text{cond}(L_{k+1}) \gamma_k},$$

with

$$\text{cond}(L_k, \vartheta^{(k+1)}) = \frac{\| |L_k^{-1}| |L_k| |\vartheta^{(k+1)}| \|_\infty}{\|\vartheta^{(k+1)}\|_\infty}, \quad \text{cond}(L_k) = \| |L_k^{-1}| |L_k| \|_\infty.$$

The basis vectors can be computed using any version of the Gram–Schmidt algorithm. For instance, for the sake of parallelism, one can use the iterated classical Gram–Schmidt algorithm or some block algorithms (see the next section). The least

squares problem is solved at the end by using a QR factorization of the Hessenberg matrix. This gives a particularly simple and parallel implementation of GMRES.

5.10 Parallel computing

GMRES is a two-stage algorithm. In the standard sequential implementation, at each iteration, after a matrix–vector product a new basis vector is computed using the modified Gram–Schmidt process and a new column of the matrix H is obtained. Then, the last computed column is modified as well as the right-hand side by applying Givens rotations. Generally, the most expensive part of the computation is the first stage. Moreover, as we will see in Chapter 11, in most cases, GMRES is used with restarting and this limits not only the amount of storage, but also the size of the least squares problem to be solved. Therefore, the second stage is not considered as a problem on parallel computers. In fact, this stage can even be executed redundantly on each of the processing units.

We have already addressed the parallelization of the first stage in Chapter 4 when we discussed the construction of Krylov subspaces bases. The goal is to reduce the communication overhead but to maintain the stability of the algorithm; see [193]. The main idea that was used, starting in the 1990s, is to compute s basis vectors in parallel and then to orthogonalize them by some variant of the QR factorization. In the first attempts the monomial basis $(v \ A v \ \cdots \ A^s v)$ was used and the method was restarted after these s iterations. As we have already seen several times, the use of this basis severely limits the values of s that can be used. An improvement was to use a Newton basis. However, the shifts were only the Ritz values computed by a few iterations of the standard Arnoldi process in the initialization phase and this also limited the value of s ; see [173, 263, 321, 584, 585, 596, 597]. Note that in [263] the modified Gram–Schmidt algorithm was used for the orthogonalization but the vectors are partitioned in batches such that the communications can overlap with computations. In [321] a parallel QR factorization, called RODDEC, combining Householder reflections with Givens rotations is used; see the references therein.

Other proposals were made later under the name *communication-avoiding* methods. A CA-GMRES method was described in [265, 527, 528]. The goal was to have a method allowing to restart after m iterations but with $m \geq s$ and to use some efficient parallel computational kernels. The first one is the matrix powers kernel to compute $(p_0(A)v \ p_1(A)v \ \cdots \ p_s(A)v)$ where the polynomials p_j satisfy the three-term recurrence

$$p_{j+1}(\xi) = \alpha_j \xi p_j(\xi) - \beta_j p_j(\xi) - \gamma_j p_{j-1}(\xi), \quad j = 1, \dots, s \text{ with } \alpha_j \neq 0.$$

Note that this covers the cases of the monomial, Newton and Chebyshev bases. By a careful coding (for details, see [527]) the $s + 1$ vectors can be computed much faster than when they are computed one after the other. For a discussion of these issues, see also [67, 188]. The second kernel is the QR factorization of the matrix whose

columns are the $s + 1$ vectors obtained from the matrix powers kernel. Since usually s is much smaller than n , the order of the matrix, we have what is called a tall and skinny matrix with many rows and only a few columns. Parallel methods were proposed to do this QR factorization; see [116, 165, 321, 962, 963]. In [527] a method named TSQR (meaning tall and skinny QR) was described. Note that in CA-algorithms, it is usually not possible to normalize the basis vectors during the computation of the basis since this reintroduces data dependencies. The vectors are eventually normalized after the whole basis has been computed but this is generally replaced by matrix equilibration (that is, row and column scalings) done in a preprocessing phase; see [188, 527].

If one wants to obtain an orthogonal basis without restarting every s iterations, the last block of vectors has to be orthogonalized against the previous ones (in fact we only have to orthogonalize the last s vectors of the last block). This is equivalent to updating the QR factorization of the previous vectors by adding s columns to the right. In [527] this is done using a block Gram–Schmidt algorithm. Let us see how to proceed.

We follow [891] and [527] using more or less the same notation that is in [527]. It is enough to consider the first two blocks. Let s be the block size. It is assumed to be constant except may be at the end of a cycle if restarting is used. Let us assume that we have already generated $s + 1$ orthogonal unit vectors v_1, v_2, \dots, v_{s+1} and let

$$V_1 = (v_1 \cdots v_s), \quad \underline{V}_1 = (V_1 \ v_{s+1}).$$

Note that in this section the underbar does not correspond to appending a row at the bottom of the matrix. Now, starting from the vector $v = v_{s+1}$, we compute $s + 1$ vectors $q_1 = v, q_2, \dots, q_{s+1}$ which is a basis of the Krylov subspace $\mathcal{K}_{s+1}(A, v)$. Let

$$Q_2 = (q_1 \cdots q_s), \quad \underline{Q}_2 = (Q_2 \ q_{s+1}),$$

$$\dot{Q}_2 = (q_2 \cdots q_s), \quad \underline{\dot{Q}}_2 = (\dot{Q}_2 \ q_{s+1}).$$

If we use the monomial basis for the new Krylov subspace, we get

$$\underline{Q}_2 = (v \ A v \ A^2 v \ \cdots \ A^s).$$

Our goal is to orthogonalize the vectors and to recover the corresponding upper Hessenberg matrix H . Assume that there exist an $(s + 1) \times s$ matrix \underline{B} such that $AQ_2 = \underline{Q}_2 \underline{B}$. For the monomial basis the matrix \underline{B} is zero except for the entries of the subdiagonal which are all equal to 1.

For the first block orthogonalization, we compute

$$\underline{X} = \underline{V}_1^T \underline{\dot{Q}}_2, \quad \underline{Y} = \underline{\dot{Q}}_2 - \underline{V}_1 \underline{X}.$$

Then, we orthogonalize \underline{Y} using, for instance, QR,

$$\underline{Y} = \tilde{Q}_2 \underline{R}_2, \quad \tilde{Q}_2 = (\tilde{q}_{s+2} \cdots \tilde{q}_{2s} \ \tilde{q}_{2s+1}) = (\hat{Q}_2 \ \tilde{q}_{2s+1}).$$

Note that if $\underline{Y} = (Y \ y)$, we have $Y = \hat{Q}_2 R_2$ where R_2 is the principal matrix of order $s - 1$ of \underline{R}_2 .

The orthogonal basis we were looking for is

$$Q = \left(\underline{V}_1 \ \hat{Q}_2 \right), \quad \underline{Q} = (Q \ \tilde{q}_{2s+1}).$$

We have

$$\underline{Q}_2 = \underline{Y} + \underline{V}_1 \underline{X} = \tilde{Q}_2 \underline{R}_2 + \underline{V}_1 \underline{X}.$$

Therefore,

$$[\underline{V}_1 \ \underline{Q}_2] = [\underline{V}_1 \ \tilde{Q}_2] \begin{pmatrix} I_{s+1} & X \\ 0 & \underline{R}_2 \end{pmatrix}.$$

The Arnoldi relation obtained before computing the second block is denoted as

$$AV_1 = \underline{V}_1 \begin{pmatrix} H_1 \\ h_{s+1,s} e_s^T \end{pmatrix}.$$

Using the relations for V_1 and Q_2 , we have

$$A[V_1 \ Q_2] = [V_1 \ \underline{Q}_2] \underline{\mathcal{B}},$$

with

$$\underline{\mathcal{B}} = \begin{pmatrix} H_1 & 0 \\ \tilde{H}_1 & B \end{pmatrix},$$

where \tilde{H}_1 is a zero matrix except for the first row which is $h_{s+1,s} e_s^T$. We observe that

$$[V_1 \ \underline{Q}_2] = [\underline{V}_1 \ \underline{Q}_2] = [\underline{V}_1 \ \tilde{Q}_2] \begin{pmatrix} I_{s+1} & X \\ 0 & \underline{R}_2 \end{pmatrix},$$

but $[\underline{V}_1 \ \tilde{Q}_2] = \underline{Q}$. Let $\underline{X} = (X \ x)$, then

$$A[V_1 \ Q_2] = A[\underline{V}_1 \ \underline{Q}_2] = A[\underline{V}_1 \ \hat{Q}_2] \begin{pmatrix} I_{s+1} & X \\ 0 & R_2 \end{pmatrix} = A\underline{Q} \begin{pmatrix} I_{s+1} & X \\ 0 & R_2 \end{pmatrix}.$$

Finally,

$$A\underline{Q} = \underline{Q}\mathcal{H},$$

with an upper Hessenberg matrix

$$\mathcal{H} = \begin{pmatrix} I_{s+1} & \underline{X} \\ 0 & \underline{R}_2 \end{pmatrix} \begin{pmatrix} H_1 & 0 \\ \tilde{H}_1 & \underline{B} \end{pmatrix} \begin{pmatrix} I_{s+1} & X \\ 0 & R_2 \end{pmatrix}^{-1}. \quad (5.52)$$

Note that the zero blocks are not of the same size in the first and third matrices. If the matrix Y is badly conditioned, this is reflected in the matrix R_2 and we may have troubles for computing its inverse. Of course, we do not want to compute all the entries of \mathcal{H} at the end of each block, we just want to compute the new columns. To do so we have to repartition the left and right matrices in relation (5.52),

$$\begin{pmatrix} I_{s+1} & \underline{X} \\ 0 & \underline{R}_2 \end{pmatrix} = \begin{pmatrix} I_s & \underline{Z} \\ 0 & \underline{W} \end{pmatrix},$$

where \underline{Z} is obtained from \underline{X} by deleting the last row and adding a zero column at the left. For the right matrix we first compute the inverse and then do the repartition,

$$\begin{pmatrix} I_{s+1} & X \\ 0 & R_2 \end{pmatrix}^{-1} = \begin{pmatrix} I_{s+1} & -XR_2^{-1} \\ 0 & R_2^{-1} \end{pmatrix} = \begin{pmatrix} I_s & S \\ 0 & T \end{pmatrix}.$$

With this notation we find that the top part of the new columns is given by

$$HS + \underline{Z}H_1S + \underline{Z}BT = HS + \underline{Z}BT,$$

since, because of the structure of H_1 , the term $\underline{Z}H_1$ is the zero matrix. The bottom part of the new columns is

$$\underline{W}H_1S + \underline{W}BT.$$

We observe that we have to compute the inverse R_2^{-1} and, as we said above, this can be a problem if R_2 is close to singularity. If we use the monomial basis the multiplication from the right with \underline{B} is just a shift of the columns.

After this computation we have to decide if it is useful to do a second round of block orthogonalization. We proceed as in [891], we reorthogonalize if the reduction in norms of all the projected vectors is less than 0.5 (see the code of `bgssro` in [891]). Let us describe the second round. For the first one we had

$$\underline{X} = \underline{V}_1^T \underline{\hat{Q}}_2, \quad \underline{Y} = \underline{\hat{Q}}_2 - \underline{V}_1 \underline{X}, \quad \underline{Y} = \underline{\hat{Q}}_2 \underline{R}_2.$$

We orthogonalize $\underline{\hat{Q}}_2$ against \underline{V}_1 ,

$$\underline{Z} = \underline{V}_1^T \underline{\hat{Q}}_2, \quad \underline{W} = \underline{\hat{Q}}_2 - \underline{V}_1 \underline{Z}, \quad \underline{W} = \underline{\hat{Q}}_2 \underline{\hat{R}}_2.$$

From this we find that

$$\underline{\hat{Q}}_2 = \underline{\hat{Q}}_2 \underline{\hat{R}}_2 \underline{R}_2 + \underline{V}_1 (\underline{Z} \underline{R}_2 + \underline{X}),$$

and

$$[\underline{V}_1 \hat{\underline{Q}}_2] = [\underline{V}_1 \hat{Q}_2] \begin{pmatrix} I_{s+1} & \frac{ZR_2 + X}{\hat{R}_2 R_2} \\ 0 & \hat{R}_2 R_2 \end{pmatrix}.$$

For \hat{Q}_2 we obtain

$$\hat{Q}_2 = \hat{Q}_2 \hat{R}_2 R_2 + V_1(ZR_2 + X),$$

where X (resp. Z) is equal to \underline{X} (resp. \underline{Z}) without the last column and \hat{R}_2 (resp. R_2) is the principal submatrix of \hat{R}_2 (resp. R_2) whose order has been decreased by 1. Then, we can proceed similarly to what we have done before to recover the matrix \mathcal{H} .

In [545] it has been suggested to use a variable s in the s-step GMRES algorithm. This idea was also proposed for the conjugate gradient method in the symmetric positive definite case in [189].

CA-GMRES and other algorithms were recently considered on parallel computers equipped with GPU (Graphics Processing Unit) accelerators; see [529, 997–999].

We observe that, mathematically, CA-GMRES is equivalent to GMRES. Pipelined GMRES, an algorithm that interleaves the calculation of dot products with the matrix–vector product and the vector additions was proposed in [415]. Unfortunately, this method uses classical Gram–Schmidt orthogonalization and avoids the explicit normalization of the new vector with a formula that uses the global orthogonality of the basis vectors. For the simplest one-step variant, a predictor z_{k+1} of Av_{k+1} is computed using the previous z_j . This doubled the storage and, moreover, one can have serious doubts about the stability of this method.

Orthogonalization methods based on CGS or MGS minimizing the number of synchronization points are derived in [901] and applied to GMRES. These methods seem to be stable.

5.11 Finite precision arithmetic

In this section we assume that the matrix A and the right-hand side b are real and we study FOM and GMRES in finite precision arithmetic.

From what we have seen in Chapter 4, we know that when constructing the basis vectors in finite precision arithmetic, they can lose their orthogonality and, even in the worst cases, their linear independence. This is usually the case when using the classical Gram–Schmidt algorithm. Therefore, it is important to study what are the effects of a potential loss of orthogonality and what is the best variant of GMRES. Something which is also interesting to look at is the maximum attainable accuracy, that is, the smallest possible value of $\|b - Ax_k\|$ that we could obtain in finite precision arithmetic. Ideally, this should be a small multiple of the unit roundoff.

It was known from the beginning of the use of these methods that a straightforward way to cure these potential problems is to reorthogonalize the basis vectors as soon as they are computed. As we have seen in Chapter 4 one step of reorthogonalization

is enough to achieve orthogonality to roundoff levels (the so-called “twice is enough” technique). A code for reorthogonalizing is given below. It could be inserted in the GMRES-MGS code before the statement `nw = norm(w)`,

```
% reorthogonalization
for j = 1:k
    vj = V(:,j);
    alpha = vj' * w;
    w = w - alpha * vj;
    H(j,k) = H(j,k) + alpha;
end % for j
```

Unfortunately, reorthogonalization is very costly and its cost increases with the iteration number. Hence, it has been suggested to use selective reorthogonalization; see, for instance, [421]. We replace the preceding code by

```
% selective reorthogonalization
if norm(w) < lorth * gin
    for j = 1:k
        vj = V(:,j);
        alpha = vj' * w;
        w = w - alpha * vj;
        H(j,k) = H(j,k) + alpha;
    end % for j
end % if selective reorth
```

The value of `gin` has to be initialized before the Arnoldi loop, after `w = A * v`; to `gin = norm(w)`. Hence, to trigger reorthogonalization we compare $\|Av_k\|$ to the norm of the vector obtained by the first orthogonalization. The parameter `lorth` can be taken to be between 0.5 and 1. However, we will see that in MGS-GMRES it is not necessary to reorthogonalize even though reorthogonalization may improve the maximum attainable accuracy.

Concerning stability, it was first proved in [279] that the Householder implementation of GMRES was backward stable; see also [774]. In the following the variables denote the computed quantities. The study begins with the Arnoldi relation in finite precision arithmetic. From what is known about the QR factorization we have

$$(v_1 \ fl(Av_1) \cdots \ fl(Av_k)) = V_{n,k+1} R_{k+1} + F_k^0,$$

and using results for matrix–vector product,

$$(fl(Av_1) \cdots \ fl(Av_k)) = AV_{n,k} + F_k^A.$$

It leads to a perturbed Arnoldi relation,

$$AV_{n,k} = V_{n,k+1}\underline{H}_k + F_k,$$

the perturbation term being bounded by

$$\|F_k\| \leq \|F_k^0\| + \|F_k^A\| \leq c_1(k^{\frac{3}{2}}n + k^{\frac{1}{2}}\ell n^{\frac{1}{2}})\varepsilon\|A\| \leq c'_1 k n^{\frac{3}{2}} \varepsilon \|A\|,$$

where ℓ is the maximum number of nonzeros in a row of A and ε is the machine epsilon. In what follows all the c_j are positive constants independent of the problem data assumed to be larger than 1. The loss of orthogonality in this implementation is bounded by

$$\|I - V_{n,k}^T V_{n,k}\| \leq c_2 k^{\frac{3}{2}} n \varepsilon.$$

Moreover, there exists an exactly orthogonal matrix $\hat{V}_{n,k+1}$ such that

$$AV_{n,k} = \hat{V}_{n,k+1}\underline{H}_k + \tilde{F}_k,$$

with

$$\|\tilde{F}_k\| \leq c_3 k n^{\frac{3}{2}} \varepsilon \|A\|,$$

and

$$\|V_{n,k+1} - \hat{V}_{n,k+1}\| \leq c_4 k^{\frac{3}{2}} n \varepsilon.$$

Then, in [279], a comparison was done between the norms of the true residual $b - Ax_k$ and the Arnoldi residual $\|r_0\| e_1 - \underline{H}_k y^G$. In finite precision arithmetic they are different since $V_{n,k}$ is not perfectly orthogonal. Moreover, we have

$$x_k = x_0 + V_{n,k} y^G + d_k,$$

where d_k is the error term. Let $\vartheta(k, \ell, n) = k^{\frac{3}{2}}n + k^{\frac{1}{2}}\ell n^{\frac{1}{2}}$ and

$$\vartheta(A, b, x_0) = O(\ell n^{\frac{1}{2}} + n)\varepsilon \|A\| \|x_0\| + O(n)\varepsilon \|b\|.$$

We observe that unless $\|x_0\|$ is large, $\vartheta(A, b, x_0)$ is small. If $x_0 = 0$ it is proportional to the norm of b . Then, we have

$$\|(b - Ax_k) - V_{n,k+1}(\|r_0\|e_1 - \underline{H}_k y^G)\| \leq c_5 \vartheta(k, \ell, n) \varepsilon \|A\| \|y^G\| + \vartheta(A, b, x_0).$$

However, we have not bounded yet the norm of y^G . Since it is obtained as the solution of a least squares problem, one can apply known results on this problem. After some difficult manipulations, it yields that $\|b - Ax_k\|$ is bounded from below by

$$(1 - c_4 k^{\frac{3}{2}} n \varepsilon) \|r_0\| \|e_1 - \underline{H}_k y^G\| - \|r_0\| \omega(A) - \vartheta(A, b, x_0),$$

and from above by

$$(1 + c_4 k^{\frac{3}{2}} n \varepsilon) \| \|r_0\| e_1 - \underline{H}_k y^G \| + \|r_0\| \omega(A) + \vartheta(A, b, x_0),$$

where

$$\omega(A) = \frac{c_6 \eta(k, \ell, n) \varepsilon \kappa(A)}{1 - c_7 \eta'(k, \ell, n) \varepsilon \kappa(A)},$$

with $\eta'(k, \ell, n) = c_4 k^{\frac{3}{2}} n + c_1 \eta(k, \ell, n) + 2c_8 k^{\frac{3}{2}}$ and $\kappa(A)$ is the condition number of A . To obtain this result it was assumed that

$$c_4 k^{\frac{3}{2}} n \varepsilon \ll 1, \quad 1 - c_7 \eta'(k, \ell, n) \varepsilon \kappa(A) > 0.$$

Other bounds can be obtained when the Arnoldi residual norm is small. The lower bound is

$$(1 - c_4 k^{\frac{3}{2}} n \varepsilon) \| \|r_0\| e_1 - \underline{H}_k y^G \| - c_5 \eta(k, \ell, n) \varepsilon \|A\| (\theta \|x\| + \|x_0\|) - \vartheta(A, b, x_0),$$

and the upper bound is

$$(1 + c_4 k^{\frac{3}{2}} n \varepsilon) \| \|r_0\| e_1 - \underline{H}_k y^G \| + c_5 \eta(k, \ell, n) \varepsilon \|A\| (\theta \|x\| + \|x_0\|) + \vartheta(A, b, x_0),$$

where

$$\theta = \frac{1 + \kappa(A)\tau}{1 - \kappa(A)\tau}, \quad \tau = \psi(2 + 2\mu) + \varepsilon c_8 \eta(k, \ell, n)(1 + \mu) + O(\varepsilon^2 + \psi\varepsilon),$$

where ψ is an upper bound

$$\frac{\| \|r_0\| e_1 - \underline{H}_k y^G \|}{\| \underline{H}_k \| \|y^G\| + \|r_0\|} \leq \psi, \quad \mu = \frac{\|x_0\|}{\|x\|},$$

and it is assumed that $1 - \kappa(A)\tau > 0$. This result gives an estimate of the maximum attainable accuracy which is $\varepsilon \|A\| \|x\|$. The constants and the polynomial terms in k or n are not what is important in these results.

Since, in finite precision arithmetic, it is likely that GMRES does not show an early termination, the paper [279] considers the backward error at iteration n ,

$$\varepsilon_B(x_n) \leq c_9(1 + \theta)n^{\frac{5}{2}}\varepsilon(2 + 2\mu) + O(\varepsilon^2),$$

if $\kappa(A)\tau < 1$. Once again, the values of the constants are not important. What is interesting is that the normwise backward error is proportional to $n^{\frac{5}{2}}\varepsilon$ which means that the Householder implementation of GMRES can be used safely.

The backward stability of GMRES-MGS was proved much later in [736]. Let us assume that the matrix A is real as well as the right-hand side and that A satisfies the restriction $\sigma_{\min}(A) \ll n^2 u \|A\|_F$ even though it is stated in [736] that this is an overestimate and that, in practice, we can replace n^2 by n . Let us also assume that $x_0 = 0$.

The authors of [736] started by considering the modified Gram–Schmidt algorithm to orthogonalize a given set of vectors as we have seen in Chapter 4. They used the fact that we have already mentioned that the MGS algorithm for the QR factorization of a matrix B can be interpreted as an orthogonal transformation applied to the matrix B augmented with a square matrix of zero elements on top. If the orthogonal transformation is Householder’s method this is true in the presence of rounding errors as well.

The derivation of the final result, which is quite technical and too complicated to be reported here, is based on some previous works [103, 420, 618, 740]. The authors had to refine some results concerning the modified Gram–Schmidt orthogonalization algorithm given in [420]. Let us just give the result. Let k be the first integer for which $\kappa(V_{n,k+1}) > 4/3$. It is proved that the approximate solution computed using MGS and Givens rotations to solve the least squares problems satisfies

$$(A + \Delta A_k + \Delta A'_k)x_k = b + \Delta b_k(y_k) + \Delta b'_k,$$

with

$$\|\Delta A_k + \Delta A'_k\| \leq \tilde{\gamma}_{kn} \|A\|_F, \quad \|\Delta b_k(y_k) + \Delta b'_k\| \leq \tilde{\gamma}_{kn} \|b\|,$$

where

$$\Delta A'_k = \frac{\|A\|_F \|x_k\|}{\|A\|_F \|x_k\| + \|b\|} \frac{r_k(y_k) x_k^T}{\|x_k\|^2},$$

$$\Delta b'_k = \frac{\|b\|}{\|A\|_F \|x_k\| + \|b\|} r_k(y_k),$$

y_k being the computed solution of the least squares problem, $\Delta A_k = AV_{n,k} + \Delta V_{n,k}(y_k)$, the computed residual being $r_k(y_k) = b_k - A_k y_k$, $b_k = b + \Delta b_k(y_k)$ satisfying the bounds

$$\|\Delta b_k(y_k)\| \leq \tilde{\gamma}_{kn} \|b\|, \quad \|\Delta V_{n,k}(y_k)\|_F \leq \tilde{\gamma}_{kn} \|A\|_F.$$

In these bounds

$$\tilde{\gamma}_j = \frac{cju}{1 - cju},$$

where c is a small integer and u is the unit roundoff, that is, half of the machine epsilon.

Therefore, the authors of [736] showed that GMRES-MGS is backward stable. It is also shown that GMRES-MGS gives a backward stable least squares solution at every step and, until we reach the k th iteration defined above, $\kappa(V_{n,j}) < 4/3$, $\forall j < k$.

The previous results about GMRES in finite precision arithmetic are illustrated by numerical examples in the next section.

5.12 Numerical examples

5.12.1 FOM and GMRES

Let us do a few numerical experiments with FOM and GMRES which use the orthogonal bases that we have constructed in Chapter 4.

Since in FOM and GMRES there is only one matrix–vector product per iteration it makes sense to plot the residual norms as functions of the iteration number. It would even make more sense to plot them against the computing time as the cost of one iteration increases as the computation proceeds but this is not easy to do reliably in Matlab for small matrices since the computing times are small and difficult to measure correctly.

First we compare the standard modified Gram–Schmidt versions of FOM and GMRES on some examples that we have already used in Chapter 4. We stress that, from a practical point of view, the linear systems in our sets of examples must not be solved with full GMRES or FOM (and probably not with any iterative method). We use them only to exhibit some properties of the methods in finite precision arithmetic.

GMRES is minimizing the residual norm at each iteration and the curves of the residual norms are decreasing. At worst, they can only stagnate. Figure 5.1 displays the GMRES and FOM relative computed residual norms for the matrix `fs_183_6` using MGS with $x_0 = 0$. As it is well known, due to the peak–plateau phenomenon, the FOM residual norms are oscillatory when GMRES almost stagnates.

Figure 5.2 shows the relative true residual norms, that is, $\|b - Ax_k\|/\|b\|$. The maximum attainable accuracy is almost the same for both methods around $3 \cdot 10^{-16}$. Note that for this example $\|r_0\| = \|b\| = 1.1808 \cdot 10^9$ is large. Hence, the absolute true residual norms at the end are of the order of 10^{-7} .

It is also interesting to look at the error norms. Here we compute the error taking the result of $A \backslash b$ as the exact solution. It yields a true residual norm equal to $5.9852 \cdot 10^{-8}$ and the relative residual norm is $5.0686 \cdot 10^{-17}$. There are not many differences between the GMRES and FOM error norms except at the end of the computation; see Figure 5.3. We observe that, for this example, the shape of the error norm is quite different from that of the residual norm.

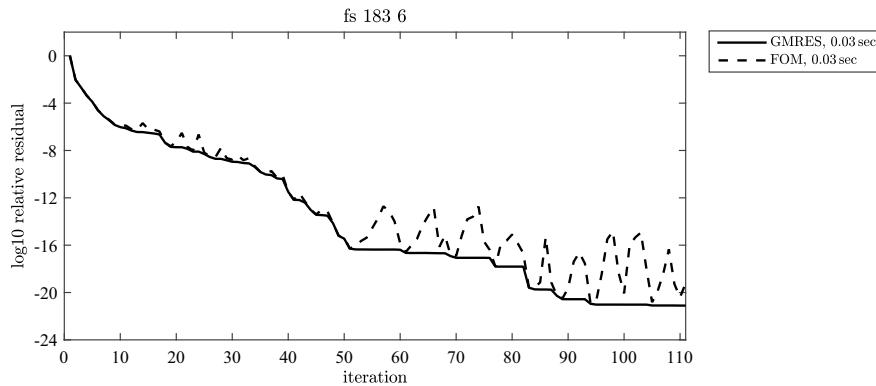


Fig. 5.1 *fs 183 6*, \log_{10} of the relative residual norm, GMRES (plain) and FOM (dashed)

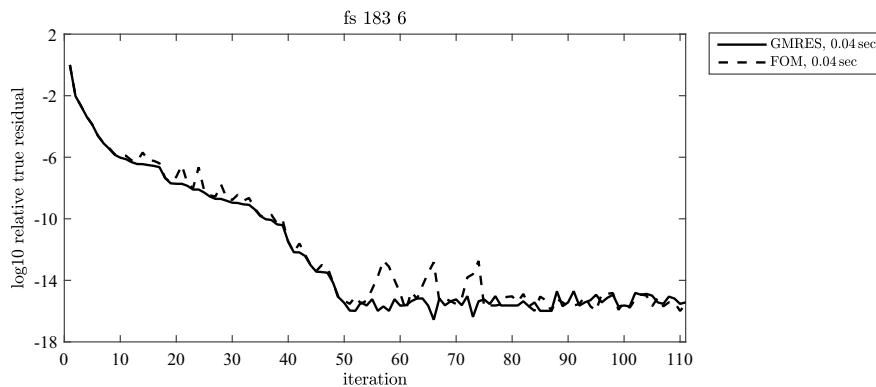


Fig. 5.2 *fs 183 6*, \log_{10} of the relative true residual norm, GMRES (plain) and FOM (dashed)

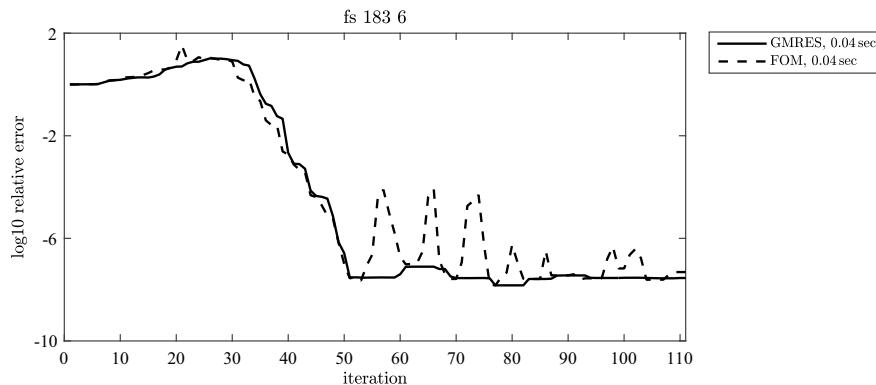


Fig. 5.3 *fs 183 6*, \log_{10} of the relative error norm, GMRES (plain) and FOM (dashed)

Let us now consider our example **fs 680 1c** of order 680. Figure 5.4 compares the FOM and GMRES residual norms using MGS and $x_0 = 0$. The convergence is almost the same but we observe large oscillations of the FOM residual norm when the final stagnation level has been reached for GMRES. The same happens for the true residual norm as well as for the error norm but we do not show them.

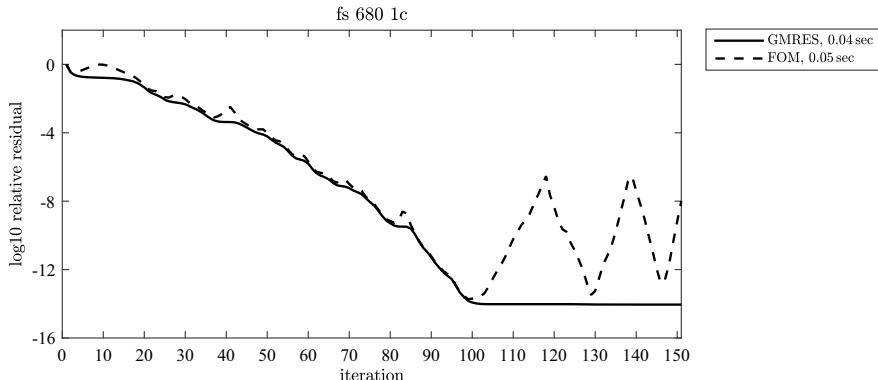


Fig. 5.4 **fs 680 1c**, \log_{10} of the relative residual norm, GMRES (plain) and FOM (dashed)

Unfortunately GMRES (or FOM) does not always converge as nicely as for the two previous examples. Figure 5.5 shows the FOM and GMRES residual norms using MGS and $x_0 = 0$ for the matrix **e05r0500** of order 236. GMRES almost stagnates up to $k = 170$ and then the decrease is slow until we reach the order of the matrix. When GMRES almost stagnates, FOM is heavily oscillating according to the peak–plateau phenomenon. In this example there is not much loss of orthogonality for MGS.

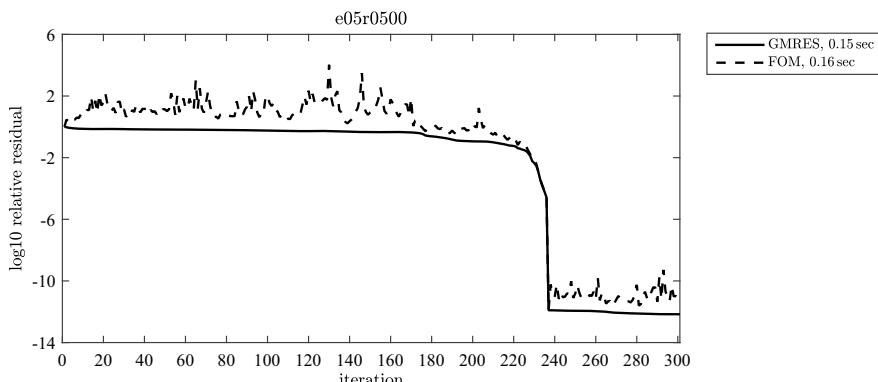


Fig. 5.5 **e05r0500**, \log_{10} of the relative residual norm, GMRES (plain) and FOM (dashed)

5.12.2 Variants of GMRES

Then, we compare GMRES with the different orthogonal bases we studied in Chapter 4, that is, classical Gram–Schmidt (CGS), modified Gram–Schmidt (MGS) and Householder (Hous). Figure 5.6 shows the relative true residual norms. We observe that the maximum attainable accuracy obtained with CGS is much worse by many orders of magnitude than with MGS or Householder. This is consistent with what we have seen in Figure 4.3 since the CGS basis vectors loose their linear independence around iteration 30 but the smallest singular value of $V_{n,k}$ starts to decrease fast around iteration 20 and orthogonality is completely lost. The MGS and Householder bases give almost the same maximum attainable accuracy.

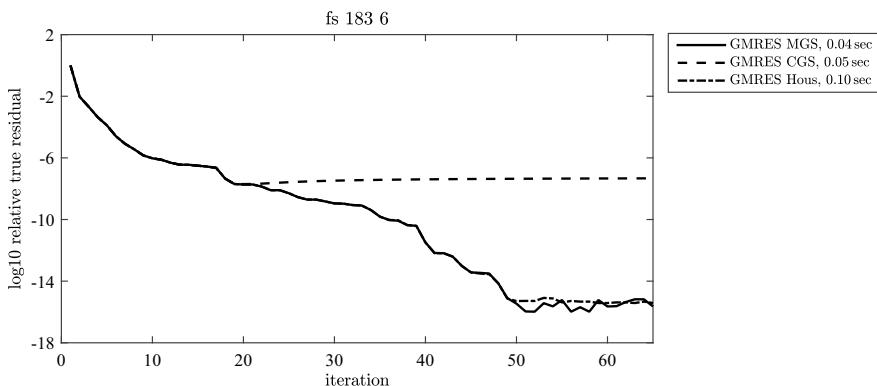


Fig. 5.6 *fs 183 6*, \log_{10} of the relative true residual norm, GMRES, MGS (plain), CGS (dashed) and Householder (dot-dashed)

Figure 5.7 displays the relative true residual norm and the level of orthogonality that we have already seen in Figure 4.1. We observe that, as predicted in [461], the two curves are almost symmetrical with respect to a parallel to the x -axis at the level 10^{-8} . Moreover, the minimum of the residual norm reaches its minimum value at iteration 51, that is, when for the first time $\kappa(V_{n,k}) > 4/3$. Hence, with MGS, orthogonality is completely lost only when the true residual norm has reached its maximum attainable accuracy.

In Figure 5.8 we compare the double-precision version of GMRES-MGS with the same algorithm in variable precision using 64 decimal digits. We observe that there are some differences only at the end of the computation around iteration 45, that is, when the level of orthogonality is already quite large. Hence, rounding errors do not have much influence in this computation.

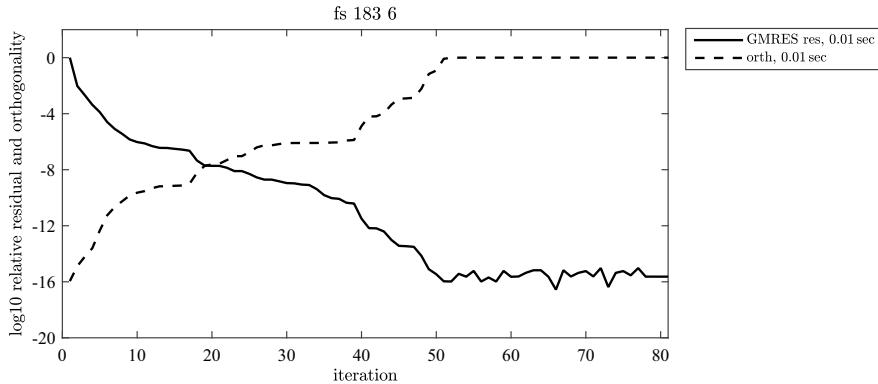


Fig. 5.7 fs 183 6, \log_{10} of the relative true residual norm, GMRES-MGS (plain) and level of orthogonality(dashed)

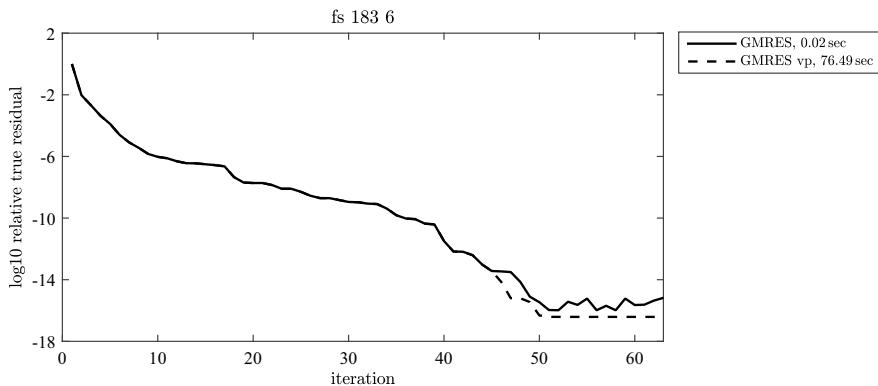


Fig. 5.8 fs 183 6, \log_{10} of the relative true residual norm, GMRES-MGS double-precision (plain) and GMRES-MGS variable precision with 64 decimal digits (dashed)

Figure 5.9 compares the variants of GMRES for fs 680 1c. For this example CGS is not as worse as it was for fs 183 6 but, nevertheless, the maximum attainable accuracy is worse than with MGS or Householder. Again, this is linked to the loss of linear independence for the CGS basis vectors.

Figure 5.10 displays the relative true residual norm and the level of orthogonality. Once again the curves are almost symmetrical with respect to a parallel to the x -axis.

In Figure 5.11 we plot the relative true residual norms for GMRES-MGS in double-precision and in variable precision with 64 decimal digits. There are noticeable differences only after iteration 100 after the double-precision version has reached its maximum attainable accuracy. We can conclude that, for this example, rounding errors do not have an influence on the convergence of GMRES but the final loss

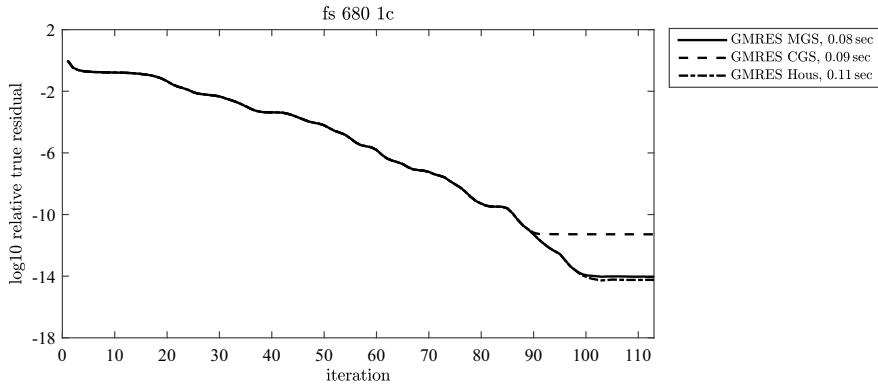


Fig. 5.9 *fs 680 1c*, \log_{10} of the relative true residual norm, GMRES, MGS (plain), CGS (dashed) and Householder (dot-dashed)

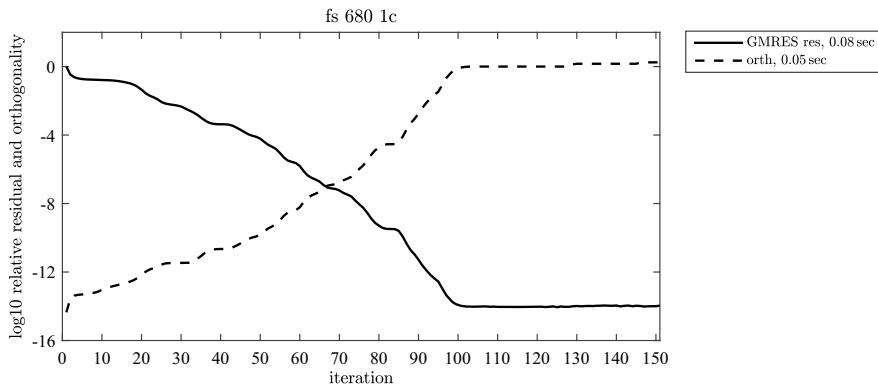


Fig. 5.10 *fs 680 1c*, \log_{10} of the relative true residual norm, GMRES-MGS (plain) and level of orthogonality(dashed)

of orthogonality in the double-precision version has an influence on the maximum attainable accuracy.

Tables 5.1 and 5.2 show the minimum relative true residual norms $\|b - Ax_k\|/\|b\|$ among k_{\max} iterations for the three variants of GMRES for our two sets of test matrices. These values roughly correspond to the maximum attainable accuracy. The matrices in Table 5.1 are ordered by increasing condition number. We also add the results for the variant (Ay) in [49], described in Section 5.9, which uses the MGS basis but solves the least squares problem in a different way. We observe that MGS gives approximately the same results as the Householder variant which is more expensive. In many cases CGS gives much worse results and cannot be used reliably.

Table 5.1 GMRES, minimal relative true residual norms in k_{\max} iterations at most

matrix	k_{\max}	MGS	CGS	Hous	Ay
pde225	150	1.66541e-15	2.02553e-15	1.69785e-15	1.67515e-15
gre 343	300	6.61396e-15	1.72608e-14	6.66235e-15	6.32267e-15
jphw 991	200	1.07847e-14	5.36012e-14	8.55011e-15	1.16436e-14
pde2961	320	1.65763e-14	1.82716e-11	1.46514e-14	1.65149e-14
jagmesh1	300	4.83286e-16	9.75700e-16	5.05858e-16	4.38282e-16
bawa782	350	1.39710e-14	1.19008e-09	1.37715e-14	1.47806e-14
dw2048	1100	1.24776e-15	2.20976e-15	2.48198e-15	1.13315e-15
jagmesh2	800	5.73936e-16	3.46726e-15	6.28434e-16	5.79634e-16
raefsky2	450	7.17708e-15	3.76155e-11	5.53480e-15	6.67254e-15
fs 680 1c	150	8.85755e-15	5.20219e-12	5.31928e-15	8.30622e-15
add20	600	3.03778e-13	2.18928e-11	1.97913e-13	2.89110e-13
raefsky1	350	6.06386e-14	1.10400e-07	4.10392e-14	5.94621e-14
jagmesh4	700	4.62824e-15	1.11342e-14	4.72563e-15	1.18695e-15
fs 680 1	200	7.02671e-15	2.92822e-05	3.87286e-15	7.19970e-15
sherman1	500	4.72052e-14	2.92323e-12	4.31378e-14	4.67654e-14
nos3	320	9.26899e-14	2.04352e-12	4.48234e-14	9.72356e-14
sherman5	1200	1.21236e-11	2.37157e-01	1.18730e-11	1.20748e-11
cavity05	600	1.65859e-13	6.28791e-04	1.77171e-13	1.82144e-13
e05r0500	260	1.76589e-12	4.50556e-01	2.32899e-12	1.82263e-12
comsol	300	7.24522e-11	8.14341e-02	6.37706e-11	7.43836e-11
olm1000	600	3.26048e-14	2.50997e-10	2.74235e-14	3.51355e-14
cavity10	900	1.68665e-13	5.31509e-05	1.31367e-13	2.43626e-13
steam2	200	2.24697e-10	7.29929e-05	2.79280e-10	2.28696e-10
1138bus	700	5.68921e-12	2.42595e-09	5.15116e-12	5.24992e-12
steam1	250	3.71137e-10	4.11960e-02	4.02497e-10	3.63323e-10
bcsstk26	1000	4.13181e-08	4.13181e-08	4.13181e-08	4.13181e-08
nos7	500	9.11821e-08	7.99355e-03	9.41437e-08	1.02464e-07
watt1	350	7.17226e-09	8.58551e-03	8.46024e-09	8.42763e-09
bcsstk14	1000	1.38535e-01	1.38535e-01	1.38535e-01	1.38535e-01
fs 183 6	60	1.04100e-16	1.98691e-08	3.79984e-16	1.68800e-14
bcsstk20	500	1.81822e-06	7.89539e-02	1.23138e-06	1.94633e-06
mfcf	820	4.61980e-06	7.71514e-01	7.66934e-06	4.38417e-06
nnc	250	4.55189e-04	5.21410e-01	4.30314e-04	3.85336e-04
Insp	420	1.66223e-03	9.53073e-01	2.02611e-03	1.75295e-03

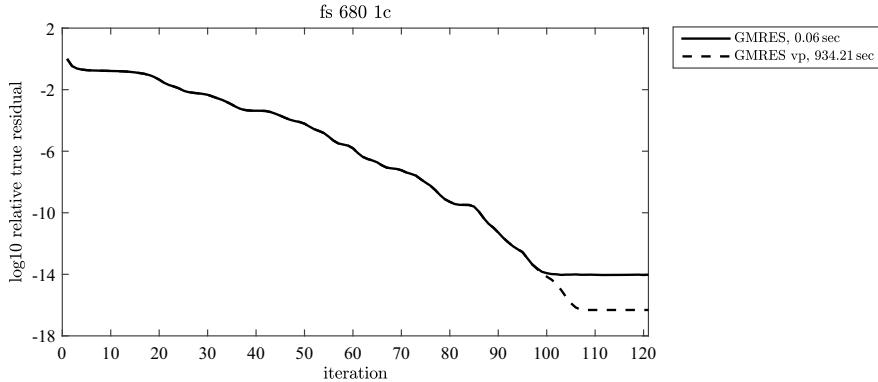


Fig. 5.11 fs 680 1c, \log_{10} of the relative true residual norm, GMRES-MGS double-precision (plain) and GMRES-MGS variable precision with 64 decimal digits (dashed)

Table 5.2 GMRES, minimal relative true residual norms in k_{\max} iterations at most

matrix	k_{\max}	MGS	CGS	Hous	Ay
add32	200	7.16632e-15	9.43044e-15	7.84972e-15	8.06230e-15
ex37	150	1.75693e-13	1.68561e-12	7.67951e-14	1.76392e-13
memplus	900	9.17631e-12	3.34279e-09	6.90679e-12	9.16237e-12
sherman3	800	1.93397e-10	4.28744e-07	3.36321e-11	1.93152e-10
wang4	700	7.40182e-15	4.49377e-04	6.85406e-15	7.28993e-15
supg 100 6400	1000	4.30334e-14	9.21842e-12	2.32286e-14	4.41575e-14
supg 1 6400	1100	3.67990e-14	4.74708e-12	2.05530e-14	3.68537e-14
supg 01 6400	500	2.97491e-14	9.79334e-13	1.67327e-14	3.01096e-14
supg 001 6400	150	1.02025e-14	2.82313e-14	5.16872e-15	1.05888e-14
supg 0001 6400	120	2.06592e-14	2.09656e-14	1.07026e-14	2.05083e-14
supg 00001 6400	220	5.95295e-14	8.02870e-14	3.62220e-14	5.90865e-14
supg 000001 6400	240	1.19743e-13	1.68388e-13	6.91583e-14	1.20858e-13
convdiff xu 500 8100	1100	6.30921e-14	5.36650e-02	3.51598e-12	5.36787e-02
convdiff xu 1000 8100	700	2.36299e-13	9.64156e-02	1.43854e-13	9.64137e-02
convdiff xu 5000 8100	400	3.55342e-13	3.71635e-04	5.21509e-12	3.55330e-13

Table 5.3 shows the computing times obtained on the set of “large” matrices using the GMRES variants. Some of the computing times are quite large because we do a large number of iterations (given by k_{\max}) and the GMRES iterations are more and

more expensive. Therefore, as we have said above, we must not use full GMRES to solve these linear systems up to the maximum attainable accuracy. What is interesting here is to look at the differences between the several variants. As we have guessed our implementation of the Householder variant is more (and sometimes much more) expensive than MGS. CGS is a little cheaper than MGS but we have seen that it cannot be used reliably. The Ay variant is most of the time cheaper than MGS but the differences in the computing times are small.

Table 5.3 GMRES, computing times (seconds)

matrix	k_{\max}	MGS	CGS	Hous	Ay
add32	200	0.980611	0.973275	1.51398	0.717938
ex37	150	0.449253	0.410927	0.6884	0.3378
memplus	900	54.304	55.8388	80.2101	57.8597
sherman3	800	16.6504	16.3521	24.6492	16.4256
wang4	700	49.6746	49.0182	70.1525	48.8638
supg 100 6400	1000	33.8077	33.0857	48.234	32.5157
supg 1 6400	1100	40.9134	40.1707	58.3521	39.2235
supg 01 6400	500	8.09161	8.02283	11.9791	5.52345
supg 001 6400	150	0.702982	0.723042	1.06341	0.701577
supg 0001 6400	120	0.457168	0.482617	0.67911	0.455286
supg 00001 6400	220	1.50916	1.5167	2.26997	1.50868
supg 000001 6400	240	1.7984	1.77896	2.71219	1.79106
convdiff xu 500 8100	1100	48.119	46.8244	69.101	–
convdiff xu 1000 8100	700	19.0653	18.606	27.8888	–
convdiff xu 5000 8100	400	6.07594	5.98999	9.02163	6.00582

In many practical problems the users do not need to solve the linear systems up to the maximum attainable accuracy. Tables 5.4 and 5.5 show the number of iterations needed to have $\|r_k\| \leq 10^{-10}\|b\|$ starting with $x_0 = 0$. In some cases (denoted by a “–”) the stopping criterion was not satisfied within k_{\max} iterations. In most cases the number of iterations of the MGS and Householder variants are very close if not equal. Note that the stopping criterion uses the computed residual norm and not the norm of the true residual.

Table 5.4 GMRES, number of iterations for $\|r_k\| \leq 10^{-10}\|b\|$

matrix	k_{\max}	MGS	CGS	Hous	Ay
pde225	150	75	75	75	75
gre 343	300	260	263	277	260
jphwh 991	200	68	68	68	68
pde2961	320	238	238	238	238
jagmesh1	300	203	203	207	203
bfsa782	350	280	—	280	280
dw2048	1100	882	882	882	882
jagmesh2	800	700	700	704	700
raefsky2	450	361	361	361	361
fs 680 1c	150	86	86	86	86
add20	600	370	370	370	370
raefsky1	350	267	—	267	267
jagmesh4	700	601	601	609	601
fs 680 1	200	100	—	100	100
sherman1	500	388	388	391	388
nos3	320	266	266	266	266
sherman5	1200	1034	—	1034	1034
cavity05	600	463	—	463	463
e05r0500	260	236	—	236	236
comsol	300	247	—	247	247
olm1000	600	507	513	510	507
cavity10	900	714	—	714	714
steam2	200	—	—	178	193
1138bus	700	622	—	622	622
steam1	250	—	—	200	230
bcsstk26	1000	—	—	—	—
nos7	500	—	—	477	490
watt1	350	—	—	—	—
bcsstk14	1000	—	—	—	—
fs 183 6	60	35	—	35	35
bcsstk20	500	—	—	485	—
mcf6	820	—	—	—	—
nnc	250	—	—	219	—
Insp	420	—	—	—	—

Table 5.5 GMRES, number of iterations for $\|r_k\| \leq 10^{-10}\|b\|$

matrix	k_{\max}	MGS	CGS	Hous	Ay
add32	200	85	85	85	85
ex37	150	87	87	87	87
memplus	900	749	—	748	749
sherman3	800	—	—	—	—
wang4	700	446	—	446	446
supg 100 6400	1000	628	628	628	628
supg 1 6400	1100	888	888	888	888
supg 01 6400	500	331	331	331	331
supg 001 6400	150	117	117	117	117
supg 0001 6400	120	99	99	99	99
supg 00001 6400	220	184	185	189	184
supg 000001 6400	240	200	200	207	200
convdiff xu 500 8100	1100	933	—	1073	—
convdiff xu 1000 8100	700	535	—	586	—
convdiff xu 5000 8100	400	290	—	293	290

5.12.3 Prescribed convergence and stagnation

We consider constructing linear systems with a prescribed convergence curve for the GMRES residual norms. We take $n = 10$ and we define the relative residual norms as

$$g_1 = 1, \quad g_k = \alpha^{k-1} g_{k-1}, \quad k = 2, \dots, n, \quad (5.53)$$

with $\alpha = 0.9$. The eigenvalues are prescribed as complex conjugate pairs on the circle of radius $r = 2$ and the Ritz values are prescribed as complex conjugate pairs on a circle of radius βr with $\beta = 0.8$ and angles equal to those of the eigenvalues. Hence, if β is close to 1 we have Ritz values close to the eigenvalues and if β is small, the Ritz values are far from the eigenvalues.

The condition number of the constructed real matrix A is $6.9025 \cdot 10^6$ and $\|b\| = 1$. Figure 5.12 displays the prescribed eigenvalues and Ritz values at $k = 9$. Figure 5.13 shows the relative differences between the values of g and the true residual norms of GMRES-MGS. They are of the order of 10^{-10} .

If we modify the eigenvalues and/or the Ritz values, we obtain another linear system (even though the right-hand side may be the same) but with the same residual norms when we use GMRES to solve it.

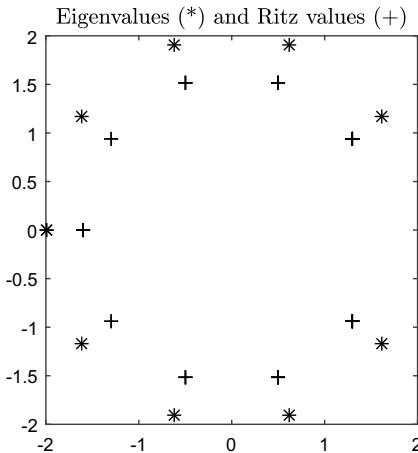


Fig. 5.12 Eigenvalues of A and Ritz values

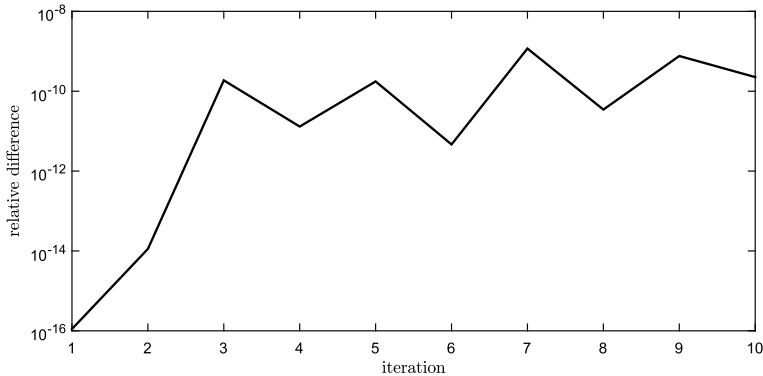


Fig. 5.13 Relative difference between the prescribed values g_k and the GMRES true generated residual norms

As we have seen, we can also easily construct linear systems for which GMRES exhibits partial or complete stagnation. Here, we construct a vector g as in (5.53) but with $\alpha = 0.6$ and we set $g_7 = g_8 = g_9$. It yields a matrix A with a condition number of 3251.2. The GMRES-MGS relative true residual norms are shown in Figure 5.14. The residual norms stagnate from iteration 7 to iteration 9.

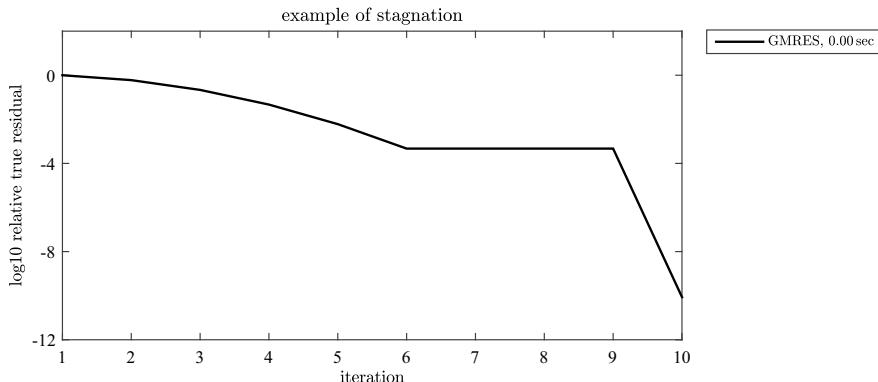


Fig. 5.14 GMRES-MGS true residual norms

5.12.4 Residual norm bounds

To show examples of residual bounds we first consider a linear system that was used in [463] and [914]. Let

$$Z = \begin{pmatrix} 1 & \sqrt{1-\delta} & 0 & \cdots & 0 \\ & \sqrt{\delta} & 0 & \cdots & 0 \\ & & 1 & \cdots & 0 \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}, \quad D = \text{diag}(20, 10, \mu_3, \dots, \mu_n),$$

$A = ZDZ^{-1}$ with $n = 20$, $\delta = 10^{-12}$ and $\mu_i, i = 3, \dots, 20$ are random numbers with a normal distribution in $[1, 5]$. Hence, the matrix A has two well isolated large eigenvalues. The matrix A has a condition number $5 \cdot 10^{11}$. The right-hand side is a vector with all components equal to 1. Comments on using the ε -pseudospectrum to bound the residual norms in this example are given in [463].

Figure 5.15 shows the GMRES-MGS relative true residual norms starting from $x_0 = 0$ and the bound obtained from

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \|X\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(\Lambda)c\|, \quad (5.54)$$

where $c = X^{-1}r_0/\|r_0\|$, X being the matrix of the eigenvectors of A ; see [914]. The norm of X is equal to $\sqrt{2}$. For this example (5.54) is a good bound of the residual norm except at the beginning of the iterations. The dot-dashed curve is the ideal GMRES bound, that is,

$$\frac{\|r_k^G\|}{\|r_0\|} \leq \min_{\substack{p \in \pi_k \\ p(0)=1}} \|p(A)\|.$$

This bound, which is the worst-case bound, is quite sharp for this example because the solution is relatively insensitive to changes in the right-hand side at least in the first part of the iterations. The bound was computed using the SDPT3 toolbox, see [916].

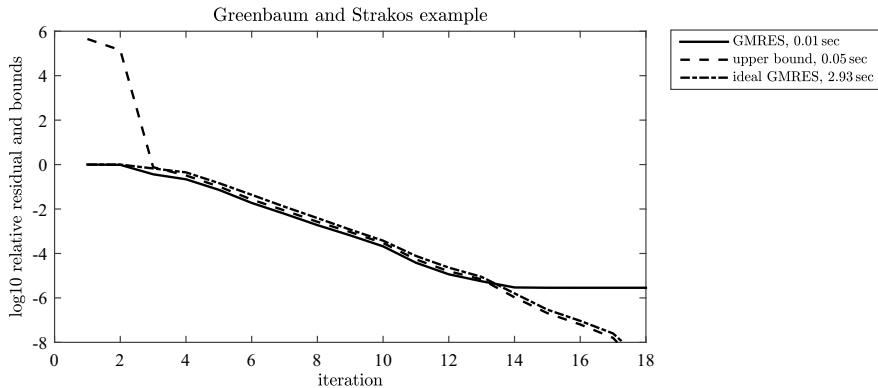


Fig. 5.15 Example from [914], GMRES-MGS true residual norms and bound (5.54)

The bound (5.54) is not so sharp for the problem `fs 680 1c` even though the slope of the curve is correct as we can see from Figure 5.16.

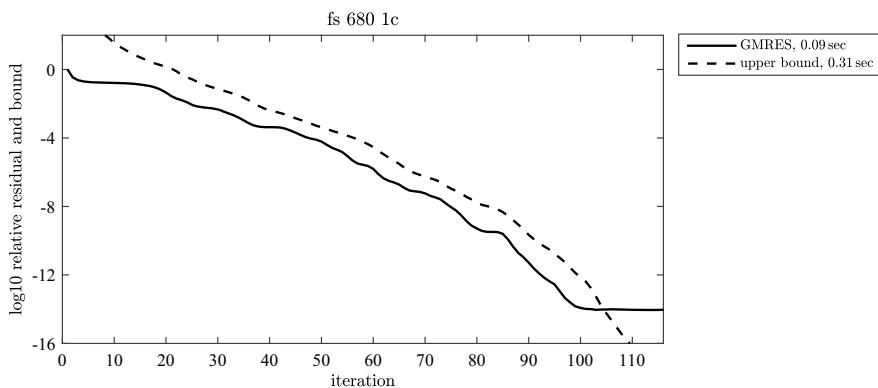


Fig. 5.16 `fs 680 1c`, GMRES-MGS true residual norms and bound

For the problem `supg 001` of order 1225 the bound (5.54) describes the slope of the decrease of the residual norm but not the values and it is much too large during the initial quasi-stagnation phase; see Figure 5.17.

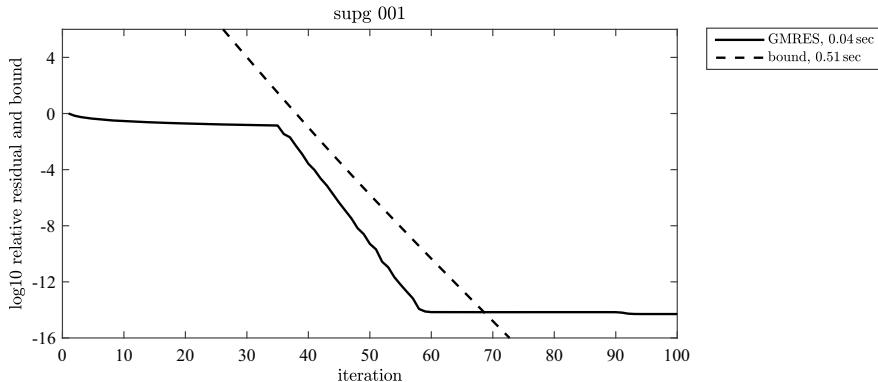


Fig. 5.17 `supg 001`, $n = 1225$, GMRES-MGS true residual norms and bound

5.12.5 Error norm estimates

Let us compare the error norm estimates we proposed in Section 5.8 with the “exact” error norm (computed with the result of $A \backslash b$). For `fs 183 6` and `fs 680 1c` we obtain good estimates of the error norm for a delay $d = 1$; see Figures 5.18–5.19. Unfortunately, things are not so nice when we have a long period of quasi-stagnation as we can observe in Figure 5.20 for the matrix `e05r0500`. At the beginning of the iterations the estimate is far from the error norm and it does not reproduce the stagnation. But things are getting better as the algorithm proceeds even though we have oscillations of the estimate.

Increasing the delay generally improves the approximation of the error norm. Unfortunately, so far, there is no known strategy to increase the delay adaptively. For discussions of this issue for symmetric problems, see [672, 674, 689].

5.13 Historical notes

Minimization techniques were introduced in numerical methods for solving linear equations soon after World War II. The Conjugate Gradient (CG) algorithm of Magnus R. Hestenes and Eduard Stiefel [515] published in 1952 minimizes the

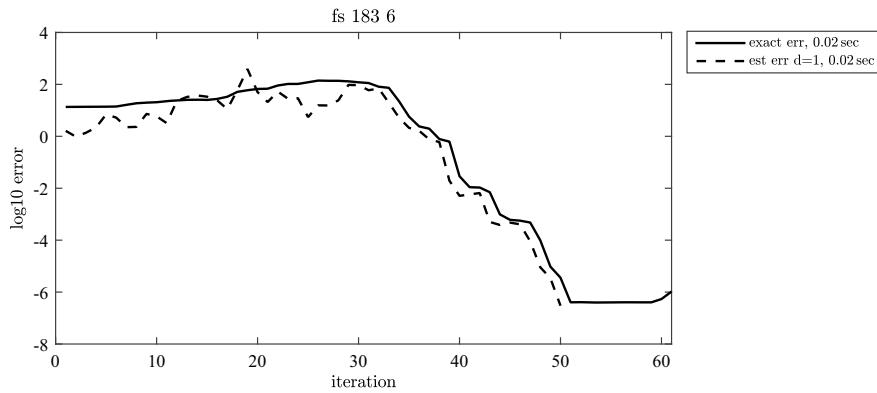


Fig. 5.18 fs 183 6, GMRES-MGS, exact error norm (plain) and estimate, $d = 1$ (dashed)

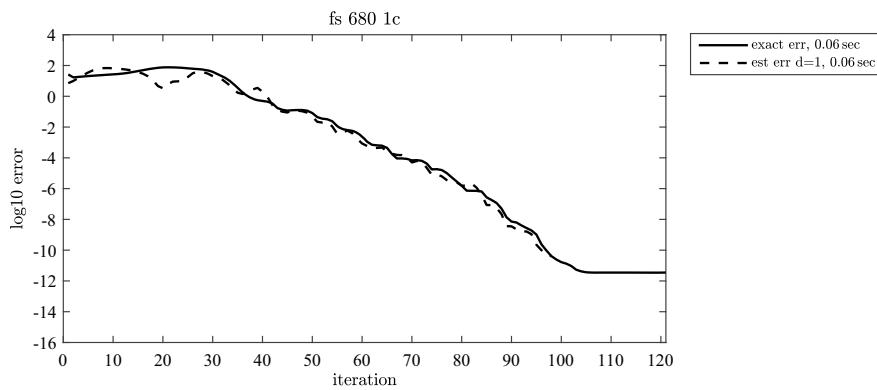


Fig. 5.19 fs 680 1c, GMRES-MGS, exact error norm (plain) and estimate, $d = 1$ (dashed)

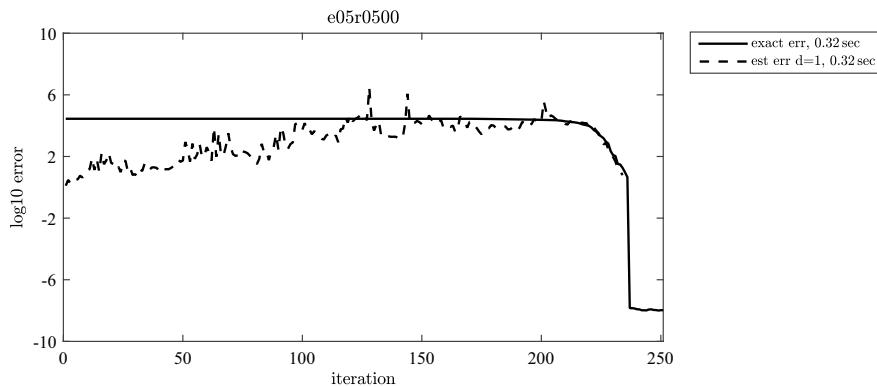


Fig. 5.20 e05r0500, GMRES-MGS, exact error norm (plain) and estimate, $d = 1$ (dashed)

A-norm of the error at every iteration for matrices that are symmetric and positive definite.

An early paper on iterative minimization techniques for general matrices is by Ishaq M. Khabaza [594] in 1963. The residual is written recursively as

$$r_k = (I - Ap_m(A))r_{k-1},$$

where p_m is a polynomial of degree m with $p_m(0) = 0$. The coefficients $\gamma_i^{(k)}$ of the polynomial are computed such that the norm $\|r_k\|$ is minimized. This is done by solving the normal equations of the corresponding least squares problem. We observe that, mathematically, the computed residual vectors are those that are obtained at the end of one cycle of the restarted method GMRES($m+1$). However, the basis which is used is the monomial basis with the vectors $A^j r_{k-1}$. Therefore, numerically, only small values of m can be used. In the (small) numerical examples of [594] the value $m = 2$ was used. There was no theoretical results about the convergence of the method.

Essentially the same method (without a reference to Khabaza) was proposed by Gury I. Marchuk and Yuri A. Kuznetsov [660] in 1968. The formulation for the residual was slightly more general,

$$r_{k+1} = r_k - \left[\sum_{i=1}^m \gamma_i^{(k)} (AB)^i \right] r_k,$$

where B is a symmetric positive definite matrix. However, conditions for the convergence of the algorithm are studied (Theorem 1, page 1042). A more detailed paper (117 pages) is [661] published in a book in 1974. The method is described in Chapter 4 page 77. However, what is more interesting is that, in Chapter 5 page 106, it is stated that solving the normal equations to compute the coefficients of the polynomial using the natural basis is not satisfactory and it is proposed to use an orthogonal basis for the subspace spanned by the vectors $B(AB)^i r_k$, $i = 1, \dots$. When $B = I$ this algorithm is essentially the restarted GMRES method with a different orthogonal basis. The differences are that the basis vectors are not of unit norm and the Gram–Schmidt algorithm is used to orthogonalize $A^i v_1$ and not Av_i . Unfortunately, this paper, which can be considered as describing an ancestor of GMRES, remained largely unnoticed, probably because it was published in a book and written in French. This book is, in fact, an outcome of the French–Soviet scientific relations on numerical analysis that were mainly due to the good relationships between G.I. Marchuk and Jacques-Louis Lions (1928–2001) on the French side.

Christopher C. Paige and Michael A. Saunders [737] in 1975 described the method MINRES that minimizes the residual norm for symmetric indefinite matrices. It is based on the symmetric Lanczos method. Note that when the matrix is symmetric, GMRES (which was introduced later in the western literature) reduces to MINRES.

In 1980 Yousef Saad [788] used the Arnoldi process for computing approximations of eigenvalues and proved some convergence results. He introduced the Incomplete Orthogonalization Method (IOM) by truncating the Arnoldi relation.

In 1981 Saad [789] introduced FOM (Full Orthogonalization Method) under the name “The method of Arnoldi” to solve linear systems. He considered again the IOM method and proved some convergence results using ellipses enclosing the eigenvalues. The name FOM was first used in [792]. In this paper Saad used the LU factorization with pivoting for IOM as well as the QR factorization of the upper Hessenberg matrix.

In 1985 Y. Saad and Martin H. Schultz [799] presented a general Petrov–Galerkin framework from which several Krylov methods known at that time can be obtained as particular cases.

GMRES was presented in the report [800] in May 1985 by Y. Saad and M.H. Schultz. However, the original version of the report was from August 1983. They used the Arnoldi process and Givens rotations to solve the least squares problems, a technique which is now considered as the standard implementation of GMRES. The paper [801] was received originally by the editors in November 1983, in revised form in May 1985 and published in July 1986. The published version is similar to the report [800].

We observe that bounds on GMRES residual norms can be obtained mathematically from any method which is equivalent to GMRES. However, these methods may not have exactly the same numerical behavior. Therefore, in this note, we concentrate on bounds that had been devised specifically for GMRES. Hundreds of papers have been written about the convergence of FOM and GMRES since the Saad and Schultz paper [801] appeared in 1986. It has been noticed from the beginning that the FOM residual norms may have an erratic behavior whence GMRES residual norms are monotonically decreasing by construction.

Peter N. Brown [152] in 1991 established the relation between FOM (called the Arnoldi algorithm in that paper) and GMRES residual norms and the fact that when GMRES stagnates the upper Hessenberg matrices H_k are singular. This phenomenon is called the peak–plateau behavior. It has been studied also in a more general setting by Jane K. Cullum [242, 244] in 1995 and J.K. Cullum and Anne Greenbaum [245] in 1996.

In 1993 Henk van der Vorst and Cornelis (Kees) Vuik [943] proved some bounds for the GMRES residual norms. Their aim was to explain the superlinear convergence behavior of GMRES that is sometimes observed. They attributed this phenomenon to the convergence of the Ritz values to the eigenvalues of the matrix. Unfortunately, we have seen that this is not always true since we can construct examples with a prescribed (superlinear) convergence curve and prescribed eigenvalues and Ritz values. The potential convergence of the harmonic Ritz values does not also explain the superlinear convergence phenomenon.

A. Greenbaum and Lloyd N. Trefethen [464] formulated in 1994 the ideal Arnoldi and GMRES matrix approximation problems to get rid of the influence of the right-hand side of the linear system. They stated the uniqueness of the solution of the ideal GMRES approximation problem when the matrix is nonsingular. See the paper

[645] by Jörg Liesen and Petr Tichý in 2009 for extensions and a better proof of uniqueness.

Vance Faber, Wayne D. Joubert, Emanuel Knill and Thomas A. Manteuffel [333] in 1996 constructed a small example for which the worst-case GMRES bound was different from the ideal GMRES one. Kim-Chuan Toh [915] in 1997 provided a simpler example. Toh and Trefethen [917] described in 1998 an algorithm to compute the polynomial that minimizes $\|p(A)\|$. These polynomials are called Chebyshev polynomials of a matrix.

The worst-case GMRES problem was studied later on by J. Liesen and P. Tichý in 2004 [642], 2007 (with V. Faber) [911] and 2013 [337] (again with V. Faber). In the same vein, in 2009 J. Liesen and P. Tichý [645] studied the approximation of functions of matrices $f(A)$ by polynomials in the Euclidean norm. See also the paper [647] in 2014. An unpublished report by Mario Arioli [23] in 2009 also studied the worst-case problem.

Now, let us consider the three main tools people used to study GMRES convergence. The first one is the field of values of a matrix. This was introduced as a way to study the convergence properties of some iterative methods (such as the semi-iterative Chebyshev method) in 1993 by Michael Eiermann [295]. This tool was also used by Gerhard Starke [874] in 1997. The results for GMRES using the field of values are summarized in the Habilitation thesis of Oliver G. Ernst [328] in 2000 and in the nice paper by M. Eiermann and O. G. Ernst [296] in 2001 (page 47). In 2012 J. Liesen and P. Tichý [646] gave a simple and direct proof that the bounds using the field of values also holds for the ideal GMRES approximation.

The second tool is the polynomial numerical hull which can be considered as a generalization of the field of values. The polynomial numerical hull of degree k was introduced by Olavi Nevanlinna [715] in 1993. In the paper [457] in 2002, Anne Greenbaum studied properties and equivalent definitions of the polynomial numerical hull. She also computed some examples of polynomial numerical hulls. In 2003 V. Faber, A. Greenbaum and Donald E. Marshall characterized the polynomial numerical hulls of Jordan blocks [332]. In 2004 James V. Burke and A. Greenbaum gave six characterizations of the polynomial numerical hull of a matrix and proved results for polynomial numerical hulls of Toeplitz matrices; see [161, 162]. A. Greenbaum in 2004 derived lower bounds on the norms of functions of Jordan blocks and triangular Toeplitz matrices [458]. She also derived estimates of the convergence rate of ideal GMRES applied to a Jordan block. Upper and lower bounds on norms of functions of matrices were considered in the paper [459] in 2009. Daeshik Choi and A. Greenbaum in 2015 examined the relations of Crouzeix's conjecture to GMRES convergence; see [219]. In [911] in 2007 P. Tichý, J. Liesen and V. Faber showed that for a Jordan block of order n ideal and worst-case GMRES are identical at steps k and $n - k$ such that k divides n . They also derived explicit expressions for the $(n - k)$ th ideal GMRES approximation; see also [644, 648].

The third tool is the pseudospectrum. This was essentially promoted by L.N. Trefethen. Bounds for the GMRES residual norms were given in [924] in 1990. The

computation of the pseudospectrum is discussed in [925] in 1999. For applications of the pseudospectrum, see the book [926] co-authored with Mark Embree in 2005.

The successes and failures of these three tools for non-normal matrices were discussed by M. Embree in the report [317] in 1999. This is illustrated with six examples. Unfortunately, none of these tools was able to succeed in all the examples, showing the limits of these approaches.

Some other papers were concerned with GMRES convergence without relying specifically on the three tools we have seen above.

Stephen L. Campbell, Ilse C.F. Ipsen, Carl T. Kelley and Carl D. Meyer [177] in 1996 gave some bounds for the residual norms for cases where the matrix A has its eigenvalues in one or several clusters with a few outliers. Their model says that in the case of a single cluster the asymptotic rate of convergence of GMRES (if this has any sense!) is proportional to the size of the cluster with a constant reflecting the non-normality of A and the distance of the cluster from the outliers. They also extended these results to the case of several clusters.

In 1998 Marlis Hochbruck and Christian Lubich [526] gave bounds for the error norm for FOM, GMRES, BiCG and QMR in terms of projections. They also studied the relations between the residual norms of these methods.

In the report [546] in 1998, I.C.F. Ipsen showed that the GMRES residual norm is large as long as the Krylov basis is well conditioned. This was obtained by expressing the minimal residual norm in iteration k in terms of the pseudo-inverse of the Krylov matrix in iteration $k + 1$. Examples of exact expressions for residual norms were given for scaled Jordan blocks. The factorization of the Krylov matrix for a normal matrix was used to derive upper and lower bounds for the residual norms, involving a subset of the eigenvalues of A . Some of these results were published in the paper [548] in 2000.

In 2000 J. Liesen introduced some computable residual norm bounds for GMRES in [633]. His work using GMRES-equivalent matrices will be described below together with other papers using the same technique.

In the paper [635] in 2002 J. Liesen, Miroslav Rozložník and Zdeněk Strakoš presented identities and bounds for the norm of the residual of overdetermined least squares problems and applied these results to minimal residual norm Krylov methods.

In 2005 Bernhard Beckermann, Serguei A. Goreinov and Eugene E. Tyrtyshnikov [84] improved Howard Elman's bound for the residual norm. Elman's bound was, in fact, derived for GCR (see Chapter 6) and is valid only if the origin is not in the field of values of A . See also [83].

Valeria Simoncini and Daniel B. Szyld [836] in 2005 studied the so-called super-linear convergence of GMRES using spectral projectors. They essentially show that when the subspace $A\mathcal{K}_k(A, r_0)$ has captured some linearly independent vectors close to vectors in Q , an invariant subspace of A , GMRES behaves almost like an other GMRES process applied to an initial vector $(I - P_Q)r_k$ with no components in the invariant subspace where P_Q is the spectral projector onto the range of Q .

In 2011 the paper [678] provided formulas for the residual norms of FOM and GMRES involving a triangular submatrix of the Hessenberg matrix H_k . Lower and upper bounds using the singular values of this submatrix were given.

Bounds involving the initial residual for diagonalizable matrices were given in 2014 in the paper [914] by David Titley-Peloquin, Jennifer Pestana and Andrew J. Wathen.

Convergence of GMRES has also been studied for particular classes of linear systems. Problems arising from the discretization of convection–diffusion equations have been considered by O.G. Ernst [329] in 2000. This problem was studied by J. Liesen and Z. Strakoš [638] in 2005; see also J. Liesen and P. Tichý [641] in 2004.

A class of matrices that has received much attention are tridiagonal Toeplitz matrices, that is, matrices with constant diagonals. Eigenvalues and eigenvectors of these matrices are known explicitly. J. Liesen and Z. Strakoš [637] in 2004 considered the problem where the Toeplitz matrix is a small perturbation of a scaled Jordan block. The behavior of GMRES for tridiagonal Toeplitz systems was also studied in Wei Zhang’s Ph.D. thesis [1031] in 2007. He used Chebyshev polynomials of the first and second kind. These results were published in the paper [630] by Ren-Cang Li and W. Zhang in 2009. Simpler formulas were obtained for right-hand sides $b = e_1$ and $b = e_n$ by the same authors [631] in 2009 using Chebyshev polynomials of the second kind. Symmetric and normal tridiagonal Toeplitz matrices were considered by R-C. Li in [629].

GMRES for matrices with a skew-symmetric part of low rank was studied by Bernhard Beckermann and Lothar Reichel in 2008 [85]. This method was studied and improved by Mark Embree, Josef A. Sifuentes, Kirk M. Soodhalter, Daniel B. Szyld and Fei Xue [320] in 2012.

It was already known at the beginning of the 1990s that eigenvalues alone cannot explain GMRES convergence for non-normal matrices. This was first clearly shown in the 1994 paper [463] by Anne Greenbaum and Zdeněk Strakoš. They studied the matrices B that generate the same Krylov residual space as the one given by the pair (A, b) . Moreover, it was shown that the spectrum of B can consist of arbitrary nonzero values. In the paper [460] with Vlastimil Pták, this was extended in 1996 by proving that any non-increasing sequence of residual norms can be generated by GMRES. The paper [34] by M. Arioli, V. Pták and Z. Strakoš in 1998 closed this series of papers with a full parametrization of the class of matrices and right-hand sides giving prescribed convergence history while the system matrix has prescribed nonzero spectrum; see also [681], for a survey we refer to [640], Section 5.7. In [287] in 2012 a parametrization was given of the class of matrices and right-hand sides generating, in addition to prescribed residual norms and eigenvalues, prescribed Ritz values in all iterations. In the paper [281] with Kui Du in 2017 it was shown that, instead of the Ritz values, one can prescribe the harmonic Ritz values. Note that the harmonic Ritz values are the roots of the GMRES residual polynomials. Prescribing the behavior of early terminating GMRES was studied in the paper [288] in 2014. A study of the role eigenvalues play in forming GMRES residual norms with non-

normal matrices was done in 2015 [686]. Some of these results were extended to block GMRES in 2019 by Marie Kubínová and Kirk Soodhalter, see [610].

A small example where GMRES stagnates for one iteration was given by Saad and Schultz [801] page 865. A larger example was given in 1992 by Nöel M. Nachtigal, Satish C. Reddy and L.N. Trefethen [711]. It is an orthogonal matrix with $1 - z^n$ as minimal polynomial. Such an example was also mentioned by P.N. Brown [152].

Complete stagnation of GMRES was studied by Ilya Zavorin, Dianne P. O'Leary and Howard C. Elman [1017] in 2003; see also Zavorin's thesis [1016] in 2001. They used $x_0 = 0$, $\|b\| = 1$ and the factorization (2.36) $K_{n,k} = XD_c \mathcal{V}_{n,k}$ that was originally presented in [548]. Theorem 2.2 states that GMRES completely stagnates if and only if $\mathcal{V}_{n,k}^* D_c^* W y = e_1$ where $W = X^* X$ and $y = X^{-1} b$. They also proved that there are no normal matrices with real eigenvalues that stagnate, but there are stagnating normal matrices with complex eigenvalues.

In 2008 V. Simoncini and D.B. Szyld [838] gave conditions involving the symmetric (or Hermitian) part and the skew-symmetric parts of A implying that GMRES is not stagnating. These results were somehow extended by V. Simoncini [829] in 2010.

The complete stagnation of GMRES was studied for $n \leq 4$ [679] in 2012. This paper considered the problem of the (non-)existence of stagnation vectors, that is, for a given matrix does it exist right-hand sides giving complete stagnation? Results from algebraic geometry were used. Unfortunately only small systems could be studied in this way.

Necessary and sufficient conditions for GMRES complete and partial stagnation were given in the paper [682] in 2014. This paper gives a complete characterization of those linear systems for which we have complete or partial stagnation. Moreover, it showed that it is easy to construct examples of linear systems for which GMRES stagnates.

Stagnation of block GMRES was considered in 2017 by K. Soodhalter [868].

Many researchers concentrated on obtaining bounds for the residual norms. However, exact expressions for these norms were found starting in 2000. I.C.F. Ipsen gave in [548, Theorem 4.1] in 2000 an exact expression for the GMRES residual norm $\|r_k^G\|$ with normal matrices using a minimization problem over $k + 1$ distinct eigenvalues.

In [642, Theorem 2.1] in 2004 J. Liesen and P. Tichý gave an exact formula for the residual norms for normal matrices when $k = n - 1$.

Hassane Sadok [806] in 2005 established expressions for the residual norms involving the singular values of the matrices H_k and \underline{H}_k .

J. Duintjer Tebbens, G. Meurant, H. Sadok and Z. Strakoš [291] gave in 2014 exact expressions for the GMRES residual norms for normal matrices. They involve the eigenvalues, the eigenvector matrix X and the projections $X^* r_0$ of the initial residual on the eigenvectors. The idea of using Cramer's rule and the Cauchy-Binet formula to obtain these expressions is originally from H. Sadok, G. Meurant and J. Duintjer Tebbens [686] in 2015 extended these results to diagonalizable matrices.

Concerning the residual polynomials, Roland W. Freund [367] introduced the concept of quasi-kernel polynomials in 1992 to study Q-MR iterative methods. Serge Goossens and Dirk Roose [440] proved in 1999 that the harmonic Ritz values are the roots of the GMRES residual polynomials.

Exact expressions for the coefficients of the FOM and GMRES residual polynomials as functions of the eigenvalues and eigenvectors were given in the paper [684] in 2017; see Section 5.6.

The study of GMRES convergence using unitary matrices was motivated by the paper of A. Greenbaum and Z. Strakoš [463] in 1994. In this paper the authors wrote “*If, for each vector b , we can find a matrix B of the given form, for which we can analyze the behavior of the GMRES method applied to B , then we can also analyze the behavior of the GMRES method applied to A* ”. They suggested to find such an equivalent matrix B which is unitary. This was considered by J. Liesen [633] in 2000. He used an orthogonal basis of $A\mathcal{K}_k(A, r_0)$ and constructed matrices that are GMRES-equivalent by using the QR factorization of H . He derived bounds for the residual norms when the matrix is unitary. These bounds involve the gaps of the eigenvalues on the unit circle. A large gap may induce fast convergence. One of this bound is computable if one uses the simpler GMRES algorithm of Homer F. Walker and Lu Zhou [969] described in Chapter 6. His bounds depend on the initial residual. Leonid Knizhnerman [604] in 2000 showed an inverse result, namely that fast GMRES convergence implies a large gap in the spectrum of Q in a certain RQ factorization of an upper Hessenberg matrix obtained from GMRES. It was also shown that the entries of this particular Q can be expressed in terms of the residual norms only [604, Section 6.1]. Knizhnerman’s results were used in 2003 by Gang Xie [992] to obtain bounds on the GMRES residual norms. These issues were further explored in the paper [291] in 2014 where the authors studied to what extent GMRES convergence can be explained using unitary GMRES (A, b) -equivalent pairs, eventually involving different right-hand sides. They characterized the matrices B in terms of orthonormal bases for the sequence of Krylov subspaces $\mathcal{K}_k(A, b)$ and Krylov residual subspaces $A\mathcal{K}_k(A, b)$, $k = 1, 2, \dots, n$. This shows that a possible linking of the spectral properties of unitary GMRES (A, b) -equivalent matrices, which influence GMRES convergence behavior, to some simple properties of A would be, in general, rather difficult. They presented exact expressions giving the residual norms for normal matrices and showed that they can, for some particular eigenvalue distributions, explain acceleration of GMRES convergence observed after a number of iterations (the so-called superlinear convergence).

Estimates of the norm of the error for FOM and GMRES were given in the paper [677] in 2011. This is done using the same techniques as for symmetric matrices and CG; see [249, 250, 350, 430, 435, 672, 674, 675]. Even though their computations is more expensive than in the symmetric case, these estimates can provide a more reliable way to stop the iterations than the criterion based on the residual norms. Other methods to obtain estimates of the norm of the error are described by Giles Auchmuty in [42] and Claude Brezinski in [128]; see also [145, 146] in 2008–2009 by C. Brezinski, Giuseppe Rodriguez and Sebastiano Seatzu.

Implementations of GMRES different from the standard one using the solve of the least squares problems with Givens rotations were proposed over the years. Since, at that time, the backward stability of GMRES-MGS has not been proved yet, H.F. Walker [964] proposed in 1985 to use Householder reflections to compute the orthonormal basis vectors. This was published in a journal [965] in 1988; see also [966].

Another proposal was done by El Hassan Ayachour [49] in 2003. The least squares problem is solved by a partitioning of the Hessenberg matrix H_k . The number of operations per iteration is smaller than for the GMRES implementation using Givens rotations. So far, the stability of this approach has not been studied theoretically.

Weighted inner products for GMRES were considered by M. Embree, Ronald B. Morgan and Huy V. Nguyen [319] in 2017.

Codes implementing GMRES were described by Valérie Frayssé, Luc Giraud, Serge Gratton and Julien Langou [364] in 2005. Some other implementations were considered to introduce more parallelism in GMRES. We could cite Zhaojun Bai, Dan T. Hu and Lothar Reichel [58, 59] in 1992 and 1994, Daniela Calvetti, Johnny Petersen and L. Reichel [173] in 1993, Jocelyne Erhel [321] in 1995, Eric de Sturler and Henk van der Vorst [263] in 1995, Roger B. Sidje [824] in 1997 and Bernard Philippe and L. Reichel [756] in 2012. All these papers used Newton bases to parallelize the construction of the basis vectors. A study of the influence of the orthogonalization scheme on the parallel performance of GMRES was done in the paper [365] by Valérie Frayssé, Luc Giraud and Hatim Kharraz-Aroussi in 1998.

The use of s -step Krylov methods for parallel computers started at the end of the 1980s with the conjugate gradient method for symmetric positive definite systems; see, for instance, [224] by Anthony T. Chronopoulos and Charles W. Gear; see also [222, 225]. The study of communication-avoiding algorithms started at the beginning of the 21st century; see [265] by Jim Demmel, Laura Grigori, Mark Hoemmen and Julien Langou in 2012. CA-GMRES was described in the Ph.D. thesis of M. Hoemmen [527] in 2010. Previous works were analyzed in chapter 1 of this thesis.

Hiding global communication latency in GMRES was considered by Pieter Ghysels, Thomas J. Ashby, Karl Meerbergen and Wim Vanroose [415] in 2013. Unfortunately, this work did not pay too much attention to the stability and to the maximum attainable accuracy of the method.

Since there are different variants of GMRES, their behavior in finite precision arithmetic has been a concern from the beginning. In fact, this was the motivation for the introduction of the Householder implementation of GMRES which was assumed to be more stable than those based on CGS or MGS. The report [589] by R. Karlson in 1991 studied some rounding error effects in GMRES. This work is based on forward estimates and experimental observations. The influence of orthogonality on the Arnoldi process was considered by Thierry Braconnier and Philippe Langlois and Jean-Claude Rioual [119] in 2000.

The numerical stability of the Householder implementation of GMRES was considered in the paper [279] by Jitka Drkošová, A. Greenbaum, M. Rozložník and

Z. Strakoš in 1995. The relation between the true and computed residual in the presence of rounding errors is analyzed. Under some restrictions on the condition number of A , the backward stability of the Householder implementation of GMRES is proved. For more details, see the Ph.D. thesis of M. Rozložník [774] in 1996, chapters 3 and 4. It is also proved that the Arnoldi basis vectors for the modified Gram–Schmidt implementation lose their linear independence only after the GMRES residual norm has been reduced to its final level of accuracy. The conclusion is that GMRES-MGS can be used safely.

M. Arioli and Claudia Fassino [28] in 1996 did a forward rounding error analysis of the Householder implementations of FOM and GMRES.

The influence of orthogonality of the basis vectors in the GMRES method was studied by means of a numerical example by M. Rozložník, Z. Strakoš and Miroslav Tůma [777] in 1996.

In [461] A. Greenbaum, M. Rozložník and Z. Strakoš in 1997 considered the linear independence of the MGS-based basis vectors. The backward stability of GMRES-MGS was finally proved by C.C. Paige, M. Rozložník and Z. Strakoš [736] in 2006. This is based on previous results on the stability of MGS but the last steps to obtain the backward stability were far from being trivial.

Curiously enough, some people still do think that one must use the Householder-based implementation to obtain a stable version of GMRES.

Chapter 6

Methods equivalent to FOM or GMRES



In this chapter we review iterative methods which are equivalent to FOM or GMRES in the sense that, mathematically, they must deliver the same residual norms. However, as we will see, this is not always the case in finite precision arithmetic. The algorithms mathematically equivalent to GMRES either construct residual vectors r_k orthogonal to $A\mathcal{K}_k(A, r_0)$ or explicitly minimize the residual norms.

6.1 GCR, Orthomin, Orthodir and Axelsson's method

The Generalized Conjugate Residual method (GCR) was introduced in [301, 308] as a generalization of the Conjugate Residual (CR) method for symmetric matrices [888]. We first consider a simple descent method. Let $p_0 = r_0 = b - Ax_0$. Then, for $k = 0, \dots$ the iterates and the residual vectors are constructed as

$$\begin{aligned}\alpha_k &= \frac{(r_k, Ap_k)}{(Ap_k, Ap_k)}, \\ r_{k+1} &= r_k - \alpha_k Ap_k, \\ x_{k+1} &= x_k + \alpha_k p_k.\end{aligned}$$

The choice of α_k minimizes the norm of $\|b - A(x_k + \alpha_k p_k)\|$ as a function of α . In the steepest descent method, one takes $p_k = r_k$. However, we would like to minimize the residual norm in $\mathcal{K}_{k+1}(A, r_0)$. This is done by constructing the direction vectors p_j to be $A^T A$ (or $A^* A$ in the complex case) orthogonal. Since the vectors Ap_j will give a basis of $A\mathcal{K}_{k+1}(A, r_0)$, we will obtain residual vectors orthogonal to $A\mathcal{K}_{k+1}(A, r_0)$. The vectors p_j are defined by a long recurrence involving all the previous direction vectors,

$$p_{k+1} = r_{k+1} + \sum_{j=0}^k \beta_j^{(k)} p_j.$$

The coefficients $\beta_j^{(k)}$ are

$$\beta_j^{(k)} = -\frac{(Ar_{k+1}, Ap_j)}{(Ap_j, Ap_j)}.$$

We note that to compute the coefficients we need to have Ar_{k+1} and Ap_j for $j = 0, \dots, k$. To avoid two matrix–vector products per iteration the vectors Ap_j are constructed by

$$Ap_{k+1} = Ar_{k+1} + \sum_{j=0}^k \beta_j^{(k)} Ap_j,$$

and stored. The properties of the residual and direction vectors are given in the following theorem from [301] (Theorem 3.1, p. 348).

Theorem 6.1 *For the GCR method as defined above,*

- $(Ap_i, Ap_j) = 0, i \neq j$,
- $(r_i, Ap_j) = 0, i > j$,
- $(r_i, Ap_i) = (r_i, Ar_i)$,
- $(r_i, Ar_j) = 0, i > j$,
- $(r_j, Ap_i) = (r_0, Ap_i), i \geq j$,
- $\{p_0, \dots, p_{k-1}\}$ and $\{r_0, \dots, r_{k-1}\}$ span $\mathcal{K}_k(A, r_0)$,
- x_{k+1} minimizes the residual norm over $x_0 + \text{span}(p_0, \dots, p_k)$.

Most of these properties can be proved using induction. The vectors p_0, \dots, p_{k-1} and r_0, \dots, r_{k-1} give two bases of $\mathcal{K}_k(A, r_0)$. The first one is $A^T A$ orthogonal, and the second one is semi A-conjugate. This property shows that r_k is orthogonal to $A\mathcal{K}_k(A, r_0)$. From Theorem 5.1 this minimizes the norm of r_k over the Krylov subspace. Therefore, GCR is equivalent to GMRES in the sense defined above. GCR was originally intended for matrices with a definite symmetric part. If this is the case then, $r_i \neq 0$ implies $p_i \neq 0$. It was for GCR that the bound in relation (5.4) was proved when the symmetric part of A is positive definite. In this case, GCR converges and, mathematically, it gives the exact solution in at most n iterations, like GMRES.

In GCR, at iteration k , we have $k + 3$ dot products (if we include the computation of the norm of the residual), $2k + 4$ additions as well as multiplications and $k + 1$ divisions plus one matrix–vector product. GCR is easy to implement as we can see in the following code.

```
function [x,ni,resn] = GCR(A,b,x0,epsi,nitmax);
%
n = size(A,1);
nb = norm(b);
x = x0;
```

```

r = b - A * x;
P = zeros(n,nitmax+1);
AP = zeros(n,nitmax+1);
resn = zeros(1,nitmax+1);
pAAp = zeros(nitmax+1,1);
p = r;
Ap = A * p;
Ar = Ap;
P(:,1) = p;
AP(:,1) = Ap;
pAAp(1) = Ap' * Ap;
resn(1) = norm(r);
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    alpha = (r' * Ap) / pAAp(k);
    r = r - alpha * Ap;
    x = x + alpha * p;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    if nresidu < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nresidu
    Ar = A * r; % matrix vector product
    beta = -(Ar' * AP(:,1:k))' ./ pAAp(1:k);
    p = r + P(:,1:k) * beta;
    Ap = Ar + AP(:,1:k) * beta;
    P(:,k+1) = p;
    AP(:,k+1) = Ap;
    pAAp(k+1) = Ap' * Ap;
end % for k
resn = resn(1:ni+1);

```

In this code the vector β is computed in this strange way to avoid transposing the array AP . Only two vector transposes are needed.

Mathematically, GCR is equivalent to GMRES and its implementation is simpler since there is no need to solve a linear system with an upper Hessenberg matrix. However, contrary to GMRES, GCR can potentially break down if for some index j , $(Ap_j, Ap_j) = 0$.

A predecessor of GCR named Orthomin(d) was proposed in [953]. In this method, the recurrence defining the direction vectors is truncated,

$$p_{k+1} = r_{k+1} + \sum_{j=k-d+1}^k \beta_j^{(k)} p_j.$$

Thus p_{k+1} is only $A^T A$ -orthogonal to the last d vectors p_j . Orthomin is only mathematically equivalent to GCR when we keep all the vectors p_j . This is known as Orthomin(∞). The convergence of Orthomin(d) when the symmetric part of A is positive definite was proved in [301, 308].

Orthodir was introduced in [1014]; see also [559, 560]. The method was defined using an auxiliary matrix Z which defined the dot product. A simplified version using $Z = A^T$ was given in [308]; see also [1018]. The residual vectors and the approximate solutions are computed as in GCR, and the direction vectors are computed by

$$p_{k+1} = Ap_k + \sum_{j=0}^k \beta_j^{(k)} p_j, \quad \beta_j^{(k)} = -\frac{(A^2 p_k, Ap_j)}{(Ap_j, Ap_j)}.$$

The coefficients are chosen to give a set of $A^T A$ -orthogonal vectors. As GCR, Orthodir can also break down. A code for Orthodir is the following. Note that to avoid two matrix–vector products per iteration, Ap_{k+1} is computed as

$$Ap_{k+1} = A(Ap_k) + \sum_{j=0}^k \beta_j^{(k)} Ap_j.$$

But this may give some stability problems.

```
function [x,ni,resn] = Orthodir(A,b,x0,epsi,nitmax);
%
n = size(A,1);
nb = norm(b);
x = x0;
r = b - A * x;
P = zeros(n,nitmax+1);
AP = zeros(n,nitmax+1);
resn = zeros(1,nitmax+1);
pAAp = zeros(nitmax+1,1);
p = r;
Ap = A * p;
P(:,1) = p;
AP(:,1) = Ap;
pAAp(1) = Ap' * Ap;
resn(1) = norm(r);
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    alpha = (r' * Ap) / pAAp(k);
    r = r - alpha * Ap;
    x = x + alpha * p;
```

```

nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
A2p = A * Ap; % matrix vector product
beta = -(A2p' * AP(:,1:k))' ./ pAAp(1:k);
p = Ap + P(:,1:k) * beta;
Ap = A2p + AP(:,1:k) * beta;
P(:,k+1) = p;
AP(:,k+1) = Ap;
pAAp(k+1) = Ap' * Ap;
end % for k
resn = resn(1:ni+1);

```

An algorithm called GCG-LS was proposed in [43, 44] by O. Axelsson. The formulas, as given in [308], are

$$x_{k+1} = x_k + \sum_{j=0}^k \alpha_j^{(k)} p_j, \quad r_{k+1} = r_k - \sum_{j=0}^k \alpha_j^{(k)} A p_j \quad (6.1)$$

$$\beta_k = -\frac{(Ar_{k+1}, Ap_k)}{(Ap_k, Ap_k)},$$

$$p_{k+1} = r_{k+1} + \beta_k p_k, \quad Ap_{k+1} = Ar_{k+1} + \beta_k Ap_k.$$

In this method we have a long recurrence for r_{k+1} and a short recurrence for p_{k+1} . The coefficients $\alpha_j^{(k)}$ are computed to minimize the norm of the residual. It amounts to solving the least squares problem,

$$\min \|B^{(k)}\alpha^{(k)} - r_k\|,$$

with $B^{(k)} = (Ap_0 \cdots Ap_k)$ and $\alpha^{(k)}$ is the vector of the coefficients $\alpha_j^{(k)}$. We observe that the matrix $B^{(k)}$ is an $n \times (k+1)$ dense matrix. Since the cost of solving the least squares problem increases with k , what was introduced in [43] was a truncated version of this algorithm by keeping only a small number of vectors in the recurrences (6.1).

We have $(r_{k+1}, Ar_j) = 0$, $j \leq k$. Hence, we have a conjugate residual method. In [43] monotone convergence was proved when the symmetric part of A is positive definite. There is a breakdown when r_{k+1} and p_k , and hence p_{k+1} and p_k , are collinear.

A code for the full version of this algorithm is the following, where the least squares problems are solved with Householder reflections.

```

function [x,ni,resn] = Axel(A,b,x0,epsi,nitmax);
%
n = size(A,1);
nb = norm(b);
x = x0;
r = b - A * x;
P = zeros(n,nitmax+1);
AP = zeros(n,nitmax+1);
R = zeros(nitmax,nitmax);
resn = zeros(1,nitmax+1);
p = r;
Ap = A * p;
P(:,1) = p;
AP(:,1) = Ap;
pAAp = Ap' * Ap;
resn(1) = norm(r);
resnt(1) = resn(1);
ni = 1;
[Q,bet] = household(Ap,2);
qa = Q(:,1)' * Ap;
R(1,1) = Ap(1) - bet(1) * qa * Q(1,1);
rhs = apply_house(Q,bet,1,1,r);
alpha = R(1,1) / rhs(1); % least squares problem
r = r - AP(:,1) * alpha;
x = x + P(:,1) * alpha;
resn(2) = norm(r);
resnt(2) = norm(b - A * x);
Ar = A * r;
beta = -(Ar' * Ap) / pAAp;
p = r + beta * p;
Ap = Ar + beta * Ap;
P(:,2) = p;
AP(:,2) = Ap;
pAAp = Ap' * Ap;
%
for k = 2:nitmax
    ni = ni + 1; % number of iterations
    u = apply_house(Q,bet,1,k-1,AP(:,k));
    [Q,bet] = qr_add_col(Q,k-1,bet,u);
    R(1:k-1,k) = u(1:k-1);
    qa = Q(:,k)' * u;
    R(k,k) = u(k) - bet(k) * qa * Q(k,k);
    rhs = apply_house(Q,bet,1,k,r);
    alpha = R(1:k,1:k) \ rhs(1:k); % least squares problem
    r = r - AP(:,1:k) * alpha;
end

```

```

x = x + P(:,1:k) * alpha;
nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
Ar = A * r; % matrix vector product
beta = -(Ar' * Ap) / pAAp;
p = r + beta * p;
Ap = Ar + beta * Ap;
P(:,k+1) = p;
AP(:,k+1) = Ap;
pAAp = Ap' * Ap;
end % for k
resn = resn(1:ni+1);

```

The functions that are used in the previous code will be defined later in the code for ASGMRES; see Section 6.2.

If, as in [44], we would have chosen

$$p_{k+1} = r_{k+1} + \sum_{j=0}^k \beta_j^{(k)} p_j, \quad \beta_j^{(k)} = -\frac{(Ar_{k+1}, Ap_j)}{(Ap_j, Ap_j)}, \quad (6.2)$$

we would have obtained vectors p_k which are $A^T A$ orthogonal. Then, the matrix $[B^{(k)}]^T B^{(k)}$ is diagonal and, since the components of $[B^{(k)}]^T r_k$ are zero except for the first one, the coefficients $\alpha_j^{(k)}$ are zero except for $\alpha_k^{(k)}$ and we recover GCR. However, in [44] it is a truncated version of this algorithm which is proposed. Depending on how we choose the number of vectors kept in relations (6.1) and (6.2) we obtain different methods going from GCG-LS to GCR.

6.2 Simpler, residual-based simpler and adaptive simpler GMRES

Here we follow the lines of [577]. Let us assume that we have a basis of $\mathcal{K}_k(A, r_0)$ given by the columns of the matrix $V_{n,k}$. This basis is not necessarily orthogonal but the basis vectors are of unit norm. Let $W_{n,k}$ be an orthonormal matrix obtained by the QR factorization of $AV_{n,k}$,

$$AV_{n,k} = W_{n,k} T_k, \quad (6.3)$$

where T_k is upper triangular. The columns of $W_{n,k}$ give an orthonormal basis of $A\mathcal{K}_k(A, r_0)$. We would like to have residual vectors which are orthogonal to $A\mathcal{K}_k(A, r_0)$, that is, $r_k = (I - W_{n,k} W_{n,k}^*) r_0$. This is obtained by

$$r_j = r_{j-1} - \alpha_j w_j, \quad \alpha_j = (w_j, r_{j-1}), \quad j \leq k. \quad (6.4)$$

Let $R_{n,k+1} = (r_0 \cdots r_k)$, D_k be a diagonal matrix with the α_j 's as diagonal entries and $L_{k+1,k}$ be a lower bidiagonal matrix with 1's on the diagonal and -1 's on the first subdiagonal. Then, we have the matrix relation,

$$R_{n,k+1} L_{k+1,k} = W_{n,k} D_k.$$

As in GMRES we define the iterates as

$$x_k = x_0 + V_{n,k} y_k.$$

From relation (6.3) it yields

$$r_k = r_0 - W_{n,k} T_k y_k.$$

Multiplying by $W_{n,k}^*$ and using the orthogonality property of the residual vectors, the vector of coefficients y_k is obtained by solving a triangular linear system,

$$T_k y_k = W_{n,k}^* r_0 = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_k \end{pmatrix}.$$

The last equality is obtained because $(w_j, r_0) = (w_j, r_{j-1}) = \alpha_j$, see (6.4). This method is called the *generalized simpler approach* in [577].

We now have to specify how to choose the matrices $V_{n,k}$. The simpler GMRES method (which we abbreviate into SGMRES) proposed in [969] corresponds to choosing

$$V_{n,k} = \left(\frac{r_0}{\|r_0\|} \quad W_{n,k-1} \right). \quad (6.5)$$

If r_0 is not in $A\mathcal{K}_{k-1}(A, r_0)$, the columns of $V_{n,k}$ form a basis of $\mathcal{K}_k(A, r_0)$ and

$$\frac{\|r_0\|}{\|r_{k-1}\|} \leq \kappa(V_{n,k}) \leq 2 \frac{\|r_0\|}{\|r_{k-1}\|}. \quad (6.6)$$

This bound was first proved in [969] and then in [577]. However, we can exactly compute the condition number of $V_{n,k}$, see also [635] for a different proof.

Proposition 6.1 *Let $\alpha^{(k-1)} = (\alpha_1 \cdots \alpha_{k-1})^T$ in (6.4). Then the condition number of $V_{n,k}$ in (6.5) is*

$$\kappa(V_{n,k}) = \frac{\|r_0\| + \|\alpha^{(k-1)}\|}{\|r_{k-1}\|}. \quad (6.7)$$

Proof Let $\tilde{r}_0 = r_0/\|r_0\|$. Then $V_{n,k} = (\tilde{r}_0 \quad W_{n,k-1})$ and, since $W_{n,k}^* W_{n,k} = I$,

$$V_{n,k}^* V_{n,k} = \begin{pmatrix} 1 & \tilde{r}_0^* W_{n,k-1} \\ W_{n,k-1}^* \tilde{r}_0 & I \end{pmatrix}.$$

But $W_{n,k-1}^* r_0 = \alpha^{(k-1)}$. It yields

$$V_{n,k}^* V_{n,k} = I + \frac{1}{\|r_0\|} \begin{pmatrix} 0 & [\alpha^{(k-1)}]^* \\ \alpha^{(k-1)} & 0 \end{pmatrix} = I + \frac{1}{\|r_0\|} B_k.$$

Hence, $V_{n,k}^* V_{n,k}$ is equal to the identity plus a rank-2 matrix. It is not difficult to see that the two nonzero eigenvalues of B_k are $\pm \|\alpha^{(k-1)}\|$. The eigenvector corresponding to the eigenvalue $\pm \|\alpha^{(k-1)}\|$ is

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \pm \frac{\alpha^{(k-1)}}{\|\alpha^{(k-1)}\|} \end{pmatrix}.$$

Therefore, the singular values of $V_{n,k}$ are $k - 2$ times 1 and $\sqrt{1 \pm \frac{\|\alpha^{(k-1)}\|}{\|r_0\|}}$, and its condition number is

$$\kappa(V_{n,k}) = \left(\frac{\|r_0\| + \|\alpha^{(k-1)}\|}{\|r_0\| - \|\alpha^{(k-1)}\|} \right)^{\frac{1}{2}} = \frac{\|r_0\| + \|\alpha^{(k-1)}\|}{(\|r_0\|^2 - \|\alpha^{(k-1)}\|^2)^{\frac{1}{2}}}.$$

But we have $r_{k-1} = r_0 - W_{n,k-1} \alpha^{(k-1)}$ and, because of the orthogonality properties,

$$\|r_{k-1}\|^2 = \|r_0\|^2 - \|W_{n,k-1} \alpha^{(k-1)}\|^2 = \|r_0\|^2 - \|\alpha^{(k-1)}\|^2.$$

This proves relation (6.7). \square

Since $\|\alpha^{(k-1)}\| \leq \|r_0\|$, we obtain straightforwardly the bounds (6.6) from relation (6.7). The SGMRES basis is well conditioned if and only if the residual norms do not decrease too fast. Small residual norms lead to a badly conditioned basis and to large condition numbers for $AV_{n,k}$ and for T_k . This may lead to instability of SGMRES.

In fact, the matrix which has to be generated in SGMRES is $W_{n,k}$ which is obtained by the Arnoldi process with the modified Gram–Schmidt implementation starting from Ar_0 . If we do so, we obtain the Arnoldi relation

$$AW_{n,k-1} = W_{n,k} \underline{\mathcal{H}}_{k-1},$$

where $\underline{\mathcal{H}}_{k-1}$ is a $k \times (k - 1)$ upper Hessenberg matrix and

$$AV_{n,k} = \left(\frac{Ar_0}{\|r_0\|} \quad W_{n,k} \underline{\mathcal{H}}_{k-1} \right).$$

It yields

$$T_k = W_{n,k}^* A V_{n,k} = \begin{pmatrix} \frac{w_1^* A r_0}{\|r_0\|} & & \\ 0 & \underline{\mathcal{H}}_{k-1} & \\ \vdots & & \\ 0 & & \end{pmatrix}.$$

Hence, the columns 2 to k of T_k are those of $\underline{\mathcal{H}}_{k-1}$, and the $(1, 1)$ entry is

$$\frac{w_1^* A r_0}{\|r_0\|} = \frac{\|A r_0\|}{\|r_0\|}.$$

The interest of SGMRES compared to GMRES is that we do not have to compute the QR factorization of the upper Hessenberg matrix $\underline{\mathcal{H}}_k$ using Givens rotations. We just have to compute the upper triangular matrix T_k . For a comparison of GMRES and SGMRES, see also [216].

Another choice for $V_{n,k}$ was proposed in [577]. The method is called the residual-based SGMRES (RB-SGMRES for short) and

$$V_{n,k} = \begin{pmatrix} \frac{r_0}{\|r_0\|} & \cdots & \frac{r_{k-1}}{\|r_{k-1}\|} \end{pmatrix}. \quad (6.8)$$

If r_0 is not in $A\mathcal{K}_{k-1}(A, r_0)$ and the residual norms are strictly decreasing, the columns of $V_{n,k}$ are linearly independent. If stagnation occurs, the residual vectors are linearly dependent and the method breaks down. If the norms are strictly decreasing, it was proved in [577] that we have

$$1 \leq \kappa(V_{n,k}) \leq \sqrt{k} \phi_k, \quad \phi_k = \left(1 + \sum_{j=1}^{k-1} \frac{\|r_{j-1}\|^2 + \|r_j\|^2}{\|r_{j-1}\|^2 - \|r_j\|^2} \right)^{\frac{1}{2}}.$$

However, it turns out that we can explicitly compute the entries of $S_k = V_{n,k}^* V_{n,k}$.

Proposition 6.2 *The entries of S_k are given by*

$$[S_k]_{i,i} = 1, \quad [S_k]_{i,j} = \frac{\|r_i\|}{\|r_j\|}, \quad i > j, \quad i = 1, \dots, k. \quad (6.9)$$

Proof Let $\tilde{r}_i = r_i/\|r_i\|$. By definition the entry (i, j) of S_k is $(\tilde{r}_i, \tilde{r}_j)$. It is obvious that $[S_k]_{i,i} = 1$, $i = 1, \dots, k$. For simplicity, let us consider (r_i, r_j) . From the definition of the residual vectors we have

$$\begin{aligned} (r_i, r_j) &= (r_0 - W_{n,i} \alpha^{(i)}, r_0 - W_{n,j} \alpha^{(j)}), \\ &= \|r_0\|^2 + (\alpha^{(i)}, W_{n,i}^* W_{n,j} \alpha^{(j)}) - (r_0, W_{n,j} \alpha^{(j)}) - (W_{n,i} \alpha^{(i)}, r_0), \\ &= \|r_0\|^2 + (\alpha^{(i)}, W_{n,i}^* W_{n,j} \alpha^{(j)}) - \|\alpha^{(j)}\|^2 - \|\alpha^{(i)}\|^2. \end{aligned}$$

If $i > j$,

$$W_{n,i}^* W_{n,j} \alpha^{(j)} = \begin{pmatrix} \alpha^{(j)} \\ 0 \end{pmatrix}.$$

Hence $(\alpha^{(i)}, W_{n,i}^* W_{n,j} \alpha^{(j)}) = \|\alpha^{(j)}\|^2$ and

$$(r_i, r_j) = \|r_0\|^2 - \|\alpha^{(i)}\|^2.$$

We note that $\|r_i\|^2 = \|r_0\|^2 - \|\alpha^{(i)}\|^2$ and, dividing by $\|r_i\| \|r_j\|$ we obtain

$$(\tilde{r}_i, \tilde{r}_j) = \frac{\|r_0\|^2 - \|\alpha^{(i)}\|^2}{\|r_i\| \|r_j\|} = \frac{\|r_i\|^2}{\|r_i\| \|r_j\|} = \frac{\|r_i\|}{\|r_j\|},$$

which proves the result. \square

Using (6.9), we obtain a matrix which has a very particular structure,

$$V_{n,k}^* V_{n,k} = S_k = \begin{pmatrix} 1 & \frac{\|r_1\|}{\|r_0\|} & \frac{\|r_2\|}{\|r_0\|} & \dots & \frac{\|r_{k-1}\|}{\|r_0\|} \\ \frac{\|r_1\|}{\|r_0\|} & 1 & \frac{\|r_2\|}{\|r_1\|} & \dots & \frac{\|r_{k-1}\|}{\|r_1\|} \\ \frac{\|r_2\|}{\|r_0\|} & \frac{\|r_2\|}{\|r_1\|} & 1 & \dots & \frac{\|r_{k-1}\|}{\|r_2\|} \\ \vdots & \vdots & \ddots & & \vdots \\ \frac{\|r_{k-1}\|}{\|r_0\|} & \frac{\|r_{k-1}\|}{\|r_1\|} & \dots & & 1 \end{pmatrix}. \quad (6.10)$$

Using the results of Section 2.8 we know that the inverse of $V_{n,k}^* V_{n,k}$ is tridiagonal. The diagonal entries α_i , as well as the subdiagonal entries β_i can be obtained using Proposition 2.1.

Proposition 6.3 *The nonzero coefficients of the inverse of the matrix $V_{n,k}^* V_{n,k}$ in (6.10) are given by*

$$\begin{aligned} \alpha_1 &= \frac{\|r_0\|^2}{\|r_0\|^2 - \|r_1\|^2}, \quad \beta_1 = -\frac{\|r_0\| \|r_1\|}{\|r_0\|^2 - \|r_1\|^2}, \\ \alpha_i &= \|r_{i-1}\|^2 \left(\frac{1}{\|r_{i-2}\|^2 - \|r_{i-1}\|^2} + \frac{1}{\|r_{i-1}\|^2 - \|r_i\|^2} \right), \quad i = 2, \dots, k-1, \\ \alpha_k &= \frac{\|r_{k-2}\|^2}{\|r_{k-2}\|^2 - \|r_{k-1}\|^2}, \\ \beta_i &= -\frac{\|r_{i-1}\| \|r_i\|}{\|r_{i-1}\|^2 - \|r_i\|^2}, \quad i = 2, \dots, k-1. \end{aligned}$$

Proof We use Proposition 2.1 in Section 2.8. Taking the two sequences u_i and v_i , $i = 1, \dots, k$ defined there (not to be confused with the notation for the basis vectors v_j) as

$$u_i = \frac{1}{\|r_{i-1}\|}, \quad v_i = \|r_{i-1}\|, \quad i = 1, \dots, k,$$

we obtain the result. \square

Proposition 6.3 can be used to bound the smallest singular value of the matrix $V_{n,k}$ defined in (6.8) as well as the condition number.

Theorem 6.2 *Let*

$$\begin{aligned} \gamma_1 &= \frac{\|r_0\|}{\|r_0\| - \|r_1\|}, \\ \gamma_i &= \|r_{i-1}\| \left(\frac{1}{\|r_{i-2}\| - \|r_{i-1}\|} + \frac{1}{\|r_{i-1}\| - \|r_i\|} \right), \quad i = 2, \dots, k-1, \\ \gamma_k &= \frac{\|r_{k-2}\|}{\|r_{k-2}\| - \|r_{k-1}\|}. \end{aligned}$$

Then, for the matrix of RB-SGMRES we have

$$\sigma_{\min}(V_{n,k}) \geq \frac{1}{\max_{i=1,\dots,k} \sqrt{\gamma_i}},$$

and the condition number of $V_{n,k}$ is bounded by

$$\kappa(V_{n,k}) \leq \sqrt{k} \max_{i=1,\dots,k} \sqrt{\gamma_i}.$$

Proof We use Proposition 6.3 and the Gershgorin intervals to bound from above the largest eigenvalue of the inverse of $V_{n,k}^* V_{n,k}$. Taking the square root, this gives a lower bound for the smallest singular value of $V_{n,k}$. Since the columns of $V_{n,k}$ are of unit norm, we have $\|V_{n,k}\| \leq \|V_{n,k}\|_F \sqrt{k} = \sqrt{k}$ which yields the upper bound for the condition number. \square

We can sometimes improve the upper bound \sqrt{k} for $\|V_{n,k}\|$ by considering the lower bound obtained from the Gershgorin intervals. We define

$$\begin{aligned} \delta_1 &= \frac{\|r_0\|}{\|r_0\| + \|r_1\|}, \\ \delta_i &= \|r_{i-1}\| \left(\frac{1}{\|r_{i-1}\| + \|r_i\|} - \frac{1}{\|r_{i-2}\| + \|r_{i-1}\|} \right), \quad i = 2, \dots, k-1, \\ \delta_k &= \frac{\|r_{k-2}\|}{\|r_{k-2}\| + \|r_{k-1}\|}. \end{aligned}$$

The minimum of the δ_j 's gives a lower bound for the smallest eigenvalue of $V_{n,k}^* V_{n,k}$. However, some δ_j 's may be negative, if this is the case we replace them by $1/j$. Moreover, to obtain the best bound, we redefine $\delta_i = \max(\delta_i, 1/i)$, $i = 2, \dots, k-1$. Then, the upper bound on the condition number is

$$\kappa(V_{n,k}) \leq \frac{\max_{i=1,\dots,k} \sqrt{\gamma_i}}{\min_{i=1,\dots,k} \sqrt{\delta_i}}.$$

The upper bounds on the condition number of $V_{n,k}$ are only valid if the residual norms are strictly decreasing. We observe that the condition number is nicely bounded if the residual norms decrease fast. If some consecutive norms are close, the upper bound can be quite large.

With this basis, we have $v_k = r_{k-1}/\|r_{k-1}\|$. Orthogonalizing Av_k against w_1, \dots, w_{k-1} using MGS we obtain a vector \tilde{w} and $w_k = \tilde{w}/\|\tilde{w}\|$. The entries of the k th column of T_k are the coefficients of the orthogonalization and $(T_k)_{k,k} = \|\tilde{w}\|$. Note that $(T_k)_{1,1} = \|Ar_0\|/\|r_0\|$.

By definition the residual vector r_k is orthogonal to $A\mathcal{K}_k(A, r_0)$. Note that RB-SGMRES is a Q-OR method since the basis vectors of $\mathcal{K}(A, r_0)$ are proportional to the residual vectors. However, this method minimizes the residual norms, and the matrix $V_{n,k}^*AV_{n,k}$ is upper triangular. This is very similar to what we have with the Q-OR optimal basis of Section 4.5, even though the implementation is quite different.

Another way to compute the iterates was considered in [577]. Let $p_k = A^{-1}w_k$ and $P_{n,k} = (p_1 \cdots p_k)$. Then,

$$x_k = x_{k-1} + \alpha_k p_k, \quad (6.11)$$

and $X_{n,k+1}L_{k+1,k} = -P_{n,k}D_k$ with $X_{n,k+1} = (x_0 \cdots x_k)$. If we know the vectors p_j we can compute the iterates x_k with a simple update instead of computing y_k . This is called the *generalized update approach* in [577]. Since we have $AP_{n,k} = W_{n,k}$, we obtain

$$V_{n,k} = P_{n,k}T_k.$$

It yields a (long) recurrence to compute p_k from v_k and p_1, \dots, p_{k-1} .

With the choice of basis (6.5) this method was introduced in [776] under the name $A^T A$ variant of GMRES. It is equivalent to Orthodir. If we choose (6.8) as the basis, we obtain a scaled version of GCR.

In order to cure the problems we may have with SGMRES and RB-SGMRES, an adaptive version of this method was proposed in [576]. Whenever the residual norm nearly stagnates we use the vector w_{k-1} at iteration k . Otherwise, when we observe sufficient residual norm decrease, we set the new direction vector equal to the normalized residual vector $r_{k-1}/\|r_{k-1}\|$. This adaptive choice keeps the basis well conditioned. Let $v_1 = r_0/\|r_0\|$ and

$$v_k = \begin{cases} \frac{r_{k-1}}{\|r_{k-1}\|}, & \text{if } \|r_{k-1}\| \leq \nu \|r_{k-2}\| \\ w_{k-1}, & \text{otherwise} \end{cases}$$

where ν is a given number in $[0, 1)$ (note that this is a “nu” and not a “v”). If $\nu = 0$ we obtain SGMRES and if $\nu = 1$ this is RB-SGMRES. When the basis has been

chosen, we must update the QR factorization of $AV_{n,k}$ to obtain the matrix T_k . This algorithm, which we denote by ASGMRES, does not break down unless the exact solution has been found. Bounds for the condition number of $V_{n,k}$ depending on ν are given in [576]. In many problems choosing a value of ν equal to 0.8 or 0.9 works well.

A code for ASGMRES using Householder reflections is following, in which the input value nu corresponds to ν .

```

function [x,ni,resn] = ASGMRES(A,b,x0,epsi,nitmax,nu);
%
n = size(A,1);
V = zeros(n,nitmax+1);
Q = zeros(n,nitmax);
betaq = zeros(nitmax,1);
T = zeros(nitmax,nitmax);
rhs = zeros(nitmax,1);
%
nb = norm(b);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
nrq = norm(r);
resn(1) = nrq;
V(:,1) = r / nrq;
Av = A * V(:,1);
[q,betaq] = household(Av,2);
Q(:,1) = q;
betaq(1) = betaq;
qa = q' * Av;
T(1,1) = Av(1) - betaq * qa * q(1); % apply the reflection
qk = -betaq * q(1) * q;
qk(1) = 1 + qk(1);
rhs(1) = r' * qk;
r = r - rhs(1) * qk;
nrold = nrq;
nrq = norm(r);
resn(2) = nrq;
% adaptive choice for the new basis vector
if nrq <= nu * nrold
    V(:,2) = r / nrq;
else
    V(:,2) = qk;
end % if nrq
resn(2) = nrq;
ni = 1;

```

```

%
for k = 2:nitmax
    ni = ni + 1; % number of iterations
    Av = A * V(:,k); % matrix vector product
    % apply the preceding reflections to the column of A V
    t = apply_house(Q,betq,1,k-1,Av);
    % add the new column, add to Q and betq
    [Q,betq] = qr_add_col(Q,k-1,betq,t);
    T(1:k-1,k) = t(1:k-1);
    % apply the last reflection for the diagonal entry
    qa = Q(:,k)' * t;
    T(k,k) = t(k) - betq(k) * qa * Q(k,k);
    % we need the k-th column of Q
    ek = zeros(n,1); ek(k) = 1;
    % apply the reflections in reverse order
    qk = apply_house_rev(Q,betq,1,k,ek);
    rhs(k) = r' * qk;
    r = r - rhs(k) * qk;
    nrold = nrq;
    nrq = norm(r);
    % adaptive choice for the new basis vector
    if nrq <= nu * nrold
        V(:,k+1) = r / nrq;
    else
        V(:,k+1) = qk;
    end % if nrq
    resn(ni+1) = nrq;
    if nrq < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nrq
end % for k
%
t = T(1:k,1:k)  rhs(1:k);
x = x0 + V(:,1:k) * t;
resn = resn(1:ni+1);
end % function

```

The functions called in the previous code are listed below.

```

function [v,beta,P] = household(x,m);
%
nx = length(x);
x = x(m-1:nx);
n = length(x);
sig = x(2:n)' * x(2:n);
v = [1; x(2:n)];

```

```

if sig == 0
  beta = 0;
else
  mu = sqrt(x(1) * x(1) + sig);
  if x(1) <= 0
    v(1) = x(1) - mu;
  else
    v(1) = -sig/(x(1) + mu);
  end % if x
  beta = 2 * v(1) * v(1) / (sig + v(1) * v(1));
  v = v / v(1);
end % if sig
v = [zeros(m-2,1); v];
if nargout == 3
  P = speye(nx,nx) - beta * (v * v');
end % if nargout
end % function
%
function y = apply_house(V,bet,ns,ne,x);
%
nv = size(V,2);
for k = ns:ne
  vk = V(:,k);
  x = x - bet(k) * (vk' * x) * vk;
end % for k
y = x;
end % function
%
function [V,bet] = qr_add_col(V,nv,bet,u);
%
[n,m] = size(V);
bet = bet(:);
if nv+2 > n
  return
end
% we must zero the components of u from nv+2 to n
[v,beta] = household(u,nv+2);
V(:,nv+1) = v;
bet(nv+1) = beta;
end % function
%
function y = apply_house_rev(V,bet,ns,ne,x);
%
nv = size(V,2);
for k = ne:-1:ns

```

```

vk = V(:,k);
x = x - bet(k) * (vk' * x) * vk;
end % for k
y = x;
end % function

```

6.3 Orthores

Orthores was proposed in [1014] (see also [559, 560]) as a generalization of the three-term variant of the Conjugate Gradient (CG) algorithm. As for Orthodir, the method was defined using an auxiliary matrix Z which defined the dot product. We describe a simplified version using $Z = I$ that was considered in [308]. The formulas for computing the residual vectors and the approximations of the solutions are

$$\begin{aligned}\alpha_j^{(k)} &= \frac{(Ar_k, r_j)}{(r_j, r_j)}, \quad j = 0, \dots, k, \\ \beta_k &= \frac{1}{\sum_{j=0}^k \alpha_j^{(k)}}, \quad \gamma_j^{(k)} = \beta_k \alpha_j^{(k)}, \\ x_{k+1} &= \beta_k r_k + \sum_{j=0}^k \gamma_j^{(k)} x_j, \\ r_{k+1} &= -\beta_k Ar_k + \sum_{j=0}^k \gamma_j^{(k)} r_j.\end{aligned}$$

This algorithm yields orthogonal residual vectors which generate a basis of $\mathcal{K}(A, r_0)$, such that $(r_i, r_j) = 0$ for $i \neq j$. Therefore, r_k is orthogonal to $\mathcal{K}_k(A, r_0)$. This version of Orthores is mathematically equivalent to FOM. It can break down if $\sum_{j=0}^k \alpha_j^{(k)} = 0$.

```

function [x,ni,resn] = Orthores(A,b,x0,epsi,nitmax);
%
n = size(A,1);
nb = norm(b);
x = x0;
r = b - A * x;
R = zeros(n,nitmax+1);
X = zeros(n,nitmax+1);
resn = zeros(1,nitmax+1);
rr = zeros(nitmax+1,1);
R(:,1) = r;
X(:,1) = x;
rr(1) = r' * r;
resn(1) = norm(r);

```

```

ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    Ar = A * r; % matrix vector product
    alpha = (Ar' * R(:,1:k))' ./ rr(1:k);
    beta = 1 / sum(alpha(1:k));
    gamma = beta * alpha;
    x = beta * r + X(:,1:k) * gamma;
    r = -beta * Ar + R(:,1:k) * gamma;
    R(:,k+1) = r;
    X(:,k+1) = x;
    rr(k+1) = r' * r;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    if nresidu < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nresidu
end % for k
resn = resn(1:ni+1);

```

6.4 Q-OR Optinv

In this section we use the Q-OR optimal basis constructed in Section 4.5. We showed how to compute the basis vectors v_j and the entries of the upper Hessenberg matrices H_k . We define a Q-OR method with this basis; see [685]. Therefore, the iterates are defined as

$$x_k = x_0 + V_{n,k}y^{(k)}, \quad H_k y^{(k)} = \|r_0\|e_1.$$

The linear system with H_k may be solved by any method but we choose to transform H_k to upper triangular form by Givens rotations as in FOM. The residual vectors r_k are proportional to the basis vectors which are of unit norm and we have $\|r_k\| = h_{k+1,k} |[y^{(k)}]_k|$. The basis vectors were constructed to maximize the entries on the first row of the inverses of the matrices U_k in the triangular Hessenberg decomposition $H_k = U_k C^{(k)} U_k^{-1}$ of Theorem 2.3 in Chapter 2. Doing so we minimize the residual norms in the Krylov subspaces and we also have $\|r_k\| = \|r_0\| / |\vartheta_{1,k+1}|$ where the $\vartheta_{1,j}$'s are the entries on the first row of U_k^{-1} . This means that, when there is no breakdown, this method delivers the same residual norms as GMRES. The minimization property can also be proved using Lemma 4.5. This lemma shows that the matrices $V_{n,k}^* A V_{n,k}$ are upper triangular. Since the residual vectors are proportional to the basis vectors, it shows that r_k is orthogonal to $A\mathcal{K}_k(A, r_0)$ proving that the residual norms are minimized. Theorem 4.4 gives a lower bound on the smallest singular value of

$V_{n,k}$. If the residual norms decrease fast enough, the basis is well behaved and, since $\|V_{n,k}\| \leq \sqrt{k}$, the condition number of $V_{n,k}$ is not large. However, if some consecutive residual norms are close, the lower bound on the smallest singular value becomes small and the condition number may grow.

We remark that this Q-OR optimal method uses the same basis as RB-SGMRES described in Section 6.2 but in a different way. A code implementing this method is as follows. As in Chapter 4, we use the inverses of the Cholesky factors of the matrices $V_{n,k}^*$ $V_{n,k}$.

```

function [xi,ni,resn] = QORoptinv(A,b,x0,epsi,nitmax);
%
% Initialization phase
n = size(A,1);
V = zeros(n,nitmax+1);
H = zeros(nitmax+1,nitmax);
Lt = zeros(nitmax,nitmax);
nu = zeros(nitmax+1,1);
rhs = zeros(nitmax+1,1);
rot = zeros(2, nitmax); % init Givens rotations
resn = zeros(1,nitmax);
x = x0;
r = b - A * x;
ni = 0;
nb = norm(b);
bet = norm(r);
resn(1) = bet;
rhs(1) = bet;
% End of initialization
%
for k = 1:nitmax
    ni = ni + 1;
    if k == 1
        V(:,1) = r / bet;
        Av = A * V(:,1); % matrix vector product
        Lt(1,1) = 1;
        omega = V(:,1)' * Av;
        alpha = Av' * Av - omega^2;
        H(1,1) = omega + alpha / omega;
        vt = Av - H(1,1) * V(:,1);
        nw = norm(vt);
        H(2,1) = nw;
        V(:,2) = vt / H(2,1);
        nu(1) = 1;
        nu(2) = -H(1,1) / H(2,1);
    else % if k

```

```

Av = A * V(:,k); % matrix vector product
vV = V(:,1:k-1)' * V(:,k);
vtA = V(:,1:k)' * Av;
lt = Lt(1:k-1,1:k-1) * vV;
yt = lt' * Lt(1:k-1,1:k-1);
if lt' * lt < 1
    lkk = sqrt(1 - lt' * lt);
else % if lt
    % error
end % if lt
Lt(k,1:k) = [-yt / lkk, 1 / lkk];
lA = Lt(1:k,1:k) * vtA;
s = Lt(1:k,1:k)' * lA;
alpha = Av' * Av - lA' * lA;
beta = alpha / vtA(k);
H(1:k-1,k) = s(1:k-1);
H(k,k) = s(k) + beta;
if k <= n
    vt = Av - V(:,1:k) * H(1:k,k);
    nw = norm(vt);
    H(k+1,k) = nw;
    nu(k+1) = -(nu(1:k)' * H(1:k,k)) / H(k+1,k);
    V(:,k+1) = vt / H(k+1,k); % next basis vector
end % if k
end % if k == 1
nw1 = nw;
% apply the preceding Givens rotations to the last column
for kk = 1:k-1
    h1 = H(kk,k);
    h2 = H(kk+1,k);
    H(kk+1,k) = -rot(2,kk) * h1 + rot(1,kk) * h2;
    H(kk,k) = rot(1,kk) * h1 + conj(rot(2,kk)) * h2;
end % for kk
% compute, store and apply a new rotation to zero
% the last term in kth column
nw = H(k,k);
cs = sqrt(abs(nw1)^2 + abs(nw)^2);
if abs(nw) < abs(nw1)
    mu = nw / nw1;
    tau = conj(mu) / abs(mu);
else
    mu = nw1 / nw;
    tau = mu / abs(mu);
end % if abs
% store the rotation for the next columns

```

```

rot(1,k) = abs(nw) / cs; % cosine
rot(2,k) = abs(nw1) * tau / cs; % sine
% modify the diagonal entry and the right-hand side
% estimate of the residual norm
nresidu = H(k+1,k) * abs(rhs(k) / H(k,k));
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
if k < nitmax
    H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
    c = rhs(k);
    rhs(k) = rot(1,k) * c;
    rhs(k+1) = -rot(2,k) * c;
end % if k
end % for k
%
% computation of the solution
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + V(:,1:k) * y;
resn = resn(1:ni+1);

```

6.5 Parallel computing

There are not many papers related to the use of the methods described in this chapter on parallel computers, probably because many researchers were more interested in parallelizing GMRES. We observe that, for instance, there are data dependencies in GCR. This reminds of what we have for the conjugate gradient or the conjugate residual algorithms for symmetric systems.

Nevertheless, parallel s -step versions of GCR were proposed in [222]. First, a general s -step minimal residual method was derived where the iterates are defined as $x_{k+1} = x_k + q_{s-1}(A)r_k$, q_{s-1} being a polynomial of degree $s - 1$. The coefficients of the polynomial are chosen to minimize the residual norm. Then, s -step generalizations of GCR were described. The iterates are

$$x_{k+1} = x_k + \sum_{j=1}^s \alpha_j^{(k)} p_j^{(k)},$$

where the coefficients $\alpha_j^{(k)}$ are computed to minimize the residual norm. Let

$$p_i^{(k+1)} = A^{i-1}r_k + \sum_{j=1}^k \sum_{\ell=1}^s \beta_j^{(i,\ell)} p_\ell^{(j)}, \quad i = 1, \dots, s.$$

The coefficients $\beta_j^{(i,\ell)}$ are computed to $A^T A$ -orthogonalize the s new directions against *all* the previous ones.

We immediately see that we can have stability problems with this method since the monomial basis is used by computing $A^{i-1}r_k$ for $i = 1, \dots, s$. This will severely limit the values of s that can be used safely. In [228, 900] the modified Gram–Schmidt algorithm was used to orthogonalize the directions within one block. It allowed to use larger values of s . This can probably be improved by using a Newton or Chebyshev basis.

Pipelined parallel versions of CGR were more recently considered in [811].

6.6 Finite precision arithmetic

There are not many results in the literature about the stability of the methods we have described in this chapter. The two recurrences

$$x_k = x_{k-1} + \alpha_{k-1} p_{k-1}, \quad r_k = r_{k-1} - \alpha_{k-1} A p_{k-1},$$

in finite precision arithmetic were studied in [454]. However, we observe that this does not completely cover the case of methods like GCR, Orthomin or Orthodir since, in these methods, in the computation of r_k , the term $A p_{k-1}$ is not computed as A times p_{k-1} but comes from another recurrence. In [454] what is studied is the difference between the updated residual r_k and what is usually called the true residual $b - Ax_k$. The result is

$$\frac{\|(b - Ax_k) - r_k\|}{\|A\| \|x\|} \leq (\varepsilon + O(\varepsilon^2)) \left[k + 1 + (1 + c + k(10 + 2c)) \max_{j \leq k} \frac{\|x_j\|}{\|x\|} \right],$$

where c is a constant such that $\|f(Ap_{k-1}) - Ap_{k-1}\| \leq c\varepsilon\|A\|\|p_{k-1}\|$ and ε is the machine epsilon. Of course, the constants are not important since the bound is not sharp, the interesting term is the ratio of the norms of the iterates and the exact solution. If there are iterations for which this ratio is large, the relative difference between the true and the updated residual vectors can be large.

The numerical stability of simpler GMRES was considered in [635] where it is stated that decreasing residual norms can lead to the ill-conditioning of the upper triangular matrix T_k independently of the conditioning of A . This may negatively affect the numerical accuracy of the approximate solution. For results on SGMRES in finite precision arithmetic, see also [216].

Simpler GMRES was further studied in [577]. The authors assumed that rounding errors occur only in selected, most relevant parts of the computation. They gave

bounds on the maximum attainable accuracy measured by the (restricted) normwise backward error. While for the generalized simpler approach the bounds do not depend on the condition number of A , the bound for the generalized update approach is proportional to $\kappa(A)$, see [577]. However, in practice, the generalized simpler approach and the generalized update approach behave almost equally well when choosing the same basis. In [577] it is proved that the simpler GMRES choice $V_{n,k} = (r_0/\|r_0\|, W_{n,k-1})$ leads to inherently unstable or numerically less stable schemes. The choice of basis with the normalized residuals gives rise to a conditionally stable algorithm provided that the residual norms decrease fast enough. In particular, it is shown that the RB-SGMRES implementation is conditionally backward stable.

In finite precision arithmetic, instead of relation (6.3), we have

$$AV_{n,k} = W_{n,k}T_k + F_k, \quad \|F_k\| \leq cu\|A\|\|V_{n,k}\|,$$

where F_k is the perturbation term arising from rounding errors and u is the roundoff unit.

In the inequalities below we assume that the condition numbers are such that the denominators are positive. For the generalized simpler GMRES the difference between the true residual and the updated residual is bounded as

$$\frac{\|b - Ax_k - r_k\|}{\|A\|\|x_k\|} \leq c u \kappa(V_{n,k}) \left(1 + \frac{\|x_0\|}{\|x_k\|}\right).$$

A bound involving the singular values $\sigma_j(V_{n,j})$ of the matrices $V_{n,j}$, $j = 1, \dots, k$ is

$$\|b - Ax_k - r_k\| \leq \frac{c u \kappa(A)}{1 - c u \kappa(A) \kappa(V_{n,k})} \sum_{j=1}^k \frac{\|r_{j-1}\|}{\sigma_j(V_{n,j})}.$$

For the generalized update approach the corresponding bounds are

$$\frac{\|b - Ax_k - r_k\|}{\|A\|\|x_k\|} \leq \frac{c u \kappa(A) \kappa(V_{n,k})}{1 - c u \kappa(A) \kappa(V_{n,k})} \left(1 + \frac{\|x_0\|}{\|x_k\|}\right),$$

$$\|b - Ax_k - r_k\| \leq \frac{c u [\kappa(A)]^2}{1 - c u \kappa(A) \kappa(V_{n,k})} \sum_{j=1}^k \frac{\|r_{j-1}\|}{\sigma_j(V_{n,j})}.$$

We observe that we have an additional multiplying factor $\kappa(A)$ in the numerators. The bounds on the ultimate relative residual norms (which, more or less, describe the maximum attainable accuracy) show that the relative residuals in the generalized simpler approach will reach a level approximately equal to $u\kappa(A)$, while in the generalized update approach this level is $u\kappa^2(A)$. The potentially ill-conditioned bases in SGMRES and Orthodir mean that these methods can behave unstably.

However, it was shown in [577] that the forward error of the generalized update approach is essentially on the same level as for the generalized simpler approach. For both approaches we have

$$\frac{\|(x - x_k) - A^{-1}r_k\|}{\|x\|} \leq \frac{c\kappa(A)\kappa(V_{n,k})}{1 - c\kappa(A)\kappa(V_{n,k})} \frac{\|x_k\| + \|x_0\|}{\|x\|}.$$

Bounds for the adaptive algorithm ASGMRES were given in [576].

In [1018] the author studied perturbed Arnoldi-like relations

$$AV_{n,k} = V_{n,k}H_k + h_{k+1,k}v_{k+1}e_k^T - F_k, \quad (6.12)$$

where F_k is the perturbation matrix with columns f_j . For a rounding error analysis, it is likely that the entries of the matrix F_k will be “small”. In Theorem 4.37 of [1018] (see also Theorem 2.1 in [1020]) it is proved that the computed basis vectors satisfy

$$v_{k+1} = \frac{p_k^C(A)}{\prod_{j=1}^k h_{j+1,j}} v_1 + \sum_{i=1}^k \frac{p_{i+1:k}^C(A)}{\prod_{j=i}^k h_{j+1,j}} f_i,$$

where $p_k^C(A)$ is the characteristic polynomial of H_k evaluated at the matrix A and with a similar notation for $H_{i+1:k}$ which is a trailing principal submatrix of H_k with $p_{k+1:k}^C(A) \equiv 1$ by extension.

In [1020] the projections of the basis vectors on the left eigenvectors of A were studied. Let q be a left eigenvector corresponding to the eigenvalue λ of A and s be a right eigenvector of H_k corresponding to the eigenvalue θ . Then,

$$q^* v_{k+1} = \frac{(\lambda - \theta)q^* V_{n,k}s}{h_{k+1,k}e_k^T s} + \frac{q^* F_k s}{h_{k+1,k}e_k^T s}.$$

We observe that if θ is close to λ and $h_{k+1,k}$ is not too small, the projection is dominated by the perturbation term. For our Q-OR Optinv method we can use results in Theorem 4.4 of [1020].

Theorem 6.3 *The residual vectors obtained from the Q-OR method based on the perturbed Arnoldi-like relation (6.12) are given by*

$$r_k = p_k^O(A)r_0 + \|r_0\| \sum_{j=1}^k \left(\prod_{i=1}^{j-1} h_{i+1,i} \right) \frac{p_{j+1:k}^C(A) - p_{j+1:k}^C(0)I}{p_k^C(0)} f_j, \quad (6.13)$$

where p_k^O is the residual polynomial, $p_{j+1:k}^C$ is the characteristic polynomial of the trailing principal submatrix of H_k starting at row $j+1$ and p_k^C is the characteristic polynomial of H_k .

If all the matrices $H_{j+1:k}$ are nonsingular, we also have

$$r_k = p_k^O(A)r_0 - \sum_{j=1}^k y_j^{(O)} p_{j+1:k}^O(A)f_j + F_k y^{(O)},$$

where $H_k y^{(O)} = \|r_0\|e_1$ and $p_{j+1:k}^O(\lambda) = p_{j+1:k}^C(\lambda)/p_{j+1:k}^C(0)$.

The perturbation term in (6.13) depends on the roots of the polynomials $p_{j+1:k}^C$. In the other expression it depends on the decay of the polynomials $p_{j+1:k}^O(A)$. If A is diagonalizable, it depends on the values of these polynomials at the eigenvalues of A .

6.7 Numerical examples

In this section, we illustrate the properties of the methods described in this chapter and compare them to the modified Gram–Schmidt GMRES implementation.

6.7.1 GCR, Orthodir and Axelsson

Let us compare GMRES with some methods that are mathematically equivalent even though their implementations are completely different, that is, GCR, Orthodir and Axelsson's method GCG-LS.

Figure 6.1 shows the \log_{10} of the relative true residual norms for `fs 183 6`. Even though the residual norms are almost the same at the beginning, the maximum attainable accuracy is much larger for GCR than for GMRES and Axelsson's method. Orthodir has a breakdown at iteration 31, and it is stagnating before that. Hence, GCR and Orthodir cannot be used reliably for this problem.

The results are a little better for `fs 680 1c` even though the maximum attainable accuracy of Orthodir is much worse than for the other methods. We observe that for this problem the final accuracy of Axelsson's method is better than that of GMRES; see Figure 6.2.

The results are more satisfactory for `supg 001` of order 1225. All the methods give approximately the same results as we see in Figure 6.3. The GCR curve is hidden behind Axelsson's method curve. We observe that as for the two previous examples, in our Matlab implementation, GCR requires less time than GMRES.

Table 6.1 displays the minimum true residual norms for our first set of test matrices (ordered by increasing condition number) in k_{\max} iterations. In most of the examples Orthodir breaks down or gives very bad results. In many cases GCR gives residual norms which are approximately the same as those of GMRES but there are a few cases for which the accuracy is much worse; see `sherman5`, `e05r0500`, `comsol`, `olm1000`, `steam1`. Except for very few cases Axelsson's method gives residual norms comparable to those of GMRES.

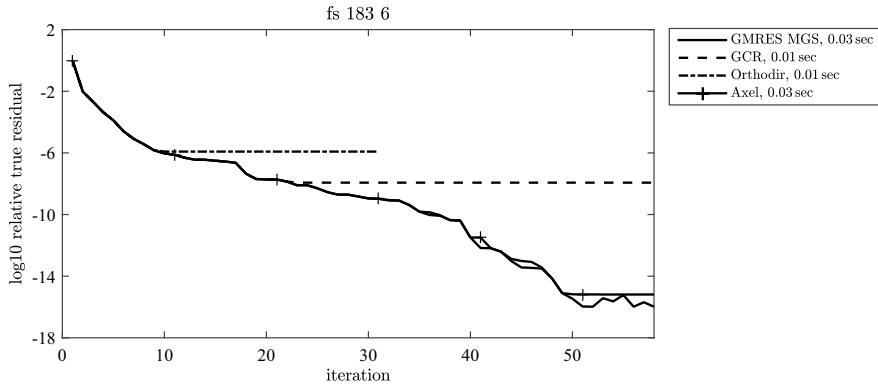


Fig. 6.1 fs 183 6, true residual norms, GMRES-MGS (plain), GCR (dashed), Orthodir (dot-dashed), Axelsson (+)

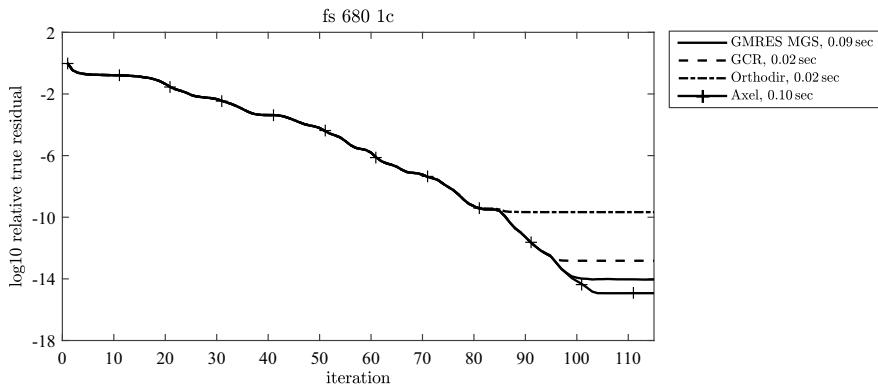


Fig. 6.2 fs 680 1c, true residual norms, GMRES-MGS (plain), GCR (dashed), Orthodir (dot-dashed), Axelsson (+)

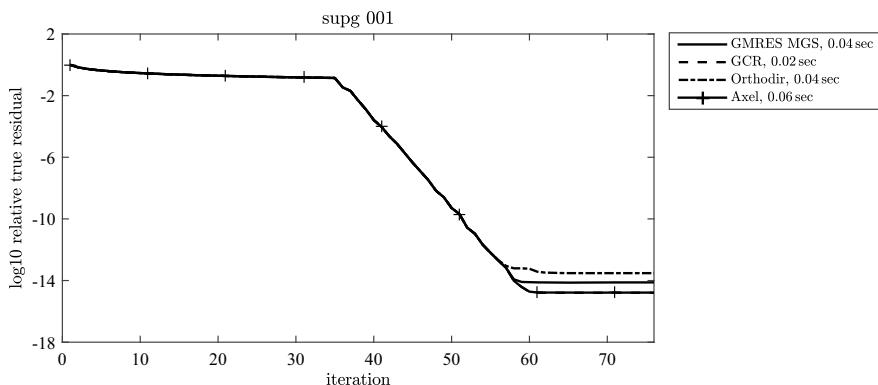


Fig. 6.3 supg 001, true residual norms, GMRES-MGS (plain), GCR (dashed), Orthodir (dot-dashed), Axelsson (+)

Table 6.1 Minimum relative true residual norms in k_{\max} iterations

matrix	k_{\max}	GMRES-MGS	GCR	Orthodir	Axelsson
pde225	150	1.66541e-15	1.46430e-15	2.91531e-14	1.55214e-15
gre 343	300	6.61396e-15	4.80781e-15	5.73005e-14	4.63215e-16
jphw 991	200	1.07847e-14	2.29525e-14	2.07772e-13	7.12904e-15
pde2961	320	1.65763e-14	9.34220e-15	1.10308e-11	9.47479e-15
jagmesh1	300	4.83286e-16	3.62348e-15	5.25246e-14	4.98721e-16
bfgwa782	350	1.39710e-14	5.57470e-13	7.19295e-10	4.87486e-14
dw2048	1100	1.24776e-15	1.27272e-15	2.59970e-06	1.14108e-15
jagmesh2	800	5.73936e-16	2.14993e-15	1.09206e-13	1.68136e-15
raefsky2	450	7.17708e-15	3.61059e-15	1.76245e-10	3.67270e-15
fs 680 1c	150	8.85755e-15	1.49639e-13	2.14086e-10	1.16438e-15
add20	600	3.03778e-13	4.13183e-13	3.08924e-06	3.13947e-13
raefsky1	350	6.06386e-14	2.61742e-14	1.87171e-07	2.41625e-14
jagmesh4	700	4.62824e-15	2.95788e-15	1.22705e-06	4.93713e-16
fs 680 1	200	7.02671e-15	3.54483e-13	3.76794e-02	7.95586e-16
sherman1	500	4.72052e-14	4.99483e-14	2.95998e-11	5.16575e-14
nos3	320	9.26899e-14	1.73153e-14	7.02630e-03	1.61219e-14
sherman5	1200	1.21236e-11	7.78394e-01	7.86651e-01	1.56253e-10
cavity05	600	1.65859e-13	2.13994e-13	6.36396e-04	6.15184e-15
e05r0500	230	1.76589e-12	5.92331e-01	4.59803e-01	2.65648e-05
comsol	300	7.24522e-11	9.63437e-03	8.88626e-02	1.08799e-10
olm1000	600	3.26048e-14	7.67344e-09	7.82169e-03	2.71017e-12
cavity10	900	1.68665e-13	6.55202e-14	6.22458e-05	7.96919e-15
steam2	200	2.24697e-10	3.64412e-08	6.93157e-01	9.66955e-12
1138bus	700	5.68921e-12	5.88611e-12	3.36060e-01	6.49904e-12
steam1	250	3.71137e-10	2.40789e-03	5.51316e-01	3.47708e-13
bcsstk26	1000	4.13181e-08	4.13181e-08	4.91423e-02	4.13181e-08
nos7	500	9.11821e-08	9.81666e-08	9.02044e-01	1.05576e-07
watt1	350	7.17226e-09	3.91691e-10	3.35082e-02	3.91690e-10
bcsstk14	1000	1.38535e-01	1.38535e-01	6.67904e-01	1.38535e-01
fs 183 6	60	1.04100e-16	1.17318e-08	1.21830e-06	6.47953e-16
bcsstk20	500	1.81822e-06	7.14773e-08	8.28850e-01	1.67269e-03
mcf6	820	4.61980e-06	8.23908e-01	9.58337e-01	6.63593e-06
nnc	250	4.55189e-04	7.29368e-01	6.60684e-01	1.12281e-02
Insp	420	1.66223e-03	9.73845e-01	9.50983e-01	1.14402e-01

Table 6.2 Minimum relative true residual norms in k_{\max} iterations

matrix	k_{\max}	GMRES-MGS	GCR	Orthodir	Axelsson
add32	200	7.16632e-15	6.50945e-15	5.71436e-10	3.59043e-15
ex37	150	1.75693e-13	6.06755e-14	1.14658e-11	1.44076e-15
memplus	900	9.17631e-12	1.71153e-12	1.69653e-04	1.73125e-12
sherman3	800	1.93397e-10	1.18363e-11	6.91201e-01	1.14955e-11
wang4	700	7.40182e-15	1.51696e-07	4.27034e-03	5.21643e-15
supg 100 6400	1000	4.30334e-14	1.42449e-14	2.59426e-03	1.32544e-14
supg 1 6400	1100	3.67990e-14	1.18987e-14	1.44253e-11	1.12271e-14
supg 01 6400	500	2.97491e-14	6.99061e-15	1.96020e-03	4.62538e-15
supg 001 6400	150	1.02025e-14	6.44773e-15	3.05486e-03	1.93567e-15
supg 0001 6400	120	2.06592e-14	6.20977e-15	1.78910e-03	4.23897e-15
supg 00001 6400	220	5.95295e-14	1.96216e-14	1.08524e-01	2.64094e-14
supg 000001 6400	240	1.19743e-13	3.78478e-14	4.30639e-01	3.50620e-14
convdiff xu 500 8100	1100	6.30921e-14	5.36716e-02	5.36678e-02	5.32819e-02
convdiff xu 1000 8100	700	2.36299e-13	9.65421e-02	9.64615e-02	6.13947e-05
convdiff xu 5000 8100	400	3.55342e-13	4.61104e-01	4.60961e-01	3.40864e-12

For the set of larger matrices in Table 6.2 GCR gives sometimes better minimum residual norms than GMRES. For the last three examples, GCR, Orthodir and Axelsson's method are much worse than GMRES.

6.7.2 ASGMRES

Here we compare GMRES with simpler GMRES (SGMRES) [969], the residual-based GMRES (RB-GMRES) [577] and the adaptive version proposed in [576] with the parameter $\nu = 0.8$. Figure 6.4 shows that for the matrix `fs_183_6`, if SGMRES gives residual norms similar to those of GMRES at the beginning up to iteration 18, the residual norms blow up showing that this method is unstable. The residual norms of RB-GMRES are approximately the same as those of GMRES up to the end of the computation. The adaptive method works well up to the final stagnation phase but then the residual norms increase very rapidly showing that the parameter ν may not have been chosen optimally.

For the matrix `fs_680_1c` RB-GMRES and the adaptive method work as well as GMRES; see Figure 6.5. Even though this happens later than in the previous example, the residual norms of SGMRES blow up.

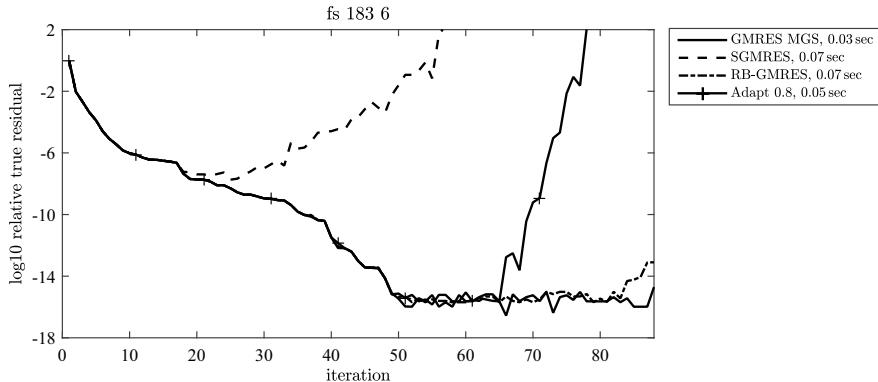


Fig. 6.4 *fs 183 6*, relative true residual norms, GMRES-MGS (plain), SGMRES (dashed), RB-GMRES (dot-dashed), ASGMRES $v = 0.8$ (+)

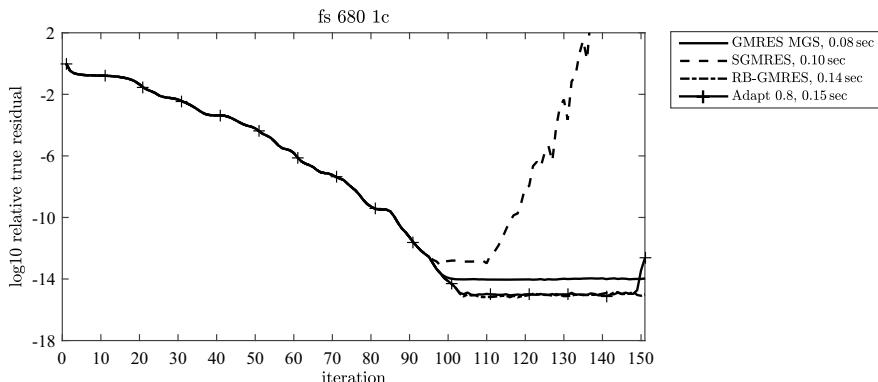


Fig. 6.5 *fs 680 1c*, relative true residual norms, GMRES-MGS (plain), SGMRES (dashed), RB-GMRES (dot-dashed), ASGMRES $v = 0.8$ (+)

The same conclusions apply for `supg 001` of order 1225; see Figure 6.6. Again, SGMRES shows its instability when reaching the maximum attainable accuracy.

Since the residual norms of SGMRES blow up sooner or later, in Table 6.3 we show the minimum value of the true residual norm over all the k_{max} iterations. Otherwise the final values of SGMRES would have been very large. However, in many examples, the SGMRES minimum residual norms are larger than for the other methods.

For the set of larger matrices, the results of SGMRES in Table 6.4 seem better, but this is because we show the minimum, remember that its residual norms blow up at some point.

Table 6.3 Minimum relative true residual norms in k_{\max} iterations

matrix	k_{\max}	GMRES-MGS	SGMRES	RB-GMRES	ASGMRES $\nu = 0.8$
pde225	150	1.66541e-15	2.55544e-14	1.27989e-15	3.68902e-15
gre 343	300	6.61396e-15	3.14564e-14	6.68145e-16	5.98457e-16
jphw 991	200	1.07847e-14	1.37119e-13	9.28314e-15	9.27958e-15
pde2961	320	1.65763e-14	3.92361e-13	9.26102e-15	9.37769e-14
jagmesh1	300	4.83286e-16	3.25052e-14	4.02757e-16	3.55361e-16
bfgwa782	350	1.39710e-14	4.44671e-14	5.39375e-14	1.58833e-14
dw2048	1100	1.24776e-15	6.46532e-15	1.09747e-15	1.21797e-15
jagmesh2	800	5.73936e-16	3.31125e-13	6.72391e-16	7.79371e-16
raefsky2	450	7.17708e-15	1.88798e-12	3.38847e-15	2.29060e-13
fs 680 1c	150	8.85755e-15	7.67984e-14	6.40334e-16	7.77313e-16
add20	600	3.03778e-13	3.40230e-11	2.03511e-13	1.58690e-13
raefsky1	350	6.06386e-14	1.94738e-13	2.54626e-14	6.13499e-13
jagmesh4	700	4.62824e-15	1.15546e-12	5.08710e-16	8.72015e-16
fs 680 1	200	7.02671e-15	4.90941e-12	5.03639e-16	5.10840e-13
sherman1	500	4.72052e-14	6.89126e-12	3.94778e-14	3.57346e-12
nos3	320	9.26899e-14	3.31972e-12	1.62635e-14	1.51686e-12
sherman5	1200	1.21236e-11	1.78932e-11	2.08652e-10	1.69157e-11
cavity05	600	1.65859e-13	2.62270e-10	6.10359e-15	1.66516e-11
e05r0500	260	2.65648e-05	2.65648e-05	4.33243e-11	2.02145e-10
comsol	300	7.24522e-11	1.88602e-10	3.61145e-10	1.30673e-10
olm1000	600	3.26048e-14	3.77989e-12	1.81744e-13	6.25317e-14
cavity10	900	1.68665e-13	7.96853e-10	6.39104e-15	1.15959e-10
steam2	200	2.24697e-10	1.44576e-09	1.65479e-11	4.03557e-10
1138bus	700	5.68921e-12	1.80553e-09	4.14365e-12	1.80553e-09
steam1	250	3.71137e-10	7.83740e-09	6.36823e-13	2.15918e-09
bcsstk26	1000	4.13181e-08	4.18784e-08	4.13181e-08	4.13247e-08
nos7	500	9.11821e-08	1.16390e-07	9.67950e-08	1.16808e-07
watt1	350	7.17226e-09	2.32619e-07	6.57789e-10	2.93042e-07
bcsstk14	1000	1.38535e-01	1.38535e-01	1.38535e-01	1.38535e-01
fs 183 6	60	1.04100e-16	1.83298e-08	2.03522e-16	1.44756e-16
bcsstk20	500	1.67269e-03	1.70309e-03	1.18616e-07	3.10737e-04
mcf6	820	1.01382e-05	2.20585e-03	2.56926e-07	1.29097e-03
nnc	250	4.55189e-04	4.53490e-04	1.62978e-02	4.10112e-04
lnsp	420	1.66223e-03	2.45371e-02	6.42957e-01	2.29107e-02

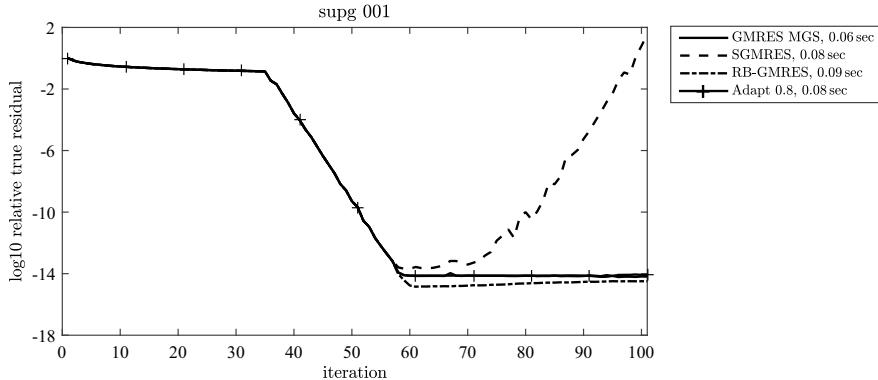


Fig. 6.6 supg 001, relative true residual norms, GMRES-MGS (plain), SGMRES (dashed), RB-GMRES (dot-dashed), ASGMRES $\nu = 0.8$ (+)

Table 6.4 Minimum relative true residual norms in k_{\max} iterations

matrix	k_{\max}	GMRES-MGS	SGMRES	RB-GMRES	ASGMRES $\nu = 0.8$
add32	200	7.16632e-15	1.44708e-13	2.38712e-15	2.10829e-15
ex37	150	1.75693e-13	7.69855e-14	1.72103e-15	5.15337e-15
memplus	900	9.17631e-12	6.38342e-11	1.77987e-12	2.21473e-12
sherman3	800	1.93397e-10	3.60560e-11	6.98711e-12	3.60560e-11
wang4	700	7.40182e-15	1.75513e-11	4.44880e-15	1.53437e-12
supg 100	1000	4.30334e-14	5.35141e-12	1.34670e-14	9.83047e-14
supg 1	1100	3.67990e-14	2.05605e-12	1.09792e-14	4.37149e-13
supg 01	500	2.97491e-14	5.48092e-14	4.59160e-15	5.04748e-14
supg 001	150	1.02025e-14	2.36916e-15	2.00625e-15	3.72895e-15
supg 0001	120	2.06592e-14	2.70764e-14	4.63257e-15	1.50725e-14
supg 00001	220	5.95295e-14	1.06197e-13	2.03015e-14	8.37108e-14
supg 000001	240	1.19743e-13	1.35488e-13	3.54637e-14	9.90349e-14
convdiff xu 500	1100	6.30921e-14	1.05856e-12	1.11331e-02	6.06198e-13
convdiff xu 1000	700	2.36299e-13	2.25159e-13	1.51743e-13	3.52318e-14
convdiff xu 5000	400	3.55342e-13	6.44338e-13	2.91552e-12	2.01965e-13

6.7.3 FOM and Orthores

Orthores is mathematically equivalent to FOM, but Figure 6.7 shows that for some problems (here $\text{fs } 183\ 6$) the results may be quite different. The maximum attainable accuracy of Orthores is much worse than that of FOM.

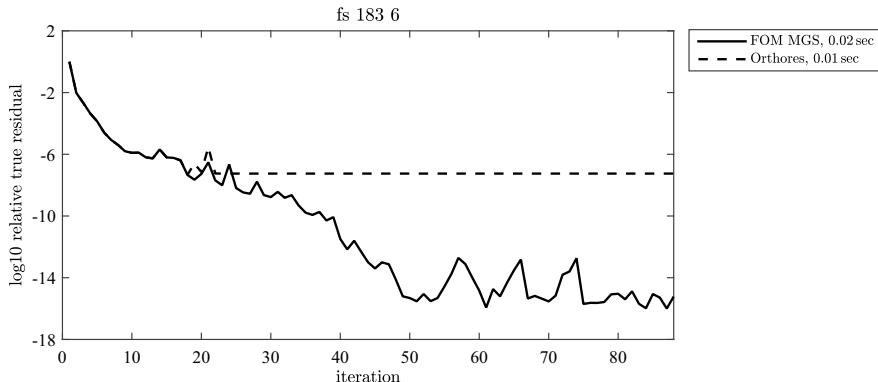


Fig. 6.7 $\text{fs } 183\ 6$, relative true residual norms, FOM-MGS (plain), Orthores (dashed)

For $\text{fs } 680\ 1c$ the residual norms of FOM and Orthores are approximately the same except at the end of the computation where Orthores does not show the same oscillations as FOM (which are related to the stagnation of GMRES), see Figure 6.8. This is also what happens for $\text{supg } 001$ of order 1225; see Figure 6.9.

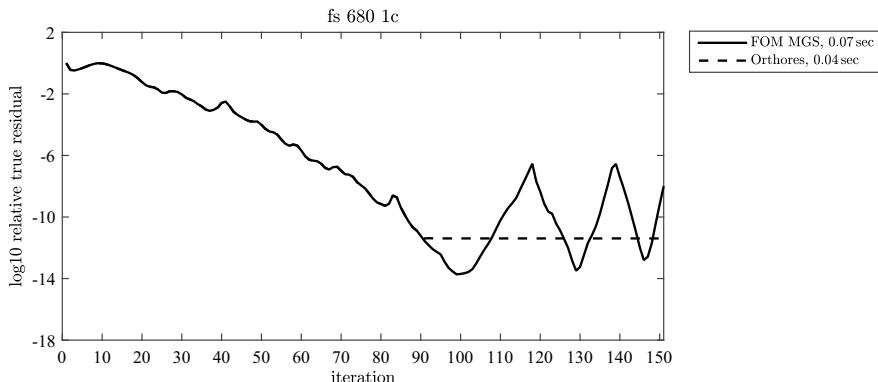


Fig. 6.8 $\text{fs } 680\ 1c$, relative true residual norms, FOM-MGS (plain), Orthores (dashed)

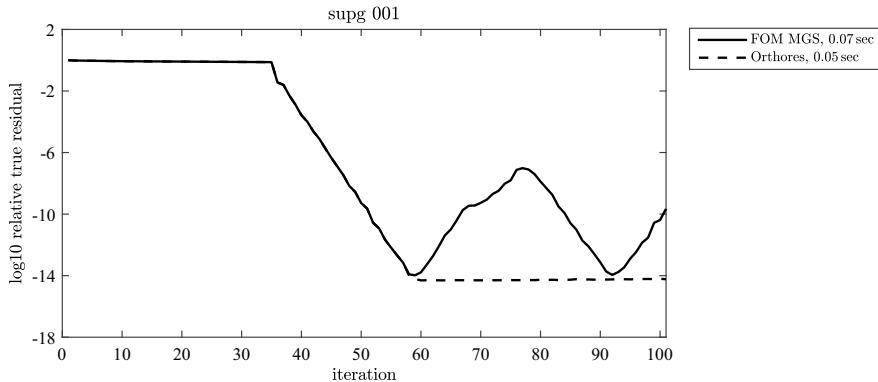


Fig. 6.9 supg 001, order 1225, relative true residual norms, FOM-MGS (plain), Orthores (dashed)

6.7.4 Q-OR Optinv

Q-OR Optinv is also a method which is mathematically equivalent to GMRES. However, for the difficult problem fs 183 6 in Figure 6.10, the residual norms are not exactly the same for both methods with an advantage for GMRES.

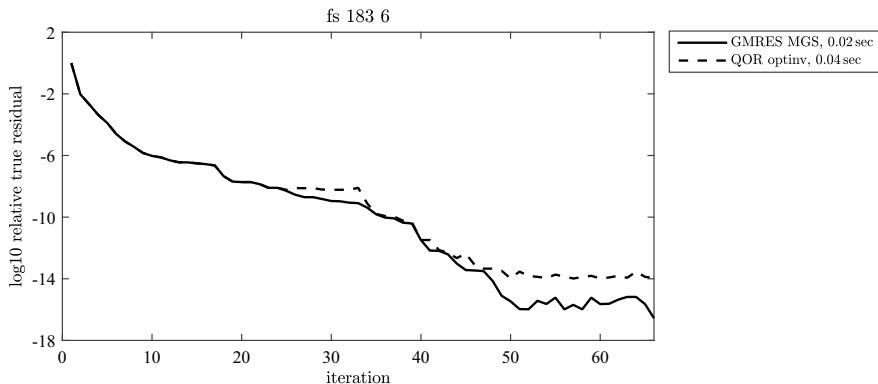


Fig. 6.10 fs 183 6, relative true residual norms, GMRES-MGS (plain), Q-OR Optinv (dashed)

For fs 680 1c the residual norms are the same and the maximum attainable accuracy of Q-OR Optinv is better than that of GMRES; see Figure 6.11. We have the same situation for supg 001 of order 1225 as one can see in Figure 6.12. In these examples, Q-OR Optinv also requires less computational time.

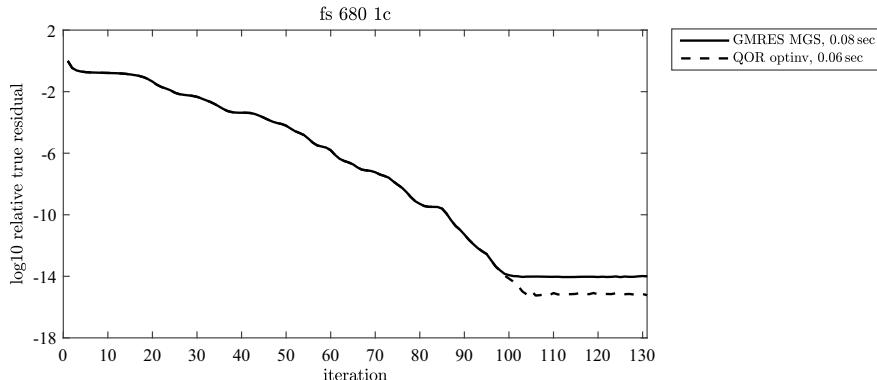


Fig. 6.11 fs 680 1c, relative true residual norms, GMRES-MGS (plain), Q-OR Optinv (dashed)

Table 6.5 gives the minimum true residual norms for GMRES and Q-OR Optinv as well as their ratios. Of 42 linear systems, Q-OR Optinv gives better results in 19 of them, GMRES is better in 12 of them and in 11 of them the results are almost the same. Looking at the properties of the matrices (see Appendix A) one can see that Q-OR Optinv has difficulties for the matrices which are strongly indefinite or close to skew-symmetric.

For the set of larger matrices in Table 6.6 Q-OR Optinv gives better results than GMRES except for the last three examples whose matrices are strongly indefinite.

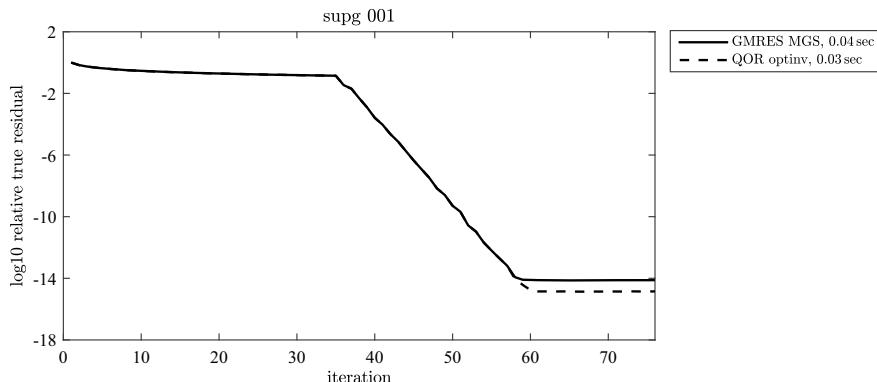


Fig. 6.12 supg 001, order 1225, relative true residual norms, GMRES-MGS (plain), Q-OR Optinv (dashed)

Table 6.5 Minimum relative true residual norms and ratio

matrix	k_{\max}	GMRES-MGS	Q-OR optimv	GMRES/Q-OR
pde225	150	2.33894e-15	1.24803e-15	1.87411
gre 343	300	6.82014e-15	4.35870e-15	1.56472
jphw 991	200	1.43062e-14	7.82225e-15	1.82891
pde2961	320	1.70615e-14	9.69758e-15	1.75935
jagmesh1	300	7.05437e-16	1.82244e-15	3.87084e-01
bawa782	350	1.50836e-14	5.35295e-13	2.81782e-02
dw2048	1100	1.47704e-15	4.00677e-15	3.68636e-01
jagmesh2	800	6.62812e-16	1.55461e-14	4.26352e-02
raefsky2	450	7.85611e-15	4.40245e-15	1.78449
fs 680 1c	150	1.09448e-14	5.92448e-16	18.4739
add20	600	3.68830e-13	2.28542e-13	1.61384
raefsky1	350	6.24376e-14	2.60170e-14	2.39988
jagmesh4	700	5.17707e-15	6.40820e-15	8.07882e-01
fs 680 1	200	7.07705e-15	7.10933e-16	9.95460
sherman1	500	5.21414e-14	4.24228e-14	1.22909
nos3	320	9.26989e-14	1.70657e-14	5.43189
sherman5	1200	1.43433e-11	1.47142e-08	9.74796e-04
cavity05	600	1.67647e-13	1.87813e-14	8.92627
e05r0500	260	2.65648e-05	2.65655e-05	9.99971e-01
comsol	300	7.79617e-11	6.21624e-10	1.25416e-01
olm1000	600	4.19834e-14	6.28224e-13	6.68288e-02
cavity10	900	1.75702e-13	7.62644e-15	23.0386
steam2	200	2.92273e-10	1.93172e-10	1.51302
1138bus	700	6.05676e-12	6.58031e-12	9.20437e-01
steam1	250	5.54249e-10	2.08447e-11	26.5894
bcsstk26	1000	4.13181e-08	4.13181e-08	1.00000
nos7	500	1.50911e-07	1.43264e-07	1.05338
watt1	350	7.23976e-09	1.21450e-02	5.96109e-07
bcsstk14	1000	1.38535e-01	1.38535e-01	1.00000
fs 183 6	60	2.38970e-16	1.21458e-14	1.96752e-02
bcsstk20	500	1.67269e-03	1.67269e-03	1.00000
mcf6	820	1.01382e-05	—	—
nnc	250	6.17146e-04	—	—
Insp	420	2.24727e-03	—	—

Table 6.6 Minimum relative true residual norms and ratio

matrix	k_{\max}	GMRES-MGS	Q-OR optimv	GMRES/Q-OR
add32	200	7.76040e-15	2.87075e-15	2.70326
ex37	150	1.76266e-13	1.59900e-15	110.235
memplus	900	9.17996e-12	1.73394e-12	5.29428
sherman3	800	2.06205e-10	1.01562e-11	20.3033
wang4	700	7.67237e-15	4.70488e-15	1.63073
supg 100	1000	4.35841e-14	1.30874e-14	3.33025
supg 1	1100	3.72296e-14	1.12041e-14	3.32285
supg 01	500	3.03543e-14	4.85663e-15	6.25007
supg 001	150	1.07687e-14	2.12622e-15	5.06472
supg 0001	120	2.06803e-14	4.28876e-15	4.82199
supg 00001	220	5.97492e-14	2.05436e-14	2.90841
supg 000001	240	1.22615e-13	4.46706e-14	2.74488
convdiff xu 500	1100	6.39745e-14	1.17262e-01	5.45571e-13
convdiff xu 1000	700	2.36826e-13	1.16127e-03	2.03937e-10
convdiff xu 5000	400	3.59316e-13	6.09165e-10	5.89850e-04

6.8 Historical notes

The conjugate residual (CR) method was introduced for symmetric problems by Eduard Stiefel [888] in 1955.

The paper about Orthomin [953] was published in 1976 by Paul K.W. Vinsome, who was working for the Brunei Shell Petroleum Company, in the proceedings of the Fourth Symposium on Reservoir Simulation organized by the Society of Petroleum Engineers in Los Angeles (USA).

A generalized conjugate gradient algorithm was introduced by Owe Axelsson in 1980 in the paper [43] received in 1978. Vinsome is cited in this paper. The method GCG-LS described in this chapter was proposed in 1987 [44] as an extension of the 1980 method. A truncated version is also studied. The paper was received in 1986.

The generalized conjugate residual (GCR) method was introduced in the Ph.D. thesis of Howard C. Elman [308] in 1982; see chapter 5 of the thesis. Convergence was proved for matrices having a positive definite symmetric part and a bound provided for the norm of the residual vector. Axelsson's generalization of the conjugate residual method was studied in chapter 6 of the thesis. Relations between GCR and Orthomin were also considered. Parts of this work as well as other results published in Yale University reports were summarized in the paper by Stanley C. Eisenstat, H.C. Elman and Martin H. Schultz [301] in 1983. This paper was first received in 1981 and in revised form in 1982. One can read also with interest the paper [799] published in 1985 by Yousef Saad and M.H. Schultz.

Orthodir and Orthores were proposed by Kang C. Jea and David M. Young in 1980-1983. A paper titled “Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods” [1014] was published in 1980. Three generalizations of CG were introduced in that paper, namely, Orthomin, Orthodir and Orthores. Simplifications in certain cases were studied in the 1983 paper [560]. For these methods, see also the Ph.D. thesis of K.C. Jea [559] from 1982.

To avoid the triangularization of the upper Hessenberg matrix in GMRES Homer F. Walker and Lu Zhou [969] introduced the simpler GMRES method. Their paper was received in 1992.

Minimum residual methods using sequences of least squares problems were studied in the paper [635] by Jörg Liesen, Miroslav Rozložník and Zdeněk Strakoš in 2002. They stressed that the choice of the basis is fundamental for the numerical stability of the implementation and proved that simpler GMRES is less numerically stable than the standard GMRES implementation.

The influence of the choice of the basis on the stability of the method was also considered in a paper by Pavel Jiránek, M. Rozložník and Martin H. Gutknecht [577] published in 2008; see also [575]. They introduced a generalized simpler approach of which simpler GMRES is a special case as well as the residual-based simpler GMRES (RB-SGMRES). They analyzed the maximum attainable accuracies of these methods in finite precision arithmetic. In particular they showed that the RB-SGMRES implementation is conditionally backward stable. Note that the maximum attainable accuracy of recursively computed residual methods was studied in 1997 by Anne Greenbaum [454].

An adaptive method mixing simpler GMRES and RB-SGMRES was proposed by P. Jiránek and M. Rozložník [576] in 2010.

The optimal Q-OR Krylov subspace method was proposed in 2017 [685].

The parallel s -step GCR method was proposed by Anthony T. Chronopoulos [222] in 1991; see also the joint papers with Charles D. Swanson [228, 900] in 1992 and 1996.



7.1 Derivation of the methods

CMRH (which is an acronym for Changing Minimal Residual method based on the Hessenberg process) was introduced in [805]. It is a Q-MR method which uses the Hessenberg basis whose construction was described in Chapter 4, Section 4.2. The corresponding Q-OR method is called the Hessenberg method [518].

The Hessenberg basis is obtained with a progressive LU factorization of the Krylov matrix K with partial pivoting. In the following code for CMRH the matrix LP is a row permuted lower triangular matrix. The vector s is a permutation vector. If we assume that we can iterate up to $k = n$ and I is the identity matrix, the permutation matrix is defined by $P^T = I(s, :)$ and $P^T K = P^T LP U = LU$ with $L = P^T LP$, unit lower triangular and U upper triangular. Moreover, we have $A LP = LP H$ which gives an Arnoldi-like relation. In the code below we use, as in GMRES, Givens rotations for solving the Q-MR least squares problem. To obtain a code for the Hessenberg method we have to do the same code modifications that we did to obtain FOM from GMRES.

```
function [x,ni,resn] = CMRH(A,b,x0,epsi,nitmax);
%
n = size(A,1);
rhs = zeros(nitmax+1,1);
H = zeros(nitmax+1,nitmax);
rot = zeros(2, nitmax); % init Givens rotations
resn = zeros(1,nitmax+1);
LP = zeros(n,nitmax+1);
s = [1:n]';
x = x0;
r = b - A * x;
ni = 0;
nb = norm(b);
```

```

i0 = index(r);
LP(:,1) = r / r(i0);
s = swap(s,1,i0);
bet = r(i0);
resn(1) = norm(r);
rhs(1) = bet;
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    w = A * LP(:,k); % matrix vector product
    for i = 1:k
        c = w(s(i));
        H(i,k) = c;
        w(s(i)) = 0;
        si = s(i+1:n);
        w(si) = w(si) - c * LP(si,i);
    end % for i
    if k+1 < n
        sj = s(k+1:n);
        i1 = index(w(sj));
        i0 = i1 + k;
        s = swap(s,k+1,i0);
        maxw = w(sj(i1));
    else
        maxw = w(n);
    end % if
    H(k+1,k) = maxw;
    LP(:,k+1) = w / maxw;
    nw1 = maxw;
    % apply the preceding Givens rotations to the last column
    for kk = 1:k-1
        h1 = H(kk,k);
        h2 = H(kk+1,k);
        H(kk+1,k) = -rot(2,kk) * h1 + rot(1,kk) * h2;
        H(kk,k) = rot(1,kk) * h1 + conj(rot(2,kk)) * h2;
    end % for kk
    % compute, store and apply a new rotation to zero
    % the last term in kth column
    nw = H(k,k);
    [cc,ss] = givens(nw,nw1);
    % store the rotation for the next columns
    rot(1,k) = cc; % cosine
    rot(2,k) = ss; % sine
    % modify the diagonal entry and the right-hand side
    H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
    c = rhs(k);

```

```

rhs(k) = rot(1,k) * c;
rhs(k+1) = -rot(2,k) * c;
nresidu = abs(rhs(k+1)); % estimate of the residual norm
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k
% computation of the solution
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + LP(:,1:k) * y;
resn = resn(1:ni+1);
end

function s = swap(s,i,j);
ss = s(i);
s(i) = s(j);
s(j) = ss;
end
%
function i0 = index(u);
[y,I] = max(abs(u));
i0 = I(1);
end

```

It turns out that the preceding code, which is a more or less straightforward translation of the algorithm, is quite slow because of the indirect addressing which is done in $w(si) = w(si) - c * LP(si, i)$. The following code, which is using explicit permutations of the rows, is much faster but less easy to understand. In this code LPs is a row permuted version of LP . For convenience, we also use LP but this can be avoided if it is necessary to save storage. Note that this code is not fully optimized. For instance, it is not necessary to recompute all the inverse permutations at each iteration since only two components have been swapped in the previous iteration. The maximum can be sought in ws and not in w . We also had to introduce additional functions to swap rows of some matrices. The new code is 3 to 5 times faster than the old one. This illustrates the fact that, when comparing algorithms, we have to be careful about the coding which may have a large influence on the computing times (which, moreover, can be misleading in Matlab when they are small).

```

function [x,ni,resn] = CMRH_new(A,b,x0,epsi,nitmax);
%
n = size(A,1);
nitmax = min(nitmax,n);
rhs = zeros(nitmax+1,1);
H = zeros(nitmax+1,nitmax);
rot = zeros(2, nitmax); % init Givens rotations

```

```

resn = zeros(1,nitmax+1);
LP = zeros(n,nitmax+1);
LPs = zeros(n,nitmax+1);
s = [1:n]';
x = x0;
r = b - A * x;
ni = 0;
nb = norm(b);
i0 = index(r);
LP(:,1) = r / r(i0);
s = swap(s,1,i0);
LPs(:,1) = swap(LP(:,1),1,i0);
bet = r(i0);
resn(1) = norm(r);
rhs(1) = bet;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    w = A * LP(:,k); % matrix vector product
    ws = w(s);
    wst = LPs(1:k,1:k) \ ws(1:k);
    H(1:k,k) = wst;
    ws(1:k) = 0;
    ws(k+1:n) = ws(k+1:n) - sum(LPs(k+1:n,1:k) * wst,2);
    sp = invperm(s);
    w = ws(sp);
    if k+1 < n
        sj = s(k+1:n);
        i1 = index(w(sj));
        i0 = i1 + k;
        s = swap(s,k+1,i0);
        maxw = w(sj(i1));
    else
        maxw = w(n);
    end % if
    H(k+1,k) = maxw;
    LP(:,k+1) = w / maxw;
    LPs(:,k+1) = ws / maxw;
    LPs = swapLP(LPs,k+1,i0);
    nw1 = maxw;
    % apply the preceding Givens rotations to the last column
    for kk = 1:k-1
        h1 = H(kk,k);
        h2 = H(kk+1,k);
        H(kk+1,k) = -rot(2,kk) * h1 + rot(1,kk) * h2;
    end
end

```

```

H(kk,k) = rot(1, kk) * h1 + conj(rot(2, kk)) * h2;
end % for kk
% compute, store and apply a new rotation to zero
% the last term in kth column
nw = H(k,k);
[cc,ss] = givens(nw,nw1);
% store the rotation for the next columns
rot(1,k) = cc; % cosine
rot(2,k) = ss; % sine
% modify the diagonal entry and the right-hand side
H(k,k) = rot(1,k) * nw + conj(rot(2,k)) * nw1;
c = rhs(k);
rhs(k) = rot(1,k) * c;
rhs(k+1) = -rot(2,k) * c;
nresidu = abs(rhs(k+1)); % estimate of the residual norm
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
  break % get out of the k loop
end % if nresidu
end % for k
% computation of the solution
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + LP(:,1:k) * y;
resn = resn(1:ni+1);
end
%
function ip=invperm(perm);
%inverse permutation of perm
n = length(perm);
in = [1:n];
ip(perm) = in;
end
%
function L = swapLP(L,i,j);
ss = L(i,:);
L(i,:) = L(j,:);
L(j,:) = ss;
end

```

CMRH is an interesting method, unfortunately not so well known, because there are no dot products involved. But, for pivoting, we have to find the maximum component of a given vector (this is done in the function `index` in the codes above) and this can also be a problem on parallel computers. We note that there is only one matrix–vector product per iteration as in GMRES. The number of operations per iteration is smaller than in GMRES since the computation of the basis vectors

involves smaller and smaller vectors as the iteration number increases. This may be mitigated by the fact that, if the algorithm is coded straightforwardly, there is some indirect addressing which can slow down the computation.

7.2 Comparison with GMRES and convergence of CMRH

To compare CMRH with GMRES we can use the results of Section 3.9 in Chapter 3 and those of the paper [807]. Let V, U and V^A, U^A be the matrices arising from CMRH and GMRES (the index A referring to Arnoldi). We assume these matrices to be nonsingular. We have the relation

$$K = VU = PLU = V^AU^A.$$

Therefore, since $(V^A)^*V^A = I$, we have

$$(V^A)^*PL = U^AU^{-1}. \quad (7.1)$$

We note that we have also $K_{n,k} = PL_{n,k}U_k = V_{n,k}^AU_k^A$. Hence, $L_{n,k} = P^T V_{n,k}^A U_k^A U_k^{-1} = P^T V_{n,k}^A R_k$, R_k being an upper triangular matrix.

Let r_k^G (resp. r_k^C) be the residual vectors of GMRES (resp. CMRH). Since both methods use the same Krylov subspace and CMRH is a Q-MR method, we have $\|r_k^G\| \leq \|r_k^C\|$. The residual norms are related by the following result which was proved in [807].

Theorem 7.1 *We have the following relations between the residual norms of GMRES and CMRH,*

$$\|r_k^G\| \leq \|r_k^C\| \leq \kappa(L_{n,k+1}) \|r_k^G\|. \quad (7.2)$$

Proof We proceed as in [807]. The residual vectors are in $\mathcal{K}_{k+1}(A, r_0)$. Then we can write

$$\begin{aligned} r_k^C &= PL_{n,k+1}z^C, \\ r_k^G &= PL_{n,k+1}z^G = (V^A)_{n,k+1}w^G. \end{aligned}$$

The matrix $(V^A)_{n,k+1}$ is unitary and the second equation yields

$$w^G = (V^A)_{n,k+1}^*PL_{n,k+1}z^G = R_{k+1}z^G.$$

Hence, since CMRH minimizes the norm of the quasi-residual,

$$\|z^C\| \leq \|z^G\| \leq \|R_{k+1}^{-1}\| \|w^G\| = \|R_{k+1}^{-1}\| \|r_k^G\|.$$

Moreover, we have

$$\|r_k^C\| \leq \|L_{n,k+1}\| \|z^C\|.$$

Hence, since $\|L_{n,k+1}\| = \|R_{k+1}\|$,

$$\begin{aligned}\|r_k^C\| &\leq \|L_{n,k+1}\| \|z^C\|, \\ &\leq \|L_{n,k+1}\| \|R_{k+1}^{-1}\| \|r_k^G\| \\ &= \kappa(R_{k+1}) \|r_k^G\|.\end{aligned}$$

But, we note that $\kappa(R_{k+1}) = \kappa(L_{n,k+1})$, where $\kappa(L_{n,k+1})$ is defined using the pseudo-inverse of $L_{n,k+1}$, which proves the result. \square

This result could also have been proved using Theorem 3.23. This theorem can also be used to obtain results for the Hessenberg method in comparison with FOM. Theorem 7.1 shows that, as long as the condition number of $L_{n,k+1}$ is not too large, the residual norm of CMRH is not much larger than the residual norm of GMRES at a given iteration. The norm of $L_{n,k+1}$ is bounded by the Frobenius norm. It yields

$$\|L_{n,k+1}\| \leq \left(\frac{k+1}{2} (2n-k) \right)^{\frac{1}{2}}.$$

Even though, there is no formal proof of that fact, lower triangular matrices obtained in Gaussian elimination with partial pivoting tend to be well conditioned, see [874].

7.3 Prescribed convergence

To construct a matrix A with prescribed eigenvalues and a right-hand side b such that we generate a given Hessenberg residual norm convergence curve, we first proceed as for any Q-OR method in Chapter 3. We take any upper triangular matrix U^{-1} such that the moduli of the inverses of the entries of the first row are the given Hessenberg relative residual norms (allowing zero entries to force infinite residual norms). Then, C being the companion matrix corresponding to the given eigenvalues, we set $H = UCU^{-1}$. Finally, we have to choose the matrix V such that $A = VHV^{-1}$, $b = Ve_1$. The Hessenberg/CMRH methods are based on an LU factorization with partial pivoting of the Krylov matrix giving the factorization $P^T K = LU$, hence the decomposition of the Krylov matrix takes the form

$$K = VU, \quad V = PL,$$

with V a nonsingular row permuted lower triangular matrix. We observe that in CMRH the basis vectors are not of unit norm. They are scaled such that the maximum of the moduli of the components is equal to 1. Therefore we choose V as any nonsingular row permuted lower triangular matrix with the columns scaled by their maximum absolute values. Using the same construction but with the first row of U^{-1} satisfying the conditions (3.17) of Chapter 3, we can prescribe the quasi-residual norms for the CMRH method, with any nonzero eigenvalues.

This construction shows that any residual (resp. quasi-residual) norm history (with, for the Hessenberg method, infinite residual norms being allowed) is possible for the Hessenberg (resp. CMRH) method with any nonzero eigenvalues of the system matrix.

With a particular choice of V we can not only prescribe CMRH quasi-residual norms, but even CMRH residual norms. Therefore, any non-increasing residual norm history is possible for the CMRH method with any nonzero eigenvalues of the system matrix. This claim follows when we apply CMRH to the matrix $H = UCU^{-1}$ with $b = e_1$, where U and C are chosen as before. Then both P and L are the identity matrix and so is V . Thus V is in particular unitary and the residual norms equal the prescribed quasi-residual norms.

7.4 Stagnation of CMRH

Stagnation is just a particular case of prescribed convergence curve. If we want to construct a matrix giving complete stagnation of the CMRH quasi-residual norms, it is sufficient to construct U^{-1} with a zero first row except for the first entry being equal to 1.

7.5 Residual norms

We obtain bounds for the CMRH residual norms by using Theorem 7.1,

$$\|r_k^C\| \leq \kappa(L_{n,k+1}) \|r_k^G\|.$$

Then, we can use all the bounds of the GMRES residual norms that we have described in Chapter 5, Section 5.2. As long as $\kappa(L_{n,k+1})$ is not too large, this gives reasonable upper bounds for CMRH.

7.6 Parallel computing

Up to our knowledge, CMRH has only been used on parallel computers to solve linear systems with a dense matrix, see [292, 293, 519]. An implementation of CMRH minimizing the storage is described in [519]. A parallel version of CMRH is considered in [292].

7.7 Finite precision arithmetic

As far as we know, there is no rounding error analysis of CMRH. However, since this method is implicitly doing a partial pivoting LU factorization of the Krylov matrix K , it is likely that it cannot be proved to be backward stable since the growth factor may be large; see [521].

7.8 Numerical experiments

CMRH is a Q-MR method and therefore does not minimize the residual norm but only the quasi-residual norm. Of course, the CMRH residual norms must be larger than those of GMRES, at least before the final stagnation phase. However, in many cases, CMRH produces residual norms which are close to those of GMRES (at least on a log-scale). This is what we observe on Figure 7.1 for the difficult problem `fs 183 6`. The differences in the final stagnation phase are not much significant since the residual norms are at the round-off level. For this example, on the first 60 iterations, the maximum of the condition number of $L_{n,k}$ is 39.55. The maximum of the ratio of the norms of the residual and quasi-residual is 5.31.

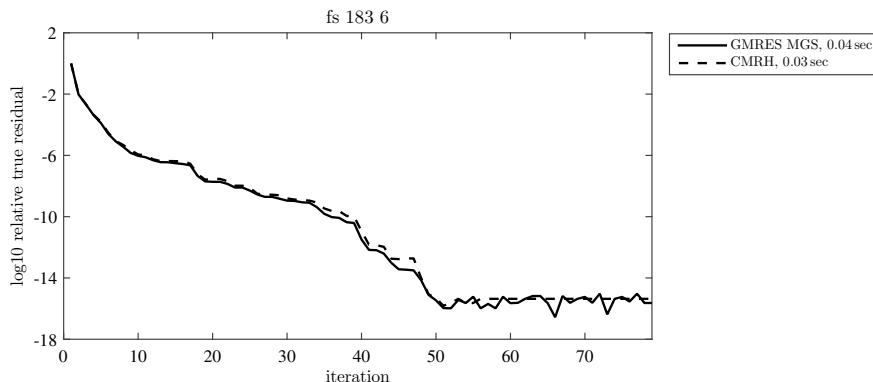


Fig. 7.1 `fs 183 6`, relative true residual norms, GMRES-MGS (plain), CMRH (dashed)

For `fs 680 1c` the CMRH residual norms are slightly larger than those of GMRES up to iteration 100 but the CMRH maximum attainable accuracy is better than that of GMRES; see Figure 7.2. On the first 110 iterations, the maximum of the condition number of $L_{n,k}$ is 396.5. The maximum of the ratio of the norms of the residual and quasi-residual is 33.90.

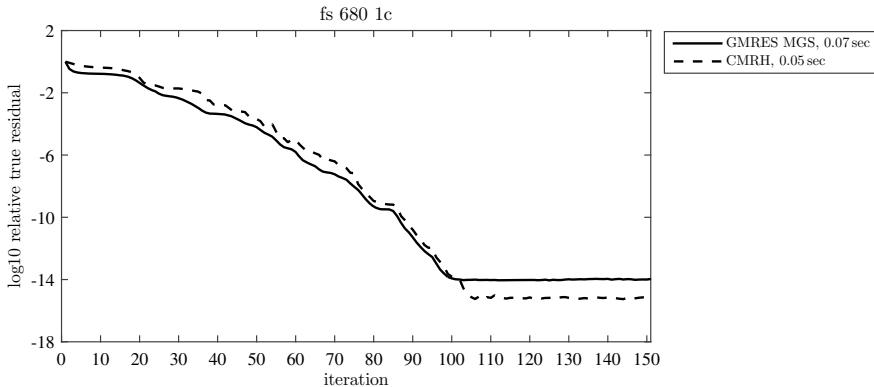


Fig. 7.2 *fs 680 1c*, relative true residual norms, GMRES-MGS (plain), CMRH (dashed)

The relative true residual norms of both methods are even closer for the problem *supg 001* of order 1225 than for the other examples as we can see in Figure 7.3. On the first 60 iterations, the maximum of the condition number of $L_{n,k}$ is 20.80. The maximum of the ratio of the norms of the residual and quasi-residual is 4.51.

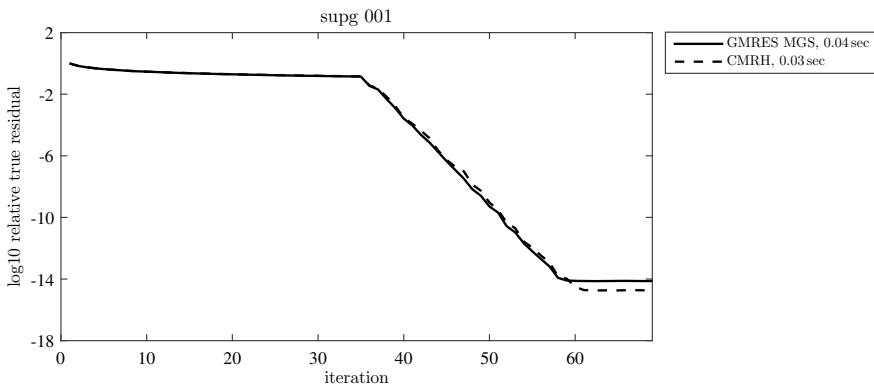


Fig. 7.3 *supg 001*, order 1225, relative true residual norms, GMRES-MGS (plain), CMRH (dashed)

Table 7.1 shows the minimum relative true residual norms for GMRES and CMRH as well as their ratio for our first set of matrices. The accuracy of CMRH is better than that of GMRES for 22 examples out of 42.

Table 7.2 gives the results for the set of larger matrices. There is only one problem for which the accuracy of CMRH is much worse than that of GMRES. All these

Table 7.1 Minimum relative true residual norms in k_{\max} iterations and ratio

matrix	k_{\max}	GMRES-MGS	CMRH	GMRES/CMRH
pde225	150	1.66541e-15	1.07562e-15	1.54833
gre 343	300	6.61396e-15	5.38541e-16	12.2813
jphw 991	200	1.07847e-14	7.46209e-15	1.44527
pde2961	320	1.65763e-14	1.05942e-14	1.56466
jagmesh1	300	4.83286e-16	3.67303e-16	1.31577
bawa782	350	1.39710e-14	1.20095e-14	1.16333
dw2048	1100	1.24776e-15	1.00148e-15	1.24592
jagmesh2	800	5.73936e-16	4.54574e-16	1.26258
raefsky2	450	7.17708e-15	2.75695e-14	0.260327
fs 680 1c	150	8.85755e-15	5.35415e-16	16.5433
add20	600	3.03778e-13	1.02541e-13	2.96249
raefsky1	350	6.06386e-14	1.73714e-13	0.349071
jagmesh4	700	4.62824e-15	9.84334e-16	4.70190
fs 680 1	200	7.02671e-15	6.05256e-16	11.6095
sherman1	500	4.72052e-14	2.37795e-14	1.98512
nos3	320	9.26899e-14	6.43040e-14	1.44143
sherman5	1200	1.21236e-11	1.96716e-11	0.616300
cavity05	600	1.65859e-13	4.41359e-13	0.375792
e05r0500	260	2.65648e-05	8.07078e-05	0.329147
comsol	300	7.24522e-11	2.26358e-10	0.320077
olm1000	600	3.26048e-14	3.47460e-14	0.938377
cavity10	900	1.68665e-13	3.10370e-13	0.543431
steam2	200	2.24697e-10	5.53152e-11	4.06212
1138bus	700	5.68921e-12	3.43473e-11	0.165638
steam1	250	3.71137e-10	2.31688e-10	1.60189
bcsstk26	1000	4.13181e-08	8.39759e-08	0.492023
nos7	500	9.11821e-08	4.90886e-08	1.85750
watt1	350	7.17226e-09	1.32247e-09	5.42340
bcsstk14	1000	1.38535e-01	2.94518e-01	0.470381
fs 183 6	60	1.04100e-16	1.52136e-16	0.684254
bcsstk20	500	1.67269e-03	5.11757e-03	0.326852
mcf6	820	1.01382e-05	1.14949e-05	0.881972
nnc	250	4.55189e-04	4.67723e-03	0.0973201
Insp	420	1.66223e-03	1.25113e-02	0.132859

results show that CMRH is an interesting alternative to GMRES. We recall that there are no dot products needed in CMRH and there is only one reduction operation per iteration to find the index of the pivot.

Table 7.2 Minimum relative true residual norms in k_{\max} iterations and ratio

matrix	k_{\max}	GMRES-MGS	CMRH	GMRES/CMRH
add32	200	7.16632e-15	2.92470e-15	2.45028
ex37	150	1.75693e-13	1.36624e-15	128.596
memplus	900	9.17631e-12	1.50488e-11	0.609771
sherman3	800	1.93397e-10	2.06393e-11	9.37031
wang4	700	7.40182e-15	1.66806e-14	0.443739
supg 100 6400	1000	4.30334e-14	3.48557e-14	1.23462
supg 1 6400	1100	3.67990e-14	1.13878e-14	3.23143
supg 01 6400	500	2.97491e-14	5.70203e-15	5.21728
supg 001 6400	150	1.02025e-14	2.17970e-15	4.68069
supg 0001 6400	120	2.06592e-14	4.00580e-15	5.15733
supg 00001 6400	220	5.95295e-14	1.90996e-14	3.11679
supg 000001 6400	240	1.19743e-13	4.38339e-14	2.73176
convdiff xu 500 8100	1100	6.30921e-14	2.81234e-11	0.00224341
convdiff xu 1000 8100	700	2.36299e-13	5.19691e-14	4.54691
convdiff xu 5000 8100	400	3.55342e-13	1.63721e-12	0.217042

7.9 Historical notes

The construction of the Hessenberg basis originated in the work of Karl Hessenberg [512] in 1940. This basis was considered by James H. Wilkinson [982] page 377 in his famous 1965 book as a particular case of the generalized Hessenberg process.

It does not seem that this basis attracted much attention until Hassane Sadok used it to define a new Q-MR method in the paper [805] introducing CMRH in 1999. This paper was submitted in June 1998. The corresponding Q-OR method, named the Hessenberg method, was proposed in 1998 in the paper [518] by Mohammed Heyouni and H. Sadok.

An implementation specifically designed for dense linear systems was introduced in another paper [519] by M. Heyouni and H. Sadok in 1998.

In 2012, H. Sadok and Daniel B. Szyld published theoretical comparisons of the residual norms in GMRES and CMRH; see [807].

The parallel implementation of the method for dense linear systems was addressed in the paper [292] by Sébastien Duminil in 2013.

Some software for dense linear systems is described in the paper [293] by S. Duminil, M. Heyouni, Philippe Marion and H. Sadok in 2016.

Chapter 8

BiCG/QMR and Lanczos algorithms



In this chapter we describe methods that are based on the use of the biorthogonal bases we have introduced in Chapter 4, Section 4.3. Their main advantage is that they use short recurrences. Therefore, the storage is not increasing with the iteration number, contrary to methods like GMRES or CMRH.

8.1 The Lanczos algorithms

For simplicity let us assume in this chapter that the data is real; the complex case is well treated in [482]. Moreover, by using A^T instead of A^* and the dot product $x^T y$ even when the data is complex, we can avoid conjugating some coefficients and most of the algorithms in this chapter work for complex problems. Assume we have two sets of basis vectors that satisfy recurrences like,

$$\begin{aligned} AV_{n,k} &= V_{n,k}T_k + \rho_{k+1}v_{k+1}e_k^T, \\ A^TW_{n,k} &= W_{n,k}\tilde{T}_k + \zeta_{k+1}w_{k+1}e_k^T, \end{aligned}$$

where the matrices T_k and \tilde{T}_k are tridiagonal. Choosing properly the initial vectors and as long as there is no breakdown, the vectors v_j , $j = 1, \dots, k$ give a basis of $\mathcal{K}_k(A, r_0)$ and the vectors w_j , $j = 1, \dots, k$ give a basis of $\mathcal{K}_k(A^T, \tilde{r}_0)$ with $\tilde{r}_0 = b - A\tilde{x}_0$ or any other vector since the iterates of the second sequence are generally not needed. Then, we proceed as usual for Q-OR/Q-MR methods, we set the iterates as

$$x_k = x_0 + V_{n,k}y^{(k)}.$$

It yields

$$r_k = r_0 - AV_{n,k}y^{(k)} = r_0 - V_{n,k}T_ky^{(k)} - \rho_{k+1}[y^{(k)}]_k v_{k+1}.$$

Assume $r_0 = \rho_0 v_1$. We obtain

$$r_k = V_{n,k}[\rho_0 e_1 - T_k y^{(k)}] - \rho_{k+1}[y^{(k)}]_k v_{k+1}.$$

The basic Q-OR method is obtained by computing $y^{(k)}$ as the solution of $T_k y^{(k)} = \rho_0 e_1$ and the residual vector is proportional to v_{k+1} . This is sometimes called the Lanczos method for linear systems. However, this method is not very appealing since, even though the basis vectors are obtained by short recurrences, we have to keep all the basis vectors to compute the iterates. Of course, this may be done only once after convergence but, nevertheless, all the basis vectors have to be available at the end of the computation.

Since the residual vector r_k is proportional to v_{k+1} which is orthogonal to the vectors w_j , $j = 1, \dots, k$, it is obvious that r_k is orthogonal to $\mathcal{K}_k(A^T, \tilde{r}_0)$. Now, an interesting question is: What are the conditions needed to be able to construct a residual vector $r_{k+1} \in r_0 + A\mathcal{K}_{k+1}(A, r_0)$ which is orthogonal to $\mathcal{K}_{k+1}(A^T, \tilde{r}_0)$? Let us write

$$r_{k+1} = r_0 - A \sum_{j=0}^k \beta_j^{(k)} A^j r_0.$$

The orthogonality conditions can be written as $(r_{k+1}, [A^T]^j \tilde{r}_0) = 0$, $j = 0, \dots, k$. Let $\psi_j = (\tilde{r}_0, A^j r_0)$, $j = 0, \dots$. From the orthogonality conditions, the coefficients $\beta_j^{(k)}$ must be the solution of the linear system,

$$\begin{pmatrix} \psi_1 & \psi_2 & \cdots & \psi_{k+1} \\ \psi_2 & \psi_3 & \ddots & \psi_{k+2} \\ \vdots & \vdots & & \vdots \\ \psi_{k+1} & \psi_{k+2} & \cdots & \psi_{2k+1} \end{pmatrix} \begin{pmatrix} \beta_0^{(k)} \\ \beta_1^{(k)} \\ \vdots \\ \beta_{k+1}^{(k)} \end{pmatrix} = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_k \end{pmatrix}.$$

The residual vector can be constructed if and only if the determinant of the Hankel matrix on the left-hand side is nonzero. A similar condition exists for the construction of $\tilde{r}_{k+1} \in \tilde{r}_0 + A\mathcal{K}_{k+1}(A^T, \tilde{r}_0)$ which is orthogonal to $\mathcal{K}_{k+1}(A, r_0)$. Of course, these conditions are not always satisfied. Note that the entries of the Hankel matrix depend on the choice of \tilde{r}_0 .

However, since the basis vectors satisfy a three-term recurrence and the residual vectors are proportional to the basis vectors, it is clear that we can also compute the residual vectors (and eventually the iterates) with three-term recurrences. This leads to different variants depending on the normalization that is chosen for the basis vectors.

A first algorithm is as follows. The residual vectors are given by

$$\begin{aligned} r_{k+1} &= (Ar_k - \alpha_k r_k - \beta_k r_{k-1})/\rho_{k+1}, \\ \tilde{r}_{k+1} &= (A^T \tilde{r}_k - \alpha_k \tilde{r}_k - \xi_k \tilde{r}_{k-1})/\zeta_{k+1}. \end{aligned}$$

Let $\delta_k = (r_k, \tilde{r}_k)$. The coefficients of the recurrences, chosen to satisfy the biorthogonality conditions, are given (assuming that the δ_j 's are nonzero) by

$$\alpha_k = \frac{(\tilde{r}_k, Ar_k)}{\delta_k}, \quad \beta_k = \zeta_k \frac{\delta_k}{\delta_{k-1}}, \quad \xi_k = \rho_k \frac{\delta_k}{\delta_{k-1}}.$$

The coefficient ρ_{k+1} is computed as $\rho_{k+1} = -(\alpha_k + \beta_k)$. This condition implies that the polynomial p_{k+1} such that $r_{k+1} = p_{k+1}(A)r_0$ satisfies $p_{k+1}(0) = 1$. It allows to compute the iterates as

$$x_{k+1} = -(r_k + \alpha_k x_k + \beta_k x_{k-1})/\rho_{k+1}.$$

The other coefficient can be chosen as $\zeta_{k+1} = \rho_{k+1}$. By analogy with the methods having long recurrences that we have described in Chapter 6 this method is called Lanczos/Orthores (or BioRes in [482]); see [560, 581]. We have a breakdown of the algorithm at iteration k if $\rho_{k+1} = 0$ and/or if $(r_k, \tilde{r}_k) = 0$.

Another possibility is to use Lanczos three-term recurrences to construct direction vectors p_j, \tilde{p}_j which are formally A -biorthogonal (another name is biconjugate). This can be done as follows. Let $p_0 = r_0$, $\tilde{p}_0 = \tilde{r}_0$ and $\delta_0 = (\tilde{p}_0, Ap_0)$. For $k = 0, 1, \dots$

$$p_{k+1} = (Ap_k - \alpha_k p_k - \beta_k p_{k-1})/\rho_{k+1}, \\ \tilde{p}_{k+1} = (A^T \tilde{p}_k - \alpha_k \tilde{p}_k - \xi_k \tilde{p}_{k-1})/\zeta_{k+1}.$$

The coefficients can be defined in different ways, but let us choose

$$\alpha_k = \frac{(A^T \tilde{p}_k, Ap_k)}{\delta_k}, \quad \beta_k = \zeta_k \frac{\delta_k}{\delta_{k-1}}, \quad \xi_k = \rho_k \frac{\delta_k}{\delta_{k-1}},$$

with $\beta_0 = \xi_0 = 0$. The coefficient δ_{k+1} is computed as $\delta_{k+1} = (\tilde{p}_{k+1}, Ap_{k+1})$. One can check that, if there is no breakdown, this yields vectors such that $(\tilde{p}_i, Ap_j) = 0$ if $i \neq j$. The coefficients ρ_{k+1} and ζ_{k+1} can be chosen such that $\|p_{k+1}\| = \|\tilde{p}_{k+1}\| = 1$.

We use these directions vectors to define the residuals and the iterates as

$$r_{k+1} = r_k - \omega_k Ap_k, \\ x_{k+1} = x_k + \omega_k p_k,$$

with $\omega_k = (\tilde{p}_k, r_k)/\delta_k$. It is clear that (without breakdown) the vectors $\tilde{p}_0, \dots, \tilde{p}_k$ give a basis of $\mathcal{K}_{k+1}(A^T, \tilde{r}_0)$ and the choice of ω_k makes r_{k+1} orthogonal to that subspace. Again, by analogy, this method is called Lanczos/Orthodir (or BioDir in [482]); see [560, 581]. We have a breakdown if $\rho_{k+1} = 0$, $\zeta_{k+1} = 0$ and/or $(\tilde{p}_k, Ap_k) = 0$. Note that the use of the direction vectors p_j and \tilde{p}_j introduce another possible cause of breakdown with the dot product (\tilde{p}_k, Ap_k) .

The algorithm derived from the nonsymmetric Lanczos algorithm which is the most well known is described in the next section. It is called Lanczos/Orthomin although it is most well known as the BiConjugate Gradient (BiCG) algorithm.

8.2 Derivation of BiCG

In this section we show how to derive the standard form of the BiCG algorithm from the Lanczos nonsymmetric recurrences. We proceed as it is done for deriving the Conjugate Gradient algorithm from the symmetric Lanczos method; see, for instance, [676]. Our derivation is a little bit lengthy but we wanted to show all the details.

Let us choose the variant of the Lanczos biorthogonal basis where the vectors v_j and w_j are of unit norm. We have the relations

$$\begin{aligned}\rho_{j+1}v_{j+1} &= Av_j - \alpha_j v_j - \beta_j v_{j-1}, \\ \zeta_{j+1}w_{j+1} &= A^T w_j - \alpha_j w_j - \xi_j w_{j-1},\end{aligned}$$

for $j = 1, \dots, k$. The coefficients ρ_{j+1}, ζ_{j+1} are the normalizing factors and the other coefficients are chosen to enforce the biorthogonality relations. The coefficients give two tridiagonal matrices T_k and \tilde{T}_k of order k which are similar, see Lemma 4.1.

The first step is to incrementally factorize the matrices T_k . We use the factorization described in Section 2.8, $T_k = \bar{L}_k \Delta_k^{-1} \bar{U}_k$ where Δ_k is a diagonal matrix with diagonal entries δ_i and \bar{L}_k (resp. \bar{U}_k) is lower (resp. upper) bidiagonal with the same entries as those of T_k on the subdiagonal (resp. upper diagonal) and the δ_i 's on the diagonal. However, it is convenient to modify this factorization as

$$T_k = \bar{L}_k \Delta_k^{-1} \Omega_k^{-1} \Omega_k \bar{U}_k,$$

where Ω_k is a diagonal matrix with diagonal entries ω_i that we will choose later on to simplify some formulas. We denote this factorization as

$$T_k = L_k \Omega_k^{-1} U_k, \quad L_k = \bar{L}_k \Delta_k^{-1}, \quad U_k = \Omega_k \bar{U}_k, \quad (8.1)$$

where L_k (resp. U_k) is lower (resp. upper) bidiagonal and Ω_k is diagonal. L_k also has unit diagonal; its lower bidiagonal entries are ρ_i/δ_{i-1} . Replacing T_k by its factorization, we have the matrix relation

$$AV_{n,k} = V_{n,k} L_k \Omega_k^{-1} U_k + G_{n,k},$$

where $G_{n,k} = \rho_{k+1} v_{k+1} e_k^T$ is zero except for the last column. We multiply this relation to the right by U_k^{-1} to obtain

$$AV_{n,k}U_k^{-1} = V_{n,k}L_k\Omega_k^{-1} + G_{n,k}U_k^{-1}.$$

We introduce a new matrix $P_{n,k} = V_{n,k}U_k^{-1}$ and we denote the columns of $P_{n,k}$ as p_0, p_1, \dots, p_{k-1} . Therefore,

$$AP_{n,k} = V_{n,k}L_k\Omega_k^{-1} + G_{n,k}U_k^{-1}.$$

Let us now consider the iterates x_k , which are, for the QOR method,

$$x_k = x_0 + \rho_0 V_{n,k}T_k^{-1}e_1.$$

We plug the factorization of T_k in this relation and we obtain

$$x_k = x_0 + \rho_0 V_{n,k}U_k^{-1}\Omega_k L_k^{-1}e_1 = x_0 + \rho_0 P_{n,k}\Omega_k L_k^{-1}e_1.$$

We are interested in updating the first column $L_k^{-1}e_1$ of the inverse of the lower triangular matrix L_k . It can be expressed with the inverse of L_{k-1} as

$$L_k^{-1}e_1 = \begin{pmatrix} L_{k-1}^{-1}e_1 \\ \tau_k \end{pmatrix},$$

where τ_k is the last element of the first column of the inverse. Introducing this last result in the expression of x_k and noticing that $x_{k-1} = x_0 + \rho_0 P_{n,k-1}\Omega_{k-1}L_{k-1}^{-1}e_1$, we have

$$x_k = x_{k-1} + \rho_0 \omega_k \tau_k p_{k-1}. \quad (8.2)$$

Provided we can easily compute the vectors p_j , we see that the new iterate x_k is directly obtained from the previous one x_{k-1} and p_{k-1} . Solving the linear system with the matrix L_k and e_1 as the right-hand side we obtain

$$\tau_k = (-1)^{k-1} \frac{\rho_k \cdots \rho_2}{\delta_{k-1} \cdots \delta_1}. \quad (8.3)$$

Let us denote the coefficient of p_{k-1} in the relation (8.2) for x_k by γ_{k-1} . Then

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}, \quad \gamma_{k-1} = \rho_0 \omega_k \tau_k.$$

The residual $r_k = b - Ax_k$ is given by the recurrence

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}. \quad (8.4)$$

But the residual is also given by $r_k = -\rho_{k+1}[y^{(k)}]_k v_{k+1}$. Let us compute $[y^{(k)}]_k = \rho_0 T_k^{-1} e_1 = \rho_0 (L_k \Omega_k^{-1} U_k)^{-1} e_1$ by a forward and a backward solve. The last component after the forward solve is $\rho_0 \tau_k$. Then, we have to divide by the bottom right

entry of \bar{U}_k which is δ_k . It yields

$$[y^{(k)}]_k = \frac{\rho_0 \tau_k}{\delta_k} = (-1)^{k-1} \rho_0 \frac{\rho_k \cdots \rho_2}{\delta_k \delta_{k-1} \cdots \delta_1},$$

and

$$r_k = -\frac{\rho_0 \rho_{k+1} \tau_k}{\delta_k} v_{k+1} = \rho_0 \tau_{k+1} v_{k+1}.$$

Moreover,

$$\|r_k\| = \rho_0 |\tau_{k+1}|.$$

Similarly, we have $\tilde{r}_k = -\zeta_{k+1} [\tilde{y}^{(k)}]_k w_{k+1}$. The diagonal entries of the Cholesky-like factorization for \tilde{T}_k are the same as for T_k and we obtain

$$[\tilde{y}^{(k)}]_k = (-1)^{k-1} \rho_0 \frac{\zeta_k \cdots \zeta_2}{\delta_k \delta_{k-1} \cdots \delta_1} = \rho_0 \frac{\tilde{\tau}_k}{\delta_k},$$

where

$$\tilde{\tau}_k = (-1)^{k-1} \frac{\zeta_k \cdots \zeta_2}{\delta_{k-1} \cdots \delta_1}, \quad (8.5)$$

which gives

$$\tilde{r}_k = \rho_0 \tilde{\tau}_{k+1} w_{k+1}.$$

We return to the relationship between $V_{n,k}$ and $AP_{n,k}$,

$$AP_{n,k} = V_{n,k} L_k \Omega_k^{-1} + G_{n,k} U_k^{-1},$$

where $G_{n,k} = \rho_{k+1} v_{k+1} e_k^T$. Writing the last column of this matrix identity, we have

$$Ap_{k-1} - \frac{1}{\omega_k} v_k = \frac{\rho_{k+1}}{\delta_k \omega_k} v_{k+1}. \quad (8.6)$$

Let us check the consistency with the expressions of the residual vectors. We have, with (8.4),

$$\frac{1}{\gamma_{k-1}} r_k = \frac{1}{\gamma_{k-1}} r_{k-1} - Ap_{k-1}.$$

Plugging in the expression $-\frac{\rho_0 \rho_{k+1} \tau_k}{\delta_k}$ for r_k , the coefficient of v_{k+1} in (8.6) is, using (8.3),

$$-\frac{1}{\gamma_{k-1}} \frac{\rho_0 \rho_{k+1} \tau_k}{\delta_k} = -\frac{1}{\rho_0 \omega_k \tau_k} \frac{\rho_0 \rho_{k+1} \tau_k}{\delta_k} = -\frac{\rho_{k+1}}{\delta_k \omega_k},$$

which is correct. The coefficient of v_k is

$$-\frac{1}{\gamma_{k-1}} \frac{\rho_0 \rho_k \tau_{k-1}}{\delta_{k-1}} = -\frac{1}{\rho_0 \omega_k \tau_k} \frac{\rho_0 \rho_k \tau_{k-1}}{\delta_{k-1}}.$$

But, from (8.3),

$$\frac{\rho_k \tau_{k-1}}{\delta_{k-1}} = -\tau_k,$$

and we obtain

$$-\frac{1}{\gamma_{k-1}} \frac{\rho_0 \rho_k \tau_{k-1}}{\delta_{k-1}} = \frac{1}{\omega_k},$$

which is what we need.

Now we have to find how to compute the vectors p_j . We use the definition $P_{n,k+1} U_{k+1} = V_{n,k+1}$. Let us write, using the definition (8.1) for U_{k+1} , the last column of this relation,

$$v_{k+1} = \omega_{k+1} \delta_{k+1} p_k + \omega_k \beta_{k+1} p_{k-1}.$$

Using $v_{k+1} = r_k / \rho_0 \tau_{k+1}$ we have

$$p_k = \frac{1}{\rho_0 \tau_{k+1} \omega_{k+1} \delta_{k+1}} r_k - \frac{\omega_k \beta_{k+1}}{\omega_{k+1} \delta_{k+1}} p_{k-1}.$$

It is time to choose ω_{k+1} . We define it such that the coefficient of r_k is equal to 1 to simplify the relation for p_k . It yields

$$\omega_{k+1} = \frac{1}{\rho_0 \tau_{k+1} \delta_{k+1}}.$$

The coefficient of p_{k-1} then is

$$\begin{aligned} \mu_k &= -\frac{\tau_{k+1} \delta_{k+1}}{\tau_k \delta_k} \frac{\beta_{k+1}}{\delta_{k+1}}, \\ &= -\frac{\tau_{k+1} \beta_{k+1}}{\tau_k \delta_k}. \end{aligned}$$

But,

$$\frac{\tau_{k+1}}{\tau_k} = -\frac{\rho_{k+1}}{\delta_k},$$

and

$$\mu_k = \frac{\beta_{k+1} \rho_{k+1}}{\delta_k^2}.$$

The vector p_k is given by

$$p_k = r_k + \mu_k p_{k-1}. \quad (8.7)$$

Let us consider the factorization

$$\tilde{T}_k = \tilde{L}_k \tilde{\Omega}_k^{-1} \tilde{U}_k, \quad (8.8)$$

analog to (8.1), where \tilde{L}_k (resp. \tilde{U}_k) is lower (resp. upper) bidiagonal and $\tilde{\Omega}_k$ is the diagonal matrix with entries $\tilde{\omega}_i = \frac{1}{\rho_0 \tilde{\tau}_i \delta_i}$. \tilde{L}_k also has unit diagonal; its lower bidiagonal entries are ζ_i / δ_{i-1} . We have the matrix relation

$$A^T W_{n,k} = W_{n,k} \tilde{L}_k \tilde{\Omega}_k^{-1} \tilde{U}_k + \tilde{G}_{n,k},$$

where $\tilde{G}_{n,k}$ is zero except for the last column. We multiply this relation to the right by \tilde{U}_k^{-1} to obtain

$$A^T W_{n,k} \tilde{U}_k^{-1} = W_{n,k} \tilde{L}_k \tilde{\Omega}_k^{-1} + \tilde{G}_{n,k} \tilde{U}_k^{-1}.$$

We introduce the matrix $\tilde{P}_{n,k} = W_{n,k} \tilde{U}_k^{-1}$ and we denote the columns of $\tilde{P}_{n,k}$ as $\tilde{p}_0, \tilde{p}_1, \dots, \tilde{p}_{k-1}$. Therefore,

$$A^T \tilde{P}_{n,k} = W_{n,k} \tilde{L}_k \tilde{\Omega}_k^{-1} + \tilde{G}_{n,k} \tilde{U}_k^{-1}.$$

We write the last column of this identity,

$$A^T \tilde{p}_{k-1} - \frac{1}{\tilde{\omega}_k} w_k = \frac{\zeta_{k+1}}{\delta_k \tilde{\omega}_k} w_{k+1}.$$

Using the relation between w_k and $\tilde{r}_k = \rho_0 \tilde{\tau}_{k+1} w_{k+1}$, we obtain

$$A^T \tilde{p}_{k-1} - \frac{1}{\rho_0 \tilde{\tau}_k \tilde{\omega}_k} \tilde{r}_{k-1} = \frac{\zeta_{k+1}}{\delta_k \tilde{\omega}_k \rho_0 \tilde{\tau}_{k+1}} \tilde{r}_k.$$

We observe that the coefficients of \tilde{r}_{k-1} and \tilde{r}_k are the same, because of (8.5). From the definition of $\tilde{\omega}_k$, we have

$$-\rho_0 \tilde{\tau}_k \tilde{\omega}_k = -\frac{1}{\delta_k} = -\gamma_{k-1}. \quad (8.9)$$

The recurrence relation for the residuals \tilde{r}_k is the same as for r_k ,

$$\tilde{r}_k = \tilde{r}_{k-1} - \gamma_{k-1} A^T \tilde{p}_{k-1}.$$

The relation for \tilde{p}_k is obtained from the last column of $\tilde{P}_{n,k+1} \tilde{U}_{k+1} = W_{n,k+1}$. We can do the same reasoning as for p_k , and we find a coefficient before \tilde{p}_{k+1} , which is

$$\frac{\zeta_{k+1} \xi_{k+1}}{\delta_k^2},$$

but, because we assume we use the variant with unit basis vectors, $\zeta_{k+1} \xi_{k+1} = \beta_{k+1} \rho_{k+1}$ (see Section 4.3) and we have

$$\tilde{p}_k = \tilde{r}_k + \mu_k \tilde{p}_{k-1}.$$

Let us write the relations we have obtained so far in matrix form assuming there is no termination before iteration k . Let

$$R_{n,k+1} = (r_0 \ r_1 \ \cdots \ r_k), \quad P_{n,k+1} = (p_0 \ p_1 \ \cdots \ p_k),$$

and similar definitions for $\tilde{R}_{n,k+1}$ and $\tilde{P}_{n,k+1}$. Assume, as it is usual that $p_0 = r_0$. We use the relation $r_j - r_{j-1} = -\gamma_{j-1} A p_{j-1}$ for $j = 1, \dots, k$ to obtain

$$R_{n,k+1} \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & -1 \\ & & & & 1 \end{pmatrix} = [r_0, \ -AP_{n,k}] \begin{pmatrix} 1 & & & \\ & \gamma_0 & & \\ & & \ddots & \\ & & & \gamma_{k-1} \end{pmatrix}.$$

Let B_k be the upper bidiagonal matrix of order $k+1$ on the left-hand side. Its inverse is an upper triangular matrix with all the entries of the upper part equal to 1. Let Γ_k be the diagonal matrix on the right-hand side. Then,

$$R_{n,k+1} = [r_0, \ -AP_{n,k}] \Gamma_k B_k^{-1}. \quad (8.10)$$

Using the relations $p_j - \mu_j p_{j-1} = r_j$ for $j = 1, \dots, k$ we obtain

$$P_{n,k+1} \begin{pmatrix} 1 & -\mu_1 & & & \\ & 1 & -\mu_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & -\mu_k \\ & & & & 1 \end{pmatrix} = R_{n,k+1}.$$

Let M_{k+1} be the upper triangular matrix on the left-hand side. Its inverse is the matrix

$$M_{k+1}^{-1} = \begin{pmatrix} 1 & \mu_1 & \mu_1 \mu_2 & \cdots & \cdots & \mu_1 & \cdots & \mu_k \\ & 1 & \mu_2 & \cdots & \cdots & \mu_2 & \cdots & \mu_k \\ & & 1 & \ddots & \ddots & & \vdots & \\ & & & \ddots & \ddots & \mu_{k-1} \mu_k & & \\ & & & & 1 & \mu_k & & \\ & & & & & 1 & & \end{pmatrix},$$

and

$$P_{n,k+1} = R_{n,k+1} M_{k+1}^{-1}. \quad (8.11)$$

From this last relation we see that the vectors r_j and p_j , $j = 0, \dots, k$ span the same subspace, that is, $\mathcal{K}_{k+1}(A, r_0)$.

We obtain similar relations for $\tilde{R}_{n,k+1}$ and $\tilde{P}_{n,k+1}$ but

$$\tilde{R}_{n,k+1} = [r_0, -A^T \tilde{P}_{n,k}] \Gamma_k B_k^{-1}. \quad (8.12)$$

The coupling between both sets of relations is only in the coefficients γ_j and μ_j . From relation (8.11) and $\tilde{R}_{n,k+1}^T R_{n,k+1} = \Upsilon_{k+1}$, Υ_{k+1} being a diagonal matrix of order $k+1$ (as the residual vectors are proportional to the basis vectors, which are biorthogonal), we have

$$\tilde{R}_{n,k+1}^T P_{n,k+1} = \Upsilon_{k+1} M_{k+1}^{-1} = R_{n,k+1}^T \tilde{P}_{n,k+1}.$$

It shows that $\tilde{R}_{n,k+1}^T P_{n,k+1}$ and $R_{n,k+1}^T \tilde{P}_{n,k+1}$ are both upper triangular. Hence,

$$(\tilde{r}_i, p_j) = (r_i, \tilde{p}_j) = 0, \quad i > j.$$

Moreover, since the diagonal entries of M_{k+1}^{-1} are equal to 1,

$$(\tilde{r}_i, p_i) = (r_i, \tilde{p}_i) = (\tilde{r}_i, r_i), \quad i = 1, \dots, k+1. \quad (8.13)$$

From (8.10) we have

$$\tilde{P}_{n,k+1}^T R_{n,k+1} B_k \Gamma_k^{-1} = [\tilde{P}_{n,k+1}^T r_0, -\tilde{P}_{n,k+1}^T A P_{n,k}].$$

Since $\tilde{P}_{n,k+1}^T R_{n,k+1}$ is lower triangular, the matrix $\tilde{P}_{n,k+1}^T R_{n,k+1} B_k \Gamma_k^{-1}$ is lower Hessenberg. This shows that the submatrix with rows $1, \dots, k$ and columns $2, \dots, k+1$ which is $-\tilde{P}_{n,k}^T A P_{n,k}$, is lower triangular. Similarly, from (8.12), we have

$$P_{n,k+1}^T \tilde{R}_{n,k+1} B_k \Gamma_k^{-1} = [P_{n,k+1}^T r_0, -P_{n,k+1}^T A^T \tilde{P}_{n,k}],$$

and $P_{n,k} A^T \tilde{P}_{n,k}$ is also lower triangular. The matrix $\tilde{P}_{n,k}^T A P_{n,k}$ being both lower and upper triangular is diagonal. It shows that the vectors p_j and \tilde{p}_j are A -biorthogonal or biconjugate, that is, $(\tilde{p}_i, A p_j) = 0$, $i \neq j$. Hence, the name of the method.

Now, we would like to find expressions for the coefficients involving only the vectors computed in the algorithm and not the entries of the matrices T_k and \tilde{T}_k . We multiply relation (8.4) by \tilde{p}_{k-1}^T ,

$$0 = (\tilde{p}_{k-1}, r_k) = (\tilde{p}_{k-1}, r_{k-1}) - \gamma_{k-1} (\tilde{p}_{k-1}, A p_{k-1}).$$

It yields, using (8.13),

$$\gamma_{k-1} = \frac{(\tilde{p}_{k-1}, r_{k-1})}{(\tilde{p}_{k-1}, A p_{k-1})} = \frac{(\tilde{r}_{k-1}, r_{k-1})}{(\tilde{p}_{k-1}, A p_{k-1})}. \quad (8.14)$$

We multiply relation (8.7) by \tilde{r}_{k-1}^T ,

$$(\tilde{r}_{k-1}, p_k) = (\tilde{r}_{k-1}, r_k) + \mu_k (\tilde{r}_{k-1}, p_{k-1}) = 0 + \mu_k (\tilde{r}_{k-1}, r_{k-1}).$$

Now, we have

$$\begin{aligned} (\tilde{r}_k, r_k) &= (\tilde{r}_k, p_k), \\ &= (\tilde{r}_{k-1}, p_k) - \gamma_{k-1} (A^T \tilde{p}_{k-1}, p_k), \\ &= (\tilde{r}_{k-1}, p_k) - \gamma_{k-1} (\tilde{p}_{k-1}, A p_k), \\ &= (\tilde{r}_{k-1}, p_k). \end{aligned}$$

It yields

$$\mu_k = \frac{(\tilde{r}_k, r_k)}{(\tilde{r}_{k-1}, r_{k-1})}. \quad (8.15)$$

The method we have just described is what is usually called the Biconjugate Gradient method (BiCG) as it was proposed in [355] as a generalization of the Conjugate Gradient method. A code implementing this method is given below. However, we note that there are many other ways to scale the vectors; for details, see [482].

```
function [x,ni,resn] = BiCG(A,b,x0,epsi,nitmax);
%
nb = norm(b);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resn(1) = norm(r);
rt = r;
p = r;
pt = rt;
At = A';
rrt = rt' * r;
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    Ap = A * p; % matrix vector product
    Atp = At * pt; % matrix vector product with the transpose
    gamma = rrt / (pt' * Ap);
    x = x + gamma * p;
    r = r - gamma * Ap;
    rt = rt - gamma * Atp;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    if nresidu < (epsi * nb) || ni >= nitmax
```

```

break % get out of the k loop
end % if nresidu
rrtn = rt' * r;
mu = rrtn / rrt;
rrt = rrtn;
p = r + mu * p;
pt = rt + mu * pt;
end % for k

```

The interest of this method is that it is very simple to code. Moreover, when A is symmetric, it is the same as the CG method. However, this algorithm breaks down if $(\tilde{r}_k, r_k) = 0$ and/or $(\tilde{p}_k, Ap_k) = 0$. The mathematical properties of the method that have been obtained during the previous derivation are summarized in the following theorem.

Theorem 8.1 *Assuming that $p_0 = r_0$, $\tilde{p}_0 = \tilde{r}_0$ and that there is no breakdown until at least iteration $k \leq n$, the vectors computed by BiCG satisfy the following properties,*

- the vectors r_0, r_1, \dots, r_k and p_0, p_1, \dots, p_k span $\mathcal{K}_{k+1}(A, r_0)$,
- the vectors $\tilde{r}_0, \tilde{r}_1, \dots, \tilde{r}_k$ and $\tilde{p}_0, \tilde{p}_1, \dots, \tilde{p}_k$ span $\mathcal{K}_{k+1}(A^T, \tilde{r}_0)$,
- $(\tilde{r}_i, r_j) = 0$, $i \neq j$,
- $(\tilde{p}_i, Ap_j) = 0$, $i \neq j$,
- $(\tilde{r}_i, p_j) = (r_i, \tilde{p}_j) = 0$, $i > j$.

We can also obtain the coefficients $\alpha_j, \beta_j, \rho_j, \zeta_j, \xi_j, \delta_j$ of the Lanczos recurrences from the BiCG coefficients. This is useful if we want to compute approximations of the eigenvalues of A . We have already seen (see (8.9)) that we have

$$\mu_k = \frac{\beta_{k+1}\rho_{k+1}}{\delta_k^2}, \quad \gamma_{k-1} = \frac{1}{\delta_k}.$$

The first relation gives us the product

$$\beta_{k+1}\rho_{k+1} = \mu_k \delta_k^2 = \frac{\mu_k}{\gamma_{k-1}^2}.$$

From the δ_j recurrence described in Section 2.8, we see that

$$\delta_{k+1} = \alpha_{k+1} - \frac{\beta_{k+1}\rho_{k+1}}{\delta_k} = \alpha_{k+1} - \mu_k \delta_k.$$

It yields

$$\alpha_{k+1} = \delta_{k+1} + \mu_k \delta_k = \frac{1}{\gamma_k} + \frac{\mu_k}{\gamma_{k-1}}.$$

So far, we only know the product of the non-diagonal entries of T_k . But, since in our variant the ρ_j 's are norms, they are real and positive. We also know that

$$\frac{\|r_k\|}{\|r_{k-1}\|} = \frac{|\tau_{k+1}|}{|\tau_k|} = \frac{\rho_{k+1}}{|\delta_k|} = \rho_{k+1} |\gamma_{k-1}|.$$

Hence,

$$\rho_{k+1} = \frac{1}{|\gamma_{k-1}|} \frac{\|r_k\|}{\|r_{k-1}\|}, \quad \beta_{k+1} = \frac{\mu_k}{|\gamma_{k-1}|} \frac{\|r_{k-1}\|}{\|r_k\|}.$$

We can proceed in the same way for the entries of \tilde{T}_k , and we obtain

$$\zeta_{k+1} = \frac{1}{|\gamma_{k-1}|} \frac{\|\tilde{r}_k\|}{\|\tilde{r}_{k-1}\|}, \quad \xi_{k+1} = \frac{\mu_k}{|\gamma_{k-1}|} \frac{\|\tilde{r}_{k-1}\|}{\|\tilde{r}_k\|}.$$

We derived BiCG from the Lanczos biorthogonalization process but this method can also be derived from scratch from the conditions

$$x_k \in x_0 + \mathcal{K}_k(A, r_0), \quad r_k \perp \mathcal{K}_k(A^T, \tilde{r}_0),$$

$$\tilde{r}_k \in \tilde{r}_0 + A^T \mathcal{K}_k(A^T, \tilde{r}_0), \quad \tilde{r}_k \perp \mathcal{K}(A, r_0).$$

This was done in [912].

Above we chose $\tilde{r}_0 = r_0$ but this is not mandatory. The initial vector \tilde{r}_0 is often called the shadow vector. Its choice for the Lanczos biorthogonalization process was considered in [454, 910]. In [910] it is shown that for a given linear system with a given starting vector, one can choose the shadow vector in such a way to have the same residual vector as another Krylov method at one given iterate k . Defining the shadow vector in another way, one can obtain the same residual vectors every 2^ℓ iterations with $\ell = 0, 1, \dots, k$. However, there is no mathematical study in the literature about the influence of the shadow vector on the convergence curve. The two most common choices are $\tilde{r}_0 = r_0$ and \tilde{r}_0 random. The latter is sometimes used to try to avoid breakdowns. Variants of BiCG were considered in [136].

The positive aspects of BiCG and more generally of the Lanczos methods are that we have short recurrences; it is not necessary to store too many vectors and the number of operations is constant per iteration. Moreover, the algorithm is very easy to code. The negative aspects are that there are two matrix–vector products per iteration. One product is with the transposed (or adjoint) matrix and this may not be easy to do, particularly on parallel computers. Moreover, there may be breakdowns or near-breakdowns that can be eventually cured or not. Finally, on many examples, the BiCG residual norms have a very oscillatory behavior. In finite precision computations,

this may have a negative influence on the speed of convergence and the maximum attainable accuracy.

8.3 QMR

The QMR algorithm was proposed in [379]; see also [371–373, 378, 381–383]. Originally it was intended to have a smoother behavior of the residual norms than with BiCG but still using only short recurrences. In its basic form it can be seen as using the general Q-MR framework described in Section 3.1 with the biorthogonal bases generated by the Lanczos nonsymmetric process. So, instead of minimizing the residual norm as in GMRES, we minimize the quasi-residual norm, that is,

$$\min_y \| \|r_0\| e_1 - \underline{T}_k y \| . \quad (8.16)$$

The matrix \underline{T}_k is a $(k+1) \times k$ tridiagonal matrix. The least squares problem (8.16) is solved using Givens rotations. Since \underline{T}_k is tridiagonal, the upper triangular matrix that is obtained when applying the Givens rotations has only three nonzero diagonals, the main diagonal and the two diagonals next to it. The iterates are defined as

$$x_k = x_0 + V_{n,k} y^{(k)},$$

where $y^{(k)}$ is the solution of the least squares problem. However, we can do better than that since it turns out that the iterates can be computed with short recurrences. Applying the Givens rotations to \underline{T}_k and using the same notation as in [379], the matrix that is obtained is

$$\begin{pmatrix} R_k \\ 0 \dots 0 \end{pmatrix}, \quad R_k = \begin{pmatrix} \delta_1 & \epsilon_2 & \theta_3 & 0 & \cdots & 0 \\ & \delta_2 & \epsilon_3 & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \ddots & 0 \\ & & & \delta_{k-2} & \epsilon_{k-1} & \theta_k \\ & & & & \delta_{k-1} & \epsilon_k \\ & & & & & \delta_k \end{pmatrix}.$$

The solution of the least squares problem (8.16) is obtained by solving $R_k y^{(k)} = z^{(k)}$ where the right-hand side $z^{(k)}$ is computed by applying the Givens rotations to $\|r_0\| e_1$. The last rotation which is used to zero the entry $(k+1, k)$ of \underline{T}_k modifies only the last component of the right-hand side because we use only the k first components. Therefore,

$$z^{(k)} = \begin{pmatrix} z^{(k-1)} \\ [z^{(k)}]_k \end{pmatrix}.$$

The solution can be written as

$$y^{(k)} = \begin{pmatrix} y^{(k-1)} \\ 0 \end{pmatrix} + [z^{(k)}]_k R_k^{-1} e_k,$$

which yields

$$x = x_0 + V_{n,k} y^{(k)} = x_0 + V_{n,k-1} y^{(k-1)} + [z^{(k)}]_k V_{n,k} R_k^{-1} e_k = x_{k-1} + [z^{(k)}]_k V_{n,k} R_k^{-1} e_k.$$

Let $P_{n,k} = (p_1 \cdots p_k) = V_{n,k} R_k^{-1}$. Note that this notation is different from what we used for BiCG. We are interested in the last column of $P_{n,k}$. Writing $P_{n,k} R_k = V_{n,k}$, we obtain

$$p_k = \frac{1}{\delta_k} (v_k - \epsilon_k p_{k-1} - \theta_k p_{k-2}),$$

and

$$x_k = x_{k-1} + [z^{(k)}]_k p_k.$$

Therefore, like in BiCG, it is not necessary to keep all the previous basis vectors. Note that introducing the vectors p_k is of interest only because R_k is a banded matrix. We observe that the same technique can be applied to the Q-OR Lanczos algorithm of Section 8.1.

From Chapter 3 we know that there exist simple relations between the iterates of Q-OR and Q-MR methods using the same basis. Let us denote the quantities related to BiCG (resp. QMR) by a superscript B (resp. Q). Even though they are not computed in this way the iterates are mathematically defined as

$$x_k^B = x_0 + V_{n,k} y^B, \quad x_k^Q = x_0 + V_{n,k} y^Q.$$

From Theorem 3.2 and remarks thereafter in Section 3.2 we have (when T_k is nonsingular),

$$y^Q = y^B - \gamma [y^B]_k T_k^{-1} T_k^{-T} e_k, \quad \gamma = \frac{t_{k+1,k}^2}{1 + t_{k+1,k}^2 \|T_k^{-T} e_k\|^2} = t_{k+1,k}^2 c_k^2,$$

where c_k is the cosine in the Givens rotation used to annihilate $t_{k+1,k}$. Let R_k^B (resp. R_k) be the upper triangular matrix obtained before (resp. after) applying the rotation used to annihilate $t_{k+1,k}$. They differ only by the (k, k) entries that we denote by $r_{k,k}^B$ and $r_{k,k}^Q$. Similarly, the right-hand sides obtained by applying the rotations to $\|r_0\|e_1$ differ only in the last components that we denote by τ^B and $\tau^Q = [z^{(k)}]_k$ for simplicity. They are related by $\tau^Q = c_k \tau^B$. Since we reduce T_k to upper triangular form by unitary transformations we have $T_k^{-1} T_k^{-T} = (R_k^B)^{-1} (R_k^B)^{-T}$. Moreover, we have $(R_k^B)^{-T} e_k = e_k / r_{k,k}^B$. The last component of y^B is equal to $\tau^B / r_{k,k}^B$. Therefore,

$$x_k^Q = x_k^B - \frac{t_{k+1,k}^2 c_k^2}{(r_{k,k}^B)^2} \tau^B V_{n,k} (R_k^B)^{-1} e_k.$$

Using the definition of the sine and cosine, s_k and c_k , of the last rotation one can show that

$$\frac{t_{k+1,k}^2}{(r_{k,k}^B)^2} = \frac{|s_k|^2}{c_k^2}.$$

Defining $P_{n,k}^B = (p_0^B \cdots p_{k-1}^B) = V_{n,k} (R_k^B)^{-1}$, we have

$$x_k^Q = x_k^B - \frac{|s_k|^2}{c_k^2} \tau^Q p_{k-1}^B.$$

Looking at the formulas to recursively compute p_k and p_{k-1}^B , we observe that $p_{k-1}^B = p_k/c_k$. Hence, using only quantities which are computed in the QMR algorithm, we obtain

$$x_k^B = x_k^Q + \frac{|s_k|^2}{c_k^2} \tau^Q p_k = x_k^Q + \frac{|s_k|^2}{c_k^2} [z^{(k)}]_k p_k.$$

This formula was proved differently in [379]. It shows how we can recover the BiCG iterates from the QMR iterates if this is of any interest. We observe that the BiCG iterates are not defined when $c_k = 0$.

A code for the three-term recurrence QMR method is following. In this code, we use $w^T v$ as the dot product of vectors w and v even if these vectors are complex. It allows this code to work also for complex data without having to conjugate some coefficients of the recurrences.

```
function [x,ni,resn] = qmr_3t(A,b,x0,epsi,nitmax);
%
% 3-term recurrences, without look-ahead
n = size(A,1);
resn = zeros(1,nitmax);
trid = zeros(nitmax+1,nitmax+1);
rot = zeros(2,nitmax);
v = b;
nb = norm(b);
nr0 = norm(v);
v = v / nr0;
w = v;
wv = transpose(w) * v;
wv_old = wv;
v1 = zeros(n,1);
w1 = v1;
ss = zeros(n,1);
```

```
ss(1) = nr0;
At = transpose(A);
r = b - A * x0;
x = x0;
resn(1) = norm(r);
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1;
    Av = A * v;
    Aw = At * w;
    alpha = (transpose(w) * Av) / wv;
    beta = (transpose(w1) * Av) / wv_old;
    vt = Av - alpha * v - beta * v1;
    gamma = (transpose(v1) * Aw) / wv_old;
    wt = Aw - alpha * w - gamma * w1;
    rho = norm(vt);
    zeta = norm(wt);
    v1 = v;
    w1 = w;
    v = vt / rho;
    w = wt / zeta;
    wv_new = transpose(w) * v;
    wv_old = wv;
    wv = wv_new;
    trid(k,k) = alpha;
    trid(k+1,k) = rho;
    if k > 1
        trid(k-1,k) = beta;
    end % if
    % apply the two preceding rotations to column k
    for l = max(k-2,1):k-1
        t1 = trid(l,k);
        t2 = trid(l+1,k);
        trid(l,k) = rot(1,l) * t1 - rot(2,l) * t2;
        trid(l+1,k) = rot(2,l) * t1 + rot(1,l) * t2;
    end % for l
    % compute store and apply rotation to zero the entry (k+1,k)
    tk = trid(k,k);
    cs = sqrt(trid(k+1,k)^2 + tk^2);
    rot(1,k) = tk / cs; % cosine
    rot(2,k) = -trid(k+1,k) / cs; % sine
    trid(k,k) = cs;
    c = ss(k) / cs;
    ss(k) = tk * c;
```

```

ss(k+1) = -trid(k+1,k) * c;
trid(k+1,k) = 0;
switch k
case 1
p = v1 / trid(1,1);
p1 = p;
case 2
p_old = p;
p = (v1 - trid(1,2) * p1) / trid(2,2);
p1 = p_old;
otherwise
p_old = p;
p = (v1 - trid(k-1,k) * p - trid(k-2,k) * p1) / trid(k,k);
p1 = p_old;
end % switch k
x = x + ss(k) * p;
r = b - A * x;
nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
break % get out of the k loop
end % if nresidu
end % for k

```

However, as in BiCG, we can also use coupled two-term recurrences instead of the three-term recurrences. Two-term recurrences are known to be more stable than three-term recurrences; see [493]. Unfortunately, using them, we introduce new possibilities of breakdowns. A code for the two-term recurrence QMR method is following.

```

function [x,ni,resn] = qmr_2t(A,b,x0,epsi,nitmax);
%
% 2-term recurrences, without look-ahead
n = size(A,1);
resn = zeros(1,nitmax);
v = b;
nb = norm(b);
rho = nb;
v = v / rho;
w = v;
p = zeros(n,1);
q = p;
d = p;
At = transpose(A);
r = b - A * x0;
x = x0;

```

```

resn(1) = norm(r);
c1 = 1;
eps = 1;
zeta = 1;
theta = 0;
eta = -1;
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1;
    delta = transpose(w) * v;
    p = v - ((zeta * delta) / eps) * p;
    q = w - ((rho * delta) / eps) * q;
    Ap = A * p;
    Atq = At * q;
    eps = transpose(q) * Ap;
    beta = eps / delta;
    v = Ap - beta * v;
    w = Atq - beta * w;
    rho1 = rho;
    rho = norm(v);
    zeta = norm(w);
    theta1 = theta;
    theta = rho / (c1 * abs(beta));
    c = 1 / sqrt(1 + theta^2);
    eta = -eta * (rho1 * c^2) / (beta * c1^2);
    c1 = c;
    d = eta * p + (theta1 * c)^2 * d;
    v = v / rho;
    w = w / zeta;
    x = x + d;
    r = b - A * x;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    if nresidu < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nresidu
end % for k

```

8.4 Breakdowns and look-ahead in BiCG/QMR

The nonsymmetric Lanczos process breaks down at iteration k if $(v_{k-1}, w_{k-1}) = 0$ with $v_{k-1} \neq 0$, $w_{k-1} \neq 0$ which causes β_k and ξ_k to be infinite. We can also have $v_k = 0$ and/or $w_k = 0$ for some iteration k . Some of these events can happen before

the complete termination of the algorithm which must occur at iteration $d(A, r_0)$, the grade of r_0 with respect to A . The names of the different kinds of breakdown is, unfortunately, not fully unified in the literature.

The case $v_k = 0$ and $w_k = 0$ is called full termination. It yields

$$AV_{n,k} = V_{n,k}T_k, \quad A^TW_{n,k} = W_{n,k}\tilde{T}_k,$$

which means that $\mathcal{K}_k(A, r_0)$ (resp. $\mathcal{K}_k(A^T, \tilde{r}_0)$) is an invariant subspace of A (resp. A^T). The eigenvalues of T_k are a subset of the eigenvalues of A .

The case $v_k = 0$ or $w_k = 0$ (but not both) is called a one-sided termination in [482]. Having $v_k = 0$ is a nice situation since, mathematically, we have found the exact solution of the linear system. Unless we solve a parallel system with A^T , having $w_k = 0$ is a bad situation. Generally one restarts the algorithm with a different shadow vector or a different initial vector x_0 .

Having $(v_{k-1}, w_{k-1}) = 0$ with $v_{k-1} \neq 0$, $w_{k-1} \neq 0$ is sometimes called a serious breakdown. One possibility is to restart the algorithm. Another possibility is to use so-called *look-ahead* techniques to cure these breakdowns. Roughly speaking, curing a serious breakdown at iteration k requires that $(w_k, A^m v_k) \neq 0$ for some m . If such an index does not exist before we reach the maximum dimension of the Krylov subspaces, the breakdown is said to be incurable. It was shown in [786] that, for a given linear system, there exist r_0 and \tilde{r}_0 such that there is no serious breakdown or early termination. Unfortunately, those pairs of vectors are generally not known.

Several types of breakdown are also possible in the BiCG method, most of them originating from a breakdown in the underlying nonsymmetric Lanczos process. We may have $(r_j, \tilde{r}_j) = 0$ and/or $(\tilde{p}_j, Ap_j) = 0$.

One can easily construct small toy problems where exact serious breakdowns occur in the nonsymmetric Lanczos process and therefore in BiCG and QMR. However, what is likely to happen in practice are near-breakdowns where the dot products are not zero but tiny, causing numerical problems later on. These are the situations that must be handled in practical codes.

Look-ahead variants of the Lanczos nonsymmetric process allow to skip over the iterations where we have a breakdown, that is, a division by a zero (or a tiny) value in the standard algorithm; see [478, 481]. Deriving look-ahead algorithms is quite technical. In this section we just give one example following the papers [371–373, 378, 379, 381–383]. Possible (near-)breakdowns are avoided by relaxing the usual biorthogonality conditions to block biorthogonality. Hence, the matrix T_k will not be tridiagonal any longer, but block tridiagonal with square diagonal blocks. The basis vectors now satisfy the relation $W_{n,k}^T V_{n,k} = D_k$ where D_k is a block diagonal matrix. The vectors that satisfy the usual biorthogonality condition

$$w_i^T v_j = w_j^T v_i = 0, \quad j < i,$$

are called regular vectors. Let

$$1 = k_1 < k_2 < \cdots < k_\ell \leq k < k_{\ell+1},$$

be the indices of the regular vectors. They are called regular indices. All the other basis vectors are called inner vectors. The regular indices define the partitioning of the basis vectors into blocks, a block being defined by the first vector of the block which is regular, all the other vectors being the inner vectors following the first vector. We define

$$V_{n,(j)} = (v_{k_j} \ v_{k_j+1} \ \cdots \ v_{k_{j+1}-1}),$$

and the corresponding matrices for the vectors w_j . A block $V_{n,(j)}$ can be made of just one vector, if two successive vectors are regular. With this notation, the matrices of the basis vectors can be written with regular indices as

$$V_{n,k} = (V_{n,(1)} \ V_{n,(2)} \ \cdots \ V_{n,(\ell)}), \quad W_{n,k} = (W_{n,(1)} \ W_{n,(2)} \ \cdots \ W_{n,(\ell)}).$$

The matrix D_k is written in block form as

$$D_k = \text{diag}(D_{(1)} \ D_{(2)} \ \cdots \ D_{(\ell)}), \quad D_{(j)} = W_{n,(j)}^T V_{n,(j)}.$$

When k is not a regular index, we will enforce the conditions

$$v_k \perp \mathcal{K}_{k_\ell}(A^T, \tilde{r}_0), \quad w_k \perp \mathcal{K}_{k_\ell}(A, r_0), \text{ if } k_\ell \leq k < k_{\ell+1}.$$

To derive the recurrences for the inner vectors we have also to consider incomplete blocks,

$$V_{n,(\ell)}^{(k)} = (v_{k_\ell} \ v_{k_\ell+1} \ \cdots \ v_k), \quad W_{n,(\ell)}^{(k)} = (w_{k_\ell} \ w_{k_\ell+1} \ \cdots \ w_k),$$

and

$$D_{(\ell)}^{(k)} = [W_{n,(\ell)}^{(k)}]^T V_{n,(\ell)}^{(k)}.$$

The algorithm is constructed in such a way that the matrices $D_{(\ell)}$ are nonsingular. In matrix notation, we have

$$AV_{n,k} = V_{n,k+1} T_k, \quad A^T W_{n,k} = W_{n,k+1} \tilde{T}_k.$$

The matrices T_k and \tilde{T}_k are block tridiagonal. The recurrences for computing the basis vectors can be written as

$$\begin{aligned} \rho_k v_{k+1} &= Av_k - V_{n,(k)}^{(k)} a_k - V_{n,(k-1)} b_k, \\ \zeta_k w_{k+1} &= A^T w_k - W_{n,(k)}^{(k)} \tilde{a}_k - W_{n,(k-1)} \tilde{b}_k. \end{aligned}$$

The vector b_k is computed to enforce the biorthogonality with the previous block and, in a regular step, a_k is computed to enforce the biorthogonality with the last completed block. It yields

$$b_k = D_{(\ell-1)}^{-1} W_{n,(\ell-1)}^T A v_k, \quad \tilde{b}_k = D_{(\ell-1)}^{-T} V_{n,(\ell-1)}^T A^T w_k, \quad \text{if } k_\ell < k \leq k_{\ell+1},$$

$$a_k = D_{(\ell)}^{-1} W_{n,(\ell)}^T A v_k, \quad \tilde{a}_k = D_{(\ell)}^{-T} V_{n,(\ell)}^T A^T w_k, \quad \text{if } k = k_{\ell+1}.$$

Inner vectors in a block are computed with three-term recurrences,

$$\begin{aligned} v_{k+1} &= A v_k - \xi_k v_k - \eta_k v_{k-1}, \\ w_{k+1} &= A^T w_k - \xi_k w_k - \eta_k w_{k-1}, \end{aligned}$$

where ξ_k and η_k are a priori chosen coefficients with $\eta_{k_\ell} = 0$. Implementation details are given in the series of papers cited at the beginning of Section 8.3.

The look-ahead algorithm is not complete without a *look-ahead strategy*, that is, when to start a new block and when to end it. The main goal is that we would like to avoid singular or badly conditioned blocks $D_{(j)}$. Moreover, it is desirable that the algorithm performs standard Lanczos steps most of the time. It seems that the strategy could be based on the smallest singular value of the incomplete block matrices $D_{(\ell)}^{(k)}$, but numerical experiments reported in [373] show that this is not the case. It is also important to maintain a strong linear independence of the basis vectors. In addition to the criterion

$$\sigma_{\min}(D_{(\ell)}^{(k)}) \geq \varepsilon,$$

where ε is the machine epsilon, four other criteria were defined in [373]. They are based on the 1-norm of the vector coefficients,

$$\sum_{j=k_{\ell-1}}^{k_\ell-1} |(b_k)_j| \leq n(A), \quad \sum_{j=k_\ell}^k |(a_k)_j| \leq n(A), \quad (8.17)$$

and

$$\sum_{j=k_{\ell-1}}^{k_\ell-1} |(\tilde{b}_k)_j| \leq n(A), \quad \sum_{j=k_\ell}^k |(\tilde{a}_k)_j| \leq n(A). \quad (8.18)$$

If all these criteria are satisfied v_{k+1} and w_{k+1} are build as regular vectors. In these tests, $n(A)$ is a factor based on the norm of A . In practice it is set to 10 times an approximation of $\|A\|$. Such an approximation can be obtained incrementally during the QMR iterations, see [373].

In [382] a QMR algorithm with look-ahead using coupled two-term recurrences is described. Here the situation is even more complicated than with the Lanczos three-term recurrence algorithm since there may be (near-)breakdowns for the second sequence of vectors p_j, q_j . The breakdowns may not occur at the same iterations in the two sequences. However, numerical experiments show that the two-term algorithm is sometimes more stable than the three-term one; see [382]. For a review of these algorithms, see [482].

The codes for these QMR algorithms are too long and complicated to be shown here. QMRPACK, a Fortran 77 package implementing them, is still available in Netlib (www.netlib.org). However, there are some features in the code which are slightly different from the description of the algorithms in the papers. For instance, for a regular vector, only the exact value zero is tested for $w_i^T v_i$ which is unlikely to happen. Therefore, the decision for the next vector to be regular or inner is only based on criteria (8.17) and (8.18). But then the criterion on the smallest singular value is used to determine the length of the jump. There is also an upper bound on the length of the jump with recovery procedures when it is reached which are not described in the published papers.

In [68, 69] algorithms were proposed to treat some of the BiCG breakdowns. We have seen in Section 8.2 that we derive BiCG from the nonsymmetric Lanczos process by doing an LU factorization of the tridiagonal matrix T_k . However, this matrix may be singular or nearly singular. In [68] it is proved that T_k and T_{k-1} cannot both be singular, see also Theorem 8.3 later in this chapter. Hence, it was proposed to factor the tridiagonal matrix as $T_k = L_k D_k U_k$ where L_k is unit lower block bidiagonal, U_k is unit upper block bidiagonal and D_k is block diagonal with 1×1 and 2×2 diagonal blocks. In a sense, it is a look-ahead algorithm with jumps of size at most 2. The resulting algorithm was called the composite step biconjugate gradient algorithm (CSBCG). Criteria for deciding when to have 1×1 or 2×2 blocks are given in [68] as well as some variants of the algorithm. However, this technique does not cure the breakdowns arising from the underlying Lanczos algorithm.

Another proposal was done in [538]. The idea is the following. If there is a breakdown at iteration j in the QMR underlying Lanczos process, the matrix A is replaced by a rank-1 modification $\tilde{A} = A + uv^T$ where u and v are chosen in such a way that the Lanczos process applied on \tilde{A} has no breakdown at step j , and that the already computed numbers and vectors remain unchanged. It is assumed that $x_0 = 0$. In case of breakdown the vector Av_j cannot be orthogonalized against w_j and $A^T w_j$ against v_j . However, $A^T w_j$ can be orthogonalized against v_1, \dots, v_{j-1} using short recurrences. It yields a unit norm vector \hat{w}_{j+1} for which $A^T \hat{w}_{j+1}$ can also be orthogonalized. Continuing in this way we can obtain a unit vector \hat{w}_{j+m-1} such that $|\hat{w}_{j+m-1}^T Av_j|$ is large enough. Let $\hat{w}_{j+m} = A^T \hat{w}_{j+m-1}$. Then, $w_j, \hat{w}_{j+1}, \dots, \hat{w}_{j+m-1}$ are orthogonal to v_1, \dots, v_{j-1} and \hat{w}_{j+m} is orthogonal to v_1, \dots, v_{j-2} . Moreover, $\hat{w}_{j+m}^T A^{-1} v_{j-1} = 0$. The new matrix is defined as

$$\tilde{A} = A + \lambda_j v_{j-1} \hat{w}_{j+m}^T.$$

With λ_j chosen ‘large enough’, one can show that the Lanczos process on \tilde{A} can continue without breakdown and that \tilde{A} is nonsingular. We note that with $x_0 = 0$ we have the same first residual for A and \tilde{A} . We look for a solution of $\tilde{A}\tilde{x} = b$. But the Sherman-Morrison formula shows that $\tilde{x} = x$. Choices for the parameter λ_j are proposed in [538]. Note that this strategy can be used for both BiCG and QMR.

8.5 Comparison of QMR and GMRES

To compare the residual norms obtained with GMRES and QMR we can use the general results proved in Section 3.9, in particular Theorem 3.23. It compares the quasi-residual norms of two Q-MR methods and states that

$$\|\tilde{z}_k^M\| \leq \|W_k\| \|z_k^M\|, \quad (8.19)$$

where W_k is the principal matrix of order k of $\tilde{V}^{-1}V$, with \tilde{V} and V being the matrices of the basis vectors of the two methods. Let us assume that we have no early termination in QMR. To avoid confusion of notation, let us denote by V^Q and W^Q the matrices of QMR and by V^G the matrix of GMRES. The matrices V^Q and W^Q satisfy $(W^Q)^T V^Q = D$ where D is a diagonal matrix and V^G is an orthonormal matrix. Since GMRES minimizes the residual norm over the Krylov subspace we have

$$\|r_k^G\| \leq \|r_k^Q\| \leq \|V_{n,k+1}^Q\| \|z_k^Q\| \leq \sqrt{k+1} \|z_k^Q\|.$$

However, using relation (8.19) and $\|z_k^G\| = \|r_k^G\|$, we have

$$\|r_k^G\| \leq \|(V_{n,k}^G)^T V_{n,k}^Q\| \|z_k^Q\| \leq \|V_{n,k}^Q\| \|z_k^Q\| \leq \sqrt{k} \|z_k^Q\|.$$

Conversely, if we want to bound the QMR quasi-residual norm using (8.19) we have to consider the principal matrix of order k of $(V^Q)^{-1}V^G$. From the biorthogonality property, we have

$$(V^Q)^{-1} = D^{-1}(W^Q)^T.$$

Therefore, the matrix we have to consider is $D_k^{-1}(W_{n,k}^Q)^T V_{n,k}^G$, where the diagonal matrix D_k is the principal submatrix of order k of D . Then,

$$\|z_k^Q\| \leq \|D_k^{-1}(W_{n,k}^Q)^T V_{n,k}^G\| \|r_k^G\| \leq \|D_k^{-1}(W_{n,k}^Q)^T\| \|r_k^G\|.$$

It is interesting to remark that the matrix $D_k^{-1}(W_{n,k}^Q)^T V_{n,k}^G$ is upper triangular, see the discussion after (3.45). Finally, we have

$$\|r_k^G\| \leq \|r_k^Q\| \leq \sqrt{k+1} \|D_k^{-1}(W_{n,k}^Q)^T\| \|r_k^G\|.$$

Since the vectors w_j^Q are of unit norm, the norm of $D_k^{-1}(W_{n,k}^Q)^T$ is bounded by

$$\left(\sum_{j=1}^k \frac{1}{d_{j,j}^2} \right)^{\frac{1}{2}} = \left(\sum_{j=1}^k \frac{1}{(w_j^Q, v_j^Q)^2} \right)^{\frac{1}{2}}.$$

Note that $|(\mathbf{w}_j^Q, \mathbf{v}_j^Q)|$ is the cosine of the angle of the two vectors. If they are orthogonal the bound is infinite, but then we have a breakdown in QMR.

8.6 Prescribed convergence in BiCG and QMR

If we wish to prescribe convergence curves for the BiCG/QMR pair the situation is less straightforward than for FOM/GMRES and Hessenberg/CMRH. A first complication is that the Hessenberg matrix H generated in BiCG/QMR is tridiagonal, which puts additional constraints in the construction process of $H = UCU^{-1}$, see Section 3.5. Second, infinite BiCG residual norms (which correspond to H_k being singular) cannot be prescribed arbitrarily. In fact, we will show that some convergence curves with infinite BiCG residual norms are not admissible. Also, the choice of the matrix V in the parametrization $A = VHV^{-1}$ is a little more delicate.

We note that the problem of constructing matrices with a prescribed BiCG convergence curve was also considered in [244]. This was done using the relations between FOM/GMRES and BiCG/QMR. Also, given the results of BiCG on $Ax = b$, in [244] a matrix B with the same eigenvalues as A and a right-hand side c are constructed such that FOM applied to $Bx = c$ yields the same convergence curve. As we have seen above, in [910] choices of the shadow vector \tilde{r}_0 are presented which lead, for some selected iteration numbers, to the same BiCG residual norms as the residual norms obtained when applying a different Krylov method like, e.g., the GMRES method to the same linear system.

The BiCG and QMR methods use the nonsymmetric Lanczos process to generate a pair of biorthogonal bases. It generates a pair of bases with two three-term recurrences. The first basis is an ascending basis for the Krylov subspaces $\mathcal{K}_k(A, r_0)$, $k \leq n$, represented by the columns v_k of a nonsingular matrix V . The second basis is an ascending basis for the Krylov subspaces $\mathcal{K}_k(A^T, \tilde{r}_0)$ (or $\mathcal{K}_k(A^*, \tilde{r}_0)$), $k \leq n$, represented by the columns w_k of a nonsingular matrix W . The shadow vector \tilde{r}_0 must satisfy $(\tilde{r}_0, r_0) \neq 0$, $d(A^T, \tilde{r}_0) = n$ and is frequently chosen as $\tilde{r}_0 = r_0$; in the following we will consider this choice. The biorthogonality condition $w_i^T v_i \neq 0$ and $w_i^T v_j = 0$ for $i \neq j$ can be written as

$$W^T V = \Phi = \text{diag}(\phi_1, \dots, \phi_n), \quad \phi_k \neq 0, \quad k = 1, \dots, n.$$

Now suppose we have any pair of biorthogonal bases stored in the nonsingular matrices V, W such that V has unit norm columns. It can be generated by applying the nonsymmetric Lanczos process to *any* matrix \hat{A} and starting vector \hat{r}_0 provided the biorthogonalization process runs to completion. Also, suppose we have a tridiagonal matrix of the form $T = UCU^{-1}$ such that the absolute values of the inverses of the entries of the first row of the upper triangular matrix U^{-1} are prescribed BiCG residual norms and C is the companion matrix corresponding to prescribed nonzero eigenvalues. We will discuss the existence of such a matrix T below. Then we can define $A = VTV^{-1}$ and $r_0 = b = Ve_1$ with $x_0 = 0$. It follows that

$$\begin{aligned} K &= (b \ A b \ A^2 b \ \cdots \ A^{n-1} b), \\ &= V (e_1 \ Te_1 \ T^2 e_1 \ \cdots \ T^{n-1} e_1), \\ &= VU. \end{aligned}$$

Moreover, defining $\tilde{T} \equiv \Phi^{-1}T^T\Phi$ and using $V = W^{-T}\Phi$, we have $A^T = W\tilde{T}W^{-1}$ and therefore

$$\begin{aligned} (b \ A^T b \ (A^T)^2 b \ \cdots \ (A^T)^{n-1} b) &= W (e_1 \ \tilde{T} e_1 \ \tilde{T}^2 e_1 \ \cdots \ \tilde{T}^{n-1} e_1), \\ &= W\tilde{U}. \end{aligned}$$

Thus the nonsymmetric Lanczos process applied to A and b generates the biorthogonal bases represented by V and W , A has the desired spectrum and U has the desired first row forcing the prescribed BiCG residual norms.

We have just seen that if we wish to prescribe eigenvalues and BiCG residual norms simultaneously, we have to construct an upper triangular matrix U such that the first row of U^{-1} is prescribed and such that the upper Hessenberg matrix $H = UCU^{-1}$ is tridiagonal with prescribed eigenvalues given by the companion matrix C . This can be seen as an inverse nonsymmetric tridiagonal eigenproblem, with an additional constraint for the change of basis matrix U . We will first solve this problem when we prescribe finite BiCG residual norms only, i.e., the first row of U^{-1} has no zero entries.

We use the results in [581] which state that for a nonderogatory complex matrix, almost every starting vector r_0 (that is, except for a measure zero set of vectors) with shadow vector $\tilde{r}_0 = r_0$ will generate a BiCG process that does not yield any breakdown, especially no so-called hard breakdown. A hard breakdown in BiCG according to [581] is an iteration, say number k , where the generated tridiagonal matrix of size k is singular. This result is not true when A is real. However, a sufficient condition for the result to be true for a real matrix A is that A has at least one real eigenvalue. This is, for instance, the case if the order of A is odd. Hence, with as input matrix the companion matrix C corresponding to the prescribed spectrum, almost every starting vector r_0 with shadow vector $\tilde{r}_0 = r_0$ will generate, after n iterations, a tridiagonal matrix \hat{T} of size n . Clearly, \hat{T} has the same prescribed spectrum as C and, in addition, no leading principal submatrix of \hat{T} is singular. Then we can generate the upper triangular matrix

$$\hat{U} = [e_1, \hat{T}e_1, \dots, \hat{T}^{n-1}e_1].$$

It follows that

$$\hat{T} = \hat{U}C\hat{U}^{-1}.$$

The matrix \hat{U}^{-1} need not have the first row equal to what we would like for given residual norms $\|r_k\|$. But the first row of \hat{U}^{-1} has no zero entries, otherwise some leading principal submatrix of \hat{T} would be singular. We can always find a nonsingular

diagonal matrix D such that the first row of $\hat{U}^{-1}D$ equals the needed values. Then with $U = D^{-1}\hat{U}$,

$$D^{-1}\hat{T}D = D^{-1}\hat{U}C\hat{U}^{-1}D = UCU^{-1},$$

and $T \equiv D^{-1}\hat{T}D$ is the desired tridiagonal matrix. We proved the following.

Theorem 8.2 *Any residual norm history with finite residual norms is possible for the BiCG method with any nonzero eigenvalues of the system matrix. Any decreasing residual norm history is possible for the QMR method with any nonzero eigenvalues of the system matrix.*

Proof Both claims follow when we apply the BiCG and QMR method to the matrix $T = UCU^{-1}$ with $b = e_1$ from the preceding discussion. Then V is the identity matrix and is in particular unitary and the QMR residual norms equal the prescribed quasi-residual norms. \square

We next show that infinite residual norms cannot be forced at arbitrary predetermined iteration numbers. This result is a nonsymmetric analog of a well-known fact for the symmetric Lanczos process applied to indefinite systems, see, e.g., [220, Section 2.1]. We formulate it for the situation where the underlying nonsymmetric Lanczos process runs till completion, but it holds also when the process terminates prematurely, having found an A -invariant or A^T -invariant Krylov subspace.

Theorem 8.3 *Let the BiCG method be applied to a linear system $Ax = b$ with $x_0 = 0$, and a nonsingular matrix A such that $d(A, b) = n$ and $d(A^T, \tilde{r}_0) = n$. The BiCG residual norms of two consecutive iteration numbers cannot be both infinite.*

Proof Let C be the companion matrix corresponding to the characteristic polynomial of A , let T be the tridiagonal matrix produced by the nonsymmetric Lanczos process and let $U = [e_1, Te_1, \dots, T^{n-1}e_1]$, then $U^{-1}T = CU^{-1}$. The first row of U^{-1} denoted by $\vartheta = [1, \dots, \vartheta_{1,n}]$ contains the BiCG residual norms through $|\vartheta_{1,k+1}|^{-1} = \|r_k\|/\|r_0\|$. Equating the k th column and the first row in $U^{-1}T = CU^{-1}$ for some k , $1 < k < n$, we obtain

$$\vartheta_{1,k-1}\beta_k + \vartheta_{1,k}\alpha_k + \vartheta_{1,k+1}\rho_{k+1} = 0. \quad (8.20)$$

Per assumption, all entries on the lower and upper subdiagonals of T are nonzero, otherwise the nonsymmetric Lanczos process would have terminated early. Therefore relation (8.20) cannot be satisfied if $\vartheta_{1,k-1} = \vartheta_{1,k} = 0$ in combination with $\vartheta_{1,k+1} \neq 0$ or if $\vartheta_{1,k-1} \neq 0$ in combination with $\vartheta_{1,k} = \vartheta_{1,k+1} = 0$. By consecutively using the same rule for neighboring columns we conclude that either (1) all $\vartheta_{1,k}$, $2 \leq k \leq n$, are nonzero, (2) all $\vartheta_{1,k}$, $2 \leq k \leq n$, are zero, or (3) no two consecutive $\vartheta_{1,k-1}, \vartheta_{1,k}$ are zero. The claim follows if we can exclude the case (2) where $\vartheta_{1,k} = 0$ for all $2 \leq k \leq n$. This would lead, with the equation $\alpha_1 + \vartheta_{1,2}\rho_2 = 0$ for the $(1, 1)$ entry of $U^{-1}T = CU^{-1}$, to $\alpha_1 = 0$. Then

$$\det \begin{pmatrix} 0 & \beta_2 \\ \rho_2 & \alpha_2 \end{pmatrix} \neq 0,$$

contradicting $\vartheta_{1,2} = 0$. \square

Note that a similar result was proved in [68].

Thus if we allow infinite residual norms, not every convergence curve is admissible for BiCG. This is in contrast with the situation for FOM and the Hessenberg method. Similarly, in contrast with GMRES and CMRH, two consecutive steps of stagnation of quasi-residual norms are impossible for QMR.

The question whether we can prescribe with a given spectrum of the system matrix, infinite BiCG residual norms at given, non-consecutive iteration numbers amounts to solving an inverse tridiagonal eigenproblem where principal leading submatrices of predetermined sizes must be singular. We did not find any text addressing this problem in the literature. Therefore, we consider prescribing particular $\vartheta_{1,k}$'s to be zero as an open problem.

8.7 Residual norms

Bounds for the QMR residual norms were given in [379]. From Chapter 3, Section 3.7, we have the bound

$$\|r_k^Q\| \leq \|r_0\| \|X_H\| \|X_H^{-1}\| \sqrt{k+1} \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|,$$

where π_k is the set of polynomials of degree k and X_H comes from the spectral factorization of the involved Hessenberg matrix (which is in fact tridiagonal here) $H = X_H \Lambda X_H^{-1}$. This is the bound in [379]. However, using the relations between QMR and GMRES in Section 8.5 we obtain for a diagonalizable matrix $A = X \Lambda X^{-1}$,

$$\|r_k^Q\| \leq \|r_0\| \sqrt{k+1} \|D_k^{-1}(W_{n,k}^Q)^T\| \|X\| \|X^{-1}\| \min_{\substack{p \in \pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|.$$

Of course, we have the same troubles as in GMRES to bound the min max problem. Bounds for the BiCG residual norms can be obtained by using the relations between Q-OR and Q-MR methods in Chapter 3.

8.8 Lanczos algorithms with look-ahead and formal orthogonal polynomials

Methods based on the Lanczos nonsymmetric process including look-ahead variants can also be derived using results on polynomials. We would like to have

$$x_k \in x_0 + \mathcal{K}_k(A, r_0), \quad r_k = b - Ax_k \perp \mathcal{K}_k(A^T, \tilde{r}_0).$$

Therefore the residual vector r_k is a polynomial of degree k in A applied to r_0 . The orthogonality conditions are written as

$$([A^T]^i \tilde{r}_0, r_k) = 0, \quad i = 0, 1, \dots, k-1.$$

Let

$$\varphi_i = ([A^T]^i \tilde{r}_0, r_0) = (\tilde{r}_0, A^i r_0), \quad i = 0, 1, 2, \dots$$

We define a linear functional Φ on the set of polynomials with complex coefficients by $\Phi(\xi^i) = \varphi_i$ for $i = 0, 1, \dots$. The orthogonality conditions are

$$\Phi(\xi^i p_k(\xi)) = 0, \quad i = 0, 1, \dots, k-1. \quad (8.21)$$

A family of polynomials p_k satisfying conditions (8.21) for all values of k is called a family of formal orthogonal polynomials (FOP) with respect to the linear functional Φ . They are defined up to a normalizing factor which we choose such that $p_k(0) = 1$. For properties of FOPs, see [276] or [478, 481]. A noteworthy difference with ordinary orthogonal polynomials is that FOPs may not exist for some values of k . The polynomial p_k exists and is unique if and only if the Hankel determinant

$$h_k^{(1)} = \begin{vmatrix} \varphi_1 & \varphi_2 & \cdots & \varphi_k \\ \varphi_2 & \varphi_3 & \cdots & \varphi_{k+1} \\ \vdots & \vdots & & \vdots \\ \varphi_k & \varphi_{k+1} & \cdots & \varphi_{2k-1} \end{vmatrix},$$

is different from zero. The case $h_k^{(1)} = 0$ corresponds to a breakdown. Let us assume for a moment that the FOPs exist. With the chosen normalization they can be written as

$$p_k(\xi) = \frac{1}{h_k^{(1)}} \begin{vmatrix} 1 & \xi & \cdots & \xi^k \\ \varphi_0 & \varphi_1 & \cdots & \varphi_k \\ \vdots & \vdots & & \vdots \\ \varphi_{k-1} & \varphi_k & \cdots & \varphi_{2k-1} \end{vmatrix}.$$

It is well known that such sequences of formal orthogonal polynomials are the denominators of Padé approximants lying on a diagonal or a staircase of a Padé table

and that these Padé approximants are also the convergents of continued fractions. For details about these connections, see [478, 481] and [129]. A review of formal orthogonality in Lanczos methods was presented in [144].

Let us introduce another family of polynomials

$$p_k^{(1)}(\xi) = \frac{1}{h_k^{(1)}} \begin{vmatrix} \varphi_1 & \varphi_2 & \cdots & \varphi_{k+1} \\ \vdots & \vdots & & \vdots \\ \varphi_k & \varphi_{k+1} & \cdots & \varphi_{2k} \\ 1 & \xi & \cdots & \xi^k \end{vmatrix}.$$

We choose the normalization such that $p_k^{(1)}$ is monic (that is, the coefficient of ξ^k is equal to 1). Defining the linear functional $\Phi^{(1)}$ as

$$\Phi^{(1)}(\xi^i) = \Phi(\xi^{i+1}) = \varphi_{i+1}, \quad i = 0, 1, \dots,$$

the orthogonality conditions give

$$\Phi^{(1)}(\xi^i p_k^{(1)}(\xi)) = 0, \quad i = 0, 1, \dots, k-1.$$

Hence, $p_k^{(1)}$ is also a family of FOPs. The two families are said to be adjacent. The polynomials p_k (when they exist) can be computed by a three-term recurrence. But they can also be computed with the help of the adjacent family. These two families of FOPs were used to develop Lanczos methods; see [125, 135, 143, 148] and [478, 481].

There are many ways of recursively computing the two families of FOPs. A systematic study of these relations was done in [53]; see also [54]. Ten possibilities were identified for p_k as well as for $p_k^{(1)}$. They are listed in Table 8.1.

Table 8.1 Possible recurrences for p_k and $p_k^{(1)}$

A ₁	$p_{k-2}, p_{k-2}^{(1)} \rightarrow p_k$	B ₁	$p_{k-2}, p_{k-2}^{(1)} \rightarrow p_k^{(1)}$
A ₂	$p_{k-2}, p_{k-1}^{(1)} \rightarrow p_k$	B ₂	$p_{k-2}, p_{k-1} \rightarrow p_k^{(1)}$
A ₃	$p_{k-2}, p_k^{(1)} \rightarrow p_k$	B ₃	$p_{k-2}, p_k \rightarrow p_k^{(1)}$
A ₄	$p_{k-2}, p_{k-1} \rightarrow p_k$	B ₄	$p_{k-2}, p_{k-1} \rightarrow p_k^{(1)}$
A ₅	$p_{k-2}^{(1)}, p_{k-1} \rightarrow p_k$	B ₅	$p_{k-2}^{(1)}, p_{k-1} \rightarrow p_k^{(1)}$
A ₆	$p_{k-2}^{(1)}, p_{k-1}^{(1)} \rightarrow p_k$	B ₆	$p_{k-2}^{(1)}, p_{k-1}^{(1)} \rightarrow p_k^{(1)}$
A ₇	$p_{k-2}^{(1)}, p_k \rightarrow p_k$	B ₇	$p_{k-2}^{(1)}, p_k \rightarrow p_k^{(1)}$
A ₈	$p_{k-1}, p_{k-1}^{(1)} \rightarrow p_k$	B ₈	$p_{k-1}, p_{k-1}^{(1)} \rightarrow p_k^{(1)}$
A ₉	$p_{k-1}, p_k^{(1)} \rightarrow p_k$	B ₉	$p_{k-1}, p_k \rightarrow p_k^{(1)}$
A ₁₀	$p_{k-1}^{(1)}, p_k \rightarrow p_k$	B ₁₀	$p_{k-1}^{(1)}, p_k \rightarrow p_k^{(1)}$

Not all combinations of A_i and B_j are possible. For instance, A_3/B_3 is not feasible since p_k needs $p_k^{(1)}$ and $p_k^{(1)}$ needs p_k . Similarly A_7/B_7 , A_9/B_9 and A_{10}/B_{10} are not feasible. We must also observe that the orthogonality conditions can use any monic polynomial q_i of exact degree i instead of the monomial ξ^i . Therefore, we can write them as

$$\Phi(q_i(\xi)p_k(\xi)) = 0, \quad \Phi^{(1)}(q_i(\xi)p_k^{(1)}(\xi)) = 0, \quad i = 0, 1, \dots, k-1.$$

Interesting choices for q_i are ξ^i , p_i and $p_i^{(1)}$. It is shown in [53] that the only interesting relations are A_1 , A_4 , A_5 , A_8 and B_1 , B_5 , B_6 , B_8 , B_{10} . The recurrence relations are given in [53]. Some of these methods are already known. The relation A_4 where p_k is computed from p_{k-1} and p_{k-2} gives Lanczos/Orthores, Lanczos/Orthodir is obtained with A_8/B_6 and Lanczos/Orthomin is obtained from A_8/B_{10} with a different scaling of the polynomials $p_k^{(1)}$. Each pair of relations can be considered with any choice of the additional polynomial q_i , particularly ξ^i , p_i and $p_i^{(1)}$. Some combinations need three matrix–vector products per iteration and therefore are not so interesting. Of course, when the monomials are used for q_i , the algorithms tend to be unstable. The residual norms oscillate without decreasing too much for some problems; they can even diverge. From numerical experiments in [53] it seems that the most interesting methods are A_5/B_{10} , A_8/B_8 , A_5/B_5 , A_8/B_{10} when $q_i = p_i$ and A_5/B_{10} , A_8/B_8 , A_5/B_5 when $q_i = p_i^{(1)}$. The methods described in [53] involve polynomials whose degrees are differing at most by two. Relations involving polynomials whose degrees differ by more than two were considered in [342] but only for the choice $q_i(\xi) = \xi^i$.

All the methods described above can suffer from breakdowns since some FOPs may not exist. A possible cure is to jump over the non-existing polynomials. If p_{n_k} is a regular polynomial and p_{n_k+1} does not exist, we must look for the next regular polynomial $p_{n_{k+1}}$. The next index is $n_{k+1} = n_k + m_k$ where m_k is the length of the jump. In theory and in the case of an exact breakdown m_k is determined by finding a nonzero dot product; for instance, if the exact breakdown is determined by $(u_k, Az_k) = 0$, we look for the smallest integer m_k such that $(u_k, A^{m_k} z_k) \neq 0$. In practice, zero is replaced by a small value. If such a value does not exist, we have an incurable breakdown.

When a finite (hopefully small) m_k has been found, the polynomials can be computed by some recurrences. For instance, for a relation of type A_8 we write

$$p_{n_{k+1}}(\xi) = s_k(\xi)p_{n_k}(\xi) + \xi t_k(\xi)p_{n_k}^{(1)}(\xi),$$

where s_k and t_k are polynomials of degree at most $m_k - 1$. Since m_k must not be very large, we could use the natural (monomial) basis for the additional polynomials and write

$$s_k(\xi) = \alpha_{m_k-1}\xi^{m_k-1} + \dots + \alpha_1\xi + 1, \quad t_k(\xi) = \beta_{m_k-1}\xi^{m_k-1} + \dots + \beta_1\xi + 1.$$

There are $2m_k - 2$ coefficients to compute. From [276] we know that

$$\begin{aligned}\Phi(\xi^i p_{n_{k+1}}(\xi)) &= 0 \text{ for } i = 0, \dots, n_k + m_k - 1, \\ \Phi(\xi^i p_{n_k}(\xi)) &= 0 \text{ for } i = 0, \dots, n_k - 1, \\ \Phi(\xi^{i+1} p_{n_k}^{(1)}(\xi)) &= 0 \text{ for } i = 0, \dots, n_k + m_k - 2, \\ \Phi(\xi^{i+1} p_{n_k}^{(1)}(\xi)) &\neq 0 \text{ for } i = n_k + m_k - 2.\end{aligned}$$

After some manipulations, these relations lead to the choice $s_k(\xi) \equiv 1$ and the coefficients of t_k are obtained by solving a triangular system of order $m_k - 1$; see [53].

A relation of type B_6 can be written as

$$p_{n_{k+1}}^{(1)}(\xi) = \gamma_k p_{n_{k-1}}^{(1)}(\xi) + (\xi^{m_k} + \alpha_{m_k-1} \xi^{m_k-1} + \dots + \alpha_1 \xi + \alpha_0) p_{n_k}^{(1)}(\xi),$$

where γ_k is a scalar and the coefficients α_j are obtained by solving a triangular system of order m_k . A method called MRZ (Method of Recursive Zoom), implementing A_8/B_6 (that is, Lanczos/Orthores) with look-ahead, was proposed in [138–142]. Denoting $z_k = p_{n_k}^{(1)}(A)r_0$, the length of the jump is determined by

$$(\tilde{r}_0, A^{i+1} z_k) = 0, \quad i = 0, \dots, n_k + m_k - 2, \quad (\tilde{r}_0, A^{n_k+m_k} z_k) \neq 0.$$

Variants of this method, called SMRZ (using A_8/B_8) and BMRZ (using A_8/B_{10}), were also described. Other methods were considered in [53]. Of course, exact breakdowns are unlikely, except in contrived examples. A method, called BSMRZ, designed to handle near-breakdowns was also presented. It is necessary to jump over the non-existing polynomials but also over the regular but badly computed ones. Then, the recurrences become more complicated and the linear systems for the computation of their coefficients are no longer triangular. However, these methods are using $q_i(\xi) = \xi^i$ and therefore are not really useful for solving practical problems. But the same techniques can be used for other choices of q_i . An algorithm called MRZ-stab using $q_i = p_i^{(1)}$ was proposed in [142, 143]. It is, of course, much more stable than MRZ. To reduce the storage in these algorithms it was proposed in [142] to use Horner's rule to evaluate some polynomials. It allows to have a storage which is independent of the lengths of the jumps. These methods are named HMRZ (with $q_i(\xi) = \xi^i$) and HMRZ-stab (with $q_i = p_i^{(1)}$). A similar technique was also described in [46, 47]. It uses intermediate polynomials that are biorthogonal and satisfy a simple three-term recurrence relation. They can be considered as an alternative for orthogonal polynomials which are not regular or non-existent. A code for HMRZ-stab is as follows. The input arguments in varargin are epss, the threshold used to determine the lengths of the jumps, and max_jump, the maximum allowed length for a jump.

```
function [x,ni,resn,nk] = HMRZ_stab(A,b,x0,w0,epsi,nitmax,
varargin);
%
if nargin < 7
epss = 1e-15;
else
```

```
epss = varargin{1};  
end % if  
if length(varargin) < 2 || nargin < 7  
    max_jump = nitmax + 1;  
else  
    max_jump = varargin{2};  
end % if  
nb = norm(b);  
nA = size(A,1);  
x = x0;  
r = b - A * x;  
resn = zeros(1,nitmax+1);  
resnt = zeros(1,nitmax+1);  
nk = zeros(1,nitmax+1);  
d = zeros(1,max_jump+1);  
resn(1) = norm(r);  
resnt(1) = resn(1);  
At = A';  
if isempty(w0)  
    w0 = randn(nA,1);  
end % if  
z1 = zeros(nA,1);  
zt1 = z1;  
z = r;  
zt = w0;  
c = 0;  
b0 = 1;  
ni = 0;  
%  
for k = 1:nitmax  
    ni = ni + 1;  
    d(1) = zt' * r;  
    yt = At * zt; % matrix vector product with the transpose  
    ut = yt;  
    b01 = b0;  
    b0 = yt' * z;  
    nz = norm(z);  
    nynz = norm(yt) * nz;  
    mk = 1;  
    while (abs(b0) <= nynz * epss) && mk <= max_jump  
        mk = mk + 1;  
        d(mk) = yt' * r;  
        yt = At * yt; % matrix vector product with the transpose  
        b0 = yt' * z;  
        nynz = norm(yt) * nz;
```

```

end % while
if mk > max_jump
    error(' The jump is too large')
end % if
nk(ni+1) = nk(ni) + mk;
if k ~= 1
    c = b0 / b01;
end % if k
t = z;
z_old = z;
z = -c * z1;
tt = zt;
zt_old = zt;
zt = -c * zt1;
for i = 1:mk
    u = A * t; % matrix vector product
    beta = d(mk-i+1) / b0;
    x = x + beta * t;
    r = r - beta * u;
    gamma = -(yt' * u) / b0;
    t = u + gamma * z_old;
    if i ~= 1
        ut = At * tt; % matrix vector product with the transpose
    end % if i
    tt = ut + gamma * zt_old;
end % for i
z1 = z_old;
zt1 = zt_old;
z = z + t;
zt = zt + tt;
nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k
resn = resn(1:ni+1);
resnt = resnt(1:ni+1);
nk = nk(1:ni+1);

```

A technique for expressing a particular polynomial in a basis formed by the regular polynomials and the non-regular ones was proposed in [474].

One aspect of Lanczos methods is the implicit or explicit use of a diagonal sequence of Padé approximants; see [478]. The look-ahead algorithms use a strategy to jump across any blocks in the Padé table. The look-ahead strategy is based on

using inner polynomials when zero divisors are encountered. In [446] good pairs of approximants are constructed which can be considered as being on the edges of square blocks in the Padé table. The method is equivalent to going around the blocks instead of going across them. Moreover, this method does not need the (conjugate) transpose of A .

Handling of breakdowns in BiCG was also considered in [580, 581] using switches between Lanczos/Orthomin and Lanczos/Orthodir methods. A somehow similar strategy was proposed later in [342], restarting with another algorithm of type A_i/B_j when a breakdown occurs. This may sometimes lead to convergence but the cost may be higher compared to a method that does not suffer a breakdown.

In [1006] a breakdown-free variation of the nonsymmetric Lanczos algorithm was proposed. The idea is, when there is a breakdown, to replace one of the two Krylov subspaces by a sum of two Krylov subspaces. The Lanczos three-term recurrence no longer holds, but the Gram–Schmidt biorthogonalization method is still applicable for the new pair of subspaces. At the iteration where the breakdown occurs, the recurrence for the w_j vectors is augmented with another vector which is called the new-start vector. For the subsequent iterations we have four-term recurrences. In fact, each time a (near-)breakdown is detected the lengths of the two recurrences are increased by one. Hence, the matrices T_k are upper Hessenberg. They start as being tridiagonal and the upper bandwidth is increased by one at each breakdown. The difficult point is the choice of the new-start vector. It must be chosen as orthogonal to all the previous v_i up to $k - 1$ if the breakdown occurs at iteration k and with an angle large enough with v_k . Some choices are proposed in Section 5 of [1006] where this method was presented for eigenvalue computations. The solution of linear systems was considered in [920].

8.9 Parallel computing

We have derived the BiCG algorithm from the nonsymmetric Lanczos algorithm. Similarly, parallel versions of BiCG can be derived from parallel variants of the Lanczos algorithm. In Chapter 4 we have reviewed some parallel versions of the Lanczos algorithm; see [188, 527, 597]. One of these variants is the CA-Lanczos algorithm, a communication-avoiding method. A CA-BiCG algorithm was derived from it in [67]. However, CA-BiCG can be derived from scratch without referring to CA-Lanczos, see [188, 193].

Starting from $p_{sk+1}, \tilde{p}_{sk+1}, r_{sk+1}, \tilde{r}_{sk+1}$ and polynomials q_j given by three-term recurrences (monomial, Newton or Chebyshev), we define matrices

$$\begin{aligned}\mathcal{P}_{n,(k)} &= (q_0(A)p_{sk+1} \ q_1(A)p_{sk+1} \cdots q_s(A)p_{sk+1}), \\ \tilde{\mathcal{P}}_{n,(k)} &= (q_0(A)\tilde{p}_{sk+1} \ q_1(A)\tilde{p}_{sk+1} \cdots q_s(A)\tilde{p}_{sk+1}), \\ \mathcal{R}_{n,(k)} &= (q_0(A^T)r_{sk+1} \ q_1(A^T)r_{sk+1} \cdots q_{s-1}(A^T)r_{sk+1}), \\ \tilde{\mathcal{R}}_{n,(k)} &= (q_0(A^T)\tilde{r}_{sk+1} \ q_1(A^T)\tilde{r}_{sk+1} \cdots q_{s-1}(A^T)\tilde{r}_{sk+1}).\end{aligned}$$

This is computed using the matrix powers kernel we described in Chapter 4. We set

$$\mathcal{Z}_{n,(k)} = (\mathcal{P}_{n,(k)} \ \mathcal{R}_{n,(k)}), \quad \tilde{\mathcal{Z}}_{n,(k)} = (\tilde{\mathcal{P}}_{n,(k)} \ \tilde{\mathcal{R}}_{n,(k)}),$$

and $G_{(k)} = \tilde{\mathcal{Z}}_{n,(k)}^T \mathcal{Z}_{n,(k)}$ of order $2s + 1$. These are the only steps in which communications are needed. The previous polynomial relations can be written in matrix form

$$A\underline{\mathcal{P}}_{n,(k)} = \mathcal{P}_{n,(k)} B_{(k)}^{\mathcal{P}}, \quad A^T \underline{\tilde{\mathcal{P}}}_{n,(k)} = \tilde{\mathcal{P}}_{n,(k)} B_{(k)}^{\tilde{\mathcal{P}}},$$

$$A\underline{\mathcal{R}}_{n,(k)} = \mathcal{R}_{n,(k)} B_{(k)}^{\mathcal{R}}, \quad A^T \underline{\tilde{\mathcal{R}}}_{n,(k)} = \tilde{\mathcal{R}}_{n,(k)} B_{(k)}^{\tilde{\mathcal{R}}}.$$

where $\underline{\mathcal{P}}_{n,(k)}$ is the same as $\mathcal{P}_{n,(k)}$ but with the last column set to zero, and the same for the other matrices. Note that this differs from our usual underline notation. It yields

$$A\underline{\mathcal{Z}}_{n,(k)} = \mathcal{Z}_{n,(k)} B_{(k)}^{\mathcal{Z}}, \quad A^T \underline{\tilde{\mathcal{Z}}}_{n,(k)} = \tilde{\mathcal{Z}}_{n,(k)} B_{(k)}^{\tilde{\mathcal{Z}}},$$

with block diagonal matrices,

$$B_{(k)}^{\mathcal{Z}} = \begin{pmatrix} B_{(k)}^{\mathcal{P}} & \\ & B_{(k)}^{\mathcal{R}} \end{pmatrix}, \quad B_{(k)}^{\tilde{\mathcal{Z}}} = \begin{pmatrix} B_{(k)}^{\tilde{\mathcal{P}}} & \\ & B_{(k)}^{\tilde{\mathcal{R}}} \end{pmatrix},$$

and $\underline{\mathcal{Z}}_{n,(k)}$ (resp. $\underline{\tilde{\mathcal{Z}}}_{n,(k)}$) is constructed from $\underline{\mathcal{P}}_{n,(k)}$ and $\underline{\mathcal{R}}_{n,(k)}$ (resp. $\underline{\tilde{\mathcal{P}}}_{n,(k)}$ and $\underline{\tilde{\mathcal{R}}}_{n,(k)}$). Then, using the BiCG recurrence relations, we introduce vectors of coefficients with $2s + 1$ components $\hat{p}_j, \check{p}_j, \hat{r}_j, \check{r}_j$ and \hat{x}_j in the new basis. They are given recursively by the relations

$$\begin{aligned}\hat{x}_{j+1} &= \hat{x}_j + \gamma_{sk+j} \hat{p}_j, \\ \hat{r}_{j+1} &= \hat{r}_j - \gamma_{sk+j} \mathcal{B}_{(k)}^{\mathcal{Z}} \hat{p}_j, \\ \check{r}_{j+1} &= \check{r}_j - \gamma_{sk+j} \mathcal{B}_{(k)}^{\tilde{\mathcal{Z}}} \check{p}_j, \\ \hat{p}_{j+1} &= \hat{r}_{j+1} + \mu_{sk+j} \hat{p}_j, \\ \check{p}_{j+1} &= \check{r}_{j+1} + \mu_{sk+j} \check{p}_j,\end{aligned}$$

for $j = 1, \dots, s$ with initial values $\hat{p}_1 = \check{p}_1 = e_1, \hat{r}_1 = \check{r}_1 = \hat{e}$ a vector which is zero except for the component $s + 2$ which is equal to 1 and $\hat{x}_1 = 0$. The coefficients are computed as

$$\gamma_{sk+j} = \frac{\delta_{sk+j}}{(\check{p}_j, G_{(k)} B_{(k)}^Z \hat{p}_j)}, \quad \delta_{sk+j+1} = (\check{r}_{j+1}, G_{(k)} \hat{r}_{j+1}), \quad \mu_{sk+j} = \frac{\delta_{sk+j+1}}{\delta_{sk+j}}.$$

The iterates p_{sk+s+1} , \tilde{p}_{sk+s+1} , r_{sk+s+1} , \tilde{r}_{sk+s+1} and x_{sk+s+1} are recovered by

$$(p_{sk+s+1} \ r_{sk+s+1} \ x_{sk+s+1} - x_{sk+1}) = \mathcal{Z}_{n,(k)} (\hat{p}_{s+1} \ \hat{r}_{s+1} \ \hat{x}_{s+1}),$$

$$(\tilde{p}_{sk+s+1} \ \tilde{r}_{sk+s+1}) = \tilde{\mathcal{Z}}_{n,(k)} (\check{p}_{s+1} \ \check{r}_{s+1}).$$

Analyses of CA-BiCG in finite precision arithmetic were done in [188, 190]. It is shown that bounds for the residual norms and the maximum attainable accuracy can be written in the same form as the corresponding bounds for the sequential method multiplied by an amplification factor depending on the condition number of the s -step local bases. If the local basis is badly conditioned (like with the monomial basis and s large) the amplification factor can be large.

A variant of BiCG was proposed in [156, 157] derived from a parallel version of the nonsymmetric Lanczos algorithm. The goal was to reduce the data dependencies of the standard BiCG by introducing new vectors and to be able to overlap computation and communication. Today this could be called a communication-hiding algorithm. In this new version there are 5 vector updates and 4 independent dot products that can be all computed in parallel instead of 3 in the standard BiCG (if we compute the norm of the residual). A slight variation of this algorithm was used in [1003, 1004] under the name *improved BiCG*.

From the parallel Lanczos algorithm, a parallel QMR algorithm was also devised, see [155] and also [1002].

As we have seen in Chapter 4 s -step versions of the nonsymmetric Lanczos algorithm were proposed in [596, 597]. Almost at the same time, an s -step QMR algorithm was devised in [374, 375]. It is using similar techniques as QMR with look-ahead except that blocks of inner vectors are used systematically with, in addition, a variable block size and a combination with look-ahead.

Another s -step nonsymmetric Lanczos algorithm was presented in [347] from which BiCG and QMR algorithms were derived. It is a synchronization-reducing algorithm which, unfortunately, uses the monomial basis and only small values of $s \leq 5$ can be used. It has only one single global synchronization point per s iterations for which it needs $2s$ dot products. The main difference with [596, 597] is the normalization of vectors.

8.10 Finite precision arithmetic

The interesting questions when using the methods based on the Lanczos nonsymmetric methods are related to the loss of biorthogonality, the linear independence of the basis vectors and the accuracy of the computations.

Contrary to what happens for the symmetric case, there are not many papers proving results about the behavior of Lanczos methods in finite precision arithmetic when A is nonsymmetric. One thing that is obvious is that BiCG cannot be backward stable because when A is symmetric BiCG reduces to CG and it is known that CG is not backward stable; see, for instance, [676].

The paper [55] studied the Lanczos nonsymmetric process in finite precision arithmetic. The author was interested in eigenvalue computations. His analysis was in the same spirit as what Paige did for the symmetric Lanczos algorithm; see [730] and [676, 688] for a summary of the results. The normalization is such that $\omega_k = w_k^T v_k = 1$. It is shown that the computed quantities satisfy

$$\begin{aligned} AV_{n,k} &= V_{n,k} T_k + \rho_{k+1} v_{k+1} e_k^T + F_k, \\ AW_{n,k} &= W_{n,k} T_k^T + \zeta_{k+1} w_{k+1} e_k^T + G_k, \end{aligned}$$

and the perturbation terms due to rounding errors are bounded entrywise by

$$\begin{aligned} |F_k| &\leq (3+m)\varepsilon|A|\|V_{n,k}\| + 4\varepsilon\|V_{n,k}\|\|T_k\| + O(\varepsilon^2), \\ |G_k| &\leq (3+m)\varepsilon|A|^T\|W_{n,k}\| + 4\varepsilon\|W_{n,k}\|\|T_k\|^T + O(\varepsilon^2), \end{aligned}$$

where m is the maximum number of nonzero entries in a row of A and ε is the machine precision. In finite precision arithmetic the biorthogonality among the basis vectors v_j and w_j can be lost. Under the assumption that local biorthogonality is satisfied up to machine precision and that no (near-)breakdown has occurred, it is shown in [55] that the loss of biorthogonality is linked to the convergence of the Lanczos process as it was the case for symmetric problems. The derivation of these results is too technical to be reported here. However, in the nonsymmetric Lanczos process, even without any breakdown (i.e., $\omega_i \neq 0$), the algorithm is still prone to potential instabilities (near-breakdown), that is, at least some ω_i can be tiny. Hence, huge intermediate quantities $\|v_i\|$ and $\|w_i\|$ could appear when these vectors are not normalized. If this happens, Ritz values will have a huge condition number and the relation between the loss of biorthogonality and the convergence of Ritz triplets may no longer hold.

An analysis of BiCG in finite precision arithmetic was done in [921]. We have seen in Section 8.2 how to derive BiCG from the Lanczos algorithm. In this paper the opposite direction is taken. The authors first derived the terms due to rounding errors in all the steps of BiCG and then eliminate the \tilde{r}_j and \tilde{p}_j to obtain a matrix equation for the computed residual vectors. This equation is similar to the one we have started from plus a perturbation term, which is different from the perturbation term for the Lanczos algorithm. The exactly normalized residual vectors are denoted by $z_j = r_j / \|r_j\|$. They satisfy

$$AZ_{n,k} = Z_{n,k} \hat{T}_k + \frac{1}{\mu_k} \frac{r_{k+1}}{\|r_0\|} e_k^T + \varepsilon \Delta_k,$$

with $\mu_k = \alpha_k \|r_k\|/\|r_0\|$, \hat{T}_k is a tridiagonal matrix and the columns of the perturbation matrix Δ_k are $d_{j+1}/\|r_j\|$ with

$$|d_j| \leq \left(c_1 |A| + \frac{1}{|\alpha_j|} + \frac{|\beta_j|}{|\alpha_{j-1}|} \right) |r_j| + c_2 |A| |p_j| + O(\varepsilon),$$

where α_j and β_j are the computed BiCG coefficients and c_1 and c_2 are constants depending only on n or on the largest number of nonzeros per row of A . From this result and if the vectors r_j are linearly independent, it is proved in [921] that

$$\|r_k\| \leq (1 + \nu_k) \min_{p(0)=1} \|p(A + \delta A_k)r_0\|,$$

where p is a polynomial of degree k , $\delta A_k = -\varepsilon \Delta_k Z_{n,k}^+$ and $\nu_k = \|(AZ_{n,k} - \varepsilon \Delta_k)T_k^{-1}V_0^T\|$. The matrix $Z_{n,k}^+$ is the pseudo-inverse of $Z_{n,k}$ and V_0 is a matrix such that $V_0^T Z_{n,k} = I$ and $V_0^T z_{k+1} = 0$. The term with the polynomial p is the norm of (exact) GMRES applied to the perturbed matrix $A + \delta A_k$. Unfortunately, ν_k can grow unboundedly but BiCG convergence can be achieved if ν_k increases at a rate slower than the rate at which the GMRES norm decreases. The case where the vectors r_j are linearly dependent is also considered in [921].

The nonsymmetric Lanczos process (there called the two-sided Lanczos tridiagonalization process) was also considered in [734]. In [732] an “augmented” rounding error analysis was done for the symmetric case. This means that the computed tridiagonal matrix comes from an error-free Lanczos process applied to a slightly symmetrically perturbed block diagonal matrix which is not the same at each iteration. At iteration $k + 1$ this matrix is $\text{diag}(T_k, A)$ where T_k is the Lanczos tridiagonal matrix at iteration k . The authors of [734] studied how this can be extended to the nonsymmetric case. They considered perturbed recurrences relations,

$$\begin{aligned} AV_{n,k} + E_k &= V_{n,k} \underline{T}_k, \\ A^T W_{n,k} + F_k &= W_{n,k} \underline{T}_k^T. \end{aligned}$$

We observe that such type of perturbed relations were studied in [1018, 1020]. When there is no perturbation, the basis vectors satisfy $W_{n,k}^T V_{n,k} = I$. Through a slight modification of the perturbation terms it can be assumed that the diagonal entries of $W_{n,k}^T V_{n,k}$ are equal to 1. Then, it is proved in [734] Corollary 4.1 that running k steps of the Lanczos process on A in the presence of perturbations is equivalent to running k steps of an exact Lanczos process on a perturbation of the augmented matrix $\text{diag}(T_k, A)$. However, contrary to the symmetric case, the perturbation terms cannot be bounded a priori in terms of $\|E_k\|$ and $\|F_k\|$ because the perturbation terms depend on $V_{n,k}$ and $W_{n,k}$ whose column norms can grow unboundedly. Hence, the nonsymmetric Lanczos process is not numerically stable in the augmented sense that the symmetric Lanczos process was proved to be.

Ways to correct the loss of orthogonality in the nonsymmetric Lanczos algorithm were considered in [258, 259] for the version of the algorithm with unit basis vectors. Let $D_k = W_{n,k}^T V_{n,k}$. If the algorithm produces candidate vectors which we denote as v_{k+1}^c and w_{k+1}^c and if we decide that they are not orthogonal “enough” to the previous vectors (that is, $\|W_{n,k}^T v_{k+1}^c\|$ and $\|V_{n,k}^T w_{k+1}^c\|$ are “too large”), then the Gram–Schmidt algorithm is used to correct this situation. Let w_{k+1}^G and v_{k+1}^G be the new vectors after correction. Measures of biorthogonality are defined as

$$\| |D_k|^{-\frac{1}{2}} V_{n,k}^T w_{k+1}^G \|_1, \quad \| (v_{k+1}^G)^T W_{n,k} |D_k|^{-\frac{1}{2}} \|_\infty.$$

Semi-biorthogonality is said to hold if the maximum of these two quantities is smaller than $\sqrt{\epsilon}$. Local orthogonality is defined as

$$\max_{1 \leq i \leq k} (|(w_i v_{i-1}|), |(w_{i-1}, v_i)|) \leq 4\epsilon.$$

Variants of the Lanczos nonsymmetric process to maintain local orthogonality and semi-biorthogonality are described in [258, 259].

8.11 Numerical examples

In this section we discuss the results of numerical experiments with some of the methods presented in this chapter.

8.11.1 BiCG

We first compare BiCG (which is Lanczos–Orthomin) with Lanczos–Orthodir and Lanczos–Orthores. For the problem `fs 183 6` BiCG converges but it needs more than 1000 iterations to reach the maximum attainable accuracy with large oscillations. Lanczos–Orthores stagnates and Lanczos–Orthodir blows up; see Figure 8.1.

Figure 8.2 shows that rounding errors have an important influence on the convergence of BiCG for this example. We compare BiCG in double precision and in variable precision with 32 and 64 decimal digits. The convergence is much faster in extended precision showing that the rounding errors and the loss of biorthogonality are responsible for the slow convergence observed in Figure 8.1.

The problem `fs 680 1c` is easier to solve and the three Lanczos methods give more or less the same residual norms for a little more than 80 iterations. Then, the residual norms of Lanczos–Orthodir do not decrease any longer. Lanczos–Orthores does better but its final accuracy is worse than that of BiCG as we observe in Figure 8.3.

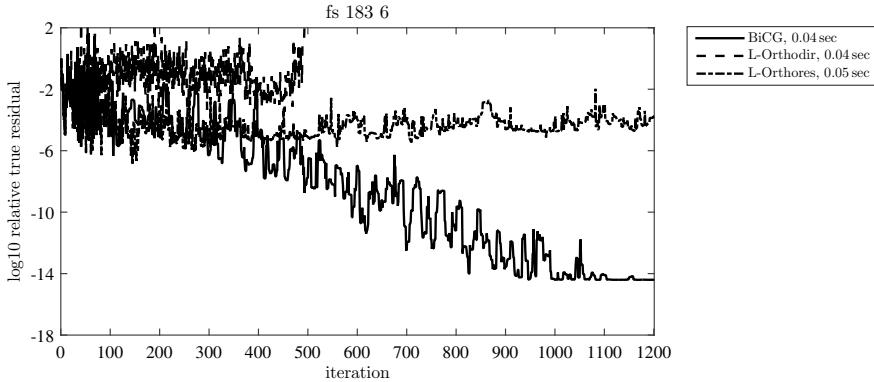


Fig. 8.1 fs 183 6, relative true residual norms, BiCG (plain), Lanczos–Orthodir (dashed), Lanczos–Orthores (dot-dashed)

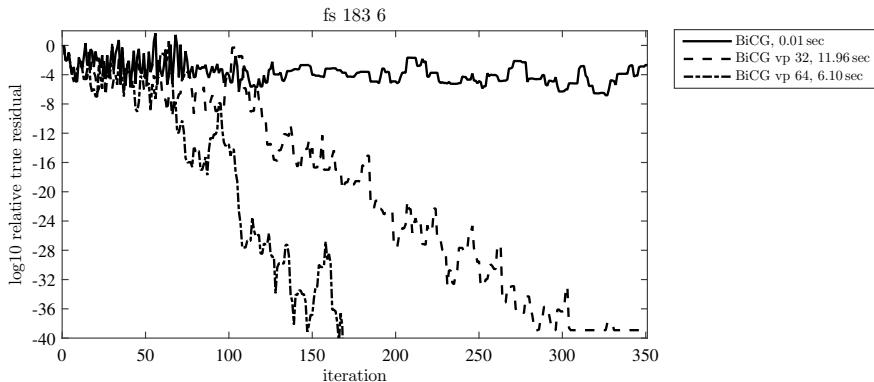


Fig. 8.2 fs 183 6, \log_{10} of the relative true residual norm, BiCG (plain), BiCG variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

In Figure 8.4 we compare BiCG in double precision and in variable precision with 32 and 64 decimal digits for fs 680 1c. Contrary to the previous example the convergence of the three versions is almost the same until they reach their maximum attainable accuracy. This is because the double-precision basis is of full rank for this problem.

Figure 8.5 shows that for supg 001 of order 1225 Lanczos–Orthodir blows up almost immediately and Lanczos–Orthores does not converge. Only BiCG converges but the final accuracy is not very good if we compare to what is obtained with GMRES.

Table 8.2 shows the minimum relative true residual norms in at most 5000 iterations. The method which gives the best results is clearly BiCG (Lanczos–Orthomin).

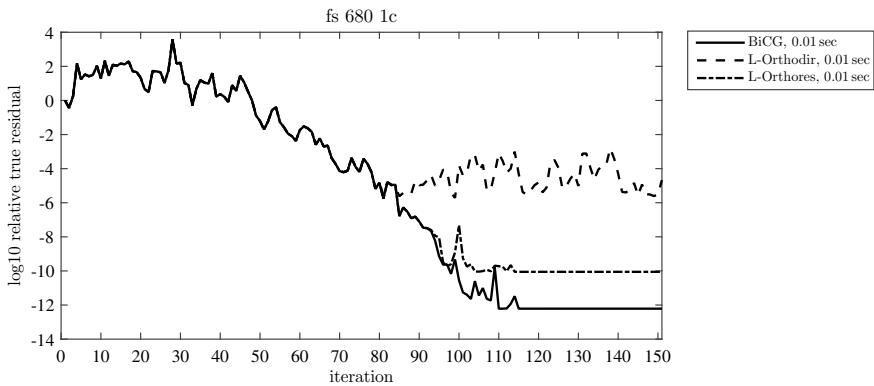


Fig. 8.3 *fs 680 1c*, relative true residual norms, BiCG (plain), Lanczos–Orthodir (dashed), Lanczos–Orthores (dot-dashed)

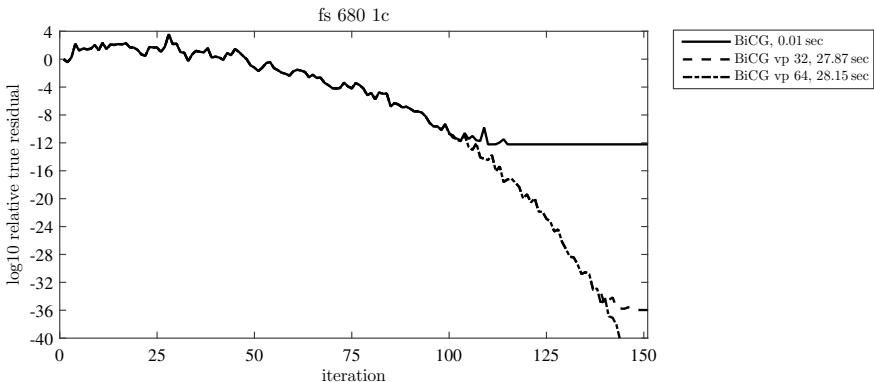


Fig. 8.4 *fs 680 1c*, \log_{10} of the relative true residual norm, BiCG (plain), BiCG variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

There are a few examples for which none of the three methods is converging in 5000 iterations (the relative residual norm is equal to 1).

For the set of larger matrices there are many examples for which none of the three methods is converging well; see Table 8.3.

8.11.2 QMR

We consider BiCG and QMR with the two-term (2t) and three-term (3t) recurrences without look-ahead. As a shadow vector we use $r_0 = b$ since we choose $x_0 = 0$. We

Table 8.2 Minimum relative true residual norms in at most 5000 iterations

matrix	BiCG	L-Orthodir	L-Orthores
pde225	2.23120e-15	1.86832e-11	6.67620e-13
gre 343	5.25537e-15	5.04910e-11	4.03196e-10
jphw 991	1.00000e+00	1.00000e+00	1.00000e+00
pde2961	4.25456e-12	3.35176e-03	2.78599e-10
jagmesh1	3.38975e-16	7.17520e-16	2.81834e-15
bfsa782	1.56558e-13	1.77116e-11	2.01544e-10
dw2048	2.02265e-15	4.75414e-10	1.77435e-07
jagmesh2	6.70201e-16	1.33754e-15	1.75518e-15
raefsky2	8.56197e-14	8.22366e-09	2.04180e-08
fs 680 1c	6.09486e-13	4.13235e-07	8.82775e-11
add20	1.47896e-13	3.04845e-12	1.30856e-10
raefsky1	9.12030e-13	4.50253e-10	2.08693e-08
jagmesh4	7.20366e-16	1.25349e-15	2.22990e-15
fs 680 1	1.15296e-14	3.33430e-04	3.23325e-10
sherman1	6.24477e-14	6.40434e-14	4.25965e-12
nos3	1.70667e-14	1.78636e-14	4.90849e-12
sherman5	2.73876e-10	1.77722e-02	3.53565e-01
cavity05	2.42755e-03	2.72204e-03	1.87482e-03
e05r0500	1.63195e-01	6.95623e-01	1.82839e-01
comsol	4.48313e-11	1.52550e-10	1.95270e-07
olm1000	6.44749e-13	1.23971e-04	5.07592e-08
cavity10	2.89071e-09	1.44624e-06	4.09930e-06
steam2	5.41725e-14	4.03213e-10	1.88479e-08
1138bus	1.39110e-11	1.66563e-11	1.94624e-08
steam1	1.29588e-12	1.14352e-04	1.85659e-05
bcsstk26	2.79203e-06	3.02010e-06	3.05206e-06
nos7	4.02374e-07	1.09982e-03	2.40758e-01
watt1	9.08182e-14	6.74031e-09	4.67610e-08
bcsstk14	1.00000e+00	1.00000e+00	1.00000e+00
fs 183 6	3.96972e-15	9.16488e-06	1.21768e-07
bcsstk20	1.00000e+00	1.00000e+00	1.00000e+00
mcf6	1.00000e+00	1.00000e+00	1.00000e+00
nnc	1.00000e+00	1.00000e+00	1.00000e+00
Insp	1.00000e+00	1.00000e+00	1.00000e+00

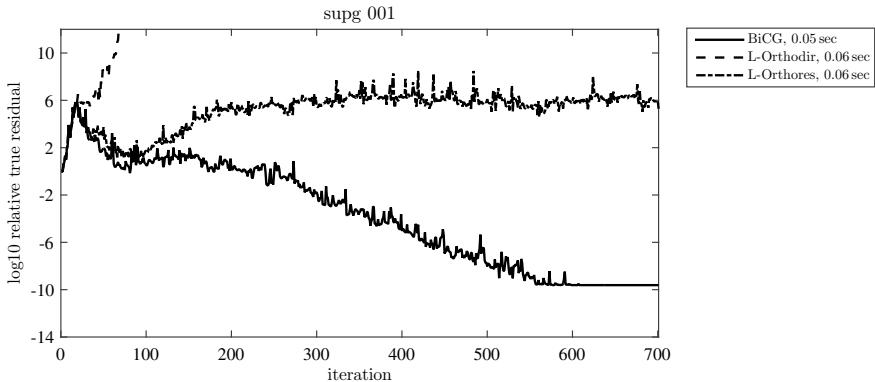


Fig. 8.5 supg 001, order 1225, relative true residual norms, BiCG (plain), Lanczos-Orthodir (dashed), Lanczos-Orthores (dot-dashed)

Table 8.3 Minimum relative true residual norms in at most 5000 iterations

matrix	BiCG	L-Orthodir	L-Orthores
add32	2.85275e-15	3.02142e-15	1.78499e-14
ex37	1.54595e-15	1.77874e-15	1.69681e-14
memplus	1.13169e-13	5.37353e-10	3.76046e-06
sherman3	4.61841e-07	2.78393e-05	1.00000e+00
wang4	5.27292e-12	2.16606e-04	7.48552e-09
supg 100	1.23889e-14	1.25318e-14	5.46230e-11
supg 1	4.14650e-04	6.65186e-04	4.46428e-04
supg 01	5.15819e-11	3.29782e-02	3.29782e-02
supg 001	1.54702e-01	1.54702e-01	1.54702e-01
supg 0001	6.47855e-01	6.47855e-01	6.47855e-01
supg 00001	1.00000e+00	1.00000e+00	1.00000e+00
supg 000001	1.00000e+00	1.00000e+00	1.00000e+00
convdiff xu 500	8.04236e-02	8.04236e-02	8.04236e-02
convdiff xu 1000	1.39383e-01	1.39097e-01	1.39381e-01
convdiff xu 5000	9.42723e-01	9.42723e-01	9.42723e-01

note that the following figures were obtained with our implementations and not with the codes described in [379, 382] or the Matlab implementation of QMR.

For `fs_183_6` Figure 8.6 shows that the residual norms of the two versions of Q-MR decrease very slowly and, even though there are large oscillations with BiCG, it converges faster. In fact, it takes 2500 iterations for Q-MR 2t (which is the best of the two QMR versions) to reach its maximum attainable accuracy which is $1.67 \cdot 10^{-15}$.

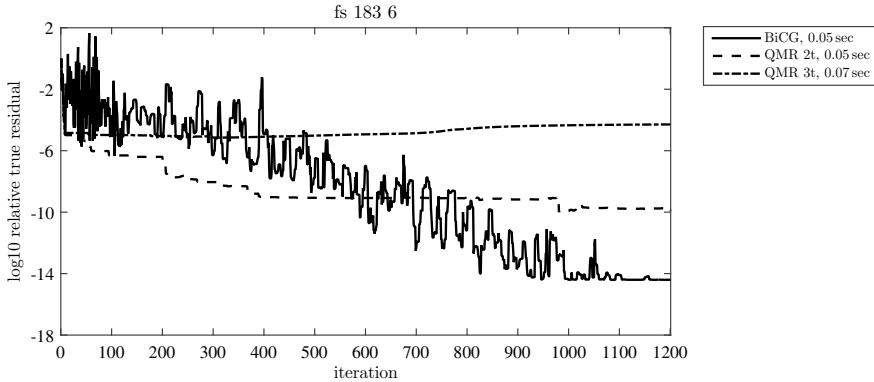


Fig. 8.6 *fs 183 6*, relative true residual norms, BiCG (plain), QMR 2t (dashed), QMR 3t (dot-dashed)

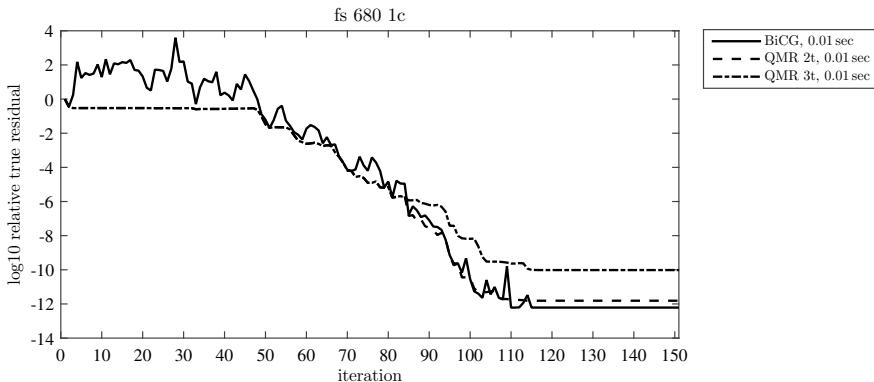


Fig. 8.7 *fs 680 1c*, relative true residual norms, BiCG (plain), QMR 2t (dashed), QMR 3t (dot-dashed)

For *fs 680 1c* the convergence of the three methods is approximately the same. The residual norms of the two Q-MR variants are, of course, smoother than those of BiCG and Q-MR 2t is better than Q-MR 3t as we may have guessed; see Figure 8.7.

In Figure 8.8 one can see that for *supg 001* of order 1225 Q-MR 2t converges much faster than BiCG and Q-MR 3t stagnates. We note that this does not happen if we use a random shadow vector or if we slightly perturb the shadow vector equal to b .

Table 8.4 shows the minimum relative true residual norms in at most 10000 iterations for BiCG, QMR 2t and QMR 3t. One can see that, in general, QMR 2t gives better results than QMR 3t. The accuracy of QMR 2t is roughly the same as that of BiCG.

Table 8.4 Minimal relative true residual norms in 10000 iterations at most

matrix	BiCG	Q-MR 2t	Q-MR 3t
pde225	2.23120e-15	3.96621e-15	3.97981e-13
gre 343	5.25537e-15	1.19057e-15	3.69278e-07
jphw 991	1.00000e+00	9.21304e-01	9.21304e-01
pde2961	4.25456e-12	2.92197e-12	2.69334e-10
jagmesh1	3.38975e-16	3.54956e-16	3.38557e-16
bawa782	1.56558e-13	1.83981e-13	1.75966e-11
dw2048	2.02265e-15	1.83384e-15	7.79838e-10
jagmesh2	6.70201e-16	6.49954e-16	5.77560e-16
raefsky2	8.56197e-14	9.25557e-14	2.14574e-09
fs 680 1c	6.09486e-13	1.52560e-12	9.64092e-11
add20	1.47896e-13	1.16289e-13	1.21101e-12
raefsky1	9.12030e-13	9.21259e-14	2.04252e-09
jagmesh4	7.20366e-16	7.06053e-16	5.62582e-16
fs 680 1	1.15296e-14	2.00631e-14	3.05655e-13
sherman1	6.24477e-14	5.75918e-14	1.45360e-13
nos3	1.70667e-14	1.65232e-14	2.46540e-13
sherman5	2.73876e-10	8.42731e-10	7.95982e-01
cavity05	2.42755e-03	9.18534e-04	9.38150e-04
e05r0500	8.63944e-04	2.33246e-03	4.56630e-01
comsol	4.48313e-11	5.21360e-11	6.26855e-08
olm1000	6.44749e-13	6.63670e-13	3.90722e-08
cavity10	1.17729e-13	1.12701e-13	1.33346e-06
steam2	5.41725e-14	6.71524e-14	5.69442e-11
1138bus	1.39110e-11	1.30265e-11	2.63290e-09
steam1	1.29588e-12	4.26214e-12	4.94343e-07
bcsstk26	1.84377e-07	1.18125e-08	1.66862e-08
nos7	4.02374e-07	3.22363e-07	1.98490e-01
watt1	9.08182e-14	1.00608e-11	3.66408e-02
bcsstk14	8.81060e-03	7.37935e-04	1.00836e-03
fs 183 6	3.96972e-15	1.66870e-15	7.13346e-06
bcsstk20	1.00000e+00	8.03685e-02	8.41425e-02
mfcf	1.00000e+00	8.76059e-01	8.64303e-01
nnc	1.00000e+00	8.13956e-01	8.29548e-01
Insp	1.00000e+00	9.98864e-01	9.98864e-01

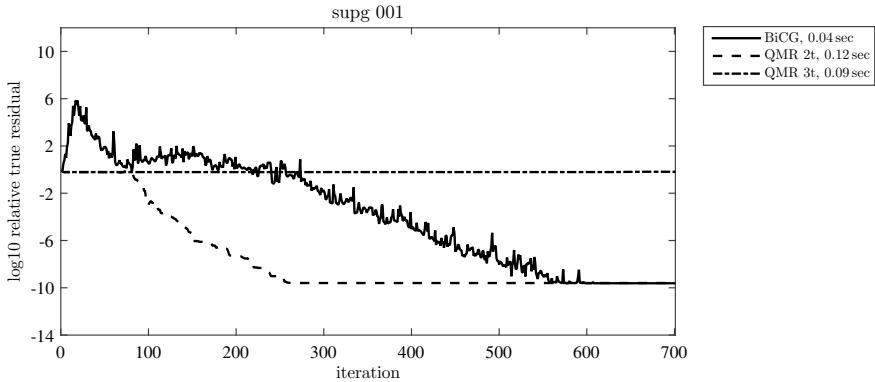


Fig. 8.8 supg 001, order 1225, relative true residual norms, BiCG (plain), QMR 2t (dashed), QMR 3t (dot-dashed)

Table 8.5 Minimal relative true residual norms in 10000 iterations at most

matrix	BiCG	Q-MR 2t	Q-MR 3t
add32	2.85275e-15	2.72896e-15	3.64867e-15
ex37	1.54595e-15	1.56241e-15	2.46570e-15
memplus	1.13169e-13	9.29851e-14	5.48982e-10
sherman3	6.34459e-11	3.29707e-05	7.82643e-07
wang4	5.27292e-12	1.08472e-11	8.81782e-09
supg 100	1.23889e-14	1.21026e-14	8.26342e-14
supg 1	4.14650e-04	4.49600e-04	4.09518e-04
supg 01	5.15819e-11	5.47258e-11	7.10205e-03
supg 001	1.54702e-01	5.49926e-02	5.49926e-02
supg 0001	6.47855e-01	5.30781e-01	5.30781e-01
supg 00001	1.00000e+00	9.04683e-01	9.04683e-01
supg 000001	1.00000e+00	9.84089e-01	9.84089e-01
convdif Xu 500	8.04236e-02	6.27688e-02	6.27697e-02
convdif Xu 1000	1.39383e-01	1.22812e-01	1.22843e-01
convdif Xu 5000	9.42723e-01	6.12779e-01	6.12779e-01

The conclusions are the same for the set of larger matrices in Table 8.5 except for sherman3. We observe that none of the three methods works well for the last seven problems.

8.11.3 Comparisons with GMRES

Let us illustrate the difficulties of comparing the convergence of iterative methods with different characteristics. We would like to compare full GMRES, BiCG and QMR. GMRES minimizes the residual norm at each iteration with only one matrix–vector product but the number of floating-point operations per iteration k is increasing with k . BiCG does not do any minimization, does two matrix–vector products per iteration including one with the transpose of the matrix but the work per iteration is constant with 13 vector operations. QMR with two-term recurrences minimizes the quasi-residual norm with two matrix–vector products at each iteration and does (in our implementation) 23 vector operations per iteration.

We choose the matrix `raefsky1` of order 3242 for this comparison. This sparse matrix has 293409 nonzero entries but it is banded and with a regular nonzero pattern. We do 350 iterations starting from $x_0 = 0$.

The worse way to compare the three methods is to plot the residual norms against the iteration number as it is done in Figure 8.9. This is not fair because the work done per iteration is not the same for the three methods. However, this plot shows that, for this problem, the convergence of the residual norms is very similar. We observe that the GMRES computing time is much larger than the computing times of BiCG and QMR.

A way of comparing iterative methods that is used in many papers is to plot the residual norms against the number of matrix–vector products. Here in GMRES we have one matrix–vector product per iteration and two for BiCG and QMR. This is supposed to be fair if the bulk of the computation is in the matrix–vector products. For our example, we obtain Figure 8.10. If we look at this figure we may think that GMRES is clearly better than BiCG and QMR.

However, this example is large enough to do reliable time measurements for each iteration. In Figure 8.11 we plot the residual norms against the computing time. Clearly BiCG and QMR are much faster than full GMRES, in fact about five times faster for this problem.

We must be careful about this conclusion because things may not be the same for other examples since the computing time depends on the ratio of the times spent in the matrix–vector products and the other parts of the iteration. Moreover, full GMRES cannot be used on larger examples because of time and memory limitations. GMRES has to be restarted as we will see in Chapter 11 but this has generally a negative impact on the convergence. There are also problems like `fs 183 6` for which BiCG and QMR do many more iterations than full GMRES to reach a given accuracy. Finally, when preconditioning is used, the time spent in the solve with the preconditioner comes also into play.

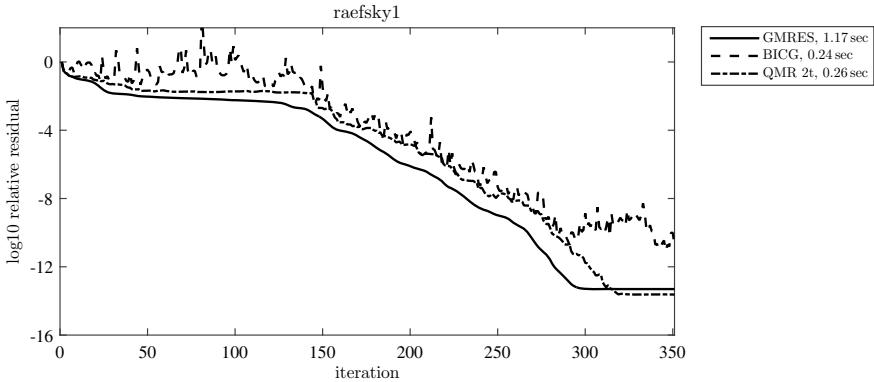


Fig. 8.9 *raefsky1*, relative residual norms versus iteration number, GMRES-MGS (plain), BiCG (dashed), QMR 2t (dot-dashed)

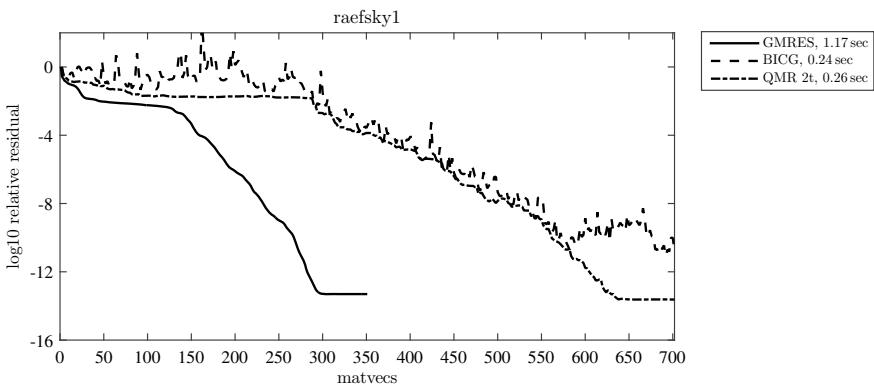


Fig. 8.10 *raefsky1*, relative residual norms versus number of matrix–vector products, GMRES-MGS (plain), BiCG (dashed), QMR 2t (dot-dashed)

8.11.4 Methods with look-ahead

Let us consider an example for which look-ahead techniques could help. The only nonzero entries of the matrix A are on the first subdiagonal with a value 1 and $a_{1,n} = -1$. The right-hand side is such that the exact solution has components $1, 2, \dots, n$; see [142]. We use $n = 20$ and $x_0 = 0$.

For this problem BiCG does not converge in 5000 iterations with very large oscillations of the residual norm. GMRES stagnates until $k = 20$ and then drops to the final residual norm. HMRZ-stab uses look-ahead and a jump of length 14 at iteration 3. There is a large drop of the residual norm to 10^{-8} and then a slow decrease; see Figure 8.12. Of course, as we have remarked in Subsection 8.11.3, this

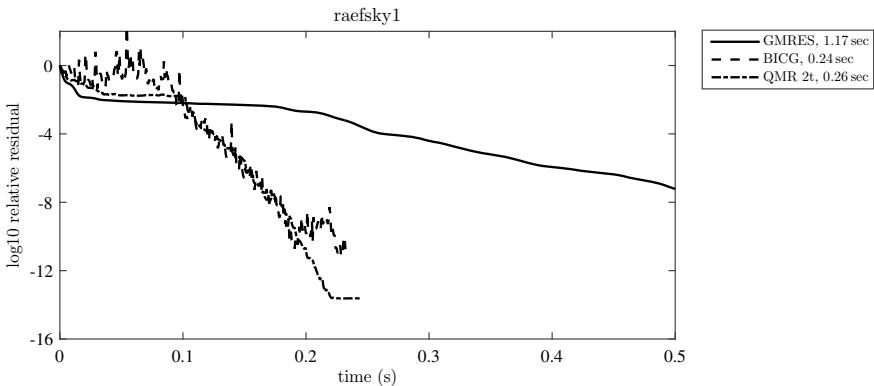


Fig. 8.11 *raefsky1*, relative residual norms versus computing time, GMRES-MGS (plain), BiCG (dashed), QMR 2t (dot-dashed)

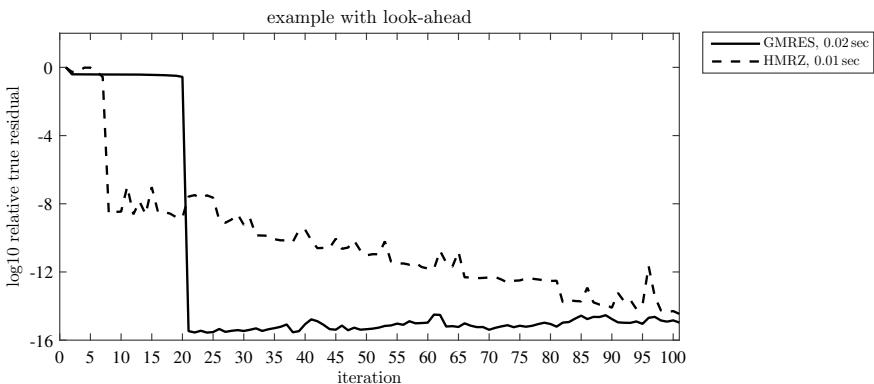


Fig. 8.12 Relative true residual norms, GMRES-MGS (plain), HMRZ-stab (dashed)

comparison is not really fair since we plotted the residual norms against the iteration number but our purpose was just to show how look-ahead can help.

8.11.5 A_i/B_j

For the Lanczos algorithms A_i/B_j only the methods based on polynomials p_k or $p_k^{(1)}$ eventually work in finite precision arithmetic. The methods based on the monomial basis are unstable. On our examples, of the methods we tried, only A_8/B_8 is working satisfactorily, even though it is not better than BiCG (which is roughly A_8/B_6).

8.12 Historical notes

The Lanczos algorithms were proposed by Cornelius Lanczos in 1950 to compute eigenvalues in [615] and 1952 for solving linear systems [616]. However, for symmetric matrices, the Lanczos algorithm is related to the Stieltjes procedure that was presented in 1884 by Thomas Stieltjes [889] to compute the coefficients of the three-term recurrence satisfied by orthogonal polynomials with respect to an inner product defined by a Riemann–Stieltjes integral; for more details see [432]. Curiously, concerning orthogonality, Lanczos did not refer to Gram or Schmidt but, in a footnote, to Otto Szász [902]: “The idea of the successive orthogonalization of a set of vectors was probably first employed by O. Szász, in connection with a determinant theorem of Hadamard”.

There is also an interesting footnote on the first page of [615] with a reference to Krylov: “The literature available to the author showed no evidence that the methods and results of the present investigation have been found before. However, A.M. Ostrowski of the University of Basle and the Institute for Numerical Analysis informed the author that his method parallels the earlier work of some Russian scientists; . . . (reference to Krylov and Luzin). . . On the basis of the reviews of these papers in the Zentralblatt, the author believes that the two methods coincide only in the point of departure. The author has not, however, read these Russian papers”.

Comments on the Lanczos’ papers by David M. Young and Kang C. Jea can be found in [561].

BiCG was proposed by Roger Fletcher in 1976 and published in the proceedings of the 1975 Dundee Conference on Numerical Analysis, see [355]. Originally the method was introduced to solve symmetric indefinite problems. A direct derivation of BiCG from the biorthogonality conditions was done by Petr Tichý and Jan Zítka [912] in 1998. Remarks on the choice of the shadow vector were given by P. Tichý [910] in 1999. Variants of BiCG were proposed by Claude Brezinski and Michela Redivo-Zaglia [136] in 1999. We observe that BiCG was used in 2008 by Zdeněk Strakoš and P. Tichý [895] to compute approximations of the bilinear form $c^*A^{-1}b$ where b and c are given vectors without solving the linear system $Ax = b$.

After their work on methods with long recurrences [1014] in 1980, David M. Young and Kang C. Jea considered Lanczos/Orthodir, Lanczos/Orthores and Lanczos/Orthomin [560] in 1983; see also Jea’s thesis [559]. They mentioned that Lanczos/Orthomin is essentially equivalent to BiCG and briefly review the work of other researchers. They stated that “the Lanczos/Orthomin method converges if and only if the Lanczos/Orthores method converges, and if both converge, then the Lanczos/Orthodir method converges and all three methods are equivalent. From this it would appear that the Lanczos/Orthodir method is the safest of the three”. This was a statement about the mathematical properties of the methods but, as we have seen, this may not hold in finite precision arithmetic.

The nonsymmetric Lanczos algorithm was considered by Yousef Saad [790] in 1982 as an oblique projection method. Feasibility conditions for the Lanczos

algorithm and BiCG using moment matrices were given as well as an analysis of convergence; see also [792].

Relations between the residual norms of BiCG and GMRES were obtained in [526] by Marlis Hochbruck and Christian Lubich in 1998.

The QMR algorithms were introduced at the beginning of the 1990s as an improvement of BiCG whose residual norm convergence is sometimes very oscillatory. The first paper [379] in 1991 was by Roland W. Freund and Noël M. Nachtigal. They used the three-term Lanczos nonsymmetric process (with look-ahead as we will see below); see also Nachtigal's thesis. A joint paper with Martin H. Gutknecht [373] was published in 1993. These papers followed the technical reports [371, 372, 378, 380, 381] from 1990 to 1992. An algorithm, supposed to be more stable, using two-term recurrences, was proposed [382] in 1994.

In [581] in 1992 Wayne D. Joubert characterized the breakdown situations in Lanczos algorithms. These characterizations are based on the Krylov matrices and not on the algorithmic details of each algorithm. For instance, Lanczos/Orthodir breaks down at iteration k if the matrix $K_k(A^T, \tilde{r}_0)^T A K_k(A, r_0)$ is singular. He also studied the likelihood of breakdowns using results on zeros of polynomials. The conclusion is that if A is complex, breakdowns cannot occur for the three Lanczos algorithms except for a set of vectors r_0 which is of measure zero in \mathbb{C}^n . For the same result to be true when A is real it is sufficient that A has at least one real eigenvalue.

The problem of breakdowns in the nonsymmetric Lanczos process was already considered in James H. Wilkinson's book [982] in 1965 on pages 388–394. The proposed remedy was to restart the algorithm. There is a whole zoo of names for the possible breakdowns in Lanczos algorithms and in BiCG. Beresford N. Parlett, Derek R. Taylor and Zhishun A. Liu [755] in 1985 said we have a *serious breakdown* if $w_k^T v_k = 0$ with v_k and w_k nonzero. This naming is attributed to Wilkinson but we did not find it in the pages devoted to the Lanczos process in [982]. Randolph E. Bank and Tony F. Chan [68, 69] in 1994 called this a *Lanczos breakdown* and said we have a *pivot breakdown* if the factorization of the tridiagonal matrix T_k (without pivoting) does not exist. W.D. Joubert [581] in 1992 defined a *hard breakdown* as $K_k(A^T, \tilde{r}_0)^T A K_k(A, r_0)$ being singular and a *soft breakdown* as $K_k(A^T, \tilde{r}_0)^T K(A, r_0)$ being singular. C. Brezinski and his co-authors [131, 141] in 1994 said we have a *true breakdown* if a formal orthogonal polynomial does not exist and a *ghost breakdown* if there is a division by zero in the relation for the computation of a polynomial. R.W. Freund and N.M. Nachtigal [379] in 1991 said we have a *breakdown of the first kind* in BiCG if $\tilde{p}_k^T A p_k = 0$ with $\tilde{p}_k, p_k \neq 0$ and a *breakdown of the second kind* if $\tilde{r}_k^T r_k = 0$ with $\tilde{r}_k, r_k \neq 0$. The only thing on which all these authors agree is the *incurable breakdown* which means that there is no way to cure it before we reach the grade of r_0 with respect to A .

In 1974 William B. Gragg [441] considered a linear functional on the space of polynomials whose values are given by the moments. He showed the links to formal Laurent series, the Padé table and the Lanczos nonsymmetric process as well as the definition of formal orthogonal polynomials (even though he did not call them in this way) through determinants. He was studying the regular case, but added in the

conclusion section that “The Lanczos polynomials may be generalized to maintain their connection with the Padé numerators and denominators, which are defined even though nontrivial blocks may occur in the Padé table”. This can be considered as an incentive to study the singular case. These ideas were used in a joint paper with Anders Lindquist [443] in 1983 on the partial realization problem in systems theory.

There was a lot of activity in the 1980s and 1990s concerned with look-ahead techniques. It seems that the name “look-ahead” was coined by B.N. Parlett, D.R. Taylor and Z.A. Liu [755]; see also D.R. Taylor’s thesis [907] in 1982. In this work they used 2×2 diagonal blocks when needed to factorize the tridiagonal matrix in the nonsymmetric Lanczos algorithm. They were mainly interested in eigenvalue computations.

Then, we have to move to the beginning of the 1990s to see papers concerned with look-ahead. There were mainly two groups of people interested in these problems. The first one was composed of M.H. Gutknecht, R.W. Freund and N.M. Nachtigal. The first paper on QMR with look-ahead [379] was published in 1991 but the report from RIACS [371] appeared in 1990, see also N.M. Nachtigal’s thesis [710] in 1991. The algorithm is based on the theoretical work by M.H. Gutknecht [478, 481] which was published in 1992 and 1994. A paper about the implementation of the algorithm [373] based on a report from 1991 was published in 1993. Freund and Nachtigal published the two-term recurrence version of QMR with look-ahead [382] in 1994. The details of its implementation are given in [380]. The QMRPACK package of Fortran subroutines implementing these algorithms is still available in Netlib.

The second group was composed of C. Brezinski, M. Redivo-Zaglia and Hassane Sadok. Except in [143] where a comparison is done with the matrix approach, they advocated the use of the formal orthogonal polynomial framework. These polynomials were considered in the thesis of Herman van Rossum [949] in 1953, but not under this name. They also appeared in a paper by Peter Wynn [991] in 1967 and a paper by C. Brezinski [123] in 1979. According to the introduction of André Draux’s thesis [276] it was C. Brezinski who suggested the name *formal orthogonal polynomials* instead of generalized orthogonal polynomials as they were sometimes named before, see [124, 130].

A first paper [138] was published in 1991 but what described the beginning of the work of this group on look-ahead techniques [140] was only published in 1992. In this paper they introduced the MRZ algorithm which was intended to cure exact breakdowns in the Lanczos algorithm. To jump across non-existing polynomials they used the relations that were exhibited by A. Draux [276] in 1983; see also [277]. In the 1991 paper they considered also near-breakdowns and introduced several variants BMRZ, SMRZ and BSMRZ and give some numerical results on small problems; see also [139]. Unfortunately, these methods use the monomial (natural) basis for $\mathcal{K}_k(A^T, \tilde{r}_0)$ and therefore do not give good results on large problems. Another discussion of these methods was given in [141] in 1997. In 1999 they proposed to use other bases for $\mathcal{K}_k(A^T, \tilde{r}_0)$ in [142]. These methods are denoted MRZ-stab, HMRZ and HMRZ-stab. The last two methods use also the Horner scheme to evaluate some polynomials. Successive multiplications by A^T were still used but just to find the

lengths of the jumps, which are in general small. Moreover, the storage is independent of the lengths of the jumps. HMRZ-stab gives much better numerical results than, for instance, MRZ. A review of formal orthogonality in Lanczos-based methods [144] was published in 2002.

A systematic study of the relations that can be used for the formal orthogonal polynomials and the adjacent family was done in 1994 in the thesis of Carole Baheux [53] who was a student of C. Brezinski; see also [54].

There were a few other papers dealing with the breakdown problems. An interesting paper by B.N. Parlett about reduction to tridiagonal form and, in particular, dealing with incurable breakdowns is [752] in 1992.

W.D. Joubert [580, 581] in 1990 proposed to switch between Lanczos/Orthomin and Lanczos/Orthodir or to use the Parlett, Taylor and Liu algorithm.

Daniel L. Boley, Sylvan Elhay, Gene H. Golub and M.H. Gutknecht [106] in 1991 considered the problem of computing a set of indefinite weights for a discrete inner product and the associated orthogonal polynomials given a set of initial moments or modified moments. They described a modified nonsymmetric Lanczos process which recovers when there is a breakdown. The biorthogonality is only enforced in a blockwise sense.

R.E. Bank and T.F. Chan [68, 69] in 1993–1994 proposed a variant of BiCG in which they used 2×2 blocks in case of pivot breakdown. The resulting algorithm is called the composite step biconjugate gradient (CSBCG).

In 1994 Q. Ye [1006] proposed another way to deal with breakdowns. When one occurs, the length of the recurrence using A^T is increased in such a way that biorthogonality relations are maintained. The matrix that is obtained is no longer tridiagonal but banded.

To cure a serious breakdown, Thomas Huckle [538] in 1995 used a rank-one modification of the matrix A . The interest of this approach is that it can be used with any of the iterative methods we have reviewed in this chapter.

In 1997 Peter R. Graves-Morris [446] proposed a technique he called “look-around Lanczos”. In his words “The algorithms proposed use a method based on selecting well-conditioned pairs of neighbouring polynomials (in the associated Padé table), and the method is equivalent to going round the blocks instead of going across them, as is done in the well-known look-ahead methods”.

El Hassan Ayachour [47] in 1998–1999 introduced new biorthogonal polynomials satisfying a three-term recurrence relation instead of jumping over non-existing blocks; see also Ayachour’s thesis [46]. He made use of the intermediate biorthogonal polynomials for computing the regular orthogonal ones. This technique is applied to Padé approximation, extrapolation methods and Lanczos algorithms.

It is interesting to note that, after the year 2000, there is almost no paper considering the treatment of (near-)breakdowns in Krylov methods. This field of research seems completely abandoned. This is surprising since the problems of deciding when we have to cure a near-breakdown or of maintaining a sufficient level of biorthogonality (or at least linear independence) in the basis vectors have not been completely solved yet.

The problem of prescribing BiCG convergence was partly considered by Jane Cullum [244] in 1996. She proved that given the residual norms obtained with BiCG on $Ax = b$ one can construct a matrix B with the same eigenvalues as A and a right-hand side c such that the FOM residual norms when solving $Bx = c$ are the same as those obtained in the BiCG computation.

In 2016 the authors of this book [289] showed that any residual norm history with finite residual norms is possible for the BiCG method with any nonzero eigenvalues of the matrix A . Any decreasing residual norm history is possible for the QMR method with any nonzero eigenvalues of the system matrix. The approach is constructive but prescribing infinite residual norms in BiCG is still an open problem.

Zhaojun Bai [55] in 1994 studied the nonsymmetric Lanczos process in finite precision arithmetic. This work was in the same spirit as Christopher C. Paige's work on the symmetric Lanczos algorithm [730]. In 2000 Charles H. Tong and Qiang Ye [921] gave an analysis of BiCG in finite precision arithmetic. More recently, in 2014, C.C. Paige, Ivo Panayotov and Jens-Peter M. Zemke [734] published an “augmented” rounding error analysis of the nonsymmetric Lanczos process. All these papers stressed the difficulty of obtaining good bounds because, in case of (near-)breakdown some quantities grow unbounded.

Concerning parallel computing, early versions of s -step nonsymmetric Lanczos algorithms [596, 597] were proposed in 1991–1992 by Sun Kyung Kim and Anthony T. Chronopoulos. At the same time, first versions of parallel QMR algorithms [374, 375] were presented by Roland W. Freund and Marlis Hochbruck. A parallel version of BiCG was published [156, 157] by Hans Martin Bücker and Manfred Sauren in 1997 and 1999. A slight variation of the same method [1003, 1004] was used under the name *improved BiCG* by Laurence T. Yang and Richard P. Brent in 2003.

Communication-avoiding nonsymmetric Lanczos and BiCG methods were briefly described in 2010 by Mark F. Hoemmen in [527]. CA-BiCG was derived from CA-Lanczos [67] in 2014 by Grey Ballard, Erin Carson, James Demmel, M. Hoemmen, Nicholas Knight and Oded Schwartz. A direct derivation of CA-BiCG [188] was done by Erin Carson in 2015.

Chapter 9

Transpose-free Lanczos methods



The Lanczos methods studied in Chapter 8 have two shortcomings. First, they can suffer from breakdowns, but this can be solved, in some cases, by using look-ahead techniques or changing the shadow vector. Second, in each iteration we need to do a matrix–vector product with A^T (or A^*). This can be a problem if the transpose (or conjugate transpose) matrix is not easily available. This could be the case, for instance, on parallel computers where the data for the matrix A is distributed in the local memories or in matrix-free implementations where the matrix–vector product is replaced with a difference operator.

In this chapter we study iterative methods, derived from those of Chapter 8, that do not need a multiplication with the transpose of A . They are sometimes called *product-type* or *transpose-free* methods. These methods are based on the straightforward but useful remark that if we have two polynomials p and q with real coefficients then,

$$(q(A^T)\tilde{r}_0, p(A)r_0) = (\tilde{r}_0, q(A)p(A)r_0). \quad (9.1)$$

Hence, we can understand where the name of this kind of methods is coming from. For the methods we have described in Chapter 8, this remark can only be used if we know how to derive recurrence relations for the product of the polynomials. In this chapter we assume, most of the time, that the data is real.

9.1 CGS

The first product-type method that was proposed was named Conjugate Gradient Squared (CGS) [861]. It is based on the BiCG method we have described in Section 8.2. Therefore, it would have been better to name this method BiCGS. We

will use that name in the sequel, except in the historical notes. For BiCG it can be easily proved by induction that the residual and the direction vectors are given by polynomials in A and A^T applied to the initial residuals,

$$\begin{aligned} r_k &= \phi_k(A)r_0, & p_k &= \theta_k(A)r_0, \\ \tilde{r}_k &= \phi_k(A^T)\tilde{r}_0, & \tilde{p}_k &= \theta_k(A^T)\tilde{r}_0, \end{aligned}$$

where ϕ_k and θ_k are polynomials of degree less than or equal to k , see Section 8.2. They satisfy the following coupled recurrences,

$$\phi_{k+1}(\xi) = \phi_k(\xi) - \gamma_k \xi \theta_k(\xi), \quad \theta_{k+1}(\xi) = \phi_{k+1}(\xi) + \mu_{k+1} \theta_k(\xi), \quad (9.2)$$

see (8.4), (8.7). In BiCG we have to compute expressions like

$$(\tilde{r}_k, r_k) = (\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0) = (\tilde{r}_0, [\phi_k(A)]^2 r_0).$$

But, from relation (9.2),

$$[\phi_k(A)]^2 = [\phi_{k-1}(A)]^2 + \gamma_{k-1}^2 A^2 [\theta_{k-1}(A)]^2 - 2\gamma_{k-1} A \phi_{k-1}(A) \theta_{k-1}(A). \quad (9.3)$$

Hence, we need to find expressions for the last polynomial in the right-hand side. We also have

$$[\theta_k(A)]^2 = [\phi_k(A)]^2 + \mu_k^2 [\theta_{k-1}(A)]^2 + 2\mu_k \phi_k(A) \theta_{k-1}(A), \quad (9.4)$$

and we have also to find relations for $\phi_k(A) \theta_{k-1}(A)$.

Proposition 9.1 *For $k > 1$,*

$$\begin{aligned} \phi_{k-1}(A) \theta_{k-1}(A) &= [\phi_{k-1}(A)]^2 + \mu_{k-1} \phi_{k-1}(A) \theta_{k-2}(A), \\ \phi_{k-1}(A) \theta_{k-2}(A) &= \phi_{k-2}(A) \theta_{k-2}(A) - \gamma_{k-2} A [\theta_{k-2}(A)]^2. \end{aligned}$$

Proof For the first relation we multiply

$$\theta_{k-1}(A) = \phi_{k-1}(A) + \mu_{k-1} \theta_{k-2}(A),$$

with $\phi_{k-1}(A)$ on the left. For the second relation, we multiply

$$\phi_{k-1}(A) = \phi_{k-2}(A) - \gamma_{k-2} A \theta_{k-2}(A),$$

by $\theta_{k-2}(A)$ on the right. □

We want to construct a method such that the residual vector is $r_k = [\phi_k(A)]^2 r_0$. To do so, changing the notation we had for BiCG, we introduce new vectors

$$\begin{aligned} r_k &= [\phi_k(A)]^2 r_0, \\ p_k &= [\theta_{k-1}(A)]^2 r_0, \\ u_k &= \phi_{k-1}(A) \theta_{k-1}(A) r_0, \\ q_k &= \phi_k(A) \theta_{k-1}(A) r_0. \end{aligned}$$

Note that $\phi_k(A)r_0$ is the residual vector we would have obtained from the BiCG algorithm starting from r_0 . Then, we apply again the same polynomial but the resulting residual vector is not what we would have obtained after k iterations by running BiCG starting from $\phi_k(A)r_0$.

Proposition 9.2 *The vectors defined above satisfy the following recurrence relations,*

$$\begin{aligned} r_k &= r_{k-1} - \gamma_{k-1} A(u_k + q_k), \\ p_k &= u_k + \mu_{k-1} [\mu_{k-1} p_{k-1} + q_{k-1}], \\ u_k &= r_{k-1} + \mu_{k-1} q_{k-1}, \\ q_k &= u_k - \gamma_{k-1} A p_k. \end{aligned}$$

Proof From Proposition 9.1 and (9.3), (9.4), these vectors satisfy the following recurrence relations,

$$\begin{aligned} r_k &= r_{k-1} + \gamma_{k-1}^2 A^2 p_k - 2\gamma_{k-1} A u_k = r_{k-1} - \gamma_{k-1} A [2u_k - \gamma_{k-1} A p_k], \\ p_k &= r_{k-1} + \mu_{k-1}^2 p_{k-1} + 2\mu_{k-1} q_{k-1} = r_{k-1} + \mu_{k-1} [\mu_{k-1} p_{k-1} + 2q_{k-1}], \\ u_k &= r_{k-1} + \mu_{k-1} q_{k-1}, \\ q_k &= u_k - \gamma_{k-2} A p_{k-1}. \end{aligned}$$

Then,

$$r_k = r_{k-1} - \gamma_{k-1} A(u_k + q_k),$$

and

$$p_k = u_k + \mu_{k-1} [\mu_{k-1} p_{k-1} + q_{k-1}]. \quad \square$$

We need to see how to compute the BiCG coefficients γ_k and μ_k , see (8.14), resp. (8.15). Since we have changed the notation, let $r_k^B = \phi_k(A)r_0$, $p_k^B = \theta_k(A)r_0$ (and similar definitions for the vectors with a tilde) be the BiCG vectors. To compute μ_k and the numerator of γ_k we need the dot product of residual vectors,

$$\rho_k = (\tilde{r}_k^B, r_k^B) = (\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0) = (\tilde{r}_0, [\phi_k(A)]^2 r_0) = (\tilde{r}_0, r_k),$$

and $\mu_k = \rho_k / \rho_{k-1}$. To compute γ_k we also need

$$\begin{aligned} (\tilde{p}_k^B, Ap_k^B) &= (\theta_k(A^T)\tilde{r}_0, A\theta_k(A)r_0) = (\tilde{r}_0, \theta_k(A)A\theta_k(A)r_0) \\ &= (\tilde{r}_0, A[\theta_k(A)]^2r_0) = (\tilde{r}_0, Ap_{k+1}). \end{aligned}$$

Hence, we see that we can easily compute the coefficients provided we store \tilde{r}_0 . We do not need the other BiCG vectors \tilde{r}_k^B and \tilde{p}_k^B . Note that since the residual vectors are defined as $r_k = [\phi_k(A)]^2 r_0$, BiCGS has the finite termination property.

Rearranging the formulas gives the following code for BiCGS. In this code we use $\tilde{r}_0 = r_0$ but this is not mandatory. We can choose this shadow vector as we wish.

```
function [x,ni,resn,resnt] = BiCGS(A,b,x0,epsi,nitmax);
%
nb = norm(b);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resnt = zeros(1,nitmax+1);
resn(1) = norm(r);
resnt(1) = resn(1);
p = r;
u = r;
r0 = r;
Ap = A * p;
rho0 = transpose(r0) * r;
ni = 0;
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    gamma = rho0 / (transpose(Ap) * r0);
    q = u - gamma * Ap;
    qu = q + u;
    x = x + gamma * qu;
    Aqu = A * qu; % matrix vector product
    r = r - gamma * Aqu;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    resnt(ni+1) = norm(b - A * x);
    if nresidu < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nresidu
    rho = transpose(r) * r0;
    mu = rho / rho0;
    rho0 = rho;
    u = r + mu * q;
    p = u + mu * (q + mu * p);
```

```

Ap = A * p; % matrix vector product
end % for k
resn = resn(1:ni+1);
resnt = resnt(1:ni+1);

```

BiCGS needs two matrix–vector products per iteration as BiCG does but the transpose (or conjugate transpose) of A is not required. The number of vector operations per iteration is slightly larger than in BiCG (13 instead of 10 in our implementations) and we have also two dot products per iteration. It is as easy to code as BiCG and if BiCG converges well, the convergence of BiCGS is better. This means that, in some cases, we may have a better reduction of the residual norm for almost the same amount of work as in BiCG. However, it is well known that the residual norms of BiCG can be very oscillatory; see Chapter 8. In this case the BiCGS residual norms are also oscillatory and the magnitude of the oscillations is larger since the residual polynomial has been squared. In some problems the magnitude of the oscillations can be so large that this may spoil the approximate solution in finite precision arithmetic. We also note that, like BiCG, BiCGS may have (near-) breakdowns if one or both of the denominators are zero or too small.

We observe that it will not be fair to compare the results of BiCGS and say, GMRES, on the basis of the iteration number since at a given iteration k they do not use the same Krylov subspace. By construction the residual r_k of BiCGS is a polynomial of degree $2k$ in A applied to r_0 . Hence, $x_k - x_0$ is a polynomial of degree $2k - 1$ applied to r_0 . It means that $x_k \in x_0 + \mathcal{K}_{2k-1}(A, r_0)$ whence the k th GMRES iterate is in $x_0 + \mathcal{K}_k(A, r_0)$. Therefore, we have to be careful when comparing different Krylov methods if they are not using Krylov subspaces of the same dimension.

The sequence of the BiCGS residual vectors does not give a basis of the Krylov subspaces since they belong only to the subspaces of even indices. If we want, for any reason, to build a basis, we have to consider other vectors to fill the gaps. The vectors q_k are appropriate since they are given by $\phi_k(A)\theta_{k-1}(A)r_0$ and, therefore, they belong to $\mathcal{K}_{2k-1}(A, r_0)$. Hence, a basis can be

$$\{r_0, q_1, r_1, q_2, r_2, \dots\},$$

or linear combinations of these vectors as long as they are independent.

Theorem 9.1 *Assume that there is no breakdown in BiCGS, that $2k + 1$ is smaller than the grade of r_0 with respect to A and let*

$$V_{n,m} = (r_0 \ q_1 \ \cdots \ r_{k-1} \ q_k) \text{ if } m = 2k,$$

$$V_{n,m} = (r_0 \ q_1 \ \cdots \ q_k \ r_k) \text{ if } m = 2k + 1.$$

Then, every second residual vector of the Q-OR method using the columns of $V_{n,m}$ as basis vectors is mathematically equal to a BiCGS residual vector.

Proof This result may seem almost trivial since Q-OR residual vectors are proportional to the basis vectors. But our goal is to prove that for every second Q-OR vector the coefficient of proportionality is equal to 1.

Our first goal is to write down an Arnoldi-like relation for the basis vectors. Since the BiCGS relations are

$$\begin{aligned} r_k &= r_{k-1} - \gamma_{k-1} A(u_k + q_k), \\ p_k &= u_k + \mu_{k-1} [\mu_{k-1} p_{k-1} + q_{k-1}], \\ u_k &= r_{k-1} + \mu_{k-1} q_{k-1}, \\ q_k &= u_k - \gamma_{k-1} A p_k, \end{aligned}$$

we need to express r_k and q_k only in terms of the other vectors r_j, q_j , that is, we have to eliminate the u_j 's and p_j 's. This can be done easily for the relation for r_k since we just have to replace u_k by its definition,

$$r_k = r_{k-1} - \gamma_{k-1} A(r_{k-1} + \mu_{k-1} q_{k-1} + q_k). \quad (9.5)$$

It is more difficult to eliminate p_k in the relation for q_k . To do this we have to use the relation for p_{k-1} and we have to write a long recurrence to eliminate all the p_j 's. In the end we obtain for p_k a relation involving all the vectors r_j and q_j with $j = k-1$ to 0. It yields

$$p_k = \sum_{i=1}^k \left[\prod_{j=1}^{i-1} \mu_{k-j}^2 \right] (r_{k-i} + 2\mu_{k-i} q_{k-i}).$$

The last term in the sum is the product of the μ_j^2 's up to μ_1^2 times r_0 . Writing relation (9.5) in a different order we have

$$A(r_{k-1} + \mu_{k-1} q_{k-1} + q_k) = \frac{1}{\gamma_{k-1}} (r_{k-1} - r_k).$$

This can be written as

$$A(q_{k-1} \ r_{k-1} \ q_k) \begin{pmatrix} \mu_{k-1} \\ 1 \\ 1 \end{pmatrix} = (q_{k-1} \ r_{k-1} \ q_k \ r_k) \begin{pmatrix} 0 \\ 1/\gamma_{k-1} \\ 0 \\ -1/\gamma_{k-1} \end{pmatrix}.$$

Initially, we have

$$A(r_0 + q_1) = \frac{1}{\gamma_0} (r_0 - r_1) \Rightarrow A(r_0 \ q_1) \begin{pmatrix} 1 \\ 1 \end{pmatrix} = (r_0 \ q_1 \ r_1) \begin{pmatrix} 1/\gamma_0 \\ 0 \\ -1/\gamma_0 \end{pmatrix}.$$

Let $\nu_i^{(k)} = \prod_{j=1}^{i-1} \mu_{k-j}^2$ and $\nu_1^{(k)} = 1$. Using the relation for q_k , we have

$$A \left(\sum_{i=1}^k \nu_i^{(k)} (r_{k-i} + 2\mu_{k-i} q_{k-i}) \right) = \frac{1}{\gamma_{k-1}} (r_{k-1} + \mu_{k-1} q_{k-1} - q_k).$$

The expression within parenthesis on the left-hand side can be written as the matrix $(r_0 \ q_1 \ \dots \ q_{k-1} \ r_{k-1})$ times a full vector of coefficients. The right-hand side is the product of the matrix $(r_0 \ q_1 \ \dots \ q_{k-1} \ r_{k-1} \ q_k)$ times a sparse vector with only three nonzero components. Initially we have

$$Ar_0 = \frac{1}{\gamma_0} (r_0 - q_1),$$

and

$$A (r_0 \ q_1 \ r_1) \begin{pmatrix} \nu_2^{(2)} \\ 2\mu_1 \nu_1^{(2)} \\ \nu_1^{(2)} \end{pmatrix} = (r_0 \ q_1 \ r_1 \ q_2) \begin{pmatrix} 0 \\ \mu_1/\gamma_1 \\ 1/\gamma_1 \\ -1/\gamma_1 \end{pmatrix}.$$

Using the definition of $V_{n,m}$, the relations we have derived above can be written in matrix form as

$$AV_{n,m}R_m = V_{n,m+1}\underline{\tilde{H}}_m,$$

where R_m is a square upper triangular matrix and $\underline{\tilde{H}}_m$ is an $(m+1) \times m$ tridiagonal matrix. This has been called a generalized Hessenberg relation in [1022]. To be clear, let us write the first columns of $\underline{\tilde{H}}_m$ and R_m ,

$$\underline{\tilde{H}}_m = \begin{pmatrix} \frac{1}{\gamma_0} & \frac{1}{\gamma_0} & & & \\ -\frac{1}{\gamma_0} & 0 & \frac{\mu_1}{\gamma_1} & & \\ & -\frac{1}{\gamma_0} & \frac{1}{\gamma_1} & \frac{1}{\gamma_1} & \\ & & -\frac{1}{\gamma_1} & 0 & \frac{\mu_2}{\gamma_2} \\ & & & -\frac{1}{\gamma_1} & \frac{1}{\gamma_2} \\ & & & & \ddots \\ & & & & -\frac{1}{\gamma_2} & \ddots \\ & & & & & \ddots \end{pmatrix},$$

$$R_m = \begin{pmatrix} 1 & 1 & \nu_2^{(2)} & 0 & \nu_3^{(3)} & 0 & \dots \\ & 1 & 2\mu_1 & \mu_1 & 2\mu_1\nu_2^{(3)} & 0 & \dots \\ & & 1 & 1 & \nu_2^{(3)} & 0 & \dots \\ & & & 1 & 2\mu_2 & \mu_2 & \dots \\ & & & & 1 & 1 & \dots \\ & & & & & 1 & \ddots \\ & & & & & & \ddots \end{pmatrix}.$$

Note that the tridiagonal matrix \underline{H}_m can be factorized as

$$\underline{H}_m = \underline{H}_m^{(\mu)} [D^{(\gamma)}]^{-1} = \begin{pmatrix} 1 & 1 & & & & & \\ -1 & 0 & \mu_1 & & & & \\ & -1 & 1 & 1 & & & \\ & & -1 & 0 & \mu_2 & & \\ & & & -1 & 1 & \ddots & \\ & & & & -1 & \ddots & \\ & & & & & \ddots & \end{pmatrix} [D^{(\gamma)}]^{-1},$$

where $D^{(\gamma)}$ is a diagonal matrix with diagonal entries $\gamma_0, \gamma_0, \gamma_1, \gamma_1, \dots$. Let $\tilde{H}_m^{(\mu)}$ be the matrix of the first m rows of $\underline{H}_m^{(\mu)}$. For m even, the inverse of $\tilde{H}_m^{(\mu)}$ has a very special sparsity structure. In particular the even entries of the first column are equal to 1 and the odd entries are zero (see, e.g., Theorem 2.2) For m odd the inverse of $\tilde{H}_m^{(\mu)}$ is a dense matrix. But, let us define

$$f_1 = 1, \quad f_k = 1 + \mu_{k-1} f_{k-1}, \quad k = 1, \dots$$

Then, for $m = 2k + 1$, the last two entries of the first column of the inverse of $\tilde{H}_m^{(\mu)}$ are $1/f_{k+1}$. For $m = 2k$ we can write explicitly the solution $y^{(m)}$ of $\tilde{H}_m^{(\mu)} [D^{(\gamma)}]^{-1} R_m^{-1} y^{(m)} = e_1$. First, we solve $\tilde{H}_m^{(\mu)} [D^{(\gamma)}]^{-1} z = e_1$ whose solution is

$$z = \begin{pmatrix} 0 \\ \gamma_0 \\ 0 \\ \gamma_1 \\ 0 \\ \gamma_2 \\ 0 \\ \vdots \end{pmatrix}.$$

Then, $y^{(m)} = R_m z$ and, because of the sparsity structure of R_m , we obtain

$$y^{(m)} = \begin{pmatrix} \gamma_0 \\ \gamma_0 + \mu_1 \gamma_1 \\ \gamma_1 \\ \gamma_1 + \mu_2 \gamma_2 \\ \gamma_2 \\ \vdots \\ \gamma_{k-1} + \mu_{k-1} \gamma_k \\ \gamma_k \\ \gamma_k \end{pmatrix}.$$

Since the diagonal entries of the upper triangular matrix R_m are equal to 1, we can write the Arnoldi-like relation,

$$AV_{n,m} = V_{n,m+1} \underline{H}_m R_m^{-1} = V_{n,m+1} \underline{H}_m,$$

where \underline{H}_m is a $(m+1) \times m$ upper Hessenberg matrix whose entries depend only on the BiCG coefficients.

Let \tilde{H}_m be the matrix of the m first rows of \underline{H}_m and $H_m = \tilde{H}_m R_m^{-1}$. The matrix \tilde{H}_m is tridiagonal. The Q-OR method is obtained by solving $H_m y^{(m)} = e_1$ (since the basis vectors are not normalized and $V_{n,m} e_1 = r_0$) and setting $x_m = x_0 + V_{n,m} y^{(m)}$. The residual vectors are $r_m = -h_{m+1,m} [y^{(m)}]_m v_{m+1}$. Hence, as usual, the residual vectors of Q-OR are proportional to the basis vectors. The even values of m correspond to the computation of the BiCGS residual vectors. Since $(R_m)_{m,m} = 1$, we have $h_{m+1,m} = \tilde{h}_{m+1,m} = -1/\gamma_k$ which is, as we have seen above, the negative of the reciprocal of the last component of $y^{(m)}$. This shows that every second residual vector of the Q-OR method is equal to a BiCGS residual. The other residuals are only proportional to the vectors q_j . \square

The Q-OR basis has no particular properties except that every second vector (proportional to q_j) is orthogonal to \tilde{r}_0 . Of course, the Q-OR method equivalent to BiCGS has only a theoretical interest since BiCGS is a much more efficient algorithm in terms of storage. However, since we have a basis given by the columns of $V_{n,m}$ we can also construct a Q-MR method by minimizing the quasi-residual norms. In fact, to do so it is easier to also use $\hat{V}_{n,m} = V_{n,m} R_m$ as a second basis and to define a generalized Q-MR algorithm. Then, we have

$$A\hat{V}_{n,m} = V_{n,m+1} \underline{H}_m.$$

Since $\hat{V}_{n,1} = V_{n,1} = r_0$, if we define the iterates as $x_m = x_0 + \hat{V}_{n,m} y^{(m)}$, we have

$$r_m = r_0 - A\hat{V}_{n,m} y^{(m)} = V_{n,m+1} (e_1 - \underline{H}_m y^{(m)}).$$

We can compute $y^{(m)}$ to minimize the quasi-residual norm $\|e_1 - \underline{H}_m y\|$. Since \underline{H}_m is tridiagonal, this can be done as in the QMR algorithm [379]. The approximate

solution x_m can be updated by a short recurrence from x_{m+1} . However, the drawback is that, to do so, we need the last basis vector \hat{v}_m which is not easily available without storing all the basis vectors. A more clever way to use the BiCGS basis vectors is used in the TFQMR algorithm [369] that we will briefly review in Section 9.3.

The fact that BiCGS is a particular Q-OR method allows us to use the theoretical results we have described in Chapter 3. For some of them we need that the basis vectors be of unit norm. Therefore, consider

$$AV_{n,m}D_m^{-1} = V_{n,m+1}D_{m+1}^{-1}D_{m+1}\underline{H}_mD_m^{-1},$$

where the diagonal entries of D_m are the norms of the columns of $V_{n,m}$. In particular, we obtain that the zeros of the BiCGS residual polynomial are the eigenvalues of the matrices $\tilde{H}_mR_m^{-1}$, that is, they are given by a generalized eigenvalue problem with a tridiagonal and an upper triangular matrix. Note that, since the BiCGS residual polynomial is the square of the BiCG residual polynomial, the matrices $\tilde{H}_mR_m^{-1}$ have only double eigenvalues which are equal to the roots of the BiCG residual polynomial.

We also have a relation between the norms of the BiCG and BiCGS residual vectors.

Proposition 9.3 *Let r_k (resp. r_k^B) be the BiCGS (resp. BiCG) residual vectors at iteration k , ϕ_k be the BiCG residual polynomial which is one at the origin and ψ_{k-1} the polynomial of degree $k - 1$ such that $\psi_{k-1}(\xi) = \frac{1}{\xi}(\phi_k(\xi) - 1)$. Then, a necessary condition for the BiCGS residual norm to be smaller than the BiCG residual norm is*

$$(r_k^B, A\psi_{k-1}(A)r_k^B) \leq 0,$$

that is,

$$(r_k^B, \phi_k(A)r_k^B) = (r_k^B, r_k) \leq \|r_k^B\|^2.$$

Proof We have

$$\begin{aligned} r_k &= [\phi_k(A)]^2 r_0, \\ &= \phi_k(A)r_k^B, \\ &= (I + \phi_k(A) - I)r_k^B, \\ &= r_k^B + A\psi_{k-1}(A)r_k^B. \end{aligned}$$

Therefore,

$$\|r_k\| = \|r_k^B\|^2 + 2(r_k^B, A\psi_{k-1}(A)r_k^B) + \|A\psi_{k-1}(A)r_k^B\|^2,$$

and it is necessary to have $(r_k^B, A\psi_{k-1}(A)r_k^B) \leq 0$ to have a residual norm reduction. \square

Another derivation of BiCGS was given in [299] starting from the nonsymmetric Lanczos algorithm. However, this derivation used three-term recurrences and the algorithm may be less stable than the classical BiCGS algorithm; see also [298].

For a derivation and the properties of BiCGS as well as variants of the algorithm, see also [477]. A class of generalized conjugate gradient squared methods was introduced in [358].

9.2 BiCGStab and extensions

In Lanczos-type methods the residual vectors r_k must be orthogonal to the Krylov subspace $\mathcal{K}_k(A^T, \tilde{r}_0)$. In BiCGS the basis for this subspace is constructed using the residual polynomial for r_k but, in fact, we can choose any polynomial we wish for constructing the auxiliary basis as long as the vectors spanning $\mathcal{K}_k(A^T, \tilde{r}_0)$ are linearly independent. Hence, we can construct the residual vectors of the new method as

$$r_k = q_k(A)\phi_k(A)r_0,$$

as long as $q_k(0)\phi_k(0) = 1$ to be able to obtain x_k from r_k . The choice which is made in BiCGStab (see [940, 941]) is the Newton form of a polynomial

$$q_k(\xi) = (1 - \omega_k \xi)(1 - \omega_{k-1} \xi) \cdots (1 - \omega_1 \xi) = (1 - \omega_k \xi) q_{k-1}(\xi),$$

where the inverses of the roots ω_i can be computed during the iterations. In BiCGStab, ω_k is chosen to (locally) minimize the norm of r_k . To derive the recurrence relations we need, as in BiCGS, we use as a starting point the BiCG polynomials,

$$\phi_k(\xi) = \phi_{k-1}(\xi) - \gamma_{k-1} \xi \theta_{k-1}(\xi), \quad \theta_k(\xi) = \phi_k(\xi) + \mu_k \theta_{k-1}(\xi).$$

Then, since $q_k(A) = (1 - \omega_k A)q_{k-1}(A)$, we have

$$\begin{aligned} q_k(A)\phi_k(A) &= (1 - \omega_k A)q_{k-1}(A)(\phi_{k-1}(A) - \gamma_{k-1} A\theta_{k-1}(A)), \\ &= q_{k-1}(A)(\phi_{k-1}(A) - \gamma_{k-1} A\theta_{k-1}(A)) \\ &\quad - \omega_k A q_{k-1}(A)(\phi_{k-1}(A) - \gamma_{k-1} A\theta_{k-1}(A)). \end{aligned}$$

Hence, a recurrence relation for $q_k(A)\theta_k(A)$ must be found. We have

$$\begin{aligned} q_k(A)\theta_k(A) &= q_k(A)(\phi_k(A) + \mu_k \theta_{k-1}(A)), \\ &= q_k(A)\phi_k(A) + \mu_k(1 - \omega_k A)q_{k-1}(A)\theta_{k-1}(A), \\ &= q_k(A)\phi_k(A) + \mu_k q_{k-1}(A)\theta_{k-1}(A) - \omega_k \mu_k A q_{k-1}(A)\theta_{k-1}(A). \end{aligned}$$

We set $r_k = q_k(A)\phi_k(A)r_0$ and $p_k = q_k(A)\theta_k(A)r_0$ to obtain

$$p_k = r_k + \mu_k(p_{k-1} - \omega_k A p_{k-1}),$$

$$r_k = r_{k-1} - \gamma_{k-1} A p_{k-1} - \omega_k A(r_{k-1} - \gamma_{k-1} A p_{k-1}).$$

With $s_k = r_{k-1} - \gamma_{k-1} A p_{k-1}$, this is

$$r_k = s_k - \omega_k A s_k. \quad (9.6)$$

Finally, we have to find expressions for the BiCG coefficients γ_{k-1} and μ_k , see (8.14), resp. (8.15). The two quantities that were involved in the BiCG coefficients are

$$(\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0), \quad (\theta_k(A^T)\tilde{r}_0, A\theta_k(A)r_0).$$

As it was remarked in [940], $\phi_k(A)r_0$ is mathematically orthogonal to all vectors $t_{k-1}(A^T)\tilde{r}_0$ where t_{k-1} is any polynomial of degree less than or equal to $k-1$. Note that this is based on global orthogonality properties that may not be valid in finite precision arithmetic. Therefore, instead of $\phi_k(A^T)\tilde{r}_0$, it is enough to consider only the term of degree k in $(\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0)$. Using the recurrence relation for ϕ_k , it is equal to

$$(-1)^k \gamma_{k-1} \cdots \gamma_0 (A^T)^k,$$

and

$$\begin{aligned} (\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0) &= (-1)^k \gamma_{k-1} \cdots \gamma_0 ((A^T)^k \tilde{r}_0, \phi_k(A)r_0), \\ &= (-1)^k \gamma_{k-1} \cdots \gamma_0 (\tilde{r}_0, A^k \phi_k(A)r_0). \end{aligned} \quad (9.7)$$

Now, let us consider

$$(q_k(A^T)\tilde{r}_0, \phi_k(A)r_0) = (-1)^k \omega_k \cdots \omega_1 (\tilde{r}_0, A^k \phi_k(A)r_0).$$

Hence,

$$(\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0) = \frac{\gamma_{k-1} \cdots \gamma_0}{\omega_k \cdots \omega_1} (q_k(A^T)\tilde{r}_0, \phi_k(A)r_0).$$

The coefficient μ_k of (8.15) is

$$\mu_k = \frac{(\phi_k(A^T)\tilde{r}_0, \phi_k(A)r_0)}{(\phi_{k-1}(A^T)\tilde{r}_0, \phi_{k-1}(A)r_0)} = \frac{\gamma_{k-1}}{\omega_k} \frac{(\tilde{r}_0, q_k(A)\phi_k(A)r_0)}{(\tilde{r}_0, q_{k-1}(A)\phi_{k-1}(A)r_0)}.$$

To obtain μ_k we just have to compute $(\tilde{r}_0, q_k(A)\phi_k(A)r_0) = (\tilde{r}_0, r_k)$ at each iteration and

$$\mu_k = \frac{\gamma_{k-1}}{\omega_k} \frac{(\tilde{r}_0, r_k)}{(\tilde{r}_0, r_{k-1})}.$$

For the other coefficient γ_{k-1} in (8.14), we remark, using (9.7), that the leading coefficient of θ_k is the same as for ϕ_k . Then, all the products of coefficients cancel and we obtain

$$\gamma_{k-1} = \frac{(\tilde{r}_0, A^{k-1}\phi_{k-1}(A)r_0)}{(\tilde{r}_0, A^k\theta_{k-1}(A)r_0)} = \frac{(\tilde{r}_0, r_{k-1})}{(\tilde{r}_0, Ap_{k-1})}.$$

Note that the notation here is slightly different from the original paper [940]. The roots of the polynomials q_k are chosen to locally minimize the residual norm. From (9.6) we see that minimizing the norm of the residual as a function of ω yields

$$\omega_k = \frac{(s_k, As_k)}{(As_k, As_k)}. \quad (9.8)$$

A code implementing BiCGStab is following. We chose $\tilde{r}_0 = r_0$, but this is not mandatory. As in BiCGS there are two matrix–vector products per iteration.

```
function [x,ni,resn] = BiCGStab(A,b,x0,epsi,nitmax,xec);
%
nb = norm(b);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resn(1) = norm(r);
p = r;
r0 = r;
Ap = A * p;
rkrt = r0' * r;
ni = 0;
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    gamma = rkrt / (r0' * Ap);
    s = r - gamma * Ap;
    As = A * s; % matrix vector product
    om = (As' * s) / (As' * As);
    x = x + gamma * p + om * s;
    r = s - om * As;
    nresidu = norm(r);
    resn(ni+1) = nresidu;
    if nresidu < (epsi * nb) || ni >= nitmax
        break % get out of the k loop
    end % if nresidu
    rkr = r0' * r;
    mu = (rkr / rkrt) * (gamma / om);
    rkrt = rkr;
    p = r + mu * (p - om * Ap);
```

```

Ap = A * p; % matrix vector product
end % for k
resn = resn(1:ni+1);

```

The generalization to complex data is done in [480]. BiCGStab is as simple to code as BiCGS and, most of the time, the residual norms are less oscillatory. As BiCG and BiCGS, BiCGStab may also suffer from (near-)breakdowns. As for BiCGS we can obtain a basis for the Krylov subspaces by mixing the residual vectors and the vectors s_k by eliminating the vectors p_k . It yields an Arnoldi-like relation and BiCGStab is an implementation of the corresponding generalized Q-OR method.

Theorem 9.2 *Assume that there is no breakdown in BiCGStab, that $2k + 1$ is smaller than the grade of r_0 with respect to A and let*

$$\begin{aligned} V_{n,m} &= (r_0 \ s_1 \cdots r_{k-1} \ s_k) \text{ if } m = 2k, \\ V_{n,m} &= (r_0 \ s_1 \cdots s_k \ r_k) \text{ if } m = 2k + 1. \end{aligned}$$

Then, every second residual vector of the Q-OR method using the columns of $V_{n,m}$ as basis vectors is mathematically equal to a BiCGStab residual vector.

Proof The BiCGStab vectors satisfy the following relations,

$$\begin{aligned} p_k &= r_k + \mu_k(p_{k-1} - \omega_k Ap_{k-1}), \\ s_k &= r_{k-1} - \gamma_{k-1}Ap_{k-1}, \\ r_k &= s_k - \omega_k As_k. \end{aligned}$$

It yields

$$Ap_k = \frac{1}{\gamma_k}(r_k - s_{k+1}), \quad As_k = \frac{1}{\omega_k}(s_k - r_k),$$

and

$$p_k = r_k + \mu_k(p_{k-1} - \frac{\omega_k}{\gamma_{k-1}}(r_{k-1} - s_k)).$$

We have to eliminate the vectors p_j in the last relation and to recur down to $p_0 = r_0$. Let

$$z_j = \frac{\omega_j}{\gamma_{j-1}}(r_{j-1} - s_j), \quad \tau_1^{(k)} = 1, \quad \tau_2^{(k)} = \mu_k, \quad \tau_j^{(k)} = \tau_{j-1}^{(k)}\mu_{k-j+2}, \quad j = 3, \dots, k+1.$$

Then,

$$\begin{aligned} p_k &= r_k + \sum_{j=0}^{k-1} \tau_{k-j+1}^{(k)} (r_j - z_{j+1}), \\ &= r_k + \sum_{j=0}^{k-1} \tau_{k-j+1}^{(k)} \left[\left(1 - \frac{\omega_{j+1}}{\gamma_j} \right) r_j + \frac{\omega_{j+1}}{\gamma_j} s_{j+1} \right]. \end{aligned}$$

Initially, we have

$$Ar_0 = \frac{1}{\gamma_0} (r_0 - s_1), \quad As_1 = \frac{1}{\omega_1} (s_1 - r_1).$$

We have to write these relations in matrix form. Let

$$\underline{\underline{H}}_m = \begin{pmatrix} \frac{1}{\gamma_0} & & & & & \\ -\frac{1}{\gamma_0} & \frac{1}{\omega_1} & & & & \\ & -\frac{1}{\omega_1} & \frac{1}{\gamma_1} & & & \\ & & -\frac{1}{\gamma_1} & \frac{1}{\omega_2} & & \\ & & & -\frac{1}{\omega_2} & \frac{1}{\gamma_2} & \ddots \\ & & & & -\frac{1}{\gamma_2} & \ddots \\ & & & & & \ddots \end{pmatrix}$$

and

$$R_m = \begin{pmatrix} 1 & 0 & \tau_2^{(1)} \left(1 - \frac{\omega_1}{\gamma_0} \right) & 0 & \tau_3^{(2)} \left(1 - \frac{\omega_1}{\gamma_0} \right) & 0 & \dots \\ & 1 & \tau_2^{(1)} \frac{\omega_1}{\gamma_0} & 0 & \tau_3^{(2)} \frac{\omega_1}{\gamma_0} & 0 & \dots \\ & & 1 & 0 & \tau_2^{(2)} \left(1 - \frac{\omega_2}{\gamma_1} \right) & 0 & \dots \\ & & & 1 & \tau_2^{(2)} \frac{\omega_2}{\gamma_1} & 0 & \dots \\ & & & & 1 & 0 & \dots \\ & & & & & 1 & \ddots \\ & & & & & & \ddots \end{pmatrix}.$$

Then, we have

$$AV_{n,m} R_m = V_{n,m+1} \underline{\underline{H}}_m,$$

that is, the same kind of relation we had for BiCGS but with different matrices $\underline{\underline{H}}_m$ and R_m . Here the matrix R_m has a every special nonzero structure since the even columns are equal to the corresponding columns of the identity matrix. The inverse of R_m has the same nonzero structure and the even columns of $\underline{\underline{H}}_m R_m^{-1}$ are equal to the corresponding columns of $\underline{\underline{H}}_m$. Again, we can write the Arnoldi-like relation,

$$AV_{n,m} = V_{n,m+1} \tilde{H}_m R_m^{-1}.$$

The proof for the residual vectors of the Q-OR method using $V_{n,m}$ is even easier than for BiCGS. We observe that $H_m = \tilde{H}_m R_m^{-1}$ is an LU factorization of H_m . The matrix \tilde{H}_m is lower bidiagonal and the solution of $\tilde{H}_m z = e_1$ is given by

$$z_{2k-1} = \gamma_{k-1}, \quad z_{2k} = \omega_k, \quad k = 1, \dots$$

It shows that the last entry of z times $h_{m+1,m}$ is equal to minus one. Hence, all the Q-OR residual vectors are equal to the columns of the matrices $V_{n,m}$. The vector of coefficients $y^{(m)}$ of $x_m - x_0$ in this basis is $R_m z$ and $x_m = x_0 + V_{n,m} R_m z$. We note that the vectors s_j are also residual vectors of the Q-OR method. \square

Theorem 9.2 shows that BiCGStab is a Q-OR method and we can apply the results of Chapter 3.

It was observed experimentally that BiCGStab can sometimes almost stagnate. This happens, in particular, for linear systems arising from the discretization of convection–diffusion partial differential equations. It seems that this is due to the fact that, for this type of equations, the matrix may have almost pure imaginary eigenvalues depending on the ratio of diffusion and convection coefficients. Also ω_k may sometimes be close to zero, and this may cause stagnation or even a breakdown. Experiments showed that this is likely to happen if A is real and has non real eigenvalues with an imaginary part that is large relative to the real part (apparently, eigenvalues do govern convergence for these problems).

Motivated by the fact that to have good convergence the residual polynomials must be small at the eigenvalues of A and that the zeros of the polynomials q_k in BiCGStab are real in case of real data (see (9.8)), it was proposed in [480] to modify the method to let the polynomial having complex conjugate roots. This was also supposed to give a better damping effect for the residuals. The resulting method is known as BiCGStab2. To be able to still use real arithmetic the polynomial q_k is not chosen in the same way in the odd and even iterates,

$$\begin{aligned} q_{2j+1}(\xi) &= (1 - \omega_j \xi) q_{2j}(\xi), \\ q_{2j+2}(\xi) &= (1 - \nu_j) q_{2j}(\xi) + (\nu_j - \eta_j \xi) q_{2j+1}(\xi). \end{aligned}$$

When the data is real, the coefficients are real and q_{2j+2} has real or complex conjugate roots. The parameters are chosen to minimize the residual norms in a one- or two-dimensional subspace. Recurrence relations for the polynomials we need were obtained in [480] even though the derivation is more complicated than for BiCGStab and one has to distinguish the cases of odd and even iterations. The odd and even steps can be combined into one iteration but, in this case, we have to be careful when we compare the methods on the basis of the number of iterations. Of course, the code is a little more complicated than for BiCGStab.

Another extension of BiCGStab was proposed in [841]; see also [849]. This method, called BiCGStab(ℓ), is doing ℓ BiCG-like steps and then a residual norm minimization in a subspace of dimension ℓ which is called the MR part. The residual vector and the approximate solution are only available at the end of these two steps. This is called outer iteration steps in [841] but here we just call them iterations.

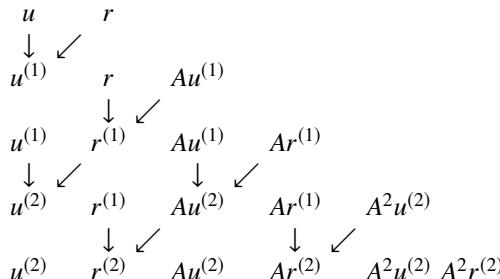
Even though the ideas on which the algorithm is based are easy to understand, its derivation in [841] is quite involved, mainly because of the notation. BiCGStab(ℓ) is a kind of inner–outer iterative method but we are only interested in the residual vectors and the approximate solution at the end of the two steps. To be consistent with what we did for the other methods we still denote them as r_k and x_k . Let \hat{r}_k be the residual at the end of the BiCG step. Then,

$$r_k = m_k(A)\hat{r}_k,$$

where m_k is the product of locally minimizing polynomials $p_{m,\ell}$ (with $k = m\ell$) of degree ℓ with a value 1 at the origin,

$$m_k = p_{m,l}m_{k-\ell}.$$

As we said above, these polynomials are chosen to minimize the norm of r_k . In the BiCG part we construct a basis of the Krylov subspace of dimension ℓ in which we want to minimize the residual norm. The way this is done is illustrated in the scheme below (borrowed from [841]) which corresponds to $\ell = 2$. We start from two vectors u and r which are the results of the MR step of the previous iteration. At the initialization u is equal to zero and $r = r_0$. The down and east-south arrows correspond to linear combinations of the vectors using the BiCG coefficients. For instance, $Au^{(2)}$ is obtained combining $Au^{(1)}$ and $Ar^{(1)}$ without the need of a multiplication by A . Matrix–vector products are only done on the upper diagonal of the scheme. Here, for $\ell = 2$ we have four matrix–vector products.



In this example, $r^{(2)}$ is what we would get after two iterations of BiCG starting from r . We also have at hand $Ar^{(2)}$ and $A^2r^{(2)}$. More generally, starting from known u and r , at the end of the BiCG part we have a basis of the Krylov subspace $\{r, Ar, \dots, A^\ell r\}$. Since we want to minimize the residual norm we use the basis given by the columns

of the matrix

$$\hat{R} = (Ar \ A^2r \ \cdots \ A^\ell r).$$

Then, we minimize $\|r - \hat{R}y\|$. This can be done in different ways. Since ℓ is generally small, the easiest way is to use the normal equations,

$$y = (\hat{R}^T \hat{R})^{-1} \hat{R}^T r,$$

and, then, the new residual vector at the end of the MR step is $r - \hat{R}(\hat{R}^T \hat{R})^{-1} \hat{R}^T r$.

However, as stated in [356, 357], to obtain accurate coefficients it is recommended, following [846], to take a convex combination of the minimum residual polynomial with a polynomial $t_{m,l}(A)r$ such that $t_{m,l}(A)r$ is orthogonal to $\{r, Ar, \dots, A^{\ell-1}r\}$. According to [357] this can be done as follows. Let

$$R = (r \ Ar \ \cdots \ A^\ell r), \quad Z = R^T R,$$

and

$$\tilde{r} = R\tilde{y}, \quad \tilde{y} = (1 \ -[(Z_{2:\ell,2:\ell})^{-1} Z_{2:\ell,\ell+1}]^T \ 0)^T,$$

$$\check{r} = R\check{y}, \quad \check{y} = (0 \ -[(Z_{2:\ell,2:\ell})^{-1} Z_{2:\ell,\ell+1}]^T \ 1)^T.$$

It yields

$$(\tilde{r}, \check{r}) = \check{y}^T Z \tilde{y}, \quad \|\tilde{r}\| = \sqrt{\check{y}^T Z \tilde{y}}, \quad \|\check{r}\| = \sqrt{\check{y}^T Z \check{y}}.$$

The new residual vector using the combined polynomial is

$$\tilde{r} - \hat{\gamma} \frac{\|\tilde{r}\|}{\|\check{r}\|} \check{r},$$

with, for a given κ ,

$$\hat{\gamma} = \frac{\rho}{|\rho|} \max(|\rho|, \kappa), \quad \rho = \frac{(\tilde{r}, \check{r})}{\|\tilde{r}\| \|\check{r}\|}.$$

This is what is implemented in the following code which is similar to those in the IFISS package [310, 312] or on G. Sleijpen's homepage. It also implements a safeguard for the computation of ω_k ; see [846, 847]. The two inputs `ell` and `kappa` represent, respectively, ℓ and κ .

```
function [x,ni,resn,resnt] = BiCGStab1(A,b,x0,epsi,nitmax,
                                         ell,kappa);
%
if isempty(ell)
    l = 2; % BiCGStab(2)
else
```

```
l = ell;
end
if isempty(kappa)
    kappa = 0.7;
end
nb = norm(b);
nA = size(A,1);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resnt = zeros(1,nitmax+1);
resn(1) = norm(r);
resnt(1) = resn(1);
rt = r;
u = zeros(nA,l+1);
gamma = 1;
omega = 1;
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    gamma = -omega * gamma;
    y = r;
    for j = 1:l
        rho = rt' * y;
        beta = rho / gamma;
        u = r - beta * u;
        y = A * u(:,j); % matrix vector product
        u(:,j+1) = y;
        gamma = rt' * y;
        alpha = rho / gamma;
        x = x + alpha * u(:,1);
        r = r - alpha * u(:,2:j+1);
        y = A * r(:,j); % matrix vector product
        r(:,j+1) = y;
    end % for j
    G = r' * r;
    Gamma0 = [1; -G(2:l,2:l) \ G(2:l,1); 0];
    Gamma1 = [0; -G(2:l,2:l) \ G(2:l,l+1); 1];
    NGamma0 = Gamma0' * G * Gamma0;
    NGamma1 = Gamma1' * G * Gamma1;
    omega = Gamma0' * G * Gamma1;
    cosine = abs(omega) / sqrt(NGamma0 * NGamma1);
    omega = omega / NGamma1;
    if cosine < kappa
```

```

omega = (kappa / cosine) * omega;
end
Gamma = Gamma0 - omega * Gamma1;
x = x - r * [Gamma(2:l+1); 0];
r = r * Gamma;
u = u * Gamma;
nresidu = norm(r(:,end));
resn(ni+1) = nresidu;
resnt(ni+1) = norm(b - A * x);
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k
resn = resn(1:ni+1);
resnt = resnt(1:ni+1);

```

We observe that we are (implicitly) using the monomial basis for the subspace of dimension ℓ . Therefore, it is likely that we will not be able to use large values of ℓ .

9.3 Other product-type methods

The relatively good performances and the ease of coding of BiCGS, BiCGStab and BiCGStab(ℓ) triggered an interest for this type of methods in the 1990s and a plethora of methods appeared.

In [386] (see also [384, 385]) a variant of QMR was introduced which does not need A^T . It is called the quasi-minimum residual squared algorithm (QMRS). It is obtained by squaring the QMR polynomials. The authors started by recalling the relations between BiCG and the QMR algorithm without look-ahead. We have seen these relations in Chapter 8, Section 8.3. We used them to show that we can obtain the BiCG vectors from QMR but they can be used in the opposite way, yielding the QMR vectors from BiCG. Of course, this still needs A^T but now we can square the BiCG polynomial as in BiCGS. Then, relations can be found to compute the QMRS polynomial from the BiCGS polynomial. In fact, weights have to be introduced in the QMR algorithm to avoid computing some extra dot products. From these relations the QMRS algorithm is obtained from BiCGS. However, we have three matrix–vector products per iteration instead of two in BiCGS. QMRS gives a much smoother residual norm convergence curve but if the BiCGS oscillations would spoil the computed quantities, the same would happen for the QMRS vectors.

The TFQMR method [369] uses BiCGS basis vectors to construct iterates that satisfy a quasi-minimum residual property. However, TFQMR is not mathematically equivalent to the original QMR algorithm in [379]. Let us briefly describe the principles on which the method is based. Let

$$y_m = u_k \text{ if } m = 2k - 1, \quad y_m = q_k \text{ if } m = 2k$$

where u_k and q_k are the BiCGS vectors, and

$$w_m = [\phi_k(A)]^2 r_0 \text{ if } m = 2k - 1, \quad w_m = \phi_k(A)\phi_{k-1}(A)r_0 \text{ if } m = 2k.$$

Let

$$Y_m = (y_1 \ y_2 \ \cdots \ y_m), \quad W_{m+1} = (w_1 \ w_2 \ \cdots \ w_{m+1}).$$

One can check that the two bases are related by

$$AY_m = W_{m+1}\underline{B}_m,$$

where \underline{B}_m is a lower bidiagonal $(m + 1) \times m$ matrix whose nonzero entries depend on the BiCGS coefficients μ_j . The vectors y_1, \dots, y_m span the Krylov subspace $\mathcal{K}_m(A, r_0)$. Writing iterates as $x_m = x_0 + Y_m z$, we can choose z to minimize the norm of the quasi-residual. It turns out that, since \underline{B}_m has a simple nonzero structure, the solution of the least squares problem can be updated cheaply during the iterations; see [369]. The resulting algorithm only needs two matrix–vector products per iteration. Note that we may have to normalize the columns of W_{m+1} in order for the residual norm to be not too far from the quasi-residual norm.

In [205] the authors derived a quasi-minimum residual extension of BiCGStab. They called their method QMRCGSTAB but it seems more appropriate to name it QMRBiCGStab. Constructing such a method may seem a strange idea since BiCGStab is already doing some local minimization of the residual norm. However, the QMR idea amounts to do a global minimization of the quasi-residual norm. The method is derived from BiCGStab in the same way as TFQMR was derived from BiCGS, introducing new vectors y_m and w_m and minimizing the quasi-residual norm. There are two matrix–vector products per iteration. This algorithm yields smoother residual norm curves than BiCGStab but it is slightly more expensive. Some variants are also discussed in [205].

In [203, 204] a “squared Lanczos” procedure was developed, which generates the tridiagonal matrix of the standard nonsymmetric Lanczos algorithm without accessing A^T . The authors formulated the three-term recurrences of the nonsymmetric Lanczos process in terms of polynomials and used the same technique as in BiCGS. The resulting algorithm TFiQMR for solving linear systems requires three matrix–vector products with A per iteration. The third matrix–vector product is needed to construct the Lanczos vectors from the coefficients of the tridiagonal matrix. Once we have them we can proceed as in the standard QMR algorithm (without look-ahead) to obtain the TFiQMR algorithm. Mathematically, it is equivalent to the three-term version of QMR. The other algorithm in [204], TFiBiCG, produces the BiCG iterates from those of BiCGS. It is a simple modification of BiCGS.

In [223] the same idea of squaring the nonsymmetric Lanczos algorithm was considered; see also [227]. Then, methods for solving linear systems were derived. They are called SBiLQR (a Q-OR method) and BiLMINRES (a Q-MR method).

In [358] the authors considered transpose-free Lanczos methods where the polynomial $q_k(A)$ to be multiplied with the BiCG polynomial is computed by

general two-term recurrences. They called them generalized conjugate gradient squared (GCGS) but it would be better to call it GBiCGS. BiCGS and BiCGStab are obtained as particular cases of this more general framework. Analyzing problems that can occur with BiCGS and BiCGStab when they are used as inner linear solvers in Newton nonlinear iterations, the authors proposed two other methods. The first one which is called CGS2 (but, to be consistent we call it BiCGS2) uses for q_k what is referred as “a nearby BiCG polynomial” starting from a vector $\tilde{s}_0 \neq \tilde{r}_0$ which may be chosen as a random vector. It turns out that this can be done with only two additional vector updates and two more dot products than in BiCGS; see [358]. As in BiCGS there are only two matrix–vector products per iteration. The other method which is called shifted CGS uses

$$q_k(\xi) = (1 - \lambda\xi)\phi_{k-1}(\xi),$$

where λ is a parameter. It is suggested to choose λ as the real part of an estimate of the largest eigenvalue of A . But, of course, this may not be available and computing it before starting the method could be costly. This choice was made heuristically to reduce the influence of the largest eigenvalue on the convergence of the residual norms.

Variants of BiCGS were derived in [477, 482]. They are called BIORESS and BIODIRS and are obtained by squaring the Orthores and Orthodir polynomials.

General product-type methods were considered in [486] where the polynomials q_k satisfy a three-term recurrence relation. From this framework many different methods can be derived including BiO \times MR2 that was first introduced in 1994. Except for the first iteration the coefficients of the three-term recurrence are determined by solving a two-dimensional minimization problem. Q-MR methods were also proposed in [486]. They are related to smoothing techniques.

A somehow similar framework was introduced in [1029] (see also [1030]) under the name GPBiCG, “GP” meaning generalized product-type. The polynomials q_k were taken to satisfy a three-term recurrence relation and relations for the needed products of polynomials were derived. Then, the coefficients were chosen to minimize the residual norm. It is known that GPBiCG and BiO \times MR2 are mathematically equivalent even though their formulations are different. A derivation using two-term coupled recurrences was also given.

This kind of algorithm was also studied in [773]. The authors proposed an algorithm named BiCG \times MR2_2 \times 2 which is similar to GPBiCG, but formulated differently.

Lanczos product-type methods sparked a great interest in Japan, see, for instance [389–392, 821]. A hybrid of BiCGStab and GPBiCG methods called GPBiCG(m, ℓ) was proposed in [389]. In this method parameters of the BiCGStab method are used for m consecutive iterations and afterwards those of the GPBiCG method are used for ℓ iterations; see also [5]. A variant of GPBiCG called BiCGSafe was proposed in [390]; see also [392, 821].

Transpose-free Lanczos methods can also be obtained using the formal orthogonal polynomial framework that we have described in Chapter 8. The linear functional

Φ on the set of polynomials with complex coefficients is defined as $\Phi(\xi^i) = \varphi_i$ for $i = 0, 1, \dots$ with

$$\varphi_i = ([A^*]^i \tilde{r}_0, r_0) = (\tilde{r}_0, A^i r_0), \forall \text{ integers } i.$$

The Lanczos orthogonality conditions are $\Phi(\xi^i p_k) = 0$, $i = 0, 1, \dots, k - 1$ where p_k is the residual polynomial. But this can be formulated as $\Phi(q_i p_k) = 0$ where q_i is an arbitrary polynomial of exact degree i . The polynomial p_k exists and is unique if and only if the Hankel determinant $h_k^{(1)}$ is nonzero (see Chapter 8). The polynomials p_k can also be computed using their adjacent family of polynomials $p_k^{(1)}$. The orthogonality conditions give $\Phi^{(1)}(q_i^{(1)} p_k^{(1)}) = 0$, $i = 0, \dots, k - 1$. The recurrences for the polynomials without specifying q_i and $q_i^{(1)}$ were given in [135]. We have seen in Chapter 8 that with three-term recurrences we obtain Lanczos/Orthores and with coupled two-term recurrences we obtain Lanczos/Orthomin. Choosing q_i as p_i or $p_i^{(1)}$ yields BiCGS. Choosing $q_i(\xi) = (1 - \omega_k \xi) q_{i-1}(\xi)$ yields BiCGStab. Other choices for q_i were proposed in [134]. In [135] the authors considered more generally polynomials q_i and $q_i^{(1)}$ satisfying three-term recurrences; see also [127]. There are many ways to compute the products of polynomials that are needed and the coefficients can be computed without using A^T (or A^*). Using for q_i the BiCGStab damping polynomial gives the transpose-free Lanczos/Orthores BiCGStab algorithm named TFres-BiCGStab and the transpose-free Lanczos/Orthomin BiCGStab algorithm named TFminBiCGStab. Other algorithms can also be derived from Lanczos/Orthores. The drawback of these algorithms is that they require three matrix–vector products per iteration.

9.4 Look-ahead for transpose-free methods

Since all transpose-free methods are obtained from the nonsymmetric Lanczos algorithm or its variants like BiCG, they can suffer from breakdowns. Therefore, it was tempting to try to derive look-ahead versions of these transpose-free methods.

Handling breakdowns and near-breakdowns for BiCGS was considered in [133, 147] using formal orthogonal polynomials. In the second paper the authors used the relations for the polynomials in BSMRZ (see Chapter 8) and squared them obtaining the BSMRZS algorithm. However, this algorithm does not reduce to the implementation of BiCGS when look-ahead is not used since it uses a different polynomial formulation. When there is no breakdown, BSMRZS requires three matrix–vector products per iteration compared to two for BiCGS. A Fortran code is still available in Netlib (www.netlib.org).

In [206, 207] the composite step technique introduced in [68] for BiCG was extended to BiCGS and BiCGStab as well as BiCGStab2. However, as we have seen previously, this does only cure the so-called pivot breakdowns. Unfortunately, there

are some typographical errors and misses in the pseudocode which is given in Table 2 on page 1501. For instance, on line 13 of the code the variable e_{n+1} has not been defined before.

Algorithms to cure (near-) breakdowns in general Lanczos product-type methods were proposed in [134] with a particular emphasis on BiCGStab. One of them is an amalgamation of the MRZ look-ahead method (see Chapter 8) with a BiCGStab stabilization to obtain a reliable method. However, it was stated that the numerical results were quite sensitive to the choices of the thresholds which are used to decide if there is or not a near-breakdown.

The theory of formal orthogonal polynomials was used in [178] to construct a breakdown-free BiCGStab algorithm and a breakdown-free BiCGStab2 algorithm. However, only exact breakdowns were cured accurately.

The look-around method was introduced in [446]. It was combined with BiCGStab-like “stabilization” in [450]. The algorithm is derived using the framework of Padé approximation.

In [487] look-ahead procedures were considered for transpose-free methods where the polynomials q_k satisfy a general three-term recurrence relation. In particular, this applies to BiCGStab. The same technique as for QMR was used by introducing blocks of vectors defined by regular and inner vectors and relaxing the orthogonality conditions. Criteria for deciding when a near-breakdown occurs were discussed but this is quite tedious. In the authors’ own words “larger problems indicate that further work should be directed to finding an improved look-ahead criterion that more reliably avoids critical perturbations of the Lanczos coefficients by roundoff errors”.

9.5 Parallel computing

Several modifications intended to improve parallelism were proposed for BiCGStab. The method in [1001], denoted as *improved BiCGStab*, eliminates the vector denoted p in our implementation of BiCGStab and replaces the dot products by differences of quantities computed in previous iterations such that these dot products can be computed in parallel. The authors claimed that there is only one single global synchronization point per iteration; see also [1003]. The reader should be warned that there are some mistakes in the derivation of the algorithm and in the given pseudocode. The second equality of relation (7) in [1001] is wrong because $\alpha_n p_{n-1}$ is not equal to z_{n-1} . The correct relation is

$$z_n = \alpha_n r_{n-1} + \left(\frac{\alpha_n}{\alpha_{n-1}} \right) \beta_n z_{n-1} - \alpha_n \beta_n v_{n-1}.$$

Moreover ϕ_0 has to be initialized as $r_0^T r_0$ and not to zero and δ_n has to be set to ρ_n at the first iteration and otherwise to δ_n / τ_{n-1} .

A reordered BiCGStab algorithm was considered in [607], mainly for the preconditioned case, the aim being to be able to overlap the communications for the dot products and the computations with the preconditioner. There are less vector updates and dot products than in the variant in [1001]. However, in finite precision arithmetic, the maximum attainable accuracy can be worse than what is obtained with the standard version of BiCGStab.

Hierarchical Krylov methods and nested Krylov methods that reduce the number of global dot products and allow vector dot products across smaller subsets of the entire computer were proposed in [667].

Communication-avoiding s -step versions of (Bi)CGS and BiCGStab were derived in [188] using the techniques we have described in Section 8.9 of Chapter 8. As with CA-BiCG, the goal was to be able to use the matrix powers kernel and to reduce the number of global synchronization points.

A technique called *pipelining* was proposed for BiCGStab in [235]. A general framework for deriving a pipelined Krylov algorithm was presented. The authors introduced auxiliary vectors to decouple matrix–vector products and dot products. The data dependencies due to the dot products are alleviated by deriving recurrences for the vectors involved and by computing these dot products with variables of the previous iterations. The goal was to allow overlapping of the global reduction communication phases with independent matrix–vector products computations. However, the number of dot products is larger than with standard BiCGStab (6 instead of 4) as well as the number of vector updates (8 instead of 4) and the resulting algorithm tends to be unstable. It was proposed to improve this by replacing the computed residual by the true residual $b - Ax_k$ every m iterations. This is a very simple strategy that may not work for all problems. A mathematical study of the residual gap is done in [233].

A parallel version of BiCGStab(2) was presented in [469]. It involves more dot products and vector updates than in the standard version.

In [20] BiCGStab algorithms were tested for geoscience dynamic simulations. The results on problems from reservoir simulation show that, for most of the tested matrices, s -step CA-BiCGStab requires more iterations to converge than BiCGStab. Variants of CA-BiCGStab using orthonormalization of the $4s + 1$ basis vectors constructed at each step were proposed. They seem to improve the robustness of the method.

The convergence speeds of parallel variants of BiCGStab are studied in [2].

9.6 Finite precision arithmetic

As far as we know, there is no finite precision analysis of BiCGS and BiCGStab available, except for the maximum attainable accuracies, see, for instance, [233]. This does not come as a surprise since we have previously seen that it was already difficult

to analyze BiCG in finite precision arithmetic. However, since we have proved that these methods are equivalent to Q-OR methods using the bases generated by some of their vectors, one possibility would be to obtain the rounding error terms associated with each step of the algorithms and to write a perturbed Arnoldi-like relation for the computed quantities,

$$AV_{n,m}R_m + F_{n,m} = V_{n,m+1}\tilde{H}_m.$$

Then, we would be able to apply the general results in [1020] on perturbed Krylov methods. It would give polynomial expressions relating the Ritz vectors, Q-OR residual iterates and Q-MR residual iterates to the starting vector r_0 and the perturbation term. These polynomials are related only to the entries of the computed \tilde{H}_m . The Q-OR and Q-MR residual vectors could then be obtained by using theorems 4.4 and 5.6 in [1020]. But such a study is outside the scope of this book.

What has been discussed in the literature is the choice of implementations of the methods to improve the accuracy of the computed BiCG coefficients and the maximum attainable accuracy. It was shown in [847] that to obtain accurate BiCG coefficients in BiCGStab(ℓ) it would be better to use an orthogonal residual polynomial rather than a minimal residual polynomial (in the ℓ dimensional subspace constructed on the residual of the BiCG part). However, this choice may amplify the norm of the error. A compromise is to take a convex combination of the OR and MR polynomials which depend on the angle between \hat{r} and $A\hat{r}$ where \hat{r} is the residual after the BiCG part; see [846, 847] for details. However, these proposals are mainly based on heuristics and analysis of numerical experiments. Residual replacement strategies for improving the maximum attainable accuracy are considered in [848]. Further results about the implementation of BiCGStab(ℓ) are given in [849].

9.7 Numerical experiments

In this section we report the results of numerical experiments with several of the methods that have been described in this chapter.

9.7.1 BiCGS

Let us compare BiCG and BiCGS. Since both methods do two matrix–vector products per iteration, we can consider residual norms as functions of the iteration number. For [fs 183 6](#) both methods converge very slowly with oscillations but, as one can see in Figure 9.1, the height of the BiCGS oscillations are larger than those of BiCG.

For the problem [fs 680 1c](#) BiCG converges reasonably well and, as one can expect in this case, BICGS converges faster; see Figure 9.2. But note that we have large oscillations of the residual norms.

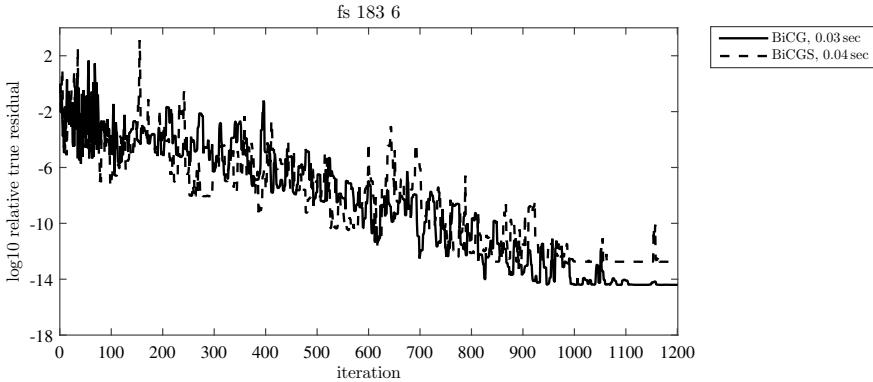


Fig. 9.1 *fs 183 6*, relative true residual norms, BiCG (plain), BiCGS (dashed)

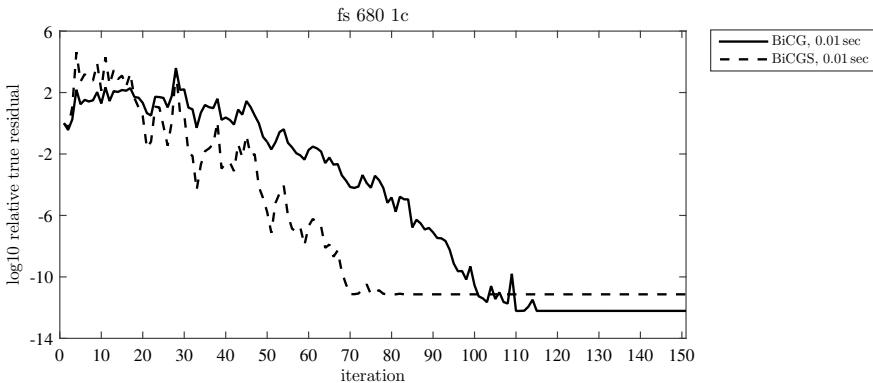


Fig. 9.2 *fs 680 1c*, relative true residual norms, BiCG (plain), BiCGS (dashed)

Things are different for `supg 001` of order 1225; BiCG converges slowly but BiCGS stagnates. This is probably due to the very large values of the residual norms at the beginning of the iterations that spoil the computation; see Figure 9.3.

9.7.2 *BiCGStab and variants*

Let us consider BiCGStab and its variants or extensions. In Figure 9.4 one can see that for *fs 183 6* the convergence of BiCGStab is smoother than that of BiCG but it does not converge faster.

Figure 9.5 shows the relative true residual norms for computations in double precision and in variable precision using 32 and 64 decimal digits. It shows that, for this example, rounding errors have a great impact on the convergence of BiCGStab.

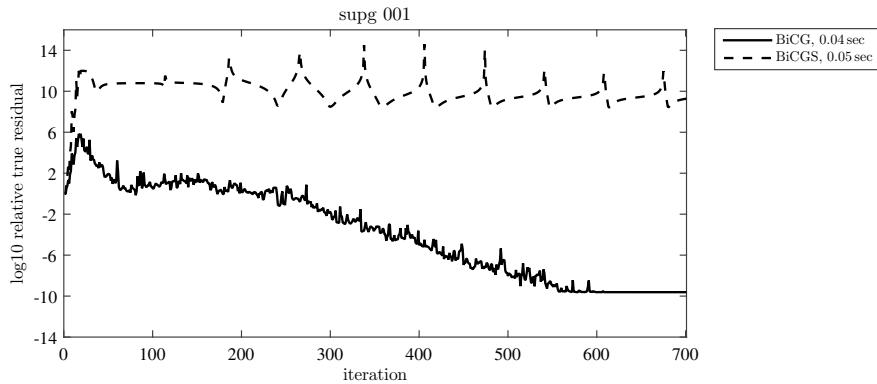


Fig. 9.3 supg 001, order 1225, relative true residual norms, BiCG (plain), BiCGS (dashed)

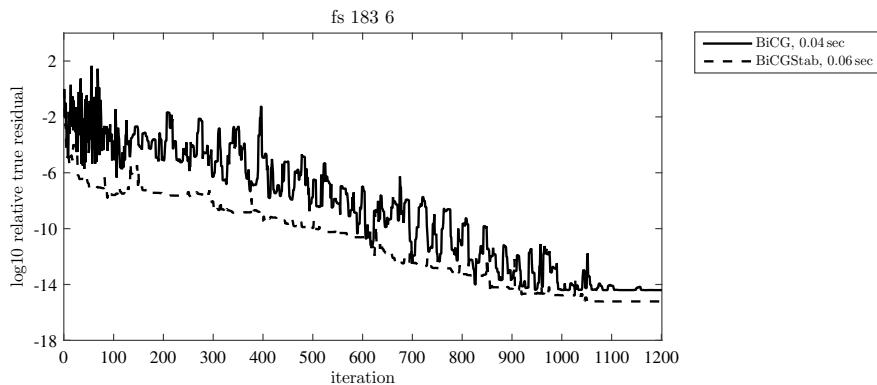


Fig. 9.4 fs 183 6, relative true residual norms, BiCG (plain), BiCGStab (dashed)

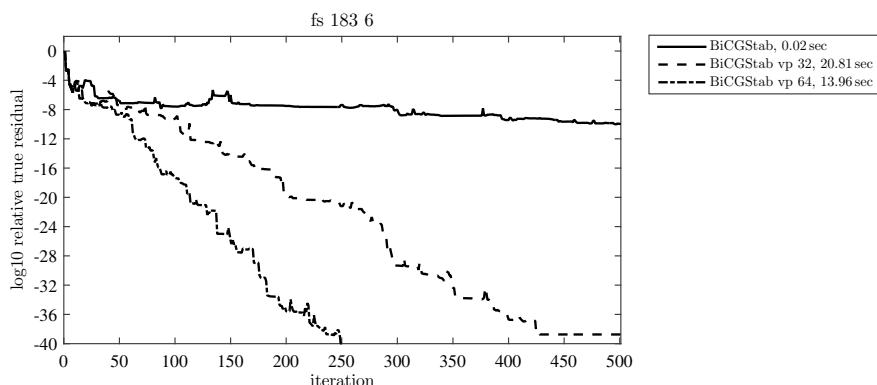


Fig. 9.5 fs 183 6, log₁₀ of the relative true residual norm, BiCGStab (plain), BiCGStab variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

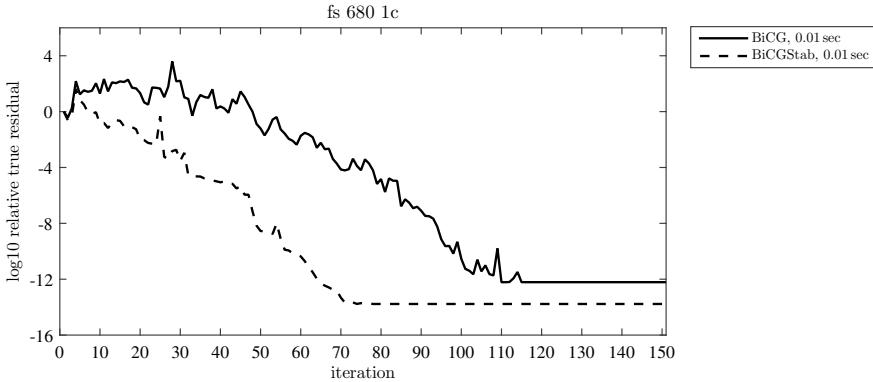


Fig. 9.6 *fs 680 1c*, relative true residual norms, BiCG (plain), BiCGStab (dashed)

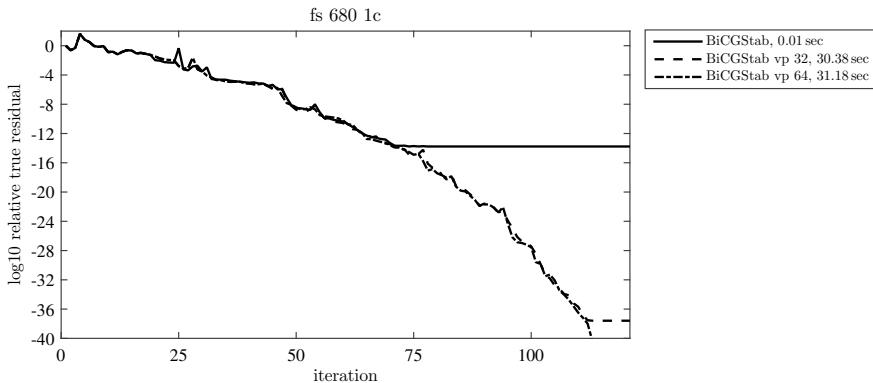


Fig. 9.7 *fs 680 1c*, \log_{10} of the relative true residual norm, BiCGStab (plain), BiCGStab variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

Even if there are less oscillations than with BiCG, BiCGStab is also prone to large delays in convergence due to rounding errors.

For the problem *fs 680 1c* BiCGStab converges faster than BiCG and the final accuracy is better; see Figure 9.6.

Figure 9.7 shows that the double-precision computation gives more or less the same residual norms as the computations in extended precision until the maximum attainable accuracy is reached. Hence, for this problem, there is not much influence of the rounding errors as it was also the case for BiCG, except of course, for the maximum attainable accuracy.

The convergence of BiCGStab is much better than that of BiCG for *supg 001* of order 1225 as we can see in Figure 9.8.

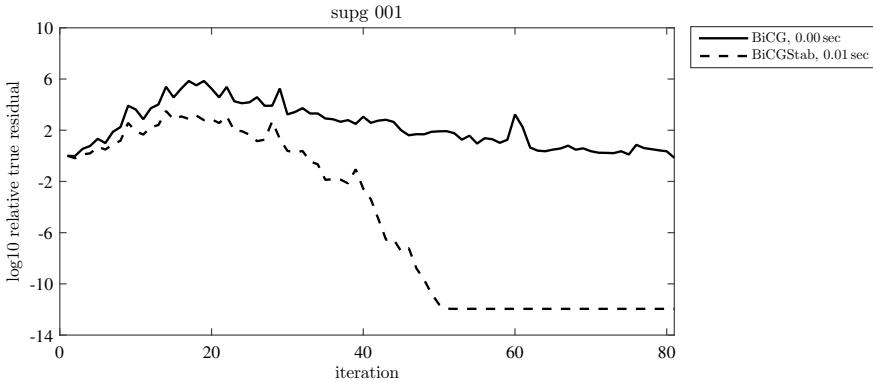


Fig. 9.8 `supg 001`, order 1225, relative true residual norms, BiCG (plain), BiCGStab (dashed)

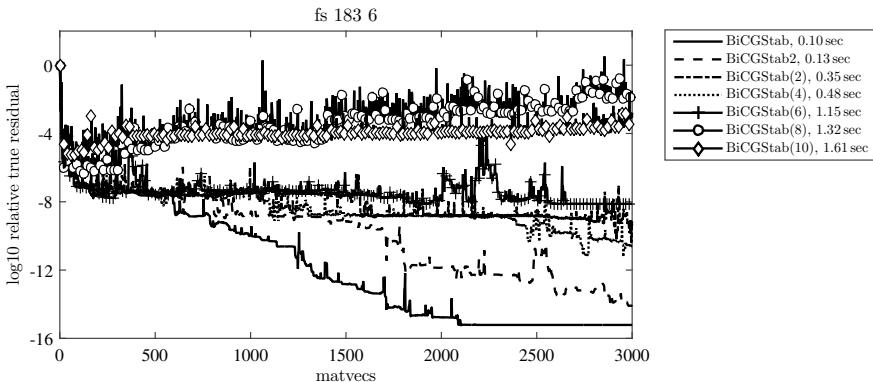


Fig. 9.9 `fs 183 6`, relative true residual norms, BiCGStab (plain), BiCGStab2 (dashed), BiCGtab(ℓ) $\ell = 2 : 2 : 10$

Now we compare BiCGStab with BiCGStab2 and the methods BiCGStab(ℓ) for $\ell = 2, 4, 6, 8, 10$. The results for the difficult problem `fs 183 6` are shown in Figure 9.9. The relative true residual norms are plotted as function of the number of matrix–vector products. Only BiCGstab and BiCGstab2 are converging reasonably well. All the other methods stagnate or diverge.

Things are different for `fs 680 1c` for which when plotted as functions of the number of matrix–vector products all the residual norms behave similarly. The main differences are for the maximal attainable accuracies. They are worse for the largest values of ℓ in BiCGStab(ℓ); see Figure 9.10.

Figure 9.11 shows that the conclusions are roughly the same for `supg 001` of order 1225. Note that all residual norms are increasing at the beginning of the iterations.

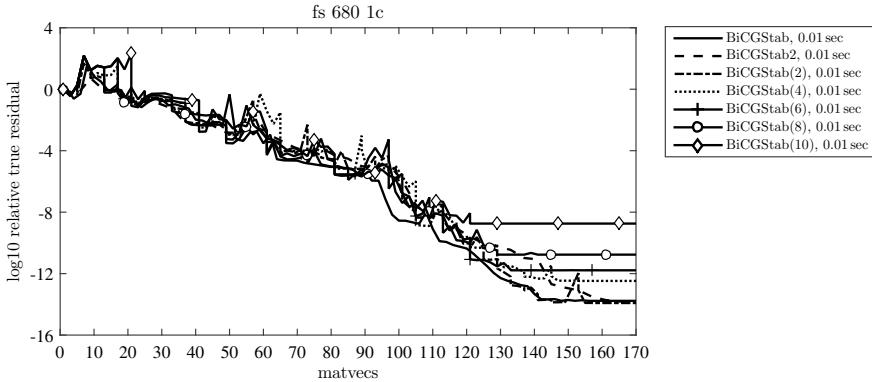


Fig. 9.10 *fs 680 1c*, relative true residual norms, BiCGStab (plain), BiCGStab2 (dashed), BiCGtab(ℓ) $\ell = 2 : 2 : 10$

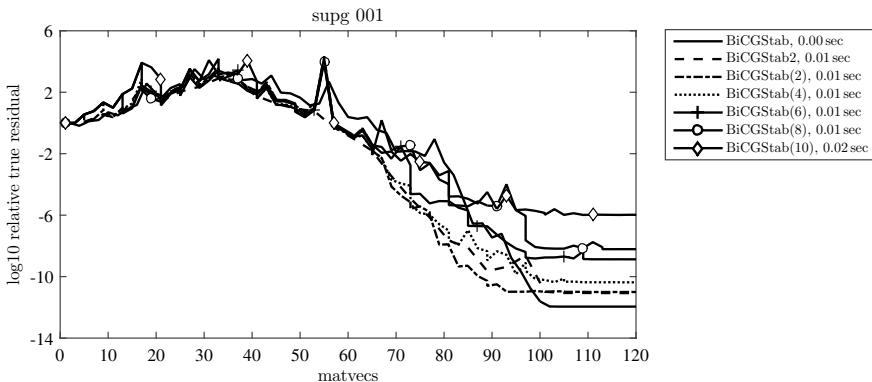


Fig. 9.11 *supg 001*, order 1225, relative true residual norms, BiCGStab (plain), BiCGStab2 (dashed), BiCGtab(ℓ) $\ell = 2 : 2 : 10$

To do some time measurements we consider the problem `raefsky1`. Figure 9.12 shows that when plotted as functions of the number of matrix–vector products the residual norms of all the methods behave the same. Things are slightly different when they are plotted as functions of time. On this example BiCGStab2 and BiCGStab(4) are the fastest methods. We observe that it does not pay to increase ℓ too much (Fig. 9.13).

Except a few examples for which BiCG is worse, Tables 9.1 and 9.2 show that the best accuracies of BiCG, BiCGStab and BiCGStab2 are roughly the same.

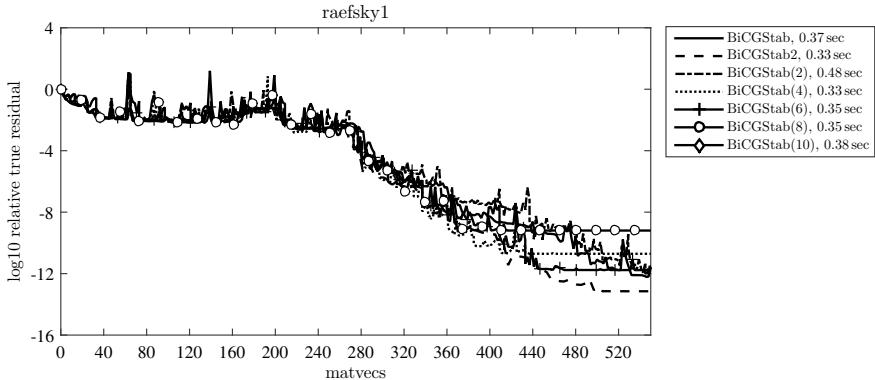


Fig. 9.12 raeftsky1, relative true residual norms, BiCGStab (plain), BiCGStab2 (dashed), BiCGtab(ℓ) $\ell = 2 : 2 : 10$

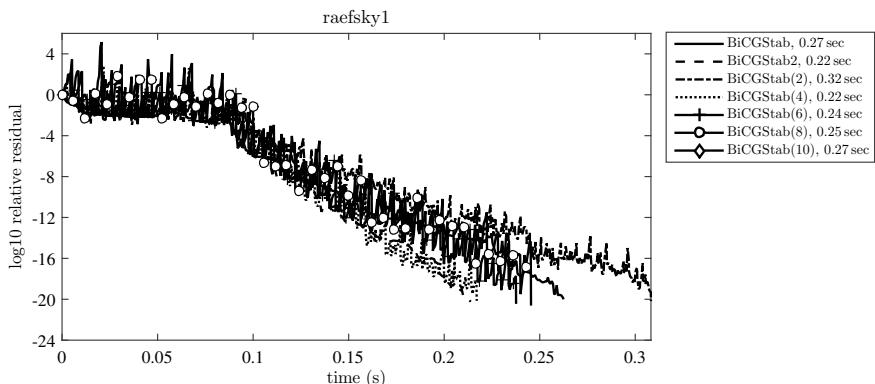


Fig. 9.13 raeftsky1, relative residual norms, BiCGStab (plain), BiCGStab2 (dashed), BiCGtab(ℓ) $\ell = 2 : 2 : 10$

Tables 9.3 and 9.4 show the number of matrix–vector products needed to obtain $\|r_k\| \leq 10^{-10}\|b\|$. A “-” means that the stopping criterion was not satisfied within 10000 iterations. We observe in Table 9.3 that there are cases for which BiCG does better in terms of matrix–vector products than BiCGStab or BiCGStab2. But there are also cases for which BiCG does not converge contrary to BiCGStab or BiCGStab2. This is clearly seen for the supp examples in Table 9.4. These examples arise from the discretization of convection–diffusion problems. BiCG has difficulties to converge when the diffusion term is small compared to the convection term. The examples are listed in a decreasing diffusion coefficient order. We observe that BiCGStab and BiCGStab2 converge but they need more matrix–vector products when the diffusion coefficient decreases. The last three examples arise also from a convection–diffusion problem but with a different discretization scheme and none of the three methods is really working well.

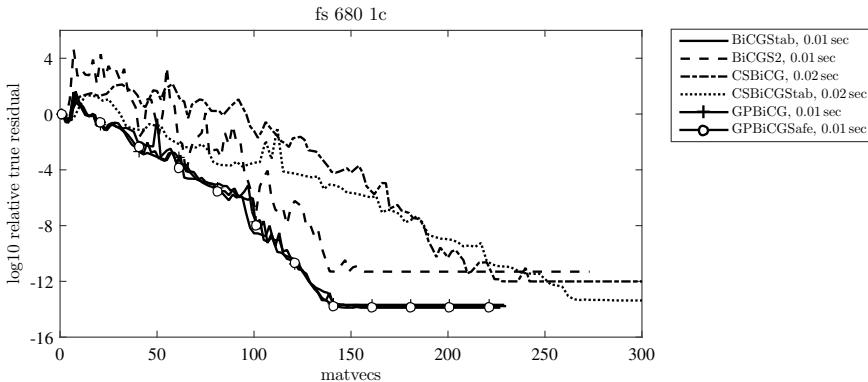


Fig. 9.14 *fs 680 1c*, relative true residual norms, different BiCG-like algorithms

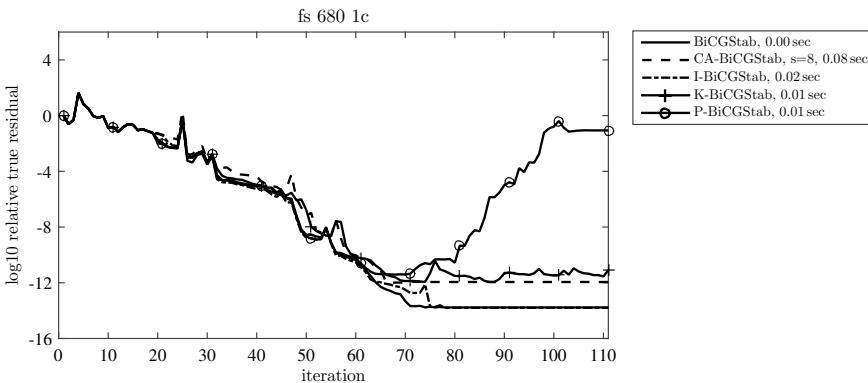


Fig. 9.15 *fs 680 1c*, relative true residual norms, BiCGStab (plain), CA-BiCGStab (dashed), I-BiCGStab (dot-dashed), K-BiCGStab (+), P-BiCGStab (o)

Figure 9.14 displays the relative true residual norms of some methods derived from BiCG or BiCGStab for the problem *fs 680 1c*. We observe that only GPBiCG and GPBiCGSafe behave similarly to BiCGStab.

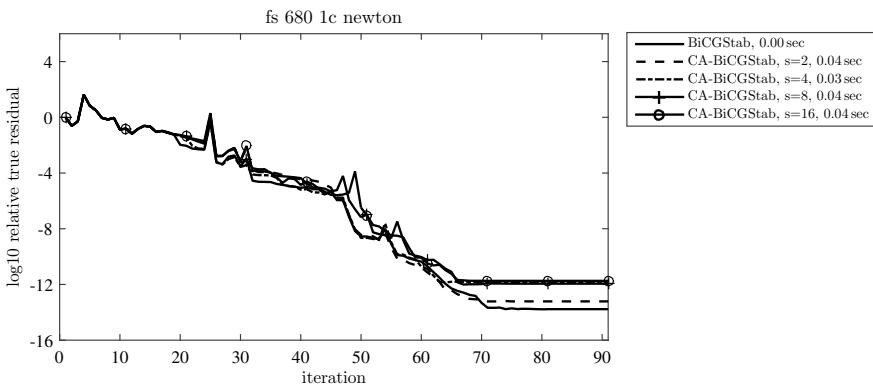
Figures 9.15–9.18 show results for parallel variants of BiCGStab. In CA-BiCGStab we use the Newton basis constructed with 10 Arnoldi iterations and $s = 8$. I-BiCGStab and K-BiCGStab correspond, respectively, to the methods in [1001] and [607]. We observe that the pipelined BiCGStab (P-BiCGStab) in [235] is clearly unstable. This is why it was proposed to use periodic residual replacement to improve it. For the other methods, the maximum attainable accuracy is slightly worse than that of BiCGStab. Figure 9.16 displays the results of CA-BiCGStab for different values of s .

Table 9.1 Minimum relative true residual norms in 10000 iterations at most

matrix	BiCG	BiCGStab	BiCGStab2
pde225	6.92038e-11	2.37490e-11	6.93968e-11
gre 343	9.94142e-11	1.16447e-02	1.83231e-03
jphw 991	1.00000e+00	1.00000e+00	1.00000e+00
pde2961	6.33946e-11	3.45940e-11	5.56858e-11
jagmesh1	5.27938e-11	6.81982e-11	9.77202e-11
bfsa782	4.34038e-11	2.07770e-11	3.25610e-11
dw2048	9.56734e-11	9.09978e-11	4.61005e-11
jagmesh2	7.06850e-11	2.50894e-11	6.33164e-11
raefsky2	7.65026e-11	7.70185e-11	2.46830e-11
fs 680 1c	6.72317e-11	7.29951e-11	9.59722e-11
add20	7.41168e-11	7.82054e-11	8.33912e-11
raefsky1	5.98909e-11	4.21399e-11	7.61647e-11
jagmesh4	7.56573e-11	1.75541e-11	1.60250e-11
fs 680 1	8.67408e-11	8.51807e-11	9.93314e-11
sherman1	9.42360e-11	6.39045e-11	9.82083e-11
nos3	8.91685e-11	6.73433e-11	9.75269e-11
sherman5	2.90316e-10	8.32279e-10	8.52993e-11
cavity05	2.42755e-03	5.95500e-11	4.24562e-11
e05r0500	8.63944e-04	1.00000e+00	4.85787e-01
comsol	9.22722e-11	9.47344e-11	9.72773e-11
olm1000	9.86959e-11	6.85106e-03	1.29145e-03
cavity10	6.82759e-11	9.90222e-11	7.46110e-11
steam2	3.07461e-11	6.41193e-11	8.64238e-11
1138bus	1.00438e-10	8.58226e-11	8.99481e-11
steam1	8.45745e-11	1.38707e-02	1.39302e-03
bcsstk26	1.84377e-07	1.67486e-07	6.86591e-10
nos7	4.02374e-07	1.00000e+00	1.00000e+00
watt1	7.38573e-11	8.54390e-11	9.88099e-03
bcsstk14	8.81060e-03	1.45650e-02	1.82495e-08
fs 183 6	4.53035e-11	9.87447e-11	3.61574e-12
bcsstk20	1.00000e+00	4.27294e-01	3.37627e-01
mcf6	1.00000e+00	1.00000e+00	1.00000e+00
nnc	1.00000e+00	1.00000e+00	1.00000e+00
Insp	1.00000e+00	9.99884e-01	9.62485e-01

Table 9.2 Minimum relative true residual norms in 10000 iterations at most

matrix	BiCG	BiCGStab	BiCGStab2
add32	2.85275e-15	2.75300e-15	3.65644e-15
ex37	1.54595e-15	1.54270e-15	2.30888e-15
memplus	1.13169e-13	2.24759e-13	1.56170e-13
sherman3	6.34459e-11	5.73137e-11	4.79818e-11
wang4	5.27292e-12	1.06559e-14	1.35276e-14
supg 100	1.23889e-14	1.36070e-14	1.39139e-14
supg 1	4.14650e-04	1.60722e-14	1.55904e-14
supg 01	5.15819e-11	4.27320e-12	4.78132e-13
supg 001	1.54702e-01	8.80590e-11	1.96268e-11
supg 0001	6.47855e-01	2.35791e-09	1.95092e-10
supg 00001	1.00000e+00	3.19233e-09	2.11639e-09
supg 000001	1.00000e+00	4.54458e-08	1.09498e-08
convdiff xu 500	8.04236e-02	6.37240e-02	5.87368e-02
convdiff xu 1000	1.39383e-01	1.00840e-01	1.01607e-01
convdiff xu 5000	9.42723e-01	5.47665e-01	1.36500e-10

**Fig. 9.16** fs 680 1c, relative true residual norms, BiCGStab (plain), CA-BiCGStab $s = 2$ (dashed), $s = 4$ (dot-dashed), $s = 8$ (+), $s = 16$ (o)

9.7.3 QMR-like methods

Let us now consider some methods which are related to QMR. Figure 9.19 compares the two-term version of QMR which needs multiplications with A^T and TFQMR which is a transpose-free method not mathematically equivalent to QMR. For the examples fs 183 6 and fs 680 1c TFQMR converges faster than QMR 2t; see Figures 9.19 and 9.20.

Table 9.3 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, nitmax = 10000

matrix	BiCG	BiCGStab	BiCGStab2
pde225	162	106	101
gre 343	2204	—	—
jphw 991	—	—	—
pde2961	600	326	322
jagmesh1	430	840	630
bfsa782	684	810	865
dw2048	3720	4080	3009
jagmesh2	1830	3482	2566
raefsky2	774	742	626
fs 680 1c	194	114	125
add20	856	1176	1050
raefsky1	566	476	409
jagmesh4	1350	2382	1821
fs 680 1	814	516	577
sherman1	1208	958	997
nos3	566	458	453
sherman5	4726	—	4802
cavity05	—	1642	1853
e05r0500	—	—	—
comsol	534	648	610
olm1000	2762	—	—
cavity10	12654	2390	2445
steam2	370	5604	2046
1138bus	6860	10642	6689
steam1	3872	—	—
bcsstk26	—	—	—
nos7	10980	—	—
watt1	986	572	—
bcsstk14	—	—	—
fs 183 6	1176	990	1713
bcsstk20	—	—	—
mcf6	—	—	—
nnc	—	—	—
Insp	—	—	—

Table 9.4 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, nitmax = 10000

matrix	BiCG	BiCGStab	BiCGStab2
add32	172	116	106
ex37	192	124	129
memplus	2310	3020	2298
sherman3	14154	17434	14457
wang4	1432	784	725
supg 100 6400	1308	856	817
supg 1 6400	—	1794	1793
supg 01 6400	1698	646	638
supg 001 6400	—	222	222
supg 0001 6400	—	244	233
supg 00001 6400	—	350	349
supg 000001 6400	—	410	421
convdiff xu 500 8100	—	—	—
convdiff xu 1000 8100	—	—	—
convdiff xu 5000 8100	—	—	9209

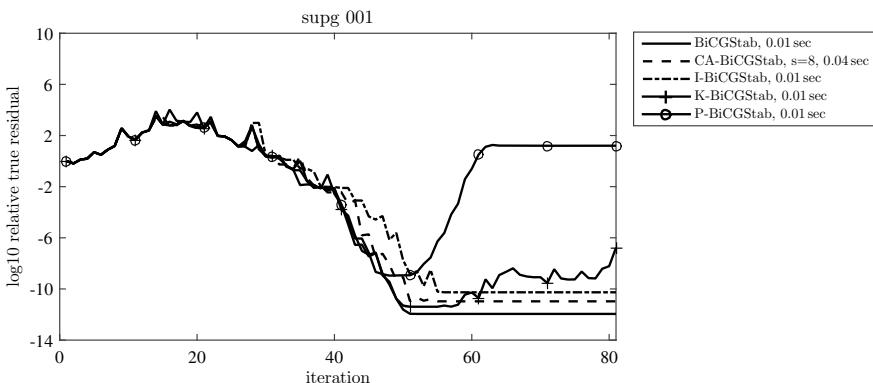
**Fig. 9.17** supg 001, order 1225, relative true residual norms, BiCGStab (plain), CA-BiCGStab (dashed), I-BiCGStab (dot-dashed), K-BiCGStab (+), P-BiCGStab (o)

Figure 9.21 displays the results for some QMR-like methods. Only QMRBiCGStab converges almost as fast as TFQMR when the residual norms are plotted as functions of the number of matrix–vector products. All the other methods do not give very good results.

Things are different for `fs 680 1c` as we see in Figure 9.22. With respect to matrix–vector products the best methods are QMRBiCGStab and QMRBiCGStab2 followed by TFQMR. We observe that maximum attainable accuracies are quite different for these methods. For this example the worst method is TFiQMR.

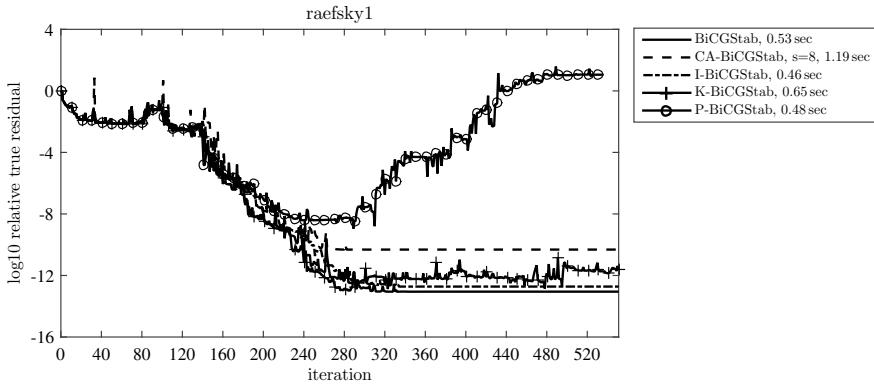


Fig. 9.18 raeftsky1, relative true residual norms, BiCGStab (plain), CA-BiCGStab (dashed), I-BiCGStab (dot-dashed), K-BiCGStab (+), P-BiCGStab (o)

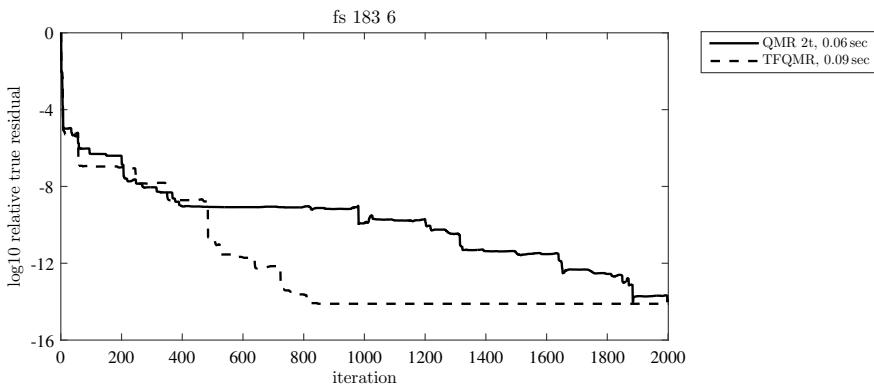


Fig. 9.19 fs 183 6, relative true residual norms, QMR 2t (plain), TFQMR (dashed)

9.8 Historical notes

At the end of the 1970s Peter Sonneveld (Delft University of Technology, The Netherlands) developed a method he called the Induced Dimension Reduction (IDR) method for solving linear systems. It was presented at a Symposium held by the International Union of Theoretical and Applied Mechanics (IUTAM) at the University of Paderborn, Germany, in September 1979 as a joint paper with Peter Wesseling (from the same university); see the proceedings [981] published in 1980. In this first version of IDR the residual vectors can be seen as being given by the product of two polynomials in A times the initial residual, one of the polynomials being such that $q_k(\xi) = (1 - \omega_k \xi) q_{k-1}(\xi)$. This algorithm used a minimization of every second residual norm. IDR did not use the transpose of A . This method did not receive much attention from the numerical linear algebra community.

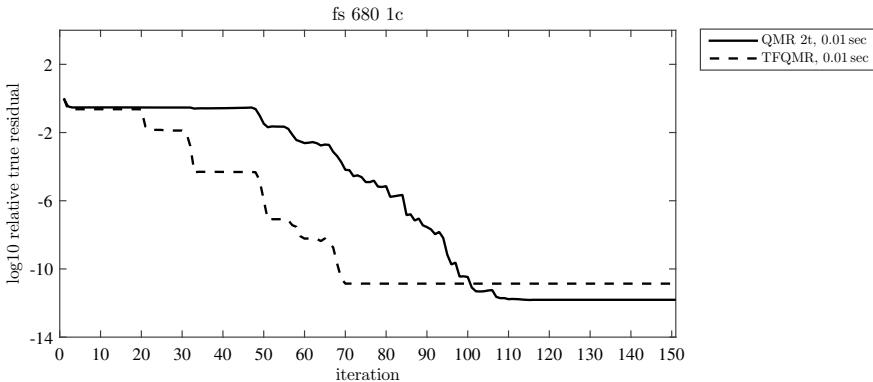


Fig. 9.20 *fs 680 1c*, relative true residual norms, QMR 2t (plain), TFQMR (dashed)

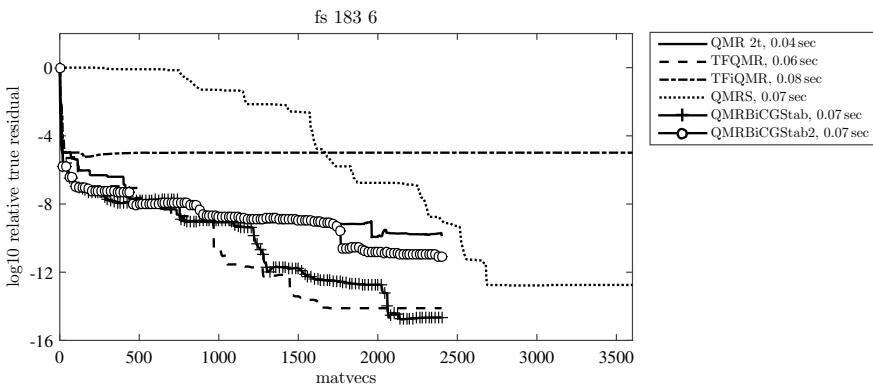


Fig. 9.21 *fs 183 6*, relative true residual norms, different QMR-like algorithms

Nevertheless, Sonneveld retained this idea of product of polynomials and applied it to the Lanczos polynomials by squaring them. This gave the method he (strangely) called Conjugate Gradient Squared (CGS). The paper was submitted to the SIAM J. on Scientific and Statistical Computing (SISSC) on April 24, 1984 but the revised version was only accepted on February 2, 1988 and finally published in January 1989. CGS was based on choosing the auxiliary polynomial equal to the Lanczos polynomial. We have seen that the advantages of CGS (or BiCGS as we called it) is that A^T is not needed and when BiCG converges well, the reduction of the residual norm is better with CGS. However, there are many problems for which the CGS residual norms are even more erratic than those of BiCG. This may also give a large maximum attainable accuracy.

A collaboration between Sonneveld and Henk Van der Vorst (who was also in Delft at that time) resulted in 1990 in a report [942] whose title was *CGSTAB, a more*

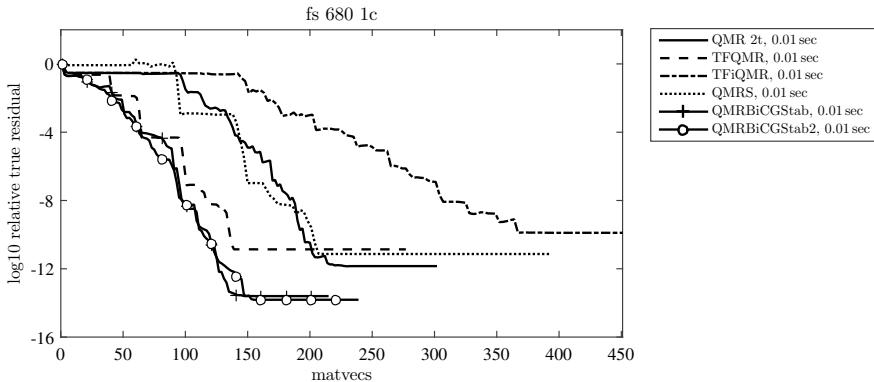


Fig. 9.22 *fs 680 1c*, relative true residual norms, different QMR-like algorithms

smoothly converging variant of CG-S. This method was presented at the Householder Symposium in Tylösand (Sweden) in June 1990 by Van der Vorst.

In Sonneveld's words [864] referring to the problems with CGS, “*To overcome this disadvantage Henk van der Vorst suggested to replace one of the polynomials ϕ_n in the square ϕ_n^2 by a polynomial that could be used for damping out unwished behaviour of CGS. This lead to the development of BiCGSTAB, a method meant as a stable variant of CGS, although many people consider it as a stabilization of BiCG. In fact BiCGSTAB was mathematically equivalent to IDR, only the recurrence formulae differ completely, since they stem from the α 's and β 's in the (Bi)-CG method. Both the author and van der Vorst were well aware of that. But apparently the BiCGSTAB algorithm had a significantly better numerical stability, for which reason the IDR-interpretation was buried! This, after all, appears to be wrong. It was not the theoretical basis (IDR versus BiCG) that caused IDR instability, but a not so lucky implementation of the old IDR algorithm*”.

In an interview in 1992, H. van der Vorst wrote: “*Early ideas by Sonneveld (1984) for improvements in the bi-Conjugate Gradient (Bi-CG) method, for the solution of unsymmetric linear systems, intrigued me for a long time. Sonneveld had a brilliant idea for doubling the speed of convergence of Bi-CG for virtually the same computational costs: CGS. He also published a rather obscure method under the name of IDR. I doubt whether that paper got more than two or three citations altogether. The eventual understanding of that method and the reformulation of it, so that rounding errors had much less bad influence on its speed of convergence, led to the so frequently cited Bi-CGSTAB paper*”.

Van der Vorst joined the Mathematical Institute of the University of Utrecht in 1990. Finally, H. van der Vorst published a paper whose title is *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*. The paper was submitted to SISSC May 21, 1990, accepted for publication (in revised form) February 18, 1991 and finally published in March 1992.

Other researchers soon realized that the new methods CGS and BiCGStab are based on residual polynomials which are products of auxiliary polynomials and the Lanczos polynomials. Martin H. Gutknecht coined the term “Lanczos-type product method” (LTPM) for these methods. In the 1990s and still even up to now many people tried to improve these LTPM methods and developed new ones. This yields a plethora of new Krylov subspace methods.

In 1989 Anthony T. Chronopoulos and Sangbak Ma [227] proposed squaring methods based on the nonsymmetric Lanczos algorithm.

M.H. Gutknecht [480] proposed a complex version of BiCGStab in 1993. He pointed out that the damping coefficients ω_j of BiCGStab are real and this may cause problems since nonsymmetric matrices generally have complex eigenvalues. He proposed to modify BiCGStab to allow the polynomial q_k to have complex zeros. This method was called BiCGStab2. The paper was submitted to the SIAM J. on Scientific Computing (SISC) on September 9, 1991, accepted for publication on August 17, 1992 and published in September 1993.

To further improve on BiCGStab, in 1993 Gerard L.G. Sleijpen and Diederik R. Fokkema [841] proposed BiCGStab(ℓ) in which a polynomial of degree ℓ is used instead of $1 - \omega_j \xi$ in BiCGStab. Hence, the minimization of the residual norm occurs in a subspace of dimension ℓ . This paper was received by ETNA on March 24, 1993, accepted for publication on July 16, 1993 and published in September 1993. Note that BiCGStab(2) is not equivalent to BiCGStab2. See also the report [356] by D.R. Fokkema.

In 1994 Victor Eijkhout [298] considered computational variants of the CGS and BiCGstab methods in a technical report from the University of Tennessee.

In 1996 D.R. Fokkema, G.L.G. Sleijpen and H. van der Vorst [358] introduced a class of methods named generalized CGS (GCGS) methods of which CGS and BiCGStab are special cases. They discussed the disadvantages of squaring the iteration polynomial and they introduced two new methods, CGS2 using another BiCG polynomial (with a different starting shadow vector) and shifted CGS using an approximation of the eigenvalue of largest modulus. These methods were intended to be used as inner solvers in Newton-like iterative methods for solving nonlinear systems of equations. In numerical examples CGS2 gives a smoother convergence behavior than CGS.

In 1997 Tony F. Chan and Qiang Ye [209] considered a combination of (Bi)CGS and BiCGStab. At each iteration the algorithm carries out either a (Bi)CGS step or a BiCGStab step without restarting.

Generalized product-type methods were considered in 1997 by Shao-Liang Zhang [1029]. The polynomials q_k were taken to satisfy a three-term recurrence relation. This paper was submitted in August 1995. This class of methods was further discussed by the same author in 2002 [1030].

A review of Lanczos-type solvers for nonsymmetric linear systems of equations was published by M.H. Gutknecht in Acta Numerica in 1997 [482]. This 127-page paper describes many variants of Lanczos-like methods.

A paper by Stefan Röllin and M.H. Gutknecht published in 2002 whose title is “Variations of Zhang’s Lanczos-type product method” [773] proposed alterna-

tive algorithms that are mathematically equivalent to GPBiCG in [1029] but were intended to have a better maximum attainable accuracy. A look-ahead version was considered in Röllin's diploma thesis.

As we said above, many papers were written on product-type methods by Japanese researchers. Let us cite a few of them. In 2002 Seiji Fujino published a method called GPBiCG(m, ℓ) which is a hybrid of BiCGStab and GPBiCG methods [389]. In 2003 Shoji Itoh and Yusuke Namekawa considered an improvement of DS-BiCGstab(ℓ) and its application for linear systems in lattice QCD [552]. In 2005 S. Fujino, Maki Fujiwara and Masahiro Yoshida proposed the BiCGSafe method [390]. In 2010 Masaaki Tanio and Masaaki Sugihara published GBi-CGSTAB(s, L) [905]. In 2012 Takashi Sekimoto and S. Fujino proposed variants of BiCGSafe [821]. These authors did a performance evaluation of the GPBiCGSafe method in 2012 [392].

A very interesting method related to BiCGStab was proposed in 1999 by Man-Chung Yeung and Tony F. Chan [1011] but we will comment on it in the chapter devoted to IDR. The derivation was quite involved, but more recently it was reformulated by M.C. Yeung; see [1008, 1009].

Using the Biconjugate Residual (BiCR) method of Tomohiro Sogabe, M. Sugihara and Shao-Liang Zhang [859], Kuniyoshi Abe and G.L.G. Sleijpen [3] in 2010 introduced product-type variants, replacing the "BiCG part" of those methods by BiCR. This yields BiCRStab, GPBiCR and BiCRstab(2). On some examples these methods converge faster than their BiCG counterparts. See also [4] in 2012.

In 2010 G.L.G. Sleijpen, P. Sonneveld and Martin B. Van Gijzen [842] showed how BiCGStab can be considered as an induced dimension reduction method.

After the introduction of product-type methods several people observed that smoothed variants can be squared and that product-type methods can also be smoothed. Many look-ahead variants were also proposed. This added many other papers to the already long product-type list.

The TFQMR algorithm from Roland W. Freund [369] used the CGS basis vectors to compute iterates satisfying a quasi-minimal residual property. The paper was submitted in September 1991 and published in 1993. Note that TFQMR and the QMR algorithm of Freund and Nachtigal which uses A^T are not equivalent. A Fortran code for TFQMR is available in QMRPACK from Netlib but without look-ahead.

R.W. Freund and Theodore Szeto proposed the QMRS algorithm which was obtained by squaring the QMR polynomial. Two reports [384, 385] were issued in 1991–1992 and a paper appeared in the proceedings of a conference [386] in 1992.

T.F. Chan, Lisette De Pillis and H. Van der Vorst squared the Lanczos polynomials to obtain transpose-free implementations of both the QMR and the BiCG methods which are called TFiQMR and TFiBiCG; see [203, 204]. Note that three matrix–vector products are needed per iteration. The paper was published in 1998.

In 1994, motivated by TFQMR, T.F. Chan, Efstratios Gallopoulos, Valeria Simoncini, T. Szeto and Charles H. Tong [205] proposed QMRCGStab which is a quasi-minimum residual extension of BiCGStab.

A family of transpose-free quasi-minimal residual algorithms QMRCGStab(k) was introduced in 1994 by C.H. Tong [919]. Numerical results for QMRCGStab(2) and QMRCGStab(4) were presented.

Unified formulas for obtaining QMR improved approximate solutions were derived by Zhi-Hao Cao [179] in 1998. The resulting QMR variants were applied to the BiCGStab2 algorithm of M.H. Gutknecht [480].

Following an approach by Lu Zhou and Homer F. Walker [1034], QMR smoothing for product-type methods was considered in 1998 by Klaus J. Ressel and M.H. Gutknecht [768].

Most of the papers already cited in this section did not consider the problem of breakdown of the underlying Lanczos process. However, starting in the 1990s some researchers considered applying look-ahead techniques to the product-type methods. This was mainly done by people who had been working on look-ahead for the methods derived from the Lanczos process or BiCG.

In 1991 Claude Brezinski and Hassane Sadok [147] showed how to avoid breakdown in Sonneveld's CGS algorithm. Near-breakdowns were handled using formal orthogonal polynomials in [133] in 1994 by C. Brezinski and Michela Redivo-Zaglia. The resulting Fortran code for the algorithm BSMRZS is still available from Netlib. Look-ahead for BiCGStab was considered in [134] in 1995. In 1998 they considered look-ahead for transpose-free Lanczos-type algorithms [135].

In 1994 and 1996 T.F. Chan and T. Szeto [206, 207] applied the composite step technique of Bank and Chan [68, 69] using 1×1 and 2×2 pivots to BiCGStab and BiCGStab2. It yields the two algorithms CS-BiCGStab and CS-BiCGStab2. But, this does not cure all the possible breakdowns.

In 1997, using the theory of the formal orthogonal polynomials Z.-H. Cao [178] presented breakdown-free BiCGStab and BiCGStab2 algorithms. This paper is purely theoretical, without any numerical experiments. The paper by Brezinski and Redivo-Zaglia [134] is not cited but this may be explained by the fact that Cao's paper was submitted in 1995.

Peter R. Graves-Morris and Ahmed Salam [450] introduced the BiCGStab damping polynomials q_k in a Lanczos/Orthomin algorithm and used the look-around strategy of Graves-Morris to avoid breakdowns. This paper was published in 1999.

In 2000 M.H. Gutknecht and K.J. Ressel [487] considered look-ahead procedures for general Lanczos-type product methods based on three-term recurrences. They also discussed different look-ahead strategies. In Gutknecht's inimitable style the proposed methods have strange names like LABiOSTab and LABiOXMR2.

Unfortunately there are not many papers containing an analysis of product-type methods in finite precision arithmetic. In most papers all we have are numerical experiments on a few examples of small dimensions. As far as we know, there is no complete finite precision analysis of, for instance, BiCGStab which is the most cited of the product-type methods. There is also no study of the (possible) loss of biorthogonality which may lead to inaccurate coefficients and to a delay of convergence.

However, tentatives were made to improve the accuracy of BiCGStab and BiCGStab(ℓ). We have seen that the BiCG coefficients are used in methods like BiCGS or BiCGStab. In 1995 G.L.G. Sleijpen and H. Van der Vorst [847] proposed some approaches that help to reduce the effects of local rounding errors on the BiCG iteration coefficients in hybrid schemes. They also proposed heuristics to vary ℓ for

improving the accuracy. The paper [846] is also devoted to the computation of the BiCG coefficients and can be viewed as a complement of [847].

In 1996 the authors discussed strategies aimed at maintaining the computed residuals close to the true residuals $b - Ax_k$; see [848]. This has also an effect on the maximum attainable accuracy.

For detecting near-breakdowns in the look-ahead algorithms the user needs to define some thresholds, which may be problem dependent, to be able to detect the lengths of the jumps correctly. To avoid this, Jean-Marie Chesneaux and Ana Matos [218] in 1996 used stochastic arithmetic on the code BSMRZS of C. Brezinski and M. Redivo-Zaglia (which implements a look-ahead CGS algorithm). They were able to obtain good results without using any thresholds but stochastic arithmetic is much more expensive than usual double-precision finite precision arithmetic.

The effect of changing the rounding mode in BiCGStab computations was studied experimentally using the CADNA software (see [563]) by Marc Montagnac and J-M. Chesneaux [694] in 2000. They presented a dynamic strategy which allows to detect breakdowns and near-breakdowns and to choose between a look-ahead technique and a restart.

Concerning parallel versions of product-type methods, the improved BiCGStab variant [1001] was presented by Laurence T. Yang and Richard P. Brent in 2002. As we said above the presentation of this algorithm contains some mistakes that can fortunately be easily corrected.

A parallel BiCGStab(2) algorithm [469] was published in 2009 by Tong-Xiang Gu, Xian-Yu Zuo, Xing-Ping Liu and Pei-Lu Li.

The reordered BiCGStab method [607] was published in 2012 by Boris Krasnopolsky.

Communication-avoiding (Bi)CGS and BiCGStab methods were described in the Ph.D. thesis [188] of Erin C. Carson. Matlab codes implementing sequential versions of CA-CGS and CA-BiCGStab are available.

A communication-hiding pipelined variant of BiCGStab [235] was proposed in 2016 by Siegfried Cools and Wim Vanroose; see also [233, 234].

Chapter 10

The IDR family



In this chapter we describe several algorithms using the same framework. They are collectively known as IDR algorithms. The acronym IDR means *Induced Dimension Reduction*. This name comes from the fact that the successive residual vectors are constructed to live in subspaces of decreasing dimensions. To explain where these ideas are coming from we start by considering algorithms that were designed in the 1970s.

10.1 The primitive IDR algorithm

In this section we describe a simple algorithm studied by P. Sonneveld in the 1970s. It can be considered as the root of the IDR family tree.

Proposition 10.1 *Let B be a nonsingular square matrix of order n , r_0 be a given vector in \mathbb{C}^n and $r_1 = Br_0$. We define iterates for $k \geq 1$ as*

$$r_{k+1} = B[r_k - \gamma_k(r_k - r_{k-1})], \quad \gamma_k = \frac{p^*r_k}{p^*(r_k - r_{k-1})},$$

where p is a given vector not orthogonal to any invariant subspace of B . Then, $r_{2n} = 0$.

Proof The scalar γ_k is computed such that $r_k - \gamma_k(r_k - r_{k-1}) = (1 - \gamma_k)r_k + \gamma_k r_{k-1}$ is orthogonal to p . Let p^\perp be the orthogonal complement of p . Then, $r_{k+1} \in B(p^\perp)$. This applies to r_2 and r_3 but since they are both in $B(p^\perp)$ it follows that $r_4 \in B^2(p^\perp)$. By induction we see that $r_{2k} \in B^k(p^\perp)$. It also yields that r_{2k+1} belongs to the same subspace. But r_{2k} belongs also to all the subspaces $B^j(p^\perp)$, $j = 1, \dots, k-1$ and therefore to the intersection of these subspaces.

Let $y_k = r_{2k-1} - \gamma_k(r_{2k-1} - r_{2k-2}) = B^{-1}r_{2k}$. We have $y_k^* p = 0$ and $r_{2k}^* B^{-*} p = 0$. It yields $r_{2k}^* [B^{-*}]^j p = 0$, $j = 1, \dots, k$. This shows that r_{2k} is orthogonal to the Krylov subspace $\mathcal{K}_k(B^{-*}, B^{-*}p)$. The dimension of this Krylov subspace increases with k up to the grade of $B^{-*}p$ with respect to B^{-*} . If the dimension of the subspace to which r_{2k} is orthogonal does not grow it means that we have found an invariant subspace of B which is orthogonal to p . But the generic situation is that the final Krylov subspace is of dimension n and $r_{2n} = 0$. \square

The result can be used for solving $Ax = b$ by writing the problem as $x = (I - A)x + b = Bx + b$, with

$$B \equiv I - A.$$

Let $r_0 = b - Ax_0$ and $r_1 = (I - A)r_0$. From the formulas we clearly see that $r_k = p_k(A)r_0$ where p_k is a polynomial of degree k with $p_k(0) = 1$. So this can be considered as a Krylov method. The polynomials p_k satisfy a very simple recurrence relation. From $r_{k+1} = (I - A)(r_k - \gamma_k(r_k - r_{k-1}))$, we can also recover the approximate solutions by

$$x_{k+1} = x_k + r_k - \gamma_k(x_k - x_{k-1} + r_k - r_{k-1}).$$

Since the dimension of the subspaces to which the residual vectors belong is decreasing this technique received the name Induced Dimension Reduction (IDR). We have seen in Proposition 10.1 that, under mild conditions, we have $r_{2n} = 0$. This is to be compared with what we obtain with GMRES for which at least we have $r_n = 0$. But, in GMRES or BiCG we enforce orthogonality or biorthogonality with a Krylov subspace of growing dimension. In the primitive IDR method orthogonality is enforced with a single vector p . We note that contrary to BiCG we do not need the transpose matrix. We observe also that when $k \geq n$ the vectors r_k become linearly dependent if this was not already the case before.

This algorithm was never published probably since it does not always converge in finite precision arithmetic and it is unstable for some problems. Moreover, eventually having to do $2n$ iterations is not very appealing for large problems. It was also proposed to use a Gauss–Seidel splitting of $A = D + L + U$ where D is diagonal and L (resp. U) is lower (resp. upper) triangular. Then, we can solve $x = Bx + (D + L)^{-1}b$ with $B = -(D + L)^{-1}U$. This method was called Accelerated Gauss–Seidel (AGS); see [862, 864]. We observe that it is a kind of preconditioning of the basic iteration.

10.2 The first IDR algorithm

The derivation of what is considered to be the really first IDR algorithm, published in [981], is based on the so-called IDR theorem. We use the formulation given in [842].

Lemma 10.1 Let \mathcal{G}_0 and \mathcal{S} be subspaces of \mathbb{C}^n , μ_0, μ_1 be complex numbers and $\mathcal{G}_1 = (\mu_0 I - A)(\mathcal{G}_0 \cap \mathcal{S})$. Then,

1. If $\mathcal{G}_1 \subset \mathcal{G}_0$, then $(\mu_1 I - A)(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$.
2. If $\mathcal{G}_1 = \mathcal{G}_0 \neq 0$ then $\mathcal{G}_0 \cap \mathcal{S}$ contains an eigenvector of A .

Proof Assume $\mathcal{G}_1 \subset \mathcal{G}_0$. Then,

$$(\mu_0 I - A)(\mathcal{G}_1 \cap \mathcal{S}) \subset (\mu_0 I - A)(\mathcal{G}_0 \cap \mathcal{S}) = \mathcal{G}_1.$$

This implies that $(\mu_1 I - A)(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$ since this is true if and only if $(\mu_0 I - A)(\mathcal{G}_1 \cap \mathcal{S}) \subset \mathcal{G}_1$.

If $\mathcal{G}_1 = \mathcal{G}_0$ we have

$$\dim(\mathcal{G}_0) = \dim((\mu_0 I - A)(\mathcal{G}_0 \cap \mathcal{S})) \leq \dim(\mathcal{G}_0 \cap \mathcal{S}) \leq \dim(\mathcal{G}_0).$$

It yields $\dim(\mathcal{G}_0 \cap \mathcal{S}) = \dim(\mathcal{G}_0)$. Since these subspaces are of finite dimension, we have $\mathcal{G}_0 = \mathcal{G}_0 \cap \mathcal{S}$. Therefore,

$$\mathcal{G}_0 = \mathcal{G}_1 = (\mu_0 I - A)(\mathcal{G}_0 \cap \mathcal{S}) = (\mu_0 I - A)(\mathcal{G}_0),$$

and $\mu_0 I - A$ has an eigenvector in \mathcal{G}_0 unless $\mathcal{G}_0 = \{0\}$. □

In most IDR algorithms the subspace \mathcal{S} is defined as the orthogonal complement of the linear subspace spanned by the columns of a given matrix that we denote as $\tilde{\mathcal{R}}_0$ which is $n \times s$ where s is a given (small) integer. This is denoted as $\mathcal{S} = \tilde{\mathcal{R}}_0^\perp$. Then, we can formulate the IDR theorem.

Theorem 10.1 Let $\mathcal{G}_0 = \mathbb{C}^n$ and $\mu_j, j = 1, \dots$, be a sequence of complex numbers. We define

$$\mathcal{G}_{j+1} = (\mu_j I - A)(\mathcal{G}_j \cap \tilde{\mathcal{R}}_0^\perp), j = 0, 1, \dots \quad (10.1)$$

If $\tilde{\mathcal{R}}_0^\perp$ does not contain an eigenvector of A , then for $j = 0, 1, \dots$ and unless $\mathcal{G}_j = \{0\}$, $\mathcal{G}_{j+1} \subset \mathcal{G}_j$ and $\dim(\mathcal{G}_{j+1}) < \dim(\mathcal{G}_j)$.

Proof We proceed by induction using Lemma 10.1 and $\mathcal{S} = \tilde{\mathcal{R}}_0^\perp$. □

This theorem is also true if $\mathcal{G}_0 = \mathcal{K}_n(A, v)$ for a given vector v . The first or sometimes called the “classical” IDR algorithm to be described below exploits the idea that we have seen at work in the primitive IDR algorithm, that is, to generate residual vectors which are in subspaces of decreasing dimensions until we get a zero residual vector. This is done by enforcing orthogonality against a single given vector. The new ingredient is to use a variable splitting $I - \omega_j A$ instead of $I - A$. The coefficients ω_j are computed every second iteration and chosen to minimize the residual norm; see [981]. Note that this is slightly different from what we have in Theorem 10.1 in which we use $\mu_j I - A$ to avoid an hypothesis on ω_j . In the algorithm, the linear factor is chosen as $I - \omega_j A$ to obtain a residual polynomial

with value 1 at the origin but we have to assume that $\omega_j \neq 0$. We observe that, even though the IDR theorem is more general, this first algorithm used only one vector p to define \tilde{R}_0 .

In this algorithm, we have $r_0 = b - Ax_0$ and $r_1 = (I - \omega_0 A)r_0$. For even steps, the residuals are defined as

$$r_{2\ell} = (I - \omega_\ell A)(r_{2\ell-1} + \gamma_{2\ell}(r_{2\ell-1} - r_{2\ell-2})), \quad (10.2)$$

similar to what is done in the primitive IDR algorithm. But for odd steps, the residuals are defined as

$$r_{2\ell+1} = (I - \omega_\ell A)(r_{2\ell} + \gamma_{2\ell+1}(r_{2\ell-1} - r_{2\ell-2})), \quad (10.3)$$

using the same parameter ω_ℓ as in relation (10.2).

A code corresponding to the algorithm published in [981] is following. In this code, the coefficients `gamma` are computed to make the vector `r + gamma * dg` orthogonal to the vector p . Hence this vector belongs to $\mathcal{S} = \{p\}^\perp$. The vector p is chosen as a random vector but it can be user-defined as well. We could have tested the residual norm for convergence immediately after computing `r`. The random number generator is initialized using `rng` to obtain reproducible results.

```
function [x,ni,resn] = IDR_classical(A,b,x0,epsi,nitmax);
%
nb = norm(b);
nA = size(A,1);
x = x0;
r = A * x - b;
resn = zeros(1,nitmax+1);
resn(1) = norm(r);
rng('default');
p = randn(nA,1); % shadow vector
s = r;
t = A * s;
om = (t' * s) / (t' * t);
dx = -om * s;
dr = -om * t;
x = x + dx;
r = r + dr;
dg = dr;
dy = dx;
gamma = -(p' * r) / (p' * dr);
resn(2) = norm(r);
ni = 2;
%
for k = 2:nitmax
    ni = ni + 1; % number of iterations
    s = r + gamma * dg;
```

```

t = A * s; % matrix vector product
if mod(k,2) == 0
    om = (t' * s) / (t' * t);
end % if
dx = gamma * dy - om * s;
dr = gamma * dg - om * t;
x = x + dx;
r = r + dr;
if mod(k,2) == 1
    dg = dr;
    dy = dx;
end % if
gamma = -(p' * r) / (p' * dg);
nresidu = norm(r);
resn(ni) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k

```

This algorithm can be in trouble if ω_j becomes small. A possibility to get away from that problem is to use the technique recommended in [847]. We replace the computation of om by $\text{om} = \text{omega}(t,s)$ where the function is defined as follows.

```

function om = omega(t,s)
angle = 0.7;
ns = norm(s);
nt = norm(t);
ts = t' * s;
rho = abs(ts / (nt * ns));
om = ts / (nt * nt);
if rho < angle
    om = om * angle / rho;
end % if
end

```

Note that in the previous IDR code, contrary to our usual notation, the residual vectors are defined as $r_k = Ax_k - b$. We can rewrite (10.2) and (10.3) as

$$\begin{aligned} A[\omega_\ell \{r_{2\ell-1} + \gamma_{2\ell}(r_{2\ell-1} - r_{2\ell-2})\}] &= -r_{2\ell} + (r_{2\ell-1} + \gamma_{2\ell}(r_{2\ell-1} - r_{2\ell-2})), \\ A[\omega_\ell \{r_{2\ell} + \gamma_{2\ell+1}(r_{2\ell-1} - r_{2\ell-2})\}] &= -r_{2\ell+1} + (r_{2\ell} + \gamma_{2\ell+1}(r_{2\ell-1} - r_{2\ell-2})). \end{aligned}$$

Let $R_{n,k} = (r_0 \ r_1 \ \cdots \ r_{k-1})$. The above relations for the residuals can be written in matrix form as

$$AR_{n,k}Y_kD_\omega^{(k)} = R_{n,k+1}\underline{H}_k, \quad (10.4)$$

where the columns of the $(k+1) \times k$ upper Hessenberg matrix \underline{H}_k are

$$\underline{H}_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\gamma_j \\ 1 + \gamma_j \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{for } j \text{ even,} \quad \underline{H}_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\gamma_j \\ \gamma_j \\ 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{for } j \text{ odd } j > 1.$$

The bottom entry -1 is in position $(j+1, j)$. The matrix Y_k is the upper triangular part of \underline{H}_k , that is, of H_k . The first column of \underline{H}_k is $e_1 - e_2$. The matrix $D_\omega^{(k)}$ is a diagonal matrix whose j th diagonal entry is ω_ℓ with $\ell = \lfloor j/2 \rfloor$.

Relation (10.4) is a generalized Hessenberg (or Arnoldi-like) relation. Let us try to show that this IDR algorithm is a generalized Q-OR method as defined in Section 3.10. When solving $H_k y = e_1$ (where y is not related to the columns of Y_k) one can see that $[y]_k = 1$. Since $h_{k+1,k} = -1$ we see that the residual vectors satisfy

$$R_k(H_k y - e_1) + h_{k+1,k}[y]_k r_k = -r_k.$$

Hence, this IDR algorithm is a generalized Q-OR method, y is given by $H_k y = e_1$ but since the “basis” vectors are the residuals themselves they are obtained directly from (10.4) without having to solve a linear system.

10.3 IDR(s) and variants

The first IDR algorithm remained mostly unnoticed for many years but a rebirth took place in [947] where the idea of using more than one shadow vector was used. As we have seen, this idea was already there in the IDR theorem.

Before stating the algorithms, let us observe that the subspaces used by the IDR algorithms and defined in Theorem 10.1 are related to Krylov subspaces; see [839, 842]. Let p_j be the polynomial of degree j defined as $p_j(\xi) = \prod_{i=0}^{j-1} (\mu_i - \xi)$ where the μ_j ’s are as in Theorem 10.1. The following proposition was proved in [842].

Proposition 10.2 *Assume \tilde{R}_0^\perp does not contain an eigenvector of A . The subspaces \mathcal{G}_j defined by (10.1) are characterized by*

$$\mathcal{G}_j = \{p_j(A)v \mid v \perp \mathcal{K}_j(A^*, \tilde{R}_0)\} = p_j(A)[\mathcal{K}_j(A^*, \tilde{R}_0)]^\perp, \quad (10.5)$$

where $\mathcal{K}_j(A^*, \tilde{R}_0)$ is defined as in Chapter 2. Moreover, let

$$\mathcal{K}(A^*, \tilde{R}_0) = \cup_{j=0}^{\infty} \mathcal{K}_j(A^*, \tilde{R}_0).$$

Then the following statements are equivalent:

1. There is an eigenvector of A that is orthogonal to \tilde{R}_0 .
2. $\mathcal{K}(A^*, \tilde{R}_0) \neq \mathbb{C}^n$.
3. There exists $x \neq 0$ such that $\mathcal{K}(A, x) \perp \tilde{R}_0$.

Proof The proof is by induction. Assume (10.5) holds. Then,

$$\mathcal{G}_j \cap \tilde{R}_0^\perp = \{p_j(A)v \mid p_j(A)v \perp \tilde{R}_0, v \perp \mathcal{K}_j(A^*, \tilde{R}_0)\}.$$

Since $v \perp \mathcal{K}_j(A^*, \tilde{R}_0)$, $p_j(A)v \perp \tilde{R}_0$ if and only if $A^j v \perp \tilde{R}_0$ or $v \perp [A^*]^j \tilde{R}_0$ or any linear combination of the columns of this matrix. It means that v is orthogonal to $\mathcal{K}_{j+1}(A^*, \tilde{R}_0)$ and

$$\mathcal{G}_j \cap \tilde{R}_0^\perp = \{p_j(A)v \mid v \perp \mathcal{K}_{j+1}(A^*, \tilde{R}_0)\}.$$

The subspace \mathcal{G}_{j+1} is the image by $\mu_j I - A$ of this subset. Hence,

$$\mathcal{G}_{j+1} = \{p_{j+1}(A)v \mid v \perp \mathcal{K}_{j+1}(A^*, \tilde{R}_0)\}.$$

For the equivalence of the three statements, see [842]. □

Note that the characterization (10.5) shows that to increase the order (but not the dimension) of the subspaces \mathcal{G}_j by one, we need to increase the order of $\mathcal{K}_j(A^*, \tilde{R}_0)$ by one and this costs s matrix–vector products.

Even though the dimension of \mathcal{G}_j depends on the choice of \tilde{R}_0 , the generic situation is that $\dim(\mathcal{G}_j) = n - \dim(\mathcal{K}_j(A^*, \tilde{R}_0))$. Let us assume that none of the μ_j 's is an eigenvalue of A and, following [839], let $\mathcal{W}_j = [p_j(A)]^{-*} \mathcal{K}_j(A^*, \tilde{R}_0)$. Then,

$$\mathcal{W}_j = \mathcal{K}_j(A^*, [p_j(A)]^{-*} \tilde{R}_0),$$

and let $d_j = \dim(\mathcal{W}_j) = \dim(\mathcal{K}_j(A^*, \tilde{R}_0))$. The subspaces \mathcal{W}_j are nested and

$$d_{j+1} - d_j \leq d_j - d_{j-1} \leq s,$$

see [839].

We would like to generate residual vectors such that

$$r_k \in \mathcal{G}_j \quad \text{for } j = \lfloor k/(s+1) \rfloor. \tag{10.6}$$

We have the following result from [839]; see also [842].

Proposition 10.3 *Let $\mathcal{W}_j = [p_j(A)]^{-*} \mathcal{K}_j(A^*, \tilde{R}_0)$. A vector r belongs to \mathcal{G}_j if and only if $r \perp \mathcal{W}_j$.*

Proof The result is almost obvious from Proposition 10.2. We have

$$\begin{aligned}
\mathcal{G}_j^\perp &= \{w \mid w^* p_j(A)v = 0, v \in \mathcal{K}_j(A^*, \tilde{R}_0)^\perp\}, \\
&= \{p_j(A)^{-*} y \mid y^* v = 0, v \in \mathcal{K}_j(A^*, \tilde{R}_0)^\perp\}, \\
&= p_j(A)^{-*} \{y \mid y^* v = 0, v \in \mathcal{K}_j(A^*, \tilde{R}_0)^\perp\}, \\
&= p_j(A)^{-*} \mathcal{K}_j(A^*, \tilde{R}_0), \\
&= \mathcal{W}_j.
\end{aligned}
\quad \square$$

The IDR subspaces \mathcal{G}_j are polynomial images of the orthogonal complements of subspaces of increasing dimensions. Hence, they are a sequence of nested spaces of shrinking dimension. The relation between two consecutive subspaces \mathcal{G}_j can be generalized as follows.

Proposition 10.4 *Let ℓ be a positive integer. Then,*

$$\mathcal{G}_{j+\ell} = p_{j,\ell}(A)(\mathcal{G}_j \cap \mathcal{K}_j(A^*, \tilde{R}_0)^\perp), \quad (10.7)$$

where $p_{j,\ell}$ is the polynomial

$$p_{j,\ell}(\xi) = \prod_{i=j+1}^{j+\ell} (1 - \omega_i \xi).$$

Proof See [714], Lemma 2, page 23. \square

There exist several variants of IDR(s). The first one was proposed in [865]. The algorithm described in that paper constructs the residual vectors from the previous residual vectors as $r_{k+1} = (I - \omega_j A)v_k$ with $v_k \in \mathcal{G}_j \cap \tilde{R}_0^\perp$. The vector v_k is defined as

$$v_k = r_k - \sum_{\ell=1}^s \gamma_\ell^{(k)} \Delta r_{k-\ell},$$

where Δ is the forward difference operator $\Delta r_{k-\ell} = r_{k-\ell+1} - r_{k-\ell}$. The coefficients $\gamma_\ell^{(k)}$ are chosen to enforce the condition $\tilde{R}_0^* v_k = 0$, that is, $v_k \in \mathcal{G}_j \cap \tilde{R}_0^\perp$. Let $\Delta R_k = (\Delta r_{k-1} \cdots \Delta r_{k-s})$. Then, assuming that the matrix of order s is nonsingular, the coefficients are computed by solving the linear system

$$(\tilde{R}_0^* \Delta R_k) \gamma^{(k)} = \tilde{R}_0^* r_k, \quad v_k = r_k - \Delta R_k \gamma^{(k)}, \quad r_{k+1} = v_k - \omega_j A v_k,$$

where $\gamma^{(k)}$ is the vector of the coefficients $\gamma_\ell^{(k)}$. If this can be done (that is, if the matrix is nonsingular), it yields a vector r_{k+1} in \mathcal{G}_{j+1} . This step is repeated s times (with the same value of ω_j) to obtain $s+1$ residual vectors in \mathcal{G}_{j+1} , in accordance with (10.6). Then, by dimension arguments, we may expect the next residual to be in \mathcal{G}_{j+2} . For the initialization, the first s vectors can be generated by any Krylov

method, for instance, Orthores or GMRES. Orthores was chosen in [865]. If the matrix $\tilde{R}_0^* \Delta R_k$ is (nearly) singular we have a situation of (near) breakdown of the algorithm. If this happens we may want to reduce the value of s if this is possible ($s \neq 1$). A Matlab implementation of the algorithm was provided in the Appendix of [865].

Even though the implementations are different, if $s = 1$ and $\tilde{R}_0 = r_0$, IDR(1) is mathematically equivalent to BiCGStab if the coefficients ω_j are chosen in the same way; see [484, 865]. The even-indexed IDR(1) residual vectors are mathematically the same as the BiCGStab residuals. BiCGStab was considered as an IDR method in [842].

With a slight change of notation from [865] and following [494] the recurrences of IDR(1) can be written as

$$s_{k-1} = (1 - \gamma_k)r_{k-1} + \gamma_k r_{k-2}, \quad r_k = (I - \omega_j A)s_{k-1},$$

with $j = \lfloor k/2 \rfloor$ and $k > 1$. Initially, we have $r_1 = (I - \omega_0 A)r_0$. The relation for the residual vector can be written as $\omega_j A s_{k-1} = s_{k-1} - r_k$ or, equivalently,

$$A[\omega_j\{(1 - \gamma_k)r_{k-1} + \gamma_k r_{k-2}\}] = -r_k + (1 - \gamma_k)r_{k-1} + \gamma_k r_{k-2}, \quad \omega_0 A r_0 = r_0 - r_1.$$

Let $R_{n,k} = (r_0 \ r_1 \ \cdots \ r_{k-1})$. In matrix form the relation for the residual vectors is once again,

$$A R_{n,k} Y_k D_\omega^{(k)} = R_{n,k+1} \underline{H}_k, \quad (10.8)$$

where the columns of the upper triangular $k \times k$ matrix Y_k and the upper Hessenberg $(k+1) \times k$ matrix \underline{H}_k are

$$Y_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \gamma_j \\ 1 - \gamma_j \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{H}_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \gamma_j \\ 1 - \gamma_j \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad j > 1.$$

There are $j-2$ zero components at the top of these two vectors. The first column of Y_k is just e_1 and the first column of \underline{H}_k is $e_1 - e_2$. The matrix \underline{H}_k is tridiagonal. The matrix $D_\omega^{(k)}$ is a diagonal matrix whose j th diagonal entry is ω_ℓ with $\ell = \lfloor j/2 \rfloor$.

Relation (10.8) is a generalized Hessenberg relation. Again, this shows that we have a generalized Q-OR method. The polynomials implicitly generated by IDR(1) were considered in [494] where it is shown that

$$r_k = \begin{cases} \Omega_j(A)\rho_j(A)r_0 & \text{if } k = 2j, \\ \Omega_j(A)\hat{\rho}_{j+1}(A)r_0 & \text{if } k = 2j + 1. \end{cases}$$

The polynomial Ω_j is $\Omega_j(\xi) = (1 - \omega_1\xi) \cdots (1 - \omega_j\xi)$. The polynomials ρ_j and $\hat{\rho}_j$ satisfy the following recurrences:

$$\begin{aligned} \rho_j(\xi) &= (1 - \gamma_{2j})\hat{\rho}_j(\xi) + \gamma_{2j}\rho_{j-1}(\xi), \\ \hat{\rho}_{j+1}(\xi) &= (1 - \gamma_{2j+1})(1 - \omega_j\xi)\rho_j(\xi) + \gamma_{2j+1}\hat{\rho}_{j-1}(\xi), \end{aligned}$$

with $\rho_0(\xi) = 1$ and $\hat{\rho}_0(\xi) = 1 - \omega_0\xi$. It turns out that ρ_j is the BiCG residual polynomial; see [494]. We observe that the residual polynomials have the inverses of the coefficients ω_j as zeros which are therefore eigenvalues of the matrices H_k .

The relation (10.8) is still valid for IDR(s) with $s > 1$ but, now, we have

$$Y_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \gamma_s^{(j)} \\ \gamma_{s-1}^{(j)} - \gamma_s^{(j)} \\ \vdots \\ \gamma_1^{(j)} - \gamma_2^{(j)} \\ 1 - \gamma_1^{(j)} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \underline{H}_k e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \gamma_s^{(j)} \\ \gamma_{s-1}^{(j)} - \gamma_s^{(j)} \\ \vdots \\ \gamma_1^{(j)} - \gamma_2^{(j)} \\ 1 - \gamma_1^{(j)} \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad j > s.$$

There are $j - (s + 1)$ zero components at the top of these columns. The $\gamma_\ell^{(j)}$ are the coefficients of the recurrence for the residual vectors; see [494]. The matrices Y_k and \underline{H}_k have an upper bandwidth s .

Another variant was proposed later in [948] that enforces biorthogonality conditions of the intermediate residual vectors and the columns of \tilde{R}_0 . This can be done since, in the IDR framework, the intermediate residuals can be computed in many different ways. In that version the differences $\Delta r_{k-\ell}$ are not used any longer but v_k is defined as

$$v_k = r_k - \sum_{\ell=1}^s \gamma_\ell^{(k)} g_{k-\ell}, \quad (10.9)$$

where the vectors g_i are made orthogonal to columns of \tilde{R}_0 . This is done using the modified Gram–Schmidt algorithm, meaning that we have to compute dot products. Note that these vectors are biorthogonalized but not normalized. The intermediate residual vectors are also orthogonalized against the columns of \tilde{R}_0 .

In the following code (from the pseudocode in [948]) the matrices U are used to update the approximate solution and the matrices $G=A-U$ to update the residual. Note that to avoid additional matrix–vector products these matrices are computed independently. Therefore, if s is large, the relation $G=A-U$ may not be satisfied up to working precision in finite precision arithmetic. The function `orth` orthogonalizes the columns of its input matrix.

```

function [x,ni,resn] = IDR_Bio(A,b,x0,epsi,nitmax,s);
%
% Biorthogonal IDR with s vectors
nb = norm(b);
nA = size(A,1);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resn(1) = norm(r);
rng('default');
P = randn(nA,s); % shadow vectors
P = orth(P);
Pt = P';
om = 1;
G = zeros(nA,s);
U = zeros(nA,s);
M = eye(s);
ni = 0;
%
for k = 1:nitmax
    ni = ni + 1; % number of iterations
    Ptr = Pt * r;
    for kk = 1:s
        % Solve small system and make v orthogonal to P
        c = M(kk:s,kk:s) \ Ptr(kk:s);
        v = r - G(:,kk:s) * c;
        U(:,kk) = U(:,kk:s) * c + om * v;
        G(:,kk) = A * U(:,kk); % matrix vector product
        % Bi-Orthogonalize the new basis vectors
        for j = 1:kk-1
            alp = (Pt(j,:) * G(:,kk)) / M(j,j);
            G(:,kk) = G(:,kk) - alp * G(:,j);
            U(:,kk) = U(:,kk) - alp * U(:,j);
        end % for j
        M(kk:s,kk) = Pt(kk:s,:) * G(:,kk);
        % Make r orthogonal to p_j, j = 1,...,kk
        beta = Ptr(kk) / M(kk,kk);
        r = r - beta * G(:,kk);
    end
end

```

```

x = x + beta * U(:,kk);
if kk < s
    Ptr(kk+1:s) = Ptr(kk+1:s) - beta * M(kk+1:s,kk);
end % if
end % for kk
v = r;
t = A * v; % matrix vector product
om = omega(t,r);
r = r - om * t;
x = x + om * v;
nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
end % for k

```

Note that, in this algorithm, we could have tested for convergence the norms of the residual vectors produced in the loop over kk but this does not make too much a difference if s is small. The matrix P corresponds to \tilde{R}_0 . For its columns we chose s random vectors that we orthogonalize. Codes for IDR are available on M. Van Gijzen's website.

Other implementations of IDR(s) algorithms are described and discussed in [714].

10.4 Other IDR algorithms

Let us consider the methods described more recently in [1022]. They are different from what we have seen in the previous sections of this chapter in that they generate vectors which are of unit norm and locally orthogonal. As we have seen above we would like to compute $s + 1$ vectors $g_{(j-1)(s+1)+1}, \dots, g_{j(s+1)}$ that belong to $\mathcal{G}_{j-1} \setminus \mathcal{G}_j$, like in (10.9). Let us denote

$$\begin{aligned} G_{n,s}^{(j-1)} &= (g_{(j-1)(s+1)+1} \cdots g_{j(s+1)-1}), \\ G_{n,s+1}^{(j-1)} &= \left(G_{n,s}^{(j-1)} \ g_{j(s+1)} \right). \end{aligned}$$

To simplify the notation, the paper [1022] defined local vectors $g_i^{(j-1)} = g_{(j-1)(s+1)+i}$, $1 \leq i \leq s + 1$. A global matrix is defined as

$$G_{k+1} = \left(G_{n,s+1}^{(0)} \ G_{n,s+1}^{(1)} \ \cdots \ G_{n,s+1}^{(j-1)} \ g_1^{(j)} \ \cdots \ g_i^{(j)} \right),$$

with

$$j = \left\lfloor \frac{k+1}{s+1} \right\rfloor, \quad i = k+1 - j(s+1), \quad k \geq 0.$$

The following hypothesis were made:

$$\begin{aligned} \text{rank} \left(G_{n,s+1}^{(j-1)} g_1^{(j)} \cdots g_s^{(j)} \right) &= 2s+1, \quad 1 \leq j < \left\lfloor \frac{k+1}{s+1} \right\rfloor, \\ \text{rank} \left(G_{n,s+1}^{(j-1)} g_1^{(j)} \cdots g_{k-j(s+1)}^{(j)} \right) &= s+1 + (k-j(s+1)), \quad j = \left\lfloor \frac{k+1}{s+1} \right\rfloor, \\ \text{rank}(\tilde{R}_0^* G_{n,s}^{(j)}) &= s, \quad 0 \leq j < \left\lfloor \frac{k+1}{s+1} \right\rfloor. \end{aligned}$$

The vectors $g_i^{(j)}$ are generated to be of unit norm and such that $[G_{n,s+1}^{(j)}]^* G_{n,s+1}^{(j)} = I$, that is, locally orthonormal. This is referred in [1022] as *partial orthogonalization*. The first $s+1$ vectors are computed by using the Arnoldi process for A and the initial residual $r_0 = b - Ax_0$. It yields the matrix $G_{n,s+1}^{(0)}$ of the basis vectors, an $(s+1) \times s$ upper Hessenberg matrix $H_s^{(0)}$ and $h_{1,0}^{(0)} = \|r_0\|$. Then, in each step of the method we compute a vector $c_0^{(j)}$ with s components, solution of the $s \times s$ linear system

$$\tilde{R}_0^* G_{n,s}^{(j-1)} c_0^{(j)} = \tilde{R}_0^* g_{s+1}^{(j-1)},$$

and a new vector $v_0^{(j)} = g_{s+1}^{(j)} - G_{n,s}^{(j-1)} c_0^{(j)}$ which is in the intersection of $G_{n,s+1}^{(j-1)}$ and \tilde{R}_0^\perp . This vector is mapped to \mathcal{G}_j and normalized by

$$r_1^{(j)} = Av_0^{(j)} - \mu_j v_0^{(j)}, \quad h_{1,0}^{(j)} = \|r_1^{(j)}\|, \quad g_1^{(j)} = \frac{r_1^{(j)}}{h_{1,0}^{(j)}},$$

where the coefficient μ_j is computed to minimize $\|r_1^{(j)}\|$ with the safer technique recommended in [847]. Now, we repeat this process to compute s more vectors $g_{i+1}^{(j)}$, $i = 1, \dots, s$ in \mathcal{G}_j . However, the vector $c_i^{(j)}$, $i = 1, \dots, s$ with $s+i$ components can be computed in different ways that we describe below. From this we obtain vectors

$$v_i^{(j)} = g_i^{(j)} - \left(G_{n,s+1}^{(j-1)} g_1^{(j)} \cdots g_{i-1}^{(j)} \right) c_i^{(j)}, \quad r_{i+1}^{(j)} = Av_i^{(j)} - \mu_j v_i^{(j)}, \quad i = 1, \dots, s.$$

The vector $r_{i+1}^{(j)}$ is orthogonalized with respect to the vectors $g_\ell^{(j)}$, $\ell = 1, \dots, i$ with the (modified) Gram–Schmidt method and normalized to obtain $g_{i+1}^{(j)}$ and coefficients $h_{\ell,i}^{(j)}$, $\ell = 1, \dots, i+1$.

This construction yields matrix relations. Let $V_{n,s+1}^{(j)} = \begin{pmatrix} v_0^{(j)} & \cdots & v_s^{(j)} \end{pmatrix}$. Then,

$$V_{n,s+1}^{(j)} = \left(G_{n,s+1}^{(j-1)} \ G_{n,s+1}^{(j)} \right) \underline{U}_{s+1}^{(j)},$$

where the matrix $\underline{U}_{s+1}^{(j)}$ is $2(s+1) \times (s+1)$. The matrix relation linking two consecutive blocks of vectors is

$$\begin{aligned} & A \left(G_{n,s+1}^{(j-1)} \ G_{n,s+1}^{(j)} \right) \underline{U}_{s+1}^{(j)} \\ &= \left(G_{n,s+1}^{(j-1)} \ G_{n,s+1}^{(j)} \right) \left(\begin{pmatrix} 0_{s+1} & 0_{s+1,s} \\ h_{1,0}^{(j)} & \underline{H}_s^{(j)} \end{pmatrix} + \underline{U}_{s+1}^{(j)} \text{diag}(\mu_j \cdots \mu_j) \right), \end{aligned} \quad (10.10)$$

where 0_{s+1} and $0_{s+1,s}$ are a zero vector and matrix of the corresponding dimensions and $\underline{H}_s^{(j)}$ is an $(s+1) \times s$ upper Hessenberg matrix constructed with the coefficients $h_{\ell,i}^{(j)}$.

Putting together all the local relations (10.10) and using the relation of the initial Arnoldi process, we obtain a global matrix relation,

$$AG_{n,k}U_k = G_{n,k+1}\underline{H}_k^t, \quad \underline{H}_k^t = \underline{U}_k + \underline{U}_k D_k, \quad (10.11)$$

where U_k is upper triangular with a unit diagonal and an identity matrix of order s as its first principal block, \underline{U}_k is obtained by appending a zero last row to U_k , \underline{H}_k is an upper Hessenberg matrix and D_k is a diagonal matrix with entries (i,i) equal to 0 for $i = 1, \dots, s$, to μ_1 for $i = s+1, \dots, 2s+1$, to μ_2 for $i = 2s+2, \dots, 3s+2$ and so on. Relation (10.11) is a generalized Hessenberg (or Arnoldi-like) relation as for the other variants of the IDR(s) method we have described in the previous sections; see [494] for the sake of eigenvalue computations.

It remains to know how to compute the coefficients $c_i^{(j)}$. Several ways are reviewed or proposed in [1022]; see also [767]. What we would like is to obtain a reasonable solution of

$$\tilde{R}_0^* \left(G_{n,s+1}^{(j-1)} g_1^{(j)} \cdots g_{i-1}^{(j)} \right) c_i^{(j)} = \tilde{R}_0^* g_i^{(j)}. \quad (10.12)$$

However, the matrix on the left-hand side is $s \times (s+i)$. With our assumptions this underdetermined linear system has an infinite number of solutions. The first possibility, named srIDR, is to set the first i components of $c_i^{(j)}$ to zero and (if possible) to solve an $s \times s$ linear system,

$$\tilde{R}_0^* \left(g_{i+1}^{(j-1)} \cdots g_{s+1}^{(j-1)} g_1^{(j)} \cdots g_{i-1}^{(j)} \right) y^{sr} = \tilde{R}_0^* g_i^{(j)}, \quad c_i^{(j)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ y^{sr} \end{pmatrix}.$$

This is what is used in [865, 948], that is, in IDR(s) and IDR(s)-Bio, even though the implementation is completely different. Here “sr” means “shortest recurrence” since we no longer need the vectors $g_1^{(j-1)}, \dots, g_i^{(j-1)}$.

Another possibility, named fmIDR, is to set the last i components of $c_i^{(j)}$ to zero and

$$\tilde{R}_0^* G_{n,s}^{(j-1)} y^{fm} = \tilde{R}_0^* g_i^{(j)}, \quad c_i^{(j)} = \begin{pmatrix} y^{fm} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We observe that the matrix of the linear system is the same for all the values of $i = 1, \dots, s$. Here “fm” means “factored matrix”. This is what was used in [850] to develop an extension of IDR(s) with a higher order “stabilizing” polynomial.

A third way to compute $c_i^{(j)}$ is to use the minimum norm solution of the underdetermined system. This can be done using the normal equations or preferably the pseudo-inverse of the matrix in (10.12). This is named mnIDR where “mn” stands for “minimum norm”.

A fourth way is described in [1022] but it is not as efficient as the other ones and needs more storage; so, we do not consider it here. Mathematically, the three variants described above are equivalent and lead to the same residual norms if there is no breakdown. However, they may give different results, in particular, for the maximum attainable accuracy, in finite precision arithmetic. In this respect mnIDR seems to be the best variant; see [1022].

The matrices U_k have different nonzero patterns depending on the choice of variant for the solution of the underdetermined system. For srIDR, and $s = 2, k = 9$ we have the following nonzero structure where the x ’s represent nonzero entries:

$$U_9 = \begin{pmatrix} 1 & x & & & & & & & \\ & 1 & x & x & & & & & \\ & & 1 & x & x & & & & \\ & & & 1 & x & x & & & \\ & & & & 1 & x & x & & \\ & & & & & 1 & x & x & \\ & & & & & & 1 & x & x \\ & & & & & & & 1 & x \\ & & & & & & & & 1 \end{pmatrix}.$$

The upper triangular matrices U_k are banded with bandwidth s because we used the latest vectors. The situation is different for fmIDR and mnIDR where the bandwidth is $2s$. For fmIDR we have

$$U_9 = \begin{pmatrix} 1 & x & x & x \\ & 1 & x & x & x \\ & & 1 & \\ & & & 1 & x & x & x \\ & & & & 1 & x & x & x \\ & & & & & 1 & \\ & & & & & & 1 & x \\ & & & & & & & 1 \end{pmatrix},$$

and for mnIDR,

$$U_9 = \begin{pmatrix} 1 & x & x & x \\ & 1 & x & x & x \\ & & 1 & x & x \\ & & & 1 & x & x & x & x \\ & & & & 1 & x & x & x \\ & & & & & 1 & x & x \\ & & & & & & 1 & x & x \\ & & & & & & & 1 & x \\ & & & & & & & & 1 \end{pmatrix}.$$

The nonzero structure of \underline{H}_k is the same whatever the variant is. It has the following block structure:

$$\underline{H}_9 = \begin{pmatrix} x & x \\ x & x \\ & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \\ & & & & & x & x & x \\ & & & & & & x & x \\ & & & & & & & x \\ & & & & & & & & x \end{pmatrix}.$$

Of course, the nonzero structure of \underline{H}_k^t depends on the structure of \underline{U}_k . The relation (10.11) can be used to define generalized Q-OR and Q-MR algorithms. Let $V_{n,k} = G_{n,k} U_k$. The Q-OR iterates are defined by

$$x_k^O = x_0 + V_{n,k} y^O, \quad H_k^t y^O = \|r_0\| e_1,$$

where $H_k^t = H_k + U_k D_k$. The residual vector r_k is proportional to g_k and $\|r_k\| = h_{k+1,k} |[y^O]_k|$. The Q-MR algorithm is defined by solving a small least squares problem,

$$x_k^M = x_0 + V_{n,k} y^M, \quad y^M \text{ minimizes } \| \|r_0\| e_1 - \underline{H}_k^t y \|.$$

Since the matrix \underline{H}_k^t is banded the two approximate solutions can be computed without storing all the vectors g_i . To solve the least squares problem we use, as usual, Givens rotations to annihilate the entries $h_{j+1,j}$ of \underline{H}_k^t which is factored as $\underline{H}_k^t = \underline{Q}_k R_k$ where \underline{Q}_k is orthonormal (or unitary) and R_k is upper triangular. Since \underline{H}_k^t is banded, R_k has bandwidth $s+1$ or $2s+1$ depending on the variant we use. The iterate is

$$x_k^M = x_0 + V_{n,k} R_k^{-1} z_k^M,$$

where $z_k^M = \|r_0\| [\underline{Q}_k^* e_1]_{1:k}$.

Let $W_{n,k} = V_{n,k} R_k^{-1}$. To compute w_k , the last column of $W_{n,k}$, we use $W_{n,k} R_k e_k = V_{n,k} e_k$. Since R_k is banded, the left-hand side of this relation involves only a few vectors w_j . Let ℓ be the bandwidth of R_k (that is, $s+1$ or $2s+1$), then

$$w_k = \frac{1}{[R_k]_{k,k}} \left(v_k - \sum_{i=k-\ell-1}^{k-1} [R_k]_{i,k} w_i \right),$$

where the vectors w_i are the columns of $W_{n,k}$. The vector v_k is obtained easily from the g_i 's since $v_k = G_{n,k} U_k e_k$ and U_k is banded. The approximate solution is $x_k^M = x_0 + W_{n,k} z_k^M$. The length of the vector z_k^M increases by 1 at every iteration but the previous components do not change. Hence,

$$x_k^M = x_0 + W_{n,k} z_k^M = x_0 + W_{n,k-1} z_{k-1}^M + [z_k^M]_k w_k = x_{k-1}^M + [z_k^M]_k w_k.$$

Another Q-MR algorithm based on IDR was also proposed in [946].

The Q-OR iterate x_k^O can also be obtained from a short recurrence. We leave this development to the interested reader.

10.5 Using higher order stabilizing polynomials

All the variants of IDR(s) we have seen above use a first-order polynomial $1 - \omega_j \xi$ or $\mu_j - \xi$ in each step of the algorithm as in BiCGStab. Therefore, we may have the

same problems as with BiCGStab, for instance, with linear systems for which the matrix is (nearly) skew-symmetric, even though the results are generally improved when using values of s larger than one. The need to solve this kind of problems led to the development of BiCGStab2 and BiCGStab(ℓ) with the use of higher order polynomials. Therefore it was natural to try to apply the same remedies, that is, using higher order polynomials, to IDR(s). The resulting method, first proposed in [850], is called IDR(s)stab(ℓ).

The authors of this paper first develop a variant of IDR(s) where the first step with the construction of the $s + 1$ vectors in $\mathcal{G}_j \cap \tilde{R}_0^\perp$ can be easily repeated ℓ times. This first step does not need explicitly the parameters ω_j , even though they are implicitly used through the vectors that are updated. The second step is the application of the polynomial $I - \omega_j A$ to the vectors and $n \times s$ matrices computed in the first step.

To obtain IDR(s)stab(ℓ) the first step is repeated ℓ times and the second step is replaced by a minimization problem using a polynomial of degree ℓ . A least squares problem is solved to compute the coefficients of the polynomial which are real even though the roots (corresponding to generalizations of the reciprocals of the ω_j 's) may be complex conjugate. The derivation of the first step is too technical to be explained in details here but, basically, the same technique as for BiCGStab(ℓ) in [841] is used to generate the powers of A times some vectors; see Section 9.2 and also [767] for details on this construction.

A code derived from the pseudocode in [850] is as follows. The function `pinv` computes the pseudo-inverse of the argument.

```
function [x,ni,resn] = IDRStab(A,b,x0,epsi,nitmax,s,ell);
%
nb = norm(b);
nA = size(A,1);
x = x0;
r = b - A * x;
resn = zeros(1,nitmax+1);
resn(1) = norm(r);
rng('default');
P = randn(nA,s); % shadow vectors
P = orth(P);
Pt = P';
U = zeros(2*nA,s);
v = [r; A * r] / resn(1);
U(:,1) = v;
for k = 2:s
```

```

v0 = v(nA+1:2*nA);
v = [v0; A * v0];
mu = U(1:nA,1:k-1)' * v0;
v = v - U(:,1:k-1) * mu;
v = v / norm(v(1:nA));
U(:,k) = v;
end % for k
ni = 1;
%
for k = 1:nitmax
ni = ni + 1; % number of iterations (steps)
for j = 1:ell
S = Pt * U(j*nA+1:(j+1)*nA,:);
alp = S \ (Pt * r((j-1)*nA+1:j*nA));
x = x + U(1:nA,:) * alp;
r = r - U(nA+1:(j+1)*nA,:) * alp;
r = [r; A * r((j-1)*nA+1:j*nA)]; % matrix vector product
v = r; % q = 1
vj = v(j*nA+1:(j+1)*nA);
beta = S \ (Pt * vj);
v = v - U * beta;
vj = v(j*nA+1:(j+1)*nA);
v = [v; A * vj] / norm(vj); % matrix vector product
V = v;
for q = 2:s
v = v(nA+1:end);
vj = v(j*nA+1:(j+1)*nA);
beta = S \ (Pt * vj);
v = v - U * beta;
vj = v(j*nA+1:(j+1)*nA);
v = [v; A * vj]; % matrix vector product
mu = V(j*nA+1:(j+1)*nA,1:q-1)' * vj;
v = v - V(:,1:q-1) * mu;
v = v / norm(v(j*nA+1:(j+1)*nA));
V(:,q) = v;
end % for q
if j < ell
U = V;
end % if
end % for j
% polynomial step

```

```

R = reshape(r(nA+1:end), [nA,ell]);
gamma = pinv(R) * r(1:nA);
x = x + reshape(r(1:end-nA), [nA,ell]) * gamma;
r = r(1:nA) - R * gamma;
U0 = V(1:nA,:);
U1 = V(nA+1:2*nA,:);
st0 = nA + 1;
st1 = 2 * nA + 1;
for j = 1:ell
    U0 = U0 - gamma(j) * V(st0:st0+nA-1,:);
    U1 = U1 - gamma(j) * V(st1:st1+nA-1,:);
    st0 = st0 + nA;
    st1 = st1 + nA;
end % for j
U = [U0; U1];
nresidu = norm(r);
resn(ni+1) = nresidu;
if nresidu < (epsi * nb) || ni >= nitmax
    break % get out of the k loop
end % if nresidu
%
end % for k

```

As in IDR(s) the matrix P corresponding to \tilde{R}_0 is chosen randomly and its columns are orthonormalized. The matrix U which is $2n \times s$ is initially constructed blockwise. The upper $n \times s$ block contains orthonormal vectors spanning the Krylov subspace $\mathcal{K}_s(A, r_0)$. This is done with the classical Gram–Schmidt algorithm which may be satisfactory only for small values of s . The lower part contains A times the upper part. The matrix S is $s \times s$. There are two nested loops, an outer one on j up to ℓ and an inner loop on q up to s . During the loop on j the number of columns of U is still s but the number of rows increases by n each time as well as the length of the vector r . The matrix V contains intermediate vectors. There are $\ell(s + 1)$ matrix–vector products per step.

The least squares problem to compute the coefficients of the polynomial is solved using the pseudo-inverse but this can be done in many other ways. The drawbacks of this implementation are that some vectors and matrices change size during the iterations and the orthogonalization or biorthogonalization (with respect to \tilde{R}_0) is done with classical Gram–Schmidt-like algorithms. This may be a problem for large values of s . The weaknesses of the implementation in [850] and possible other implementations are considered in [714].

A method related to IDR(s)stab(ℓ) was published independently in [905].

10.6 Residual norms and convergence

The IDR algorithms were designed to have the finite termination property $r_k = 0$ in exact arithmetic for some iteration or step k . Mathematically this must happen after around $n + n/s$ matrix–vector products. These algorithms are based on the dimension reduction given by the IDR theorem. However, except for the local minimizations of the residual norms using the ω_j 's, μ_j 's or the higher order polynomials, there is nothing guaranteeing that the residual norms will eventually decrease. In fact, there is nothing even guaranteeing what we will always get linearly independent basis vectors. We just know in which spaces the residual vectors are living.

Not much is known so far about the convergence of IDR algorithms. But, we have seen that they are generalized Q-OR algorithms and we can use our knowledge of these methods to obtain some results. The algorithms that fit best with our general framework are the IDR(s) algorithms with partial orthogonalization described in Section 10.4. They satisfy relation (10.11) and the Q-OR iterates are defined by

$$x_k^O = x_0 + V_{n,k}y^O, \quad H_k^t y^O = \|r_0\|e_1,$$

where $H_k^t = H_k + U_k D_k$. The residual vector r_k is proportional to g_k and $\|r_k\| = h_{k+1,k} |[y^O]_k|$.

Theorem 10.2 *The residual norms given by IDR(s) with partial orthogonalization on (A, b) and $x_0 = 0$ producing matrices \underline{H}_k^t and U_k for $k = 1, \dots, m$ are identical for $k = 1, \dots, m$ to those given by FOM applied to $(H_m^t, \|b\|e_1)$.*

Proof The IDR residual norms are $\|r_k^{IDR}\| = |h_{k+1,k}^t [y]_k|$ where y is the solution of $H_k^t y = \|b\|e_1$.

If we apply FOM to the problem $(H_m^t, \|b\|e_1)$ it produces the same matrices H_k^t (except may be for the signs of the subdiagonal entries) and the residual norms are $\|r_k^{FOM}\| = |h_{k+1,k}^t [z]_k|$ with $H_k^t z = \|b\|e_1$. Hence, $z = y$ and $\|r_k^{IDR}\| = \|r_k^{FOM}\|$. \square

From our knowledge of FOM convergence we obtain the following result.

Theorem 10.3 *Assume there is no breakdown. Let $H = H_m^t$ which is assumed to be diagonalizable as $H = X_H \Lambda^H X_H^{-1}$. Then, the IDR residual norm is bounded as*

$$\|r_k\| \leq \frac{\|r_0\|}{|c_k|} \|X_H\| \|X_H^{-1}\| \sqrt{\left\lceil \frac{k+1}{s+1} \right\rceil} \min_{p \in \pi_k, p(0)=1} \max_{\lambda \in \sigma(H)} |p(\lambda)|, \quad k \leq m, \quad (10.13)$$

where c_k is the cosine of the k th Givens rotation used to triangularize the upper Hessenberg matrices.

Proof This is directly obtained from what we know about the FOM algorithm; see Section 5.2. The factor $1/|c_k|$ comes from the relation between Q-OR and Q-MR methods. It accounts for the fact that the matrix H_k could be singular and the residual infinite. The factor under the square root is obtained because, with partial orthogonalization,

$$\|G_{n,k+1}\| \leq \sqrt{\left\lceil \frac{k+1}{s+1} \right\rceil},$$

see Lemma 4 in [767]. \square

We note that we would have obtained the same result by defining H to be $H_m^t U_m^{-1}$ and applying FOM to the problem $H_m^t U_m^{-1}$ since the solution of $H_k^t U_k^{-1} z = \|b\|e_1$ has the same last component as before because U_k has diagonal entries equal to 1. Unfortunately, the upper bound in (10.13) involves quantities which are unknown a priori, the eigenvalues and the eigenvectors of H . However, without breakdown, if mathematically the algorithm stops at a point where $\sigma(H) \subset \sigma(A)$, in the maximum we can replace the spectrum of $H = H_m^t U_m^{-1}$ by the spectrum of A . Then we can apply the bounds we have described in Chapter 5.

The paper [863] discussed the IDR convergence using statistical arguments. The goal was to analyze the role of the random shadow vectors in the convergence of the method and to show that for large s the convergence is close to that of GMRES in terms of matrix–vector products. However, unrealistic hypotheses were made such as having s larger than the order of the matrix. Moreover, we have seen that IDR algorithms are more related to FOM than to GMRES.

The residual polynomials were considered in [494] but we cannot get much information from that for our purposes, except that the reciprocals of the ω_j 's are roots of these polynomials.

10.7 A predecessor of IDR: ML(k)BiCGStab

An algorithm that is very close to IDR(s) was proposed in [1011]. In fact, this paper was submitted in 1997, that is, ten years before IDR(s) was proposed. This algorithm called ML(k)BiCGStab by its authors uses k ($k > 1$) left starting vectors or shadow vectors as we denoted them. This idea was in the air since, even though the paper [16] was only published in 2000, the corresponding report dates back to 1996. This paper was concerned with a block Lanczos-type method which can handle the general case of right and left starting blocks of different sizes; see also [1010].

The paper [1011] starts by considering a method called ML(k)BiCG which is an extension of BiCG using k shadow vectors instead of one. First, a Lanczos method for k linearly independent starting vectors is derived. Let v_0 and q_1, \dots, q_k be given

and define $p_{jk+i} = (A^T)^j q_i$. Then, vectors v_ℓ are constructed such that they span the usual Krylov subspaces based on A and v_0 and satisfy

$$v_\ell \perp \text{span}(p_1, \dots, p_\ell), \quad v_\ell \notin \text{span}(p_{\ell+1}), \quad \ell = 0, \dots, \varsigma - 1,$$

where ς is the grade of v_0 with respect to A . Using the same techniques used to derive BiCG from the nonsymmetric Lanczos algorithm, ML(k)BiCG is derived from this Lanczos method with multiple left vectors. We immediately see that the potential weakness of this method is the computation of powers of the transpose matrix in $p_{jk+i} = (A^T)^j q_i$.

Therefore, the authors of [1011] used the same ideas that were considered to obtain (Bi)CGS and BiCGStab from BiCG. However, they did this by successive modifications of the pseudocode of ML(k)BiCG and this is not easy to follow. As in BiCGStab a parameter is introduced to do a local minimization. All this is too technical to be reproduced here. Numerical experiments that look very promising were provided in [1011].

Unfortunately, this algorithm did not draw too much attention, probably because its derivation was too complicated. A new derivation and a reformulation of the algorithm as well as some simplifications were proposed by one of the authors in [1008]. Again, the derivation is done by successive stages, eight stages being done for going from ML(k)BiCG to the final algorithm which is much simpler and easier to code than the one in [1011]. If $k = 1$ the algorithm is equivalent to BiCGStab and if $k > \varsigma$ it is equivalent to FOM if the left vectors are chosen properly. So, it can be seen as a bridge between short recurrence methods and long recurrence methods. IDR(s) is related to ML(k)BiCGStab but not completely equivalent although this depends on the variant of ML(k)BiCGStab that is considered; see [1008].

Matlab codes are given in [1008] and are available on M.C. Yeung's website. Another variant using A^T is described in [1009].

10.8 Parallel computing

The first IDR(s) algorithm presented in [865] has a single global synchronization point per matrix–vector product except for the computation of ω . The IDR(s)-Bio variant has $s(s + 1)/2 + 2$ global synchronization points per cycle. Another variant named IDR(s)-minsync derived from IDR(s)-Bio was proposed in [230]; see also [229]. This algorithm uses the classical Gram–Schmidt algorithm instead of the modified Gram–Schmidt. It has only one global synchronization point per step. It was also proposed to use sparse column vectors for \tilde{R}_0 to minimize communications. A model of parallel performance of the algorithm was set up to choose a value of s minimizing the computing time.

See also some comments on IDR(s) and parallel computing in [1021].

10.9 Finite precision arithmetic

A paper addressing the problem of the behavior of IDR algorithms in finite precision arithmetic is [1022]. It considers perturbed generalized Hessenberg relations,

$$AG_{n,k}U_k + F_{n,k} = G_{n,k+1}\underline{H}_k^t,$$

where $F_{n,k}$ is the perturbation term due to rounding errors. This term is analyzed for the IDR variants with partial orthogonalization we described above. Under some assumptions on the condition number of A , the assumptions that the coefficients μ_j are not too large and the fact that there is no breakdown, the norm of $F_{n,k}$ can be bounded as

$$\|F_{n,k}\| \leq C\varepsilon \max_{j \geq 0} (\|A\| + |\mu_j|) \|\underline{U}_{s+1}^{(j)}\|_F,$$

where C is a constant depending on n, s , the variant used and the matrix A and ε is the machine epsilon. Here $\mu_0 = 0$. This inequality shows that the coefficients μ_j must not be chosen with an absolute value much larger than $\|A\|$. If all these conditions are satisfied, the generalized Hessenberg relation is only slightly perturbed, but, unfortunately, this does not tell us if the convergence behavior of the method is much altered by rounding errors. Numerical experiments in [1022] show that the size of the perturbation term may have a strong influence on the maximum attainable accuracy for the residual norms.

10.10 Numerical examples

In this section we describe results of numerical experiments using different versions of the IDR algorithms.

10.10.1 IDR

Let us start with the primitive IDR algorithm of Section 10.1, the classical IDR algorithm of Section 10.2 which uses only one shadow vector and the IDR(s) algorithm in [865] described in Section 10.3 for different values of s .

For the problem `fs 813 6` the primitive IDR algorithm diverges immediately but this cannot be seen easily on the left-hand side of Figure 10.1. We have to use values of s larger than or equal to 2 to see some convergence of IDR(s). For $s = 4$

or 8 we have a faster initial convergence but after some 500 matrix–vector products the residual norms start to increase.

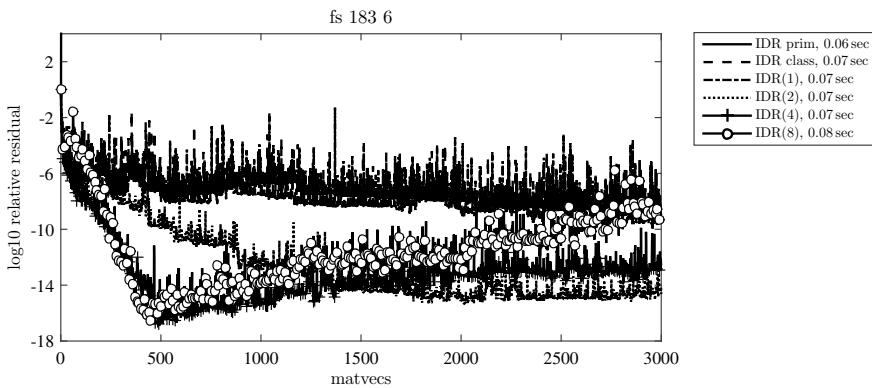


Fig. 10.1 fs 183 6, relative residual norms, different IDR algorithms

Figure 10.2 shows a comparison of IDR(s) and the biorthogonal algorithms IDR-Bio(s) in [948] for $s = 2, 4, 8$. The best result is given by IDR-Bio(8) for which the residual norms converge faster as functions of the number of matrix–vector products, even though the final accuracy is not the best one. We observe, as we have already seen above, that IDR(4) and IDR(8) seem less stable since the residual norms increase after reaching their lowest values. IDR-Bio(4) and IDR-Bio(8) are more stable since after reaching their minimum values the residual norms stagnate.

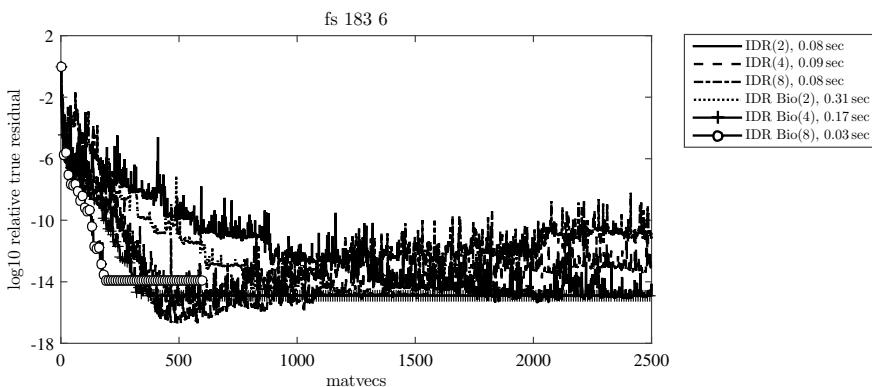


Fig. 10.2 fs 183 6, relative true residual norms, different IDR algorithms

Figure 10.3 displays the relative true residual norms of IDR-Bio(4) in double precision and in variable precision with 32 and 64 decimal digits. For this problem

IDR-Bio is also prone to the development of rounding errors which slow down convergence. Figure 10.4 shows the same information for $s = 8$.

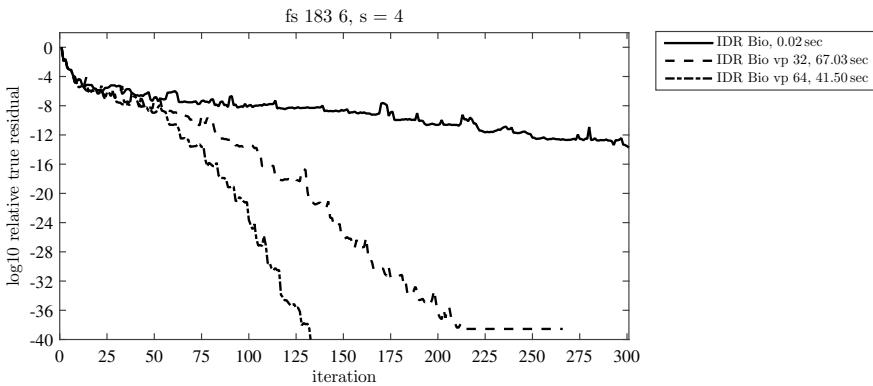


Fig. 10.3 $\text{fs } 183 \text{ 6}$, \log_{10} of the relative true residual norm, IDR-Bio(4) (plain), IDR-Bio(4) variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

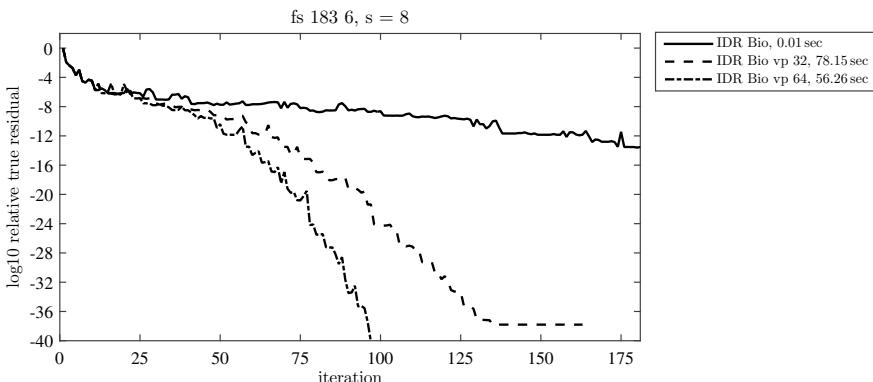


Fig. 10.4 $\text{fs } 183 \text{ 6}$, \log_{10} of the relative true residual norm, IDR-Bio(8) (plain), IDR-Bio(8) variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

For $\text{fs } 680 \text{ 1c}$ the primitive IDR algorithm converges but very slowly. The convergence of IDR(s) improves when increasing s ; see Figure 10.5.

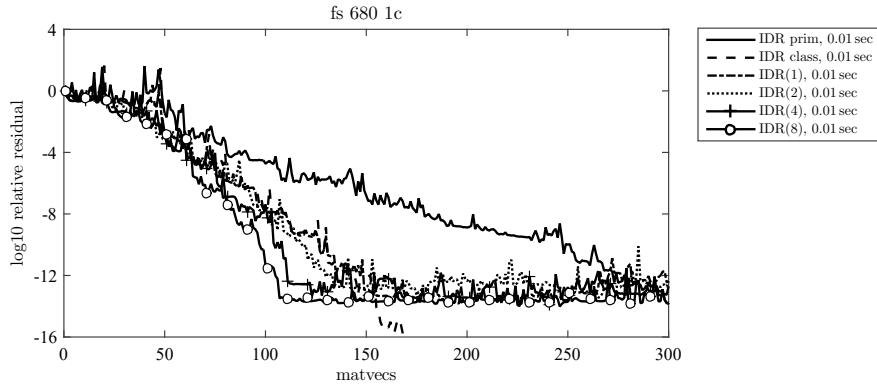


Fig. 10.5 *fs 680 1c*, relative residual norms, different IDR algorithms

Figure 10.6 shows that for this problem there are not much differences between IDR(s) and IDR-Bio(s). For IDR-Bio(s) the maximum attainable accuracy is getting worse when increasing s . For this example there is no increase of the residual norms of IDR(s) after reaching the minimum value.

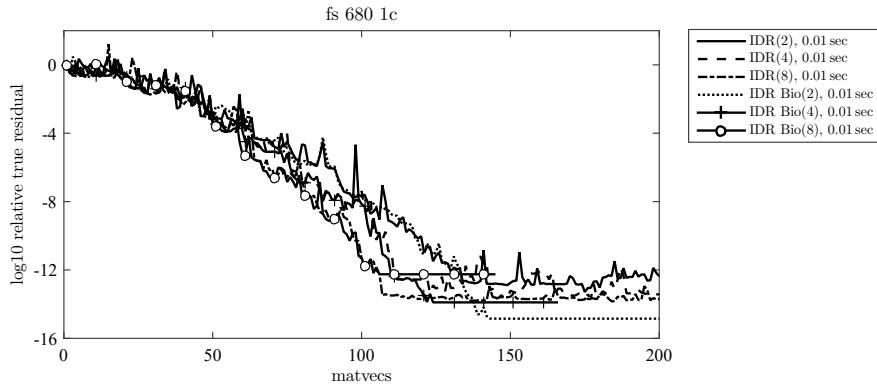


Fig. 10.6 *fs 680 1c*, relative true residual norms, different IDR algorithms

For *fs 680 1c* there is not much influence of the rounding errors on the convergence of IDR-Bio as shown in Figure 10.7. Only the maximum attainable accuracies are different in double precision and variable precision with 32 and 64 decimal digits.

Figure 10.8 displays the relative true residual norms as functions of the number of matrix–vector products for the problem *raefsky1*. The two variants IDR(s) and IDR-Bio(s) are roughly equivalent except for the maximum attainable accuracies. This is to be compared with Figure 10.9 where the relative residual norms are

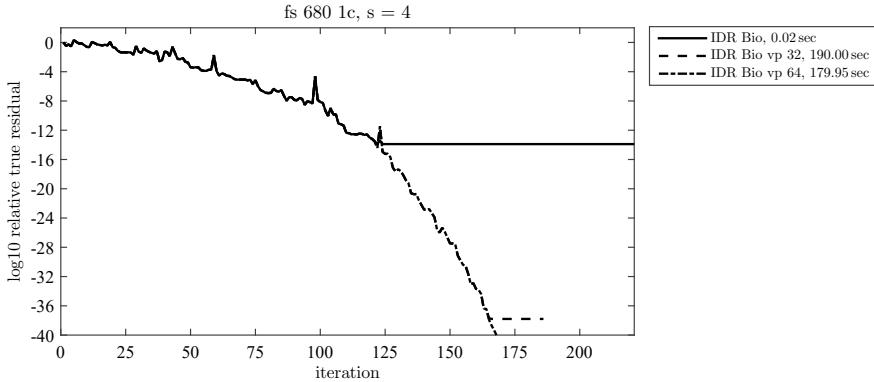


Fig. 10.7 *fs 680 1c*, \log_{10} of the relative true residual norm, IDR-Bio(4) (plain), IDR-Bio(4) variable precision with 32 decimal digits (dashed) and with 64 decimal digits (dot-dashed)

plotted against time. We observe that IDR algorithms are a little faster than IDR-Bio algorithms.

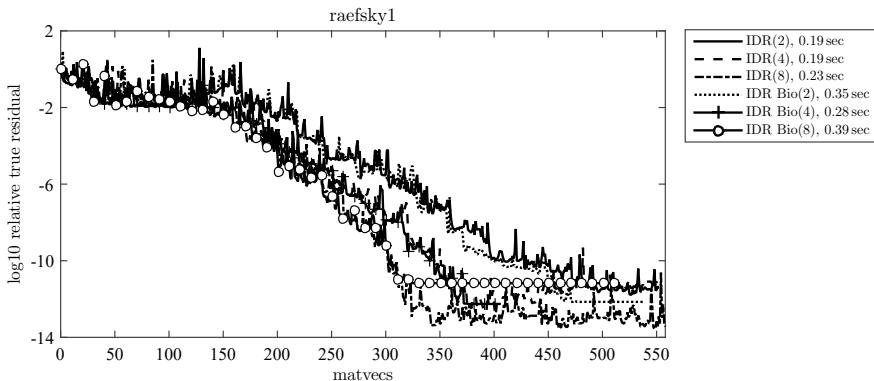


Fig. 10.8 *raefsky1*, relative true residual norms, different IDR algorithms

10.10.2 Variants of IDR and IDR-QMR

Figure 10.10 displays the relative true residual norms for *fs 680 1c* and the different variants of the Q-OR IDR method with partial orthogonalization for $s = 2, 4$; see [1022] and Section 10.4. The variant mnIDR gives the best results.

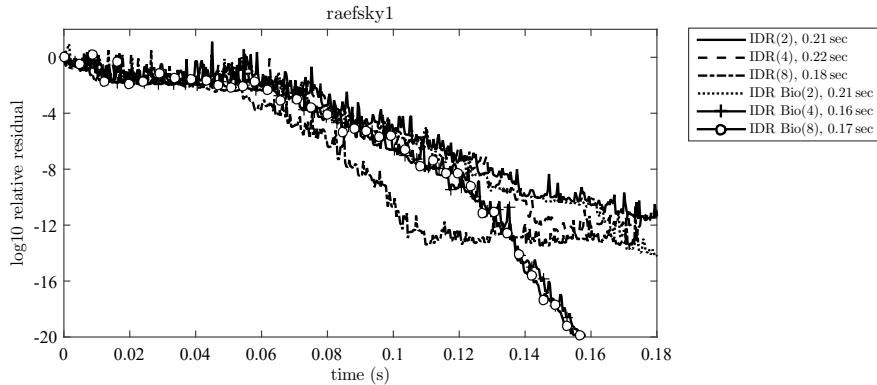


Fig. 10.9 raebsky1, relative residual norms, different IDR algorithms, residual norms vs time

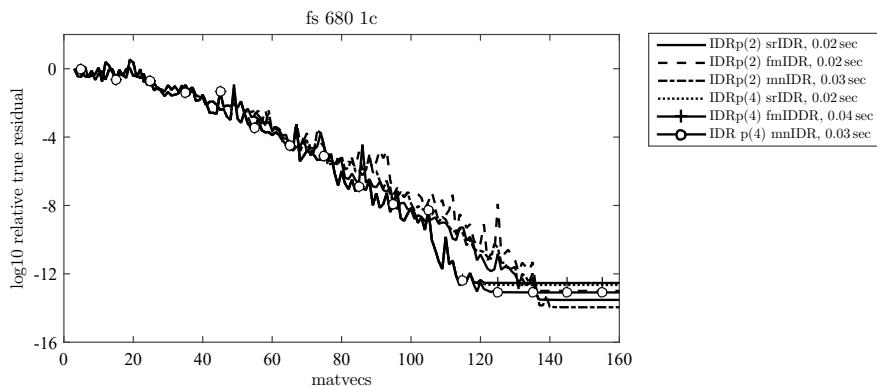


Fig. 10.10 fs 680 1c, relative true residual norms, different Q-OR IDR partial orthogonalization algorithms

Figure 10.11 shows the results for the Q-MR corresponding algorithms. Of course, the residual norms are smoother than with Q-OR but, again, mnIDR is the best variant.

Figure 10.12 displays the residual norms for more values of s . For this example there is no reason to take s larger than 4 except if we need to improve the maximum attainable accuracy.

The relative true residual norms for the Q-MR version of IDR with partial orthogonalization are plotted for raebsky1 in Figure 10.13. For this problem $s = 6$ converges faster than $s = 4$.

Figures 10.14 and 10.15 display the residual norms for the method Q-MR IDR from [946]. Once again it does not pay too much to take s larger than 4. In Figure 10.15

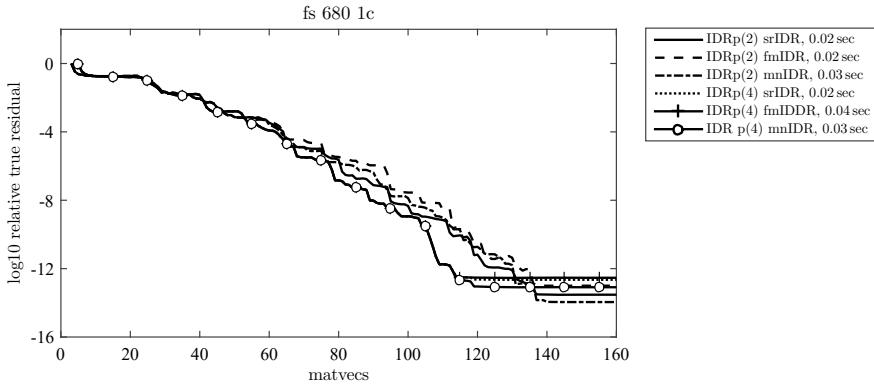


Fig. 10.11 fs 680 1c, relative true residual norms, different Q-MR IDR partial orthogonalization algorithms

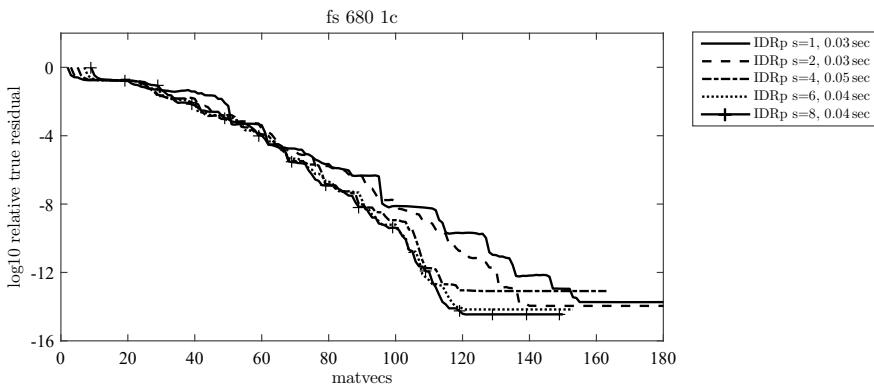


Fig. 10.12 fs 680 1c, relative true residual norms, Q-MR IDR partial orthogonalization mnIDR algorithms, different values of s

we observe that there is a large improvement when going from $s = 1$ to $s = 2$ for `raefsky1`.

The methods IDRStab from [850] have two parameters s and ℓ , see also Section 10.5. The parameter ℓ is the degree of the polynomial that is used at each iteration to smooth the residual norms. We show the results for $s = 2, 4$ and $\ell = 2, 4, 8$. We observe that increasing ℓ does not improve the convergence but the maximum attainable accuracies are getting much worse; see Figures 10.16 and 10.17. This is due to the fact that a monomial basis is used for the polynomial of degree ℓ . This can probably be cured by using a better basis.

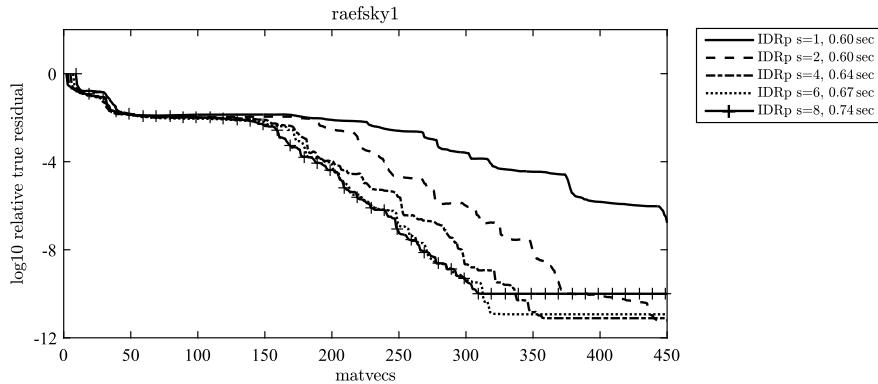


Fig. 10.13 raeFSKY1, relative true residual norms, Q-MR IDR partial orthogonalization mnIDR algorithms, different values of s

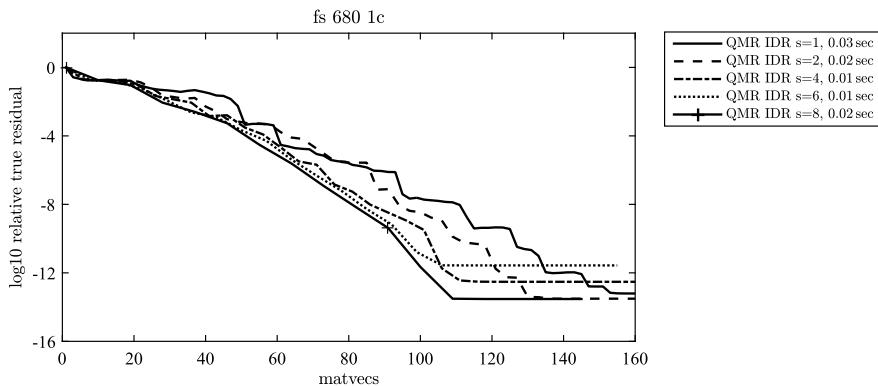


Fig. 10.14 fs 680 1c, relative true residual norms, Q-MR IDR algorithms from [946], different values of s

10.10.3 $ML(k)BiCGStab$

Let us compare IDR-Bio(s) and $ML(k)BiCGStab$ from [1008]. Figure 10.18 displays the residual norms for fs 680 1c and Figure 10.19 for raeFSKY1. For $k = s$ the convergence is more or less the same but the maximum attainable accuracies of $ML(k)BiCGStab$ are better than those of IDR-Bio(s). According to these results $ML(k)BiCGStab$ seems a little more stable than IDR-Bio(s) but this may be problem dependent.

Tables 10.1 and 10.2 show the minimal relative true residual norm obtained within 10000 iterations for BiCGStab (which is mathematically equivalent to IDR(1)), IDR(4) and IDR-Bio(4). Except in a few cases, the accuracies are roughly the same for IDR(4) and IDR-Bio(4).

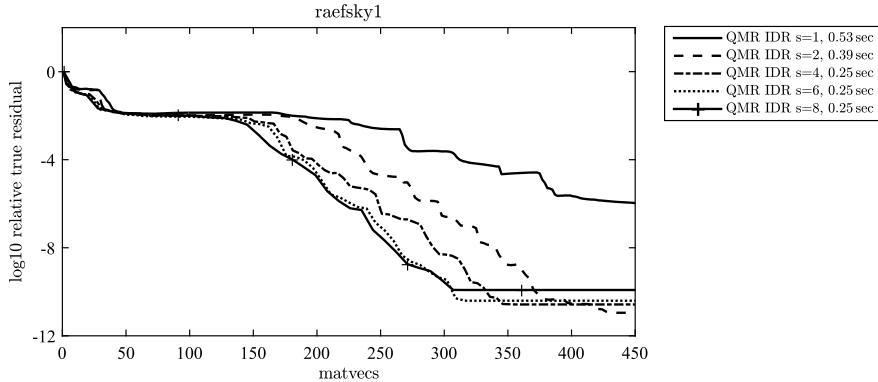


Fig. 10.15 *raefsky1*, relative true residual norms, Q-MR IDR algorithms from [946], different values of s

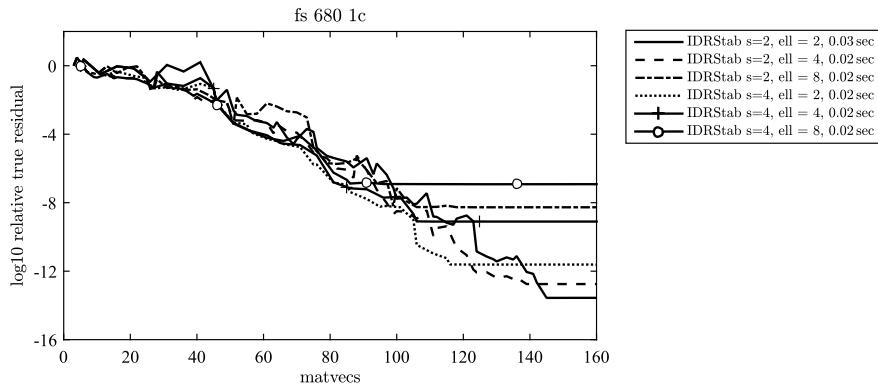


Fig. 10.16 *fs 680 1c*, relative true residual norms, IDRStab algorithms from [850], different values of s and ℓ

Tables 10.3 and 10.4 show the minimal relative true residual norm obtained within 10000 iterations for BiCGStab (which is mathematically equivalent to IDR(1)), IDR(8) and IDR-Bio(8). Except for a few cases and contrary to what could have been expected, the accuracies tend to be better for IDR(8) than for IDR-Bio(8).

Tables 10.5 and 10.6 show the number of matrix–vector products needed to obtain $\|r_k\| \leq 10^{-10}\|b\|$ with $s = 4$. A “-” means that the stopping criterion was not satisfied within 10000 iterations. There are many cases for which the numbers of matrix–vector products are not much different.

Table 10.1 Minimum relative true residual norms in 10000 iterations at most, $s = 4$

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
pde225	1.54582e-15	9.39214e-15	1.10405e-14
gre 343	1.16447e-02	1.11069e-14	2.16047e-14
jphw 991	1.00000e+00	1.64353e-14	2.34071e-14
pde2961	1.11697e-13	3.35713e-13	2.01685e-13
jagmesh1	6.62624e-16	2.21332e-15	1.79666e-14
bfsa782	1.76392e-13	3.88547e-13	4.81872e-13
dw2048	2.88259e-15	2.53764e-15	1.03398e-14
jagmesh2	1.29926e-15	1.00841e-15	3.41157e-15
raefsky2	1.15042e-14	2.95149e-13	2.93156e-13
fs 680 1c	1.68497e-14	1.86179e-13	1.26568e-14
add20	1.98130e-13	5.52613e-13	5.39448e-12
raefsky1	8.80449e-14	2.61245e-13	5.53122e-13
jagmesh4	1.15448e-15	3.42185e-13	5.21679e-15
fs 680 1	9.76008e-16	1.67419e-13	4.61699e-15
sherman1	6.32038e-14	4.62502e-13	1.05205e-10
nos3	1.84711e-14	4.55456e-13	2.42816e-12
sherman5	8.32279e-10	1.72304e-09	1.17878e-08
cavity05	1.75213e-13	2.91130e-12	6.00322e-13
e05r0500	1.00000e+00	4.45735e-01	4.66021e-01
comsol	6.25136e-11	1.43473e-10	1.53097e-10
olm1000	6.85106e-03	9.61273e-12	7.14739e-12
cavity10	7.79331e-14	1.17255e-11	2.91697e-12
steam2	1.65127e-13	2.25625e-12	5.08152e-12
1138bus	5.21455e-11	6.89288e-09	1.34964e-09
steam1	1.38707e-02	4.83712e-04	4.49944e-04
bcsstk26	1.67486e-07	1.58074e-06	9.60586e-07
nos7	1.00000e+00	8.59667e-01	8.55649e-01
watt1	3.91480e-13	2.27634e-03	5.58559e-09
bcsstk14	1.45650e-02	4.45735e-01	4.31361e-01
fs 183 6	5.27521e-16	3.23243e-12	1.11335e-15
bcsstk20	4.27294e-01	7.68011e-01	8.27012e-01
mcf6	1.00000e+00	9.07277e-01	8.97477e-01
nnc	1.00000e+00	8.36050e-01	8.37648e-01
Insp	9.99884e-01	1.00000e+00	1.00000e+00

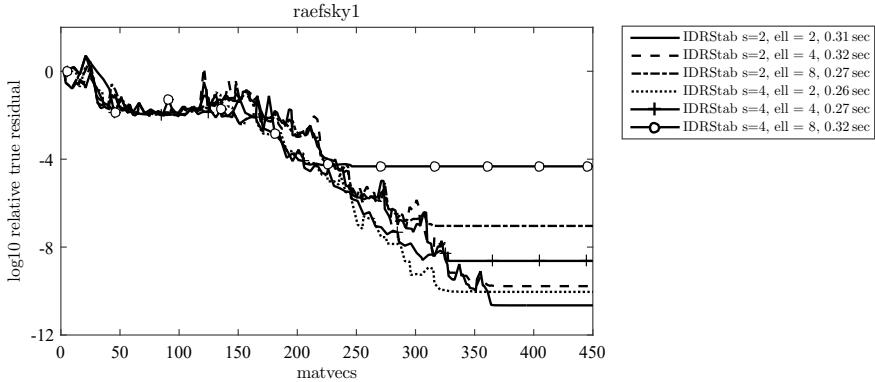


Fig. 10.17 *raefsky1*, relative true residual norms, IDRStab algorithms from [850], different values of s and ℓ

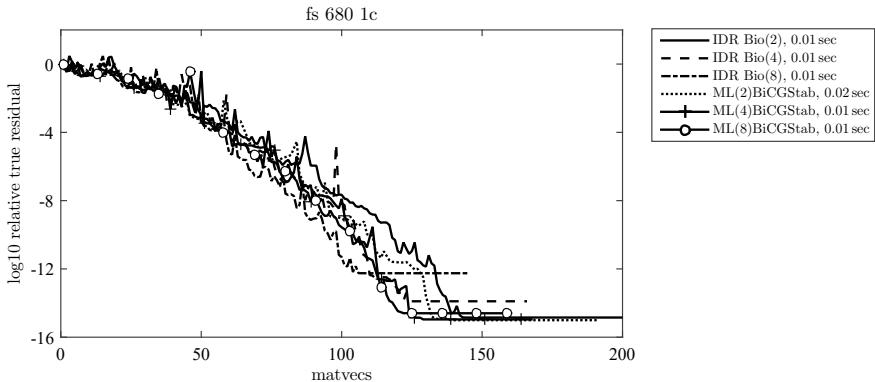


Fig. 10.18 *fs 680 1c*, relative true residual norms, IDR Bio(s) and ML(k)BiCGStab algorithms, different values of s and k

Tables 10.7 and 10.8 show the number of matrix–vector products needed to obtain $\|r_k\| \leq 10^{-10}\|b\|$ with $s = 8$. A “–” means that the stopping criterion was not satisfied within 10000 iterations. Again there are many cases for which the numbers of matrix–vector products are not much different. A noticeable exception is *sherman3* in Table 10.8.

10.11 Historical notes

Most of the information in this section was obtained from talks and papers by Jens-Peter M. Zemke.

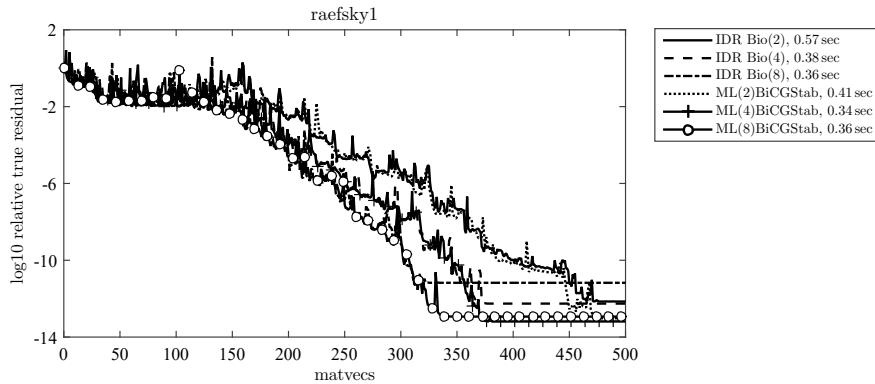


Fig. 10.19 raeftsky1, relative true residual norms, IDR Bio(s) and ML(k)BiCGStab algorithms, different values of s and k

Table 10.2 Minimum relative true residual norms in 10000 iterations at most, $s = 4$

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
add32	5.47665e-01	2.93352e-09	2.47773e-10
ex37	1.54270e-15	1.64226e-13	6.56371e-15
memplus	2.24759e-13	1.14566e-11	9.21834e-13
sherman3	5.73137e-11	3.08190e-07	9.06148e-09
wang4	1.06559e-14	4.62964e-12	2.64942e-13
supg 100 6400	1.36070e-14	4.92924e-13	7.55366e-13
supg 1 6400	1.60722e-14	1.07332e-13	1.97013e-12
supg 01 6400	4.27320e-12	7.25594e-13	2.08794e-11
supg 001 6400	8.80590e-11	5.46196e-13	2.55641e-14
supg 0001 6400	2.35791e-09	1.27532e-12	5.63736e-13
supg 00001 6400	3.19233e-09	9.25714e-14	1.26374e-13
supg 000001 6400	4.54458e-08	2.42034e-12	3.35614e-12
convdiff xu 500 8100	6.37240e-02	8.97465e-02	1.24026e-01
convdiff xu 1000 8100	1.00840e-01	8.09335e-06	1.58540e-04
convdiff xu 5000 8100	5.47665e-01	2.93352e-09	2.47773e-10

Table 10.3 Minimum relative true residual norms in 10000 iterations at most, $s = 8$

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
pde225	1.54582e-15	1.51856e-14	2.42613e-13
gre 343	1.16447e-02	5.86902e-15	6.92409e-15
jphw 991	1.00000e+00	2.75687e-14	1.98469e-14
pde2961	1.11697e-13	2.80554e-13	3.49278e-12
jagmesh1	6.62624e-16	3.40612e-15	3.59527e-13
bfsa782	1.76392e-13	2.22236e-13	7.82895e-12
dw2048	2.88259e-15	5.01680e-15	5.40573e-14
jagmesh2	1.29926e-15	1.79649e-15	9.43000e-15
raefsky2	1.15042e-14	4.12563e-14	9.85125e-12
fs 680 1c	1.68497e-14	4.70771e-14	5.57063e-13
add20	1.98130e-13	4.38848e-12	5.07218e-10
raefsky1	8.80449e-14	2.13427e-11	6.70246e-12
jagmesh4	1.15448e-15	2.47921e-15	3.25776e-15
fs 680 1	9.76008e-16	3.91592e-12	2.92481e-13
sherman1	6.32038e-14	1.42601e-11	8.68500e-11
nos3	1.84711e-14	9.19593e-13	4.62826e-12
sherman5	8.32279e-10	8.50828e-09	3.47305e-08
cavity05	1.75213e-13	7.00251e-12	4.00235e-06
e05r0500	1.00000e+00	5.12059e-01	3.52691e-01
comsol	6.25136e-11	2.43736e-09	2.52914e-08
olm1000	6.85106e-03	5.71423e-10	2.70919e-10
cavity10	7.79331e-14	2.33942e-12	1.06997e-10
steam2	1.65127e-13	1.63373e-11	4.11595e-11
1138bus	5.21455e-11	3.24817e-07	1.04824e-08
steam1	1.38707e-02	5.68639e-06	2.21448e-06
bcsstk26	1.67486e-07	2.83724e-06	8.21553e-07
nos7	1.00000e+00	9.72183e-01	1.00000e+00
watt1	3.91480e-13	2.12339e-01	3.87395e-02
bcsstk14	1.45650e-02	3.80861e-01	3.87607e-01
fs 183 6	5.27521e-16	3.22738e-05	1.23824e-14
bcsstk20	4.27294e-01	7.08795e-01	5.73424e-01
mcf6	1.00000e+00	1.00000e+00	1.00000e+00
nnc	1.00000e+00	9.20216e-01	8.85119e-01
Insp	9.99884e-01	1.00000e+00	1.00000e+00

Table 10.4 Minimum relative true residual norms in 10000 iterations at most, $s = 8$

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
add32	5.47665e-01	3.10488e-10	1.72508e-09
ex37	1.54270e-15	9.27747e-15	1.38246e-14
memplus	2.24759e-13	2.32332e-11	1.62842e-10
sherman3	5.73137e-11	8.56833e-07	2.11490e-08
wang4	1.06559e-14	1.66966e-12	1.38594e-11
supg 100 6400	1.36070e-14	8.20362e-13	1.23291e-12
supg 1 6400	1.60722e-14	1.42927e-11	5.70402e-11
supg 01 6400	4.27320e-12	1.48772e-13	1.00211e-12
supg 001 6400	8.80590e-11	2.11277e-14	5.35910e-13
supg 0001 6400	2.35791e-09	7.93687e-14	6.53904e-13
supg 00001 6400	3.19233e-09	5.38524e-12	1.46196e-11
supg 000001 6400	4.54458e-08	2.67095e-11	5.84549e-11
convdiff xu 500 8100	6.37240e-02	1.06093e-01	1.02312e-01
convdiff xu 1000 8100	1.00840e-01	1.47883e-05	2.55222e-04
convdiff xu 5000 8100	5.47665e-01	3.10488e-10	1.72508e-09

The development of IDR algorithms is largely due to Peter Sonneveld who was working at the Delft Technical University in The Netherlands. For the history of IDR methods, see Sonneveld's paper [864].

At the end of the 1970s for a course on numerical analysis Sonneveld was looking for a multidimensional generalization of the secant method. Of course, some methods were already existing but, nevertheless, he obtained what we described in Section 10.1 as the primitive IDR algorithm. The numerical properties of this simple method, namely, the fact that $r_{2n} = 0$ led him to prove what is now known as the IDR theorem.

In 1979 Sonneveld attended the IUTAM Symposium on Approximation Methods for Navier–Stokes Problems in Paderborn, Germany. There he presented the IDR algorithm we described in Section 10.2 based on multiplication with a factor $I - \omega_j A$ where ω_j is fixed for two steps and chosen to minimize every second residual norm. This algorithm was published in the proceedings from 1980 as a contribution with Peter Wesseling [981]. This paper did not receive much attention from the numerical linear algebra community.

The analysis of the IDR algorithm led Sonneveld to realize that this method could construct residual polynomials that are products of auxiliary polynomials with the Lanczos polynomials. This is what gives the (Bi)CGS algorithm that we described in Chapter 9. The paper about CGS was published in 1989 even though the method was already described in a 1984 report. In CGS the auxiliary polynomial is equal to the Lanczos polynomial. So, the factors $I - \omega_j A$ of the first IDR algorithm were abandoned. But, as we have seen in Section 9.8, they were coming back in the

Table 10.5 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, $s = 4$, nitmax = 10000

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
pde225	106	95	91
gre 343	–	700	756
jphw 991	–	80	81
pde2961	326	285	281
jagmesh1	840	325	321
bawa782	810	375	371
dw2048	4080	1595	1731
jagmesh2	3482	1300	1356
raefsky2	742	450	446
fs 680 1c	114	105	111
add20	1176	1035	791
raefsky1	476	355	346
jagmesh4	2382	1005	961
fs 680 1	516	315	301
sherman1	958	825	991
nos3	458	335	331
sherman5	–	2525	2361
cavity05	1642	1240	1076
e05r0500	–	–	–
comsol	648	550	391
olm1000	–	2795	2236
cavity10	2390	3190	3186
steam2	5604	635	791
1138bus	10642	8335	8726
steam1	–	–	–
bcsstk26	–	–	–
nos7	–	–	–
watt1	572	–	921
bcsstk14	–	–	–
fs 183 6	990	245	191
bcsstk20	–	–	–
mcf6	–	–	–
nnc	–	–	–
lnsp	–	–	–

discussions Sonneveld had with Henk van der Vorst which resulted in 1990 in a report [942] which essentially proposed BiCGStab.

The publication of CGS and BiCGStab started a stream of papers considering Lanczos-type product methods; see Section 9.8. Probably due to the successes of

Table 10.6 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, $s = 4$, nitmax = 10000

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
add32	116	110	101
ex37	124	115	106
memplus	3020	2235	2856
sherman3	17434	–	24191
wang4	784	630	661
supg 100 6400	856	890	1006
supg 1 6400	1794	1190	1161
supg 01 6400	646	400	406
supg 001 6400	222	140	141
supg 0001 6400 0	244	130	126
supg 00001 6400	350	240	236
supg 000001 6400	410	270	271
convdiff xu 500 8100	–	–	5386
convdiff xu 1000 8100	–	3335	2981
convdiff xu 5000 8100	–	750	686

these methods IDR was almost forgotten. As we already mentioned in Chapter 9, in Sonneveld’s words in [864] “*the BiCGSTAB algorithm had a significantly better numerical stability, for which reason the IDR-interpretation was buried! This, after all, appears to be wrong. It wasn’t the theoretical basis (IDR versus BiCG) that caused IDR’s instability, but a not so lucky implementation of the old IDR algorithm*”.

In 2006, J.-P. M. Zemke sent a message to Sonneveld asking what happened to IDR. To answer, Sonneveld revisited his ideas from the 1970s and realized that the IDR framework can be used with more than one shadow vector. The outcome of this revival was the first IDR(s) algorithm and the paper [865] with Martin Van Gijzen; see [862, 864].

However, the idea of using more shadow (or sometimes called left) vectors was not new. Block Krylov methods have been developed for quite some time mainly to solve problems with several right-hand sides. But in [16] published in 2000, José L. Aliaga, Daniel Boley, Roland W. Freund and Vicente Hernández proposed a nonsymmetric Lanczos-type method. This algorithm generates two sequences of biorthogonal basis vectors. It can handle the case of left and right starting blocks of different sizes.

Moreover, in 1999, Man-Chung Yeung and Tony F. Chan published a paper [1011] (received in 1997) describing a method called ML(k)BiCGStab. It can be seen as a transpose-free extension à la BiCGStab of a method ML(k)BiCG which is using k left vectors. The derivation of this method was quite complicated. This may be the reason why it did not receive much attention. This paper is cited in [865]. We

Table 10.7 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, $s = 8$, nitmax = 10000

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
pde225	106	90	91
gre 343	–	504	460
jphw 991	–	81	82
pde2961	326	270	271
jagmesh1	840	306	316
bfgwa782	810	324	316
dw2048	4080	1449	1432
jagmesh2	3482	1161	1171
raefsky2	742	414	406
fs 680 1c	114	99	100
add20	1176	585	532
raefsky1	476	306	307
jagmesh4	2382	873	820
fs 680 1	516	252	226
sherman1	958	684	676
nos3	458	315	316
sherman5	–	2169	2296
cavity05	1642	684	820
e05r0500	–	–	–
comsol	648	495	568
olm1000	–	1386	1792
cavity10	2390	1314	1153
steam2	5604	468	415
1138bus	10642	5832	4483
steam1	–	–	–
bcsstk26	–	–	–
nos7	–	–	–
watt1	572	–	–
bcsstk14	–	–	–
fs 183 6	990	243	145
bcsstk20	–	–	–
mcf6	–	–	–
nnc	–	–	–
lnsp	–	–	–

Table 10.8 Number of matrix–vector products to reach $\|r_k\| \leq 10^{-10} \|b\|$, $s = 8$, nitmax = 10000

matrix	BiCGStab	IDR(s)	IDR-Bio(s)
add32	116	108	100
ex37	124	117	109
memplus	3020	1773	1486
sherman3	17434	16965	7561
wang4	784	603	586
supg 100 6400	856	855	784
supg 1 6400	1794	1206	1072
supg 01 6400	646	387	370
supg 001 6400	222	135	136
supg 0001 6400	244	117	118
supg 00001 6400	350	234	226
supg 000001 6400	410	252	253
convdiff xu 500 8100	–	–	10081
convdiff xu 1000 8100	–	2250	2377
convdiff xu 5000 8100	–	675	604

note that $\text{ML}(k)\text{BiCGStab}$ was re-derived in a simpler way by M.C. Yeung [1008] in 2012; see also [1009].

Numerical experiments with the first IDR(s) algorithm, particularly on linear systems arising from the discretization of convection–diffusion equations, showed that this algorithm is not very stable. Increasing the value of s improves the convergence measured as a function of numbers of matrix–vector products but the maximum attainable accuracy is getting worse, that is, larger; see, for instance, figure 6.1 in [865] where the maximum attainable accuracy is larger by more than two orders of magnitude when we go from $s = 1$ to $s = 4$. This is probably due to the larger oscillations of the residual norms in the first part of the computation. See also the large oscillations of the residual norms in [947].

Contrary to the paper of 1980, the publication of IDR(s) in 2008 draws the attention of many researchers. Several papers were published by Japanese researchers that were already involved in the study of Lanczos-type product methods; see Yusuke Onoue, Seiji Fujino and Norimasa Nakashima [725–727]. In 2010 Martin H. Gutknecht published an expository paper whose title is “IDR explained” [484] in which he gave details on the IDR algorithms, related them to other Krylov methods and summarizing the early history of IDR methods. Valeria Simoncini and Daniel B. Szyld [839] in 2010 showed how one can interpret IDR(s) as a Petrov–Galerkin method. They also proposed a new variant of IDR using some Ritz values to compute the parameters ω_j . This is known as the Ritz-IDR variant.

In 2010 Gerard L.G. Sleijpen, P. Sonneveld and M.B. Van Gijzen [842] showed how BiCGStab can be seen as an induced dimension reduction method. This same year, a paper by G.L.G. Sleijpen and M.B. Van Gijzen [850] proposed the method

IDR(s, ℓ). It exploited the idea used in BiCGStab(ℓ) to apply a higher order stabilizing polynomial instead of the linear factors $I - \omega_j A$. A rather similar idea was used independently by Masaaki Tanio and Masaaki Sugihara in 2010 when they proposed the GBi-CGSTAB(s,L) method [905] which is IDR(s) with higher order stabilization polynomials; see also Kensuke Aihara, Kuniyoshi Abe and Emiko Ishiwata [11].

The “stabilized” variants of IDR were also considered in papers which appeared later by K. Aihara, K. Abe and E. Ishiwata [12–14] in 2014–2015.

To improve the stability of the first IDR algorithm, M.B. Van Gijzen and P. Sonneveld proposed in 2011 another variant of IDR using biorthogonality properties [948]. A Matlab code is still available on Van Gijzen’s website.

In 2011 Tijmen P. Collignon and M.B. Van Gijzen studied how to minimize the number of synchronization points in IDR(s) for its implementation on parallel computers [230]. They reformulated the bi-ortho variant to have only one global synchronization point per step. T.P. Collignon, G.L.G. Sleijpen and M.B. Van Gijzen showed how to interpret IDR(s) as a deflation process [231].

Lei Du, Tomohiro Sogabe and Shao-Liang Zhang considered how to apply residual smoothing techniques to IDR(s) in 2011 [284]. A block IDR(s) method was proposed by L. Du, T. Sogabe, Bo Yu, Yusaku Yamamoto and S.L. Zhang to solve linear systems with several right-hand sides [283].

In 2012 P. Sonneveld published a paper on the convergence behavior of IDR(s) [863] in which he used statistical arguments to show how close the IDR residual norms are to the GMRES residual norms. However, he made some quite unrealistic hypotheses like s being very large and $s > n$. As far as we know it is the only theoretical paper that was devoted to IDR convergence.

Since Krylov methods are also used to compute approximations to eigenvalues of A , it was tempting to do the same with IDR algorithms. This was started in 2013 by M.H. Gutknecht and J.-P. M. Zemke [494]. They considered the matrix pencils arising from the generalized Hessenberg relations satisfied by IDR algorithms. However, there are some difficulties due to the polynomial factors $1 - \omega_j \xi$. They used purification and deflation to construct smaller matrix pencils and to obtain only the Ritz values that are wanted. On this topic, see also [40, 41] by Reinaldo Astudillo and M.B. Van Gijzen in 2016.

In 2013 Olaf Rendel, Anisa Rizvanolli and J.-P.M. Zemke [767] summarized what was known about IDR algorithms by describing in detail the generalized Hessenberg relations corresponding to different variants of IDR. This allows to transfer techniques known for classical Krylov subspace methods to IDR-based methods, in particular, to develop eigenvalue or Q-MR algorithms.

In 2015 M.B. Van Gijzen, G.L.G. Sleijpen and J.-P. M. Zemke [946] proposed flexible and multi-shift Q-MR IDR algorithms, flexibility meaning that a different preconditioner can be used in each step of the algorithm.

Recent developments on IDR are given in a paper by J.-P. M. Zemke [1022] who proposed variants in which some blocks of “basis” vectors are orthogonalized (this is called *partial orthogonalization*), and in studies by Martin P. Neuenhofen [714] to obtain more reliable versions of “stabilized” IDR. So, the IDR family of methods is still an active area of research.

Chapter 11

Restart, deflation and truncation



For our purposes in this chapter the Krylov methods we have studied in chapters 5 to 10 can be seen to belong to one of two different classes. The methods of the first class use bases constructed with long recurrences, that is, to compute a new basis vector we need all the previous basis vectors. The most representative methods of this class are FOM and GMRES which use the Arnoldi process to compute the basis. The second class of methods use bases constructed with short recurrences. The most representative methods of the second class are BiCGStab and QMR. A result of Faber and Manteuffel [338] shows that, loosely speaking, it is not possible to construct an orthonormal basis of the Krylov subspace for a nonsymmetric matrix using short recurrences. For a detailed study of this problem, see chapter 4 of [640].

The advantage of a method like GMRES is that it minimizes the residual norm at each iteration. However, because of the long recurrence, the storage grows with the iteration number as well as the number of floating-point operations per iteration. When solving large problems, the storage capacity that is needed may become an issue and the standard GMRES method, which is sometimes referred as *full GMRES*, cannot be used. This problem was noticed very early (see [789, 800]) and remedies were proposed.

To avoid the storage problem, two techniques have been proposed, *restart* and *truncation*. In the restarted methods one runs the algorithm for a given number m of iterations and then the algorithm is restarted from the current approximate solution or using more sophisticated techniques that we will describe later in this chapter. Truncation amounts to truncating to a fixed number of terms the long recurrences used to compute the bases. Unfortunately, the restarted and truncated methods do not retain all the good mathematical properties of the original methods.

In this chapter we study some restarted and truncated methods. Even though any Q-OR or Q-MR method using long recurrences can be restarted or truncated, we mainly consider FOM and GMRES. For papers studying these methods, see [297, 485, 837].

11.1 Restarted FOM and GMRES

The standard way of restarting FOM and GMRES is to stop the iterations every m iterations, where m is an integer larger than or equal to 1 (we also assume that m is smaller than the grade of the restarting vectors with respect to A), to discard the previously computed basis vectors and to start again with the last computed approximation and residual vector as initial conditions. These methods are denoted as FOM(m) and GMRES(m), respectively. The set of iterations between two restarts is called a *cycle*. In this way, only at most m iterations of the Arnoldi process have to be carried out successively and the storage is under control, as well as the number of floating-point operations per cycle. It is only necessary to compute the approximation of the solution at the end of a cycle or when the stopping criterion is satisfied. A template omitting the code for an iteration (which is similar to what we described in Chapter 5) is following. The integer m has to be given as a parameter of the function.

```

ni = 0; % number of iterations
nc = 0; % number of cycles
nb = norm(b);
x = x0;
r = b - A * x;
bet = norm(r);
nresidu = realmax;
iconv = 0;
while nresidu > (epsi * nb) && (ni < nitmax)
    % -- Loop on cycles
    nc = nc + 1;
    V = zeros(n,m+1); % init basis vectors
    H = zeros(m+1,m);
    v = r / bet;
    V(:,1) = v;
    for k = 1:m % start a cycle of FOM(m) or GMRES(m)
        ni = ni + 1; % number of iterations
        %
        % FOM or GMRES code.....
        %
        nresidu = ...; % estimate of the residual norm
        % convergence test or too many iterations
        if nresidu < (epsi * nb) || ni >= nitmax
            iconv = 1;
            break % get out of the k loop
        end % if nresidu
    end % for k, end of one cycle
    % computation of the solution at the end of the cycle
    y = triu(H(1:k,1:k)) \ rhs(1:k);
    x = x0 + V(:,1:k) * y;

```

```

if iconv == 1 % we have to stop
    return
end % if iconv
% we have not converged yet, compute the residual and
% restart
r = b - A * x;
x0 = x;
bet = norm(r);
nresidu = bet;
rhs = zeros(m+1,1);
rhs(1) = bet;
end % while, loop on cycles

```

Mathematically, full FOM and GMRES obtain the solution of the linear system in at most n iterations. This is not true of the restarted versions. Restarting usually slows down convergence compared to the full algorithms. In fact, there are cases for which FOM(m) or GMRES(m) does not converge to the solution even though GMRES(m) produces non-increasing residual norms. It is possible that this method stagnates forever. In that case, GMRES (m) produces identical approximations during an entire cycle of m iterations, and consequently all subsequent cycles behave in the same way.

A common misconception about restarted methods is to believe that increasing the restart parameter m could always improve convergence. There exist examples where convergence is faster for a given value m than for other values $m' > m$, see [297, 318]. We will see some other examples in the numerical experiments section of this chapter. Of course, based on the information we could collect when iterating, we could possibly change the value of the restart parameter m from one cycle to the next. We will describe below some heuristics for doing this.

Let us denote by $v_i^{(j)}$ (resp. $r_i^{(j)}$) the Arnoldi basis vectors (resp. the residual vectors) of the j th cycle and we denote, as usual, the global basis and residual vectors by v_i and r_i . Let us consider the two first cycles. We have $v_1^{(1)} = r_0/\|r_0\|$ and we start the second cycle from $v_1^{(2)} = r_m^{(1)}/\|r_m^{(1)}\|$ with $r_m^{(1)} = r_m$. Then,

$$\text{span}\{v_1^{(1)}, \dots, v_m^{(1)}, v_1^{(2)}, \dots, v_m^{(2)}\} = \text{span}\{v_1^{(1)}, \dots, v_m^{(1)}, r_m, Ar_m, \dots, A^{m-1}r_m\}.$$

Therefore,

$$\text{span}\{v_1^{(1)}, \dots, v_m^{(1)}, v_1^{(2)}, \dots, v_m^{(2)}\} \subseteq \mathcal{K}_{2m}(A, r_0).$$

These two subspaces coincide if $[v_{m+1}^{(1)}]^* r_m \neq 0$ (see [828]) and the inclusion is strict otherwise. Now we have to distinguish between FOM(m) and GMRES(m). For FOM(m) we have $v_1^{(2)} = v_{m+1}^{(1)} = r_m/\|r_m\|$ and the two subspaces coincide. At the end of the second cycle we can write an Arnoldi-like relation

$$A[V_{n,m}^{(1)}, V_{n,m}^{(2)}] = [V_{n,m}^{(1)}, V_{n,m+1}^{(2)}] \hat{H}_{2m},$$

where \hat{H}_{2m} is an upper Hessenberg matrix of dimension $(2m + 1) \times 2m$ defined by

$$[\hat{H}_{2m}]_{1:m+1,1:m} = \underline{H}_m^{(1)}, \quad [\hat{H}_{2m}]_{m+1:2m+1,m+1:2m} = \underline{H}_m^{(2)},$$

see [828]. However, the basis is not globally orthonormal but only locally orthonormal and FOM (m) is not similar to using FOM with this global basis.

Things are different for GMRES(m) since $r_m = r_m^G$ depends on all the previous basis vectors. From Theorem 3.6 we have

$$v_1^{(2)} = \frac{r_m^G}{\|r_m^G\|} = \frac{\|r_m^G\|}{\|r_0\|} V_{n,m+1}^{(1)} \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,m+1} \end{pmatrix}, \quad (11.1)$$

where $|\vartheta_{1,j}|$ is proportional to the inverse of the corresponding FOM residual norm at iteration $j - 1$. If $\vartheta_{1,m+1} \neq 0$ we can write

$$v_{m+1}^{(1)} = -\frac{1}{\bar{\vartheta}_{1,m+1}} \sum_{j=1}^m \bar{\vartheta}_{1,j} v_j^{(1)} + \beta^{(2)} v_1^{(2)},$$

with $\beta^{(2)} = \|r_0\| / (\|r_m^G\| \bar{\vartheta}_{1,m+1})$. Using the Arnoldi relation for the first cycle, we obtain

$$\begin{aligned} AV_{n,m}^{(1)} &= V_{n,m}^{(1)} H_m^{(1)} + h_{m+1,m}^{(1)} v_{m+1}^{(1)} e_m^T, \\ &= V_{n,m}^{(1)} H_m^{(1)} + h_{m+1,m}^{(1)} \left[-\frac{1}{\bar{\vartheta}_{1,m+1}} \sum_{j=1}^m \bar{\vartheta}_{1,j} v_j^{(1)} + \beta^{(2)} v_1^{(2)} \right] e_m^T, \\ &= V_{n,m}^{(1)} \tilde{H}_m^{(1)} + h_{m+1,m}^{(1)} \beta^{(2)} v_1^{(2)} e_m^T, \end{aligned}$$

where the entries of $\tilde{H}_m^{(1)}$ are the same as those of $H_m^{(1)}$ except for the last column where $\tilde{h}_{j,m}^{(1)} = h_{j,m}^{(1)} - h_{m+1,m}^{(1)} \bar{\vartheta}_{1,j} / \bar{\vartheta}_{1,m+1}$ for $j = 1, \dots, m$. The Arnoldi relation can also be written as

$$AV_{n,m}^{(1)} = [V_{n,m}^{(1)}, v_1^{(2)}] \tilde{H}_m^{(1)},$$

with $\tilde{h}_{m+1,m}^{(1)} = h_{m+1,m}^{(1)} \beta^{(2)}$ and

$$A[V_{n,m}^{(1)}, V_{n,m}^{(2)}] = [V_{n,m}^{(1)}, V_{n,m+1}^{(2)}] \tilde{H}_{2m},$$

with

$$[\tilde{H}_{2m}]_{1:m+1,1:m} = \tilde{H}_m^{(1)}, \quad [\tilde{H}_{2m}]_{m+1:2m+1,m+1:2m} = H_m^{(2)}.$$

But, if we continue after the second cycle, the second matrix will be

$$[\tilde{H}_{2m}]_{m+1:2m+2,m+1:2m} = \tilde{H}_m^{(2)},$$

and so on.

Heuristically, we see that we can expect a slow convergence if the restart vector $r_0^{(2)}$ is not much different from $r_0^{(1)}$. This cannot happen for FOM(m) since the residual vectors are proportional to the basis vectors which are (mathematically) orthonormal. But, this is not the case for GMRES(m) if the moduli of the $\vartheta_{1,j}$'s for $j = 2, \dots, m+1$ are small.

Even though in restarted GMRES we throw away the previous basis vectors at the beginning of the second cycle, the second cycle depends strongly on the first cycle. The following theorem proves that stagnation for some of the last iterations of the first cycle implies stagnation for the same number of iterations at the beginning of the second cycle.

Theorem 11.1 *Let A be a nonderogatory matrix such that $A = VHV^*$ with $V^*V = I$ and H an unreduced upper Hessenberg matrix with triangular Hessenberg decomposition $H = UCU^{-1}$, see Theorem 2.3. Assume that for the first cycle of GMRES(m) we have*

$$\|r_{m-j-1}^{(1)}\| > \|r_{m-j}^{(1)}\| = \dots = \|r_m^{(1)}\|.$$

Then, $[r_m^{(1)}]^ A^i r_m^{(1)} = 0$, $i = 1, \dots, j$.*

Proof We have seen above that $r_m^{(1)}$ can be written as

$$r_m^{(1)} = c_1 V_{n,m+1}^{(1)} \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,m+1} \end{pmatrix},$$

where c_1 is a positive real number and the $\vartheta_{1,j}$'s are the entries of the first row of U^{-1} . From Chapter 5 we know that the stagnation assumption implies that $\vartheta_{1,\ell} = 0$ for $\ell = m+2-j, \dots, m+1$. We observe that the $m+1$ first columns of V are equal to those of $V_{n,m+1}^{(1)}$. Therefore,

$$r_m^{(1)} = c_1 V z^{(1)},$$

where $z^{(1)}$ is a vector whose only nonzero components are $z_i^{(1)} = \bar{\vartheta}_{1,\ell}$, $\ell = 1, \dots, m+1-j$. Let us denote $z^{(1)} = (\xi^{(1)} \ 0 \ \cdots \ 0)^T$ where $\xi^{(1)}$ is a vector of length $\tilde{m} = m+1-j$.

Let us consider $[r_m^{(1)}]^* A^i r_m^{(1)}$. From Chapters 2 and 3 we have $H = UCU^{-1}$ where C is a companion matrix, U is upper triangular and the $m+1$ first entries of the first row of U^{-1} are equal to $\vartheta_{1,\ell}$, $\ell = 1, \dots, m+1$. Obviously, we have $A^i = VUC^iU^{-1}V^*$ and

$$\begin{aligned} [r_m^{(1)}]^* A^i r_m^{(1)} &= c_1^2 [z^{(1)}]^* V^* V U C^i U^{-1} V^* V z^{(1)}, \\ &= c_1^2 [z^{(1)}]^* U C^i U^{-1} z^{(1)}. \end{aligned}$$

We have to consider $[z^{(1)}]^*U$ and $U^{-1}z^{(1)}$. Since only the first \tilde{m} components of $z^{(1)}$ are nonzero, the first one is given by

$$[z^{(1)}]^*U = ([\xi^{(1)}]^*U_{\tilde{m}}, y^*),$$

where y is a vector of length $n - \tilde{m}$. But,

$$[\xi^{(1)}]^*U_{\tilde{m}} = e_1^T U_{\tilde{m}}^{-1} U_{\tilde{m}} = e_1^T.$$

The first \tilde{m} components of $[z^{(1)}]^*U$ are equal to e_1^T . However, the j first components of y are zero. Let us see this for the first component for which we have to consider column $\tilde{m} + 1$ of U . By considering U as the inverse of U^{-1} , we can write U for rows 1 to \tilde{m} and columns 1 to $\tilde{m} + 1$ as

$$(U_{\tilde{m}}, -u_{\tilde{m}+1,\tilde{m}+1}U_{\tilde{m}}(U^{-1})_{1:\tilde{m},\tilde{m}+1}).$$

Multiplying from the left by $[z^{(1)}]^*$ the last column we obtain

$$-u_{\tilde{m}+1,\tilde{m}+1}[\xi^{(1)}]^*U_{\tilde{m}}(U^{-1})_{1:\tilde{m},\tilde{m}+1} = -u_{\tilde{m}+1,\tilde{m}+1}e_1^T(U^{-1})_{1:\tilde{m},\tilde{m}+1}.$$

But, the first component of $(U^{-1})_{1:\tilde{m},\tilde{m}+1}$ is equal to $\vartheta_{1,\tilde{m}+1} = 0$ since we have $\vartheta_{1,\ell} = 0$, $\ell = \tilde{m} + 1, \dots, m + 1$. Hence, $y_1 = 0$. By induction and using the same reasoning, we can prove that $y_\ell = 0$, $\ell = 1, \dots, j$.

For the other vector, we have

$$U^{-1}z^{(1)} = \begin{pmatrix} U_{\tilde{m}}^{-1}\xi^{(1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

only the first \tilde{m} components are nonzero.

Because of the nonzero structure of the companion matrix C , the powers C^i have zero entries in rows 1 to i and columns 1 to $n - i$ since the last $n - i$ columns are nonzero. Let us assume that $n \gg m$. The matrices C^i act as down-shift matrices. Hence $CU^{-1}z^{(1)}$ has a zero first component, then \tilde{m} nonzero components and all the other components are zero. Multiplying successively several times by C pushes down the nonzero components. Therefore $C^iU^{-1}z^{(1)}$ has i zeros at the top.

Comparing the nonzero structures of $[z^{(1)}]^*U$ and $C^iU^{-1}z^{(1)}$ shows that the corresponding dot product is zero for all i such that $i \leq j$. \square

Corollary 11.1 *Using the notation of Theorem 11.1 and assuming that*

$$\|r_{m-j-1}^{(1)}\| > \|r_{m-j}^{(1)}\| = \cdots = \|r_m^{(1)}\|,$$

we have

$$\|r_0^{(2)}\| = \|r_1^{(2)}\| = \cdots = \|r_j^{(2)}\|,$$

which means that we have stagnation for the first j iterations of the second cycle.

Proof The result of the corollary is obtained from Theorem 11.1 using the results of [1017] on partial initial stagnation. \square

The results of Theorem 11.1 and Corollary 11.1 hold for any two successive cycles of GMRES(m) and stagnation at the end of a cycle implies stagnation at the beginning of the next cycle. The corresponding FOM(m) iterates are not defined, the upper Hessenberg matrices involved being singular. These results show that it is a bad idea to restart with the current iterate in case of stagnation of GMRES(m). In that case it may be better to restart from the last FOM(m) approximate solution (if it is available). Other possibilities are to go back to an iteration before stagnation started or to increase m hoping that, at some near iteration, we can get out of stagnation. Of course, in practical problems, we almost never encounter exact stagnation. But, there are many problems which exhibit quasi-stagnation and we can expect restarting in this phase produces quasi-stagnation at the beginning of the next cycle.

This result shows that, even though we throw away the basis vectors computed in the previous cycle, there is a lot of information in the residual vector which is used to build the new basis. In case of stagnation this can be considered as unfortunate.

The same kind of results holds for any Q-MR method but with the stagnation of the quasi-residual norms.

As we said above and as it was suggested in [828], when there is no stagnation at the end of a cycle, it may be beneficial to restart from the FOM residual rather than from the GMRES residual because the starting vector is then different from the starting vector of the previous cycle. Then, we can monitor the decrease of the GMRES residual norm and compute the GMRES solution when needed.

There are not many theoretical results about GMRES(m) convergence in the literature. Most of the papers on GMRES(m) are concerned with finding good ways to restart the algorithm or with strategies for varying m . Exceptions are [297, 402, 582, 828]. Sufficient conditions for convergence are given in [1035–1039]. But the conditions in these papers are not easy to check a priori. However, from Section 5.2 we know that if the matrix $M = (A + A^*)/2$ is positive definite, the residual norms are strictly decreasing within each cycle. Therefore, the GMRES(m) residual norms are strictly decreasing, but this does not imply that the convergence is fast. At the end of the first cycle, we have

$$\|r_m^{(1)}\| \leq \|r_0\| \left(1 - \frac{\lambda_{\min}(M)^2}{\lambda_{\max}(A^*A)}\right)^{m/2}.$$

We can iterate that bound and, after n_c cycles, we obtain

$$\|r_m^{(n_c)}\| \leq \|r_0\| \left(1 - \frac{\lambda_{\min}(M)^2}{\lambda_{\max}(A^*A)}\right)^{(n_c m)/2}.$$

Generalizations of these bounds are given in [1038].

All the bounds for the residual norms described in Section 5.2 can be iterated in the same way. For instance, from Theorem 5.2 we have, for a diagonalizable matrix A and after n_c cycles,

$$\|r_m^{(n_c)}\| \leq \|r_0\| \left(\|X\| \|X^{-1}\| \min_{\substack{p \in \pi_m \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)| \right)^{n_c}.$$

Of course, this bound is not of any help if the term within parenthesis is larger than 1. If the matrix A is normal, we can replace $\|X\| \|X^{-1}\|$ by 1.

For a general matrix A , it is shown in [828, Corollary 6.3] that

$$\|r_m^{(2)}\| \leq \kappa([V_{n,m+1}^{(1)}, v_2^{(2)}, \dots, v_{m+1}^{(2)}]) \|r_{2m}^G\|, \quad (11.2)$$

where r_{2m}^G is the residual vector after $2m$ iterations of full GMRES. If the condition number in this inequality is close to 1, after two cycles the GMRES(m) residual norm is not too far from what is obtained with full GMRES.

From Section 5.5 we have exact expressions for the residual norms as functions of the eigenvalues, the eigenvectors and the initial residual vector. These expressions are complicated for diagonalizable matrices but we have the bound $\|r_k^{(i)}\|^2 \leq \mu_k^{(i)} \|X\|^2$ with $k = 1, \dots, m$ and

$$\mu_k^{(i)} = \frac{\sum_{1 \leq i_1 < \dots < i_{k+1} \leq n} \left[\prod_{j=1}^{k+1} |c_{i_j}|^2 \right] \prod_{i_1 \leq i_\ell < i_p \leq i_{k+1}} |\lambda_{i_p} - \lambda_{i_\ell}|^2}{\sum_{1 \leq i_1 < \dots < i_k \leq n} \left[\prod_{j=1}^k |c_{i_j}|^2 |\lambda_{i_j}|^2 \right] \prod_{i_1 \leq i_\ell < i_p \leq i_k} |\lambda_{i_p} - \lambda_{i_\ell}|^2},$$

where $c = X^{-1}r_m^{(i-1)}$ for $i = 1, \dots$. When A is normal, we have an equality and $c = X^*r_m^{(i-1)}$. We observe that, when we restart from the current approximate solution, the only thing which changes in the upper bound from one cycle to the next is the vector c . Hence, we have a fast convergence only if one cycle decreases the components of $|c|$ efficiently.

The GMRES residual polynomials were studied in Section 5.6, Theorems 5.23 and 5.24. Again, the exact expressions of these polynomials are quite complicated. For a normal matrix A and $i = 1, 2, \dots$ we have

$$r_k^{(i)} = p_k^{(i)}(A)r_m^{(i-1)} \text{ with } p_k^{(i)}(\zeta) = \frac{q_k^{(i)}(\zeta)}{q_k^{(i)}(0)},$$

and $q_k^{(i)}(\zeta)$ is given by

$$\sum_{\mathcal{I}_k} (|\lambda_{i_1}|^2 - \zeta \overline{\lambda_{i_1}}) \cdots (|\lambda_{i_k}|^2 - \zeta \overline{\lambda_{i_k}}) |c_{i_1}^{(i)}|^2 \cdots |c_{i_k}^{(i)}|^2 \prod_{\substack{i_1 \leq i_p < i_q \leq i_k \\ i_p, i_q \in \mathcal{I}_k}} |\lambda_{i_q} - \lambda_{i_p}|^2,$$

where the summation is over all sets \mathcal{I}_k of k indices (i_1, i_2, \dots, i_k) such that $1 \leq i_1 < \dots < i_k \leq n$ and $c^{(i)} = X^* r_m^{(i-1)}$. We can write the global residual polynomial as

$$r_k^{(i)} = p_k^{(i)}(A) p_m^{(i-1)}(A) \cdots p_m^{(1)}(A) r_0.$$

The vector $r_k^{(i)}$ is given by a polynomial of degree $(i-1)m + k$ applied to the initial residual vector r_0 . However, the residual norm is not minimized over the global Krylov subspace defined by the polynomial.

If the normal matrix A has the spectral factorization $A = X\Lambda X^*$, we have $p_k^{(i)}(A) = X p_k^{(i)}(\Lambda) X^*$ and the projections of the starting residual vectors on the eigenvectors are given by

$$c^{(i+1)} = X^* r_m^{(i)} = p_m^{(i)}(\Lambda) c^{(i)}.$$

Hence, $c_\ell^{(i+1)} = p_m^{(i)}(\lambda_\ell) c_\ell^{(i)}$. However, as we have seen, the coefficients of the polynomial $p_m^{(i)}$ depend on all the components of $c^{(i)}$ in a nonlinear way and it is not easy to analyze this relation. The roots of the polynomial $p_m^{(i)}$ (which is equal to 1 at the origin) are the harmonic Ritz values that can be computed during cycle i and the values of the polynomial at one eigenvalue of A depend on how close the harmonic Ritz values are to this eigenvalue. The moduli of the components of c are decreased if the value of the polynomial is small enough. Even though this depends on the starting residual vector, it is likely that for small values of m the harmonic Ritz values are not very good approximations to the eigenvalues of A and so the values of the polynomial at the eigenvalues may not be small. We have similar relations when A is only diagonalizable or even non-diagonalizable, the polynomial coefficients being more complicated but they still depend on the $|c_i|$'s.

11.2 Prescribed convergence

In recent years some results appeared on constructing linear systems that generate prescribed residual norms when restarted FOM or GMRES is applied. In [819] it was proved that any residual norms can be generated in the restarted FOM method. It was shown in [952] that any history of decreasing residual norms at the last iteration of every cycle of GMRES(m) is possible, provided that at the end of every cycle, there is a strict decrease of the residual norm for the m th iterate in comparison with the previous iteration of that cycle. The residual norm history at the last iteration of every cycle is sometimes referred to as cycle convergence. A possible stagnation at the end of one cycle implies stagnation at the beginning of the next cycle, as we have seen in the previous section. Thus in that case, the admissible decreasing residual norm history inside the next cycle is not arbitrary anymore.

We will focus in the following on prescribing the GMRES(m) residual norms *inside* cycles instead of cycle convergence and assume that there is no stagnation at the end of any cycle. In fact, we will construct linear systems that generate in every cycle fully prescribed Hessenberg matrices (all their entries are prescribed values). This will allow to fix residual norms inside cycles and as a by-product, we can prescribe the Ritz values or the harmonic Ritz values generated in the restart cycles as well.

As before, we consider the construction of matrices and right-hand sides of the form $A = VHV^*$, $b/\|b\| = Ve_1$ where V is any unitary matrix. If $\underline{H}_m^{(1)}$ is a *given* Hessenberg matrix of size $(m+1) \times m$, then trivially this Hessenberg matrix is generated in the first m iterations of the GMRES method if and only if the leading upper left $(m+1) \times m$ block of H coincides with $\underline{H}_m^{(1)}$. If $\underline{H}_m^{(2)}$ is another *given* size $(m+1) \times m$ Hessenberg matrix, we have the following simple lemma.

Lemma 11.1 *Let GMRES(m) be applied to A and b of the form $A = VHV^*$, $b/\|b\| = Ve_1$ with V unitary and let it have generated after m iterations the Arnoldi relation*

$$AV_{n,m}^{(1)} = V_{n,m+1}^{(1)} \underline{H}_m^{(1)}. \quad (11.3)$$

Then the initial Arnoldi basis vector at the beginning of the first restart cycle is $v_1^{(2)} = V_{n,m+1}^{(1)} d^{(1)}$ with

$$d^{(1)} = \frac{\|r_m^{(1)}\|}{\|r_0\|} \begin{pmatrix} \bar{\vartheta}_{1,1} \\ \vdots \\ \bar{\vartheta}_{1,m+1} \end{pmatrix}, \quad (11.4)$$

where

$$\vartheta_{1,1} = 1, \quad \vartheta_{1,j} = \sqrt{\left(\frac{\|r_0\|}{\|r_j^{(1)}\|}\right)^2 - \left(\frac{\|r_0\|}{\|r_{j-1}^{(1)}\|}\right)^2}.$$

Moreover, for a given $(m+1) \times m$ Hessenberg matrix $\underline{H}_m^{(2)}$, the restarted GMRES method has generated, at the end of the second cycle, an Arnoldi relation of the form

$$AV_{n,m}^{(2)} = V_{n,m+1}^{(2)} \underline{H}_m^{(2)}, \quad (11.5)$$

where the matrix $V_{n,m+1}^{(2)}$ has orthogonal columns and $V_{n,m+1}^{(2)} e_1 = V_{n,m+1}^{(1)} d^{(1)}$ if and only if m iterations of the Arnoldi process with input matrix H and initial vector $[(d^{(1)})^T \ 0]^T$ generates the decomposition

$$HZ_{n,m}^{(2)} = Z_{n,m+1}^{(2)} \underline{H}_m^{(2)}, \quad (11.6)$$

where the matrix $Z_{n,m+1}^{(2)} = V^ V_{n,m+1}^{(2)}$ has orthogonal columns.*

Proof The first claim follows from (11.1). For the second claim it suffices to use $A = VHV^*$ and to define $Z_{n,m+1}^{(2)} \equiv V^*V_{n,m+1}^{(2)}$. \square

Our goal is to find entries of H which ensure that (11.6) holds for a given matrix $\underline{H}_m^{(2)}$. However, the involved matrix $Z_{m+1}^{(2)}$ with orthonormal columns is not fixed. The probably easiest way to satisfy (11.6) with $\underline{H}_m^{(2)}$ given is to assume that the matrix $Z_{n,m+1}^{(2)}$ with orthonormal columns has the simple form

$$Z_{n,m+1}^{(2)} \equiv \begin{pmatrix} d^{(1)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix}. \quad (11.7)$$

We can find the columns $m + 1$ until $2m$ of the matrix H satisfying (11.6) for this specific choice directly. Equating the first column of

$$HZ_{n,m}^{(2)} = Z_{n,m+1}^{(2)}\underline{H}_m^{(2)} \quad (11.8)$$

gives, with the notation $(d^{(1)})^T = [\hat{d}^{(1)}, d_{m+1}^{(1)}]^T$,

$$HZ_{n,m}^{(2)}e_1 = \begin{pmatrix} \underline{H}_m^{(1)} \\ 0 \end{pmatrix} \hat{d}^{(1)} + d_{m+1}^{(1)}He_{m+1} = h_{1,1}^{(2)} \begin{pmatrix} d^{(1)} \\ 0 \\ \vdots \end{pmatrix} + h_{2,1}^{(2)}e_{m+2},$$

where $h_{i,j}^{(2)}$ are the prescribed entries of $\underline{H}_m^{(2)}$. Thus the nonzero entries of the $(m + 1)$ st column of H satisfy

$$\begin{pmatrix} h_{1,m+1} \\ \vdots \\ h_{m+1,m+1} \end{pmatrix} = \frac{1}{d_{m+1}^{(1)}} \left(h_{1,1}^{(2)}d^{(1)} - \underline{H}_m^{(1)}\hat{d}^{(1)} \right), \quad h_{m+2,m+1} = \frac{h_{2,1}^{(2)}}{d_{m+1}^{(1)}}. \quad (11.9)$$

For the columns $m + 2$ until $2m$ of H we obtain directly from

$$HZ_{n,m}^{(2)}(e_2, \dots, e_m) = H_{:,m+2:2m} = Z_{n,m+1}^{(2)}\underline{H}_m^{(2)}(e_2, \dots, e_m)$$

that

$$\begin{aligned}
H_{:,m+2:2m} &= \begin{pmatrix} d^{(1)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \underline{H}_m^{(2)} & \begin{pmatrix} 0 \\ I_{m-1} \\ \vdots \end{pmatrix} \\ 0 & \end{pmatrix} \\
&= \begin{pmatrix} d^{(1)} e_1^T \underline{H}_m^{(2)} & \begin{pmatrix} 0 \\ I_{m-1} \\ \vdots \end{pmatrix} \\ (0 \ I_m) \underline{H}_m^{(2)} & \begin{pmatrix} 0 \\ I_{m-1} \\ \vdots \end{pmatrix} \\ 0 & \end{pmatrix}. \tag{11.10}
\end{aligned}$$

The obtained entries (11.9) and (11.10) in the columns $m + 1$ till $2m$ of H can be summarized in the following theorem. In the statement we use that a nonsingular $m \times m$ leading principal submatrix of the generated Hessenberg matrix guarantees that there is no stagnation at the end of the corresponding GMRES(m) cycle (see, e.g., [152]).

Theorem 11.2 *Given two $(m + 1) \times m$ unreduced Hessenberg matrices $\underline{H}_m^{(1)}$ and $\underline{H}_m^{(2)}$ where $\underline{H}_m^{(1)}$ has nonsingular $m \times m$ leading principal submatrix, these two Hessenberg matrices are consecutively generated in the first two cycles of the GMRES(m) method applied to H and e_1 if the first $2m$ columns of H are of the form*

$$H_{:,1:2m} = \begin{pmatrix} \underline{H}_m^{(1)} & 0 \\ 0 & \underline{H}_m^{(2)} \\ \vdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & \underline{H}_0^{(2)} \\ \vdots & \vdots \\ \vdots & 0 \end{pmatrix},$$

where the $(m + 2) \times m$ matrix $\underline{H}_0^{(2)}$ is the rank-two matrix

$$\underline{H}_0^{(2)} = \begin{pmatrix} \hat{d}^{(1)} \\ d_{m+1}^{(1)} - 1 \\ 0 \end{pmatrix} e_1^T \underline{H}_m^{(2)} - \frac{1}{d_{m+1}^{(1)}} \left(\begin{pmatrix} (d_{m+1}^{(1)} - 1) h_{1,1}^{(2)} d^{(1)} \\ -(d_{m+1}^{(1)} - 1) h_{2,1}^{(2)} \end{pmatrix} + \begin{pmatrix} \underline{H}_m^{(1)} \hat{d}^{(1)} \\ 0 \end{pmatrix} \right) e_1^T.$$

Proof We would like to write the entries in columns $m + 1$ till $2m$ found in (11.9) and (11.10) in the form

$$H_{:,m+1:2m} = \begin{pmatrix} 0 \\ \underline{H}_m^{(2)} \\ 0 \end{pmatrix} + \begin{pmatrix} \underline{H}_0^{(2)} \\ \vdots \\ 0 \end{pmatrix}.$$

As can be seen from (11.10), in H the rows $m + 2$ till $2m + 1$ of columns $m + 2$ till $2m$ are just the trailing $m \times (m - 1)$ block of $\underline{H}_m^{(2)}$. As for the first $m + 1$ rows of

columns $m + 2$ till $2m$, they have the form

$$d^{(1)} \cdot e_1^T \underline{H}_m^{(2)} \begin{pmatrix} 0 \\ I_{m-1} \end{pmatrix} = \left(e_1^T \underline{H}_m^{(2)} \begin{pmatrix} 0 \\ I_{m-1} \end{pmatrix} \right) + \begin{pmatrix} \hat{d}^{(1)} \\ \delta^{(1)} \end{pmatrix} e_1^T \underline{H}_m^{(2)} \begin{pmatrix} 0 \\ I_{m-1} \end{pmatrix},$$

with $\delta^{(1)} = d_{m+1}^{(1)} - 1$. The $m + 1$ st column of H , according to (11.9), has the leading $m + 1$ entries

$$\begin{aligned} \begin{pmatrix} h_{1,m+1} \\ \vdots \\ h_{m+1,m+1} \end{pmatrix} &= \frac{1}{d_{m+1}^{(1)}} \left(h_{1,1}^{(2)} d^{(1)} - \underline{H}_m^{(1)} \hat{d}^{(1)} \right) \\ &= \begin{pmatrix} 0 \\ \vdots \\ 0 \\ h_{1,1}^{(2)} \end{pmatrix} + h_{1,1}^{(2)} \begin{pmatrix} \hat{d}^{(1)} \\ \delta^{(1)} \end{pmatrix} \\ &\quad - \frac{1}{d_{m+1}^{(1)}} \left(h_{1,1}^{(2)} \delta^{(1)} \begin{pmatrix} d_1^{(1)} \\ \vdots \\ d_{m+1}^{(1)} \end{pmatrix} + \underline{H}_m^{(1)} \tilde{d}^{(1)} \right). \end{aligned}$$

The last row of $\underline{H}_0^{(2)}$ follows from the fact that $h_{m+2,m+1} = \frac{h_{2,1}^{(2)}}{d_{m+1}^{(1)}}$, see (11.9). \square

In the previous theorem, the involved entries of $d^{(1)}$ are determined by the residual norms generated in the first cycle via (11.4). Please note that the displayed matrix

$$\begin{pmatrix} \underline{H}_m^{(1)} & 0 \\ 0 & \underline{H}_m^{(2)} \\ \vdots & \vdots \end{pmatrix}$$

is unreduced upper Hessenberg; its $m + 1$ st row contains m zeros, the entry $h_{m+1,m}^{(1)}$ and then the first row of $\underline{H}_m^{(2)}$ since the zero block above $\underline{H}_m^{(2)}$ is $m \times m$.

We now generalize the previous theorem for the other cycles. During the second restart cycle, let an Arnoldi relation denoted

$$AV_{n,m}^{(3)} = V_{n,m+1}^{(3)} \underline{H}_m^{(3)} \quad (11.11)$$

be generated where the matrix $V_{n,m+1}^{(3)}$ has orthogonal columns. It follows from Theorem 3.6 that

$$V_{n,m+1}^{(3)} e_1 = \frac{r_m^{(2)}}{\|r_m^{(2)}\|} = \frac{\|r_m^{(2)}\|}{\|r_0\|} V_{n,m+1}^{(2)} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(2)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(2)} \end{pmatrix}, \quad (11.12)$$

see also (11.1).

For the same reasons as explained in Lemma 11.1, the decomposition (11.11) with $\underline{H}_m^{(3)}$ given is generated by restarted GMRES(m) applied to $A = VH V^*$ and $b = Ve_1$ if (and only if)

$$HZ_{n,m}^{(3)} = Z_{n,m+1}^{(3)} \underline{H}_m^{(3)}, \quad (11.13)$$

with initial vector

$$\begin{aligned} Z_{n,m+1}^{(3)} e_1 &= V^* V_{n,m+1}^{(3)} e_1 = V^* V_{n,m+1}^{(2)} \frac{\|r_m^{(2)}\|}{\|r_0\|} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(2)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(2)} \end{pmatrix} \\ &= \frac{\|r_m^{(2)}\|}{\|r_0\|} Z_{n,m+1}^{(2)} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(2)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(2)} \\ 0 \\ \vdots \end{pmatrix} = \frac{\|r_m^{(2)}\|}{\|r_0\|} \begin{pmatrix} d^{(1)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(2)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(2)} \\ 0 \\ \vdots \end{pmatrix}. \end{aligned}$$

Let us use the notation

$$\begin{pmatrix} d^{(2)} \\ 0 \\ \vdots \end{pmatrix} \equiv \frac{\|r_m^{(2)}\|}{\|r_0\|} \begin{pmatrix} d^{(1)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(2)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(2)} \\ 0 \\ \vdots \end{pmatrix} \quad (11.14)$$

and $(d^{(2)})^T = [\hat{d}^{(2)}, d_{2m+1}^{(2)}]^T$ and let us choose the matrix $Z_{m+1}^{(3)}$ with orthonormal columns in (11.13) to have the simple form

$$Z_{m+1}^{(3)} = \begin{pmatrix} d^{(2)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix}. \quad (11.15)$$

Let $H_{1:2m+1,1:2m}$ be as defined in Theorem 11.2, which guarantees that the initial cycle and the first restart applied to H and e_1 generate the given matrices $\underline{H}_m^{(1)}$ and $\underline{H}_m^{(2)}$, respectively. To generate in the next cycle the prescribed matrix $\underline{H}_m^{(3)}$, we can

apply Theorem 11.2 analogously. It gives that the first $3m$ columns of H are of the form

$$H_{:,1:3m} = \begin{pmatrix} H_{2m+1,2m} & 0 \\ 0 & \underline{H}_m^{(3)} \\ \vdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & \underline{H}_0^{(3)} \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix},$$

where the size $(2m+2) \times m$ matrix $\underline{H}_0^{(3)}$ is the rank-two matrix

$$\begin{aligned} \underline{H}_0^{(3)} &= \begin{pmatrix} \hat{d}^{(2)} \\ d_{2m+1}^{(2)} - 1 \\ 0 \end{pmatrix} e_1^T \underline{H}_m^{(3)} \\ &\quad - \frac{1}{d_{2m+1}^{(2)}} \left(\begin{pmatrix} (d_{2m+1}^{(2)} - 1) h_{1,1}^{(3)} d^{(2)} \\ -(d_{2m+1}^{(2)} - 1) h_{2,1}^{(3)} \end{pmatrix} + \begin{pmatrix} H_{1:2m+1,1:2m} \hat{d}^{(2)} \\ 0 \end{pmatrix} \right) e_1^T. \end{aligned}$$

All further columns of H , determining the Hessenberg matrices generated in all cycles after the third cycle, can be defined analogously. We summarize this result in the following theorem. Please note that for ease of presentation, it uses triangular Hessenberg decompositions where the $(1, 1)$ entries of the triangular matrices are not equal to one.

Theorem 11.3 *Let us for N such that $Nm < n$ have Nm given positive decreasing numbers*

$$\begin{aligned} f_0^{(1)} &\geq f_1^{(1)} \geq \dots f_{m-1}^{(1)} > f_m^{(1)} = f_0^{(2)} \\ &\geq f_1^{(2)} \geq \dots f_{m-1}^{(2)} > f_m^{(2)} = f_0^{(3)} \\ &\quad \vdots \\ &\geq f_1^{(N)} \geq \dots f_{m-1}^{(N)} > f_m^{(N)} > 0 \end{aligned} \tag{11.16}$$

and N given size $(m+1) \times m$ upper Hessenberg matrices $\underline{H}_m^{(k)}$, $1 \leq k \leq N$ of the form

$$\underline{H}_m^{(k)} = \begin{pmatrix} \vartheta_{1,1}^{(k)} & \dots & \vartheta_{1,m+1}^{(k)} \\ 0 & T_m^{(k)} & \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I_m \end{pmatrix} \begin{pmatrix} \vartheta_{1,1}^{(k)} & \dots & \vartheta_{1,m}^{(k)} \\ 0 & T_{m-1}^{(k)} & \end{pmatrix}, \tag{11.17}$$

where

$$\vartheta_{1,1}^{(k)} = \frac{1}{f_0^{(k)}}, \quad \vartheta_{1,j}^{(k)} = \sqrt{\left(\frac{1}{f_{j-1}^{(k)}} \right)^2 - \left(\frac{1}{f_{j-2}^{(k)}} \right)^2},$$

and where $T_m^{(k)}$ is a nonsingular upper triangular matrix with leading principal submatrix $T_{m-1}^{(k)}$. Let

$$\begin{pmatrix} d^{(k)} \\ 0 \\ \vdots \end{pmatrix} = f_m^{(k)} \begin{pmatrix} d^{(k-1)} & 0 \\ 0 & I_m \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\vartheta}_{1,1}^{(k)} \\ \vdots \\ \bar{\vartheta}_{1,m+1}^{(k)} \\ 0 \\ \vdots \end{pmatrix},$$

where $d^{(0)} \equiv 1$ and let

$$(d^{(k)})^T = [\hat{d}^{(k)}, d_{km+1}^{(k)}]^T.$$

Then GMRES(m) applied to the unreduced upper Hessenberg matrix H and e_1 generates consecutively the residual norms satisfying $\|r_j^{(k)}\| = f_j^{(k)}$ and the upper Hessenberg matrices $\underline{H}_m^{(1)}, \dots, \underline{H}_m^{(N)} \in \mathbb{C}^{(m+1) \times m}$ if the first m columns of H are

$$\underline{H}_{:,1:m} = \begin{pmatrix} \underline{H}_m^{(1)} \\ 0 \\ \vdots \end{pmatrix}$$

and for every k , $1 \leq k < N$, the first $(k+1)m$ columns of H are of the form

$$\underline{H}_{:,1:(k+1)m} = \begin{pmatrix} H_{1:km+1,1:km} & 0 \\ 0 & \underline{H}_m^{(k+1)} \\ \vdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & \underline{H}_0^{(k+1)} \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix},$$

where the size $(km+2) \times m$ matrix $\underline{H}_0^{(k+1)}$ is the rank-two matrix

$$\begin{aligned} \underline{H}_0^{(k+1)} &= \begin{pmatrix} \hat{d}^{(k)} \\ d_{km+1}^{(k)} - 1 \\ 0 \end{pmatrix} e_1^T \underline{H}_m^{(k+1)} \\ &\quad - \frac{1}{d_{km+1}^{(k)}} \left(\begin{pmatrix} (d_{km+1}^{(k)} - 1)h_{1,1}^{(k+1)}d^{(k)} \\ -(d_{km+1}^{(k)} - 1)h_{2,1}^{(k+1)} \end{pmatrix} + \begin{pmatrix} H_{1:km+1,1:km} \hat{d}^{(k)} \\ 0 \end{pmatrix} \right) e_1^T. \end{aligned}$$

The previous theorem shows that any decreasing convergence curve is possible for restarted GMRES provided strict decrease is prescribed at the end of all cycles. However, this is proved for the N initial cycles only, where $Nm < n$. Remarks for the situation $Nm \geq n$ in the restarted FOM method can be found in [819]. The theorem also shows how to prescribe *all* the Hessenberg matrices generated during all the N restarts. By the choice of the entries of these Hessenberg matrices, we can prescribe as well the Ritz values or the harmonic Ritz values of the cycles. We remark that other choices of the entries of the Hessenberg matrices might prescribe other interesting

values. For example, in [296, Section 6.3], it is shown that the singular values can be prescribed. As for the spectrum of the system matrix, we can use the following.

Lemma 11.2 *Let H be an unreduced upper Hessenberg matrix of size n whose leading $n - 1$ columns are given and consider n complex numbers $\lambda_1, \dots, \lambda_n$. The last column of H can be chosen such that H has the eigenvalues $\lambda_1, \dots, \lambda_n$.*

Proof See [287, Theorem 2.2] or [754, Theorem 3]. \square

The construction in Theorem 11.3 represents only one particular way to prescribe the residual norm history for GMRES(m), because it assumes a specific chosen structure of the bases $Z_{n,m}^{(k)}$ in the subsequent restart cycles (see, e.g., equations (11.7), (11.15)). It is a construction with the following interesting property.

Theorem 11.4 *Consider the matrix H constructed in Theorem 11.3. The residual norms generated when GMRES(m) is applied to H and e_1 are the same as when full GMRES is applied to this linear system.*

Proof The first m residual norms are trivially identical. For the rest of the proof we will compare the sizes of the values $\vartheta_{1,i}$ corresponding to full GMRES with those corresponding to restart cycles; if they are equal, the residual norms are equal as well. The values of the $\vartheta_{1,i}$ for full GMRES can be computed from the recurrence (2.21) in Lemma 2.2. The $\vartheta_{1,i}^{(k)}$ for the k th cycle can be obtained using (2.21) applied to $\underline{H}_m^{(k)}$.

The first $\vartheta_{1,i}^{(2)}$ for the first restart are, with Lemma 2.2,

$$\vartheta_{1,1}^{(2)} \equiv \frac{1}{\|r_0^{(2)}\|}, \quad \vartheta_{1,2}^{(2)} \equiv -\frac{\vartheta_{1,1}^{(2)} h_{1,1}^{(2)}}{h_{2,1}^{(2)}}. \quad (11.18)$$

The value $\vartheta_{1,m+2}$ for full GMRES is

$$\vartheta_{1,m+2} = \frac{-1}{h_{m+2,m+1}} (\vartheta_{1,1}, \dots, \vartheta_{1,m+1})^T \begin{pmatrix} h_{1,m+1} \\ \vdots \\ h_{m+1,m+1} \end{pmatrix}, \quad (11.19)$$

where the entries $h_{1,m+1}, \dots, h_{m+1,m+1}$ are given by

$$\begin{pmatrix} h_{1,m+1} \\ \vdots \\ h_{m+1,m+1} \end{pmatrix} = \frac{1}{d_{m+1}^{(1)}} \left(h_{1,1}^{(2)} d^{(1)} - \underline{H}_m^{(1)} \begin{pmatrix} d_1^{(1)} \\ \vdots \\ d_m^{(1)} \end{pmatrix} \right), \quad h_{m+2,m+1} = \frac{h_{2,1}^{(2)}}{d_{m+1}^{(1)}},$$

see (11.9). If we multiply the first equality in the previous equation with the row vector $(\vartheta_{1,1}, \dots, \vartheta_{1,m+1})^T$ we have for the first term on the right-hand side $(\vartheta_{1,1}, \dots, \vartheta_{1,m+1})^T h_{1,1}^{(2)} d^{(1)} = \vartheta_{1,1}^{(2)} h_{1,1}^{(2)}$ because

$$(\vartheta_{1,1}, \dots, \vartheta_{1,m+1})^T d^{(1)} = 1/\|r_m^{(1)}\| = \vartheta_{1,1}^{(2)}.$$

The second term is zero because

$$\begin{aligned} (d^{(1)})^* \underline{H}_m^{(1)} &= \\ \|\underline{r}_m^{(1)}\| e_1^T \begin{pmatrix} \vartheta_{1,1} & \dots & \vartheta_{1,m+1} \\ 0 & T_m \end{pmatrix} &. \\ \begin{pmatrix} \vartheta_{1,1} & \dots & \vartheta_{1,m+1} \\ 0 & T_m \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I_m \end{pmatrix} \begin{pmatrix} \vartheta_{1,1} & \dots & \vartheta_{1,m} \\ 0 & T_{m-1} \end{pmatrix} &= 0 \cdot e_1^T. \end{aligned}$$

Substituting in (11.19) gives $\vartheta_{1,m+2} = \frac{-\vartheta_{1,1}^{(2)} h_{1,1}^{(2)}}{h_{2,1}^{(2)}}$, which equals the second expression in (11.18).

The next value $\vartheta_{1,i}$ after the first restart is, with Lemma 2.2,

$$\vartheta_{1,3}^{(2)} \equiv \frac{-(\vartheta_{1,1}^{(2)}, \vartheta_{1,2}^{(2)})^T \begin{pmatrix} h_{1,2}^{(2)} \\ h_{2,2}^{(2)} \end{pmatrix}}{h_{3,2}^{(2)}}.$$

For full GMRES we have

$$\vartheta_{1,m+3} = \frac{-(\vartheta_{1,1}, \dots, \vartheta_{1,m+1})^T d^{(1)} h_{1,2}^{(2)} - \vartheta_{1,m+2} h_{2,2}^{(2)}}{h_{3,2}^{(2)}},$$

see (11.10). Thus $\vartheta_{1,m+3} = \vartheta_{1,3}^{(2)}$. The equality of the remaining $\vartheta_{1,i}$, and therefore of the remaining residual norms follows by induction. The same proof can be used for subsequent restart cycles. \square

Thus the linear systems we had constructed in Theorem 11.3 to generate pre-specified GMRES(m) residual norms represent, in fact, the best case scenario for restarted GMRES: It converges as fast as full GMRES. In other words, the GMRES minimization process can for these systems be carried out with $m + 1$ -term recurrences. This property is due to the special structure of the Arnoldi vectors in the individual restart cycles used to construct the linear systems (see, e.g., equations (11.7) and (11.15)). It implies that all Arnoldi vectors generated in subsequent restart cycles except the initial ones are orthogonal to each other (this is, in fact, the situation where the condition number in (11.2) is one).

The matrix H constructed in Theorem 11.3 is, in general, not a normal matrix, but the GMRES minimization process can be carried out with short recurrences. This does not contradict the Faber–Manteuffel theorem [334], [338], [640, Chapter 4], because in our case the orthogonal basis can be generated with short recurrences for a particular right-hand side vector only (the first unit vector).

An interesting consequence is that no other restart length than m can give faster convergence (as no restart length can give faster convergence than full GMRES). In particular, not even *larger* restart lengths produce convergence faster than GMRES(m) for the systems constructed in Theorem 11.3. Thus we have found a class of systems exhibiting the counterintuitive behavior encountered sometimes in practice; some numerical examples are given at the end of this chapter.

11.3 Restarting techniques for FOM and GMRES

It is well known and we will see in the numerical experiments that GMRES(m) is not always very efficient, particularly for small values of m . Hence, a straightforward idea is to change adaptively the restart parameter m to try to improve convergence. This is not easy since we have already said that increasing m does not always give an improved convergence. Nevertheless, several strategies have been proposed to choose a “good” value of m . They are based on what happened in the previous cycles. Let us briefly review them by order of appearance.

The paper [582] aims at reducing the total time to the solution by examining the effect of m on both the convergence behavior and the computational costs. An adaptive method is proposed that determines whether or not to restart based on a criterion that weighs the work requirements against the estimated residual norm decrease.

In [871], the goal of the authors is to prevent stagnation. They employ a stagnation test for insufficient residual norm reduction over a cycle. GMRES(m) is declared to have stagnated and the iteration is aborted if, at the average rate of residual decrease over the last restart cycle, the stopping criterion based on the residual norm cannot be satisfied in some large multiple of the remaining number of allowed iterations. A slow progress of GMRES(m) is detected with a similar test. If quasi-stagnation occurs, the restart value m is incremented by some value d . Such a technique is used whenever needed if the restart value m is less than some given maximum value m_{\max} . When the maximum value m_{\max} is reached, adaptive GMRES(m) proceeds as GMRES(m_{\max}). The values of the parameters used in the criteria for quasi-stagnation and slow decrease are established experimentally. We observe that this paper uses the Householder implementation of GMRES and always increases the restart parameter.

The adaptive method proposed in [497] is a modification of the method in [871] that also increases the restart parameter m to avoid stagnation, but then reduces m after a fixed number of cycles to better control the computing costs.

The method in [1028] has also the goal of preventing stagnation. The authors choose m based on a comparison of the Ritz and harmonic Ritz values. They refer to [440] to show that their differences are large when GMRES stagnates. They compute the modulus of the difference between the maximum of moduli of the Ritz values and the maximum of moduli of the harmonic Ritz value and restart if it is larger than what it was before. However, this looks like a costly way to estimate if stagnation occurs since this can be obtained from the residual norms.

The paper [707] is presented as an improvement of [928] which restarts when the zeros of the residual polynomial are distributed uniformly in the complex plane. The authors compute the zeros (the harmonic Ritz values) in a complicated way, enclose them in rectangular regions and provide a criterion to decide if the distribution is adequate. The determination of the restart parameter is carried out only at even steps. They added a test using the residual vectors indicating if there is quasi-stagnation.

The authors of [63] propose a simple strategy for varying the restart parameter m and provide some heuristic explanations. They use the angles between the residual vectors at the end of successive cycles that they call *sequential angles* such that

$$\cos \theta(r_m^{(i+1)}, r_m^{(i)}) = \frac{\|r_m^{(i+1)}\|}{\|r_m^{(i)}\|}.$$

They also define minimum and maximum values $m_{\min, \max}$ for m . The first cycle begins with restart parameter m_{\max} . After that, they compute the sequential angle at the end of each cycle to determine the next parameter m_i . They decrease the restart parameter by a small positive number d until m_{\min} is reached. At that point, they increase m_i but not more than the maximum m_{\max} . But, if the sequential angle is large then they keep the current restart parameter instead of modifying it. If the sequential angle is small, they revert to m_{\max} . The authors consider “small” sequential angles to be those less than about 8 degrees and “large” angles to be those greater than 80 degrees. The aim of this paper is to improve convergence by choosing a pattern of restart parameters that includes jumps from small to large restart parameters. This technique was proposed after making the observation that the GMRES(m) residual vectors at the end of each cycle often alternate direction in a repetitive fashion, slowing down the convergence. The proposed method aims at disrupting this alternating pattern. For an other strategy for varying m , based on control, see [241].

Besides changing the value of the restart parameter m , three main techniques have been used for improving the convergence of GMRES(m). They are known as *deflation*, *augmentation* and *spectral preconditioning*. Concerning GMRES, there is some confusion about the use of the word “deflation”. Some authors use it to indicate the use of projectors constructed using some spectral information whence for others it is more or less the same as augmentation when the added subspace uses some spectral information.

We first consider some results about the general principle of augmentation. In [796] (see also [213]), augmentation was described as follows. Given a subspace \mathcal{W}

of dimension k , the approximate solutions are sought in $x_0 + K^{(m,k)}$ where

$$K^{(m,k)} = \mathcal{K}_m(A, r_0) + \mathcal{W},$$

for which the matrix $W_{n,k}$, representing a basis of \mathcal{W} , has k independent columns. In a minimum residual (MR) method the approximate solution is such that the residual vector is orthogonal to $AK^{(m,k)}$. We have seen that this minimizes the residual norm. Some algorithms to be described below yield a relation

$$A[V_{n,m}, W_{n,k}] = V_{n,m+k+1} \underline{H}_{m+k},$$

with basis vectors of $\mathcal{K}_m(A, r_0)$ as columns of $V_{n,m}$ and \underline{H}_{m+k} being an $(m+k+1) \times (m+k)$ upper Hessenberg matrix. In [796] it is proved that if there exists a vector $w \in \mathcal{W}$ such that $Aw = v_i$ for some $i \leq m$ and if H_{m+k} is nonsingular, the space $x_0 + K^{(m,k)}$ contains an exact solution of $Ax = b$. Hence, it means that it could be a good idea to augment the Krylov subspace with approximate solutions of $Aw = v_i$. This idea is used in methods with inner and outer iterations like GCRO and GMRESR to be described in Chapter 12.

Another idea is to use a subspace \mathcal{W} close to an invariant subspace of A . When two finite-dimensional subspaces \mathcal{X} and \mathcal{Y} are of the same dimension, the *gap* $\Theta(\mathcal{X}, \mathcal{Y})$ between \mathcal{X} and \mathcal{Y} is defined as

$$\Theta(\mathcal{X}, \mathcal{Y}) = \|P_{\mathcal{X}} - P_{\mathcal{Y}}\|,$$

where $P_{\mathcal{X}}$ (resp. $P_{\mathcal{Y}}$) is the orthogonal projector onto \mathcal{X} (resp. \mathcal{Y}). Let \mathcal{U} be an invariant subspace for A (which means that $A\mathcal{U} \subseteq \mathcal{U}$) and \mathcal{W} be such that $\Theta(\mathcal{U}, A\mathcal{W}) = \varepsilon$. Then, the minimum residual norm satisfies

$$\|r_m\| \leq \min_{q \in \pi_m, q(0)=1} \{\|q(A)(I - \mathcal{P}_{\mathcal{U}})r_0\| + \varepsilon\|q(A)\mathcal{P}_{\mathcal{U}}r_0\|\}, \quad (11.20)$$

where $\mathcal{P}_{\mathcal{U}}$ is any projector on \mathcal{U} . For an application when $\mathcal{P}_{\mathcal{U}} = P_{\mathcal{U}}$ is the orthogonal projector on a subspace spanned by eigenvectors of A , the following result is proved in [796]. Assume there is a polynomial q in π_m with $q(0) = 1$ such that

$$\|q(A)(I - P_{\mathcal{U}})r_0\| \leq s_m\|(I - P_{\mathcal{U}})r_0\|, \quad \|q(A)\mathcal{P}_{\mathcal{U}}r_0\| \leq c_m\|P_{\mathcal{U}}r_0\|,$$

with $s_m^2 + c_m^2 = 1$. Then,

$$\|r_m\| \leq (s_m^2 + \varepsilon^2 c_m^2)^{\frac{1}{2}} \|r_0\|.$$

See also Corollary 3.3 in [796]. For instance, q can be the residual polynomial obtained when solving $Ay = (I - P_{\mathcal{U}})r_0$. Note that in $(I - P_{\mathcal{U}})r_0$, the components associated with the subspace \mathcal{U} are removed. Hence, if ε is small, the upper bound

(11.20) on the residual norm almost corresponds to solving a problem where the eigenvalues corresponding to \mathcal{U} have been removed.

The paper [297] contains, among other things, an interesting study of restarted GMRES. The authors show that for two consecutive cycles

$$\mathcal{K}_m(A, r_m^{(i)}) \oplus \mathcal{K}_m(A, r_m^{(i+1)}) = \mathcal{K}_{2m}(A, r_m^{(i)}),$$

if and only if there is no stagnation in the last step of cycle i . They also proved the following. Let $r_{m+k} = p_{m+k}(A)r_0$ be the residual vector at the end of a cycle. The roots of the polynomial p_{m+k} are the harmonic Ritz values for this cycle. Now, let us augment the Krylov subspace with approximate eigenvectors y_1, \dots, y_k corresponding to distinct harmonic Ritz values $\theta_1, \dots, \theta_k$. Then,

$$\mathcal{K}_m(A, r_{m+k}) + \text{span}\{y_1, \dots, y_k\} = \mathcal{K}_{m+k}(A, q_m(A)r_0),$$

where q_m is a polynomial of degree m whose roots are the other harmonic Ritz values $\theta_{k+1}, \dots, \theta_{m+k}$. Therefore, the subspace that is constructed by augmentation with harmonic Ritz vectors is itself a Krylov subspace, although with a different starting vector.

An interesting review of augmentation and deflation techniques is presented in [837].

In [402] a general framework is introduced for deflation and augmentation. It is shown that augmentation can be achieved either explicitly, or implicitly, namely, by projecting the residuals appropriately and correcting the approximate solutions in a final step. The iterates are obtained as

$$x_\ell = x_0 + V_{n,\ell} y_\ell + U u_\ell,$$

with the columns of U being linearly independent and spanning a subspace \mathcal{U} . To obtain a minimal residual norm the corresponding residual vector must satisfy

$$r_\ell \perp A\mathcal{K}_\ell(A, r_0), \quad r_\ell \perp A\mathcal{U}.$$

The second condition yields

$$u_\ell = E^{-1} U^* A^* (r_0 - AV_{n,\ell} y_\ell), \quad E = U^* A^* A U.$$

The matrix E is symmetric and nonsingular. Plugging u_ℓ in the equation of the residual we obtain

$$r_\ell = (I - AUE^{-1}U^*A^*)(r_0 - AV_{n,\ell}y_\ell).$$

Let $M = UE^{-1}U^*$, $P = I - AMA^*$ and $Q = I - P$. Then,

$$r_\ell = P(r_0 - AV_{n,\ell}y_\ell), \quad x_\ell = Q(x_0 + V_{n,\ell}y_\ell) + MA^*b.$$

P and Q are projectors. P is an orthogonal projector even if E is not positive definite. As shown in [402] we have $PAU = 0$, $QU = 0$ and $PA = PAQ = AQ$. We obtain a minimal residual norm by imposing the condition

$$r_\ell = P(r_0 - AV_{n,\ell}y_\ell) \perp A\mathcal{K}_\ell(A, r_0).$$

The authors called this *explicit deflation and augmentation* when A is itself a deflated matrix. This deflated matrix can be chosen as PA , that is, A can be replaced by PA and r_0 by Pr_0 . Theorem 3.3 of [402] shows what the Jordan canonical form of PA looks like when \mathcal{U} is chosen from a right-invariant subspace or when $A\mathcal{U}$ corresponds to a left-invariant subspace of A . Loosely speaking, in both cases, the multiplication with the operator P removes certain eigenvalues of the matrix A by moving them to zero. This is the sense of “deflation” in [402], see also [326].

The paper [1012] proposes also to solve a deflated linear system. However, the authors observe that deflation seems not always be effective, for example, if the matrix A is not normal. The choice for approximate eigenvectors as deflation vectors is often justified only by numerical experiments. The proposed algorithm named D-GMRES uses deflation vectors that span an “exact” invariant subspace of A . Let

$$P = I - AZ(Z^*AZ)^{-1}Z^*, \quad \tilde{P} = I - Z(Z^*AZ)^{-1}Z^*A.$$

P and \tilde{P} are projectors and $PA = A\tilde{P}$. The authors assume that the columns of Z are basis vectors of an A -invariant subspace. Let $x_1 = Z(Z^*AZ)^{-1}Z^*b$. Then, we have to solve $PAx = Pb$ using GMRES obtaining an approximate solution \tilde{x} . With $x_2 = \tilde{P}\tilde{x}$, the approximate solution is $x_1 + x_2$.

If Z corresponds to eigenvalues $\lambda_1, \dots, \lambda_k$ of A , the spectrum of PA contains the eigenvalues of A except $\lambda_1, \dots, \lambda_k$. The deflated matrix PA is singular but the linear system $PAx = Pb$ is consistent and the authors of [1012] proved that GMRES converges to the solution. The residual norms in D-GMRES are smaller than or equal to those of GMRES applied to the original system. Unfortunately, the cases of approximate eigenvectors and restarting are not considered in this paper.

An interesting study of GMRES applied to a deflated linear system, that is, with a singular matrix, is done in [485], including the possible breakdown conditions. In that paper oblique projections are also considered, the aim being to apply deflation to methods like QMR using the nonsymmetric Lanczos algorithm.

We now turn to practical implementations of some of the ideas described above, starting with augmentation techniques.

In [64], a method named LGMRES (meaning “Loose” GMRES) is proposed, see also [60]. The authors start from the observation that, quite often, GMRES residual vectors at the end of every other cycle are almost in the same direction, that is, the angle between $r_m^{(i-1)}$ and $r_m^{(i+1)}$ is small. To prevent this behavior, they

augment the Krylov subspace with what is supposed to be an approximation of the error vector, $z^{(i)} = x_m^{(i)} - x_m^{(i-1)}$. Of course, if m is small, this can only be a very crude approximation of the error vector. The rationale for doing this is that if we add the exact error vector to the current approximate solution we obtain the exact solution of the linear system. We observe that we could use any rough approximation of the solution of $Ae = r$, see the GCRO method [260, 262] to be described in Chapter 12. LGMRES(m, k) augments the standard Krylov subspace with k previous approximations of the error vector,

$$x_m^{(i+1)} = x_m^{(i)} + q_{m-1}^{(i)}(A)r_m^{(i)} + \sum_{j=i-k+1}^i \alpha_j^{(i)} z^{(j)},$$

where the polynomial and the coefficients are chosen to minimize the corresponding residual norm. The implementation is similar to GMRES-E [695] to be described below. For simplicity of notation, we use lower indices for the z vectors. Starting from the last residual vector r_0 of the previous cycle, let $\beta = \|r_0\|$, $v_1 = r_0/\beta$ and $s = m + k$. For $j = 1, \dots, s$, we define

$$u = Av_j \text{ if } j \leq m, \quad Az_{i-(j-m-1)} \text{ otherwise.}$$

Then, as in the Arnoldi process, we orthogonalize u against the previous basis vectors obtaining the entries $h_{\ell,j}$ of the upper Hessenberg matrix \underline{H}_{s+1} and the next basis vector v_{j+1} . Let

$$V_{n,s+1} = (v_1 \cdots v_{s+1}), \quad W_{n,s} = (v_1 \cdots v_m \ z_i \cdots z_{i-k+1}).$$

Then we have the relation,

$$AW_{n,s} = V_{n,s+1} \underline{H}_s.$$

As in standard GMRES, we compute the solution y_s of $\min_y \|\beta e_1 - \underline{H}_s y\|$ and

$$z_{i+1} = W_{n,s} y_s, \quad Az_{i+1} = V_{n,s+1} \underline{H}_s y_s.$$

The new approximate solution is $x_{new} = x_{old} + z_{i+1}$. Therefore, this algorithm can be implemented by doing only slight modifications to GMRES(m). We need to store the vectors z_{i+1} and Az_{i+1} for the next cycles but this is only two vectors per cycle. There are $m + k$ matrix–vector products per cycle. We observe that the augmentation happens only at the end of a cycle. Therefore, its effect is only on the last approximate solution and on the restarting vector for the next cycle.

A somehow more efficient block implementation of LGMRES, named B-LGMRES, is considered in [61], see also [60]. A block GMRES method (see [722, 798, 954]) is used for solving the linear system with a single right-hand side. The block right-hand side is $[r_m^{(i)}, z^{(i)}, \dots, z^{(i-k+1)}]$, that is, the residual vector and k of

the $z^{(i)}$ vectors. The only drawback is that the matrix \underline{H}_s has now k subdiagonals and k^2 Givens rotations are needed to transform it to upper triangular form.

More recently, two variants of GMRES(m) named locally optimal GMRES (LOGMRES) and heavy ball GMRES (HBGMRES) were proposed in [542]. The goal is to use more information from the previous cycles than in standard GMRES(m). The idea for LOGMRES is to minimize the norm of the residual in $x_0^{(i)} + \mathcal{K}_k(A, r_0^{(i)}) + \text{span}(x_0^{(i-1)})$ or with $x_0^{(i-1)}$ replaced by a combination of $x_0^{(i-1)}$ and $x_0^{(i)}$. HBGMRES uses the difference $x_0^{(i-1)} - x_0^{(i)}$. It minimizes $\|b - A(x_0^{(i)} + z)\|$ with $z \in \mathcal{K}_k(A, r_0^{(i)}) + \text{span}(x_0^{(i)} - x_0^{(i-1)})$. These algorithms look very similar to LGMRES but the paper [64] is not cited in [542]. Note that, in this paper, only one vector is added to the Krylov subspace and the implementation is much more complicated than in the LGMRES algorithm. Related papers are [543, 544].

We now describe some methods based on augmentation of the Krylov subspace with approximate eigenvectors. The development of these methods is based on the belief that, for some problems, eigenvalues of A of small modulus slow down the convergence of GMRES. Unfortunately, we know that this is not always true. We have seen in Chapter 5 how to construct matrices and right-hand sides such that we have prescribed residual norms as well as prescribed eigenvalues and (harmonic) Ritz values. Hence, we can have a bad convergence of GMRES without having eigenvalues of small modulus. However, it is fair to say that these constructed matrices are generally badly conditioned and non-normal. For normal or near-normal matrices we have seen that GMRES convergence depends more clearly on the eigenvalue distribution. In these cases, the methods described below may be helpful, even though their efficiency depends on the problems to be solved. Another important question is the choice of the number of approximate eigenvectors to be added. Storage considerations may influence this choice but, otherwise, up to our knowledge, there is no study of this problem in the literature.

As we said, the idea is to add approximate eigenvectors to the subspace. If an exact eigenvector of A is added to the subspace, the corresponding eigenvalue is deflated, that is, eliminated from the spectrum. However, we have usually at hand only approximate eigenvectors obtained from the Arnoldi process.

Restarted FOM methods can be easily obtained by doing appropriate modifications.

In GMRES-E [695], k approximate harmonic Ritz vectors are added at the end of the Krylov subspace. The harmonic Ritz vectors y_i are obtained from GMRES iterations by solving

$$(H_j + h_{j+1,j}^2 H_j^{-*} e_j e_j^T) g_i = \zeta_i g_i, \quad y_i = V_{n,j} g_i, \quad i = 1, \dots, k,$$

when H_j with $j \geq k$ is nonsingular (which means that GMRES does not stagnate). In general, the vectors corresponding to the smallest harmonic Ritz values in magnitude are chosen.

Let $s = m + k$, $W_{n,s} = (V_{n,m} \ y_1 \ \dots \ y_k)$ and Q be an $n \times (s + 1)$ matrix whose first $m + 1$ columns are $V_{n,m+1}$ and the k last columns are obtained by orthogonalizing

the vectors Ay_i against the previous columns of Q . We have the relation

$$AW_{n,s} = Q\underline{H}_s,$$

where \underline{H}_s is an $(s+1) \times s$ upper Hessenberg matrix. The implementation is similar to what we have seen for LGMRES. The approximate solution is $x_0 + W_{n,s}y$ where y is obtained by minimizing the residual norm, that is, by solving $\min_y \|Q^*r_0 - \underline{H}_s y\|$.

The implicitly restarted GMRES, GMRES-IR [696], uses also harmonic Ritz vectors. It is mathematically equivalent to GMRES-E (if the same approximate eigenvectors are used). In this method ideas from the implicit Arnoldi algorithm (IRA) are used, see [623, 626, 870]. Compared to GMRES-E the number of matrix–vector products is smaller. GMRES-IR uses the same subspace as GMRES-E. Assuming that the same underlying Arnoldi iteration is used, the harmonic residual vectors are multiples of the GMRES residual vector. An elementary proof is the following.

Proposition 11.1 *Let ζ_i (resp. y_i) be the harmonic Ritz values (resp. vectors) obtained after m iterations of GMRES and let r_m be the residual vector. Then, the residuals of the eigenvalue problem satisfy*

$$\tilde{r}_i = Ay_i - \zeta_i y_i = \gamma_i r_m, \quad \forall i,$$

where γ_i is a scalar.

Proof For the eigenproblem residual we have

$$\begin{aligned} \tilde{r}_i &= Ay_i - \zeta_i y_i, \\ &= AV_{n,m}g_i - \zeta_i V_{n,m}g_i, \\ &= (V_{n,m}H_m + h_{m+1,m}v_{m+1}e_m^T)g_i - \zeta_i V_{n,m}g_i, \\ &= V_{n,m}(H_m g_i - \zeta_i g_i) + h_{m+1,m}[g_i]_m v_{m+1}, \\ &= V_{n,m}(-h_{m+1,m}^2[g_i]_m H_m^{-*} e_m) + h_{m+1,m}[g_i]_m v_{m+1}. \end{aligned}$$

The GMRES residual vector is

$$r_m = r_0 - (V_{n,m}H_m + h_{m+1,m}v_{m+1}e_m^T)y^M,$$

with y^M being the solution of

$$(H_m + h_{m+1,m}^2 H_m^{-*} e_m e_m^T)y^M = \|r_0\|e_1,$$

using the normal equations. Multiplying the last equation by $V_{n,m}$ and using $r_0 = \|r_0\|V_{n,m}e_1$, we obtain an expression for r_0 and

$$\begin{aligned} r_m &= h_{m+1,m}^2[y^M]_m V_{n,m}H_m^{-*} e_m - h_{m+1,m}[y^M]_m v_{m+1}, \\ &= h_{m+1,m}[y^M]_m(h_{m+1,m}V_{n,m}H_m^{-*} e_m - v_{m+1}). \end{aligned}$$

Comparing with the eigenproblem residual, we have

$$\tilde{r}_i = -\frac{[g_i]_m}{[y^M]_m} r_m.$$

and $\gamma_i = [g_i]_m / [y^M]_m$. □

Proposition 11.2 *The approximate eigenpair defined in Proposition 11.1 is an exact eigenpair of a matrix $A - E$ such that*

$$\|E\| = |\gamma_i| \|r_m\|.$$

The perturbation E can be chosen as a rank-one matrix $E = \gamma_i r_m y_i^$.*

Proof See, for instance, [794]. □

This shows that, if $|\gamma_i|$ is bounded, the norm of the perturbation E goes to zero with the norm of the GMRES residual vector. Note that the perturbation matrix is different for each approximate eigenvector.

Using Proposition 11.1 it is proved in [696] that

$$\text{span}\{y_1, Ay_1, \dots, A^{\ell-1}y_1\} = \text{span}\{y_1, r_m, Ar_m, \dots, A^{\ell-2}r_m\}.$$

This can be generalized to more eigenvectors by choosing carefully the starting vectors as linear combinations of the approximate eigenvectors to show that the generated subspace is a Krylov subspace, see [696].

The GMRES-IR method uses implicit restarting with unwanted harmonic Ritz values as shifts. Theorem 5.14 of [696] proves that GMRES-IR generates the wanted augmented subspace. The algorithm, as described in [696], is the following.

- (1) Choose x_0 and compute $r_0 = b - Ax_0$. Let $v_1 = r_0 / \|r_0\|$ and $\beta = \|r_0\|$.
- (2) Iterate: Apply the Arnoldi iteration up to step m .
- (3) Compute the approximate solution $x_m = x_0 + V_{n,m}y$, where y comes from a least squares problem. The vector y is the solution of $\min_y \|V_{n,m+1}^*r_0 - \underline{H}_m y\|$. Check for convergence.
- (4) Compute the harmonic Ritz values.
- (5) Restart: Let $x_0 = x_m$. Apply implicit restarting with the unwanted harmonic Ritz values as shifts. Both parts of a complex conjugate pair must be included. Go to (2), and resume the Arnoldi iteration from step $k + 1$.

In step (3), the vector $V_{n,m+1}^*r_0$ has zero components except in its first $k + 1$ entries. It can also be shown that after the Givens rotations are applied during solution of the least squares problem, the first k entries of this vector become zero.

The phase of implicit restarting is described by the following algorithm.

Let $p = m - k$ and ζ_1, \dots, ζ_p be the p shifts. Let $H^{(1)} = H_m$. We compute a QR factorization of a shifted matrix,

$$Q^{(i)} R^{(i)} = H^{(i)} - \zeta_i I.$$

The next matrix $H^{(i+1)}$ is defined by

$$H^{(i+1)} = R^{(i)} Q^{(i)} + \zeta_i I.$$

In practice, a double shift is used to avoid unitary factorization in the case of a complex θ_i . Define

$$Q = Q^{(1)} Q^{(2)} \cdots Q^{(p)}, \quad R = R^{(p)} R^{(p-1)} \cdots R^{(1)}.$$

Then,

$$\prod_{j=1}^p (H_m - \zeta_j I) = QR.$$

The new basis is given by $V^+ = V_{n,m} Q$ whose first k columns are used to start the next Arnoldi iteration.

However, it is known that IRA may have stability problems. In particular this happens when there are isolated eigenvalues in the spectrum. For computing approximations of eigenvalues these problems have been solved by *locking* and *purging*, see [623, 626, 870]. These techniques are quite complicated and are not implemented in GMRES-IR. Another, more complete, version of implicitly restarted GMRES using these techniques was proposed in [621].

Because of these stability problems another method was proposed in [697]. It is known as GMRES with deflated restarting or GMRES-DR, even though it is an augmentation method. It is mathematically equivalent to the methods GMRES-E and GMRES-IR. This method generalizes the thick restarting technique developed in [989] for symmetric problems.

The first cycle of GMRES-DR is done with the standard GMRES algorithm producing matrices $\tilde{V}_{n,m+1}$ and \tilde{H}_m and the residual vector of the least squares problem denoted as $c - \tilde{H}_m y^M$. In the first cycle $c = \|r_0^{(1)}\| e_1$. At the end of a cycle, the k desired harmonic Ritz pairs (ζ_i, g_i) are computed, the vectors g_i being of length m . However, we have to be careful if we want to use only real arithmetic when dealing with real data. We have to consider pairs of complex conjugate harmonic Ritz values (increasing the value of k if necessary) and combine the real and imaginary parts of the eigenvectors g_i .

The idea of the method is to add the harmonic Ritz vectors $y_i = V_{n,m} g_i$ at the beginning of the subspace for the next cycle. This is cleverly implemented in [697] using only the vectors g_i . These vectors are orthonormalized giving the columns p_1, \dots, p_k of a matrix $P_{m,k}$. This matrix is extended to the size $(m+1) \times k$ by appending a zero row at the bottom. The least squares residual (of length $m+1$) is

orthonormalized against the columns of the extended matrix yielding a vector p_{k+1} and a matrix $P_{m+1,k+1}$ of size $(m+1) \times (k+1)$. The matrices $\underline{H}_k = P_{m+1,k+1}^T \tilde{H}_m P_{m,k}$ and $V_{n,k+1} = \tilde{V}_{n,m+1} P_{m+1,k+1}$ are the beginning of the upper Hessenberg matrix and of the basis vectors of the new cycle. For stability reasons, the last basis vector v_{k+1} is orthogonalized against the other columns of $V_{n,k+1}$. From then on, the usual Arnoldi recurrence is used to compute the other basis vectors and the rest of the upper Hessenberg matrix. At the end of a cycle we obtain a relation

$$AV_{n,m} = V_{n,m+1} \underline{H}_m,$$

where \underline{H}_m is upper Hessenberg except for the (small) leading block of order $k+1$ which is a full matrix with nonzero entries. At the end of the cycle, we have to solve the least squares problem

$$\min_y \|c - \underline{H}_m y\|,$$

for y^M with $c = V_{n,m+1}^* r_{old}$, r_{old} being the residual vector at the end of the previous cycle and the approximate solution is $x_m = x_{old} + V_{n,m+1} y^M$, x_{old} being the approximate solution corresponding to r_{old} . This may seem strange since r_{old} normalized is not the first basis vector. But, this can be done (whatever is the basis) as long as r_{old} belongs to the subspace generated by the basis vectors because we can write $r_{old} = V_{n,m+1} c$. Here, since the basis vectors are orthonormal we obtain $c = V_{n,m+1}^* r_{old}$. In the general case we would have $c = (V_{n,m+1}^* V_{n,m+1})^{-1} V_{n,m+1}^* r_{old}$.

Solving the least squares problem is slightly different from what we usually do in GMRES since we have a dense block at the top-left of \underline{H}_m . A QR factorization is used to transform this block to triangular form and then Givens rotations are used for the rest of \underline{H}_m . This process is repeated for each cycle.

GMRES-DR is simpler than GMRES-IR and does not have the same numerical problems that are solved by locking and purging in the implicit Arnoldi algorithm (IRA). A (not fully optimized) code implementing GMRES-DR is following. In this code kR is the number k of harmonic Ritz vectors to be added. It can be increased by 1 to have pairs of complex conjugate harmonic Ritz values. Note that the approximate eigenvectors are recomputed at the end of each cycle.

```
function [x,ni,nc,resn] = GMRESm_DR(A,b,x0,epsi,nitmax,m,kR);
%
m = min(m,nitmax);
n = size(A,1);
rhs = zeros(m+1,1);
resn = zeros(1,nitmax+1);
nb = norm(b);
x = x0;
r = b - A * x;
rzero = r;
H = zeros(m+1,m);
```

```

HH = H;
bet = norm(r);
nc = 1; % number of cycles
ni = 0; % number of iterations
iconv = 0;
resn(1) = bet;
rhs(1) = bet;
rhsR = rhs;
%
% do a first cycle of GMRES(m)
V = zeros(n,m+1);
rot = zeros(2,m);
v = r / bet;
V(:,1) = v;
for k = 1:m
    ni = ni + 1; % number of iterations
    [V,H,HH,rhs,rhsR,rot] = gmres_iter(k,A,V,H,HH,rhs,rhsR,rot,
                                         rzero,0,[]);
    nresidu = abs(rhs(k+1));
    resn(ni+1) = nresidu;
    % convergence test or too many iterations
    if nresidu < (epsi * nb) || ni >= nitmax
        % convergence
        iconv = 1;
        break
    end % if nresidu
end % for k - end of the first cycle
% computation of the solution at the end of the first cycle
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + V(:,1:k) * y;
if iconv == 1
    % we have to stop
    resn = resn(1:ni+1);
    return
end % if iconv
% residual vector of the least squares problem
rR = rhsR - HH(1:k+1,1:k) * y;
% start the while loop with computation of the harmonic Ritz
% values and vectors sorted by modulus
kR0 = kR;
[g,kR] = harm_Ritz_vec(kR0,HH);
QQ = [];
%
while nresidu > (epsi * nb) && (ni < nitmax)
    % -- Loop on cycles

```

```

nc = nc + 1; % number of cycles
if kR >= m
    fprintf('\n Error: kR = %d is too large compare to
m = %d, ... stop \n\n',kR,m)
    return
end
rot = zeros(2,m); % init Givens rotations
rhs = zeros(m+1,1);
rhsR = rhs;
r = b - A * x;
rzero = r;
x0 = x;
if kR ~=0
    % first phase
    % orthogonalize the harmonic Ritz vectors
    P = orth_mgs(g,'dreorth');
    P = [P; zeros(1,size(P,2))]; % append a zero row
    PP = zeros(m+1,kR+1);
    PP(:,1:kR) = P;
    % double orthogonalization of rR against P
    w = orth_vec(P,rR);
    PP(:,kR+1) = w;
    V(:,1:kR+1) = V(:,1:m+1) * PP; % project V
    % orthogonalize v_(kR+1) against the previous basis vectors
    w = orth_vec(V(:,1:kR),V(:,kR+1));
    V(:,kR+1) = w;
    H(1:kR+1,1:kR) = PP' * HH(1:m+1,1:m)* P(1:m,:);% project H
    HH(1:kR+1,1:kR) = H(1:kR+1,1:kR); % save it
    HH(kR+2:m+1,1:kR) = zeros(m-kR,kR);
    H(kR+2:m+1,1:kR) = zeros(m-kR,kR);
    [QQ,RR] = qr(HH(1:kR+1,1:kR)); % QR factorization
    H(1:kR+1,1:kR) = RR(1:kR+1,1:kR);
    % part of the right-hand side
    rhs(1:kR+1) = V(:,1:kR+1)' * rzero;
    rhsR(1:kR+1) = rhs(1:kR+1); % save it
    rhs(1:kR+1)=QQ' * rhs(1:kR+1);% modify the right-hand side
else % if kR, case without augmentation
    rhs(1) = norm(rzero);
    V(:,1) = rzero / rhs(1);
end % if kR
% second phase: start a cycle of GMRES(m) from what has been
% already computed
for k = kR+1:m
    ni = ni + 1; % number of iterations
    [V,H,HH,rhs,rhsR,rot] = gmres_iter(k,A,V,H,HH,rhs,rhsR,rot,

```

```

rzero,kR,QQ); nresidu = abs(rhs(k+1));
resn(ni+1) = nresidu;
% convergence test or too many iterations
if nresidu < (epsi * nb) || ni >= nitmax
    % convergence
    iconv = 1;
    break % get out of the loop
end % if nresidu
end % for k - end of one cycle
% computation of the solution at the end of the cycle
y = triu(H(1:k,1:k)) \ rhs(1:k);
x = x0 + V(:,1:k) * y;
if iconv == 1
    % we have to stop
    resn = resn(1:ni+1);
    return
end % if iconv
% we have not converged yet,compute the harmonic eigenvectors
% and restart
if kR0 == 0
    g = [];
else
    [g,kR] = harm_Ritz_vec(kR0,HH);
end % if kR0
rR = rhsR - HH(1:k+1,1:k) * y;
end % while, loop on cycles
% if we get here we have done the max number of cycles,
% may be without convergence
resn = resn(1:ni);
end % function

```

In this code we used a few auxiliary functions. The function `orth_vec` does a double orthogonalization of the given vector against the columns of the matrix, using the modified Gram–Schmidt (MGS) algorithm.

```

function w = orth_vec(P,rR);
% double orthogonalize rR against the columns of P
kRP = size(P,2);
w = rR / norm(rR);
for j = 1:kRP
    alpha = P(:,j)' * w;
    w = w - alpha * P(:,j);
end % for j
for j = 1:kRP
    alpha = P(:,j)' * w;
    w = w - alpha * P(:,j);

```

```

end % for j
w = w / norm(w);
end % function

```

The function `orth_mgs` orthogonalizes the columns of a given matrix.

```

function V = orth_mgs(A,dreorth);
% (double) orthogonalisation of the columns of A
if nargin == 1
    dreorth = 'nodreorth';
end
[m,n] = size(A);
V = zeros(m,n);
v = A(:,1);
V(:,1) = v / norm(v);
for k = 2:n
    v = A(:,k);
    for j = 1:k-1
        alpha = v' * V(:,j);
        v = v - alpha * V(:,j);
    end % for j
    if strcmpi(dreorth,'dreorth') == 1
        for j = 1:k-1
            alpha = v' * V(:,j);
            v = v - alpha * V(:,j);
        end % for j
    end % if
    nv = norm(v);
    if nv <= 1e-15
        fprintf('\n orth_mgs: Breakdown, step %d, val=%g \n\n',k,nv)
        %return
    end % if
    v = v / norm(v);
    V(:,k) = v;
end % for k
end % function

```

The function `harm_Ritz_vec` computes the harmonic Ritz vectors. Note that the value of `kR` (the number of harmonic Ritz values) may be increased by 1 to keep pairs of complex conjugate eigenvalues.

```

function [g,kR] = harm_Ritz_vec(kR,HH);
% computation of the harmonic Ritz vectors
if kR == 0
    g = [];
else
    [theta,g] = harm_Ritz_valvec(HH);

```

```
% keep only the first kR ones (with the pairs of
% complex conjugate eigenvalues)
if isreal(theta(kR))
    theta = theta(1:kR);
    g = g(:,1:kR);
else
    if abs(abs(theta(kR)) - abs(theta(kR+1))) <= 1e-14
        kR = kR + 1; % increase kR to keep a pair
        theta = theta(1:kR);
        g = g(:,1:kR);
    else
        theta = theta(1:kR);
        g = g(:,1:kR);
    end % if abs
end % if isreal
% we need real vectors
jj = 1;
for j=1:kR
    if isreal(theta(jj))
        jj = jj + 1;
    else
        g(:,jj+1) = imag(g(:,jj));
        g(:,jj) = real(g(:,jj));
        jj = jj + 2;
    end % if
    if jj > kR
        break
    end
end % for j
end % if kR
end % function
```

The function `harm_Ritz_valvec` computes the harmonic Ritz values and vectors. The matrix H_k is assumed to be nonsingular.

```
function [lambda,g] = harm_Ritz_valvec(H);
%computes the harmonic Ritz values and eigenvectors
% of the upper Hessenberg matrix H
n = size(H,1);
k = n - 1;
Hk = H(1:k,1:k);
ek = zeros(k,1);
ek(k) = 1;
Hki = Hk' \ ek;
Hk(:,k) = Hk(:,k) + H(k+1,k)^2 * Hki;
[X,D] = eig(full(Hk));
```

```

lam = diag(D);
[lamb,I] = sort(abs(lam));
lambda = lam(I);
g = X(:,I);
end % function

```

The function `gmres_iter` does the k th iteration of GMRES-MGS. It is only a slight modification of what we have seen in Chapter 5 for GMRES-MGS.

```

function [V,H,HH,rhs,rhsR,rot] = gmres_iter(k,A,V,H,HH,rhs,
rhsR,rot,rzero,kR,QQ);
% iteration k of GMRES-DR
Av = A * V(:,k); % matrix-vector product
w = Av;
for l = 1:k
    vl = V(:,l);
    gl = vl' * w;
    H(l,k) = gl;
    w = w - gl * vl;
end % for l
dk = w;
gk = norm(dk);
dk1 = dk / gk; % normalization of the new vector
H(k+1,k) = gk;
V(:,k+1) = dk1;
gk1 = gk;
rhs(k+1) = V(:,k+1)' * rzero; % compute the right-hand side
rhsR(k+1) = rhs(k+1); % save it
HH(1:k+1,k) = H(1:k+1,k); % save the column of H
if kR ~= 0
    H(1:kR+1,k) = QQ' * H(1:kR+1,k); % apply the Q factor
    % to the last column
end % if kR
% apply the preceding Givens rotations to the last column
% just computed
for kk = kR+1:k-1
    g1 = H(kk,k);
    g2 = H(kk+1,k);
    H(kk+1,k) = -rot(2,kk) * g1 + rot(1,kk) * g2;
    H(kk,k) = rot(1,kk) * g1 + conj(rot(2,kk)) * g2;
end % for kk
% compute, store and apply a new rotation to zero
% the last term in kth column
gk = H(k,k);
if gk == 0
    rot(1,k) = 0;

```

```

rot(2,k) = 1;
elseif gk1 == 0
  rot(1,k) = 1;
  rot(2,k) = 0;
else
  cs = sqrt(abs(gk1)^2 + abs(gk)^2);
  if abs(gk) < abs(gk1)
    mu = gk / gk1;
    tau = conj(mu) / abs(mu);
  else
    mu = gk1 / gk;
    tau = mu / abs(mu);
  end % if
  % store the rotation for the next columns
  rot(1,k) = abs(gk) / cs; % cosine
  rot(2,k) = abs(gk1) * tau / cs; % sine
end % if gk
% modify the diagonal entry and the right-hand side
H(k,k) = rot(1,k) * gk + conj(rot(2,k)) * gk1;
c = rhs(k);
rhs(k) = rot(1,k) * c;
rhs(k+1) = -rot(2,k) * c;
H(k+1,k) = 0;
end % function

```

In the paper [772] the authors propose a slight modification of GMRES-DR, which is mathematically equivalent, but has better numerical properties. They provide an error analysis of a restart of the algorithm. The differences with the previous algorithm are in the construction of the matrix $P_{m+1,k+1}$ and in the computation of the right-hand side. The paper [697] and the code above use the residual vector of the least squares problem which is orthogonalized against the columns of P whence [772] uses the vector $[-h_{m+1,m}H_m^{-*}e_m, 1]^T$. This is based on the proof that the least squares residual vector $z = c - \underline{H}_m d$ is equal to

$$z = \begin{pmatrix} \omega - h_{m+1,m}[H_m^{-*}e_m]^*[c]_{1:m} \\ 1 + h_{m+1,m}^2[H_m^{-*}e_m]^*H_m^{-*}e_m \end{pmatrix} \begin{pmatrix} -h_{m+1,m}H_m^{-*}e_m \\ 1 \end{pmatrix},$$

where ω is the last component of c . For the computation of c (rhs in the code above) [697] uses $c = V_{n,m+1}^*r$ whence [772] uses $c = [r^*V_{n,k+1}, 0, \dots, 0]^*$. Doing this saves some dot products. Numerically the two implementations differ only if very accurate solutions are required.

In [650], a polynomial preconditioner is added to GMRES-DR. The preconditioner is a polynomial $p(A)$ of degree $d \leq m$ corresponding to a GMRES residual polynomial in the first cycle. The coefficients of the polynomial are computed from

the roots which are the harmonic Ritz values. The polynomial can also be applied using the Arnoldi basis vectors if they are stored or using the roots as we did when constructing Newton bases in Chapter 4.

In [699] pseudo-eigenvectors bases were studied. Pseudo-eigenvectors correspond to pseudo-eigenvalues, see Chapter 5, Section 5.2. The authors show that the approximate eigenvectors generated by GMRES-DR can be viewed as pseudo-eigenvectors. According to their own words, “the convergence is roughly as if the corresponding pseudo-eigenvalues are deflated from the spectrum”. However, there is no theoretical results substantiating that eliminating some pseudo-eigenvalues provides an improved convergence since only upper bounds are given in [699].

We have seen that approximations of the error vectors and approximate eigenvectors have been used to augment the Krylov subspace. Therefore, it was tempting to use both in the same algorithm. This is what is done in [719] with an implementation similar to what was done for LGMRES or GMRES-E.

We now turn to methods using deflation to construct preconditioners. Such techniques were proposed and studied in [163, 322]. The most advanced algorithm in these papers uses right preconditioning. The preconditioner M is defined as

$$M = I_n + U(|\lambda_n|T^{-1} - I_\ell)U^T, \quad T = U^T AU,$$

where U is $n \times \ell$ and its columns are orthonormal vectors spanning an invariant subspace of A and λ_n is the eigenvalue of largest modulus whose approximate value is supposed to be known. The preconditioner is recomputed at the end of each cycle with a value of the integer ℓ smaller than or equal to k that can be changed. If $\ell = k$ the eigenvalues of AM^{-1} are $|\lambda_n|$ of multiplicity k and the other ones are $\lambda_{k+1}, \dots, \lambda_n$.

Let $X = (U, W)$ be a basis of the whole space. Written in this basis, A is similar to

$$\begin{pmatrix} T & A_{1,2} \\ 0 & A_{2,2} \end{pmatrix}. \quad (11.21)$$

The preconditioner M is similar to

$$\begin{pmatrix} \frac{1}{|\lambda_n|}T & 0 \\ 0 & I_{n-k} \end{pmatrix},$$

and AM^{-1} is similar to

$$\begin{pmatrix} |\lambda_n|I_k & A_{1,2} \\ 0 & A_{2,2} \end{pmatrix},$$

proving the statement about the eigenvalues of AM^{-1} . It is proved that, at the end of a preconditioned GMRES(m) cycle, we have

$$\|r_m\| \leq \min_{q \in \pi_m, q(0)=1} \{ |q(|\lambda_n|)| \|r_{0,1}\| + \max_{k+1 \leq i \leq n} |q(\lambda_i)| \|r_{0,2}\| \|Y_2\| \|(Y_2^* Y_2)^{-1} Y_2^*\| \},$$

$r_0 = r_{0,1} + r_{0,2}$, $r_{0,1} = \sum_{i=1}^k \beta_i z_i$, $r_{0,2} = \sum_{i=k+1}^n \beta_i y_i$, where z_i , $i = 1, \dots, k$ are the eigenvectors of A associated with $\lambda_1, \dots, \lambda_k$, y_i , $i = k+1, \dots, n$ are the eigenvectors of AM^{-1} associated with $\lambda_{k+1}, \dots, \lambda_n$ and $Y_2 = (y_{k+1} \cdots y_n)$.

However, we do not know an exact invariant subspace of A and, in practice, the left bottom block in (11.21) is not zero but equal to $A_{2,1}$ and the matrix AM^{-1} is similar to

$$\begin{pmatrix} |\lambda_n| I_k & A_{1,2} \\ |\lambda_n| A_{2,1} T^{-1} & A_{2,2} \end{pmatrix}.$$

Hence, the eigenvalues are what we would like to have only if the block $|\lambda_n| A_{2,1} T^{-1}$ is small in some sense. In the first version [322] of the algorithm Ritz vectors were used whence in [163] harmonic Ritz vectors are used.

Deflation of this type was also considered when constructing more parallel versions of restarted GMRES in [962, 963].

The algorithm in [50] uses the Arnoldi information to determine an approximation of an invariant subspace of A associated with eigenvalues close to the origin with a basis V . The authors use the recurrence formulas of the implicit restarted Arnoldi algorithm (IRA). The preconditioner is applied from the left. If $[V, W]$ is unitary, the matrix $H = V^* AV$ is nonsingular and a preconditioner is defined as

$$M = VH V^* + WW^*,$$

then, the inverse of M is

$$M^{-1} = VH^{-1} V^* + WW^*.$$

Moreover, if the columns of V span an invariant subspace of A associated with the eigenvalues $\lambda_1, \dots, \lambda_k$, the spectrum of $M^{-1} A$ is

$$(\lambda_{k+1} \dots \lambda_n 1 \dots 1).$$

Note that $WW^* = I - VV^*$ which means that there is no need to compute W . The first preconditioner in [50] is $M_1^{-1} = V_{n,k} H_k^{-1} V_{n,k}^* + I - V_{n,k} V_{n,k}^*$. Applying IRA, one can obtain a preconditioner M_2 for the system $M_1^{-1} Ax = M_1^{-1} b$ and this process can be iterated leading to a preconditioner such that

$$M^{-1} = M_\ell^{-1} M_{\ell-1}^{-1} \cdots M_1^{-1}.$$

11.4 Restarted and augmented CMRH and Q-OR Opt

Other methods using long recurrences, like CMRH and Q-OR Opt, have also to be restarted for solving practical problems. The standard way of doing this is, as for FOM and GMRES, to restart from the current approximate solution. The modifications of the codes are similar to what we did for GMRES.

An augmented CMRH algorithm is introduced in [908]. It is in the same spirit as LGMRES but only one “error” vector is used. However, it is easily seen that more vectors can be appended successively. Approximate eigenvectors can also be used.

11.5 Truncated FOM and GMRES

We have already seen a truncated Krylov method in Chapter 6 since Orthomin(d) [953] can be seen as a truncated version of the generalized conjugate residual (GCR) method. In this section we are interested in truncated versions of FOM and GMRES.

The incomplete orthogonalization method (IOM) was introduced in [789]. It uses the truncated Arnoldi process, see Chapter 4, Section 4.7, relation (4.37), to compute the basis vectors. They are computed as

$$\tilde{v} = Av_k - \sum_{i=i_0}^k h_{i,k} v_i, \quad h_{k+1,k} = \|\tilde{v}\|, \quad v_{k+1} = \frac{1}{h_{k+1,k}} \tilde{v},$$

and $i_0 = \max(1, k - m + 1)$. The upper bandwidth of the size k matrix H_k is m .

In the original IOM(m) method, at iteration k , the linear system $H_k y^{(k)} = \|r_0\| e_1$ is solved using an LU factorization with partial pivoting and $x_k = x_0 + V_{n,k} y^{(k)}$. This is a FOM-like Q-OR method since the basis is only locally orthogonal.

The algorithm in [792] is based on updating the LU factorization $H_k = L_k U_k$ where L_k is lower bidiagonal with ones on the diagonal and subdiagonal entries $\ell_i = L_{i,i-1}$, $i = 2, \dots, k$. Using the LU factorization, we have

$$x_k = x_0 + \beta V_{n,k} U_k^{-1} L_k^{-1} e_1,$$

with $\beta = \|r_0\|$. We define $P_{n,k} = V_{n,k} U_k^{-1}$ and $z_k = \beta L_k^{-1} e_1$. It is straightforward to see that

$$z_k = \begin{pmatrix} z_{k-1} \\ \xi_k \end{pmatrix}, \quad \xi_k = \ell_k \xi_{k-1}, \quad k = 2, \dots, \quad \xi_1 = \beta.$$

Hence, the vectors z_j are easily updated. Then, $x_k = x_{k-1} + \xi_k p_k$ where p_k is the last column of $P_{n,k}$. The vector p_k is obtained by using the last columns in the relation $P_{n,k} U_k = V_{n,k}$ and the fact that U_k is banded. It yields

$$p_k = \frac{1}{u_{k,k}} \left\{ v_k - \sum_{i=i_0}^{k-1} u_{i,k} p_i \right\}.$$

In this relation, we need only the last column of U_k . The residual norm is given by $h_{k+1,k}|\xi_k|/|u_{k,k}|$. We have a breakdown of the algorithm if $u_{k,k} = 0$. The algorithm in [792] uses partial pivoting for stability reasons. Then, U_k has one more nonzero diagonal because we possibly have to permute two rows of H_k . If there is a swap of the rows $k - 1$ and k , we have

$$z_k = \begin{pmatrix} z_{k-2} \\ 0 \\ \xi_{k-1} \end{pmatrix},$$

and $x_k = x_{k-2} + \xi_{k-1}w_k$. For details, see [792]. One can also use a QR factorization of $H_k = Q_k U_k$ but the solution described in [792] is not correct. We will see how this can be done after considering truncated GMRES.

We have described the results for the IOM basis vectors obtained in [565] in Chapter 4, Section 4.7. Roughly speaking, the convergence of IOM is similar to that of FOM if the basis stays well conditioned. However, when the matrix A is strongly nonsymmetric, the basis vectors may even lose linear independence.

A GMRES algorithm, named IGMRES, using a truncated Arnoldi basis was proposed in [153]. Quasi-GMRES (QGMRES), described in [803], is just standard GMRES with a truncated Arnoldi recurrence as described above. This is similar to IGMRES. Direct quasi-GMRES (DQGMRES), proposed in [803], is a better implementation of the truncated version of GMRES where x_k is updated from x_{k-1} using the fact that the $(k + 1) \times k$ upper Hessenberg matrix \underline{H}_k is banded.

This matrix is transformed to upper triangular form by applying the product of rotation matrices $Q_{k+1}^* = G_k \cdots G_1$ (see Chapter 2, Section 2.9),

$$Q_{k+1}^* \underline{H}_k = \begin{pmatrix} \hat{R}_k \\ 0 \end{pmatrix}.$$

The vector y is obtained by solving $\hat{R}_k y = \beta[Q_{k+1}^* e_1]_{1:k}$ and, as in GMRES, the approximate solution is $x_k = x_0 + V_{n,k}y$. The upper triangular matrix \hat{R}_k is banded with $m + 1$ nonzero diagonals. Let $\hat{q}_{k+1} = Q_{k+1}^* e_1$. We can use the construction of the approximate solution that we described in Chapter 2, Section 2.9. We have

$$y = \beta \begin{pmatrix} \hat{R}_{k-1}^{-1} & -\frac{1}{\hat{r}_{k,k}} \hat{R}_{k-1}^{-1} [\hat{R}_k]_{1:k-1,k} \\ 0 & -\frac{1}{\hat{r}_{k,k}} \end{pmatrix} \begin{pmatrix} \hat{q}_k \\ [\hat{q}_{k+1}]_k \end{pmatrix}.$$

Hence,

$$y = \beta \begin{pmatrix} \hat{R}_{k-1}^{-1} \\ 0 \end{pmatrix} \hat{q}_k + \beta [\hat{q}_{k+1}]_k [\hat{R}_k^{-1}]_{:,k}.$$

It yields

$$\begin{aligned} x_k &= x_0 + \beta V_{n,k-1} \hat{R}_{k-1}^{-1} \hat{q}_k + \beta [\hat{q}_{k+1}]_k V_{n,k} [\hat{R}_k^{-1}]_{:,k}, \\ &= x_{k-1} + \beta [\hat{q}_{k+1}]_k V_{n,k} [\hat{R}_k^{-1}]_{:,k}. \end{aligned}$$

We define $P_{n,k} = V_{n,k} \hat{R}_k^{-1}$. Then,

$$x_k = x_{k-1} + \beta [\hat{q}_{k+1}]_k P_k,$$

where p_k is the last column of $P_{n,k}$. What is interesting for truncated GMRES is that this vector can be computed using $P_{n,k} \hat{R}_k = V_{n,k}$ and we can exploit the fact that \hat{R}_k has bandwidth $m + 1$. Therefore,

$$p_k = \frac{1}{\hat{r}_{k,k}} \left\{ v_k - \sum_{i=\max(1, k-m+1)}^{k-1} \hat{r}_{i,k} p_i \right\},$$

$\hat{r}_{i,j}$ being the entries of \hat{R}_k and $p_1 = v_1 / \hat{r}_{1,1}$. We have a short recurrence for the vectors p_k and we need only to store the m last vectors. Therefore, in a code, the practical indexing has to be different for p_i in the formula above. Note that introducing the vectors p_k is not interesting when the Arnoldi relation is not truncated because, in that case, we have also a long recurrence for computing p_k .

The properties of DQGMRES are studied in [803] but they can also be obtained from what we described in Chapter 3 for general Q-MR methods. In particular, we can obtain bounds for the residual norms. As for IOM, the truncated versions of GMRES may have problems if the basis obtained by truncation is badly conditioned, see [566].

Now that we have seen how to update the DQGMRES approximate solution, we can return to IOM. We use the result of Proposition 2.3 relating the coefficients of the Q-OR and Q-MR methods. To avoid confusion, let us denote by y_k^O the solution of $R_k y_k^O = q_k$ with $q_k = (G_{k-1} \cdots G_1)^* e_1$ and by y_k^M the coefficient vector for the Q-MR method. We have

$$y_k^O = \begin{pmatrix} y_{k-1}^M \\ 0 \end{pmatrix} + \beta [q_k]_k \frac{\hat{r}_{k,k}}{r_{k,k}} [\hat{R}_k^{-1}]_{:,k}.$$

Therefore, assuming that the initial vector x_0 is the same in both methods, the IOM approximate solution is

$$x_k^O = x_{k-1}^M + \beta [q_k]_k \frac{\hat{r}_{k,k}}{r_{k,k}} p_k.$$

Note that $r_{k,k}$ and $[q_k]_k$ are the diagonal entry and the last component of the right-hand side before the last rotation G_k is applied. We have seen above how x_{k-1}^M can be updated.

Restarted variants of the truncated method are proposed in [988]. Several strategies to decide when to restart are discussed in that paper. Numerical experiments show that restarting could be beneficial to improve the maximal attainable accuracy.

11.6 Truncated Q-OR

We can also use the Q-OR truncated bases that we have constructed in Chapter 4, Section 4.7. We have the relation

$$AV_{n,k} = \underline{S}_k [I + R_k]^{-1},$$

where \underline{S}_k is a banded upper Hessenberg with upper bandwidth m_S and R_k is a strictly upper triangular matrix with a bandwidth m_R . Let us first consider a Q-MR method. We have to minimize

$$\|\beta e_1 - \underline{S}_k [I + R_k]^{-1} y\| = \|\beta e_1 - \underline{S}_k z\|,$$

with $\beta = \|r_0\|$ and $z = [I + R_k]^{-1} y$. We use Givens rotations to bring \underline{S}_k to upper triangular form. As in the previous section and with the same notation, we have

$$z = \beta \begin{pmatrix} \hat{R}_{k-1}^{-1} & -\frac{1}{\hat{r}_{k,k}} \hat{R}_{k-1}^{-1} [\hat{R}_k]_{1:k-1,k} \\ 0 & -\frac{1}{\hat{r}_{k,k}} \end{pmatrix} \begin{pmatrix} \hat{q}_k \\ [\hat{q}_{k+1}]_k \end{pmatrix}.$$

We have to multiply by $I + R_k$ to obtain y . Multiplying by $V_{n,k}$ and adding x_0 we obtain

$$x_k^M = x_{k-1}^M + \beta [\hat{q}_{k+1}]_k p_k,$$

where the vectors p_k are the columns of the matrix $P_{n,k}$ defined by

$$P_{n,k} = V_{n,k} [I + R_k] \hat{R}_k^{-1}.$$

Hence, the only thing that changes compared with truncated GMRES is how the vectors p_k are computed. Using $P_{n,k} \hat{R}_k = V_{n,k} [I + R_k]$, p_k is given by

$$p_k = \frac{1}{\hat{r}_{k,k}} \left\{ \sum_{i=\max(1,k-m_R+1)}^k r_{i,k} v_i - \sum_{i=\max(1,k-m)}^{k-1} \hat{r}_{i,k} p_i \right\},$$

where $r_{i,j}$ are the entries of R_k .

For the Q-OR method we have to solve

$$S_k [I + R_k]^{-1} y = \beta e_1.$$

As in the case of IOM, we obtain

$$x_k^O = x_{k-1}^M + \beta [q_k]_k \frac{\hat{r}_{k,k}}{\tilde{r}_{k,k}} p_k,$$

where $\tilde{r}_{k,k}$ is the entry of the upper triangular matrix which is obtained from S_k , that is, before the last rotation G_k is applied to eliminate the $(k+1, k)$ entry of S_k .

As for truncated FOM and GMRES, that is, IOM and DQGMRES, it can be beneficial to restart since the basis vectors may possibly lose their numerical independence.

11.7 Parallel computing

Regarding parallel computing there is no difference from the full methods for the truncated versions. For the augmented or deflated methods using approximate eigenvectors, the computation of the (harmonic) Ritz vectors must be parallelized.

11.8 Finite precision arithmetic

As far as we know, there is no study of the augmented, deflated or truncated methods in finite precision arithmetic. Mathematically, some of these bases can become badly conditioned, see [565, 566]. Obviously, this cannot be better in finite precision arithmetic. Since the restarted methods throw away previous information, using only the last iterate and residual vector, they tend to suffer less from rounding errors than the full methods.

11.9 Numerical experiments

In the following experiments the variant of GMRES that is used is GMRES-MGS with modified Gram–Schmidt implementation starting from $x_0 = 0$. We plot the residual norms as functions of the iteration number since there is only one matrix–vector product per GMRES(m) iteration whatever the value of m is.

11.9.1 Restarted GMRES without preconditioning

Figure 11.1 displays the relative true residual norms for full GMRES and restarted GMRES(m) for different values of m and the problem `fs_183_6`. For this example

full GMRES exhibits a fast convergence. It is not always the same for GMRES(m) since for $m \leq 25$ it does not converge in 200 iterations, almost stagnating after the first restart. For $m = 30$ we see some convergence but it is very slow. Increasing m improves the convergence and also gives a better maximal attainable accuracy than with full GMRES. In fact, restarting at iteration 50, that is, when full GMRES starts its final (almost) stagnation, improves the maximal attainable accuracy.

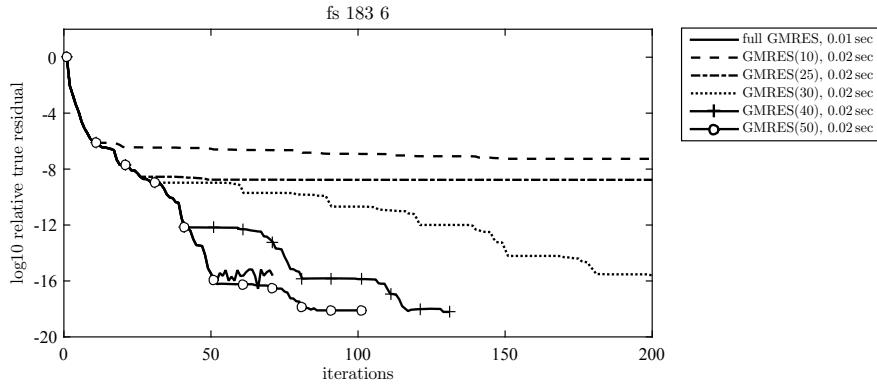


Fig. 11.1 fs 183 6, relative true residual norms, GMRES-MGS(m), different values of m

For the problem **fs 680 1c** GMRES(m) does not converge for $m < 22$. But for $m = 22$ we observe some convergence even though it is very slow at the beginning, see Figure 11.2. Increasing m to 40, 60 or 80 improves the convergence and the maximal attainable accuracy which is much better than with full GMRES.

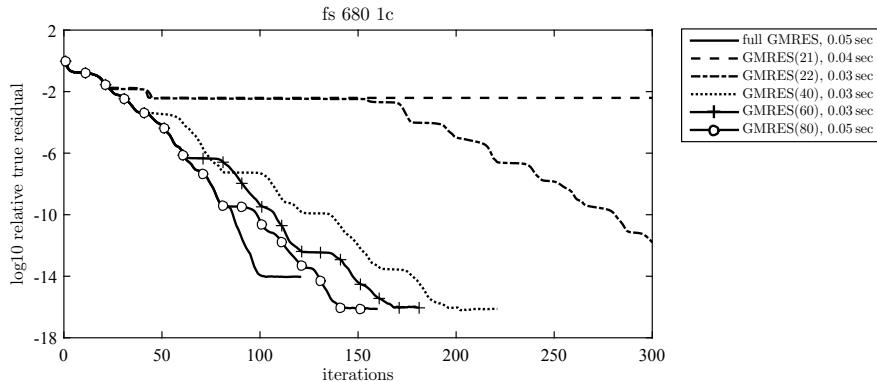


Fig. 11.2 fs 680 1c, relative true residual norms, GMRES-MGS(m), different values of m

Figure 11.3 displays the results for `sugp 001` of order 1225. Full GMRES shows a slow convergence for the first 34 iterations and then the residual norms are decreasing fast. GMRES(m) shows an interesting behavior. GMRES(20) and GMRES(30) converge very slowly. Things start to be better for $m > 34$ when GMRES(m) has been able to “see” the beginning of the fast decrease of full GMRES. We observe that, whatever the value of $m > 34$, there is a quasi-stagnation phase after each restart even though this is not completely true for $m = 50$ since there is some quasi-stagnation before the second restart.

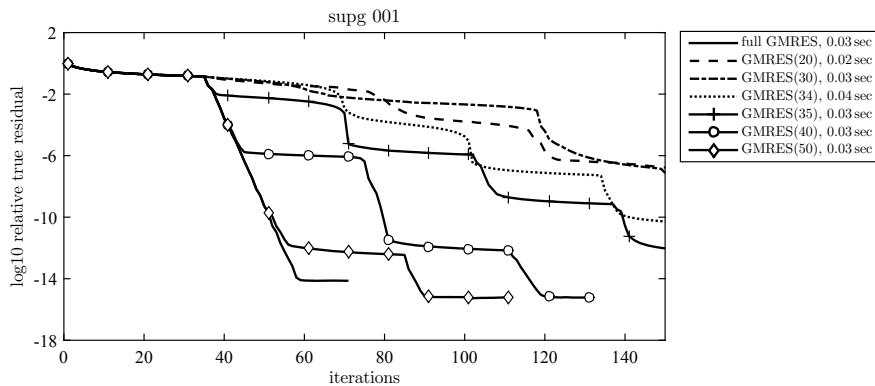


Fig. 11.3 `sugp 001`, relative true residual norms, GMRES-MGS(m), different values of m

In the previous examples we had to use large values of m to obtain convergence but, fortunately, this is not always the case. Figure 11.4 shows the residual norms for the problem `diff conv 400`. Even though the convergence is slow for $m = 1$, using $m = 2$ already improves convergence and the values $m = 5, 10$ or 20 give much better results. Note that the maximal attainable accuracies are not much improved compared to that of full GMRES.

Restarting in GMRES was introduced mainly to save memory space but also to avoid the increase in computing time for each iteration due to the orthogonalization process. Hence, it is interesting to see how the GMRES(m) residual norms behave as functions of the computing time. We use the problem `raefsky1` to illustrate this. Figure 11.5 shows the relative true residual norms as function of the iteration number. We observe that we have to use large values of m to obtain a good convergence with GMRES(m). Unfortunately this does not provide much savings in computer storage.

Figure 11.6 displays the residual norms as functions of the computing time. GMRES(160) and GMRES(200) are competitive with full GMRES but use less memory space.

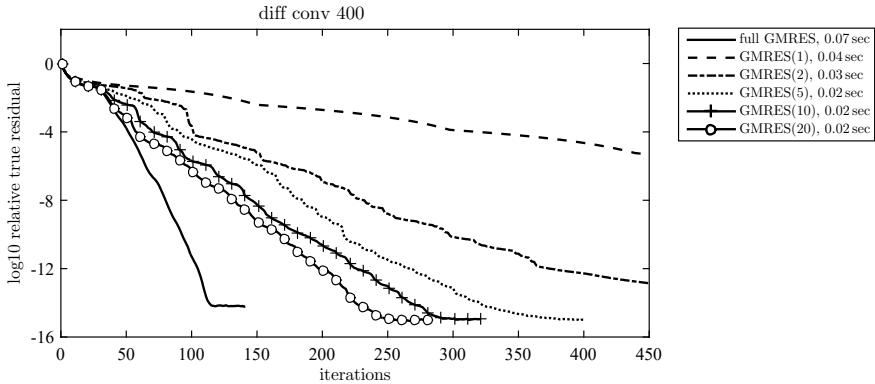


Fig. 11.4 diff conv 400, relative residual norms, GMRES-MGS(m), different values of m

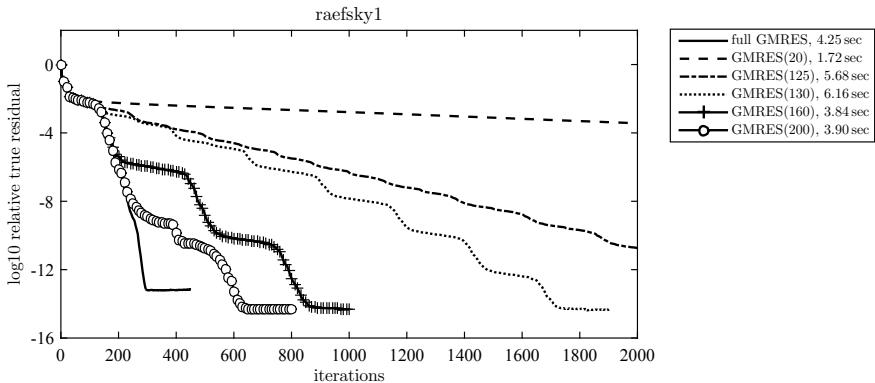


Fig. 11.5 raefsky1, relative true residual norms, GMRES-MGS(m), different values of m

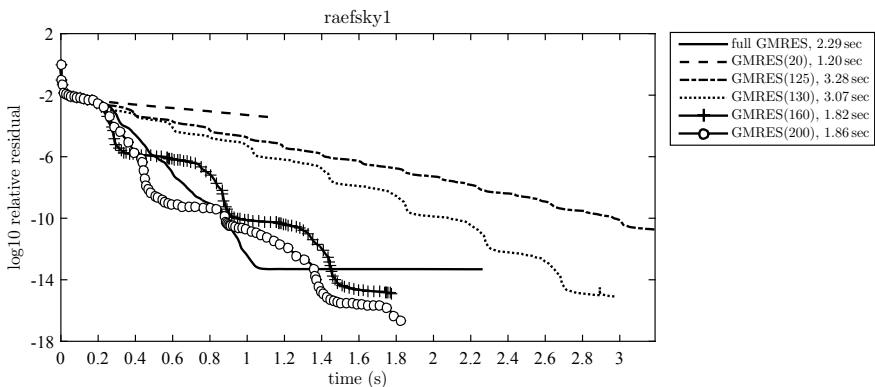


Fig. 11.6 raefsky1, relative residual norms vs time, GMRES-MGS(m), different values of m

11.9.2 Restarted GMRES with preconditioning

Figures 11.7–11.8 show the residual norms for a larger problem than the examples above. This is the problem `supg 001` but with 150 interior discretization points in each direction yielding a matrix of order 22500. It is difficult to solve this linear system without preconditioning. We use an SSOR preconditioner (see, for instance, [673]) with a relaxation parameter equal to 1 because it is very cheap to compute.

Curiously enough GMRES(2) converges better than for some other values of m we used, that is, 10 and 30. For $m = 65$ and 70 there are phases of quasi-stagnation after the restarts. Therefore, as one can see in Figure 11.8, GMRES(2) is much faster than the other variants. The norm of the final error with GMRES(2) is $4.4909 \cdot 10^{-14}$ indicating that the linear system was correctly solved.

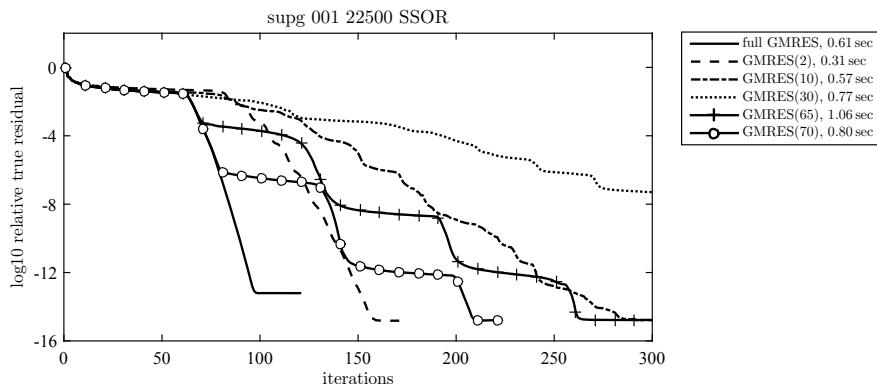


Fig. 11.7 supg 001 22500, relative true residual norms, GMRES-MGS(m) with SSOR preconditioning, different values of m

For this problem the incomplete LU factorization without fill-in ILU(0) (see, for instance, [673]) is a better preconditioner than SSOR as we can observe in Figures 11.9–11.10. As with SSOR, GMRES(2) converges faster than the other methods.

With the matrix `rajat27b` of order 20640 the results are more like what can be expected. Convergence is faster when we increase the value of m ; see Figures 11.11–11.12.

Figures 11.13–11.14 display the residual norms for the problem `matrix-new 3` of order 125329 with the ILU(0) preconditioner. Figure 11.15 is a zoom for $t \leq 3$. We observe that if the stopping criterion is not too stringent, it is beneficial to use GMRES(m) with $m = 10 - 40$. It saves a lot of memory compared to full GMRES. If we stop with $\|r_k\| \leq 10^{-8} \|b\|$, full GMRES needs 35 iterations, the true residual norm at the end is $2.5548 \cdot 10^{-7}$ and the relative difference with the solution computed

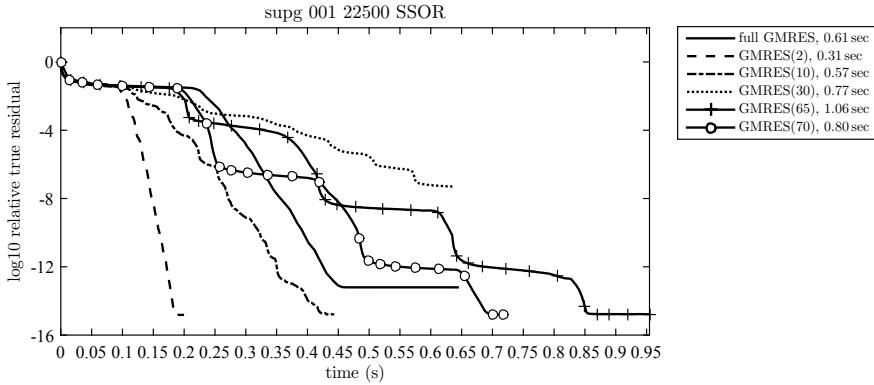


Fig. 11.8 supg 001 22500, relative true residual norms vs time, GMRES-MGS(m) with SSOR preconditioning, different values of m

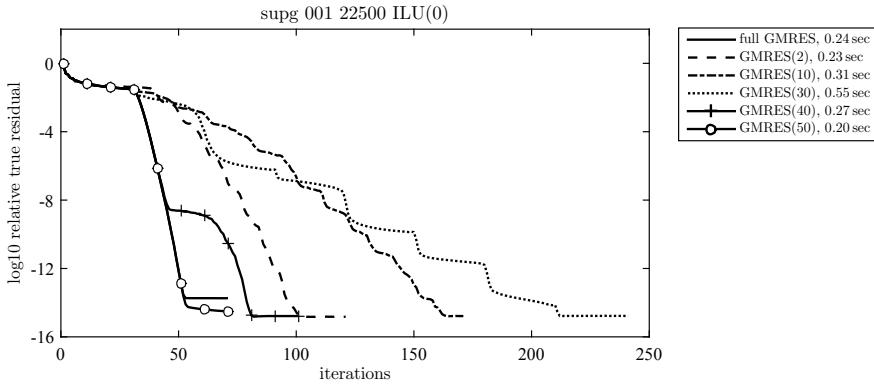


Fig. 11.9 supg 001 22500, relative true residual norms, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

as $A \backslash b$ is $1.4069 \cdot 10^{-3}$. With GMRES(10) we do 41 iterations, the residual norm is $2.6875 \cdot 10^{-7}$ and the relative difference is $1.2442 \cdot 10^{-3}$ but we have to store only 10 vectors of length 125329 instead of 35. The computing time for full GMRES is 0.97 seconds whence it is 0.7 seconds for GMRES(10). The solution with the \ operator needs 6 seconds and the residual norm is $6.5476 \cdot 10^{-17}$.

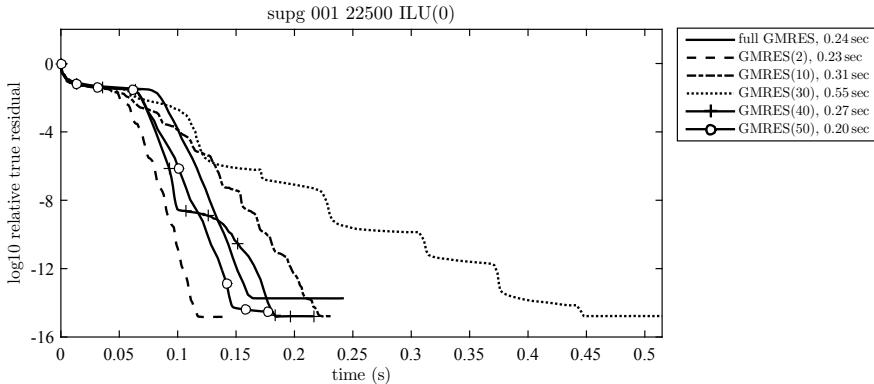


Fig. 11.10 supg 001 22500, relative true residual norms vs time, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

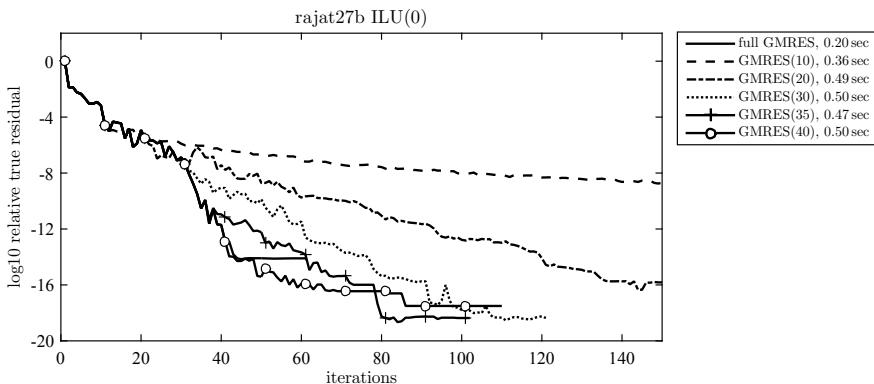


Fig. 11.11 rajat27b, relative true residual norms, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

11.9.3 Restarting with deflation and augmentation without preconditioning

Figure 11.16 displays the relative residual norms for the problem fs 680 1c solved with full GMRES, GMRES(35), GMRES-DR(35,10) which means that at each restart we use 10 harmonic Ritz vectors corresponding to the 10 harmonic Ritz values smallest in magnitude, GMRES-DEF(35,10) using the preconditioner proposed in [163, 322] and DQGMRES(35) the truncated version of GMRES. We choose to compare methods using subspaces of the same dimension. The number of iterations corresponds to the number of matrix–vector products. It can be seen that GMRES-DR gives the best results (apart from full GMRES) even though the maximum attainable accuracy is far from being the best one. The deflation preconditioner does

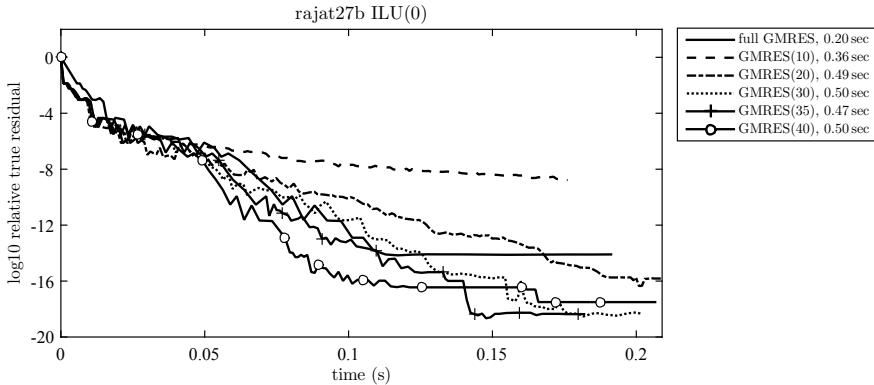


Fig. 11.12 rajab27b, relative true residual norms vs time, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

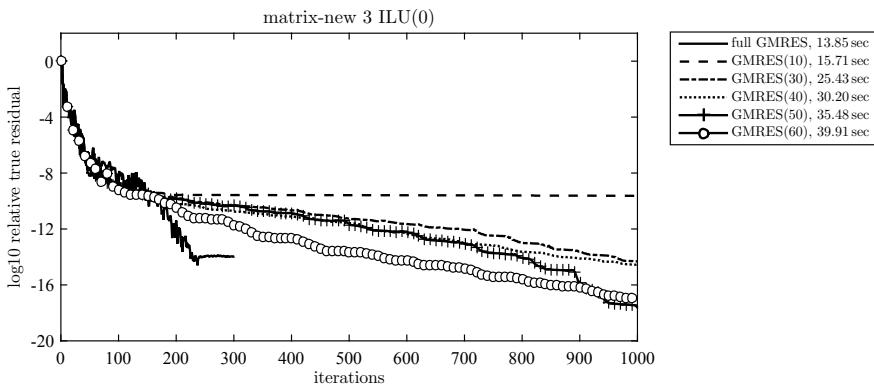


Fig. 11.13 matrix-new 3, relative true residual norms, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

not improve convergence over restarted GMRES and DQGMRES convergence is slower.

In Figure 11.17 we compare full GMRES, GMRES(40) and GMRES-DR(m, p) with $m = 40$ and different values of p . Note that, in our implementation, when we increase p , we decrease the number of Arnoldi iterations per cycle. Using more approximate eigenvectors improves the convergence but worsens the maximum attainable accuracy compared to GMRES(40).

For the experiments in Figure 11.18 we increase both m and p in such a way that we are always doing 40 Arnoldi iterations per cycle whatever is the value of p . The last experiment with $m = 60$ and $p = 20$ is, of course, very close to full GMRES which converges in about 100 iterations.

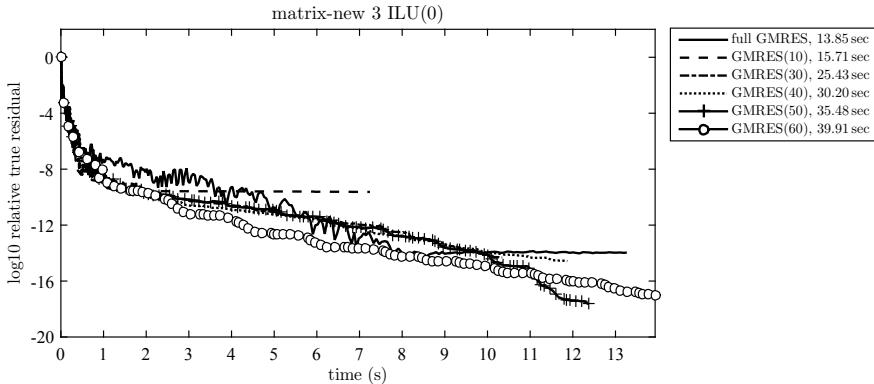


Fig. 11.14 matrix-new 3, relative true residual norms vs time, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

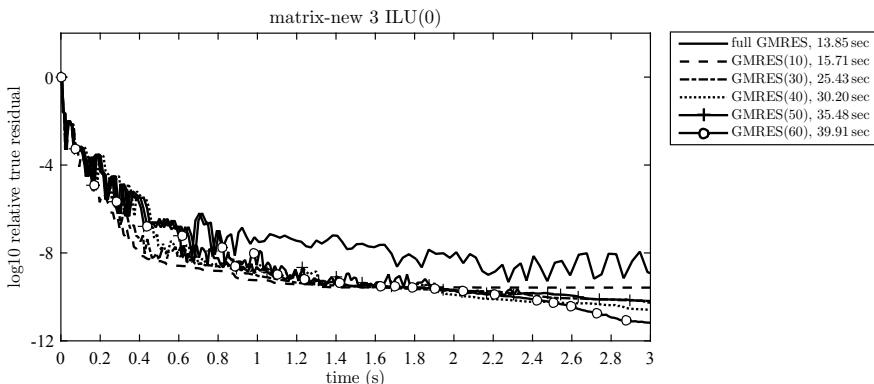


Fig. 11.15 matrix-new 3, relative true residual norms vs time, GMRES-MGS(m) with ILU(0) preconditioning, different values of m

It has been argued that GMRES-DR can improve convergence because it “removes” the influence of the eigenvalues close to the origin. Figure 11.19 shows the eigenvalues of A and the harmonic Ritz values at iterations 20, 25, 30, 35 and 40 of GMRES. The matrix A has only 18 non-real eigenvalues and the real parts of all the eigenvalues are positive. The eigenvalue of smallest magnitude is $3.0258 \cdot 10^{-2}$ which is not really close to the origin. The smallest distance of the harmonic Ritz values to the smallest eigenvalue is of order 10^{-3} . The approximate eigenvectors are also not close to any eigenvectors of A , the minimum distances being of order 1 and the angles between eigenvectors and harmonic Ritz vectors being all larger than 25 degrees. However, if we look at the perturbation result of Proposition 11.2 and consider the norm of the perturbation matrix E for $m = 40$ and $p = 5$ (which is changed to $p = 6$ to have complex conjugate pairs of harmonic Ritz values) harmonic Ritz vectors, at the end of the first cycle we have

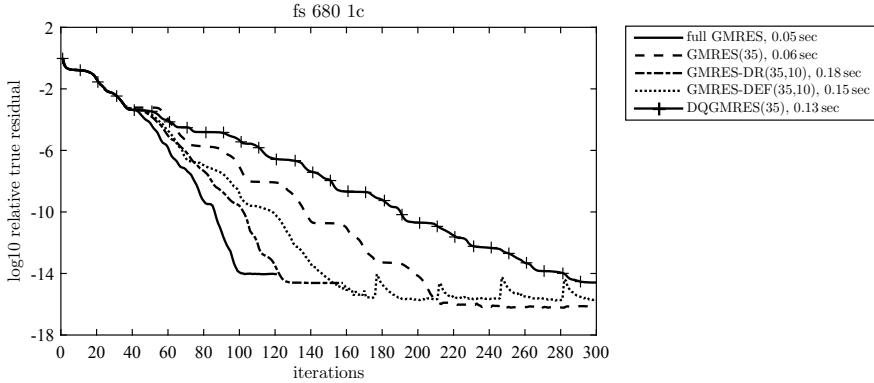


Fig. 11.16 fs 680 1c, relative true residual norms, full GMRES and GMRES(m) with different restarting techniques

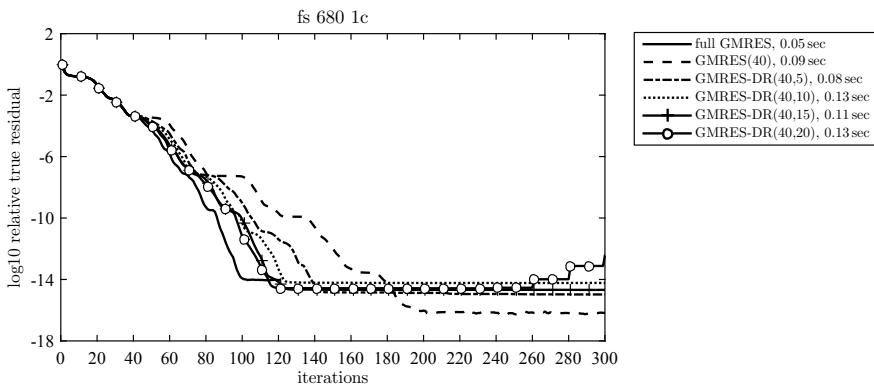


Fig. 11.17 fs 680 1c, relative true residual norms, full GMRES, GMRES(40) and GMRES-DR(40) with different number of harmonic Ritz vectors

$$\|r_m^{(1)}\| = 10^{-2}, \quad \|E\| \approx 10^{-3}.$$

For the next cycles, we obtain

$$\text{2nd cycle, } \|r_m^{(2)}\| = 1.8 \cdot 10^{-6}, \quad \|E\| \approx 10^{-4},$$

$$\text{3rd cycle, } \|r_m^{(3)}\| = 4.8 \cdot 10^{-10}, \quad \|E\| \approx 10^{-6} \text{ for two eigenvectors,}$$

$$\text{4th cycle, } \|r_m^{(4)}\| = 7.5 \cdot 10^{-13}, \quad \|E\| \approx 4 \cdot 10^{-8} \text{ for one eigenvector.}$$

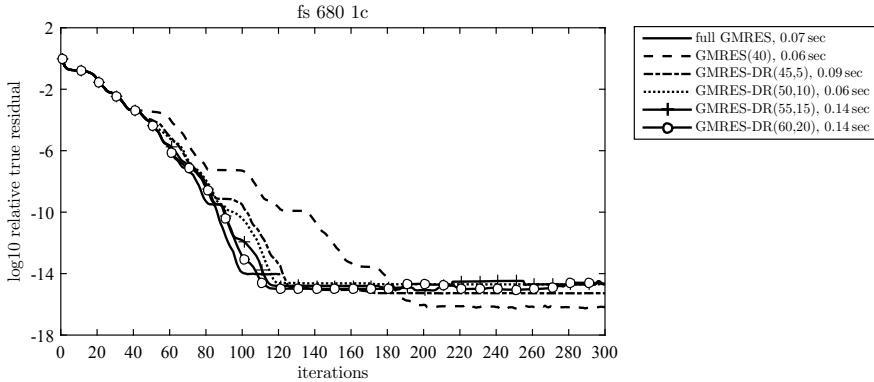


Fig. 11.18 fs 680 1c, relative true residual norms, full GMRES, GMRES(40) and GMRES-DR(m) with different number of harmonic Ritz vectors

The norm of E has to be compared with the norm of A which is 3.817. We see that the norm of the perturbation matrix decreases. In this sense, one or several approximate eigenvectors can be considered as exact eigenvectors of a matrix which is not “too far” from A . They can be considered as pseudo-eigenvectors; see [699]. This can explain, to some extent, the improvements that can be obtained compared to the standard restarted GMRES since the vectors that are used to augment the Krylov subspace are related to a “small” perturbation of A . For this example, using more approximate eigenvectors does not improve the results too much.

Figure 11.20 displays the relative residual norms for GMRES(35), GMRES-DR(35,5), LGMRES(30,5) and HBGMRES(34). LGMRES and HBGMRES do not provide an improvement of convergence on this example. In Figure 11.21 we increase the value of m but we observe the same kind of results.

We now turn to `supg_001` of order 1225. Figure 11.22 displays the relative residual norms for full GMRES, GMRES(35), GMRES-DR(35,10), GMRES-DEF(35,10) and DQGMRES(35). We observe that GMRES-DR and GMRES-DEF give almost similar results and that, contrary to the previous example, the truncated DQGMRES convergence is better than that of restarted GMRES even though there are some plateaus in the convergence curve.

With Figure 11.23 we compare full GMRES, GMRES(40) and GMRES(40, p) with different values of p . Using more approximate eigenvectors improves the convergence but worsens a little the maximum attainable accuracy.

For the experiments in Figure 11.24 we increase both m and p in such a way that we are always doing 40 Arnoldi iterations per cycle whatever is the value of p .

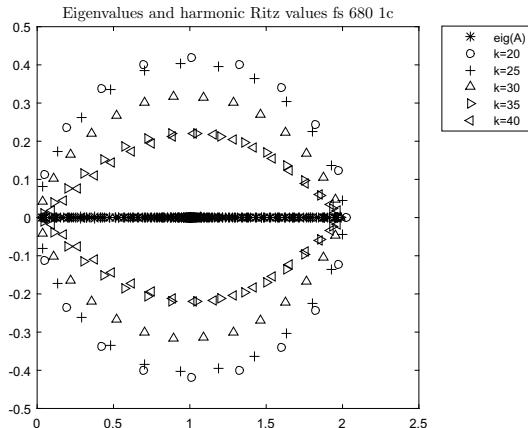


Fig. 11.19 fs 680 1c, eigenvalues and harmonic Ritz values, iterations 20:5:40

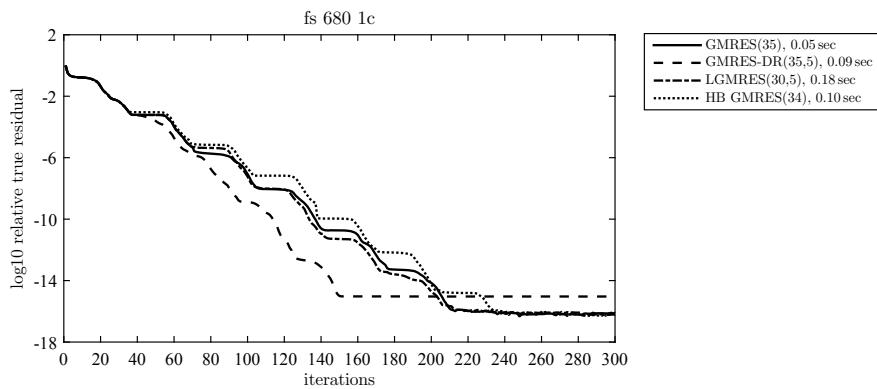


Fig. 11.20 fs 680 1c, relative true residual norms, GMRES(35), GMRES-DR(35,5), LGMRES(30,5) and HBGMRES(34)

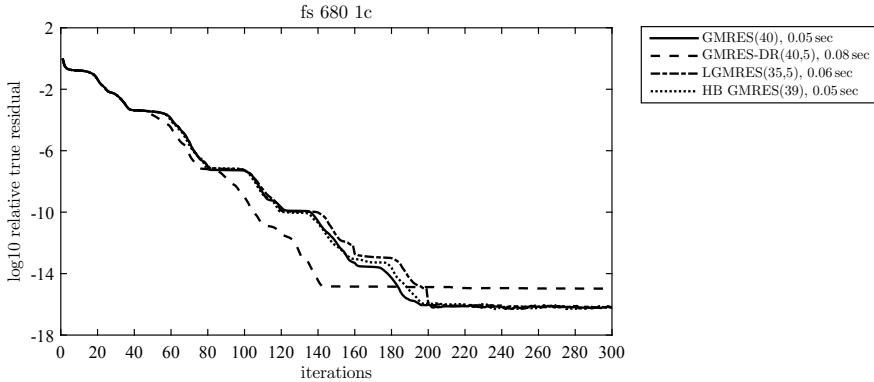


Fig. 11.21 fs 680 1c, relative true residual norms, GMRES(40), GMRES-DR(40,5), LGMRES(35,5) and HBGMRES(39)

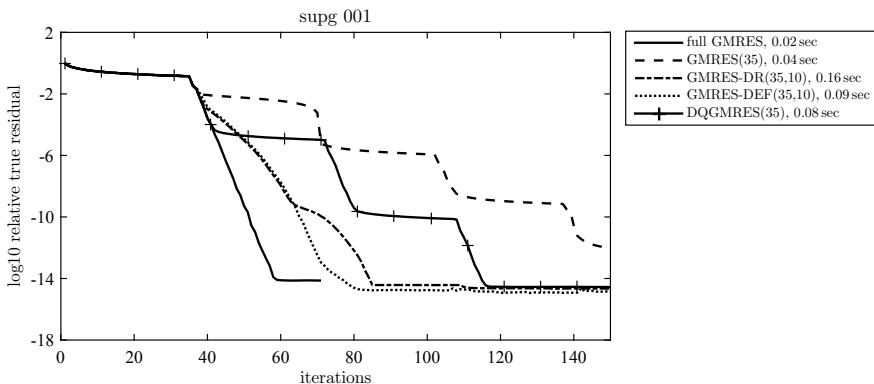


Fig. 11.22 supg 001, $n = 1225$, relative residual norms, full GMRES and GMRES(m) with different restarting techniques

Figure 11.25 shows the eigenvalues of A and the harmonic Ritz values at iterations 20, 25, 30, 35 and 40 of GMRES. The eigenvalues of A are far from the origin but we have seen that we have an initial phase of quasi-stagnation for GMRES which cannot be explained by eigenvalues of small magnitude. The harmonic Ritz values are quite far from any eigenvalue of A . The minimum distance is of order 10^{-2} . The minimum angle between the harmonic Ritz vectors and the eigenvectors is larger than 35 degrees. The approximate eigenvectors are not close to any eigenvector of A . Again, if we look at the perturbation result of Proposition 11.2 and consider the norm of the perturbation matrix E for $m = 40$ and $p = 5$ harmonic Ritz vectors, at the end of the first cycle we have

$$\|r_m^{(1)}\| = 1.2 \cdot 10^{-5}, \quad \|E\| \simeq 2 \cdot 10^{-6}.$$

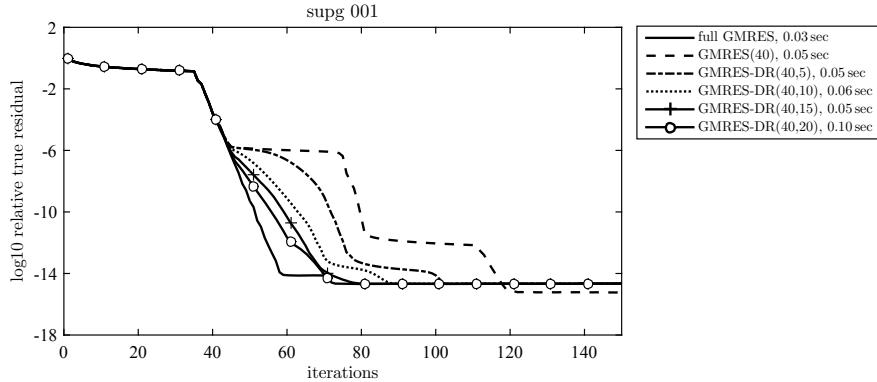


Fig. 11.23 supg001, $n = 1225$, relative true residual norms, full GMRES, GMRES(40) and GMRES-DR(40) with different number of harmonic Ritz vectors

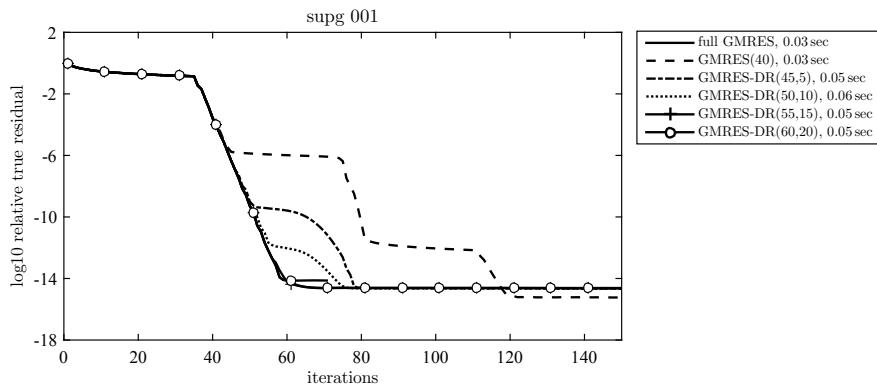


Fig. 11.24 supg001, $n = 1225$, relative true residual norms, full GMRES, GMRES(40) and GMRES-DR(m) with different number of harmonic Ritz vectors

For the next cycle, we obtain

$$\text{2nd cycle, } \|r_m^{(2)}\| = 3.2 \cdot 10^{-14}, \quad \|E\| \simeq 10^{-9}.$$

Using 10 approximate eigenvectors improves the results for the second cycle,

$$\text{2nd cycle, } \|r_m^{(2)}\| = 1.4 \cdot 10^{-14}, \quad \|E\| \simeq 1.6 \cdot 10^{-11} \text{ for one eigenvector.}$$

The norm of A is $5.5456 \cdot 10^{-2}$. The conclusions are the same as for the previous example.

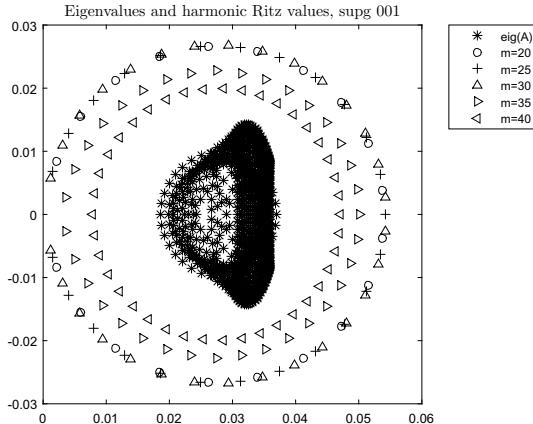


Fig. 11.25 supg 001, $n = 1225$, eigenvalues and harmonic Ritz values, iterations 20:5:40

Figure 11.26 displays the relative residual norms for GMRES(35), GMRES-DR(35,5), LGMRES(30,5) and HBGMRES(34). LGMRES and HBGMRES do not provide an improvement of convergence on this example compared to restarted GMRES. In fact, the results are even worse than with restarted GMRES (with the same dimension of subspaces). In Figure 11.27 we increase the value of m , in which case the results are better than with restarted GMRES. The convergence curve of GMRES-DR does not show the quasi-stagnation phases that we observe for the other methods.

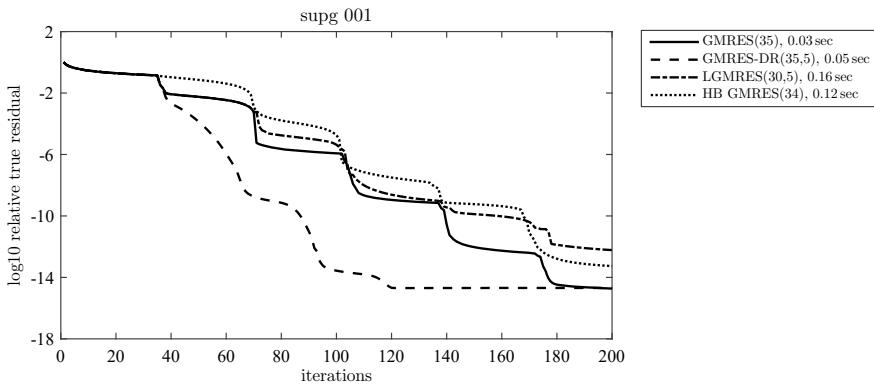


Fig. 11.26 supg 001, $n = 1225$, relative true residual norms, GMRES(35), GMRES-DR(35,5), LGMRES(30,5) and HBGMRES(34)

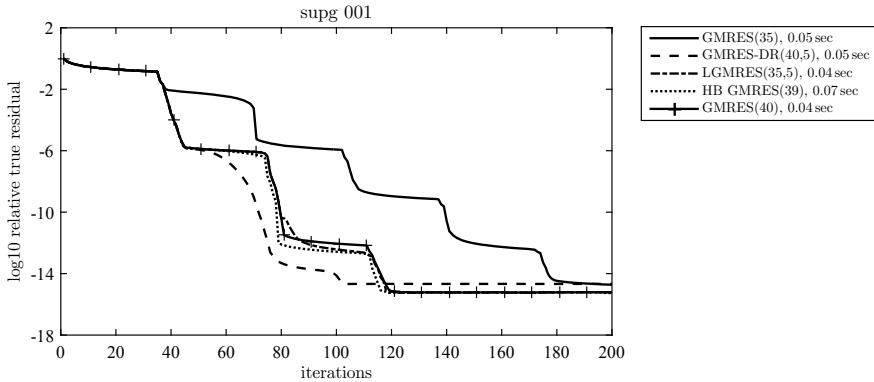


Fig. 11.27 supg 001, $n = 1225$, relative true residual norms, GMRES(35), GMRES-DR(40,5), LGMRES(35,5), HBGMRES(39) and GMRES(40)

In Table 11.1 we give the number of iterations needed to obtain $\|r_k\| \leq 10^{-6}\|b\|$ for our set of small matrices with GMRES(m). A dash indicates that the stopping criterion has not been satisfied within $10n$ iterations. There are many problems without convergence or with a large increase of the number of iterations when compared to full GMRES. For most problems, increasing m improves convergence. Note that many of these matrices are badly conditioned; see Appendix A.

Table 11.1 GMRES(m), $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	54	95	100	79	69	63	54
gre 343	216	—	—	—	—	—	—
jpwh 991	45	92	63	47	46	45	45
pde2961	188	375	340	324	376	374	542
jagmesh1	164	6562	3261	2225	1573	1286	849
bfgwa782	203	—	2297	1356	699	686	506
dw2048	406	3995	1940	1396	1219	1066	891
jagmesh2	448	7715	3803	2476	1909	1557	1292
raefsky2	315	3814	2494	2454	2426	2216	2166
fs 680 1c	60	—	—	115	72	76	60
add20	210	1829	674	530	380	289	307
raefsky1	197	10298	6086	2480	2514	1931	1824
jagmesh4	554	—	—	11703	8819	7099	5818
fs 680 1	83	1164	559	391	302	198	172
sherman1	303	5766	2865	1957	1494	1233	970
nos3	223	—	6471	4289	3087	1805	1565
sherman5	926	—	—	—	—	—	—
cavity05	353	—	—	4949	—	4426	8785
e05r0500	236	—	—	—	—	—	—
comsol	222	—	—	—	—	—	—
olm1000	456	—	—	—	—	—	—
cavity10	547	—	—	15721	12117	21998	—
steam2	123	—	5334	269	186	228	152
1138bus	516	—	—	—	—	—	—
steam1	186	—	—	—	—	—	—
bcsstk26	579	—	—	—	—	—	—
nos7	402	—	—	—	—	—	—
watt1	267	5470	2906	1986	1471	1235	1055
bcsstk14	1392	—	—	—	—	—	—
fs 183 6	9	9	9	9	9	9	9
bcsstk20	489	—	—	—	—	—	—
mcf6	776	—	—	—	—	—	—
nnc	330	—	—	—	—	—	—
lnsp	668	—	—	—	—	—	—

Table 11.2 gives the number of iterations for GMRES-DR(m) with 5 harmonic Ritz vectors computed at each restart. We observe improvements in convergence but we notice that some numbers of iterations are still quite large.

Table 11.2 GMRES-DR, 5 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6} \|b\|$, nitmax=10n

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	54	70	66	65	63	58	54
gre 343	216	—	—	—	—	—	—
jphw 991	45	54	48	46	45	45	45
pde2961	188	268	261	312	300	316	305
jagmesh1	164	2688	675	437	298	310	283
bfsa782	203	—	382	262	233	224	218
dw2048	406	1979	920	755	712	636	599
jagmesh2	448	6068	1503	917	755	666	619
raefsky2	315	937	631	555	538	538	540
fs 680 1c	60	82	86	76	69	69	60
add20	210	1582	370	262	261	242	255
raefsky1	197	660	275	246	234	237	225
jagmesh4	554	—	4928	2158	2090	1703	1469
fs 680 1	83	1274	518	334	263	206	162
sherman1	303	1765	731	560	498	468	448
nos3	223	612	321	268	265	250	251
sherman5	926	—	—	3836	3340	2830	2579
cavity05	353	2237	1535	1114	768	743	606
e05r0500	236	—	—	—	—	—	—
comsol	222	—	499	417	374	342	361
olm1000	456	—	—	—	—	—	—
cavity10	547	5262	2102	1903	1669	1537	1162
steam2	123	463	173	143	139	134	136
1138bus	516	—	—	0-	9662	8602	6272
steam1	186	-	—	—	—	—	—
bcsstk26	579	—	—	16454	11776	8990	6981
nos7	402	—	—	—	—	—	—
watt1	267	1137	443	362	327	311	306
bcsstk14	1392	—	—	—	—	—	—
fs 183 6	9	9	9	9	9	9	9
bcsstk20	—	—	—	—	—	—	—
mcf6	776	—	—	—	—	—	—
nnc	330	—	—	—	—	—	—
lnsp	668	—	—	—	—	—	—

Table 11.3 gives the number of iterations for GMRES-DR(m) with 10 harmonic Ritz vectors. We observe improvements in convergence.

Table 11.3 GMRES-DR, 10 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n

matrix	full GMRES	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	54	64	62	61	56	54
gre 343	216	—	—	—	—	—
jphw 991	45	48	45	45	45	45
pde2961	188	240	254	272	267	252
jagmesh1	164	448	284	268	263	252
bfsa782	203	245	225	220	220	217
dw2048	406	903	704	626	549	552
jagmesh2	448	1312	805	653	570	558
raefsky2	315	531	502	484	422	403
fs 680 1c	60	83	69	66	67	60
add20	210	370	279	261	242	230
raefsky1	197	233	221	217	218	215
jagmesh4	554	2971	1735	1289	948	850
fs 680 1	83	495	341	273	215	171
sherman1	303	683	512	474	457	442
nos3	223	283	255	245	243	242
sherman5	926	—	3896	2859	2217	2075
cavity05	353	9350	846	669	556	581
e05r0500	236	—	—	—	—	—
comsol	222	405	272	266	252	256
olm1000	456	—	—	—	—	—
cavity10	547	3557	456	1221	1258	1005
steam2	123	268	150	138	129	134
1138bus	516	—	10362	7512	5886	4803
steam1	186	—	—	—	—	—
bcsstk26	579	18178	6731	8081	6283	4206
nos7	402	—	—	—	—	—
watt1	267	350	329	317	309	301
bcsstk14	1392	—	—	—	—	—
fs 183 6	9	9	9	9	9	9
bcsstk20	489	—	—	—	—	—
mcfe	776	—	—	—	—	—
nnc	330	—	—	—	—	—
lnsp	668	—	—	—	—	—

Table 11.4 gives the number of iterations for LGMRES(m) with 5 approximate error vectors. We observe improvements in convergence compared to GMRES(m) but the dimension of the subspaces is not the same since it is $m + 5$ for LGMRES(m).

Table 11.4 LGMRES, 5 vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	54	98	88	68	71	54	54
gre 343	216	—	—	—	—	—	—
jphw 991	45	91	59	46	45	45	45
pde2961	188	565	502	488	435	413	440
jagmesh1	164	596	571	370	405	389	370
bfsa782	203	2311	1019	486	431	387	455
dw2048	406	941	778	675	626	607	584
jagmesh2	448	806	669	618	591	601	581
raefsky2	315	1846	952	868	620	625	592
fs 680 1c	60	—	224	91	73	77	60
add20	210	512	371	302	282	268	271
raefsky1	197	834	521	312	315	333	346
jagmesh4	554	1241	1046	987	1028	986	917
fs 680 1	83	761	536	393	265	191	159
sherman1	303	787	571	511	488	464	458
nos3	223	581	381	346	315	381	395
sherman5	926	—	—	—	—	—	—
cavity05	353	1661	1401	1184	716	766	719
e05r0500	236	—	—	—	—	—	—
comsol	222	—	—	2061	1306	1320	1205
olm1000	456	—	—	—	—	—	—
cavity10	547	2096	1493	1606	1486	1151	1051
steam2	123	—	2171	169	171	194	159
1138bus	516	7471	5821	3958	4444	4176	3718
steam1	186	—	—	—	—	—	—
bcsstk26	579	6456	4586	3863	3506	3272	3064
nos7	402	—	—	—	7286	—	7211
watt1	267	610	371	346	346	367	341
bcsstk14	1392	—	—	—	—	—	—
fs 183 6	9	9	9	9	9	9	9
bcsstk20	489	—	—	—	—	—	—
mcf6	776	—	—	—	—	—	—
nnc	330	—	—	—	—	—	—
lnsp	668	—	—	—	—	—	—

Table 11.5 gives the number of iterations for HBGMRES(m). We also observe some improvements in convergence compared to GMRES(m). Remember that HBGMRES uses only one approximate error vector.

Table 11.5 HBGMRES, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	54	99	102	79	69	63	54
gre 343	216	—	—	—	—	—	—
jphw 991	45	64	62	47	46	45	45
pde2961	188	350	360	405	440	504	468
jagmesh1	164	940	1075	840	755	600	480
bfsa782	203	2260	904	664	473	534	489
dw2048	406	940	980	799	730	766	704
jagmesh2	448	1156	1180	1045	920	754	723
raefsky2	315	2421	1441	648	760	760	644
fs 680 1c	60	—	318	109	72	76	60
add20	210	664	420	342	302	258	274
raefsky1	197	1088	549	670	571	494	452
jagmesh4	554	3269	1540	1774	1737	1650	1692
fs 680 1	83	853	480	352	294	199	172
sherman1	303	1030	1053	731	652	569	559
nos3	223	860	638	570	540	456	540
sherman5	926	—	—	—	—	—	—
cavity05	353	2390	1900	1470	1000	1200	955
e05r0500	236	—	—	—	—	—	—
comsol	222	—	1325	1994	2600	2000	1268
olm1000	456	—	—	—	—	—	—
cavity10	547	2660	3080	2940	2456	1746	1500
steam2	123	—	2536	248	154	179	152
1138bus	516	—	—	—	10683	8906	8741
steam1	186	—	—	—	—	—	—
bcsstk26	579	7923	6168	5430	5460	4706	4560
nos7	402	—	—	—	—	—	—
watt1	267	960	668	576	440	510	482
bcsstk14	1392	—	—	—	—	—	—
fs 183 6	9	9	9	9	9	9	9
bcsstk20	489	—	—	—	—	—	—
mcf6	776	—	—	—	—	—	—
nnc	330	—	—	—	—	—	—
lnsp	668	—	—	—	—	—	—

Table 11.6 displays the results of GMRES(20), GMRES-DR(20) and LGMRES(15) when using 5 vectors, all with the same subspace dimension. Notice that there are examples for which LGMRES does better than GMRES-DR. In almost all cases GMRES-DR and LGMRES greatly improve the number of iterations of restarted GMRES.

Table 11.6 $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, 5 vectors

matrix	GMRES(20)	GMRES-DR(20)	LGMRES(15)
pde225	100	66	105
gre 343	—	—	—
jphw 991	63	48	66
pde2961	340	261	567
jagmesh1	3261	675	656
bfsa782	2297	382	1516
dw2048	1940	920	802
jagmesh2	3803	1503	730
raefsky2	2494	631	1401
fs 680 1c	—	86	—
add20	674	370	376
raefsky1	6086	275	676
jagmesh4	—	4928	1149
fs 680 1	559	518	594
sherman1	2865	731	603
nos3	6471	321	384
sherman5	—	—	—
cavity05	—	1535	1785
e05r0500	—	—	—
comsol	—	499	—
olm1000	—	—	—
cavity10	—	2102	1841
steam2	5334	173	4316
1138bus	—	—	6436
steam1	—	—	—
bcsstk26	—	—	5196
nos7	—	—	—
watt1	2906	443	556
bcsstk14	—	-	—
fs 183 6	9	9	9
bcsstk20	—	—	—
mcfe	—	—	—
nnc	—	—	—
lnsp	—	—	—

Table 11.7 displays the results of GMRES(20), GMRES-DR(20) and LGMRES(10) when using 10 vectors, all with the same subspace dimension. The conclusions are the same as for Table 11.6. Unfortunately, it seems difficult to know beforehand which method will be the best of GMRES-DR and LGMRES.

Table 11.7 $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, 10 vectors

matrix	GMRES(20)	GMRES-DR(20)	LGMRES(10)
pde225	100	64	105
gre 343	—	—	—
jphw 991	63	48	66
pde2961	340	240	625
jagmesh1	3261	448	703
bfsa782	2297	245	2361
dw2048	1940	903	1131
jagmesh2	3803	1312	1031
raefsky2	2494	531	1471
fs 680 1c	—	83	—
add20	674	370	531
raefsky1	6086	233	823
jagmesh4	—	2971	1810
fs 680 1	559	495	906
sherman1	2865	683	951
nos3	6471	283	491
sherman5	—	—	—
cavity05	—	9350	1531
e05r0500	—	—	—
comsol	—	405	—
olm1000	—	—	—
cavity10	—	3557	1926
steam2	5334	268	—
1138bus	—	—	9991
steam1	—	—	—
bcsstk26	—	18178	8488
nos7	—	—	—
watt1	2906	350	731
bcsstk14	—	—	—
fs 183 6	9	9	9
bcsstk20	—	—	—
mcf6	—	—	—
nnc	—	—	—
lnsp	—	—	—

11.9.4 Restarting with deflation and augmentation with preconditioning

In Table 11.8 we give the number of iterations needed to obtain $\|r_k\| \leq 10^{-6}\|b\|$ for our set of small matrices with GMRES(m) and a left diagonal preconditioner. We removed from the list the matrices having zero entries on the diagonal. However, some matrices may have small diagonal entries and this may spoil the convergence. For some matrices like `jagmesh1` there is no improvement in the number of iterations because they have already a unit diagonal. For matrices like `bcsstk26` or `mcfe` we obtain convergence contrary to the cases without preconditioning. But, note that we stop on the magnitude of the norm of the preconditioned residual. Hence, the true residual norm might be larger.

Table 11.8 GMRES(m), $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	53	88	102	76	66	58	53
jpwh 991	35	47	42	36	35	35	35
pde2961	170	302	284	308	330	448	460
jagmesh1	164	6562	3261	2225	1573	1286	849
bfwa782	152	6485	1065	681	680	550	456
dw2048	291	—	—	—	—	—	—
jagmesh2	448	7715	3803	2476	1909	1557	1292
raefsky2	315	3736	2619	2452	2418	2191	2111
fs 680 1c	60	—	—	115	72	76	60
add20	80	203	103	101	94	91	86
raefsky1	197	8177	6033	3078	2590	2127	1823
jagmesh4	554	—	—	11703	8819	7099	5818
fs 680 1	60	—	—	115	72	76	60
sherman1	199	1121	850	677	559	365	473
nos3	191	2410	3659	1224	1297	1357	868
sherman5	118	745	638	475	311	328	291
comsol	225	—	—	—	—	—	—
olm1000	486	—	—	—	—	—	—
steam2	5	5	5	5	5	5	5
1138bus	705	—	—	—	—	—	—
steam1	9	9	9	9	9	9	9
bcsstk26	450	1847	1146	791	947	760	659
nos7	27	32	31	27	27	27	27
watt1	104	321	327	282	168	188	139
bcsstk14	12	16	12	12	12	12	12
fs 183 6	12	—	12	12	12	12	12
bcsstk20	485	—	—	—	—	—	—
mcfe	109	2765	1720	1124	826	543	387

Table 11.9 displays the number of iterations for GMRES-DR(m) using 5 harmonic Ritz vectors and a left diagonal preconditioner. Comparing with Table 11.8, we observe improvements in the number of iterations and also convergence for problems which were not converging with the standard restart procedure.

Table 11.9 GMRES-DR, 5 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	53	62	62	54	60	50	50
jphw 991	35	37	35	35	35	35	35
pde2961	170	209	222	249	245	226	268
jagmesh1	164	2688	675	437	298	310	283
bawa782	152	2915	184	155	145	162	157
dw2048	291	—	—	—	—	—	—
jagmesh2	448	6068	1503	917	755	666	619
raefsky2	315	830	627	551	534	536	540
fs 680 1c	60	82	86	76	69	69	60
add20	80	150	94	93	90	88	86
raefsky1	197	667	273	246	234	237	225
jagmesh4	554	—	4928	2158	2090	1703	1469
fs 680 1	60	—	—	—	—	—	—
sherman1	199	255	222	213	208	209	207
nos3	191	270	200	180	180	185	207
sherman5	118	223	137	129	134	130	115
comsol	225	—	788	441	389	373	375
olm1000	486	126	106	103	109	138	115
steam2	5	2	2	2	2	2	2
1138bus	705	—	—	—	1299	768	720
steam1	9	9	9	9	9	9	9
bcsstk26	450	—	—	—	—	—	—
nos7	27	28	27	31	34	34	34
watt1	104	130	110	109	106	106	106
bcsstk14	12	—	—	—	—	—	—
fs 183 6	12	10	10	10	10	10	10
bcsstk20	485	—	—	—	—	—	—
mfe	109	53	34	24	24	24	24

Table 11.10 displays the number of iterations for GMRES-DR(m) using 10 harmonic Ritz vectors and a left diagonal preconditioner.

Table 11.10 GMRES-DR, 10 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	53	58	58	57	50	50
jpwh 991	35	35	35	35	35	35
pde2961	170	196	207	219	209	209
jagmesh1	164	448	284	268	263	252
bawa782	152	159	150	158	157	155
dw2048	291	—	—	—	—	—
jagmesh2	448	1312	805	653	570	558
raefsky2	315	544	438	432	422	403
fs 680 1c	60	83	69	66	67	60
add20	80	96	88	90	85	83
raefsky1	197	233	221	217	218	215
jagmesh4	554	2971	1735	1289	948	850
fs 680 1	60	—	—	—	—	—
sherman1	199	214	208	205	204	204
nos3	191	180	170	190	170	199
sherman5	118	139	130	128	126	125
comsol	225	484	284	273	263	263
olm1000	486	100	110	100	130	110
steam2	5	2	2	2	2	2
1138bus	705	—	787	727	687	658
steam1	9	9	9	9	9	9
bcsstk26	450	—	—	—	—	—
nos7	27	27	31	34	34	34
watt1	104	108	106	106	105	105
bcsstk14	12	—	—	—	—	—
fs 183 6	12	10	10	10	10	10
bcsstk20	485	—	—	—	—	—
mcfe	109	30	24	24	24	24

Table 11.11 displays the number of iterations for LGMRES(m) using 5 approximate error vectors and a left diagonal preconditioner. Most of the results are worse than with GMRES-DR. Note that the dimensions of the subspaces are not the same.

Table 11.11 LGMRES, 5 vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	53	94	78	68	69	53	53
jphw 991	35	53	36	35	35	35	35
pde2961	170	530	414	372	350	355	370
jagmesh1	164	596	571	370	405	389	370
bawa782	152	1592	525	383	311	274	280
dw2048	291	—	—	—	—	—	—
jagmesh2	448	806	669	618	591	601	581
raefsky2	315	2164	981	868	624	622	592
fs 680 1c	60	—	224	91	73	77	60
add20	80	137	108	93	88	87	85
raefsky1	197	843	521	312	315	333	346
jagmesh4	554	1241	1046	987	1028	986	917
fs 680 1	60	—	224	91	73	77	60
sherman1	199	341	252	242	266	272	322
nos3	191	589	353	335	341	362	338
sherman5	118	402	301	255	269	256	249
comsol	225	—	—	2306	1391	1371	1171
olm1000	486	—	—	—	9671	7425	4546
steam2	5	5	5	5	5	5	5
1138bus	705	—	—	—	—	—	—
steam1	9	9	9	9	9	9	9
bcsstk26	450	0	796	766	724	676	649
nos7	27	29	29	27	27	27	27
watt1	104	146	133	147	170	164	147
bcsstk14	12	12	12	12	12	12	12
fs 183 6	12	12	12	12	12	12	12
bcsstk20	485	—	—	—	—	—	—
mcfe	109	1669	1161	688	484	351	324

Table 11.12 compares the number of iterations of GMRES(20), GMRES-DR(20) and LGMRES(15) both using 5 vectors and a left preconditioner. There are a few examples for which the number of iterations of LGMRES is smaller than that of GMRES-DR.

Table 11.12 $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, 5 vectors, diagonal preconditioner

matrix	GMRES(20)	GMRES-DR(20)	LGMRES(15)
pde225	102	62	102
jphw 991	42	35	44
pde2961	284	222	466
jagmesh1	3261	675	656
bawa782	1065	184	794
dw2048	—	—	—
jagmesh2	3803	1503	730
raefsky2	2619	627	1472
fs 680 1c	—	86	—
add20	103	94	109
raefsky1	6033	273	676
jagmesh4	—	4928	1149
fs 680 1	—	—	3774
sherman1	850	222	296
nos3	3659	200	376
sherman5	638	137	224
comsol	—	788	—
olm1000	—	106	—
steam2	5	2	5
1138bus	—	—	—
steam1	9	9	9
bcsstk26	1146	—	1057
nos7	31	27	31
watt1	327	110	132
bcsstk14	12	—	12
fs 183 6	12	10	12
bcsstk20	—	—	—
mcfe	1720	34	1306

Table 11.13 displays the number of iterations of GMRES(20), GMRES-DR(20) and LGMRES(10) both using 10 vectors and a left preconditioner. The conclusions are the same as with 5 vectors.

Table 11.13 $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, 10 vectors, diagonal preconditioner

matrix	GMRES(20)	GMRES-DR(20)	LGMRES(10)
pde225	102	58	102
jphw 991	42	35	44
pde2961	284	196	469
jagmesh1	3261	448	703
bawa782	1065	159	1448
dw2048	–	–	–
jagmesh2	3803	1312	1031
raefsky2	2619	544	1426
fs 680 1c	–	83	–
add20	103	96	109
raefsky1	6033	233	812
jagmesh4	–	2971	1810
fs 680 1	–	–	–
sherman1	850	214	324
nos3	3659	180	451
sherman5	638	139	221
comsol	–	484	–
olm1000	–	100	–
steam2	5	2	5
1138bus	–	–	–
steam1	9	9	9
bcsstk26	1146	–	–
nos7	31	27	31
watt1	327	108	132
bcsstk14	12	–	12
fs 183 6	12	10	12
bcsstk20	–	–	–
mcfe	1720	30	1989

Table 11.14 shows the number of iterations for HBGMRES(m) with different values of m and a left diagonal preconditioner. The results are generally worse than those of LGMRES but remember that HBGMRES uses only one approximate error vector.

Table 11.14 GMRESHB, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
pde225	53	100	98	76	66	58	53
jpwh 991	35	49	42	36	35	35	35
pde2961	170	300	350	365	410	395	448
jagmesh1	164	940	1075	840	755	600	480
bawa782	152	1487	646	358	540	330	283
dw2048	291	—	—	—	—	—	—
jagmesh2	448	1156	1180	1045	920	754	723
raefsky2	315	2339	980	714	661	750	642
fs 680 1c	60	—	318	109	72	76	60
add20	80	129	95	96	94	91	86
raefsky1	197	750	552	668	576	494	452
jagmesh4 1440	554	3269	1540	1774	1737	1650	1692
fs 680 1	60	0	318	109	72	76	60
sherman1	199	420	400	306	320	280	344
nos3	191	658	580	390	406	452	480
sherman5	118	450	345	328	279	271	266
comsol	225	—	—	3720	2190	1342	1441
olm1000	486	—	—	6902	8098	6502	2941
steam2	5	5	5	5	5	5	5
1138bus	705	—	—	—	—	—	—
steam1	9	9	9	9	9	9	9
bcsstk26	450	832	800	887	773	730	720
nos7	27	32	31	27	27	27	27
watt1	104	220	180	177	154	154	139
bcsstk14	12	16	12	12	12	12	12
fs 183 6	12	—	12	12	12	12	12
bcsstk20	485	—	—	—	—	—	—
mcf	109	1467	880	744	592	437	306

On the set of small matrices, with or without a diagonal preconditioner, the conclusions are that, except for a few problems, GMRES-DR gives better results than LGMRES when using the same subspace dimension and that it does not seem necessary to use a large number of approximate eigenvectors to obtain a better convergence than with the standard restarted GMRES.

We now consider a set of larger matrices for which we also use a left diagonal preconditioner since the numbers of iterations are too large without any preconditioner. The `supg` matrices are of order 6400 and the `convdiff` matrices are of order 8100. For this set of problems LGMRES gives worse results than GMRES-DR. Hence, we do not show the results of LGMRES or HBGMRES.

Table 11.15 displays the number of iterations of GMRES(m) with different values of m . The numbers of iterations for the `supg` matrices are larger than those of full GMRES and increasing m does not always improve the convergence.

Table 11.15 GMRES, $\|r_k\| \leq 10^{-6} \|b\|$, nitmax=1000, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
ex37	18	18	18	18	18	18	18
memplus	60	102	73	72	66	66	60
sherman3	289	—	—	—	—	957	—
wang4	130	—	—	—	—	635	514
supg 100	276	—	678	460	436	347	306
supg 1	640	—	723	675	667	654	652
supg 01	285	315	337	362	390	367	380
supg 001	103	128	166	162	154	151	151
supg 0001	89	140	180	178	158	182	147
supg 00001	159	202	240	276	295	313	341
supg 000001	181	209	244	279	309	350	389
convdiff xu 500	667	—	—	—	—	—	—
convdiff xu 1000	348	—	—	—	—	—	—
convdiff xu 5000	183	—	—	—	—	—	—

Table 11.16 shows the results for GMRES-DR with 5 harmonic Ritz vectors. We observe an improvement compared to the standard restarted GMRES even though it is not very spectacular.

Table 11.16 GMRES-DR, 5 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=1000, diagonal preconditioner

matrix	full GMRES	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
ex37	18	15	14	14	14	14	14
memplus	60	123	74	65	66	65	61
sherman3	289	260	184	180	180	185	225
wang4	130	—	338	182	171	167	155
supg 100	276	150	125	130	110	140	115
supg 1	644	—	675	649	632	632	647
supg 01	285	301	297	292	302	301	290
supg 001	103	112	121	130	119	106	108
supg 0001	89	105	119	115	123	108	108
supg 00001	159	185	217	233	250	238	252
supg 000001	181	204	251	276	309	291	330
convdif xu 500	—	—	—	—	—	—	—
convdif xu 1000	348	—	—	—	—	—	—
convdif xu 5000	183	—	—	—	689	625	603

We observe from Table 11.17 that using 10 harmonic Ritz vectors instead of 5 does not improve much the numbers of iterations. In fact, there are cases for which the results are worse than with 5 vectors. This is probably because, in our implementation of GMRES-DR, using 10 vectors reduces the number of Arnoldi iterations per cycle.

The conclusions for the set of larger matrices are that GMRES-DR reduces the number of iterations compared to the standard restarted GMRES and that it does not seem necessary to use a large number of approximate eigenvectors.

Table 11.17 GMRES-DR, 10 harmonic Ritz vectors, $\|r_k\| \leq 10^{-6}\|b\|$, diagonal preconditioner

matrix	full GMRES	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$
ex37	18	14	14	14	14	14
memplus	60	72	64	63	64	61
sherman3	289	178	189	189	170	209
wang4	130	275	177	167	157	149
supg 100	276	110	110	130	130	110
supg 1	644	790	639	632	648	648
supg 01	285	289	290	287	286	286
supg 001	103	109	109	107	104	104
supg 0001	89	107	107	104	102	101
supg 00001	159	208	227	229	234	232
supg 000001	181	239	267	284	286	293
convdiff xu 500	—	—	—	—	—	—
convdiff xu 1000	348	—	—	—	—	—
convdiff xu 5000	183	—	449	399	408	409

11.9.5 Truncated Q-OR

Figure 11.28 displays the relative true residual norms for the problem fs 680 1c using full GMRES, the truncated Q-OR method with parameters (1, 40) (see Chapter 4, Subsection 4.9.6) which is close to truncated GMRES (with a different implementation), truncated Q-OR with (20, 20) and the same with a restart at iteration 120. We observe that using vectors Av_j and v_j to build the basis improves the convergence. A restart at the end improves the maximum attainable accuracy.

Figure 11.29 shows the results for supg 001. We use the truncated Q-OR method with parameters (1, 20) and (10, 10).

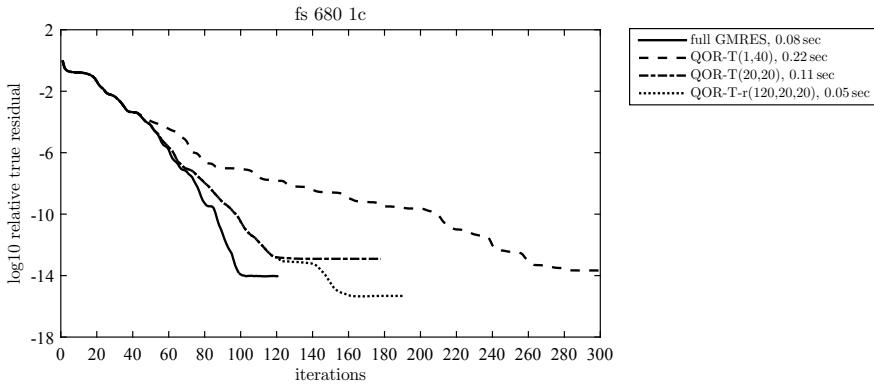


Fig. 11.28 fs 680 1c, relative true residual norms, full GMRES, QOR-T(1,40), QOR-T(20,20) and QOR-T(20,20) with restart at $k = 120$

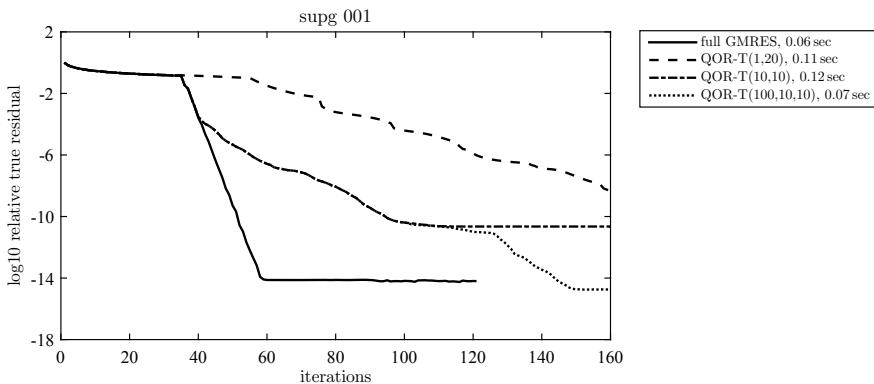


Fig. 11.29 supg 001, $n = 1225$, relative true residual norms, full GMRES, QOR-T(1,20), QOR-T(10,10) and QOR-T(10,10) with restart at $k = 100$

Table 11.18 shows the number of iterations for full GMRES and the truncated Q-OR method with different values (p, q). Most of the results are better than with GMRES(m) with $m = p + q$.

Table 11.18 QOR trunc(p,q), $\|r_k\| \leq 10^{-6} \|b\|$, nitmax=10n

matrix	full GMRES	(5, 5)	(10, 10)	(15, 15)	(20, 20)	(25, 25)	(30, 30)
pde225	54	83	61	57	55	54	54
gre 343	216	—	—	—	—	—	—
jphw 991	45	46	45	45	45	45	45
pde2961	188	381	406	303	254	223	209
jagmesh1	164	221	539	279	332	587	595
bfsa782	203	1045	617	297	230	214	210
dw2048	406	978	785	876	849	1258	728
jagmesh2	448	514	551	496	778	497	690
raefsky2	315	1431	567	519	457	428	395
fs 680 1c	60	209	104	65	62	60	60
add20	210	317	313	291	294	265	256
raefsky1	197	652	319	256	235	223	210
jagmesh4	554	652	903	827	910	1032	835
fs 680 1	83	441	330	261	200	151	119
sherman1	303	424	421	419	421	415	407
nos3	223	274	235	255	255	234	232
sherman5	926	—	—	21593	—	—	—
cavity05	353	1053	873	783	736	655	602
e05r0500	236	—	—	—	—	—	—
comsol	222	9888	865	414	314	273	250
olm1000	456	—	—	—	6327	2182	1213
cavity10	547	4659	1304	1228	1174	1050	981
steam2	123	—	3083	549	332	301	230
1138bus	516	3327	4343	2973	3836	2919	3011
steam1	186	—	—	—	—	—	—
bcsstk26	579	4306	3257	3283	2812	2729	2791
nos7	402	—	—	—	—	—	—
watt1	267	840	1260	1963	1716	2534	6788
bcsstk14	1392	—	—	—	—	—	—
fs 183 6	9	9	9	9	9	9	9
bcsstk20	489	—	—	—	—	—	—
mcfe	776	—	—	—	—	—	—
nnc	330	—	—	—	—	—	—
lnsp	668	—	—	—	—	—	—

Table 11.19 shows the number of iterations for full GMRES and the truncated Q-OR method with different values (p, q) and a diagonal preconditioner.

Table 11.19 QOR trunc(p,q), $\|r_k\| \leq 10^{-6}\|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	(5, 5)	(10, 10)	(15, 15)	(20, 20)	(25, 25)	(30, 30)
pde225	53	76	61	54	54	53	53
jphw 991	35	35	35	35	35	35	35
pde2961	170	316	312	256	233	215	183
jagmesh1	164	221	539	279	332	587	595
bfsa782	152	962	643	180	156	155	154
dw2048	291	—	—	—	—	—	—
jagmesh2	448	514	551	496	778	497	690
raefsky2	315	1625	678	542	456	418	388
fs 680 1c	60	230	99	65	62	60	60
add20	80	853	547	254	223	199	95
raefsky1	197	711	341	265	225	209	207
jagmesh4	554	652	903	827	910	1032	835
fs 680 1	60	193	89	66	62	60	60
sherman1	199	434	223	209	206	205	206
nos3	191	568	287	225	199	197	195
sherman5	118	311	194	142	130	123	120
comsol	225	—	1277	401	314	278	251
olm1000	486	—	7312	3408	2120	1446	1271
steam2	5	5	5	5	5	5	5
1138bus	705	5131	1405	2324	1310	1167	1024
steam1	9	9	9	9	9	9	9
bcsstk26	450	3007	973	764	1096	648	600
nos7	27	27	27	27	27	27	27
watt1	104	130	119	105	104	104	104
bcsstk14	12	12	12	12	12	12	12
fs 183 6	12	23	13	17	17	17	17
bcsstk20	485	—	—	—	—	—	—
mcf6	109	—	—	—	7143	5489	2311

Table 11.20 shows the number of iterations for the set of larger matrices using full GMRES and the truncated Q-OR method with different values (p, q) and a diagonal preconditioner. Most of the results are better than with GMRES(m) using $m = p + q$.

Table 11.20 QOR trunc(p, q), $\|r_k\| \leq 10^{-6} \|b\|$, nitmax=10n, diagonal preconditioner

matrix	full GMRES	(5, 5)	(10, 10)	(15, 15)	(20, 20)	(25, 25)	(30, 30)
ex37	18	18	18	18	18	18	18
memplus	60	392	235	151	76	63	60
sherman3	289	–	592	366	329	304	305
wang4	130	699	419	194	151	143	139
supg 100	276	277	277	276	276	276	276
supg 1	640	657	651	646	644	641	641
supg 01	285	406	418	364	339	289	298
supg 001	103	181	158	114	107	103	103
supg 0001	89	183	135	114	99	104	92
supg 00001	159	331	266	209	190	181	176
supg 000001	181	398	429	306	253	231	204
convdif xu 500	667	–	–	–	–	–	–
convdif xu 1000	348	–	–	–	–	–	–
convdif xu 5000	183	–	–	732	481	351	294

11.10 Historical notes

It was recognized very early that Krylov methods using long recurrences were too costly (in computing time and storage) when solving difficult problems. Restarting and truncation were proposed as remedies.

General papers about these methods are [297] by Michael Eiermann, Oliver Ernst and Olaf Schneider in 2000, [837] by Valeria Simoncini and Daniel B. Szyld in 2007 and [485] by Martin H. Gutknecht in 2012.

The convergence of GMRES(m) was studied by Wayne Joubert [582] in 1994, V. Simoncini [828] in 2000, André Gaul, M.H. Gutknecht, Jörg Liesen and Reinhard Nabben [402] in 2013. Sufficient conditions for convergence were published by Jan Zitko [1035–1038] in 2000–2008, J. Zitko and David Nádhera [1039] in 2010.

Marcel Schweitzer [819] in 2016 proved that any residual norms can be generated in the restarted FOM method. It was shown by Eugene Vecharynski and Julien Langou [952] in 2011 that any history of decreasing residual norms at the last iteration of every cycle of GMRES(m) is possible. The authors of this book extended these results by showing that prescribing the residual norms as well as the (harmonic) Ritz

values inside every cycle of GMRES(m) is possible provided there is no stagnation step at the very end of the cycle [290].

When restarting every m iterations, it is not obvious to choose a good value of m . There must be a balance between the efficiency and the cost. Unfortunately, it was noticed in [297] and in [318] by Mark Embree in 2003 that increasing m does not always improve the convergence of GMRES(m). Strategies for choosing a “good” value of m for GMRES were proposed by W. Joubert [582] in 1994, Maria Sosonkina, Layne T. Watson, Rakesh K. Kapania and Homer F. Walker [871] in 1998, Mitsuru Habu and Takashi Nodera [497] in 2000, Linjie Zhang and T. Nodera [1028] in 2005, Kentaro Moriya and T. Nodera [707] in 2007 (based on a paper by Naoto Tsuno and T. Nodera [928] in 1999), Allison H. Baker, Elizabeth R. Jessup and Tzania V. Kolev [63] in 2009 and Rolando Cuevas, Christian E. Schaerer and Amit Bhaya [241] in 2010.

Augmentation and deflation to improve convergence was first proposed for solving symmetric linear systems with the conjugate gradient (CG) method. Roy A. Nicolaides [717] used deflation for CG in 1987. Zdeněk Dostál [273, 274] constructed preconditioners using projectors to eliminate the influence of some eigenvalues in 1988–1990. Deflation was also advocated by Lois Mansfield [655] in 1988. Augmented CG algorithms were introduced by Jocelyne Erhel and Frédéric Guyomarc'h [323] in 2000 and Yousef Saad, Man Cheung Yeung, J. Erhel and F. Guyomarc'h [804] in 2000. Deflation preconditioners were used by Cornelis Vuik, Guus Segal and J.A. Meijerink [959] in 1999.

In fact, what can be seen as a kind of deflation was considered earlier by John R. Wallis, Richard P. Kendall and T.E. Little [970] in 1985. In this paper they show how to introduce constraints on the residual vector in the GCR algorithm, following ideas from James W. Watts [975] in 1973.

Analyses of augmented Krylov methods were published by Y. Saad [796] and Andrew Chapman and Y. Saad [213] in 1997. This was also considered by M. Eiermann, O. Ernst and O. Schneider [297] in 2000 and V. Simoncini [837] in 2007. Augmented GMRES methods were considered by James Baglama and Lothar Reichel [52] in 1998.

An augmented method named LGMRES (meaning “Loose” GMRES) was proposed by A.H. Baker, E.R. Jessup and Thomas A. Manteuffel [64] in 2005; see also [60] in 2003. GMRES is augmented by rough approximations of the error vector. Another implementation was published by A.H. Baker, John M. Dennis and E.R. Jessup [61] in 2003; see also [62]. This method was rediscovered with a different implementation by Akira Imakura, Ren-Cang Li and Shao-Liang Zhang [542] in 2016 who proposed two methods, the locally optimal GMRES (LOGMRES) and the heavy ball GMRES (HBGMRES); related papers are by A. Imakura, Tomohiro Sogabe and S.-L. Zhang [543, 544] in 2012 and 2018.

Augmenting GMRES with approximate eigenvectors was mainly considered in a series of papers by Ronald B. Morgan. He proposed different implementations of this idea, GMRES-E [695] in 1995, GMRES-IR [696] in 2000 based on the implicitly restarted Arnoldi algorithm and GMRES-DR [697] in 2002 generalizing

to nonsymmetric problems the idea of thick restarting by Kesheng Wu and Horst Simon [989] in 2000.

Another implementation of GMRES-IR was proposed by Caroline Le Calvez and Brigida Molina [621] in 1999. Quan Liu, R.B. Morgan and Walter Wilcox [650] considered adding a polynomial preconditioner to GMRES-DR in 2015. Augmentation with a combination of harmonic Ritz vectors and approximations of the error vector can be found in [719] by Qiang Niu and Linzhang Lu in 2012.

An augmented CMRH algorithm with error approximations was introduced by Zhongming Teng and Xuansheng Wang [908] in 2018.

The other way to use deflation is to construct preconditioners that, when applied to the matrix, remove some eigenvalues or move them to locations assumed to be better for convergence. This was considered by J. Erhel, Kevin Burrage and Bert Pohl [322] in 1996; see also [163] by K. Burrage and J. Erhel and [164] by K. Burrage, J. Erhel, B. Pohl and Alan Williams in 1998.

The construction of preconditioners using spectral information was also proposed by J. Baglama, Daniela Calvetti, Gene H. Golub and L. Reichel [50] in 1998. Other algorithms were considered by Yogi A. Erlangga and R. Nabben [326] in 2009 and M.C. Yeung, Jok Man Tang and C. Vuik [1012] in 2010.

A theoretical study of methods using a deflated matrix was published by M.H. Gutknecht [485] in 2012.

Truncation of the long recurrences was also proposed to reduce the cost of the iterations. A method named Orthomin was introduced by Paul K.W. Vinsome [953] in 1976. It can be considered as a truncated version of GCR. In 1981 Y. Saad proposed an incomplete orthogonalization method (IOM) [789]. It can be seen as a truncated FOM method; see also [792] in 1984. A truncated GMRES method named IGMRES was published by Peter N. Brown and Alan C. Hindmarsh [153] in 1989. In this method all the basis vectors are needed to compute the solution after convergence. This was relaxed in DQGMRES by Yousef Saad and Kesheng Wu [803] in 1996 where the approximate solution is computed recursively. A restarted version of DQGMRES was considered by K. Wu [988] in 1997.

The properties of the bases obtained by truncation were studied by Zhongxiao Jia [565, 566] in 1996–1998.

Parallel versions of deflated restarted GMRES were proposed by Désiré Nuentsa Wakam and J. Erhel [962] and D. Wakam, J. Erhel and William Gropp [963] in 2013.

The truncated Q-OR method is due to the first author of this book.

Chapter 12

Related topics



In this chapter we give pointers to the literature concerning topics that we have not covered in the previous chapters.

12.1 FGMRES

12.1.1 *Definition*

In preconditioned GMRES, the preconditioner M has to be a given fixed matrix in order for the subspace where the approximate solution is sought to be a proper Krylov subspace. Flexible GMRES [795] is an iterative method allowing the use of a different preconditioner at each iteration. This may be the case, for instance, if another iterative method is used as a preconditioner. Hence, the word *flexible* refers to the fact that the preconditioner may change for one iteration to the next.

The algorithm, which looks like GMRES, is as follows, with right preconditioning with matrices M_k .

Let x_0 be given, $r_0 = b - Ax_0$,
 $v_1 = \frac{r_0}{\|r_0\|}$, $f = \|r_0\|e_1$,

for $k = 1, \dots$

$M_k z_k = v_k$, $w = Az_k$,

for $i = 1, \dots, k$

$h_{i,k} = (w, v_i)$, $w = w - h_{i,k} v_i$,

end for i

$h_{k+1,k} = \|w\|$, $v_{k+1} = \frac{w}{h_{k+1,k}}$,

Apply the rotations of iterations 1 to $k - 1$ on $(h_{1,k} \dots h_{k+1,k})^T$,

Compute the Givens rotation $G_{k+1,k}$ to eliminate $h_{k+1,k}$, $f = G_{k+1,k} f$,

Solve the triangular system for y_k ,
 Compute the norm of the residual related to the last component of f ,
 if it is small enough compute $x_k = x_0 + Z_k y_k$,
 where $Z_k = (z_1 \cdots z_k)$ and stop,
 end for k .

Note that we used a CGS-like algorithm, but, as in GMRES, an MGS-like algorithm is to be preferred. The main difference with GMRES is that we have to store the vectors z_j which are used to compute the approximate solution. Here, the basic relation for the basis vectors is

$$AZ_k = V_{k+1}H_k.$$

Theorem 12.1 *The approximate solution x_k given by FGMRES minimizes the norm of the residual over $x_0 + \text{span}(z_1, \dots, z_k)$.*

Proof See Saad [795]. □

It must be noted that the convergence results about GMRES do not carry over to FGMRES because the space where the solution is sought is not a Krylov subspace. Moreover, the exact solution is obtained when $h_{k+1,k} = 0$ only if H_k is nonsingular.

Krylov methods named GMRESR and GMRES*, able to use varying preconditioners, were proposed in [944]. The idea is to precondition the linear system by an approximation of the inverse that can be derived from the first iterations. The basic method uses iterations of GMRES itself as the preconditioner, hence the name GMRESR, the R standing for recursive. Afterward, flexible versions of many Krylov methods were published in the literature.

12.1.2 Historical notes

FGMRES was introduced in 1993 by Yousef Saad [795]. In this paper he proved the optimality result we stated above in Theorem 12.1. In [796], published in 1997, he used the FGMRES framework to develop augmented Krylov methods where the Krylov subspaces are augmented with some other vectors. This was further studied by Andrew Chapman and Y. Saad in [213].

GMRESR was proposed in 1994 by Henk van der Vorst and Cornelis Vuik [944]. The algorithm is in fact based on GCR rather than on GMRES, even though these methods are mathematically equivalent. A comparison with FGMRES and proposals of new variants of both methods were published in 1995 by C. Vuik [958].

Fortran implementations of FGMRES from Valérie Frayssé, Luc Giraud and Serge Gratton were made available at CERFACS in 1998 [363].

Flexible methods were studied in 2001 by Valeria Simoncini and Daniel B. Szyld [833].

Below is a (very incomplete) long list of papers about flexible variants of some Krylov methods:

- 2001: FQMR, a flexible quasi-minimal residual method by D.B. Szyld and Julie A. Vogel [903].
- 2005: SOR preconditioned GCR-like methods by Kuniyoshi Abe and Shao-Liang Zhang [6].
- 2007: Flexible BiCG and Bi-CGSTAB by J.A. Vogel [955].
- 2007: A perturbed LDL^T preconditioner for GMRES by Mario Arioli, Iain S. Duff, S. Gratton and Stéphane Pralet [26].
- 2009: Use of FGMRES to obtain backward stability in mixed precision by M. Arioli and I.S. Duff [25].
- 2010: Deflated restarting with flexible GMRES by L. Giraud, S. Gratton, Xavier Pinel, and Xavier Vasseur [418].
- 2010: A simplified and flexible variant of GCROT by Jason E. Hicken and David W. Zingg [520].
- 2011: A flexible generalized conjugate residual method by Luiz M. Carvalho, S. Gratton, Rafael F. Lago and X. Vasseur [196].
- 2012: Flexible variants of block restarted GMRES by Henri Calandra, S. Gratton, Julien Langou, X. Pinel and X. Vasseur [167].
- 2012: A flexible GPBi-CG method by Jia-Min Wang and Tong-Xiang Gu [972].
- 2014: FGMRES for linear discrete ill-posed problems by Keichi Morikuni, Lothar Reichel and Ken Hayami [704].
- 2014: A deflated block flexible GMRES-DR method by Jing Meng, Pei-Yong Zhu, Hou-Biao Li and Xian-Ming Gu [670].
- 2014: A flexible CMRH algorithm by Ke Zhang and Chuanqing Gu [1027].
- 2015: Flexible and multi-shift induced dimension reduction algorithms by Martin B. Van Gijzen, Gerard L.G. Sleijpen and Jens-Peter M. Zemke [946].
- 2016: Analysis of flexible BICGSTAB by Jie Chen, Lois C. McInnes and Hong Zhang [217].
- 2016: Pipelined, flexible Krylov subspace methods by Patrick Sanan, Sascha M. Schnepp and Dave A. May [811].
- 2017: GMRES with multiple preconditioners by Chen Greif, Tyrone Rees and D.B. Szyld [466].
- 2018: Adaptive multilevel Krylov methods by René Kehl, Reinhard Nabben and D.B. Szyld [592].

Most papers referring to FGMRES, not cited above, are related to applications of the method to some specific scientific computing problems.

12.2 GCRO

12.2.1 *Definition*

The development of GCRO [260, 262] was seen as an improvement over GMRESR proposed in [944]. Let us assume that we have two matrices $U_{n,k}$ and $C_{n,k}$ such that

$$AU_{n,k} = C_{n,k}, \quad C_{n,k}^T C_{n,k} = I,$$

with columns u_j and c_j . Let $x_0 = 0$ and $x_k \in \text{range}(U_{n,k})$ such that the norm of the residual is minimized. It yields

$$r_k = b - AU_{n,k}y_k = b - C_{n,k}y_k,$$

with

$$y_k = \operatorname{argmin}_y \|b - C_{n,k}y\| = C_{n,k}^T b.$$

Therefore, $x_k = U_{n,k}C_{n,k}^T b$ and $r_k = (I - C_{n,k}C_{n,k}^T)b$. But, we have to compute u_{k+1} and c_{k+1} for the next iteration. Taking $c_{k+1} = r_k$ would give a zero residual vector at the next iteration. However, this is not a feasible choice since it implies $u_{k+1} = A^{-1}r_k = \varepsilon_k$. In GMRESR, u_{k+1} was chosen to be an approximation of the error vector ε_k by running a given number of GMRES starting from r_k . The improvement in GCRO (the “O” referring to “orthogonalization”) is to require that the inner iteration maintains orthogonality to the columns of $C_{n,k}$. This is obtained by running GMRES with $A_{C_{n,k}} = (I - C_{n,k}C_{n,k}^T)A$ instead of A . Let y_m be

$$y_m = \operatorname{argmin}_y \|r_k - A_{C_{n,k}}V_{n,m}y\| = \operatorname{argmin}_y \|r_k - \underline{H}_m y\|.$$

Then, the next iterate is

$$x_{k+1} = x_k + (I - C_{n,k}C_{n,k}^T)AV_{n,m}y_m$$

which is the solution of the minimization problem in $\text{range}(U_{n,k}) \oplus \text{range}(V_{n,m})$.

Unfortunately, the storage for GCRO increases with each outer iteration. A truncated version named GCROT was derived in [261]. The truncation strategy is based on trying to determine which subspace in $\text{range}(C_{n,k})$ was the most important for convergence during the inner GMRES iteration and to throw away the rest of $C_{n,k}$. This is done in [261] using the singular values of a certain $k \times m$ matrix. The vector associated to the smallest singular value is discarded. A simpler strategy was proposed in [520]. The authors just discard the oldest column in $C_{n,k}$. Their method is named GCROT(m, k) where m is the number of inner GMRES iterations and k is the truncation parameter. It is suggested to use $k \approx m$.

12.2.2 Historical notes

GCRO was initially proposed by Eric de Sturler and Diedrik Fokkema [260, 262] in 1993 and 1996. The truncation strategy chosen in GCROT was studied by E. de Sturler [261] in 1999. The simplified strategy was considered by Jason E. Hicken and David W. Zingg [520] in 2010.

A GCRO algorithm with deflated restarting was considered in [720] by Qiang Niu, Lin-Zhang Lu and Gang Liu in 2013.

12.3 Relaxation for matrix–vector products

In some problems it is possible to do the matrix–vector products inexactly at a cheaper cost. This is somehow related to Section 12.1 and known as *inexact* Krylov methods. If the accuracy of the matrix–vector products is carefully monitored, meaningful approximate solutions can still be obtained. The monitoring of the matrix–vector accuracy is known as a *relaxation strategy*.

This technique was first considered by Amina Bouras and Valérie Frayssé in a 2000 CERFACS technical report [114]. Unfortunately, the corresponding paper [115] was only published in 2005.

Following this technical report, some theoretical results about these techniques were published in 2003 by Valeria Simoncini and Daniel B. Szyld [834]; see also the paper [836] published in 2005.

Relaxation strategies were discussed by Gerard L.G. Sleijpen, Jasper Van den Eshof and Martin B. Van Gijzen in 2004 [844].

See also the paper [417] by Luc Giraud, Serge Gratton and Julien Langou about the convergence in backward error of relaxed GMRES in 2007.

12.4 Systems with multiple right-hand sides

12.4.1 Definition

The problem is to solve a linear system with multiple right-hand sides

$$AX = B, \quad (12.1)$$

where A is a real or complex matrix of order n and X, B are $n \times s$ matrices. In many problems, s is much smaller than n . The columns of the right-hand side B may be given one at a time or all known at the beginning of the solve. This has an influence on the method to be chosen.

Let us first assume that all the columns of B are given. There exist block versions of most Krylov methods that can be used to solve those linear systems. To solve equation (12.1) we first define block Krylov subspaces. The matrices generating the Krylov subspaces and the iterates X_k are often called *block vectors*. Let X_0 be a given $n \times s$ block vector and $R_0 = B - AX_0$. We define $\mathcal{K}_k^B(A, R_0)$ as the block span of the matrices $R_0, AR_0, \dots, A^{k-1}R_0$, that is,

$$\mathcal{K}_k^B(A, R_0) = \left\{ \sum_{j=0}^{k-1} A^j R_0 G_j \mid G_j \in \mathbb{R}^{s \times s}, j = 0, \dots, k-1 \right\}.$$

The block vector iterates are computed such that $X_k \in X_0 + \mathcal{K}_k^B(A, R_0)$. Let $x_k^{(j)}$ (resp. $r_k^{(j)}$) be the j th column of X_k (resp. R_k). Then, $x_k^{(j)} \in x_0^{(j)} + \mathcal{K}_k(A, R_0)$ where

$$\mathcal{K}_k(A, R_0) = \sum_{j=1}^s \mathcal{K}(A, r_0^{(j)}),$$

the subspaces in the sum being the usual Krylov subspaces. The column $x_k^{(j)}$ is sought from a subspace whose dimension is smaller than or equal to ks but usually larger than k . This is an advantage of the block methods compared to solving independently the s linear systems. The subspace $\mathcal{K}_k^B(A, R_0)$ is a Cartesian product of s subspaces,

$$\mathcal{K}_k^B(A, R_0) = \mathcal{K}_k(A, R_0) \times \cdots \times \mathcal{K}_k(A, R_0).$$

A block grade of R_0 with respect to A can be defined as the smallest integer k such that $X \in X_0 + \mathcal{K}_k^B(A, R_0)$ where X is the exact solution; see [492]. It is less than or equal to the degree of the minimal polynomial of A .

The main difficulty in generalizing the Krylov methods to the block case is that the s residual vectors may be linearly dependent. This problem is generally dealt with a process called *deflation*. It amounts to reduce the number of columns in the right-hand side of (12.1). We must warn the reader that the word *deflation* has several possible meanings in numerical linear algebra.

Deflation may or may not be useful when solving linear systems, depending on the chosen method, but it is mandatory when computing eigenvalues.

As an example of algorithm, let us consider the block GMRES algorithm. Its steps are the following:

1. Compute an orthogonal basis using the block Arnoldi process,
2. determine the coordinates of the s systems,
3. solve a least squares problem with s right-hand sides by updating the QR factorization of the new block Hessenberg matrix to which s rows and columns have been added.

The block Arnoldi process is a generalization of the Arnoldi process as follows, starting from A and R_0 .

```
[Y0, U] = rrqr(R0)
for k = 1:m
    AY = AY_{k-1}
    for j = 0:k-1
        H_{j,k-1} = Y_k^* AY
        AY = AY - Y_j H_{j,k-1}
    end % for j
    [Y_k, H_{k,k-1}] = rrqr(AY)
end % for k
```

rrqr is a rank-revealing QR factorization. In this process, deflation is done by deleting columns of Y_0 or Y_k which are multiplied by small entries. This may reduce the value of s . If the block GMRES algorithm is used with restarts, it may be enough to do the initial deflation.

Another approach for solving linear systems like (12.1) is to use *global algorithms*. Let Y and Z be two $n \times s$ real matrices, global methods use the dot product

$$\langle Y, Z \rangle_F = \text{trace}(Y^T Z).$$

Note that $\langle Y, Y \rangle_F$ is the square of the Frobenius norm of Y .

In the global GMRES algorithm the iterates are defined as

$$X_k = X_0 + \sum_{j=1}^k A^{j-1} R_0(\alpha_j I),$$

where α_j is a scalar coefficient defined by the orthogonality relation

$$R_k \perp_F \text{span}(AR_0, \dots, A^k R_0),$$

the F-orthogonality being defined by the $\langle \cdot, \cdot \rangle_F$ dot product. The method is implemented by computing an F-orthonormal basis using the global Arnoldi process. At each iteration we just need to solve a (small) least squares problem with a $(k+1) \times k$ matrix to obtain the new iterate.

The so-called *seed methods* are more suited for the cases where the columns of B are obtained sequentially one after the other. A first system with a single right-hand side is solved and the idea is to gather some information from this first solve to obtain approximate solutions for the next solves. One can, for instance, project the next problems on the subspaces obtained from the seed system or obtain information about the eigenvectors of A and use this in the subsequent solves. When the seed system has converged, a new seed system is chosen if there are still systems which have not converged. This type of method was first considered for symmetric matrices and then extended to the nonsymmetric case; see [208, 323, 796, 852].

Generalizations of this idea were used for solving linear systems $A^{(i)}x^{(i)} = b^{(i)}$ where the matrices $A^{(i)}$ do not vary too much from one system to the next. This is known as *recycling*, that is, reusing some information gathered from the previous systems.

12.4.2 Historical notes

Many papers have been published about solving linear systems with multiple right-hand sides. Therefore, we can only give references for a few of them.

The development of block Krylov methods started for symmetric matrices from the need of computing eigenvalues of multiplicity larger than one since the Lanczos method was unable to detect multiplicity; see the Ph.D. thesis of Richard Underwood [934] in 1975 and the joint paper [436] with Gene H. Golub in 1977.

Some of the papers related to multiple right-hand sides are:

- 1980: Block Krylov methods for solving linear systems were considered by Dianne O’Leary [722]. She introduced a block BiCG algorithm and derived a block conjugate gradient method from it. She also gave a convergence analysis of the block CG method.
- 1984: A block Arnoldi algorithm was introduced by Daniel Boley and G.H. Golub [107]; see also [108].
- 1990: The block GMRES algorithm was described in the Ph.D. thesis of Brigitte Vital [954].
- 1995: Algorithms for linear systems with multiple right-hand sides were considered by Valeria Simoncini and Efstratios Gallopoulos [830]. They published an analysis of the convergence of block GMRES based on matrix polynomials in 1996 [831]; see also [832].
- 1997: A QMR version of block BiCG was proposed in 1997 by V. Simoncini [826]. That same year Roland W. Freund and Manish Malhotra published a block QMR algorithm including deflation [377] based on the Lanczos-type process developed by José L. Aliaga, D. Boley, R.W. Freund and Vicente Hernández [16] in 1996 with the paper published in 2000.
- 1997: Block Krylov methods were discussed by Andrew Chapman and Yousef Saad [213].
- 1999: Zhaojun Bai, David Day and Qiang Ye introduced ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems [56].
- 1999: A global version of GMRES was described by Khalide Jbilou, Abderrahim Messaoudi and Hassane Sadok [557]. A global CMRH algorithm was considered in 2001 by Mohammed Heyouni [516].
- 2002: New two-sided Arnoldi algorithms were introduced by Jane Cullum and Tong Zhang [246] for applications in model reduction.
- 2003: Ahmed El Guennouni, K. Jbilou and H. Sadok published a block version of BiCGStab for linear systems with multiple right-hand sides [475] and a block Lanczos method [476] in 2004.
- 2006 Mickaël Robbé and Miloud Sadkane [771] studied exact and inexact breakdowns in the block GMRES method.
- 2009: Lakhdar Elbouyahyaoui, A. Messaoudi, and H. Sadok [306] considered the algebraic properties of the block GMRES and block Arnoldi methods; see also [117].

- 2010: S-step versions of block Krylov methods for the sake of parallelism were proposed by Anthony T. Chronopoulos and Andrey B. Kucherov [226].
- 2010: Application to unsteady Navier-Stokes equations by Mark H. Carpenter, Cornelis Vuik, Peter Lucas, Martin van Gijzen, and Hester Bijl [184].
- 2011 As for many other methods, a block version of IDR(s) was published by Lei Du, Tomohiro Sogabe, Bo Yu, Yusaku Yamamoto and Shao-Liang Zhang [283].
- 2013: Henri Calandra, Serge Gratton, Julien Langou, Xavier Pinel and Xavier Vasseur published a paper introducing flexible variants of block restarted GMRES methods [166]. This was followed in 2013 with a paper by H. Calandra, S. Gratton, Rafael Lago, X. Vasseur and Luiz M. Carvalho introducing a modified block flexible GMRES method with deflation [166].
- 2014: Emmanuel Agullo, Luc Giraud and Y-F. Jing proposed a block GMRES method with handling of inexact breakdowns and deflated restarting [8].
- 2014: Shusaku Saito, Hiroto Tadano and Akira Imakura published a block BiCGSTAB(ℓ) method [809].

A few papers related to seed methods are given below:

- 1989: A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields was proposed by Charles F. Smith, Andrew F. Peterson and Raj Mittra [852].
- 1997: Tony F. Chan and Wing Lok Wan analyzed projection methods for solving linear systems with multiple right-hand sides [208].
- 1997: Yousef Saad studied augmented Krylov subspace methods [796].
- 2000: Symmetric positive definite systems were considered by Jocelyne Erhel and Frederic Guyomarc'h [323].
- 2001: A block-seed algorithm using block QMR was published by Misha Kilmer, Eric Miller and Carey Rappaport [595].
- 2002: Gui-Ding Gu proposed a seed method for solving nonsymmetric linear systems with multiple right-hand sides [468].
- 2016: Khalide Jbilou wrote a note on the block and seed BiCGSTAB algorithms for multiple linear systems [556].

More recently, recycling techniques were devised. References to some papers are as follows:

- 2006: Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson and Spandan Maiti considered recycling Krylov subspaces for sequences of linear systems [748].
- 2012: Kapil Ahuja, Eric de Sturler, Serkan Gugercin and Eun R. Chang recycled BiCG with an application to model reduction [10]; see also [9] by K. Ahuja, Peter Benner, E. de Sturler and Lei Feng in 2015.
- 2014: The Ph.D. thesis of André Gaul was devoted to recycling Krylov subspace methods for sequences of linear systems [401].
- 2014: Kirk M. Soodhalter, Daniel B. Szyld and Fei Xue studied Krylov subspace recycling for sequences of shifted linear systems [869].

- 2015: Amit Amritkar, E. de Sturler, Katarzyna Swirydowicz, Danesh Tafti and K. Ahuja used recycling Krylov subspaces for CFD applications [19].
- 2016: M.L. Parks, K.M. Soodhalter and D.B. Szyld published a report on a block recycled GMRES method with investigations into aspects of solver performance [747].
- 2017: K.M. Soodhalter studied the stagnation of block GMRES and its relationship to block FOM [868].

Other papers related to solving systems with multiple right-hand sides are [1, 15, 61, 253, 491, 492, 517, 651, 698, 990, 1025, 1026].

12.5 Shifted linear systems

12.5.1 Definition

Shifted linear systems are of the form

$$(A + \mu I)x = b,$$

where μ is a real or complex number. In some practical problems, one has to solve a sequence of shifted linear systems $(A + \mu_i I)x^{(i)} = b$, $i = 1, \dots, m$. Krylov methods are attractive for solving such systems because of the shift-invariance property of Krylov subspaces,

$$\mathcal{K}_k(A + \mu I, v) \equiv \mathcal{K}_k(A, v).$$

If we have already solved $Ax = b$ or a shifted system with another shift, we can take advantage of the shift-invariance property to solve the new shifted system cheaply. How this can be done depends on the chosen Krylov method. Recycling techniques can be also related to this problem.

12.5.2 Historical notes

A selection of papers about Krylov methods for shifted linear systems is given below:

- 1993: QMR solutions of shifted linear systems were proposed by Roland W. Freund [368].
- 1998: Andreas Frommer and Uwe Glässner introduced restarted GMRES for shifted linear systems [388].
- 2003: BiCGStab(ℓ) for shifted linear systems was studied by A. Frommer [387].
- 2014: Kirk M. Soodhalter, Daniel B. Szyld and Fei Xue used recycling for sequences of shifted linear systems [869].

- 2015: Manuel Baumann and Martin B. van Gijzen considered nested Krylov methods for shifted linear systems [78].
- 2016: Two recursive GMRES-type methods for shifted linear systems were studied by K. Soodhalter [867] who also proposed block Krylov subspace recycling for shifted systems with unrelated right-hand sides [866].
- 2016: Xian-Ming Gu, Ting-Zhu Huang, Bruno Carpentieri, Akira Imakura, Ke Zhang and Lei Du proposed variants of the CMRH method for solving multi-shifted linear systems [471].

12.6 Singular systems

12.6.1 Definition

In this book we study Krylov methods for solving nonsingular linear systems. However, there are practical problems for which the matrix A is singular. An example is the study of Markov chains; see, for instance, [817]. Hence, some Krylov methods were applied to singular systems.

Most of the research on this topic was related to GMRES. Mathematically, for a nonsingular system, there is an index $m \leq n$ for which GMRES delivers the exact solution. When the matrix is singular, GMRES may break down before an acceptable approximate solution has been determined. Let i be the index of the zero eigenvalue. Let $x_0 = 0$, it was proved in [549] that GMRES produces a solution of the linear system $Ax = b$ if and only if b is in the range of A^i .

12.6.2 Historical notes

A selection of papers about Krylov methods for singular linear systems is as follows. Most papers are devoted to GMRES. Exceptions are [376] and [976].

- 1994: Roland W. Freund and Marlis Hochbruck proposed two QMR algorithms for solving singular systems with applications in Markov chain problems [376].
- 1997: Peter N. Brown and Homer F. Walker studied the properties of GMRES on (nearly) singular systems [154].
- 1998: A survey of properties of the GMRES method when applied to consistent linear systems with a singular matrix was provided by Ilse C.F. Ipsen and Carl D. Meyer [549].
- 1999: Laurent Smoch published results about GMRES in the singular case [854]; see also his Ph.D. thesis (in French) [853].
- 2000: Daniela Calvetti, Bryan Lewis and Lothar Reichel investigated under which conditions on A and b , a GMRES iterate is a least squares solution of $Ax = b$ [170]. They also introduced RRGMRES, a range restricted variant of GMRES

that, under some conditions, produces the minimal norm least squares solution. See also the paper [180] in 2002 by Zhi-Hao Cao and Mei Wang.

- 2000: Convergence properties of Krylov subspace methods for singular linear systems with arbitrary index is the topic of a paper by Yimin Wei and Hebing Wu published [976]. They also gave convergence bounds for QMR.
- 2005: L. Reichel and Qiang Ye discussed properties of GMRES solutions at breakdown and presents a modification of GMRES to overcome breakdowns [764].
- 2006: The thesis of Olaf Schneider was devoted to Krylov subspace methods and their generalizations for solving singular linear operator equations with applications to continuous time Markov chains [817].
- 2007: L. Smoch showed that the singularity of the Hessenberg matrix produced by GMRES applied to a singular system is linked to the spectral properties of A [855].
- 2008: Xiuhong Du and Daniel B. Szyld studied the application of inexact GMRES to singular linear systems [285].
- 2010: The solution of singular systems by Krylov subspace methods was considered by Man-Chung Yeung [1007].
- 2011: A geometric view of Krylov subspace methods on singular systems was provided by Ken Hayami and Masaaki Sugihara [505]; see some corrections in 2014 [506].
- 2012: Lars Eldén and Valeria Simoncini solved ill-posed linear systems with GMRES and a singular preconditioner [307].
- 2018: Keichi Morikuni and Miroslav Rozložník studied GMRES for singular EP and GP systems [705].

12.7 Least squares problems

12.7.1 Definition

A least squares problem is computing a solution of

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|,$$

where the $m \times n$ matrix A may be under- or overdetermined and may be not of full rank. For a thorough treatment of least squares problems, see Åke Björck's book [100]. A common technique for solving these problems is to apply the conjugate gradient method to the normal equations, $A^T Ax = A^T b$ expressed in such a way that the matrix $A^T A$ is not computed. This is known as CGLS.

When we wish to solve a linear system of the form $A^T Ax = A^T b$, we can use to our advantage the fact that the matrix of the linear system is such a product. An

elegant technique was devised by Gene H. Golub and William Kahan [429]. Their purpose was to compute the singular values of the matrix A .

Two different algorithms were proposed. The first one reduces A to an upper bidiagonal form and is given as follows:

Let $q_0 = b/\|b\|$, $r_0 = Aq_0$, $\delta_1 = \|r_0\|$, $p_0 = r_0/\delta_1$; then for $k = 1, 2, \dots$

$$u_k = A^T p_{k-1} - \delta_k q_{k-1},$$

$$\gamma_k = \|u_k\|,$$

$$q_k = u_k / \gamma_k,$$

$$r_k = Aq_k - \gamma_k p_{k-1},$$

$$\delta_{k+1} = \|r_k\|,$$

$$p_k = r_k / \delta_{k+1}.$$

If we denote

$$P_k = (p_0 \cdots p_{k-1}), \quad Q_k = (q_0 \cdots q_{k-1}),$$

and

$$B_k = \begin{pmatrix} \delta_1 & \gamma_1 & & \\ & \ddots & \ddots & \\ & & \delta_{k-1} & \gamma_{k-1} \\ & & & \delta_k \end{pmatrix},$$

then P_k and Q_k satisfy the equations

$$\begin{aligned} AQ_k &= P_k B_k, \\ A^T P_k &= Q_k B_k^T + \gamma_k q_k (e_k)^T. \end{aligned}$$

By eliminating P_k in these equations we obtain

$$A^T A Q_k = Q_k B_k^T B_k + \gamma_k \delta_k q_k (e_k)^T.$$

It shows that $B_k^T B_k = J_k$ is the Lanczos Jacobi matrix corresponding to $A^T A$. In fact, this algorithm has directly computed the Cholesky factorization of J_k .

This algorithm was devised to compute the singular values of A which are the square roots of the eigenvalues of $A^T A$. For solving linear systems or least squares problems, this algorithm, using a different starting vector $q_0 = A^T b / \|A^T b\|$, was considered also in a paper by Chris Paige and Michael Saunders [737, 739] under the name *Bidiag 2*.

A second algorithm named *Bidiag 1* in [737] reduces A to lower bidiagonal form. The steps are the following. Note that the coefficients δ_k and γ_k are different from the ones in the previous algorithm.

Let $p_0 = b/\|b\|$, $u_0 = A^T p_0$, $\gamma_1 = \|u_0\|$, $q_0 = u_0/\gamma_1$, $r_1 = Aq_0 - \gamma_1 p_0$, $\delta_1 = \|r_1\|$, $p_1 = r_1/\delta_1$; then for $k = 2, 3, \dots$

$$u_{k-1} = A^T p_{k-1} - \delta_{k-1} q_{k-2},$$

$$\gamma_k = \|u_{k-1}\|,$$

$$q_{k-1} = u_{k-1}/\gamma_k,$$

$$r_k = Aq_{k-1} - \gamma_k p_{k-1},$$

$$\delta_k = \|r_k\|,$$

$$p_k = r_k/\delta_k.$$

If we denote

$$P_{k+1} = (p_0 \cdots p_k), \quad Q_k = (q_0 \cdots q_{k-1}),$$

and

$$C_k = \begin{pmatrix} \gamma_1 & & & \\ \delta_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ddots & \gamma_k \\ & & & \delta_k \end{pmatrix},$$

a $(k+1) \times k$ matrix, then P_k and Q_k satisfy the equations

$$\begin{aligned} A Q_k &= P_{k+1} C_k, \\ A^T P_{k+1} &= Q_k C_k^T + \gamma_{k+1} q_k (e_{k+1})^T. \end{aligned}$$

By eliminating P_{k+1} , we obtain

$$A^T A Q_k = Q_k C_k^T C_k + \gamma_{k+1} \delta_k q_k (e_k)^T.$$

Therefore, we have

$$C_k^T C_k = B_k^T B_k = J_k,$$

and B_k is also the Cholesky factor of the product $C_k^T C_k$. This algorithm is the basis for the method LSQR of Paige and Saunders [737, 739], who used an incremental reduction of the matrix C_k to upper triangular form by using Givens rotations.

Another algorithm for solving least squares problems, based on the Golub–Kahan bidiagonalization, is LSMR [359]. For a discussion about these algorithms and solving skew-symmetric linear systems, see [300, 465].

However, some researchers proposed to use an additional $n \times m$ matrix B in the least squares problem and to solve

$$\min_{z \in \mathbb{R}^m} \|b - ABz\|,$$

involving a square matrix AB , with a Krylov method, particularly with GMRES.

12.7.2 Historical notes

We list only a few papers applying Krylov methods for solving least squares problems.

- 1991: Zhang-Shao Liang and Yoshio Oyanagi proposed to use Orthomin(k) for linear least squares problems [632].
- 1996: Cornelis Vuik, Agur G. Sevink and Gérard C. Herman considered a preconditioned Krylov subspace method for the solution of least squares problems in inverse scattering problems [960].
- 2008: Tokushi Ito and Ken Hayami used preconditioned GMRES methods for least squares problems [551].
- 2010: GMRES methods for least squares problems were further considered by K. Hayami, Jun-Feng Yin and T. Ito [507].
- 2013: Keichi Morikuni and K. Hayami studied inner-iteration Krylov subspace methods for overdetermined least squares problems [702]; see also their paper [703] in 2015 about rank-deficient least squares problems.

12.8 Ill-posed problems

12.8.1 Definition

There are practical problems where one wants to obtain a meaningful solution of a linear system $Ax = b$, but b is unknown and what is available is only

$$Ax = b + p,$$

where p denotes unknown perturbations of the data usually referred to as *noise*. Moreover, these problems are typically ill-posed, meaning that the singular values of A decay gradually to zero without a significant gap. These kinds of problems arise,

for instance, from the discretization of integral equations or from image processing problems.

Such problems cannot be solved with direct methods like LU or QR factorizations since the solutions are usually completely spoiled by the noise. A way to obtain an approximate solution is to use regularization methods like Tikhonov regularization or the truncated singular value decomposition (TSVD). However, some Krylov methods having regularization properties can also be used. Generally, the iterates start converging to a meaningful solution and after a while become to be polluted by the propagation of the noise. Hence, the problem is to know when to stop the iterations. To do so it is important to study the noise propagation in the vectors of the iterative method. It is also sometimes useful to be able to estimate the noise level, that is, the norm of p , even though p is unknown.

For these type of problems, see the books [502] and [503] by Per Christian Hansen as well as his many papers.

12.8.2 *Historical notes*

Below we are only concerned with papers related to Krylov methods applied to ill-posed problems:

- 2002: Daniela Calvetti, Bryan Lewis and Lothar Reichel studied the regularizing properties of GMRES [172]; see also [171].
- 2007: Iterative regularization with minimum-residual methods was considered by Toke K. Jensen and P.C. Hansen [562]. They also studied the noise propagation in regularizing iterations for image deblurring [504] in 2008.
- 2012: A generalized global Arnoldi method for ill-posed matrix equations was proposed by Abderrahman Bouhamidi, Khalide Jbilou, L. Reichel and Hassane Sadok [113].
- 2014: Keichi Morikuni, L. Reichel and Ken Hayami used FGMRES for linear discrete ill-posed problems [704].
- 2015: Mohammed Bellalij, L. Reichel and H. Sadok studied properties of range restricted GMRES methods [88].
- 2015: L. Reichel and Xuebo Yu considered Tikhonov regularization via flexible Arnoldi reduction [765].
- 2018: IR Tools, a MATLAB package of iterative regularization methods and large-scale test problems was proposed by Silvia Gazzola, P.C. Hansen and James G. Nagy [409].

12.9 Eigenvalue computations and rational Krylov methods

12.9.1 Definition

Iterative methods have to be used to compute approximations to eigenvalues λ and/or eigenvectors x which satisfy $Ax = \lambda x$, given a square real or complex matrix A . Thousands of papers, if not more, have been written on this topic. We refer to the book [794] by Yousef Saad.

Well-known methods are the Lanczos method for symmetric problems and Arnoldi's method for nonsymmetric problems. The approximations of the eigenvalues are obtained from the tridiagonal or the Hessenberg matrices computed by these methods. A method which is much used today is the implicitly restarted Arnoldi method [625]; see the ARPACK software.

Rational Krylov methods for computing eigenvalues can be seen as an extension of the shift-and-invert techniques. The Arnoldi process is modified as follows, given the unit norm starting vector v_1 :

```
for k = 1, ...
    v = (A - mu_k I)^-1 v_k
    h_k = V_k^* v, v = v - V_k h_k % GS orthogonalization
    h_{k+1,k} = ||v||, v_{k+1} = v / h_{k+1,k}
end
```

The values μ_k are the chosen shifts. Note that a linear system has to be solved at each iteration, so this cannot be used for solving linear systems $Ax = b$. The above pseudocode is only a basic version of rational Krylov. For more complete versions see [783–785].

Extended Krylov subspaces [280] also involve the inverse of A . Let ℓ and k be two integers, then

$$\mathcal{K}_{\ell,k}(A, v) = \text{span}(A^{-\ell+1}v, \dots, A^{-1}v, v, Av, \dots, A^{k-1}v),$$

see below the section on matrix functions.

12.9.2 Historical notes

A selection of papers is the following:

- 1950: Cornelius Lanczos introduced what is now known as the symmetric and nonsymmetric Lanczos methods for eigenvalue problems [615].
- 1957: Walter E. Arnoldi published his paper [35] about eigenvalue computations.
- 1980: Y. Saad considered some variations on Arnoldi's method [788] and projection methods [791] in 1983. The first version of his book on eigenvalue problems was published in 1992.

- 1992: Danny C. Sorensen studied the implicit application of polynomial filters in a k-step Arnoldi method [870]. This finally leads to the ARPACK software; see the Ph.D. thesis of Richard B. Lehoucq [623] in 1995. In 1996 they used deflation techniques for an implicitly restarted Arnoldi iteration [626].
- 2001: R.B. Lehoucq considered the relationship between the implicitly restarted Arnoldi method and subspace iteration [625].
- 2007: Mohammed Bellalij, Y. Saad and Hassane Sadok studied the convergence of the Arnoldi process [89]; see also their 2010 paper [91].
- 2013: Martin H. Gutknecht and Jens-Peter M. Zemke [494] showed how to obtain eigenvalue approximations using IDR.
- 2016: Exact expressions for the distance of an eigenvector to a Krylov subspace were obtained by M. Bellalij, G. Meurant and H. Sadok [87].

A list of a few papers on rational Krylov methods is given below:

- 1984: Rational Krylov sequence methods for eigenvalue computation were introduced by Axel Ruhe [782]; see also [783–785]
- 2000: A chapter written by A. Ruhe is part of the book *Templates for the solution of algebraic eigenvalue problems* by Zhaojun Bai, Jim Demmel, Jack Dongarra, A. Ruhe and Henk van der Vorst [57].
- 2010: The Ph.D. thesis of Stefan Güttel was about rational Krylov methods for operator functions [495].
- 2013: Thomas Mach, M.S. Pranić and R. Vandebril showed how to compute approximate extended Krylov subspaces without explicit inversion [653]; they considered the block version [654] in 2014.
- 2016: M. Pranić, Lothar Reichel, Giuseppe Rodriguez, Zejun Wang and Xuebo Yu considered a rational Arnoldi process with applications to matrix functions [758].

12.10 Functions of matrices

12.10.1 Definition

In some problems it is necessary to compute vectors $f(A)b$ where f is a given function well defined on A or scalars $u^T f(A)v$ where u and v are given vectors. For definitions of matrix functions, see Nicholas Higham's book [522].

Typical functions of interest are the inverse, the exponential, the logarithm, the sine and cosine. There exist methods for computing all the entries of $f(A)$ (see [522]) but they are too expensive when A is large and sparse. Moreover, for our purposes it is not necessary to compute $f(A)$. In this case, one can rely on Krylov iterative methods.

An interesting application where bilinear forms like $u^T f(A)v$ have to be evaluated is the analysis of complex networks, see [330].

12.10.2 Historical notes

Some papers related to the computation of expressions involving functions of matrices using Krylov methods are as follows:

- 1990: Computing functions of matrices using Krylov subspace methods was considered by Thomas Ericsson [324].
- 1992: Leonid Knizhnerman used Arnoldi's method for the calculation of functions of nonsymmetric matrices [601]; see also the paper with Vladimir Druskin [280] in 1998.
- 1994: Methods using quadrature rules for computing approximations of bilinear forms with A symmetric were studied by Gene H. Golub and G. Meurant [430]; see also the book [432] in 2010.
- 1997: Marlis Hochbruck and Christian Lubich obtained Krylov subspace approximations to the matrix exponential operator [525].
- 1999: Applications of anti-Gauss quadrature rules in linear algebra were considered by Daniela Calvetti, Lothar Reichel and Fiorella Sgallari [175].
- 2005: D. Calvetti, Sun-Mi Kim and L. Reichel proposed quadrature rules based on the Arnoldi process [169].
- 2008: Guillermo López Lagomasino, L. Reichel and Lena Wunderlich published a paper on matrices, moments, and rational quadrature [612].
- 2009: Carl Jagels and L. Reichel studied the extended Krylov subspace method and the relationship with orthogonal Laurent polynomials [553].
- 2010: Michele Benzi and Paola Boito developed quadrature rule-based bounds for functions of adjacency matrices in network analysis [93].
- 2010: L. Knizhnerman and Valeria Simoncini did a new investigation of the extended Krylov subspace method for matrix function evaluations [605].
- 2010: Network properties revealed through matrix functions were published by Ernesto Estrada and Desmond J. Higham [331].
- 2011: C. Jagels and L. Reichel studied recursion relations for the extended Krylov subspace method [554].
- 2011: Hongguo Xu gave formulas for functions of A in terms of Krylov matrices of A [993].
- 2012: The physics of communicability in complex networks [330] was published by Ernesto Estrada, Naomichi Hatano and M. Benzi.
- 2013: M. Benzi, E. Estrada and Christine Klymko showed how to rank hubs and authorities in network analysis using matrix functions [94].
- 2013: Network analysis via partial spectral factorization and Gauss quadrature was proposed by Caterina Fenu, David Martin, L. Reichel and Giuseppe Rodriguez [346].

12.11 Minimum error methods

12.11.1 Definition

As the name implies, minimum error methods aim to minimize the norm of the error in a given Krylov subspace. The method proposed in [978] looks for approximate solutions

$$x_k \in x_0 + A^T \mathcal{K}_k(A^T, r_0),$$

such that

$$\|x - x_k\| = \min_{y \in x_0 + A^T \mathcal{K}_k(A^T, r_0)} \|x - y\|.$$

The minimization condition is equivalent to $x - x_k$ being orthogonal to $A^T \mathcal{K}_k(A^T, r_0)$, that is, $r_k = b - Ax_k$ being orthogonal to $\mathcal{K}_k(A^T, r_0)$. The convergence of the error norm is monotonic, but the method does not have the finite termination property.

The implementation proposed in [779] computes an orthogonal basis of $A^T \mathcal{K}_k(A^T, r_0)$ and an AA^T -orthogonal basis of $\mathcal{K}_k(A^T, r_0)$. A stable implementation is obtained using Householder reflections and numerical examples are given.

Let $\varepsilon_k = x - x_k$. Then, $A\varepsilon_k = r_k$ and we observe that $\|\varepsilon_k\|_{A^T A} = \|r_k\|$. Therefore, GMRES minimizes the $A^T A$ -norm of the error.

12.11.2 Historical notes

- 1994: Error-minimizing Krylov subspace methods were proposed by Rüdiger Weiss [978].
- 1995: R. Weiss wrote a theoretical overview of Krylov subspace methods [980].
- 1998: Miroslav Rozložník and R. Weiss considered the stable implementation of the generalized minimal error method [779].

12.12 Residual replacement techniques

In the Krylov methods we have described in this book, the residual vectors r_k are generally computed by a recurrence. This is why, in finite precision arithmetic, they are called the *computed residuals*. The approximate solutions x_k are also computed by a recurrence. However, the rounding errors in these two recurrences are unrelated and it may happen that the computed residual becomes different from $b - Ax_k$ which is known as the *true residual*. In fact, what is available (but costly to compute) is the floating-point result of the computation of the true residual $fl(b - Ax_k)$. Remember

that the minimum value of $f\ell(\|f\ell(b - Ax_k)\|)$ that can be obtained is called the maximum attainable accuracy.

In some methods, like BiCG or BiCGStab, the differences between the computed and the true residual have an influence on the maximum attainable accuracy. So, strategies were devised to try to control these differences and to improve the maximum attainable accuracy. The idea is to replace the computed residual by the true residual at selected iterations. Of course, the main problem is to know when to do it. It must not be done too often because it is costly and doing it blindly may harm the convergence of the method.

Following ideas from A. Neumaier at the beginning of the 1990s, Gerard L.G. Sleijpen and Henk van der Vorst proposed strategies for product-type methods like BiCGStab, see [846, 848]. Estimating the attainable accuracy of recursively computed residual methods was studied by Anne Greenbaum [454] in 1997.

Let us consider what was proposed in 2000 by H. van der Vorst and Qiang Ye [945]. We do not give the details of the derivation but just the algorithm recommended in that paper with residual replacement and group updating. It is assumed for simplicity that the approximate solution and the computed residual are computed as

$$x_k = x_{k-1} + q_k, \quad r_k = r_{k-1} - Aq_k.$$

Let $r_0 = b - Ax_0$, $x = x_0$, $\hat{x}_0 = 0$ and $d_{init} = d_0 = u(\|r_0\| + m\|A\| \|x_0\|)$. Then, for $k = 1, 2, \dots$ until convergence we compute

$$\hat{x}_k = \hat{x}_{k-1} + q_k, \quad r_k = r_{k-1} - Aq_k,$$

$$d_k = d_{k-1} + um\|A\| \|\hat{x}_k\| + u\|r_k\|,$$

if $d_{k-1} \leq \epsilon\|r_{k-1}\|$, $d_k > \epsilon\|r_k\|$ and $d_k > 1.1d_{init}$, we do a residual replacement as

$$x = x + \hat{x}_k, \quad \hat{x}_0 = 0, \quad r_k = b - Ax, \quad d_{init} = d_k = u(\|r_k\| + m\|A\| \|x\|).$$

In the end, when the iterates have converged, the final approximate solution is $x = x + \hat{x}_k$.

In this algorithm m is supposed to be the maximum number of nonzero entries in a row of A . However, it seems that, in practice, a value $m = 1$ is satisfactory. The number ϵ is a user-defined threshold that can be taken equal to \sqrt{u} where u is the unit roundoff. We observe that we also need $\|A\|$ but one can use the ∞ -norm or an estimate of the Euclidean norm if one is available. An analysis providing a theoretical basis for this strategy and numerical experiments are given in [945].

Residual replacement techniques were also considered as remedies for the potential instability of some parallel variants of Krylov methods like CA-BiCG or CA-BiCGStab; see [188, 192, 527].

Residual replacement was also seen as a way to cure the instability of the communication-hiding pipelined BiCGStab, even though only a very simple strategy using periodic replacement was proposed; see [233, 235].

12.13 Residual smoothing techniques

12.13.1 Definition

Some of the Krylov methods we have studied in this book, in particular the biconjugate gradient (BiCG) method and its squared version BiCGS, produce residual vectors whose norms oscillate rather strongly. Large intermediate residual norms are prone to reduce the maximum attainable accuracy or even to spoil convergence. One way that has been suggested to obtain less oscillating residual norms is residual smoothing. It amounts to compute new sequences of iterates x_k^s and of corresponding residual vectors r_k^s such that

$$x_k^s = (1 - \sigma_k)x_{k-1}^s + \sigma_k x_k, \quad r_k^s = (1 - \sigma_k)r_{k-1}^s + \sigma_k r_k,$$

with $x_0^s = x_0$, $r_0^s = r_0$ and the scalar σ_k computed to minimize the norm of the residual,

$$\sigma_k = \frac{(r_{k-1}^s, r_{k-1}^s - r_k)}{\|r_{k-1}^s - r_k\|^2}.$$

For other possibilities, see [489, 1034].

12.13.2 Historical notes

- 1987: Smoothing techniques were described in the book by Willi Schönauer [818].
- 1995: Lu Zhou and Homer F. Walker considered residual smoothing techniques for iterative methods [967, 1034].
- 2001; Mohammed Heyouni and Hassane Sadok published a variable smoothing procedure for Krylov subspace methods [518].
- 2001: Martin H. Gutknecht and Miroslav Rozložník studied if the residual smoothing techniques improve the limiting accuracy of iterative solvers [489]. Unfortunately, the answer was that they don't; see also [488].

12.14 Hybrid methods

Hybrid methods may have different meanings when being used about solving linear systems. Here we consider the methods where a first algorithm is used to obtain some information about the (spectral) properties of the matrix, and then this information is used in a second algorithm. The assumption underlying this idea is that algorithms of the first type cost more per iteration than those of the second type, so that a switch from the one to the other is potentially beneficial.

A typical example is to first use the Arnoldi process or some iterations of GMRES to obtain approximation of the eigenvalues of A (the so-called Ritz values) and then to use this in a Chebyshev-like algorithm by constructing a polynomial and applying it (may be in a cyclic fashion) to obtain an approximation of the solution of the linear system.

This idea was used in 1978 by Thomas A. Manteuffel [657]. He used a modified power iteration to obtain estimates of the eigenvalues, enclosed them in an ellipse and then used a Chebyshev iteration.

Howard C. Elman, Yousef Saad and Paul E. Saylor [311] used also an ellipse enclosing the Ritz values obtained from the Arnoldi process and Chebyshev polynomials. H.C. Elman and Roy L. Streit [313] considered the union of convex hulls of subsets of the Ritz values and an L_∞ optimal approximation polynomial. Dennis C. Smolarski and P.E. Saylor [858] enclosed the Ritz values in a polygon and used a least squares polynomial. This was also used by Y. Saad [793] but the least squares polynomial was computed in a different and more stable way using modified moments.

Hillel Tal-Ezer [904] used polynomial approximations in the complex plane. For polynomial methods, see also William B. Gragg and Lothar Reichel [444].

Noël M. Nachtigal, L. Reichel and Lloyd N. Trefethen [712] used a different approach. They ran GMRES until the residual norm drops by a suitable factor, and then re-apply the polynomial implicitly constructed in the first step by GMRES in a Richardson-type iteration. They explicitly computed the coefficients of the polynomial, factored the polynomial and used the Richardson iteration with a Leja ordering of the roots. They also discussed the relevance of the (estimates of the) eigenvalues in this context. Of course, now we know that they could have directly computed the harmonic Ritz values which are the roots of the GMRES residual polynomial. We observe that this technique can be used for any Q-OR or Q-MR method for which we can compute the roots of the residual polynomials.

In 1993 Gerhard Starke and Richard S. Varga [875] used the Arnoldi process, a polygon, a conformal map, Faber polynomials and a Richardson iteration.

In 1994, Michela Redivo-Zaglia and Claude Brezinski also proposed some hybrid procedures [132].

12.15 CGNE and CGNR

Another possibility to solve $Ax = b$ when A is nonsymmetric is to symmetrize the problem and to apply the conjugate gradient (CG) algorithm. There are two ways to do this.

The method known as CGNR uses CG on the normal equations $A^T Ax = A^T b$. It minimizes the norm of the residual on the Krylov subspace $\mathcal{K}_k(A^T A, A^T r_0)$. Note that this is related to what we have seen for solving least squares problems in Section 12.7.

The method known as CGNE uses CG on $AA^T y = b$, $x = A^T y$. It minimizes the norm of the error on the Krylov subspace $\mathcal{K}_k(AA^T, r_0)$.

The main drawback of these methods is that the condition numbers of $A^T A$ and AA^T can be much worse than the condition number of A . For more details on these methods, see, for instance, [71, 673].

12.16 USYMLQ and USYMQR

Two methods, named USYMLQ and USYMQR, were proposed in [812] by Michael A. Saunders, Horst D. Simon and Elizabeth L. Yip in 1988. We did not consider them in the previous chapters because, stricto sensu, they are not Krylov methods. They are both based on a reduction of A to tridiagonal form as $P^T A Q = T$ where P and Q are orthogonal matrices and T is tridiagonal. Let p_j and q_j be the columns of P and Q and $p_0 = 0$, $q_0 = 0$, $p_1 = b/\beta$, $q_1 = c/\gamma_1$ with $\beta_1 = \|b\|$, $\gamma_1 = \|c\|$, c being a given vector. Then, the columns of P and Q are computed by

$$\begin{aligned}\beta_{j+1} p_{j+1} &= A q_j - \alpha_j p_j - \gamma_j p_{j-1}, \\ \gamma_{j+1} q_{j+1} &= A^T p_j - \alpha_j q_j - \beta_j q_{j-1},\end{aligned}$$

with $\alpha_j = p_j^T A q_j$ and β_{j+1} and γ_{j+1} are chosen such that p_{j+1} and q_{j+1} are of unit norm. In matrix form, these relations can be written as

$$\begin{aligned}A Q_k &= P_k T_k + \beta_{k+1} p_{k+1} e_k^T, \\ A^T P_k &= Q_k T_k^T + \gamma_{k+1} q_{k+1} e_k^T,\end{aligned}$$

where the matrix T_k is tridiagonal with the α_j 's on the diagonal, the β_j 's on the first subdiagonal and the γ_j 's on the first upper diagonal. The vectors p_j and q_j do not live in a proper Krylov subspace since, for instance,

$$p_{2k} \in \text{span}(b, AA^T b, \dots, (AA^T)^{k-1} b, Ac, AA^T Ac, \dots, (AA^T)^{k-1} Ac).$$

Nevertheless, the previous relations can be used to derive methods which look like Q-OR or Q-MR. For instance, if we denote

$$\underline{T}_k = \begin{pmatrix} T_k \\ \beta_{k+1} e_k^T \end{pmatrix},$$

we can define iterates $x_k = x_0 + Q_k y_k$ where y_k is the solution of the problem

$$\min_y \|\beta_1 e_1 - \underline{T}_k y\|.$$

This minimizes the residual norm. Like in the three-term recurrence version of QMR, we can reduce \underline{T}_k to upper triangular form to solve the least squares problem and to obtain a short recurrence for x_k ; see Chapter 8. It yields USYMQR. The other method is obtained by using an LQ factorization with L lower triangular.

Chapter 13

Numerical comparisons of methods



In this chapter we compare numerically some of the methods we have studied in the previous chapters. We chose the methods which seem the most interesting ones.

13.1 Introduction

In the numerical results of this chapter, GMRES(m) corresponds to the modified Gram–Schmidt version of restarted GMRES using the last iterate of the previous cycle to restart. The stopping criterion is $\|r_k\| \leq \epsilon \|b\|$ and we use $x_0 = 0$. In the tables, nit is the number of iterations, mv is the number of matrix–vector products, the next two columns are the relative true residual norm and the relative error norm (computed with $x = A \backslash b$) at the end of the computation. We do at most $nitmax$ iterations. Note that the stopping criterion uses the norm of the computed (or updated) residual when the true residual norms are shown in the tables.

We start by considering some of our test problems without preconditioning. We observe in the numerical results that the method giving the smallest residual or error norms is not always the same, it depends on the problem. It also depends on the choice of the threshold ϵ in the stopping criterion. Hence, in this sense, there is no overall best method. But, this does not come as a surprise. Moreover, one has also to consider the computing time but, quite often, the fastest method is not the one giving the smallest error norm.

Note that when the computing times are small they are not very reliable. Even when they are larger, different runs may give slightly different computing times. Hence, these computing times, even if they give an interesting information, have to be considered with caution.

13.2 Small matrices

We start by considering some small matrices that we have already used in the previous chapters. Their characteristics are described in Appendix A. The values of m that we use in GMRES(m) and CMRH(m) are adapted to the problem.

For the problem `fs 183 6` it is necessary to use small values of ϵ . Otherwise the error norms are too large. This happens because $\|b\|$ is large. For the other problems we only use $\epsilon = 10^{-6}$ and 10^{-10} . The numbers in boldface show what is the best result in each column (Tables 13.1, 13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9 and 13.10).

Table 13.1 `fs 183 6`, $n = 183$, $\epsilon = 10^{-10}$, $\text{nitmax} = 1000$

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	35	35	9.35563e-11	1.77799e-01	0.0063
GMRES(5)	1000	1000	1.59399e-07	2.75431e+00	0.0310
GMRES(10)	1000	1000	5.40795e-08	5.01202e+00	0.0338
GMRES(20)	1000	1000	1.45782e-09	9.61426e+00	0.0444
CMRH	35	35	2.36369e-10	1.99232e-01	0.0031
CMRH(5)	1000	1000	1.15897e-05	1.10857e+00	0.0593
CMRH(10)	1000	1000	6.51155e-07	1.79613e+00	0.0568
CMRH(20)	1000	1000	1.46145e-09	9.39504e+00	0.0563
BiCG	588	1176	4.53035e-11	6.65465e-05	0.0108
BiCGStab	495	990	9.87447e-11	4.24828e-02	0.0121
BiCGStab2	428	1713	3.61574e-12	1.72341e-03	0.0176
BiCGStab(2)	729	2917	6.61426e-11	2.49796e-02	0.0716
BiCGStab(4)	307	2457	5.46405e-11	1.43185e-02	0.0775
IDR Bio(2)	127	382	6.77238e-11	5.47988e-03	0.0096
IDR Bio(4)	38	191	9.05633e-11	1.72382e-02	0.0060
IDR QMR(2)	177	532	1.96814e-11	8.46531e-03	0.0275
IDR QMR(4)	32	161	1.25333e-11	7.23824e-03	0.0097
IDR porth QMR(2)	899	902	1.36933e-12	3.89511e-04	0.4262
IDR porth QMR(4)	213	218	1.20268e-12	5.99092e-04	0.4238

Table 13.2 `fs_183_6`, $n = 183$, $\epsilon = 10^{-16}$, nitmax = 2000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	50	50	1.07937e-16	2.98176e-08	0.0193
GMRES(5)	2000	2000	1.59399e-07	2.75431e+00	0.0599
GMRES(10)	2000	2000	5.40795e-08	5.01202e+00	0.0665
GMRES(20)	2000	2000	1.45782e-09	9.61426e+00	0.0882
CMRH	50	50	4.62809e-17	5.06453e-08	0.0035
CMRH(5)	2000	2000	1.45800e-05	1.06899e+00	0.1165
CMRH(10)	2000	2000	7.28117e-07	1.83192e+00	0.1112
CMRH(20)	2000	2000	1.52060e-09	9.34883e+00	0.1105
BiCG	1082	2164	3.97028e-15	2.10408e-07	0.0191
BiCGStab	1066	2132	6.15737e-16	6.99543e-08	0.0250
BiCGStab2	1114	4457	3.48358e-15	6.08535e-08	0.0453
BiCGStab(2)	1489	5957	6.56410e-16	7.16351e-08	0.1452
BiCGStab(4)	837	6697	8.99614e-14	1.08400e-04	0.2109
IDR Bio(2)	667	2002	1.42838e-15	1.56933e-07	0.0462
IDR Bio(4)	80	401	1.32498e-15	2.04227e-07	0.0109
IDR QMR(2)	2000	6001	2.71094e-14	2.75483e-06	0.3027
IDR QMR(4)	67	336	1.00710e-14	5.09302e-06	0.0189
IDR porth QMR(2)	899	902	1.36933e-12	3.89511e-04	0.4432
IDR porth QMR(4)	213	218	1.20268e-12	5.99092e-04	0.4092

Table 13.3 `fs_680_1c`, $n = 680$, $\epsilon = 10^{-6}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	60	60	7.68674e-07	4.29922e-05	0.0184
GMRES(20)	1000	1000	4.68858e-03	1.56067e+00	0.0580
GMRES(40)	72	72	8.13486e-07	5.47481e-05	0.0062
GMRES(60)	60	60	7.68674e-07	4.29922e-05	0.0074
CMRH	60	60	4.73319e-06	6.13787e-04	0.0150
CMRH(20)	1000	1000	1.03584e-02	2.10098e+00	0.1012
CMRH(40)	106	106	2.50789e-06	5.02044e-05	0.0096
CMRH(60)	60	60	4.73319e-06	6.13787e-04	0.0057
BiCG	84	168	1.65687e-07	1.03119e-07	0.0026
BiCGStab	47	94	7.20544e-08	1.93904e-06	0.0015
BiCGStab2	26	105	3.00610e-08	7.39330e-07	0.0015
BiCGStab(2)	25	101	5.70351e-08	1.54457e-06	0.0033
BiCGStab(4)	13	105	1.45303e-09	3.42328e-08	0.0032
IDR Bio(2)	31	94	5.12348e-08	9.29825e-07	0.0035
IDR Bio(4)	16	81	1.28284e-07	3.28474e-06	0.0032
IDR QMR(2)	29	88	7.21275e-08	1.52767e-06	0.0058
IDR QMR(4)	15	76	1.44609e-07	2.48029e-06	0.0058
IDR porth QMR(2)	73	76	1.82407e-06	2.96464e-05	0.0147
IDR porth QMR(4)	68	73	2.59916e-06	1.50929e-04	0.0206

Table 13.4 `fs_680_1c`, $n = 680$, $\epsilon = 10^{-10}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	86	86	4.48258e-11	4.47560e-09	0.0198
GMRES(20)	1000	1000	4.68858e-03	1.56067e+00	0.0577
GMRES(40)	135	135	9.72023e-11	5.30382e-09	0.0115
GMRES(60)	107	107	7.77223e-11	3.67117e-09	0.0129
CMRH	86	86	1.20296e-10	7.63621e-09	0.0145
CMRH(20)	1000	1000	1.03584e-02	2.10098e+00	0.0926
CMRH(40)	171	171	3.99712e-10	1.95853e-08	0.0165
CMRH(60)	115	115	3.94276e-10	3.96986e-08	0.0162
BiCG	97	194	6.72317e-11	3.85025e-11	0.0037
BiCGStab	57	114	7.29951e-11	1.66039e-09	0.0028
BiCGStab2	31	125	9.59722e-11	2.69600e-09	0.0027
BiCGStab(2)	30	121	4.55936e-11	5.04767e-10	0.0040
BiCGStab(4)	15	121	5.81437e-11	1.04791e-09	0.0038
IDR Bio(2)	39	118	6.60991e-11	1.17936e-09	0.0040
IDR Bio(4)	22	111	4.01879e-13	7.55173e-12	0.0045
IDR QMR(2)	40	121	1.76472e-12	1.86067e-11	0.0081
IDR QMR(4)	21	106	1.83000e-12	1.32178e-11	0.0078
IDR porth QMR(2)	110	113	2.33862e-10	4.83049e-09	0.0166
IDR porth QMR(4)	101	106	9.24183e-11	1.44997e-09	0.0190

Table 13.5 supg 001, $n = 1225$, $\epsilon = 10^{-6}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	44	44	4.44231e-07	8.07612e-08	0.0138
GMRES(20)	121	121	7.05478e-07	1.60582e-06	0.0130
GMRES(40)	61	61	9.95552e-07	3.34063e-06	0.0089
GMRES(50)	44	44	4.44231e-07	8.07612e-08	0.0058
CMRH	43	43	2.25139e-06	4.51633e-07	0.0227
CMRH(20)	344	344	8.30342e-06	1.80302e-05	0.0399
CMRH(40)	44	44	1.03730e-06	8.33448e-07	0.0056
CMRH(50)	43	43	2.25139e-06	4.51633e-07	0.0059
BiCG	425	850	8.68761e-07	1.96862e-07	0.0166
BiCGStab	42	84	2.78917e-07	1.74040e-07	0.0020
BiCGStab2	19	77	7.19754e-07	7.15340e-07	0.0017
BiCGStab(2)	20	81	1.05781e-08	9.95892e-09	0.0039
BiCGStab(4)	10	81	2.37991e-08	2.19722e-08	0.0038
IDR Bio(2)	20	61	3.89155e-07	1.11735e-07	0.0032
IDR Bio(4)	11	56	8.52731e-08	8.12540e-08	0.0035
IDR QMR(2)	20	61	1.79957e-07	7.29504e-08	0.0056
IDR QMR(4)	11	56	1.90402e-08	1.16762e-08	0.0056
IDR porth QMR(2)	62	65	1.45218e-09	4.91200e-10	0.0156
IDR porth QMR(4)	54	59	1.76387e-08	8.96124e-09	0.0134

Table 13.6 $\text{supg } \mathbb{0}\mathbb{0}1$, $n = 1225$, $\epsilon = 10^{-10}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	51	51	2.72477e-11	5.69998e-12	0.0149
GMRES(20)	222	222	9.99466e-11	5.67861e-11	0.0204
GMRES(40)	79	79	4.02588e-11	9.97803e-12	0.0094
GMRES(50)	52	52	2.58892e-11	6.33193e-12	0.0104
CMRH	51	51	4.65610e-11	1.19753e-11	0.0241
CMRH(20)	500	500	4.05485e-10	1.15495e-10	0.0643
CMRH(40)	78	78	8.28727e-11	5.22387e-11	0.0137
CMRH(50)	51	51	1.50930e-10	2.99320e-11	0.0091
BiCG	564	1128	2.53281e-10	9.39832e-11	0.0223
BiCGStab	48	96	1.71658e-11	1.30808e-11	0.0022
BiCGStab2	25	98	1.74800e-11	5.19699e-12	0.0022
BiCGStab(2)	22	89	2.51079e-11	1.68968e-11	0.0040
BiCGStab(4)	13	105	4.64001e-11	2.02540e-11	0.0052
IDR Bio(2)	22	67	2.39793e-11	1.34307e-11	0.0033
IDR Bio(4)	12	61	5.11849e-11	1.59218e-11	0.0041
IDR QMR(2)	22	67	1.68447e-11	6.40034e-12	0.0062
IDR QMR(4)	12	61	1.55806e-11	1.07560e-11	0.0062
IDR porth QMR(2)	67	70	1.47803e-12	8.22710e-13	0.0166
IDR porth QMR(4)	63	68	3.94360e-13	1.25302e-13	0.0137

Table 13.7 diff conv 400, $n = 400$, $\epsilon = 10^{-6}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	64	64	9.34597e-07	1.29925e-06	0.0123
GMRES(5)	153	153	9.95639e-07	8.08171e-06	0.0055
GMRES(10)	114	114	9.52603e-07	7.50727e-06	0.0043
GMRES(20)	97	97	8.79895e-07	4.46602e-06	0.0048
CMRH	62	62	4.01404e-06	1.63207e-05	0.0071
CMRH(5)	138	138	9.87806e-06	8.44446e-05	0.0107
CMRH(10)	130	130	4.94416e-06	3.76307e-05	0.0091
CMRH(20)	94	94	6.54720e-06	5.79123e-05	0.0066
BiCG	79	158	8.68421e-07	2.09531e-07	0.0018
BiCGStab	43	86	6.00283e-07	1.81650e-06	0.0013
BiCGStab2	22	89	4.37391e-07	1.36138e-06	0.0012
BiCGStab(2)	22	89	3.18007e-07	1.32979e-06	0.0026
BiCGStab(4)	11	89	2.24742e-07	5.49465e-07	0.0022
IDR Bio(2)	29	88	4.71265e-07	1.06431e-06	0.0030
IDR Bio(4)	16	81	2.96615e-07	2.16237e-06	0.0028
IDR QMR(2)	29	88	1.03251e-07	1.80277e-07	0.0057
IDR QMR(4)	16	81	2.21392e-07	1.50808e-06	0.0053
IDR porth QMR(2)	81	84	1.25467e-06	7.15442e-06	0.0183
IDR porth QMR(4)	74	79	7.12800e-07	1.35496e-06	0.0198

Table 13.8 diff conv 400, $n = 400$, $\epsilon = 10^{-10}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	92	92	8.38054e-11	6.22999e-11	0.0192
GMRES(5)	216	216	6.72436e-11	3.78889e-10	0.0086
GMRES(10)	184	184	9.57857e-11	7.98660e-10	0.0071
GMRES(20)	167	167	8.99476e-11	6.13155e-10	0.0083
CMRH	89	89	6.92040e-10	1.89143e-09	0.0097
CMRH(5)	248	248	8.13896e-10	6.90263e-09	0.0174
CMRH(10)	228	228	9.42316e-10	7.89766e-09	0.0148
CMRH(20)	187	187	8.29193e-10	5.20718e-09	0.0120
BiCG	103	206	8.40889e-11	5.79448e-11	0.0023
BiCGStab	66	132	5.69084e-11	8.45503e-11	0.0019
BiCGStab2	33	133	5.42136e-12	2.68088e-11	0.0016
BiCGStab(2)	33	133	2.01138e-11	7.44255e-11	0.0038
BiCGStab(4)	17	137	4.30427e-12	7.35725e-12	0.0032
IDR Bio(2)	38	115	4.98679e-11	1.75291e-10	0.0036
IDR Bio(4)	22	111	1.65257e-11	8.90260e-11	0.0037
IDR QMR(2)	41	124	1.08386e-12	2.48178e-12	0.0072
IDR QMR(4)	22	111	1.46696e-11	6.40913e-11	0.0070
IDR porth QMR(2)	112	115	8.87382e-11	2.53151e-10	0.0206
IDR porth QMR(4)	106	111	8.09430e-11	1.96687e-10	0.0221

Table 13.9 raeefsky1, $n = 3242$, $\epsilon = 10^{-6}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	197	197	9.29913e-07	1.51014e-06	0.3589
GMRES(20)	1000	1000	1.63975e-03	1.94928e-01	0.4257
GMRES(130)	703	703	9.97693e-07	1.06673e-04	0.8351
GMRES(160)	311	311	9.87864e-07	1.02524e-04	0.4305
CMRH	187	187	9.41872e-06	7.60299e-06	0.3374
CMRH(20)	1000	1000	1.32027e-03	1.49757e-01	0.4981
CMRH(130)	783	783	1.42232e-05	1.35034e-03	0.9801
CMRH(160)	371	371	1.60881e-05	1.59039e-03	0.5001
BiCG	219	438	7.72136e-07	1.26847e-07	0.1321
BiCGStab	159	318	8.81653e-07	6.99388e-06	0.0881
BiCGStab2	77	306	8.11753e-07	2.54554e-05	0.0852
BiCGStab(2)	85	341	6.05123e-07	1.88937e-06	0.1721
BiCGStab(4)	39	313	3.83513e-07	2.10715e-05	0.1414
IDR Bio(2)	105	316	9.44109e-07	4.13154e-07	0.1012
IDR Bio(4)	51	256	4.67302e-07	1.43544e-05	0.0869
IDR QMR(2)	104	313	1.01789e-07	2.39598e-07	0.1122
IDR QMR(4)	53	266	1.98470e-07	7.94974e-07	0.0993
IDR porth QMR(2)	405	408	6.91157e-11	1.32028e-10	0.2527
IDR porth QMR(4)	342	347	1.96241e-11	7.17557e-11	0.2653

Table 13.10 *raefsky1*, $n = 3242$, $\epsilon = 10^{-10}$, nitmax = 1000

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	267	267	7.72622e-11	2.62875e-11	0.6160
GMRES(20)	1000	1000	1.63975e-03	1.94928e-01	0.4360
GMRES(130)	1000	1000	1.42769e-08	1.63735e-06	1.2173
GMRES(160)	557	557	9.88691e-11	8.87592e-09	0.7833
CMRH	260	260	1.03710e-09	7.07006e-09	0.5848
CMRH(20)	1000	1000	1.32027e-03	1.49757e-01	0.4870
CMRH(130)	1000	1000	5.37773e-06	4.57654e-04	1.2743
CMRH(160)	804	804	1.10880e-09	6.16869e-08	1.2019
BiCG	283	566	5.98909e-11	9.60930e-10	0.1772
BiCGStab	238	476	4.21399e-11	3.37377e-11	0.1418
BiCGStab2	102	409	7.61647e-11	7.66048e-10	0.1145
BiCGStab(2)	120	481	6.74960e-11	5.60790e-11	0.2421
BiCGStab(4)	48	385	9.11203e-11	4.36091e-09	0.1814
IDR Bio(2)	134	403	8.54504e-11	1.56107e-10	0.1275
IDR Bio(4)	69	346	2.11578e-11	7.49944e-11	0.1079
IDR QMR(2)	138	415	2.54352e-11	6.04191e-11	0.1570
IDR QMR(4)	69	346	2.76991e-11	8.38975e-11	0.1424
IDR porth QMR(2)	410	413	6.59493e-11	1.42797e-11	0.2557
IDR porth QMR(4)	372	377	7.77622e-12	1.16390e-11	0.2778

Figures 13.1, 13.2, 13.3 and 13.4 display results for the small matrices listed in Appendix A. The x -axis gives the problem number and the y -axis is related to 15 iterative methods as listed in the left part of the figures. GMRES (resp. IDR QMR(4)) corresponds to 15 (resp. 1) on the y -axis. For each problem we plot a “*” for the method with the smallest number of matrix–vector products or the minimum computing time. Remember that these computing times are not very reliable and may change slightly from one run to the next. We do this for $\epsilon = 10^{-6}$ and 10^{-10} where the stopping criterion is $\|r_k\| \leq \epsilon \|b\|$. We do not use a preconditioner and the initial guess is $x_0 = 0$.

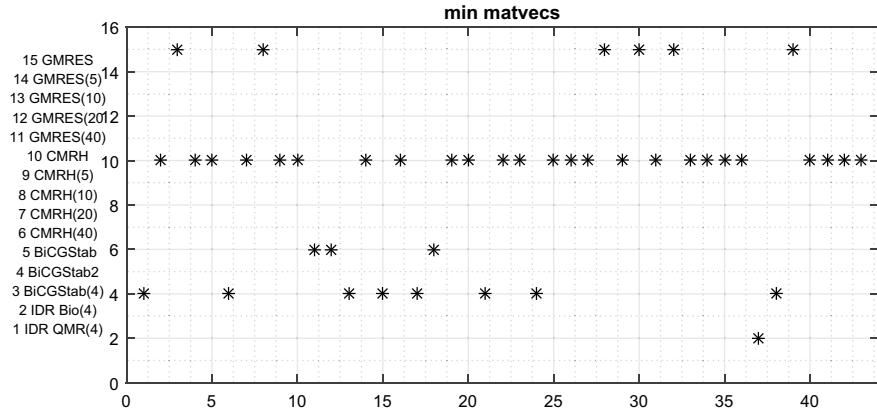


Fig. 13.1 Small matrices, minimum number of matvecs, $\epsilon = 10^{-6}$

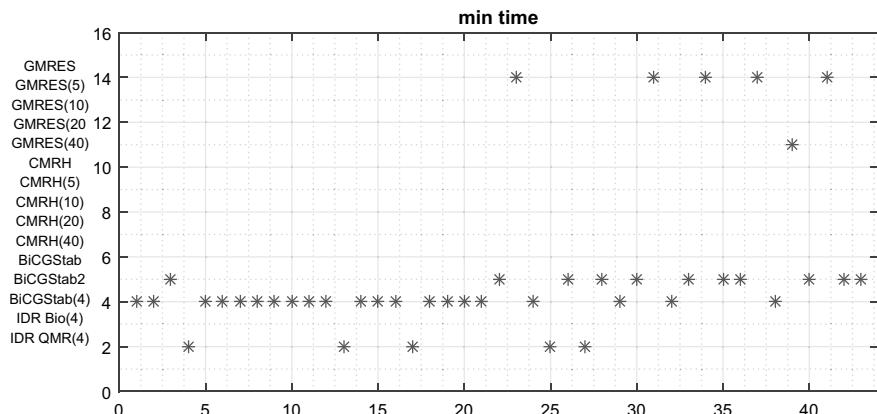


Fig. 13.2 Small matrices, minimum computing time, $\epsilon = 10^{-6}$

With $\epsilon = 10^{-6}$ the minimum number of matrix–vector products is given by full CMRH (corresponding to $y = 10$) for 25 problems. BiCGStab2 (corresponding to $y = 4$) gives the minimum for 8 problems. Things are different for the computing times. The minimum computing times are generally obtained for methods with short recurrences.

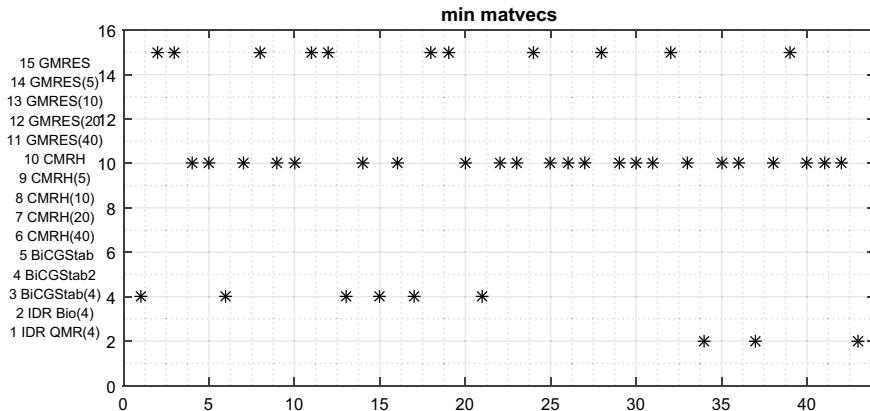


Fig. 13.3 Small matrices, minimum number of matvecs, $\epsilon = 10^{-10}$

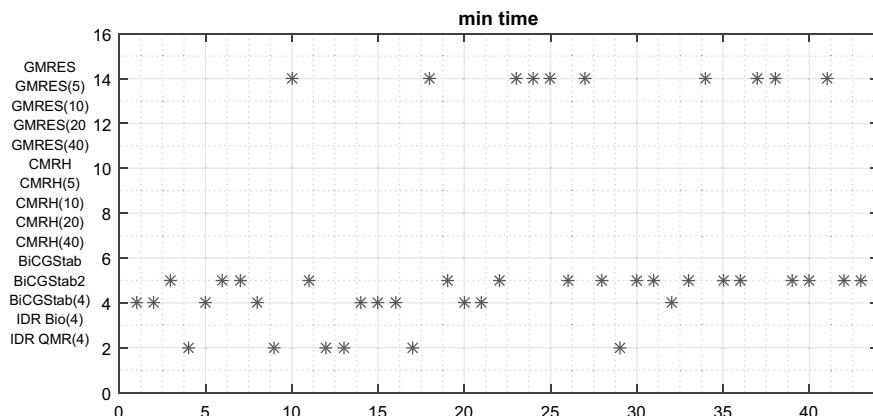


Fig. 13.4 Small matrices, minimum computing time, $\epsilon = 10^{-10}$

Things are only slightly different for $\epsilon = 10^{-10}$. The minimum number of matrix–vector products is given by CMRH for 23 problems. Methods which give a small computing time for many problems are BiCGStab2 and IDR Bio(4).

13.3 Larger matrices

Let us display the results of numerical experiments done with larger matrices for which the linear systems cannot be solved in a reasonable number of iterations without preconditioning. We use the incomplete LU factorization without fill-in ILU(0) provided by Matlab. The initial vector is $x_0 = 0$ (Tables 13.11 and 13.12).

Table 13.11 supg 001, $n = 22500$, $\epsilon = 10^{-6}$, nitmax = 1000, ILU(0)

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	40	40	7.18992e-07	2.73703e-07	0.1280
GMRES(10)	102	102	9.40293e-07	1.95907e-06	0.1766
GMRES(20)	100	100	4.30013e-07	1.61630e-06	0.1877
GMRES(40)	40	40	7.18992e-07	2.73703e-07	0.0992
CMRH	38	38	1.81738e-05	1.56589e-05	0.1992
CMRH(10)	106	106	5.02393e-05	2.02372e-04	0.2758
CMRH(20)	98	98	1.72638e-05	5.33824e-05	0.3366
CMRH(40)	38	38	1.81738e-05	1.56589e-05	0.1625
BiCGStab	35	70	6.15586e-07	1.45720e-06	0.0720
BiCGStab2	17	66	2.89230e-07	1.20480e-06	0.0656
BiCGStab(2)	17	69	3.04222e-08	1.08189e-07	0.0949
BiCGStab(4)	9	73	8.08679e-11	2.79793e-10	0.0962
IDR Bio(2)	19	58	5.21523e-07	1.06464e-06	0.0803
IDR Bio(4)	10	51	8.96298e-07	7.37999e-06	0.0729
IDR QMR(2)	20	61	8.80194e-10	8.54015e-10	0.0933
IDR QMR(4)	10	51	1.89979e-08	1.18738e-07	0.0925
IDR porth QMR(2)	58	61	1.27046e-08	2.90984e-08	0.2935
IDR porth QMR(4)	49	54	1.90035e-08	1.21756e-07	0.3975

Table 13.12 `supg 001`, $n = 22500$, $\epsilon = 10^{-10}$, nitmax = 1000, ILU(0)

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	47	47	2.34214e-11	7.20928e-12	0.1521
GMRES(10)	135	135	9.26223e-11	2.33951e-10	0.2306
GMRES(20)	162	162	9.76652e-11	9.27351e-10	0.3083
GMRES(40)	69	69	9.07831e-11	1.46449e-10	0.1625
CMRH	46	46	2.01310e-10	9.91714e-11	0.2103
CMRH(10)	151	151	1.70051e-09	2.24470e-09	0.3864
CMRH(20)	186	186	4.40290e-09	3.76471e-08	0.5547
CMRH(40)	57	57	2.12845e-09	1.38354e-09	0.2199
BiCGStab	41	82	5.65356e-11	1.77599e-10	0.0868
BiCGStab2	18	73	7.99959e-11	3.25634e-10	0.0831
BiCGStab(2)	18	73	8.42681e-11	2.99838e-10	0.0982
BiCGStab(4)	9	73	8.08679e-11	2.79793e-10	0.0964
IDR Bio(2)	21	64	5.93115e-11	5.16543e-11	0.0806
IDR Bio(4)	11	56	6.13439e-11	3.88857e-10	0.0752
IDR QMR(2)	22	67	3.62765e-13	6.01745e-13	0.1056
IDR QMR(4)	11	56	1.81107e-11	2.85534e-11	0.1039
IDR porth QMR(2)	65	68	3.75389e-13	7.39436e-13	0.3120
IDR porth QMR(4)	57	62	1.00248e-12	5.03775e-12	0.4045

Figures 13.5, 13.6 and 13.7 display the relative true residual norms for `supg 001` with $n = 22500$ as functions of the number of matrix–vector products and the computing time.

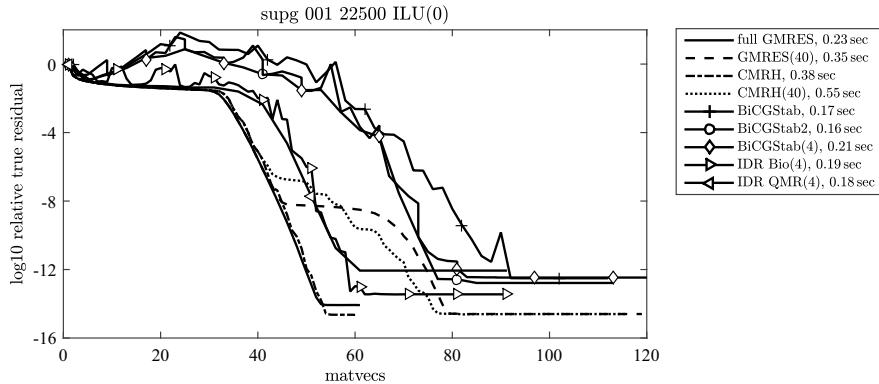


Fig. 13.5 supg 001 22500, relative residual norms vs matvecs, ILU(0) preconditioning

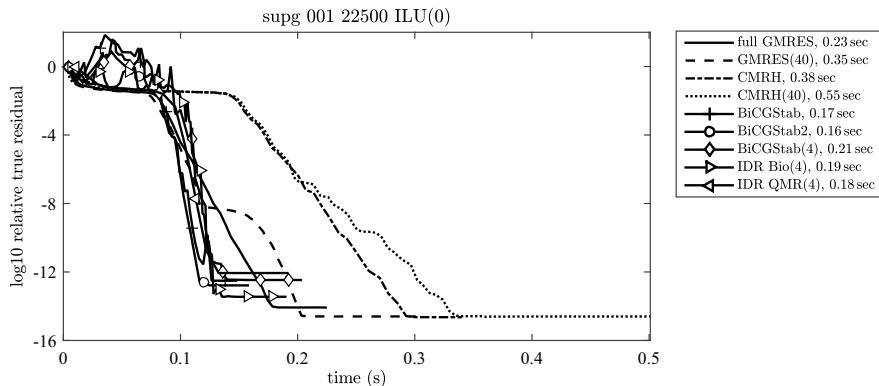


Fig. 13.6 supg 001 22500, relative residual norms vs time, ILU(0) preconditioning

Then, we consider the matrix `rajat27b` of order 20460; see Tables 13.13, 13.14 and Figures 13.8, 13.9, 13.10.

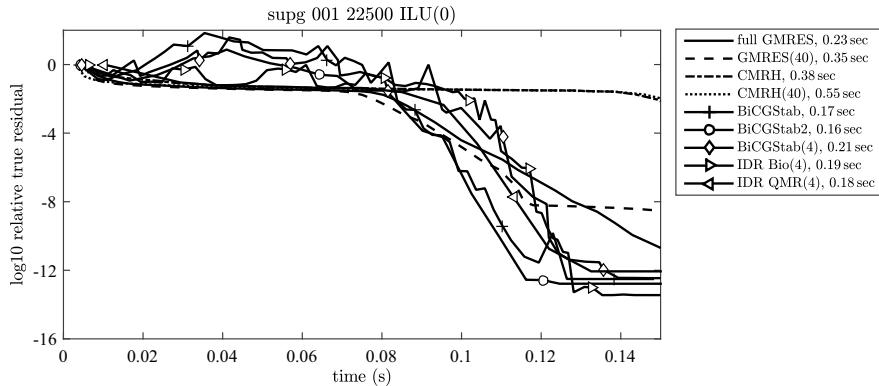


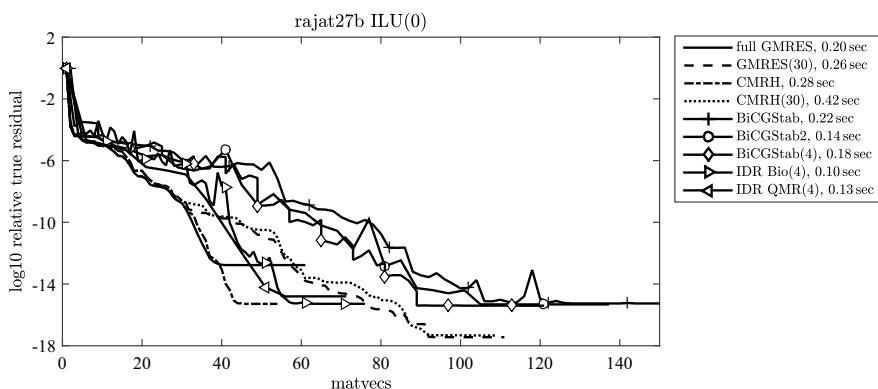
Fig. 13.7 supg 001 22500, relative true residual norms vs time, zoom, ILU(0) preconditioning

Table 13.13 rajat27b, $n = 20640$, $\epsilon = 10^{-6}$, nitmax = 1000, ILU(0)

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	16	16	4.94019e-07	3.71533e-03	0.0436
GMRES(10)	25	25	6.84689e-07	5.81952e-03	0.0428
GMRES(20)	16	16	4.94019e-07	3.71533e-03	0.0315
GMRES(30)	16	16	4.94019e-07	3.71533e-03	0.0318
CMRH	11	11	7.08533e-06	4.67751e-02	0.0676
CMRH(10)	14	14	7.48388e-06	4.28760e-02	0.0395
CMRH(20)	11	11	7.08533e-06	4.67751e-02	0.0348
CMRH(30)	11	11	7.08533e-06	4.67751e-02	0.0413
BiCGStab	16	32	5.98998e-07	4.97780e-03	0.0347
BiCGStab2	7	29	3.60620e-07	4.31319e-03	0.0310
BiCGStab(2)	7	29	2.97500e-07	4.40707e-03	0.0379
BiCGStab(4)	4	33	6.32076e-07	6.28928e-03	0.0412
IDR Bio(2)	8	25	7.75555e-07	2.02771e-02	0.0315
IDR Bio(4)	5	26	4.83347e-07	3.25537e-03	0.0404
IDR QMR(2)	8	25	3.05542e-07	5.21226e-03	0.0374
IDR QMR(4)	4	21	3.80499e-07	3.28110e-03	0.0431
IDR porth QMR(2)	22	25	4.77594e-07	6.62950e-03	0.1263
IDR porth QMR(4)	18	23	6.02154e-07	4.71217e-03	0.2004

Table 13.14 rajat27b, $n = 20640$, $\epsilon = 10^{-10}$, nitmax = 1000, ILU(0)

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	32	32	5.78015e-11	1.13075e-06	0.0775
GMRES(10)	150	150	2.45599e-10	1.76129e-05	0.2207
GMRES(20)	67	67	7.42601e-11	4.43607e-06	0.1203
GMRES(30)	42	42	9.12829e-11	4.12496e-06	0.0804
CMRH	32	32	2.65919e-10	1.98284e-06	0.1255
CMRH(10)	150	150	2.88509e-09	1.25032e-04	0.3625
CMRH(20)	58	58	3.08602e-10	7.26605e-06	0.1622
CMRH(30)	36	36	3.16828e-10	8.65119e-06	0.1182
BiCGStab	36	72	8.66307e-11	2.77366e-06	0.0685
BiCGStab2	16	65	6.27027e-11	1.28058e-06	0.0676
BiCGStab(2)	15	61	6.46450e-11	2.39335e-06	0.0687
BiCGStab(4)	8	65	6.86041e-12	9.11217e-08	0.0771
IDR Bio(2)	17	52	2.45388e-11	4.37591e-07	0.0718
IDR Bio(4)	9	46	1.11477e-12	2.31940e-08	0.0622
IDR QMR(2)	17	52	4.75486e-13	6.51741e-09	0.0791
IDR QMR(4)	9	46	4.46445e-13	1.21391e-08	0.0783
IDR porth QMR(2)	47	50	3.20331e-11	7.20371e-07	0.1505
IDR porth QMR(4)	39	44	4.19104e-11	4.43887e-07	0.2719

**Fig. 13.8** rajat27b, relative true residual norms vs matvecs, ILU(0) preconditioning

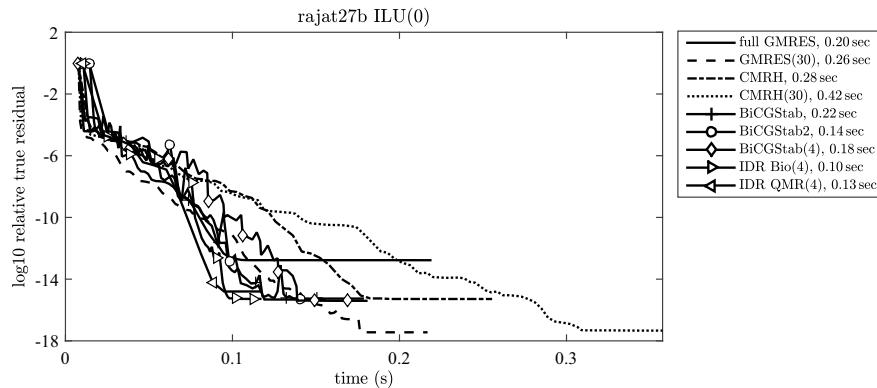


Fig. 13.9 rajat27b, relative true residual norms vs time, ILU(0) preconditioning

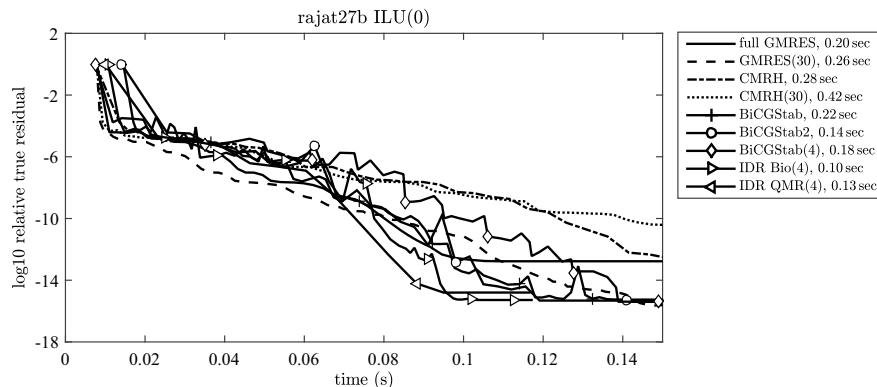


Fig. 13.10 rajat27b, relative true residual norms vs time, zoom, ILU(0) preconditioning

Tables 13.15, 13.16 and Figures 13.11, 13.12, 13.13 display the results for the matrix `matrix-new 3` of order 125329. We observe that the minimum numbers of matrix–vector products are obtained with CMRH but the relative residual norms are larger than the chosen threshold. This is explained by the fact that we use the computed residual norm for the stopping criterion.

Table 13.15 `matrix-new 3`, $n = 125329$, $\epsilon = 10^{-6}$, nitmax = 1000, ILU(0)

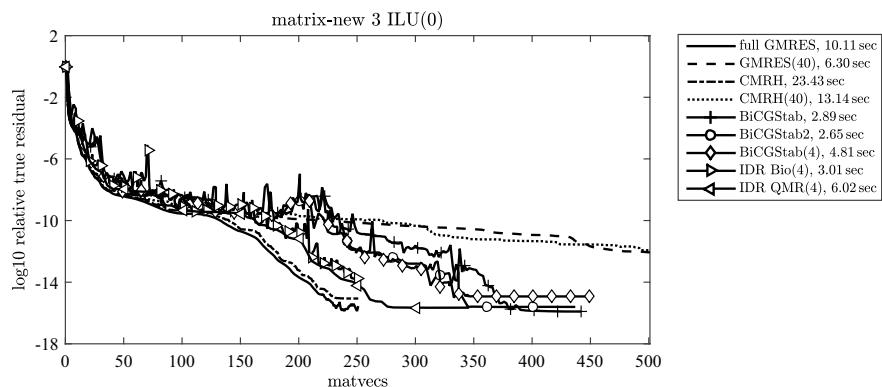
Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	17	17	7.82242e-07	7.58961e-02	0.1693
GMRES(10)	18	18	7.83493e-07	7.59121e-02	0.1652
GMRES(20)	17	17	7.82242e-07	7.58961e-02	0.1537
GMRES(40)	17	17	7.82242e-07	7.58961e-02	0.1661
CMRH	13	13	2.22291e-06	1.42691e-01	0.2433
CMRH(10)	15	15	1.56294e-06	1.33246e-01	0.2289
CMRH(20)	13	13	2.22291e-06	1.42691e-01	0.2167
CMRH(40)	13	13	2.22291e-06	1.42691e-01	0.2379
BiCGStab	15	30	5.66029e-07	2.81026e-02	0.1750
BiCGStab2	8	33	8.16645e-08	1.66950e-02	0.1955
BiCGStab(2)	8	33	9.29136e-08	1.85660e-02	0.2536
BiCGStab(4)	4	33	8.22993e-08	1.69301e-02	0.2791
IDR Bio(2)	10	31	2.80070e-07	8.61827e-03	0.2384
IDR Bio(4)	4	21	7.55821e-07	5.08420e-02	0.2251
IDR QMR(2)	10	31	1.58913e-07	1.05043e-02	0.3063
IDR QMR(4)	5	26	2.27597e-07	2.22012e-02	0.3863
IDR porth QMR(2)	20	23	6.42869e-07	5.77678e-02	0.6287
IDR porth QMR(4)	19	24	9.83545e-07	6.42877e-02	0.8941

We cannot compare all the methods for all the problems we have considered. For the results below we have selected a few methods and some “large” sparse matrices including two that we have considered before. We use ILU(0) as a preconditioner and $x_0 = 0$.

The matrices of the problems listed in the x -axis of Figures 13.14, 13.15, 13.16 and 13.17 are (from 1 to 7) `cage11`, `epb2`, `wathen100`, `rajat27b`, `raefsky3`, `multdcop 01b`, `matrix-new 3`. The y -axis corresponds to methods, GMRES (resp. IDR QMR(4)) corresponding to 15 (resp. 1).

Table 13.16 matrix-new 3, $n = 125329$, $\epsilon = 10^{-10}$, nitmax = 1000, ILU(0)

Method	nit	mv	rel. res. norm	rel. err. norm	time (s)
GMRES	132	132	9.73084e-11	7.53658e-04	2.7342
GMRES(10)	460	460	9.94819e-11	2.09691e-04	3.5265
GMRES(20)	334	334	9.95215e-11	2.13086e-04	3.2403
GMRES(40)	224	224	9.95707e-11	2.07936e-04	2.6414
CMRH	63	63	2.30663e-09	8.88374e-04	2.2754
CMRH(10)	102	102	2.50684e-09	9.18369e-04	1.5872
CMRH(20)	75	75	2.66263e-09	8.61228e-04	1.3537
CMRH(40)	74	74	2.08727e-09	8.28171e-04	1.7314
BiCGStab	117	234	3.34173e-11	3.40914e-05	1.3519
BiCGStab2	53	213	3.08936e-11	3.76911e-06	1.2402
BiCGStab(2)	55	221	5.87197e-11	8.19086e-05	1.8312
BiCGStab(4)	28	225	5.37257e-11	9.83310e-05	2.0540
IDR Bio(2)	58	175	9.20654e-11	1.53741e-05	1.3447
IDR Bio(4)	33	166	5.91386e-11	2.72476e-03	1.7124
IDR QMR(2)	66	199	2.70817e-11	2.07862e-04	2.0536
IDR QMR(4)	36	181	2.57224e-11	1.78089e-04	2.8598
IDR porth QMR(2)	188	191	6.83412e-11	3.06814e-05	6.3620
IDR porth QMR(4)	160	165	7.24521e-11	2.60206e-04	8.8787

**Fig. 13.11** matrix-new 3, relative true residual norms vs matvecs, ILU(0) preconditioning

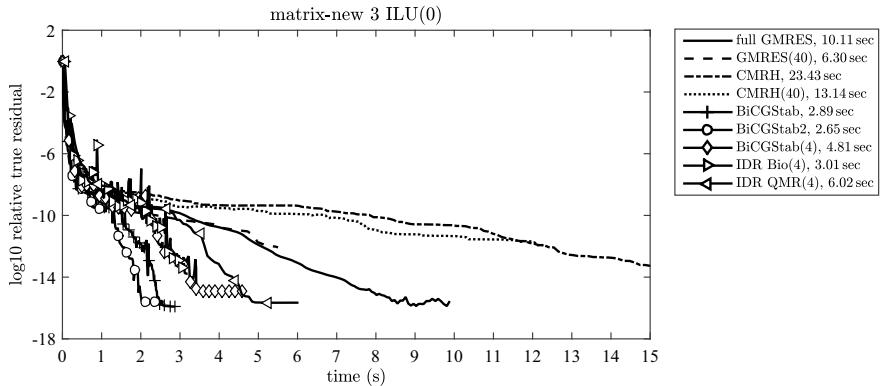


Fig. 13.12 matrix-new 3, relative true residual norms vs time, ILU(0) preconditioning

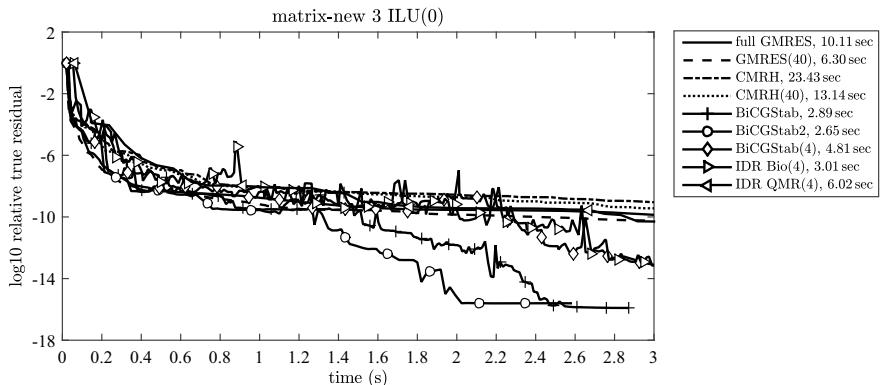


Fig. 13.13 matrix-new 3, relative true residual norms vs time, zoom, ILU(0) preconditioning

For $\epsilon = 10^{-6}$ CMRH gives the minimum number of matrix–vector products for 4 problems but does not give the smallest computing time. Moreover, the methods giving the minima do not always give the minimum relative true residual norms at convergence. As we remarked above we stop on the norm of the computed residual and not the true residual. Table 13.17 shows for each problem the relative true residual norms of the methods giving the smallest number of matrix–vector products (first line) and the smallest computing time (second line) as well as the smallest true residual norm that can be obtained with one of the methods we considered. For some methods the numbers of iterations were small; this explains that restarted methods may give the same results as full methods. We observe that by using the methods minimizing the number of matrix–vector products or the computing time we do not get the smallest residual norms.

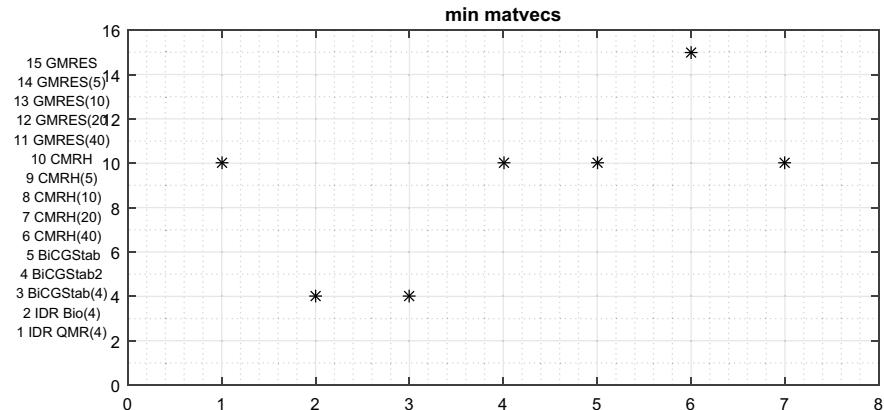


Fig. 13.14 Large matrices, minimum number of matvecs, $\epsilon = 10^{-6}$

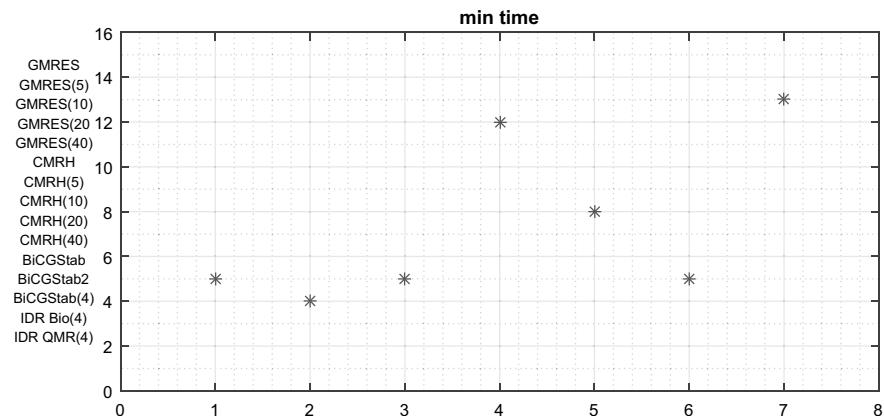


Fig. 13.15 Large matrices, minimum computing time, $\epsilon = 10^{-6}$

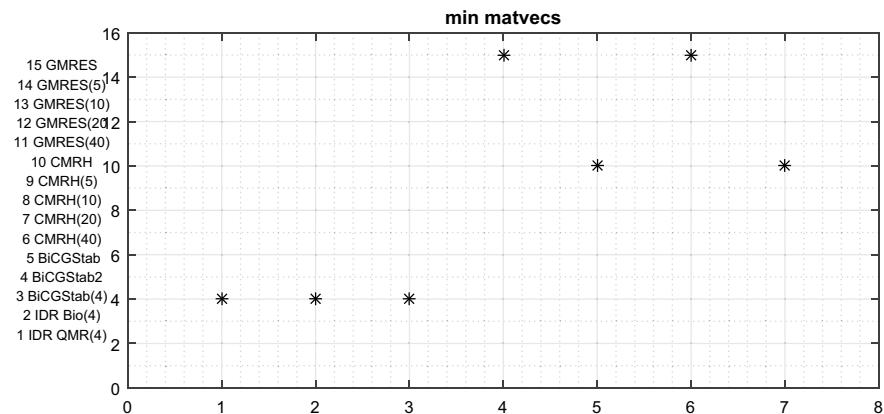
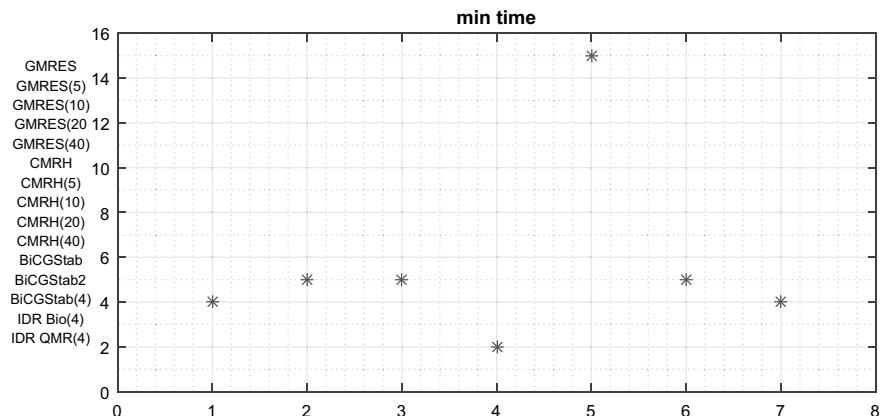


Fig. 13.16 Large matrices, minimum number of matvecs, $\epsilon = 10^{-10}$

Table 13.17 Relative residual norms, ILU(0), $\epsilon = 10^{-6}$

Matrix	best meth.	rel. res. norm	min. rel. res
cage11	CMRH	2.27679e-05	7.25669e-13
	BiCGStab	9.59707e-07	
epb2	BiCGStab2	2.32042e-07	4.52088e-08
	BiCGStab	6.71027e-07	
wathen100	BiCGStab2	3.35566e-07	1.20964e-08
	BiCGStab	5.46165e-07	
rajat27b	CMRH	7.08533e-06	3.60620e-07
	CMRH(20)	7.08533e-06	
raefsky3	CMRH	4.62901e-06	1.87436e-07
	CMRH(5)	4.62901e-06	
mult dcop 01b	GMRES	6.09189e-09	8.50146e-16
	GMRES(10)	6.09189e-09	
matrix-new 3	CMRH	2.22291e-06	8.16645e-08
	GMRES(40)	7.83493e-07	

**Fig. 13.17** Large matrices, minimum computing time, $\epsilon = 10^{-10}$

For $\epsilon = 10^{-10}$, except for one problem, the smallest computing times are given by methods using short recurrences. Table 13.18 shows the relative residual norms. Again the methods giving the smallest number of matrix–vector products or the minimum computing time do not always obtain the smallest relative true residual norms.

All the previous numerical results show that we have to be careful when selecting a method for solving a given problem. Choosing the fastest method (or the method giving the smallest number of matrix–vector products) does not always provide the

Table 13.18 Relative residual norms, ILU(0), $\epsilon = 10^{-10}$

Matrix	best meth.	rel. res. norm	min. rel. res
cage11	BiCGStab2	8.83640e-13	7.14153e-15
	BiCGStab	1.63087e-12	
epb2	BiCGStab2	1.56885e-11	1.56885e-11
	BiCGStab2	1.56885e-11	
wathen100	BiCGStab2	1.53811e-12	5.16071e-13
	BiCGStab	2.56959e-11	
rajat27b	GMRES	5.78015e-11	4.46445e-13
	BiCGStab	8.66307e-11	
raefsky3	CMRH	7.40734e-10	9.17256e-12
	GMRES	9.04751e-11	
mult_dcop_01b	GMRES	2.24578e-13	8.50146e-16
	BiCGStab2	1.04985e-15	
matrix-new_3	CMRH	2.30663e-09	2.57224e-11
	BiCGStab2	3.08936e-11	

best residual norms and also probably not the best error norms. Therefore, the choice depends on what the user is looking for, the fastest method, the best residual norm or the best error norm (which, unfortunately, is usually not known). Moreover, in many cases, solving a linear system is only a part of obtaining the solution of the given problem and this issue has also to be considered as well as the properties of the computer with which the computation has to be done.

Appendix A

Test matrices and short biographical notices

A.1 Test matrices

Small test matrices are listed in Table A.1. The matrices are ordered by increasing condition number; nnz is the number of nonzero entries, $\kappa(A)$ is the condition number (computed by Matlab), $\min svd$ is the smallest singular value of A and fov tells that 0 is inside the field of values when it is equal to 1. The last column tells if the matrix is normal. In fact, all the normal matrices in the list are symmetric and denoted by an “s”. All these matrices can be obtained from the SuiteSparse matrix collection <https://sparse.tamu.edu>.

Table A.2 lists some larger matrices. The condition number is given by the Matlab function `condest`. Some of the matrices come from the same collection as before.

The matrices whose name starts with `supg` are obtained by a finite element discretization of convection–diffusion equations. This problem was considered in [352] and [638]. The partial differential equation is

$$-\nu \Delta u + w \cdot \nabla u = 0 \text{ in } \Omega = (0, 1) \times (0, 1), \quad u = g \text{ on } \partial\Omega,$$

with a constant wind $w = (0, 1)^T$ parallel to one of the axis. As in [638] we consider an SUPG (streamline upwind Galerkin) bilinear finite element discretization on a regular grid with a mesh size $h = 1/(m + 1)$. This yields a linear system with a matrix

$$A = \nu K \otimes M + M \otimes ((\nu + \delta h)K + C),$$

where ν is the diffusion coefficient and δ is the stabilization parameter. The matrices K , M and C are tridiagonal

Table A.1 Properties of small test matrices

Matrix	order	nnz	$\kappa(A)$	min svd	fov	normal?
pde225	225	1065	3.90638e+01	2.50580e-01	0	—
gre 343	343	1310	1.11976e+02	9.01565e-03	1	—
jphw 991	991	6027	1.42045e+02	1.14696e-01	0	—
pde2961	2961	14585	6.42493e+02	1.61532e-02	1	—
jagmesh1	936	6264	1.22765e+03	5.62507e-03	1	s
bfsa782	782	7514	1.74061e+03	7.22416e-03	1	—
dw2048	2048	10114	2.09321e+03	4.67610e-04	1	—
jagmesh2	1009	6865	2.36649e+03	2.94509e-03	1	s
lshp1009	1009	6865	2.36649e+03	2.94509e-03	1	s
raefsky2	3242	293551	4.25194e+03	8.76027e-04	1	—
fs 680 1c	680	2184	8.69443e+03	4.38996e-04	1	—
add20	2395	13151	1.20471e+04	5.99434e-05	0	—
raefsky1	3242	293409	1.28851e+04	2.87892e-04	1	—
jagmesh4	1440	9504	1.51630e+04	4.52739e-04	1	s
fs 680 1	680	2184	1.54052e+04	4.66822e+09	1	—
sherman1	1000	3750	1.55953e+04	3.23487e-04	0	—
nos3	960	15844	3.77236e+04	1.82884e-02	0	s
sherman5	3312	20793	1.87941e+05	2.41965e-02	1	—
cavity05	1182	32632	5.77065e+05	2.25403e-05	0	—
e05r0500	236	5846	1.15887e+06	4.93622e-05	1	—
comsol	1500	97645	1.41522e+06	4.47174e-07	0	—
olm1000	1000	3996	1.48722e+06	6.19384e-02	1	—
cavity10	2597	76171	2.95507e+06	4.41707e-06	0	—
steam2	600	5660	3.78313e+06	1.23855e+03	0	—
1138bus	1138	4054	8.57265e+06	3.51686e-03	0	s
steam1	240	2248	2.82750e+07	7.67886e-01	0	—
bcsstk26	1922	30336	1.65934e+08	9.53833e+02	0	s
nos7	729	4617	2.37451e+09	4.15413e-03	0	s
watt1	1856	11360	4.35964e+09	2.29377e-10	1	—
bcsstk14	1806	63454	1.19232e+10	9.99998e-01	0	—
fs 183 6	183	1000	1.73678e+11	6.79901e-03	1	—
bcsstk20	485	3135	3.89279e+12	3.24066e+03	0	s
mcf6	765	24382	4.24182e+13	3.17699e+03	1	—
nnc	261	1500	2.90962e+14	3.57651e-12	1	—
lnsp	511	2796	3.32894e+15	1.26332e-05	1	—

Table A.2 Properties of larger test matrices

Matrix	order	nnz	$\kappa(A)$	fov	normal?
add32	4960	19848	1.36677e+02	0	–
supg 0001	6400	56644	1.66526e+02	0	–
supg 00001	6400	56644	1.90256e+02	0	–
ex37	3565	67591	2.26374e+02	0	–
supg 001	6400	56644	3.22630e+02	0	–
supg 000001	6400	56644	3.36032e+02	0	–
supg 01	6400	56644	1.99844e+03	0	–
supg 1	6400	56644	3.33267e+03	0	–
supg 100	6400	56644	6.51623e+03	0	–
wang4	26068	177196	4.91249e+04	1	–
memplus	17758	99147	2.66615e+05	1	–
convdiff xu 5000	8100	40140	1.17581e+07	1	–
convdiff xu 1000	8100	40140	2.69404e+12	1	–
sherman3	5005	20033	6.89814e+16	1	–
convdiff xu 500	8100	40140	8.57866e+16	1	–

$$\begin{aligned} K &= \frac{1}{h} \text{tridiag}(-1, 2, -1), \\ M &= \frac{h}{6} \text{tridiag}(1, 4, 1), \\ C &= \frac{1}{2} \text{tridiag}(-1, 0, 1). \end{aligned}$$

The stabilization parameter δ is chosen in relation to the mesh size h . For convection-dominated problems a quasi-optimal value is

$$\delta_* = \frac{1}{2} \left(1 - \frac{1}{P_h} \right),$$

where P_h is the mesh Peclet number $P_h = h\|w\|/(2\nu)$. The right-hand side b is determined by the Dirichlet boundary conditions.

In the name of the matrix in Table A.2 the number refers to the value of ν , for instance, 00001 means $\nu = 0.0001$. We use this problem with different numbers of discretization points.

The matrices whose name starts with `convdiff` arise from the finite difference discretization of the problem

$$-\Delta u + \gamma(xu_x + yu_y) + \beta u = f, \text{ in } \Omega = (0, 1) \times (0, 1), \quad u = 0 \text{ on } \partial\Omega.$$

We use upwind differencing for the first-order terms on a regular Cartesian mesh. The number in the name of the matrix is the value of γ and $\beta = -10000$.

The problem `diff conv 400` is the upwind finite difference discretization of the partial differential equation,

$$-\Delta u + 2p^x u_x = 0 \text{ in } \Omega = (0, 1) \times (0, 1), \quad u = 0 \text{ on } \partial\Omega,$$

with $p^x = e^{2(x^2+y^2)}$, using 20 interior discretization points in each direction, that is, A is of order 400. The right-hand side b is such that all the components of the solution are equal to 1.

The matrix `rajat27b` is a modification of the matrix `rajat27` from the SuiteSparse matrix collection [257] arising from a circuit simulation problem. There are zeros on the diagonal of `rajat27`. Since we would like to use preconditioners that cannot accept zeros on the diagonal we added $2I$ to the matrix. The matrix `rajat27b` is of order 20640 with 101681 nonzero entries. Its estimated condition number is $4.8588 \cdot 10^7$ and its Frobenius norm is $1.2999 \cdot 10^6$

The matrix `matrix-new 3` arising from a semiconductor device problem from the same collection as above is of order 125329 with 893984 nonzero entries. Its entries are in the interval $[-1, 1]$ and its Frobenius norm is 225.22.

We also use some other matrices from that collection. The matrix `cage 11`, arising from a directed weighted graph, is of order 39082 with a small condition number of 12.27. The matrix `epb2`, from a thermal problem, is of order 25228 and its condition number is $2.62 \cdot 10^3$. The matrix `wathen100`, from a random 2D/3D problem, is a symmetric matrix of order 30401 and its condition number is $5.8 \cdot 10^3$. The matrix `raefsky3`, from computational fluid dynamics, is of order 21200 and its condition number is $1.98 \cdot 10^{11}$. The matrix `mult_dcop 01b` is a modification of the matrix `mult_dcop 01` from a circuit simulation problem. We add $2I$ to the diagonal to avoid zero entries on the diagonal. Its order is 25187.

A.2 Short biographical notices

- ◊ Walter Edwin Arnoldi (1917–1995) was an American engineer. He was born in New York City and had resided until his ultimate death in West Hartford (Connecticut, USA) since 1950. He graduated in mechanical engineering at the Stevens Institute of Technology (New Jersey) and worked for United Aircraft Corporation from 1939 to 1977. According to the Hartford Courant newspaper obituary notice, his career included many years as a technical specialist and engineering administrator, first in mechanical vibrations and later in oxygen reclamation problems of space science.

◊ Léon César Autonne (1859–1916) was a French mathematician born in Odessa (Ukraine). He studied in Ecole Polytechnique in Paris. He obtained his thesis in 1882 and worked in differential equations, algebraic geometry, group theory and linear algebra. He was Professor in Lyon University (France).

◊ Eugenio Beltrami (1835–1900) was an Italian mathematician born in Cremona (Austrian Empire, now Italy). He was teaching in universities of Bologna, Pisa, Rome and Pavia. He worked in differential geometry and mathematical physics. He died in Rome.

◊ Jacques Philippe Marie Binet (1786–1856) was a French mathematician and astronomer born in Rennes (France). He was a student at Ecole Polytechnique in which he was teaching later on. He discovered the rule for multiplying matrices in 1812. He is also remembered for the “Binet’s form” of the Fibonacci numbers. He was elected to the Académie des Sciences in 1843. He died in Paris.

◊ Augustin-Louis Cauchy (1789–1857) was a French mathematician born in Paris (France). He was a student in Ecole Polytechnique. He was one of the most prominent mathematicians of the 19th century working in analysis, algebra, mechanics and probabilities. He died in Sceaux (France).

◊ Arthur Cayley (1821–1895) was a British mathematician and lawyer born in Richmond (UK). He studied in Trinity College, Cambridge and graduated in 1842. He was appointed Professor in Cambridge in 1863. In 1858 he published a “Memoir on the theory of matrices” which contains the first abstract definition of a matrix. He proved what is now called the Cayley–Hamilton theorem for 2×2 and 3×3 matrices. He died in Cambridge (UK).

◊ Philip J. Davis (1923–2018) was an American mathematician born in Lawrence (USA). He obtained his Ph.D. from Harvard University (USA) in 1950. His advisor was Ralph P. Boas. He is the author of the well-known book “Methods of numerical integration” (with Philip Rabinowitz). He was also the author of “The Mathematical Experience” (with Reuben Hersh). He was Professor at Brown University (USA).

◊ Dmitry Konstantinovich Faddeev (1907–1989) was a Soviet mathematician born in Yukhnov (Russia). He graduated from Leningrad State University in 1928. With his wife, Vera N. Faddeeva, he wrote the book “Numerical methods in linear algebra” in 1960. He died in Leningrad (now Saint Petersburg).

◊ Roger Fletcher (1939–2016) was a British mathematician born in Huddersfield (UK). He was a pioneer in optimization. He was the “F” in the BFGS algorithm for unconstrained nonlinear optimization. He was Professor in Dundee University. He died while hiking in the Scottish Highlands.

◊ Ferdinand Georg Frobenius (1849–1917) was a German mathematician born in Charlottenburg (Prussia, now Germany). He obtained his thesis in 1870 supervised by Karl Weierstrass. He was professor in Berlin for a short while and then in Zürich (Switzerland) from 1875 to 1892. He was appointed to the university in Berlin in

1892. He was working, among other topics, on group theory. He proved the general case of the Cayley–Hamilton theorem in 1878. He died in Berlin.

◊ Felix Ruvimovich Gantmacher (1908–1964) was a Soviet mathematician born in Odessa (Russia, now Ukraine). He was Professor at Moscow Institute of Physics and Technology. He is well known for his book “The theory of matrices” [396] and his other book “Oscillation matrices and kernels and small vibrations of mechanical systems” co-authored with Mark Krein.

◊ James Wallace Givens (1910–1993) was an American mathematician born in Alberene (USA). He got his Ph.D. from Princeton University in 1936. He was a Professor at the University of Tennessee, at Wayne State University and Northwestern University. He also worked at the Argonne National Laboratory where he was Director of the Division of Applied Mathematics from 1964 to 1970.

◊ Gene Howard Golub (1932–2007) was an American mathematician born in Chicago (USA). He obtained his Ph.D. from the University of Illinois at Urbana-Champaign (USA) in 1959. His advisor was Abraham Taub. He was Professor at Stanford University (USA). He is well known for his many contributions to numerical linear algebra and his famous book “Matrix Computations” [438] co-authored with Charles F. Van Loan. He died in Stanford.

◊ William B. Gragg (1936–2016) was an American mathematician born in Bakersfield (USA). He obtained his Ph.D. from the University of California at Los Angeles (USA) in 1964. His advisor was Peter Henrici. He was Professor at the Naval Postgraduate School in Monterey (USA). He died in Monterey.

◊ Jørgen Pedersen Gram (1850–1916) was a Danish actuary and mathematician who was born in Nustrup and died in Copenhagen (Denmark). The paper about what is now known as the Gram–Schmidt algorithm was published in 1883.

◊ Karl Adolf Hessenberg (1904–1959) was born in Frankfurt (Germany). He studied electrical engineering in Darmstadt from 1925 to 1936. He did his dissertation under the supervision of Professor Walther in Darmstadt up to 1942. Starting in 1936 he worked as an engineer for the company A.E.G. until his death; see <http://www.hessenberg.de/karl.html>.

◊ Alston Scott Householder (1904–1993) was an American mathematician born in Rockford, Illinois (USA). He obtained his Ph.D. at the University of Chicago in 1937 under the supervision of Gilbert A. Bliss. He joined the Oak Ridge National Laboratory in 1946 until 1969 and then became Professor at the University of Tennessee. He is the author of the classic book “The theory of matrices in numerical analysis” [535]. He is also known for founding the Gatlinburg Symposium on Numerical Linear Algebra in 1961. This is now known as the Householder Symposium. He died in Malibu, California (USA).

◊ Carl Gustav Jacob Jacobi (1804–1851) was born in Postdam (Prussia, now Germany). He entered the university in Berlin in 1821 where afterward he was teaching in 1825–1826. He moved to Königsberg in 1826. Jacobi had contacts with

Adrien-Marie Legendre about the theory of elliptic functions which was also studied by Niels H. Abel at the same time. He is the author of many significant contributions to mathematics. He died in Berlin.

◊ Camille Jordan (1838–1922) was a French mathematician born in Lyon. He entered Ecole Polytechnique in 1855 where he became Professor in 1876. He obtained his thesis in 1860. He was mainly interested in group theory. One of his many contributions is the Jordan normal form. He died in Paris.

◊ Alexei Nikolaevich Krylov (1863–1945) was a Russian mathematician, physicist and naval engineer. He was born in Visyaga (Russia). He attended the Naval School in Saint Petersburg from 1878 until 1884. He then worked in the Main Hydrographic Office on the problem of compass deviation and he published his first technical paper “On the arrangement of needles on the compass card” in the Maritime Digest in 1886. He graduated in 1890 from the Saint Petersburg Naval Academy and then stayed in the navy for the rest of his career. One of his teachers at the Academy was Aleksandr Nikolaevich Korkin, a student of Pafnuty L. Chebyshev. In 1899 Krylov became the first foreigner to receive a gold medal from the Royal Institution of Naval Architects in London. He was awarded a honorary doctorate in applied mathematics from Moscow University in 1914 and was elected that year a member of the Russian Academy of Sciences and a full member in 1916. He became the head of the Naval Academy in 1918. From 1927 until 1932 he was Director of the Physics-Mathematics Institute of the USSR Academy of Sciences. Krylov wrote around 300 papers and several books. He died in Leningrad, now Saint Petersburg. For a short biography, see [983] and [640] page 64. Also of interest are <https://www.staff.science.uu.nl/~vorst102/kryl.html> and <https://krylov-centre.ru/en/about/history/>. Of interest are also Krylov’s memoirs [609] and its translation into English [691].

◊ Pierre Simon de Laplace (1749–1827) was a French mathematician born in Beaumont en Auge (France). He was one of the most prominent mathematicians of his time, well known for his works in celestial mechanics and probability. He was also much involved in politics, being a minister of Napoléon for a short while. The Laplacian operator is named after him. He died in Paris.

◊ Cornelius Lanczos (1893–1974) was a Hungarian physicist and mathematician born in Székesfehérvár (Hungary). For a short biography of Lanczos see, for instance, [676], Appendix pages 333–334 and [410]. The book [411] describes Lanczos’ life and work. He died in Hungary during a visit to Budapest.

◊ Gury Ivanovich Marchuk (1925–2013) was a Soviet-Russian mathematician and physicist born in Orenburg Oblast (USSR). He became academician in 1968. He was Director of the computing center of Novosibirsk and President of the USSR Academy of Sciences in 1986–1991. He obtained many honors and awards in the USSR and abroad and authored many books and papers. He died in Moscow (Russia).

◊ Winifred J. Morrison was an American statistician.

◊ Henri Eugène Padé (1863–1963) was a French mathematician born in Abbeville (France). He obtained his thesis in 1892 under the supervision of Charles Hermite (1822–1901). He studied what is now known as the Padé approximants (even though they were known earlier to Lambert, Lagrange and Jacobi). He introduced the Padé table. He died in Aix en Provence (France).

◊ Charles Emile Picard (1856–1941) was a French mathematician born in Paris (France). He studied at the Ecole Normale Supérieure in Paris. His dissertation in 1877 was supervised by Gaston Darboux. He was Professor in Paris Sorbonne University. He died in Paris.

◊ Vlastimil Pták (1925–1999) was a Czech mathematician born in Prague. He was for a long time the Head of the Department of Functional Analysis in the Mathematical Institute of the Academy of Sciences and Professor at Charles University in Prague. He published more than 160 research papers.

◊ Axel Ruhe (1942–2015) was a Swedish mathematician born in Jönköping (Sweden). He obtained his Ph.D. thesis from Lund University (Sweden) in 1970. His advisor was Carl-Erik Fröberg. He was Professor in Umeå University, Chalmers University in Göteborg and KTH in Stockholm. His work was mainly devoted to eigenvalue problems for which he made important contributions.

◊ Heinz Rutishauser (1918–1970) was a Swiss mathematician born in Weinfelden (Switzerland). He obtained his degree from ETH Zürich in 1950 where he was professor later on. He was a pioneer in computer science being involved in the design of the Algol language. He died in his office in Zürich from a heart failure.

◊ Erhard Schmidt (1876–1959) was a German mathematician born in Dorpat (Russia, now Tartu in Estonia). He obtained his thesis in Göttingen in 1905 under David Hilbert. He was an expert in functional analysis. He died in Berlin (Germany).

◊ Charles Sheffield (1935–2002) was born in Kingston upon Hull (UK). He obtained a Ph.D. in theoretical physics in Cambridge (UK). He was a physicist, a mathematician and an award-winning science fiction writer. He moved to the USA and became a consultant with NASA and Chief Scientist at the Earth Satellite Corporation in Washington. He died in Rockville, Maryland (USA).

◊ Jack Sherman was an American statistician.

◊ Thomas Jan Stieltjes (1856–1894) was a Dutch mathematician born in Zwolle (The Netherlands). He was in relation with Charles Hermite who supported him and he moved to France in 1886. He became Professor in Toulouse University. He worked on Gauss quadrature, orthogonal polynomials, continued fractions and moment problems. He died in Toulouse (France).

◊ James Joseph Sylvester (1814–1897) was a British mathematician born in London (UK). He was professor in London, at the University of Virginia and the Johns Hopkins University in Baltimore (USA). He returned to England in 1883 and became Professor in Oxford University. He coined the term “matrix” in 1850. He died in London (UK).

◊ Alexandre-Théophile Vandermonde (1735–1796) was a French mathematician, chemist and musician born in Paris (France). He only wrote four mathematical papers making contributions to the theory of determinants but, nevertheless, he was elected to the Académie des Sciences in 1771. He died in Paris.

◊ James Hardy Wilkinson (1919–1986) was born in Strood (UK). He was one of the most famous British applied mathematicians of the twentieth century. He graduated from Trinity College in Cambridge (UK) and worked at the National Physical Laboratory. He became interested in numerical computing during World War II and was Alan Turing's assistant at the National Physical Laboratory in London in 1946. He received the Turing Award in 1970. He was the first professional numerical analyst to be elected to the Fellowship of the Royal Society. He is the author of the famous book “The algebraic eigenvalue problem” [982] first published in 1965. He died in Teddington (UK). A biography by Leslie Fox was published in Biographical Memoirs of Fellows of the Royal Society, v 33 (1987), pp. 669–708; see also the dedication by Gene H. Golub and Cleve B. Moler in Linear Algebra Appl., v 88/89 (1987).

◊ Max A. Woodbury was an American statistician. He obtained a Ph.D. from the University of Michigan in 1948.

◊ David M. Young (1923–2008) was an American mathematician born in Quincy (USA). He earned his Ph.D. in 1950 from Harvard University under the supervision of Garrett Birkhoff. He is well known for his work on SOR and SSOR iterative methods. He is the author of the book “Iterative solution of large linear systems” [1013] published in 1971. For many years he was Professor at the University of Texas at Austin (USA). He died in Austin.

References

1. Abdel-Rehim, A.M., Stathopoulos, A., Orginos, K.: Extending the eigCG algorithm to nonsymmetric Lanczos for linear systems with multiple right-hand sides. *Numer. Linear Algebra Appl.* **21**(4), 473–493 (2014)
2. Abe, K.: On convergence speed of parallel variants of BiCGSTAB for solving linear equations. In: Asian Simulation Conference, pp. 401–413. Springer, Singapore (2018)
3. Abe, K., Sleijpen, G.L.G.: BiCR variants of the hybrid BiCG methods for solving linear systems with nonsymmetric matrices. *J. Comput. Appl. Math.* **234**, 985–994 (2010)
4. Abe, K., Sleijpen, G.L.G.: Hybrid Bi-CG methods with a Bi-CG formulation closer to the IDR approach. *Appl. Math. Comput.* **218**(22), 10889–10899 (2012)
5. Abe, K., Sleijpen, G.L.G.: Solving linear equations with a stabilized GPBiCG method. *Appl. Numer. Math.* **67**, 4–16 (2013)
6. Abe, K., Zhang, S.L.: A variable preconditioning using the SOR method for GCR-like methods. *Int. J. Numer. Anal. Mod.* **2**(2), 147–161 (2005)
7. Abu-Omar, A., Kittaneh, F.: Estimates for the numerical radius and the spectral radius of the Frobenius companion matrix and bounds for the zeros of polynomials. *Ann. Funct. Anal.* **5**(1), 56–62 (2014)
8. Agullo, E., Giraud, L., Jing, Y.F.: Block GMRES method with inexact breakdowns and deflated restarting. *SIAM J. Matrix Anal. Appl.* **35**(4), 1625–1651 (2014)
9. Ahuja, K., Benner, P., de Sturler, E., Feng, L.: Recycling BiCGStab with an application to parametric model order reduction. *SIAM J. Sci. Comput.* **37**(5), S429–S466 (2015)
10. Ahuja, K., de Sturler, E., Gugercin, S., Chang, E.R.: Recycling BiCG with an application to model reduction. *SIAM J. Sci. Comput.* **34**(4), 1925–1949 (2012)
11. Aihara, K., Abe, K., Ishiwata, E.: An alternative implementation of the IDRstab method saving vector updates. *JSIAM Lett.* **3**, 69–72 (2011)
12. Aihara, K., Abe, K., Ishiwata, E.: A quasi-minimal residual variant of IDRstab using the residual smoothing technique. *Appl. Math. Comput.* **236**, 67–77 (2014)
13. Aihara, K., Abe, K., Ishiwata, E.: A variant of IDRstab with reliable update strategies for solving sparse linear systems. *J. Comput. Appl. Math.* **259**, 244–258 (2014)
14. Aihara, K., Abe, K., Ishiwata, E.: Preconditioned IDRStab algorithms for solving nonsymmetric linear systems. *Int. J. Appl. Math.* **45**(3) (2015)
15. Al Daas, H., Grigori, L., Hénon, P., Ricoux, P.: Enlarged GMRES for solving linear systems with one or multiple right-hand sides. *IMA J. Numer. Anal.* **39**(4), 1924–1956 (2019)
16. Aliaga, J.L., Boley, D., Freund, R., Hernández, V.: A Lanczos-type method for multiple starting vectors. *Math. Comput.* **69**(232), 1577–1601 (2000)

17. Ammar, G.S., Gragg, W.B., Reichel, L.: Constructing a unitary Hessenberg matrix from spectral data. In: Golub, G.H., Van Dooren, P. (eds.) Numerical Linear Algebra, Digital Signal Processing, and Parallel Algorithms, pp. 385–396. Springer (1991)
18. Ammar, G.S., He, C.: On an inverse eigenvalue problem for unitary Hessenberg matrices. *Linear Algebra Appl.* **218**, 263–271 (1995)
19. Amritkar, A., de Sturler, E., Swirydowicz, K., Tafti, D., Ahuja, K.: Recycling Krylov subspaces for CFD applications. *J. Comp. Phys.* **303**, 222–237 (2015)
20. Anciaux-Sedrakian, A., Grigori, L., Moufawad, S., Yousef, S.: S-step BiCGStab algorithms for geoscience dynamic simulations. *Oil & Gas Science and Technology, Revue d'IFP Energies nouvelles* **71**(66) (2016)
21. Anderson, D.G.: Iterative procedures for nonlinear integral equations. *J. ACM* **12**, 547–560 (1965)
22. Arioli, M.: A stopping criterion for the conjugate gradient algorithm in a finite element method framework. *Numer. Math.* **97**, 1–24 (2004)
23. Arioli, M.: An analysis of GMRES worst case convergence (2009). Unpublished report
24. Arioli, M., Demmel, J.W., Duff, I.S.: Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.* **10**(2), 165–190 (1989)
25. Arioli, M., Duff, I.S.: Using FGMRES to obtain backward stability in mixed precision. *Electron. Trans. Numer. Anal.* **33**, 31–44 (2009)
26. Arioli, M., Duff, I.S., Gratton, S., Pralet, S.: A note on GMRES preconditioned by a perturbed LDL^T decomposition with static pivoting. *SIAM J. Sci. Comput.* **29**(5), 2024–2044 (2007)
27. Arioli, M., Duff, I.S., Ruiz, D.: Stopping criteria for iterative solvers. *SIAM J. Matrix Anal. Appl.* **13**(1), 138–144 (1992)
28. Arioli, M., Fassino, C.: Roundoff error analysis of algorithms based on Krylov subspace methods. *BIT Numer. Math.* **36**(2), 189–205 (1996)
29. Arioli, M., Georgoulis, E.H., Loghin, D.: Stopping criteria for adaptive finite element solvers. *SIAM J. Sci. Comput.* **35**(3), A1537–A1559 (2013)
30. Arioli, M., Liesen, J., Międlar, A., Strakoš, Z.: Interplay between discretization and algebraic computation in adaptive numerical solution of elliptic PDE problems. *GAMM Mitt.* **36**, 102–129 (2013)
31. Arioli, M., Loghin, D.: Stopping criteria for mixed finite element problems. *Electron. Trans. Numer. Anal.* **29**, 178–192 (2007)
32. Arioli, M., Loghin, D., Wathen, A.: Stopping criteria for iterations in finite element methods. *Numer. Math.* **99**, 381–410 (2005)
33. Arioli, M., Noulard, E., Russo, A.: Stopping criteria for iterative methods: applications to PDE's. *Calcolo* **38**(2), 97–112 (2001)
34. Arioli, M., Pták, V., Strakoš, Z.: Krylov sequences of maximal length and convergence of GMRES. *BIT Numer. Math.* **38**(4), 636–643 (1998)
35. Arnoldi, W.: The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.* **9**(1), 17–29 (1951)
36. Ashby, S.F., Manteuffel, T.A., Saylor, P.: A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.* **27**(6), 1542–1568 (1990)
37. Ashby, T., Ghysels, P., Heirman, W., Vanroose, W.: The impact of global communication latency at extreme scales on Krylov methods. In: Xiang, Y., Stojmenovic, I., Apduhan, B.O., Wang, G., Nakano, K., Zomaya, A.Y. (eds.) Algorithms and Architectures for Parallel Processing, pp. 428–442, : 12th International Conference, ICA3PP 2012. Fukuoka, Japan (2012)
38. Asplund, E.: Inverses of matrices $a_{i,j}$ which satisfies $a_{i,j} = 0$ for $i > j + p$. *Math. Scand.* **7**, 57–60 (1959)
39. Astudillo, R., de Gier, J.M., van Gijzen, M.B.: Accelerating the Induced Dimension Reduction method using spectral information. *J. Comput. Appl. Math.* **345**, 33–47 (2019)
40. Astudillo, R., van Gijzen, M.B.: The induced dimension reduction method applied to convection-diffusion-reaction problems. In: Karasözen, B., Manguoglu, M., Tezer-Sezgin,

- M., Goktepe, S., Ugur, O. (eds.) Numerical Mathematics and Advanced Applications ENUMATH 2015, Lecture Notes in Computational Science and Engineering, vol. 112, pp. 295–303 (2016)
41. Astudillo, R., van Gijzen, M.B.: A restarted induced dimension reduction method to approximate eigenpairs of large unsymmetric matrices. *J. Comput. Appl. Math.* **296**, 24–36 (2016)
 42. Auchmuty, G.: A posteriori error estimates for linear equations. *Numer. Math.* **61**(1), 1–6 (1992)
 43. Axelsson, O.: Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Linear Algebra Appl.* **29**, 1–16 (1980)
 44. Axelsson, O.: A generalized conjugate gradient, least square method. *Numer. Math.* **51**(2), 209–227 (1987)
 45. Axelsson, O.: Iterative Solution Methods. Cambridge University Press (1994)
 46. Ayachour, E.H.: Application de la biorthogonalité aux méthodes de projection. Ph.D. thesis, Université de Lille, France (1998)
 47. Ayachour, E.H.: Avoiding look-ahead in the Lanczos method and Padé approximation. *Appl. Math.* **26**(1), 33–62 (1999)
 48. Ayachour, E.H.: Expanded systems and the ILU preconditioner for solving non-Hermitian linear systems. *Linear Algebra Appl.* **293**(1–3), 243–256 (1999)
 49. Ayachour, E.H.: A fast implementation for GMRES method. *J. Comput. Appl. Math.* **159**(2), 269–283 (2003)
 50. Baglama, J., Calvetti, D., Golub, G.H., Reichel, L.: Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput.* **20**(1), 243–269 (1998)
 51. Baglama, J., Calvetti, D., Reichel, L.: Fast Leja points. *Electron. Trans. Numer. Anal.* **7**, 124–140 (1998)
 52. Baglama, J., Reichel, L.: Augmented GMRES-type methods. *Numer. Linear Algebra Appl.* **14**(4), 337–350 (2007)
 53. Baheux, C.: Algorithmes d’implémentation de la méthode de Lanczos. Ph.D. thesis, Université de Lille, France (1994). (in French)
 54. Baheux, C.: New implementations of Lanczos method. *J. Comput. Appl. Math.* **57**, 3–15 (1995)
 55. Bai, Z.: Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem. *Math. Comput.* **62**(205), 209–226 (1994)
 56. Bai, Z., Day, D., Ye, Q.: ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **20**(4), 1060–1082 (1999)
 57. Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., van der Vorst, H.: Templates for the Solution of Algebraic Eigenvalue Problems. SIAM (2000)
 58. Bai, Z., Hu, D., Reichel, L.: Implementation of GMRES method using QR factorisation. In: Dongarra, J., Kennedy, K., Messina, P., Sorensen, D.C., Voigt, R.G. (eds.) Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, pp. 84–91. SIAM (1992)
 59. Bai, Z., Hu, D., Reichel, L.: A Newton basis GMRES implementation. *IMA J. Numer. Anal.* **14**, 563–581 (1994)
 60. Baker, A.H.: On improving the performance of the linear solver GMRES. Ph.D. thesis, University of Colorado, USA (2003)
 61. Baker, A.H., Dennis, J.M., Jessup, E.R.: An efficient block variant of GMRES. Tech. Rep. CU-CS-957-03, University of Colorado, Boulder (USA) (2003)
 62. Baker, A.H., Dennis, J.M., Jessup, E.R.: On improving linear solver performance: a block variant of GMRES. *SIAM J. Sci. Comput.* **27**(5), 1608–1626 (2006)
 63. Baker, A.H., Jessup, E.R., Kolev, T.V.: A simple strategy for varying the restart parameter in GMRES(m). *J. Comput. Appl. Math.* **230**(2), 751–761 (2009)
 64. Baker, A.H., Jessup, E.R., Manteuffel, T.A.: A technique for accelerating the convergence of restarted GMRES. *SIAM J. Matrix Anal. Appl.* **26**(4), 962–984 (2005)

65. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., May, D.A., McInnes, L.C., Mills, R.T., Munson, T., Rupp, K., Sanan, P., Smith, B.F., Zampini, S., Zhang, H.: PETSc user's manual. Tech. Rep. ANL-95/11 - Revision 3.10, Argonne National Laboratory (2018)
66. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) Modern Software Tools in Scientific Computing, pp. 163–202. Birkhäuser (1997)
67. Ballard, G., Carson, E., Demmel, J., Hoemmen, M., Knight, N., Schwartz, O.: Communication lower bounds and optimal algorithms for numerical linear algebra. *Acta Numer.* **23**, 1–155 (2014)
68. Bank, R.E., Chan, T.F.: An analysis of the composite step biconjugate gradient method. *Numer. Math.* **66**(1), 295–319 (1993)
69. Bank, R.E., Chan, T.F.: A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems. *Numer. Algorithms* **7**(1), 1–16 (1994)
70. Baranger, J., Duc-Jacquet, M.: Matrices tridiagonales symétriques et matrices factorisables. *RIRO* **3**, 61–66 (1971)
71. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM (1994)
72. Barrett, W.W., Feinsilver, P.J.: Inverses of banded matrices. *Linear Algebra Appl.* **41**, 111–130 (1981)
73. Barth, T., Manteuffel, T.A.: Variable metric conjugate gradient methods. In: Natori, M., Nodera, T. (eds.) Advances in Numerical Methods for Large Sparse Sets of Linear Equations, Matrix Analysis and Parallel Computing, PCG 94, pp. 165–188 (1994)
74. Barth, T., Manteuffel, T.A.: Multiple recursion conjugate gradient algorithms part I: sufficient conditions. *SIAM J. Matrix Anal. Appl.* **21**(3), 768–796 (2000)
75. Bartlett, M.S.: An inverse matrix adjustment arising in discriminant analysis. *Ann. Math. Statist.* **22**(1), 107–111 (1951)
76. Bauer, F.L., Fike, C.T.: Norms and exclusion theorems. *Numer. Math.* **2**(1), 137–141 (1960)
77. Bauer, F.L., Householder, A.S.: Moments and characteristic roots. *Numer. Math.* **2**(1), 42–53 (1960)
78. Baumann, M., van Gijzen, M.B.: Nested Krylov methods for shifted linear systems. *SIAM J. Sci. Comput.* **37**(5), S90–S112 (2015)
79. Bavier, E., Hoemmen, M., Rajamanickam, S., Thornquist, H.: Amesos2 and Belos: direct and iterative solvers for large sparse linear systems. *Sci. Program.* **20**, 241–255 (2012)
80. Bazán, F., Gratton, S.: An explicit Jordan decomposition of companion matrices. *TEMA* **7**(2), 209–218 (2011)
81. Beattie, C., Embree, M., Rossi, J.: Convergence of restarted Krylov subspaces to invariant subspaces. *SIAM J. Matrix Anal. Appl.* **25**(4), 1074–1109 (2004)
82. Beckermann, B.: The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numer. Math.* **85**, 553–577 (2000)
83. Beckermann, B.: Image numérique, GMRES et polynômes de Faber. *C.R. Acad. Sci. Paris, Sér 1* **340**, 855–860 (2005)
84. Beckermann, B., Goreinov, S.A., Tyrtyshnikov, E.E.: Some remarks on the Elman estimate for GMRES. *SIAM J. Matrix Anal. Appl.* **27**(3), 772–778 (2005)
85. Beckermann, B., Reichel, L.: The Arnoldi process and GMRES for nearly symmetric matrices. *SIAM J. Matrix Anal. Appl.* **30**(1), 102–120 (2008)
86. Bellalij, M., Jbilou, K., Sadok, H.: New convergence results on the global GMRES method for diagonalizable matrices. *J. Comput. Appl. Math.* **219**, 350–358 (2008)
87. Bellalij, M., Meurant, G., Sadok, H.: The distance of an eigenvector to a Krylov subspace and the convergence of the Arnoldi method for eigenvalue problems. *Linear Algebra Appl.* **504**, 387–405 (2016)

88. Bellalij, M., Reichel, L., Sadok, H.: Some properties of range restricted GMRES methods. *J. Comput. Appl. Math.* **290**, 310–318 (2015)
89. Bellalij, M., Saad, Y., Sadok, H.: On the convergence of the Arnoldi process for eigenvalue problems. Tech. Rep. umsi-2007-12, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN USA (2007)
90. Bellalij, M., Saad, Y., Sadok, H.: Analysis of some Krylov subspace methods for normal matrices via approximation theory and convex optimization. *Electron. Trans. Numer. Anal.* **33**, 17–30 (2008)
91. Bellalij, M., Saad, Y., Sadok, H.: Further analysis of the Arnoldi process for eigenvalue problems. *SIAM J. Numer. Anal.* **48**(2), 393–407 (2010)
92. Benhamadou, M.: On the FOM algorithm for the resolution of the linear systems $Ax = b$. *ALAMT* **4**, 156–171 (2014)
93. Benzi, M., Boito, P.: Quadrature rule-based bounds for functions of adjacency matrices. *Linear Algebra Appl.* **433**(3), 637–652 (2010)
94. Benzi, M., Estrada, E., Klymko, C.: Ranking hubs and authorities using matrix functions. *Linear Algebra Appl.* **438**(5), 2447–2474 (2013)
95. Bhatia, R., Elsner, L., Krause, G.: Bounds for the variation of the roots of a polynomial and the eigenvalues of a matrix. *Linear Algebra Appl.* **142**, 195–209 (1990)
96. Bindel, D., Demmel, J., Kahan, W., Marques, O.: On computing Givens rotations reliably and efficiently. *ACM Trans. Math. Soft. (TOMS)* **28**(2), 206–238 (2002)
97. Bini, D., Boito, P., Eidelman, Y., Gemignani, L., Gohberg, I.: A fast implicit QR eigenvalue algorithm for companion matrices. *Linear Algebra Appl.* **432**(8), 2006–2031 (2010)
98. Björck, A.: Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT Numer. Math.* **7**(1), 1–21 (1967)
99. Björck, A.: Numerics of Gram-Schmidt orthogonalization. *Linear Algebra Appl.* **197**, 297–316 (1994)
100. Björck, A.: Numerical Methods for Least Squares Problems. SIAM (1996)
101. Björck, A.: The calculation of linear least squares problems. *Acta Numer.* **13**, 1–53 (2004)
102. Björck, A., Golub, G.H.: Numerical methods for computing angles between linear subspaces. *Math. Comput.* **27**, 579–594 (1973)
103. Björck, A., Paige, C.C.: Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm. *SIAM J. Matrix Anal. Appl.* **13**(1), 176–190 (1992)
104. Björck, A., Pereyra, V.: Solution of Vandermonde systems of equations. *Math. Comput.* **24**(112), 893–903 (1970)
105. Bohnhorst, B., Bunse-Gerstner, A., Fassbender, H.: On the perturbation theory for unitary eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **21**, 809–824 (2000)
106. Boley, D.L., Elhay, S., Golub, G.H., Gutknecht, M.H.: Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights. *Numer. Algorithms* **1**(1), 21–43 (1991)
107. Boley, D.L., Golub, G.H.: The Lanczos-Arnoldi algorithm and controllability. *Syst. Control Lett.* **4**(6), 317–324 (1984)
108. Boley, D.L., Golub, G.H.: The nonsymmetric Lanczos algorithm and controllability. *Syst. Control Lett.* **16**(2), 97–105 (1991)
109. Boley, D.L., Lee, T.J., Luk, F.T.: The Lanczos algorithm and Hankel matrix factorization. *Linear Algebra Appl.* **172**, 109–133 (1992)
110. Bollen, J.A.M.: Round-off error analysis of descent methods for solving linear equations. Ph.D. thesis, Technische Hogeschool Eindhoven, The Netherlands (1980)
111. Bollen, J.A.M.: Numerical stability of descent methods for solving linear equations. *Numer. Math.* **43**, 361–377 (1984)
112. Booijhawon, R., Bhuruth, M.: Restarted simpler GMRES augmented with harmonic Ritz vectors. *Future Gener. Comput. Syst.* **20**(3), 389–397 (2004)
113. Bouhamidi, A., Jbilou, K., Reichel, L., Sadok, H.: A generalized global Arnoldi method for ill-posed matrix equations. *J. Comput. Appl. Math.* **236**, 2078–2089 (2012)

114. Bouras, A., Frayssé, V.: A relaxation strategy for inexact matrix-vector products for Krylov methods. Tech. Rep. TR/PA/00/15, CERFACS (2000)
115. Bouras, A., Frayssé, V.: Inexact matrix-vector products in Krylov methods for solving linear equations: a relaxation strategy. SIAM J. Matrix Anal. Appl. **26**(3), 660–675 (2005)
116. Bouwmeester, H., Jacquelin, M., Langou, J., Robert, Y.: Tiled QR factorization algorithms. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ACM (2011)
117. Bouyouli, R., Jbilou, K., Messaoudi, A., Sadok, H.: On block minimal residual methods. Appl. Math. Lett. **20**(3), 284–289 (2007)
118. Bozzo, E., Fasino, D.: A priori estimates on the structured conditioning of Cauchy and Vandermonde matrices. In: Numerical Methods for Structured Matrices and Applications, pp. 203–220. Birkhäuser, Basel (2010)
119. Braconnier, T., Langlois, P., Rioual, J.: The influence of orthogonality on the Arnoldi method. Linear Algebra Appl. **309**, 307–323 (2000)
120. Brand, L.: The companion matrix and its properties. Am. Math. Mon. **71**(6), 629–634 (1964)
121. Brand, L.: Applications of the companion matrix. Am. Math. Mon. **75**(2), 146–152 (1968)
122. Brandts, J.: Deliberate ill-conditioning of Krylov matrices. Tech. Rep. 1181, Mathematics Institute, University of Utrecht (2001)
123. Brezinski, C.: Rational approximation to formal power series. J. Approx. Theory **25**, 295–317 (1979)
124. Brezinski, C.: Padé-Type Approximation and General Orthogonal Polynomials, vol. ISNM 50. Birkhäuser, Basel (1980)
125. Brezinski, C.: Biorthogonality and Its Applications to Numerical Analysis. Marcel Dekker, New York (1992)
126. Brezinski, C.: The methods of Vorobyev and Lanczos. Linear Algebra Appl. **234**, 21–41 (1996)
127. Brezinski, C.: A transpose-free Lanczos/Orthodir algorithm for linear systems. C.R. Acad. Sci. Paris, Sér I **324**, 349–354 (1997)
128. Brezinski, C.: Error estimates in the solution of linear systems. SIAM J. Sci. Comput. **21**(2), 764–781 (1999)
129. Brezinski, C.: Krylov subspace methods, biorthogonal polynomials and Padé-type approximants. Numer. Algorithms **21**, 97–107 (1999)
130. Brezinski, C.: Formal orthogonal polynomials. In: Computational Aspects of Linear Control, pp. 73–85. Springer, Boston, MA (2002)
131. Brezinski, C., Redivo-Zaglia, M.: Breakdowns in the computation of orthogonal polynomials. In: Cuyt, A. (ed.) Nonlinear Numerical Methods and Rational Approximation, II, pp. 49–59. Kluwer, Dordrecht (1994)
132. Brezinski, C., Redivo-Zaglia, M.: Hybrid procedure for solving linear systems. Numer. Math. **67**, 1–19 (1994)
133. Brezinski, C., Redivo-Zaglia, M.: Treatment of near-breakdown in the CGS algorithm. Numer. Algorithms **7**, 33–73 (1994)
134. Brezinski, C., Redivo-Zaglia, M.: Look-ahead in BiCGstab and other methods for linear systems. BIT **35**, 169–201 (1995)
135. Brezinski, C., Redivo-Zaglia, M.: Transpose-free Lanczos-type algorithms for nonsymmetric linear systems. Numer. Algorithms **17**, 67–103 (1998)
136. Brezinski, C., Redivo-Zaglia, M.: On the choice of the auxiliary vectors in Lanczos' method for linear systems. Tech. Rep. ANO-402, Université de Lille, France (1999)
137. Brezinski, C., Redivo-Zaglia, M.: Variations on Lanczos' tridiagonalization process. Calcolo **37**, 159–179 (2000)
138. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: Avoiding breakdown and near-breakdown in Lanczos type algorithms. Numer. Algorithms **1**, 261–284 (1991)
139. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: Addendum to “Avoiding breakdown and near-breakdown in Lanczos type algorithms”. Numer. Algorithms **2**, 133–136 (1992)

140. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: A breakdown-free Lanczos type algorithm for solving linear systems. *Numer. Math.* **63**, 29–38 (1992)
141. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: Breakdowns in the implementation of the Lanczos method for solving linear systems. *Comput. Math. Appl.* **33**(1), 31–44 (1997)
142. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: New look-ahead Lanczos-type algorithms for linear systems. *Numer. Math.* **83**(1), 53–85 (1999)
143. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: The matrix and polynomial approaches to Lanczos-type algorithms. *J. Comput. Appl. Math.* **123**, 241–260 (2000)
144. Brezinski, C., Redivo-Zaglia, M., Sadok, H.: A review of formal orthogonality in Lanczos-based methods. *J. Comput. Appl. Math.* **140**(1), 81–98 (2002)
145. Brezinski, C., Rodriguez, G., Seatzu, S.: Error estimates for linear systems with applications to regularization. *Numer. Algorithms* **49**, 85–104 (2008)
146. Brezinski, C., Rodriguez, G., Seatzu, S.: Error estimates for the regularization of least squares problems. *Numer. Algorithms* **51**(1), 61–76 (2009)
147. Brezinski, C., Sadok, H.: Avoiding breakdown in the CGS algorithm. *Numer. Algorithms* **1**, 199–206 (1991)
148. Brezinski, C., Sadok, H.: Lanczos type algorithms for solving systems of linear equations. *Appl. Numer. Math.* **11**, 443–473 (1993)
149. Brezinski, C., Tournès, D.: André-Louis Cholesky, Mathematician. Topographer and Army Officer. Birkhäuser (2014)
150. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial, 2nd edn. SIAM (2000)
151. Brown, J.D., Chu, M.T., Ellison, D.C., Plemmons, R.J. (eds.): Proceedings of the Cornelius Lanczos International Centenary Conference, 1993. SIAM (1994)
152. Brown, P.N.: A theoretical comparison of the Arnoldi and GMRES algorithms. *SIAM J. Sci. Statist. Comput.* **12**(1), 58–78 (1991)
153. Brown, P.N., Hindmarsh, A.C.: Reduced storage matrix methods in stiff ODE systems. *Appl. Math. Comput.* **31**, 40–91 (1989)
154. Brown, P.N., Walker, H.F.: GMRES on (nearly) singular systems. *SIAM J. Matrix Anal. Appl.* **18**(1), 37–51 (1997)
155. Bücker, H.M., Sauren, M.: A parallel version of the quasi-minimal residual method based on coupled two-term recurrences. In: International Workshop on Applied Parallel Computing. Springer, Berlin, Heidelberg (1996)
156. Bücker, H.M., Sauren, M.: A variant of the biconjugate gradient method suitable for massively parallel computing. In: International Symposium on Solving Irregularly Structured Problems in Parallel. Springer, Berlin, Heidelberg (1997)
157. Bücker, H.M., Sauren, M.: Reducing global synchronization in the biconjugate gradient method. In: Yang, L.T. (ed.) Parallel Numerical Computation with Applications, pp. 63–76. Springer, Boston, MA (1999)
158. Bujanović, Z.: Krylov type methods for large scale eigenvalue problems. Ph.D. thesis, Department of Mathematics, University of Zagreb, Croatia (2011)
159. Bujanović, Z.: On the permissible arrangements of Ritz values for normal matrices in the complex plane. *Linear Algebra Appl.* **438**, 4606–4624 (2013)
160. Bunse-Gerstner, A., Elsner, L.: Schur parameter pencils for the solution of the unitary eigenproblem. *Linear Algebra Appl.* **154**(156), 741–778 (1991)
161. Burke, J.V., Greenbaum, A.: Some equivalent characterizations of the polynomial numerical hull of degree k . Tech. Rep. NA-04-29, The Mathematical Institute, University of Oxford (2004)
162. Burke, J.V., Greenbaum, A.: Characterizations of the polynomial numerical hull of degree k . *Linear Algebra Appl.* **419**, 37–47 (2006)
163. Burrage, K., Erhel, J.: On the performance of various adaptive preconditioned GMRES strategies. *Numer. Linear Algebra Appl.* **5**, 101–121 (1998)
164. Burrage, K., Erhel, J., Pohl, B., Williams, A.: A deflation technique for linear systems of equations. *SIAM J. Sci. Comput.* **19**(4), 1245–1260 (1998)

165. Buttari, A., Langou, J., Kurzak, J., Dongarra, J.: Parallel tiled QR factorization for multicore architectures. *Concurr. Comp.-Pract. E.* **20**(13), 1573–1590 (2008)
166. Calandra, H., Gratton, S., Lago, R., Vasseur, X., Carvalho, L.M.: A modified block flexible GMRES method with deflation at each iteration for the solution of non-Hermitian linear systems with multiple right-hand sides. *SIAM J. Sci. Comput.* **35**(5), S345–S367 (2013)
167. Calandra, H., Gratton, S., Langou, J., Pinel, X., Vasseur, X.: Flexible variants of block restarted GMRES methods with application to geophysics. *SIAM J. Sci. Comput.* **34**(2), A714–A736 (2012)
168. Calvetti, D., Golub, G.H., Reichel, L.: An adaptive Chebyshev iterative method for nonsymmetric linear systems based on modified moments. *Numer. Math.* **67**(11), 21–40 (1994)
169. Calvetti, D., Kim, S.M., Reichel, L.: Quadrature rules based on the Arnoldi process. *SIAM J. Matrix Anal. Appl.* **26**(3), 765–781 (2005)
170. Calvetti, D., Lewis, B., Reichel, L.: GMRES-type methods for inconsistent systems. *Linear Algebra Appl.* **316**(1–3), 157–169 (2000)
171. Calvetti, D., Lewis, B., Reichel, L.: GMRES, L-curves, and discrete ill-posed problems. *BIT Numer. Math.* **42**(1), 44–65 (2002)
172. Calvetti, D., Lewis, B., Reichel, L.: On the regularizing properties of the GMRES method. *Numer. Math.* **91**(4), 605–625 (2002)
173. Calvetti, D., Petersen, J., Reichel, L.: A parallel implementation of the GMRES method. In: Reichel, L., Ruttan, A., Varga, R.S. (eds.) *Numerical Linear Algebra*, pp. 31–45. De Gruyter, Berlin (1993)
174. Calvetti, D., Reichel, L.: On the evaluation of polynomial coefficients. *Numer. Algorithms* **33**(1), 153–161 (2003)
175. Calvetti, D., Reichel, L., Sgallari, F.: Applications of anti-Gauss quadrature rules in linear algebra. In: *Applications and Computation of Orthogonal Polynomials*, pp. 41–56. Birkhäuser, Basel (1999)
176. Calvetti, D., Reichel, L., Sgallari, F.: A modified companion matrix method based on Newton polynomials. In: *Contemporary Mathematics*, pp. 179–186. American Mathematical Society (2001)
177. Campbell, S.L., Ipsen, I.C.F., Kelley, C.T., Meyer, C.D.: GMRES and the minimal polynomial. *BIT Numer. Math.* **36**(4), 664–675 (1996)
178. Cao, Z.H.: Avoiding breakdown in variants of the BI-CGSTAB algorithm. *Linear Algebra Appl.* **263**, 113–132 (1997)
179. Cao, Z.H.: On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems. *Appl. Numer. Math.* **27**(2), 123–140 (1998)
180. Cao, Z.H., Wang, M.: A note on Krylov subspace methods for singular systems. *Linear Algebra Appl.* **350**, 285–288 (2002)
181. Carden, R.: Ritz values and Arnoldi convergence for non-Hermitian matrices. Ph.D. thesis, Rice University, Houston, USA (2011)
182. Carden, R., Embree, M.: Ritz value localization for non-Hermitian matrices. *SIAM J. Matrix Anal. Appl.* **33**(4), 1320–1338 (2012)
183. Carden, R., Hansen, D.: Ritz values of normal matrices and Ceva's theorem. *Linear Algebra Appl.* **438**(11), 4114–4129 (2013)
184. Carpenter, M.H., Vuik, C., Lucas, P., van Gijzen, M.B., Bijl, H.: A general algorithm for reusing Krylov subspace information. I. unsteady Navier-Stokes. *Tech. Rep. NASA/TM 2010 216190*, Langley Research Center, Hampton, Virginia (USA) (2010)
185. Carpentieri, B., Jing, Y.F., Huang, T.Z.: The BiCOR and CORS algorithms for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.* **33**(5), 3020–3036 (2011)
186. Carpraux, J.F., Godunov, S.K., Kuznetsov, S.V.: Condition number of the Krylov bases and subspaces. *Linear Algebra Appl.* **248**, 137–160 (1996)
187. Carriou, H., Fadel, M., Fieux, E., Lassère, P., Rodriguez, F.: Autour des matrices de Frobenius et compagnon. Report available at www.math.univ-toulouse.fr/~assere/pdf/vfcomp.pdf (2007). (in French)

188. Carson, E.: Communication-avoiding Krylov subspace methods in theory and practice. Ph.D. thesis, University of California at Berkeley (2015). Technical Report UCB/EECS-2015-179
189. Carson, E.: The adaptive s-step conjugate gradient method. *SIAM J. Matrix Anal. Appl.* **39**(3), 1318–1338 (2018)
190. Carson, E., Demmel, J.: Error analysis of the s-step biconjugate gradient method in finite precision. Tech. Rep. UCB/EECS-2014-18, EECS Department, University of California, Berkeley, CA (2014)
191. Carson, E., Demmel, J.: Error analysis of the s-step Lanczos method in finite precision. Tech. Rep. UCB/EECS-2014-55, EECS Department, University of California, Berkeley, CA (2014)
192. Carson, E., Demmel, J.: A residual replacement strategy for improving the maximum attainable accuracy of s-step Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **35**(1), 22–43 (2014)
193. Carson, E., Knight, N., Demmel, J.: Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods. *SIAM J. Sci. Comput.* **35**(5), S42–S61 (2013)
194. Carson, E., Rozložník, M., Strakoš, Z., Tichý, P., Tůma, M.: The numerical stability analysis of pipelined conjugate gradient methods: historical context and methodology. *SIAM J. Sci. Comput.* **40**(5), A3549–A3580 (2018)
195. Carvalho, L.M., Gratton, S., Lago, R.F., Vasseur, X.: Augmentation and truncation for iterative methods. *Anais do CNMAC, Brazil* **3** (2010)
196. Carvalho, L.M., Gratton, S., Lago, R.F., Vasseur, X.: A flexible generalized conjugate residual method with inner orthogonalization and deflated restarting. *SIAM J. Matrix Anal. Appl.* **32**(4), 1212–1235 (2011)
197. Catabriga, L., Coutinho, A., Franca, L.P.: Evaluating the LCD algorithm for solving linear systems of equations arising from implicit SUPG formulation of compressible flows. *Int. J. Numer. Methods Eng.* **60**(9), 1513–1534 (2004)
198. Catabriga, L., Valli, A., Melotti, B., Pessoa, L., Coutinho, A.: Performance of LCD iterative method in the finite element and finite difference solution of convection-diffusion equations. *Commun. Numer. Methods Eng.* **22**(6), 643–656 (2006)
199. Causey, R.L., Gregory, R.: On Lanczos' algorithm for tridiagonalizing matrices. *SIAM Rev.* **3**(4), 322–328 (1961)
200. Chaitin-Chatelin, F., Frayssé, V.: *Lectures on Finite Precision Computations*. SIAM (1996)
201. Chaitin-Chatelin, F., Traviesas, E., Plantié, L.: Understanding Krylov methods in finite precision. In: International Conference on Numerical Analysis and Its Applications. Springer, Berlin, Heidelberg (2000)
202. Chan, R.H., Greif, C., O'Leary, D.P.: *Milestones in Matrix Computation: The Selected Works of Gene H. Golub with Commentaries*. Oxford University Press, Oxford (2007)
203. Chan, T.F., De Pillis, L., van der Vorst, H.: A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems. Tech. Rep. 91-17 CAM, Department of Mathematics, UCLA (1991)
204. Chan, T.F., De Pillis, L., van der Vorst, H.: Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems. *Numer. Algorithms* **17**(1), 51–66 (1998)
205. Chan, T.F., Gallopoulos, E., Simoncini, V., Szeto, T., Tong, C.H.: A quasi-minimal residual variant of the Bi-CGSTab algorithm for nonsymmetric systems. *SIAM J. Sci. Comput.* **15**(2), 338–347 (1994)
206. Chan, T.F., Szeto, T.: A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems. *Numer. Algorithms* **7**(1), 17–32 (1994)
207. Chan, T.F., Szeto, T.: Composite step product methods for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.* **17**(6), 1491–1508 (1996)
208. Chan, T.F., Wan, W.: Analysis of projection methods for solving linear systems with multiple right hand sides. *SIAM J. Sci. Comput.* **18**(6), 1698–1721 (1997)
209. Chan, T.F., Ye, Q.: A mixed product Krylov subspace method for solving nonsymmetric linear systems. *Asian J. Math.* **1**(3), 422–434 (1997)

210. Chandra, R.: Conjugate gradient methods for partial differential equations. Ph.D. thesis, Department of Computer Science, Yale University (1978). Report 129
211. Chandra, R., Eisenstat, S.C., Schultz, M.H.: The modified conjugate residual method for partial differential equations. In: Vichnevetsky, R. (ed.) *Advances in Computer Methods for Partial Differential Equations II*, pp. 13–19 (1977)
212. Chang, X.W., Paige, C.C., Titley-Péloquin, D.: Characterizing matrices that are consistent with given solutions. *SIAM J. Matrix Anal. Appl.* **30**(4), 1406–1420 (2008)
213. Chapman, A., Saad, Y.: Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl.* **4**, 43–66 (1997)
214. Chen, G., Jia, Z.: A reverse order implicit Q -theorem and the Arnoldi process. *J. Comput. Math.* **20**(5), 519–524 (2002)
215. Chen, G., Jia, Z.: An analogue of the results of Saad and Stewart for harmonic Ritz vectors. *J. Comput. Appl. Math.* **167**(2), 493–498 (2004)
216. Chen, G., Jia, Z.: Theoretical and numerical comparisons of GMRES and WZ-GMRES. *Comput. Math. Appl.* **47**(8–9), 1335–1350 (2004)
217. Chen, J., McInnes, L.C., Zhang, H.: Analysis and practical use of flexible BiCGSTAB. *J. Sci. Comput.* **68**(2), 803–825 (2016)
218. Chesneaux, J.M., Matos, A.: Breakdown and near-breakdown control in the CGS algorithm using stochastic arithmetic. *Numer. Algorithms* **11**, 99–116 (1996)
219. Choi, D., Greenbaum, A.: Roots of matrices in the study of GMRES convergence and Crouzeix's conjecture. *SIAM J. Matrix Anal. Appl.* **36**(1), 289–301 (2015)
220. Choi, S.C.T., Paige, C.C., Saunders, M.A.: MINRES-QLP: a Krylov subspace method for indefinite or singular symmetric systems. *SIAM J. Sci. Comput.* **33**(4), 1810–1836 (2011)
221. Chou, A.W.: On the optimality of Krylov information. *J. Complex.* **3**(1), 26–40 (1987)
222. Chronopoulos, A.T.: S-step iterative methods for (non)symmetric indefinite linear systems. *SIAM J. Numer. Anal.* **28**(6), 1776–1789 (1991)
223. Chronopoulos, A.T.: On the squared unsymmetric Lanczos method. *J. Comput. Appl. Math.* **54**(1), 65–78 (1994)
224. Chronopoulos, A.T., Gear, C.W.: On the efficient implementation of preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy. *Parallel Comput.* **11**(1), 37–53 (1989)
225. Chronopoulos, A.T., Kim, S.: S-step Orthomin and GMRES implemented on parallel computers. Tech. Rep. 90/43R, USMI, Minneapolis, MN, USA (1990)
226. Chronopoulos, A.T., Kucherov, A.B.: Block s-step Krylov iterative methods. *Numer. Linear Algebra Appl.* **17**(1), 3–15 (2010)
227. Chronopoulos, A.T., Ma, S.: On squaring Krylov subspace iterative methods for nonsymmetric linear systems. Tech. Rep. 89-67, Department of Computer Science, University of Minnesota (1989)
228. Chronopoulos, A.T., Swanson, C.D.: Parallel iterative s-step methods for unsymmetric linear systems. *Parallel Comput.* **22**(5), 623–641 (1996)
229. Collignon, T.P.: Efficient iterative solution of large linear systems on heterogeneous computing systems. Ph.D. thesis, Delft University of Technology, The Netherlands (2013)
230. Collignon, T.P., van Gijzen, M.B.: Minimizing synchronization in IDR(s). *Numer. Linear Algebra Appl.* **18**(5), 805–825 (2011)
231. Collignon, T.P., Sleijpen, G.L.G., van Gijzen, M.B.: Interpreting IDR(s) as a deflation process. Tech. Rep. 10-21, Delft University of Technology (2010)
232. Concus, P., Golub, G.H.: A generalized conjugate gradient method for nonsymmetric systems of linear equations. In: *Computing Methods in Applied Sciences and Engineering*, pp. 56–65. Springer, Berlin, Heidelberg (1976)
233. Cools, S.: Analyzing and improving maximal attainable accuracy in the communication hiding pipelined BiCGStab method. *Parallel Comput.* **86**, 16–35 (2019). PMAA'18 Special Issue
234. Cools, S., Cornelis, J., Vanroose, W.: Numerically stable variants of the communication-hiding pipelined conjugate gradient method. *IEEE Trans. Parallel Distrib. Syst.* **30**(11), 2507–2522 (2019)

235. Cools, S., Vanroose, W.: The communication-hiding pipelined BiCGStab method for the efficient parallel solution of large unsymmetric linear systems. *Parallel Comput.* **65**, 1–20 (2017)
236. Coulaud, O., Giraud, L., Ramet, P., Vasseur, X.: Deflation and augmentation techniques in Krylov subspace methods for the solution of linear systems. arXiv preprint, [arXiv:1303.5692](https://arxiv.org/abs/1303.5692) (2013)
237. Crouzeix, M.: Bounds for analytic functions of matrices. *Integr. Equ. Oper. Theory* **48**, 461–477 (2004)
238. Crouzeix, M.: Numerical range and functional calculus in Hilbert space. *J. Funct. Anal.* **244**, 668–690 (2007)
239. Crouzeix, M.: Some constants related to numerical ranges. *SIAM J. Matrix Anal. Appl.* **37**, 420–442 (2016)
240. Crouzeix, M., Palencia, C.: The numerical range is a $(1 + \sqrt{2})$ -spectral set. *SIAM J. Matrix Anal. Appl.* **38**(2), 649–655 (2017)
241. Cuevas, R., Schaefer, C.E., Bhaya, A.: A proportional-derivative control strategy for varying the restart parameter in GMRES(m). *Anais do XXXIII CNMAC* **3**, 1000–1001 (2010)
242. Cullum, J.: Peaks, plateaus, numerical instabilities in a Galerkin minimal residual pair of methods for solving $Ax = b$. *Appl. Numer. Math.* **19**(3), 255–278 (1995)
243. Cullum, J.: Arnoldi versus nonsymmetric Lanczos algorithms for solving matrix eigenvalue problems. *BIT Numer. Math.* **36**(3), 470–493 (1996)
244. Cullum, J.: Iterative methods for solving $Ax = b$, GMRES/FOM versus QMR/BiCG. *Adv. Comput. Math.* **6**, 1–24 (1996)
245. Cullum, J., Greenbaum, A.: Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. *SIAM J. Matrix Anal. Appl.* **17**(2), 223–247 (1996)
246. Cullum, J., Zhang, T.: Two-sided Arnoldi and nonsymmetric Lanczos algorithms. *SIAM J. Matrix Anal. Appl.* **24**(2), 303–319 (2002)
247. Cunha, R.D., Hopkins, T.: A parallel implementation of the restarted GMRES iterative algorithm for nonsymmetric systems of linear equations. *Adv. Comput. Math.* **2**(3), 261–277 (1994)
248. Dahlquist, G., Björck, A.: Numerical Methods in Scientific Computing, vol. I. SIAM (2008)
249. Dahlquist, G., Eisenstat, S.C., Golub, G.H.: Bounds for the error of linear systems of equations using the theory of moments. *J. Math. Anal. Appl.* **37**, 151–166 (1972)
250. Dahlquist, G., Golub, G.H., Nash, S.G.: Bounds for the error in linear systems. In: Hettich, R. (ed.) *Proceedings of the workshop on Semi-infinite Programming*, pp. 154–172. Springer, Berlin (1978)
251. Dai, Y.H., Yuan, J.: Study on semi-conjugate direction methods for non-symmetric systems. *Int. J. Numer. Methods Eng.* **60**, 1383–1399 (2004)
252. Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comput.* **30**(136), 772–795 (1976)
253. Darnell, D., Morgan, R.B., Wilcox, W.: Deflated GMRES for systems with multiple shifts and multiple right-hand sides. *Linear Algebra Appl.* **429**(10), 2415–2434 (2008)
254. Datta, B.N.: Application of Hankel matrix to the root location problem. *IEEE Trans. Autom. Control* **21**(4), 610–612 (1976)
255. Davis, P., Rabinowitz, P.: A multiple purpose orthonormalizing code and its uses. *J. ACM* **1**, 183–191 (1954)
256. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM (2006)
257. Davis, T.A., Hu, Y.: The university of Florida Sparse Matrix Collection. *ACM Trans. Math. Soft. (TOMS)* **38**(1), 1–25 (2011)
258. Day, D.M.: Semi-duality in the two-sided Lanczos algorithm. Ph.D. thesis, University of California, Berkeley (1993)
259. Day, D.M.: An efficient implementation of the nonsymmetric Lanczos algorithm. *SIAM J. Matrix Anal. Appl.* **18**(3), 566–589 (1997)

260. de Sturler, E.: Nested Krylov methods based on GCR. *J. Comput. Appl. Math.* **67**, 15–41 (1996)
261. de Sturler, E.: Truncation strategies for optimal Krylov subspace methods. *SIAM J. Numer. Anal.* **36**(3), 864–889 (1999)
262. de Sturler, E., Fokkema, D.R.: Nested Krylov methods and preserving the orthogonality. In: Melson, D., Manteuffel, T., Mc Cormick, S.F. (eds.) Sixth Copper Mountain Conference on Multigrid Methods, pp. 111–126. NASA Conference Publication 3324 (1993)
263. de Sturler, E., van der Vorst, H.A.: Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. *Appl. Numer. Math.* **18**(4), 441–459 (1995)
264. De Terán, F., Dopico, F.M., Pérez, J.: New bounds for roots of polynomials based on Fiedler companion matrices. *Linear Algebra Appl.* **451**, 197–230 (2014)
265. Demmel, J., Grigori, L., Hoemmen, M., Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations. *SIAM J. Sci. Comput.* **34**(1), A206–A239 (2012)
266. Dennis, J.E., Turner, K.: Generalized conjugate directions. *Linear Algebra Appl.* **88**, 187–209 (1987)
267. Dennis, J.E., Walker, H.F.: Inaccuracy in quasi-Newton methods: Local improvement theorems. In: Mathematical Programming at Oberwolfach II, pp. 70–85. Springer, Berlin (1984)
268. Dieci, L., Gasparo, M.G., Papini, A.: Smoothness of Hessenberg and bidiagonal forms. *Mediterr. J. Math.* **5**(1), 21–31 (2008)
269. Dinkla, R.M.: GMRES(m) with deflation applied to nonsymmetric systems arising from fluid mechanics problems. Master's thesis, Delft University of Technology, Delft, The Netherlands (2009)
270. Dongarra, J., Tomov, S., Luszczek, P., Kurzak, J., Gates, M., Yamazaki, I., Anzt, H., Haidar, A., Abdelfattah, A.: With extreme computing the rules have changed. *Comput. Sci. Eng.* **19**(3), 52–62 (2017). IEEE
271. Dongarra, J.J., Duff, I.S., Sorensen, D.C., van der Vorst, H.A.: Solving Linear Systems on Vector and Shared Memory Computers. SIAM (1991)
272. Dongarra, J.J., Duff, I.S., Sorensen, D.C., van der Vorst, H.A.: Numerical Linear Algebra for High-Performance Computers. SIAM (1998)
273. Dostál, Z.: Conjugate gradient method with preconditioning by projector. *Int. J. Comput. Math.* **23**(3–4), 315–323 (1988)
274. Dostál, Z.: Projector preconditioning and domain decomposition methods. *Appl. Math. Comput.* **37**(2), 75–81 (1990)
275. Douglas, C.C., Lee, L., Yeung, M.C.: On solving ill conditioned linear systems. *Procedia Comput. Sci.* **80**, 941–950 (2016)
276. Draux, A.: Polynômes orthogonaux formels. Applications, vol. LNM 974. Springer (1983)
277. Draux, A.: Formal orthogonal polynomials revisited, applications. *Numer. Algorithms* **11**, 143–158 (1996)
278. van den Driessche, P., Wimmer, H.K.: Explicit polar decomposition of companion matrices. *Electron. J. Linear Algebra* **1**, 64–69 (1996)
279. Drkošová, J., Greenbaum, A., Rozložník, M., Strakoš, Z.: Numerical stability of GMRES. *BIT* **35**(3), 309–330 (1995)
280. Druskin, V., Knizhnerman, L.: Extended Krylov subspace: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.* **19**(3), 755–771 (1998)
281. Du, K., Duintjer Tebbens, J., Meurant, G.: Any admissible harmonic Ritz value set is possible for GMRES. *Electron. Trans. Numer. Anal.* **47**, 36–56 (2017)
282. Du, K., Huang, Y., Wang, Y.: On two generalized inverse eigenvalue problems for Hessenberg-upper triangular pencils and their application to the study of GMRES convergence. *Linear Algebra Appl.* **553**, 16–36 (2018)
283. Du, L., Sogabe, T., Yu, B., Yamamoto, Y., Zhang, S.L.: A block IDR(s) method for non-symmetric linear systems with multiple right-hand sides. *J. Comput. Appl. Math.* **235**(14), 4095–4106 (2011)

284. Du, L., Sogabe, T., Zhang, S.L.: Quasi-minimal residual smoothing technique for the IDR(s) method. *SIAM Lett.* **3**, 13–16 (2011)
285. Du, X., Szyld, D.B.: Inexact GMRES for singular linear systems. *BIT Numer. Math.* **48**, 511–531 (2008)
286. Duff, I.S., Erisman, A.M., Reid, J.K.: *Direct Methods for Sparse Matrices*, 2nd edn. Oxford University Press (2017)
287. Duintjer Tebbens, J., Meurant, G.: Any Ritz value behavior is possible for Arnoldi and for GMRES. *SIAM J. Matrix Anal. Appl.* **33**(3), 958–978 (2012)
288. Duintjer Tebbens, J., Meurant, G.: Prescribing the behavior of early terminating GMRES and Arnoldi iterations. *Numer. Algorithms* **65**(1), 69–90 (2014)
289. Duintjer Tebbens, J., Meurant, G.: On the convergence of Q-OR and Q-MR Krylov methods for solving linear systems. *BIT Numer. Math.* **56**(1), 77–97 (2016)
290. Duintjer Tebbens, J., Meurant, G.: On the residual norms, the Ritz values and the harmonic Ritz values that can be generated by restarted GMRES. *Algorithms* (published online December, Numer (2019))
291. Duintjer Tebbens, J., Meurant, G., Sadok, H., Strakoš, Z.: On investigating GMRES convergence using unitary matrices. *Linear Algebra Appl.* **450**, 83–107 (2014)
292. Duminil, S.: A parallel implementation of the CMRH method for dense linear systems. *Numer. Algorithms* **63**, 127–142 (2013)
293. Duminil, S., Heyouni, M., Marion, P., Sadok, H.: Algorithms for the CMRH method for dense linear systems. *Numer. Algorithms* **71**(2), 383–394 (2016)
294. Edelman, A., Murakami, H.: Polynomial roots from companion matrix eigenvalues. *Math. Comput.* **64**(210), 763–776 (1995)
295. Eiermann, M.: Fields of values and iterative methods. *Linear Algebra Appl.* **180**, 167–197 (1993)
296. Eiermann, M., Ernst, O.G.: Geometric aspects of the theory of Krylov subspace methods. *Acta Numer.* **10**, 251–312 (2001)
297. Eiermann, M., Ernst, O.G., Schneider, O.: Analysis of acceleration strategies for restarted minimal residual methods. *J. Comput. Appl. Math.* **123**(1), 261–292 (2000)
298. Eijkhout, V.: Computational variants of the CGS and BiCGstab methods. Tech. Rep. UT-CS-94-241, Department of Computer Science, University of Tennessee (1994)
299. Eijkhout, V.: Qualitative properties of the conjugate gradient and Lanczos methods in a matrix framework. Tech. Rep. LAPACK Working Note 51, Department of Computer Science, University of Tennessee (1995)
300. Eisenstat, S.C.: Equivalence of Krylov subspace methods for skew-symmetric linear systems. arXiv preprint, [arXiv:1512.00311](https://arxiv.org/abs/1512.00311) (2015)
301. Eisenstat, S.C., Elman, H.C., Schultz, M.H.: Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* **20**, 345–367 (1983)
302. Eisenstat, S.C., Ipsen, I.C.F.: Relative perturbation results for eigenvalues and eigenvectors of diagonalisable matrices. *BIT Numer. Math.* **38**(3), 502–509 (1998)
303. Eisenstat, S.C., Ipsen, I.C.F.: Three absolute perturbation bounds for matrix eigenvalues imply relative bounds. *SIAM J. Matrix Anal. Appl.* **20**(1), 149–158 (1998)
304. Eisinberg, A., Fedele, G.: A property of the elementary symmetric functions. *Calcolo* **42**(1), 31–36 (2005)
305. Eisinberg, A., Fedele, G.: On the inversion of the Vandermonde matrix. *Appl. Math. Comput.* **174**(2), 1384–1397 (2006)
306. Elbouyahyaoui, L., Messaoudi, A., Sadok, H.: Algebraic properties of the block GMRES and block Arnoldi methods. *Electron. Trans. Numer. Anal.* **33**, 207–220 (2009)
307. Eldén, L., Simoncini, V.: Solving ill-posed linear systems with GMRES and a singular preconditioner. *SIAM J. Matrix Anal. Appl.* **33**(4), 1369–1394 (2012)
308. Elman, H.C.: Iterative methods for large sparse nonsymmetric systems of linear equations. Ph.D. thesis, Computer Science Department, Yale University, USA (1982). Research Report 229

309. Elman, H.C., Ramage, A.: Fourier analysis of multigrid for a model two-dimensional convection-diffusion equation. *BIT Numer. Math.* **46**(2), 283–306 (2006)
310. Elman, H.C., Ramage, A., Silvester, D.J.: Algorithm 866: IPISS, a Matlab toolbox for modelling incompressible flow. *ACM Trans. Math. Soft. (TOMS)* **33**(2) (2007). (article 14)
311. Elman, H.C., Saad, Y., Saylor, P.E.: A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations. *SIAM J. Sci. Statist. Comput.* **7**(3), 840–855 (1986)
312. Elman, H.C., Silvester, D.J., Wathen, A.J.: *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, 2nd edn. Oxford University Press (2014)
313. Elman, H.C., Streit, R.L.: Polynomial iteration for nonsymmetric indefinite linear systems. In: *Numerical Analysis*, pp. 103–117. Springer, Berlin, Heidelberg (1986)
314. Elsner, L.: An optimal bound for the spectral variation of two matrices. *Linear Algebra Appl.* **71**, 77–80 (1985)
315. Elsner, L., Ikramov, K.D.: On a condensed form for normal matrices under finite sequences of elementary unitary similarities. *Linear Algebra Appl.* **254**(1–3), 79–98 (1997)
316. Elsner, L., Ikramov, K.D.: Normal matrices: an update. *Linear Algebra Appl.* **285**(1–3), 291–303 (1998)
317. Embree, M.: How descriptive are GMRES convergence bounds? *Tech. Rep. 99/08*, Mathematical Institute, University of Oxford, UK (1999)
318. Embree, M.: The tortoise and the hare restart GMRES. *SIAM Rev.* **45**(2), 259–266 (2003)
319. Embree, M., Morgan, R.B., Nguyen, H.V.: Weighted inner products for GMRES and GMRES-DR. *SIAM J. Sci. Comput.* **39**(5), S610–S632 (2017)
320. Embree, M., Sifuentes, J.A., Soodhalter, K.M., Szyld, D.B., Xue, F.: Short-term recurrence Krylov subspace methods for nearly Hermitian matrices. *SIAM J. Matrix Anal. Appl.* **33**(2), 480–500 (2012)
321. Erhel, J.: A parallel GMRES version for general sparse matrices. *Electron. Trans. Numer. Anal.* **3**, 160–176 (1995)
322. Erhel, J., Burrage, K., Pohl, B.: Restarted GMRES preconditioned by deflation. *J. Comput. Appl. Math.* **69**(2), 303–318 (1996)
323. Erhel, J., Guyomarc'h, F.: An augmented conjugate gradient algorithm for solving consecutive symmetric positive definite linear systems. *SIAM J. Matrix Anal. Appl.* **21**(4), 1279–1299 (2000)
324. Ericsson, T.: Computing functions of matrices using Krylov subspace methods. Chalmers University, Sweden, Tech. Rep. TR (1990)
325. Ericsson, T.: On the eigenvalues and eigenvectors of Hessenberg matrices. Chalmers University, Sweden, Tech. Rep. TR (1990)
326. Erlangga, Y.A., Nabben, R.: Deflation and balancing preconditioners for Krylov subspace methods applied to nonsymmetric matrices. *SIAM J. Matrix Anal. Appl.* **30**(2), 684–699 (2008)
327. Erlangga, Y.A., Nabben, R.: Algebraic multilevel Krylov methods. *SIAM J. Sci. Comput.* **31**(5), 3417–3437 (2009)
328. Ernst, O.G.: Minimal and orthogonal residual methods and their generalizations for solving linear operator equations. Habilitation thesis, TU Freiberg, Germany (2000)
329. Ernst, O.G.: Residual-minimizing Krylov subspace methods for stabilized discretizations of convection-diffusion equations. *SIAM J. Matrix Anal. Appl.* **21**(4), 1079–1101 (2000)
330. Estrada, E., Hatano, N., Benzi, M.: The physics of communicability in complex networks. *Phys. Rep.* **514**, 89–119 (2012)
331. Estrada, E., Higham, D.J.: Network properties revealed through matrix functions. *SIAM Rev.* **52**, 696–714 (2010)
332. Faber, V., Greenbaum, A., Marshall, D.E.: The polynomial numerical hulls of Jordan blocks and related matrices. *Linear Algebra Appl.* **374**, 231–246 (2003)
333. Faber, V., Joubert, W., Knill, E., Manteuffel, T.: Minimal residual method stronger than polynomial preconditioning. *SIAM J. Matrix Anal. Appl.* **17**(4), 707–729 (1996)

334. Faber, V., Liesen, J., Tichý, P.: The Faber-Manteuffel theorem for linear operators. *SIAM J. Numer. Anal.* **46**, 1323–1337 (2008)
335. Faber, V., Liesen, J., Tichý, P.: On orthogonal reduction to Hessenberg form with small bandwidth. *Numer. Algorithms* **51**(2), 133–142 (2009)
336. Faber, V., Liesen, J., Tichý, P.: On Chebyshev polynomials of matrices. *SIAM J. Matrix Anal. Appl.* **31**(4), 2205–2221 (2010)
337. Faber, V., Liesen, J., Tichý, P.: Properties of worst-case GMRES. *SIAM J. Matrix Anal. Appl.* **34**(4), 1500–1519 (2013)
338. Faber, V., Manteuffel, T.: Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.* **21**(2), 352–362 (1984)
339. Faber, V., Manteuffel, T.: Orthogonal error methods. *SIAM J. Numer. Anal.* **24**(1), 170–187 (1987)
340. Faddeev, D.K.: Properties of the inverse of a Hessenberg matrix. *J. Sov. Math.* **24**(1), 118–120 (1984). In: Ilin, V.P., Kublanovskaya, V.N. (eds.) *Numerical Methods and Computational Issues*, vol. 5 (1981) (in Russian)
341. Falgout, R.D., Jones, J.E., Yang, U.M.: The design and implementation of Hypre, a library of parallel high performance preconditioners. In: Bruaset, A.M., Tveito, A. (eds.) *Numerical Solution of Partial Differential Equations on Parallel Computers*, vol. 51, pp. 267–294. Springer (2006)
342. Farooq, M.: New Lanczos-type algorithms and their implementation. Ph.D. thesis, University of Essex, UK (2011)
343. Farrell, P.E.: The number of distinct eigenvalues of a matrix after perturbation. *SIAM J. Matrix Anal. Appl.* **37**(2), 572–576 (2016)
344. Fassbender, H.: Inverse unitary eigenproblems and related orthogonal functions. *Numer. Math.* **77**, 323–345 (1997)
345. Fassino, C.: On updating the least singular value: a lower bound. *Calcolo* **40**(4), 213–229 (2003)
346. Fenu, C., Martin, D., Reichel, L., Rodriguez, G.: Network analysis via partial spectral factorization and Gauss quadrature. *SIAM J. Sci. Comput.* **35**(4), A2046–A2068 (2013)
347. Feuerriegel, S., Bücker, H.M.: The non-symmetric s-step Lanczos algorithm: derivation of efficient recurrences and synchronization-reducing variants of BiCG and QMR. *Int. J. Appl. Math. Comput. Sci.* **25**(4), 769–785 (2015)
348. Fiedler, M.: Special Matrices and Their Applications in Numerical Mathematics, 1st edn. Martinus Nijhoff Publishers, Dordrecht, The Netherlands (1986). Reprinted by Dover (2008)
349. Fischer, B.: Polynomial Based Iteration Methods for Symmetric Linear Systems. Wiley-Tubner, Leipzig (1996)
350. Fischer, B., Golub, G.H.: On the error computation for polynomial based iteration methods. In: Greenbaum, A., Luskin, M. (eds.) *Recent Advances in Iterative Methods*. Springer (1993)
351. Fischer, B., Ramage, A., Silvester, D.J., Wathen, A.J.: Minimum residual methods for augmented systems. *BIT Numer. Math.* **38**(3), 527–543 (1998)
352. Fischer, B., Ramage, A., Silvester, D.J., Wathen, A.J.: On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems. *Comput. Methods Appl. Mech. Eng.* **179**, 179–195 (1999)
353. Fischer, B., Reichel, L.: Newton interpolation in Fejér and Chebyshev points. *Math. Comput.* **53**(187), 265–278 (1989)
354. Fischer, B., Reichel, L.: A stable Richardson iteration method for complex linear systems. *Numer. Math.* **54**(2), 225–242 (1989)
355. Fletcher, R.: Conjugate gradient methods for indefinite systems. In: Watson, G.A. (ed.) *Numerical Analysis, Proceedings of the 6th Biennial Dundee Conference*, University of Dundee (1975). Lecture Notes in Mathematics, vol. 506, pp. 73–89. Springer, Berlin (1976)
356. Fokkema, D.R.: Enhanced implementation of BiCGstab(ℓ) for solving linear systems of equations. University of Utrecht, Mathematics Institute, Tech. rep. (1996)
357. Fokkema, D.R.: Subspace methods for linear, nonlinear, and eigen problems. Ph.D. thesis, University of Utrecht, The Netherlands (1996)

358. Fokkema, D.R., Sleijpen, G.L.G., van der Vorst, H.: Generalized conjugate gradient squared. *J. Comput. Appl. Math.* **71**(1), 125–146 (1996)
359. Fong, D.C.L., Saunders, M.: LSMR: an iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput.* **33**(5), 2950–2971 (2011)
360. Forsythe, G.E.: Tentative classification of methods and bibliography on solving systems of linear equations. Symposium on Simultaneous Linear Equations and the Determination of Eigenvalues, National Bureau of Standards: Applied Mathematics **29**, 1–28 (1953)
361. Forsythe, G.E., Hestenes, M.R., Rosser, J.: Iterative methods for solving linear equations. *Bull. Am. Math. Soc.* **57**, 480 (1951)
362. Fox, L., Huskey, H.D., Wilkinson, J.H.: Notes on the solution of algebraic linear simultaneous equations. *Quart. J. Mech. Appl. Math.* **1**(1), 149–173 (1948)
363. Frayssé, V., Giraud, L., Gratton, S.: A set of flexible-GMRES routines for real and complex arithmetics. *Tech. Rep. TR/PA/98/20*, CERFACS, Toulouse, France (1998)
364. Frayssé, V., Giraud, L., Gratton, S., Langou, J.: Algorithm 842: a set of GMRES routines for real and complex arithmetics on high performance computers. *ACM Trans. Math. Soft. (TOMS)* **31**(2), 228–238 (2005)
365. Frayssé, V., Giraud, L., Kharraz-Aroussi, H.: On the influence of the orthogonalization scheme on the parallel performance of GMRES. In: *Euro-Par'98 Parallel Processing*. Springer, Berlin, Heidelberg (1998)
366. Freund, R., Hochbruck, M.: Gauss quadratures associated with the Arnoldi process and the Lanczos algorithm. In: Moonen, M.S., Golub, G.H., De Moor, B. (eds.) *Linear Algebra for Large Scale and Real-Time Applications*, pp. 377–380. Kluwer (1993)
367. Freund, R.W.: Quasi-kernel polynomials and their use in non-Hermitian matrix iterations. *J. Comput. Appl. Math.* **43**(1–2), 135–158 (1992)
368. Freund, R.W.: Solution of shifted linear systems by quasi-minimal residual iterations. In: Reichel, L., Ruttan, A., Varga, R.S. (eds.) *Numerical Linear Algebra*, pp. 101–121. De Gruyter (1993)
369. Freund, R.W.: A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.* **14**(2), 470–482 (1993)
370. Freund, R.W., Golub, G.H., Nachtigal, N.M.: Iterative solution of linear systems. *Acta Numer.* **1**, 57–100 (1992)
371. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, part I. *Tech. Rep. RIACS 90.45*, NASA Ames Research Center, USA (1990)
372. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *Tech. Rep. RIACS 91.09*, NASA Ames Research Center, USA (1991)
373. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.* **14**, 137–158 (1993)
374. Freund, R.W., Hochbruck, M.: A biconjugate gradient type algorithm on massively parallel architectures. In: Vichnevetsky, R., Miller, J.J.H. (eds.) *IMACS'91*, pp. 720–721. Criterion Press, Dublin (1991)
375. Freund, R.W., Hochbruck, M.: A biconjugate gradient-type algorithm for the iterative solution of non-Hermitian linear systems on massively parallel architectures. In: Brezinski, C., Kulisch, U. (eds.) *Computational and Applied Mathematics, I: Algorithms and Theory*, pp. 169–178. North Holland, Amsterdam (1992)
376. Freund, R.W., Hochbruck, M.: On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling. *Numer. Linear Algebra Appl.* **1**(4), 403–420 (1994)
377. Freund, R.W., Malhotra, M.: A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides. *Linear Algebra Appl.* **254**(1–3), 119–157 (1997)
378. Freund, R.W., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, part II. *Tech. Rep. RIACS 90.46*, NASA Ames Research Center, USA (1990)

379. Freund, R.W., Nachtigal, N.M.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.* **60**(3), 315–339 (1991)
380. Freund, R.W., Nachtigal, N.M.: Implementation details of the coupled QMR algorithm. *Tech. Rep. RIACS 92.19*, NASA Ames Research Center, USA (1992)
381. Freund, R.W., Nachtigal, N.M.: An implementation of the QMR method based on coupled two-term recurrences. *Tech. Rep. RIACS 92.15*, NASA Ames Research Center, USA (1992)
382. Freund, R.W., Nachtigal, N.M.: An implementation of the QMR method based on coupled two-term recurrences. *SIAM J. Sci. Comput.* **15**(2), 313–337 (1994)
383. Freund, R.W., Nachtigal, N.M.: Software for simplified Lanczos and QMR algorithms. *Appl. Numer. Math.* **19**(3), 319–341 (1995)
384. Freund, R.W., Szeto, T.: A quasi-minimal residual squared algorithm for non-Hermitian linear systems. *Tech. Rep. 91.26*, RIACS, NASA Ames Research Center (1991)
385. Freund, R.W., Szeto, T.: A quasi-minimal residual squared algorithm for non-Hermitian linear systems. *Tech. Rep. CAM Report 92-19*, Department of Mathematics, UCLA (1992)
386. Freund, R.W., Szeto, T.: A transpose-free quasi-minimal residual squared algorithm for non-Hermitian linear systems. In: Vichnevetsky, R., Knight, D., Richter, G. (eds.) *Advances in Computer Methods for Partial Differential Equations - VII*, IMACS, pp. 258–264. IMACS (1992)
387. Frommer, A.: BiCGStab(ℓ) for families of shifted linear systems. *Computing* **70**(2), 87–109 (2003)
388. Frommer, A., Glässner, U.: Restarted GMRES for shifted linear systems. *SIAM J. Sci. Comput.* **19**(1), 15–26 (1998)
389. Fujino, S.: GPBiCG(m, ℓ): a hybrid of BiCGSTAB and GPBiCG methods with efficiency and robustness. *Appl. Numer. Math.* **41**(1), 107–117 (2002)
390. Fujino, S., Fujiwara, M., Yoshida, M.: BiCGSafe method based on minimization of associate residual. *Transactions of JSCES* (2005)
391. Fujino, S., Iwasato, K.: A single-synchronized linear solver for the solution of problems of computational mechanics on parallel computers. In: 11th World Congress on Computational Mechanics (WCCM XI) (2014)
392. Fujino, S., Sekimoto, T.: Performance evaluation of GPBiCGSafe method without reverse-ordered recurrence for realistic problems. In: Proceedings of the International Multiconference of Engineers and Computer Scientists 2012, vol II, IMECS (2012)
393. Fukaya, T., Nakatsukasa, Y., Yanagisawa, Y., Yamamoto, Y.: CholeskyQR2: a simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system. In: Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, pp. 31–38. IEEE Press (2014)
394. Gallivan, K.A., Sameh, A.H., Zlatev, Z.: Comparison of ten methods for the solution of large and sparse linear algebraic systems. In: International Conference on Numerical Methods and Applications NMA2002, Lecture Notes in Computer Science, vol. 2542. Springer, Berlin, Heidelberg (2003)
395. Gallopoulos, E., Philippe, B., Sameh, A.: Parallelism in Matrix Computations. Springer, Dordrecht (2016)
396. Gantmacher, F.R.: The Theory of Matrices, vol. 1. AMS Chelsea Publishing, Providence, RI (1959)
397. Gao, H., Dai, H.: A global property of restarted FOM algorithm. *J. Inf. Comput. Sci.* **1**(1), 11–20 (2006)
398. Gau, H.L., Wang, K.Z., Wu, P.Y.: Crawford numbers of companion matrices. *Oper. Matrices* **10**(4), 863–879 (2016)
399. Gau, H.L., Wu, P.Y.: Companion matrices: reducibility, numerical ranges and similarity to contractions. *Linear Algebra Appl.* **383**, 127–142 (2004)
400. Gau, H.L., Wu, P.Y.: Numerical ranges of companion matrices. *Linear Algebra Appl.* **421**(2), 202–218 (2007)
401. Gaul, A.: Recycling Krylov subspace methods for sequences of linear systems. Ph.D. thesis, Technical University Berlin (2014)

402. Gaul, A., Gutknecht, M.H., Liesen, J., Nabben, R.: A framework for deflated and augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **34**(2), 495–518 (2013)
403. Gauss, C.F.: *Disquisitio de elementis ellipticis palladis*. In: Carl Friedrich Gauss Werke, vol. VI, pp. 1–24. Königlichen Gesellschaft der Wissenschaften zu Göttingen (1809)
404. Gautschi, W.: On inverses of Vandermonde and confluent Vandermonde matrices. *Numer. Math.* **4**, 117–123 (1962)
405. Gautschi, W.: On the condition of algebraic equations. *Numer. Math.* **21**(5), 405–424 (1973)
406. Gautschi, W.: Condition of polynomials in power form. *Math. Comput.* **33**, 343–352 (1979)
407. Gautschi, W.: Questions of numerical condition related to polynomials. In: Golub, G.H. (ed.) *Studies in Numerical Analysis, Studies in Mathematics*, vol. 24, pp. 140–177. MAA (1985)
408. Gautschi, W.: How (un)stable are Vandermonde systems? In: Wong, R. (ed.) *Asymptotic and Computational Analysis, Lectures Notes in Pure and Applied Mathematics*, vol. 124, pp. 193–210. Marcel Dekker (1990)
409. Gazzola, S., Hansen, P.C., Nagy, J.G.: IR tools: a MATLAB package of iterative regularization methods and large-scale test problems. *Numer. Algorithms* **81**(3), 773–811 (2019)
410. Gellai, B.: Cornelius Lanczos, a biographical essay. In: [151], pp. xxi–xlviii (1994)
411. Gellai, B.: *The Intrinsic Nature of Things: The Life and Science of Cornelius Lanczos*. American Mathematical Society (2010)
412. Gentleman, W.: Least squares computations by Givens transformations without square roots. *IMA J. Appl. Math.* **12**, 329–336 (1973)
413. George, A., Ikramov, K., Krivoshapova, A.N., Tang, W.P.: A finite procedure for the tridiagonalization of a general matrix. *SIAM J. Matrix Anal. Appl.* **16**(2), 377–387 (1995)
414. Ghai, A., Lu, C., Jiao, X.: A comparison of preconditioned Krylov subspace methods for nonsymmetric linear systems. arXiv preprint, [arXiv:1607.00351](https://arxiv.org/abs/1607.00351) (2016)
415. Ghysels, P., Ashby, T.J., Meerbergen, K., Vanroose, W.: Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM J. Sci. Comput.* **35**(1), C48–C71 (2013)
416. Gill, P., Golub, G.H., Murray, W., Saunders, M.: Methods for modifying matrix factorizations. *Math. Comput.* **28**(126), 505–535 (1974)
417. Giraud, L., Gratton, S., Langou, J.: Convergence in backward error of relaxed GMRES. *SIAM J. Sci. Comput.* **29**(2), 710–728 (2007)
418. Giraud, L., Gratton, S., Pinel, X., Vasseur, X.: Flexible GMRES with deflated restarting. *SIAM J. Sci. Comput.* **32**(4), 1858–1878 (2010)
419. Giraud, L., Langou, J.: Another proof for modified Gram-Schmidt with reorthogonalization. Tech. Rep. Working Notes WN/PA/02/53, CERFACS, Toulouse, France (2002)
420. Giraud, L., Langou, J.: When modified Gram-Schmidt generates a well-conditioned set of vectors. *IMA J. Numer. Anal.* **22**(4), 521–528 (2002)
421. Giraud, L., Langou, J.: Robust selective Gram-Schmidt reorthogonalization. *SIAM J. Sci. Comput.* **25**(2), 417–441 (2003)
422. Giraud, L., Langou, J., Gratton, S.: A note on relaxed and flexible GMRES. Tech. Rep. TR/PA/04/41, CERFACS, Toulouse, France (2004)
423. Giraud, L., Langou, J., Rozložník, M.: On the round-off error analysis of the Gram-Schmidt algorithm with reorthogonalization. Tech. Rep. TR/PA/02/33, CERFACS, Toulouse, France (2002)
424. Giraud, L., Langou, J., Rozložník, M.: On the loss of orthogonality in the Gram-Schmidt orthogonalization process. *Comput. Math. Appl.* **50**, 1069–1075 (2005)
425. Giraud, L., Langou, J., Rozložník, M., van den Eshof, J.: Rounding error analysis of the classical Gram-Schmidt orthogonalization process. *Numer. Math.* **101**, 87–100 (2005)
426. Givens, W.: Computation of plain unitary rotations transforming a general matrix to triangular form. *J. Soc. Ind. Appl. Math.* **6**(1), 26–50 (1958)
427. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* **23**(1), 5–48 (1991)
428. Golub, G.H.: Some modified matrix eigenvalue problems. *SIAM Rev.* **15**(2), 318–334 (1973)

429. Golub, G.H., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.* **2**, 205–224 (1965)
430. Golub, G.H., Meurant, G.: Matrices, moments and quadrature. In: Griffiths, D.F., Watson, G.A. (eds.) *Numerical Analysis 1993*, Pitman Research Notes in Mathematics, vol. 303, pp. 105–156. Longman Sci. Tech. (1994). Reprinted in [202]
431. Golub, G.H., Meurant, G.: Matrices, moments and quadrature II or how to compute the norm of the error in iterative methods. *BIT* **37**(3), 687–705 (1997)
432. Golub, G.H., Meurant, G.: *Matrices. Princeton University Press, Moments and Quadrature with Applications* (2010)
433. Golub, G.H., O'Leary, D.P.: Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. *SIAM Rev.* **31**, 50–102 (1989)
434. Golub, G.H., Stoll, M., Wathen, A.: Approximation of the scattering amplitude and linear systems. *Electron. Trans. Numer. Anal.* **31**, 178–203 (2008)
435. Golub, G.H., Strakoš, Z.: Estimates in quadratic formulas. *Numer. Algorithms* **8**(2), 241–268 (1994)
436. Golub, G.H., Underwood, R.: The block Lanczos method for computing eigenvalues. In: Rice, J. (ed.) *Mathematical Software III*, Proceedings of a Symposium conducted by the Mathematics Research Center, the University of Wisconsin-Madison, March 28–30, 1977, pp. 361–377. Academic Press (1977)
437. Golub, G.H., van der Vorst, H.A.: Closer to the solutions: iterative linear solvers. In: Duff, S., Watson, G.A. (eds.) *The State of the Art in Numerical Analysis*, pp. 63–92. Oxford University Press (2001)
438. Golub, G.H., Van Loan, C.: *Matrix Computations*, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
439. Gomes-Ruggiero, M., Lopes, V.R., Toledo-Benavides, J.: A safeguard approach to detect stagnation of GMRES(m) with applications in Newton-Krylov methods. *Comput. Appl. Math.* **27**(2), 175–199 (2008)
440. Goossens, S., Roose, D.: Ritz and harmonic Ritz values and the convergence of FOM and GMRES. *Numer. Linear Algebra Appl.* **6**(4), 281–293 (1999)
441. Gragg, W.B.: Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mt. J. Math.* **4**(2), 213–225 (1974)
442. Gragg, W.B.: The QR algorithm for unitary Hessenberg matrices. *J. Comput. Appl. Math.* **16**, 1–8 (1986)
443. Gragg, W.B., Lindquist, A.: On the partial realization problem. *Linear Algebra Appl.* **50**, 277–319 (1983)
444. Gragg, W.B., Reichel, L.: On the application of orthogonal polynomials to the iterative solution of linear systems of equations with indefinite or non-Hermitian matrices. *Linear Algebra Appl.* **88**, 349–371 (1987)
445. Grammont, L.: Characteristic polynomials and pseudo spectra. In: SIAM Conference on Linear Algebra. <http://www.siam.org/meetings/la03/proceedings> (2003)
446. Graves-Morris, P.R.: A "Look-around Lanczos" algorithm for solving a system of linear equations. *Numer. Algorithms* **3–4**, 247–274 (1997)
447. Graves-Morris, P.R.: Reliability of Lanczos-type product methods from perturbation theory. *Reliab. Comput.* **6**(4), 411–428 (2000)
448. Graves-Morris, P.R.: The breakdowns of BiCGStab. *Numer. Algorithms* **29**(1), 97–105 (2002)
449. Graves-Morris, P.R.: VPAStab: stabilised vector-Padé approximation with application to linear systems. *Numer. Algorithms* **33**(1–4), 293–304 (2003)
450. Graves-Morris, P.R., Salam, A.: Avoiding breakdown in van der Vorst's method. *Numer. Algorithms* **21**, 205–223 (1999)
451. Greenbaum, A.: Convergence properties of the conjugate gradient algorithm in exact and finite precision arithmetic. Ph.D. thesis, University of California, Berkeley (1981)
452. Greenbaum, A.: Behavior of slightly perturbed Lanczos and conjugate gradient recurrences. *Linear Algebra Appl.* **113**, 7–63 (1989)

453. Greenbaum, A.: The Lanczos and conjugate gradient algorithms in finite precision arithmetic. In: [151], pp. 49–60 (1994)
454. Greenbaum, A.: Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Matrix Anal. Appl.* **18**(3), 535–551 (1997)
455. Greenbaum, A.: Iterative Methods for Solving Linear Systems. SIAM (1997)
456. Greenbaum, A.: On the role of the left starting vector in the two-sided Lanczos algorithm and nonsymmetric linear system solvers. In: Griffiths, D., Higham, D., Watson, G. (eds.) Proceedings of the Dundee Meeting in Numerical Analysis. Pitman Research Notes in Mathematics Series 380, Longman (1997)
457. Greenbaum, A.: Generalizations of the field of values useful in the study of polynomial functions of a matrix. *Linear Algebra Appl.* **347**, 233–249 (2002)
458. Greenbaum, A.: Some theoretical results derived from polynomial numerical hulls of Jordan blocks. *Electron. Trans. Numer. Anal.* **18**, 81–90 (2004)
459. Greenbaum, A.: Upper and lower bounds on norms of functions of matrices. *Linear Algebra Appl.* **430**, 52–65 (2009)
460. Greenbaum, A., Pták, V., Strakoš, Z.: Any convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.* **17**(3), 465–470 (1996)
461. Greenbaum, A., Rozložník, M., Strakoš, Z.: Numerical behaviour of the modified Gram-Schmidt GMRES implementation. *BIT* **37**(3), 706–719 (1997)
462. Greenbaum, A., Strakoš, Z.: Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.* **13**(1), 121–137 (1992)
463. Greenbaum, A., Strakoš, Z.: Matrices that generate the same Krylov residual spaces. In: Golub, G.H., Greenbaum, A., Luskin, M. (eds.) Recent Advances in Iterative Methods, pp. 95–118. Springer (1994)
464. Greenbaum, A., Trefethen, L.N.: GMRES/CR and Arnoldi/Lanczos as matrix approximation problems. *SIAM J. Sci. Comput.* **15**(2), 359–368 (1994)
465. Greif, C., Paige, C.C., Titley-Peloquin, D., Varah, J.M.: Numerical equivalences among Krylov subspace algorithms for skew-symmetric matrices. *SIAM J. Matrix Anal. Appl.* **37**(3), 1071–1087 (2016)
466. Greif, C., Rees, T., Szyld, D.B.: GMRES with multiple preconditioners. *SeMA J.* **74**(2), 213–231 (2017)
467. Grigori, L., Moufawad, S., Nataf, F.: Enlarged Krylov subspace conjugate gradient methods for reducing communication. *SIAM J. Matrix Anal. Appl.* **37**(2), 744–773 (2016)
468. Gu, G.D.: A seed method for solving nonsymmetric linear systems with multiple right-hand sides. *Int. J. Comput. Math.* **79**(3), 307–326 (2002)
469. Gu, T.X., Zuo, X.Y., Liu, X.P., Li, P.L.: An improved parallel hybrid bi-conjugate gradient method suitable for distributed parallel computing. *J. Comput. Appl. Math.* **226**, 55–65 (2009)
470. Gu, X.M., Huang, T.Z., Carpentieri, B.: BiCGCR2: a new extension of conjugate residual method for solving non-Hermitian linear systems. *J. Comput. Appl. Math.* **305**, 115–128 (2016)
471. Gu, X.M., Huang, T.Z., Carpentieri, B., Imakura, A., Zhang, K., Du, L.: Variants of the CMRH method for solving multi-shifted non-Hermitian linear systems. arXiv preprint, [arXiv:1611.00288](https://arxiv.org/abs/1611.00288) (2016)
472. Gu, X.M., Huang, T.Z., Carpentieri, B., Li, L., Wen, C.: A hybridized iterative algorithm of the BiCORSTAB and GPBiCOR methods for solving non-Hermitian linear systems. *Comput. Math. Appl.* **70**(12), 3019–3031 (2015)
473. Gu, X.M., Huang, T.Z., Yin, G., Carpentieri, B., Wen, C., Du, L.: Restarted Hessenberg method for solving shifted nonsymmetric linear systems. *J. Comput. Appl. Math.* **331**, 166–177 (2018)
474. Guennouni, A.E.: A unified approach to some strategies for the treatment of breakdown in Lanczos-type algorithms. *Appl. Math.* **26**(4), 477–488 (1999)
475. Guennouni, A.E., Jbilou, K., Sadok, H.: A block version of BiCGStab for linear systems with multiple right-hand sides. *Electron. Trans. Numer. Anal.* **16**, 129–142 (2003)

476. Guennouni, A.E., Jbilou, K., Sadok, H.: The block-Lanczos method for linear systems with multiple right-hand sides. *Appl. Numer. Math.* **51**, 243–256 (2004)
477. Gutknecht, M.H.: The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm. In: Proceedings of the Copper Mountain Conference on Iterative Methods (1990)
478. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. Part I. *SIAM J. Matrix Anal. Appl.* **13**(2), 594–639 (1992)
479. Gutknecht, M.H.: Changing the norm in conjugate gradient type algorithms. *SIAM J. Numer. Anal.* **30**(1), 40–56 (1993)
480. Gutknecht, M.H.: Variants of BICGSTAB for matrices with complex spectrum. *SIAM J. Sci. Comput.* **14**(5), 1020–1033 (1993)
481. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. Part II. *SIAM J. Matrix Anal. Appl.* **15**(1), 15–58 (1994)
482. Gutknecht, M.H.: Lanczos-type solvers for nonsymmetric linear systems of equations. *Acta Numer.* **6**, 271–397 (1997)
483. Gutknecht, M.H.: A general framework for recursions for Krylov space solvers. Tech. Rep. No. 2005–09, ETH, Zürich (2005)
484. Gutknecht, M.H.: IDR explained. *Electron. Trans. Numer. Anal.* **36**(3), 126–148 (2010)
485. Gutknecht, M.H.: Spectral deflation in Krylov solvers: a theory of coordinate space based methods. *Electron. Trans. Numer. Anal.* **39**, 156–185 (2012)
486. Gutknecht, M.H., Ressel, K.J.: QMR smoothing for Lanczos-type product methods based on three-term recurrences. *SIAM J. Sci. Comput.* **19**(1), 55–73 (1998)
487. Gutknecht, M.H., Ressel, K.J.: Look-ahead procedures for Lanczos-type product methods based on three-term Lanczos recurrences. *SIAM J. Matrix Anal. Appl.* **21**(4), 1051–1078 (2000)
488. Gutknecht, M.H., Rozložník, M.: By how much can residual minimization accelerate the convergence of orthogonal residual methods? *Numer. Algorithms* **27**, 189–213 (2001)
489. Gutknecht, M.H., Rozložník, M.: Residual smoothing techniques: do they improve the limiting accuracy of iterative solvers? *BIT Numer. Math.* **41**(1), 86–114 (2001)
490. Gutknecht, M.H., Rozložník, M.: A framework for generalized conjugate gradient methods – with special emphasis on contributions by Rüdiger Weiss. *Appl. Numer. Math.* **41**, 7–22 (2002)
491. Gutknecht, M.H., Schmelzer, T.: Updating the QR decomposition of block tridiagonal and block Hessenberg matrices. *Appl. Numer. Math.* **58**(6), 871–883 (2008)
492. Gutknecht, M.H., Schmelzer, T.: The block grade of a block Krylov space. *Linear Algebra Appl.* **430**(1), 174–185 (2009)
493. Gutknecht, M.H., Strakoš, Z.: Accuracy of two three-term and three two-term recurrences for Krylov space solvers. *SIAM J. Matrix Anal. Appl.* **22**(1), 213–229 (2000)
494. Gutknecht, M.H., Zemke, J.P.M.: Eigenvalue computations based on IDR. *SIAM J. Matrix Anal. Appl.* **34**(2), 283–311 (2013)
495. Güttel, S.: Rational Krylov methods for operator functions. Ph.D. thesis, Technische Universität Bergakademie Freiberg, Freiberg, Germany (2010)
496. Güttel, S., Pestana, J.: Some observations on weighted GMRES. *Numer. Algorithms* **67**(4), 733–752 (2014)
497. Habu, M., Nodera, T.: GMRES(m) algorithm with changing the restart cycle adaptively. In: Proceedings of Algoritmy 2000 Conference on Scientific Computing, pp. 354–263 (2000)
498. Hackbusch, W.: Iterative Solution of Large Sparse Systems of Equations. Springer (1993)
499. Hageman, L.A., Young, D.M.: Applied Iterative Methods. Academic Press (1981)
500. Hager, W.W.: Updating the inverse of a matrix. *SIAM Rev.* **31**(2), 221–239 (1989)
501. Hanke, M., Hochbruck, M., Niethammer, W.: Experiments with Krylov subspace methods on a massively parallel computer. *Appl. Math.* **38**(6), 440–451 (1993)
502. Hansen, P.C.: Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. SIAM (1998)
503. Hansen, P.C.: Discrete Inverse Problems: Insight and Algorithms. SIAM (2010)

504. Hansen, P.C., Jensen, T.K.: Noise propagation in regularizing iterations for image deblurring. *Electron. Trans. Numer. Anal.* **31**(1), 204–220 (2008)
505. Hayami, K., Sugihara, M.: A geometric view of Krylov subspace methods on singular systems. *Numer. Linear Algebra Appl.* **18**(3), 449–469 (2011)
506. Hayami, K., Sugihara, M.: Corrigendum to: a geometric view of Krylov subspace methods on singular systems. *Numer. Linear Algebra Appl.* **21**(5), 701–702 (2014)
507. Hayami, K., Yin, J.F., Ito, T.: GMRES methods for least squares problems. *SIAM J. Matrix Anal. Appl.* **31**(5), 2400–2430 (2010)
508. Heller, D.: A determinant theorem with applications to parallel algorithms. *SIAM J. Numer. Anal.* **11**(3), 559–568 (1974)
509. Helsen, S., Kuijlaars, A.B.J., Van Barel, M.: Convergence of the isometric Arnoldi process. *SIAM J. Matrix Anal. Appl.* **26**(3), 782–809 (2005)
510. Hernandez, V., Roman, J., Tomas, A.: Parallel Arnoldi solvers with enhanced scalability via global communication rearrangement. *Parallel Comput.* **33**, 521–540 (2007)
511. Heroux, M.A., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., Phipps, E.T., Salinger, A.G., Thornquist, H.K., Tuminaro, R.S., Willenbring, J.M., Williams, A., Stanley, K.S.: An overview of the Trilinos project. *ACM Trans. Math. Soft.* (TOMS) **31**(3), 397–423 (2005)
512. Hessenberg, K.: Behandlung linearer eigenwertaufgaben mit hilfe der Hamilton-Cayleyschen gleichung. Tech. Rep. 1) Bericht der Reihe Numerische Verfahren, Institut für Praktische Mathematik, Technische Hochschule Darmstadt (1940)
513. Hestenes, M.R.: Iterative methods for solving linear equations. Tech. Rep. 52-9, NAML, National Bureau of Standards (1951). Reprinted in *J. of Optimization Theory and Applications*, v. 11, (1973), pp. 323–334
514. Hestenes, M.R.: The conjugate gradient method for solving linear systems. Tech. Rep. INA 54-11, National Bureau of Standards (1954)
515. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **49**(6), 409–436 (1952)
516. Heyouni, M.: The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides. *Numer. Algorithms* **26**(4), 317–332 (2001)
517. Heyouni, M., Essai, A.: Matrix Krylov subspace methods for linear systems with multiple right-hand sides. *Numer. Algorithms* **40**(2), 137–156 (2005)
518. Heyouni, M., Sadok, H.: On a variable smoothing procedure for Krylov subspace methods. *Linear Algebra Appl.* **268**, 131–149 (1998)
519. Heyouni, M., Sadok, H.: A new implementation of the CMRH method for solving dense linear systems. *J. Comput. Appl. Math.* **213**(22), 387–399 (2008)
520. Hicken, J.E., Zingg, D.W.: A simplified and flexible variant of GCROT for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.* **32**(3), 1672–1694 (2010)
521. Higham, N.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia (2002)
522. Higham, N.: Functions of Matrices: Theory and Computation. SIAM, Philadelphia (2008)
523. Hnětynková, I.: Krylov subspace approximations in linear algebraic problems. Ph.D. thesis, Charles University, Prague (2006)
524. Hnětynková, I., Kubínová, M., Plešinger, M.: Noise representation in residuals of LSQR, LSMR, and CRAIG regularization. *Linear Algebra Appl.* **533**, 357–379 (2017)
525. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.* **34**, 1911–1925 (1997)
526. Hochbruck, M., Lubich, C.: Error analysis of Krylov methods in a nutshell. *SIAM J. Sci. Comput.* **19**(2), 695–701 (1998)
527. Hoemmen, M.F.: Communication-avoiding Krylov subspace methods. Ph.D. thesis, University of California at Berkeley, USA (2010)
528. Hoemmen, M.F.: A communication-avoiding, hybrid-parallel, rank-revealing orthogonalization method. In: 2011 IEEE International Parallel and Distributed Processing Symposium, pp. 966–977. IEEE (2011)

529. Hoemmen, M.F., Yamazaki, I., Anzt, H., Tomov, S., Dongarra, J.: Studying the performance of CA-GMRES on multicores with multiple GPUs. Tech. Rep. No. SAND2013-8969C, Sandia National Lab. (SNL-NM), Albuquerque, NM USA (2013)
530. Hoffmann, W.: Iterative algorithms for Gram-Schmidt orthogonalization. Computing **41**, 335–348 (1989)
531. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press (1987). 2nd edn. (2013)
532. Horst, P.: A method for determining the coefficients of a characteristic equation. Ann. Math. Statist. **6**(2), 83–84 (1935)
533. Householder, A.S.: Unitary triangularization of a nonsymmetric matrix. J. ACM **5**(4), 339–342 (1958)
534. Householder, A.S.: Numerical Analysis. In: Saaty, T.L. (ed.) Lectures on Modern Mathematics, pp. 59–97. Wiley, New York, London (1963)
535. Householder, A.S.: The Theory of Matrices in Numerical Analysis. Blaisdell, New York (1964). Reprinted by Dover, New York (1975)
536. Householder, A.S., Bauer, F.L.: On certain methods for expanding the characteristic polynomials. Numer. Math. **1**, 29–37 (1959)
537. Huckle, T.: The Arnoldi method for normal matrices. SIAM J. Matrix Anal. Appl. **15**(2), 479–489 (1994)
538. Huckle, T.: Low-rank modification of the unsymmetric Lanczos algorithm. Math. Comput. **64**(212), 1577–1588 (1995)
539. Huhtanen, M.: Ideal GMRES can be bounded from below by three factors. Tech. Rep. A412, Helsinki University of Technology (1999)
540. Ikebe, Y.: On inverses of Hessenberg matrices. Linear Algebra Appl. **24**, 93–97 (1979)
541. Ilin, V.P.: On iterative processes in the Krylov-Sonneveld subspaces. In: Sergeyev, Y.D., Kvasov, D.E., Dell'Accio, F., Mukhametzhanov, M.S. (eds.) AIP Conference Proceedings 1776-1. AIP Publishing (2016)
542. Imakura, A., Li, R.C., Zhang, S.L.: Locally optimal and heavy ball GMRES methods. Jpn. J. Indust. Appl. Math. **33**(2), 471–499 (2016)
543. Imakura, A., Sogabe, T., Zhang, S.L.: An efficient variant of the GMRES(m) method based on the error equations. E. Asian J. Appl. Math. **2**, 1–22 (2012)
544. Imakura, A., Sogabe, T., Zhang, S.L.: A look-back-type restart for the restarted Krylov subspace methods for solving non-Hermitian linear systems. Jpn. J. Indust. Appl. Math. **35**(2), 835–859 (2018)
545. Imberti, D., Erhel, J.: Vary the s in your s-step GMRES. Electron. Trans. Numer. Anal. **47**, 206–230 (2017)
546. Ipsen, I.C.F.: A different approach to bounding the minimal residual norm in Krylov methods. Tech. Rep. CRSC-TR98-19, Department of Mathematics, North Carolina State University (1998)
547. Ipsen, I.C.F.: A note on the field of values of non-normal matrices. Tech. Rep. CRSC-TR98-26, Department of Mathematics, North Carolina State University (1998)
548. Ipsen, I.C.F.: Expressions and bounds for the GMRES residual. BIT Numer. Math. **40**(3), 524–535 (2000)
549. Ipsen, I.C.F., Meyer, C.D.: The idea behind Krylov methods. Appl. Numer. Math. **889–899**, (1998)
550. Ipsen, I.C.F., Rehman, R.: Perturbation bounds for determinants and characteristic polynomials. SIAM J. Matrix Anal. Appl. **30**(2), 762–776 (2008)
551. Ito, T., Hayami, K.: Preconditioned GMRES methods for least squares problems. Jpn. J. Indust. Appl. Math. **25**(2), 185 (2008)
552. Itoh, S., Namekawa, Y.: An improvement in DS-BiCGstab(ℓ) and its application for linear systems in lattice QCD. J. Comput. Appl. Math. **159**, 65–75 (2003)
553. Jagels, C., Reichel, L.: The extended Krylov subspace method and orthogonal Laurent polynomials. Linear Algebra Appl. **431**(3–4), 441–458 (2009)

554. Jagels, C., Reichel, L.: Recursion relations for the extended Krylov subspace method. *Linear Algebra Appl.* **434**(7), 1716–1732 (2011)
555. Jain, P., Manglani, K., Venkatapathi, M.: Significance of error estimation in iterative solution of linear systems: estimation algorithms and analysis for CG, Bi-CG and GMRES. arXiv preprint, [arXiv:1705.08806](https://arxiv.org/abs/1705.08806) (2017)
556. Jbilou, K.: A note on the block and seed BiCGSTAB algorithms for nonsymmetric multiple linear systems. *Am. J. Algorithm Comput.* **3**(1), 1–13 (2016)
557. Jbilou, K., Messaoudi, A., Sadok, H.: Global GMRES algorithm for solving nonsymmetric linear systems of equations with multiple right-hand sides. *Appl. Numer. Math.* **31**, 49–63 (1999)
558. Jbilou, K., Sadok, H., Tinaztepe, A.: Oblique projection methods for linear systems with multiple right-hand sides. *Electron. Trans. Numer. Anal.* **20**, 119–138 (2005)
559. Jea, K.C.: Generalized conjugate gradient acceleration of iterative methods. Ph.D. thesis, Center for Numerical Analysis, University of Texas at Austin (1982). Report CNA-176
560. Jea, K.C., Young, D.M.: On the simplification of generalized conjugate-gradient methods for nonsymmetrizable linear systems. *Linear Algebra Appl.* **52**, 399–417 (1983)
561. Jea, K.C., Young, D.M.: Comments on three papers by Cornelius Lanczos. Tech. Rep. CNA-252, Center for Numerical Analysis, University of Texas at Austin, USA (1990)
562. Jensen, T.K., Hansen, P.C.: Iterative regularization with minimum-residual methods. *BIT Numer. Math.* **47**(1), 103–120 (2007)
563. Jézéquel, F., Chesneauux, J.M.: CADNA: a library for estimating round-off error propagation. *Comput. Phys. Comm.* **178**(12), 933–955 (2008)
564. Jia, Z.: The convergence of generalized Lanczos methods for large unsymmetric eigenproblems. *SIAM J. Matrix Anal. Appl.* **16**(3), 843–862 (1995)
565. Jia, Z.: On IOM(q): the incomplete orthogonalization method for large unsymmetric linear systems. *Numer. Linear Algebra Appl.* **3**(6), 491–512 (1996)
566. Jia, Z.: On IGMRES: an incomplete generalized minimal residual method for large unsymmetric linear systems. *Sci. China Ser. A* **41**(12), 1278–1288 (1998)
567. Jia, Z.: Composite orthogonal projection methods for large matrix eigenproblems. *Sci. China Ser. A* **42**(6), 577–585 (1999)
568. Jia, Z.: Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi method and an implicitly restarted refined Arnoldi algorithm. *Linear Algebra Appl.* **287**(1–3), 191–214 (1999)
569. Jia, Z.: The convergence of harmonic Ritz values, harmonic Ritz vectors, and refined harmonic Ritz vectors. *Math. Comput.* **74**(251), 1441–1456 (2004)
570. Jia, Z.: Some theoretical comparisons of refined Ritz vectors and Ritz vectors. *Sci. China Ser. A* **47**, 222–233 (2004)
571. Jia, Z., Stewart, G.: An analysis of the Rayleigh-Ritz method for approximating eigenspaces. *Math. Comput.* **70**(234), 637–647 (2000)
572. Jibetean, D., de Klerk, E.: Global optimization of rational functions: a semidefinite programming approach. *Math. Program.* **106**, 93–109 (2006)
573. Jing, Y.F., Huang, T.Z., Carpentieri, B., Duan, Y.: Exploiting the composite step strategy to the biconjugate-orthogonal residual method for non-Hermitian linear systems. *J. Appl. Math. ID* **408167**, (2013)
574. Jing, Y.F., Huang, T.Z., Duan, Y., Carpentieri, B.: A comparative study of iterative solutions to linear systems arising in quantum mechanics. *J. Comp. Phys.* **229**(22), 8511–8520 (2010)
575. Jiránek, P.: Limiting accuracy of iterative methods. Ph.D. thesis, Technical University Liberec, Czech Republic (2007)
576. Jiránek, P., Rozložník, M.: Adaptive version of Simpler GMRES. *Numer. Algorithms* **53**(1), 93–112 (2010)
577. Jiránek, P., Rozložník, M., Gutknecht, M.H.: How to make simpler GMRES and GCR more stable. *SIAM J. Matrix Anal. Appl.* **30**(4), 1483–1499 (2008)
578. Jiránek, P., Strakoš, Z., Vohralík, M.: A posteriori error estimates including algebraic error and stopping criteria for iterative solvers. *SIAM J. Sci. Comput.* **32**, 1567–1590 (2010)

579. Joly, P., Eymard, R.: Preconditioned biconjugate gradient methods for numerical reservoir simulation. *J. Comp. Phys.* **91**(2), 298–309 (1990)
580. Joubert, W.: Generalized conjugate gradient and Lanczos methods for the solution of non-symmetric systems of linear equations. Ph.D. thesis, Center for Numerical Analysis, University of Texas at Austin, USA (1990)
581. Joubert, W.: Lanczos methods for the solution of nonsymmetric systems of linear equations. *SIAM J. Matrix Anal. Appl.* **13**(3), 926–943 (1992)
582. Joubert, W.: On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems. *Numer. Linear Algebra Appl.* **1**(5), 427–447 (1994)
583. Joubert, W.: A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems. *SIAM J. Sci. Comput.* **15**(2), 427–439 (1994)
584. Joubert, W., Carey, G.F.: Parallelizable restarted iterative methods for nonsymmetric linear systems. Part I: Theory. *Int. J. Comput. Math.* **44**(1-4), 243–267 (1992)
585. Joubert, W., Carey, G.F.: Parallelizable restarted iterative methods for nonsymmetric linear systems. Part II: parallel implementation. *Int. J. Comput. Math.* **44**(1-4), 269–290 (1992)
586. Joubert, W.D., Young, D.M.: Necessary and sufficient conditions for the simplification of generalized conjugate-gradient algorithms. *Linear Algebra Appl.* **88**, 449–485 (1987)
587. Kahan, W., Parlett, B.N., Jiang, E.: Residual bounds on approximate eigensystems of non-normal matrices. *SIAM J. Numer. Anal.* **19**(3), 470–484 (1982)
588. Kamgnia, E., Philippe, B.: Counting eigenvalues in domains of the complex field. *Electron. Trans. Numer. Anal.* **40**, 1–16 (2013)
589. Karlsson, R.: A study of some roundoff effects of the GMRES method. Tech. Rep. Li-THE-MAT-R-1990-11, Department of Mathematics, Linköping University, Sweden (1991)
590. Kasenally, E.M.: GMBACK: a generalised minimum backward error algorithm for nonsymmetric linear systems. *SIAM J. Sci. Comput.* **16**(3), 698–719 (1995)
591. Kasenally, E.M., Simoncini, V.: Analysis of a minimum perturbation algorithm for nonsymmetric linear systems. *SIAM J. Numer. Anal.* **34**(1), 48–66 (1997)
592. Kehl, R., Nabben, R., Szyld, D.B.: Adaptive multilevel Krylov methods. *Electron. Trans. Numer. Anal.* **51**, 512–528 (2017)
593. Kenney, C., Laub, A.J.: Controllability and stability radii for companion form systems. *Math. Control Signals Systems* **1**(3), 239–256 (1988)
594. Khabaza, I.M.: An iterative least-square method suitable for solving large sparse matrices. *Comput. J.* **6**(2), 202–206 (1963)
595. Kilmer, M., Miller, E., Rappaport, C.: QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides. *SIAM J. Sci. Comput.* **23**(3), 761–780 (2001)
596. Kim, S.K., Chronopoulos, A.T.: A class of Lanczos-like algorithms implemented on parallel computers. *Parallel Comput.* **17**, 763–778 (1991)
597. Kim, S.K., Chronopoulos, A.T.: An efficient nonsymmetric Lanczos method on parallel vector computers. *J. Comput. Appl. Math.* **42**(3), 357–374 (1992)
598. Kittaneh, F.: Singular values of companion matrices and bounds on zeros of polynomials. *SIAM J. Matrix Anal. Appl.* **16**(1), 333–340 (1995)
599. Kittaneh, F.: Bounds and a majorization for the real parts of the zeros of polynomials. *Proc. Am. Math. Soc.* **135**(3), 659–664 (2007)
600. Kittaneh, F., Shebrawi, K.: Some decomposition results for companion matrices. *J. Math. Anal. Appl.* **318**, 626–633 (2006)
601. Knizhnerman, L.: Calculation of functions of unsymmetric matrices using Arnoldi's method. *Comput. Math. Math. Phys.* **31**(1), 1–9 (1992)
602. Knizhnerman, L.: Error bounds in Arnoldi's method: the case of a normal matrix. *Comput. Math. Math. Phys.* **32**(9), 1199–1211 (1992)
603. Knizhnerman, L.: Error bounds for the Arnoldi method: a set of extreme eigenpairs. *Linear Algebra Appl.* **296**, 191–211 (1999)
604. Knizhnerman, L.: On GMRES-equivalent bounded operators. *SIAM J. Matrix Anal. Appl.* **22**(1), 195–212 (2000)

605. Knizhnerman, L., Simoncini, V.: A new investigation of the extended Krylov subspace method for matrix function evaluations. *Numer. Linear Algebra Appl.* **17**(4), 615–638 (2010)
606. Kopal, J., Rozložník, M., Tůma, M., Smoktunowicz, A.: Rounding error analysis of orthogonalization with a non-standard inner product. *Numer. Math.* **52**(4), 1035–1058 (2012)
607. Krasnopolsky, B.: The reordered BiCGStab method for distributed memory computer systems. *Procedia Comput. Sci.* **1**, 213–218 (2012)
608. Krylov, A.N.: O čislennoj rešenii uravnenija, kotorym v tehnicheskikh voprasah opredelja-jutsja častoy malyh kolebanij material'nyh. *Izv. Adad. Nauk SSSR, ser Fiz.-Mat.* **4**, 491–539 (1931). Possible translation: “On the numerical solution of the equation by which the frequency of small oscillations is determined in technical problems”
609. Krylov, A.N.: *Moi Vospominaniya*. Izdatel'stvo Akademii Nauk SSSR, Moscow and Leningrad (1942). Translated to English by Laura N. Meyerovich with the title Professor Krylov's Navy, Memoir of Alexey N. Krylov, Magnet Publishing (2014)
610. Kubínová, M., Soodhalter, K.: Admissible and attainable convergence behavior of block Arnoldi and GMRES. *SIAM J. Matrix Anal. Appl.* **41**(2), 464–486 (2020)
611. Kuznetsov, S.V.: Perturbation bounds of the Krylov bases and associated Hessenberg forms. *Linear Algebra Appl.* **265**(1–3), 1–28 (1997)
612. Lagomasino, G.L., Reichel, L., Wunderlich, L.: Matrices, moments, and rational quadrature. *Linear Algebra Appl.* **429**(10), 2540–2554 (2008)
613. Lambers, J.V.: Krylov subspace spectral methods for variable-coefficient initial-boundary value problems. *Electron. Trans. Numer. Anal.* **20**, 212–234 (2005)
614. Lambers, J.V.: Practical implementation of Krylov subspace spectral methods. *J. Sci. Comput.* **32**, 449–476 (2007)
615. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.* **45**, 255–282 (1950)
616. Lanczos, C.: Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.* **49**, 33–53 (1952)
617. Lang, M., Frenzel, B.C.: Polynomial root finding. *IEEE Signal Process. Lett.* **1**(10), 141–143 (1994)
618. Langou, J.: Iterative methods for solving linear systems with multiple right-hand sides. Ph.D. thesis, Institut National des Sciences Appliquées de Toulouse, France (2003)
619. Langou, J.: Translation and modern interpretation of Laplace's Théorie Analytique des Probabilités, pp. 505–512, 516–520. arXiv preprint, [arXiv:0907.4695v1](https://arxiv.org/abs/0907.4695v1) (2009)
620. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. SIAM, Philadelphia (1995)
621. Le Calvez, C., Molina, B.: Implicitly restarted and deflated GMRES. *Numer. Algorithms* **21**(1–4), 261–285 (1999)
622. Le Calvez, C., Saad, Y.: Modified Krylov acceleration for parallel environments. *Appl. Numer. Math.* **30**(2–3), 191–212 (1999)
623. Lehoucq, R.B.: Analysis and implementation of an implicitly restarted Arnoldi iteration. Ph.D. thesis, Rice University, Houston, USA (1995)
624. Lehoucq, R.B.: The computation of elementary unitary matrices. *ACM Trans. Math. Soft. (TOMS)* **22**(4), 393–400 (1996)
625. Lehoucq, R.B.: Implicitly restarted Arnoldi methods and subspace iteration. *SIAM J. Matrix Anal. Appl.* **23**(2), 551–562 (2001)
626. Lehoucq, R.B., Sorensen, D.C.: Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.* **17**(4), 789–821 (1996)
627. Leon, S.J., Björck, A., Gander, W.: Gram-Schmidt orthogonalization: 100 years and more. *Numer. Linear Algebra Appl.* **20**(3), 492–532 (2013)
628. Leyk, Z.: Breakdowns and stagnation in iterative methods. *BIT Numer. Math.* **37**(2), 377–403 (1997)
629. Li, R.C.: Convergence of CG and GMRES on a tridiagonal Toeplitz linear system. *BIT Numer. Math.* **47**(3), 577–599 (2007)
630. Li, R.C., Zhang, W.: The rate of convergence of GMRES on a tridiagonal Toeplitz linear system. *Numer. Math.* **112**, 267–293 (2009)

631. Li, R.C., Zhang, W.: The rate of convergence of GMRES on a tridiagonal Toeplitz linear system. II. *Linear Algebra Appl.* **431**, 2425–2436 (2009)
632. Liang, Z.S., Oyanagi, Y.: Orthomin(k) method for linear least squares problem. *J. Inf. Process.* **14**(2), 121–125 (1991)
633. Liesen, J.: Computable convergence bounds for GMRES. *SIAM J. Matrix Anal. Appl.* **21**(3), 882–903 (2000)
634. Liesen, J., Parlett, B.N.: On nonsymmetric saddle point matrices that allow conjugate gradient iterations. *Numer. Math.* **108**(4), 605–624 (2008)
635. Liesen, J., Rozložník, M., Strakoš, Z.: Least squares residuals and minimal residual methods. *SIAM J. Sci. Comput.* **23**(5), 1503–1525 (2002)
636. Liesen, J., Saylor, P.E.: Orthogonal Hessenberg reduction and orthogonal Krylov subspace bases. *SIAM J. Numer. Anal.* **42**(5), 2148–2158 (2005)
637. Liesen, J., Strakoš, Z.: Convergence of GMRES for tridiagonal Toeplitz matrices. *SIAM J. Matrix Anal. Appl.* **26**(1), 266–251 (2004)
638. Liesen, J., Strakoš, Z.: GMRES convergence analysis for a convection-diffusion model problem. *SIAM J. Sci. Comput.* **26**(6), 1989–2009 (2005)
639. Liesen, J., Strakoš, Z.: On optimal short recurrences for generating orthogonal Krylov subspace bases. *SIAM Rev.* **50**(3), 485–503 (2008)
640. Liesen, J., Strakoš, Z.: *Krylov Subspace Methods. Oxford University Press, Principles and Analysis* (2013)
641. Liesen, J., Tichý, P.: Convergence analysis of Krylov subspace methods. *GAMM Mitt.* **Band 27**(Heft 2) (2004)
642. Liesen, J., Tichý, P.: The worst-case GMRES for normal matrices. *BIT Numer. Math.* **44**, 79–98 (2004)
643. Liesen, J., Tichý, P.: On the worst-case convergence of MR and CG for symmetric positive definite tridiagonal Toeplitz matrices. *Electron. Trans. Numer. Anal.* **20**, 180–197 (2005)
644. Liesen, J., Tichý, P.: GMRES convergence and the polynomial numerical hull for a Jordan block. *Tech. Rep. Preprint 34-2006*, Institute of Mathematics, Technische Universität Berlin (2006)
645. Liesen, J., Tichý, P.: On best approximations of polynomials in matrices in the matrix 2-norm. *SIAM J. Matrix Anal. Appl.* **31**(2), 853–863 (2009)
646. Liesen, J., Tichý, P.: The field of values bound on ideal GMRES. arXiv preprint, [arXiv:1211.5969](https://arxiv.org/abs/1211.5969) (2012)
647. Liesen, J., Tichý, P.: Max-min and min-max approximation problems for normal matrices revisited. *Electron. Trans. Numer. Anal.* **41**, 159–166 (2014)
648. Liesen, J., Tichý, P.: The field of value bounds on ideal GMRES. Charles University, Prague, Tech. Rep. preprint (2018)
649. Linden, H.: Bounds for the zeros of polynomials from eigenvalues and singular values of some companion matrices. *Linear Algebra Appl.* **271**, 41–82 (1998)
650. Liu, Q., Morgan, R.B., Wilcox, W.: Polynomial preconditioned GMRES and GMRES-DR. *SIAM J. Sci. Comput.* **37**(5), S407–S428 (2015)
651. Liu, Q., Shen, D., Jia, Z.: Making global simpler GMRES more stable. *Numer. Linear Algebra Appl.* **e2203** (2018). <https://doi.org/10.1002/nla.2203>
652. Ma, S., Chronopoulos, A.T.: Implementation of iterative methods for large sparse nonsymmetric linear systems on a parallel vector machine. *Int. J. Supercomputing Appl.* **4**(4), 9–24 (1990)
653. Mach, T., Pranić, M.S., Vandebril, R.: Computing approximate extended Krylov subspaces without explicit inversion. *Electron. Trans. Numer. Anal.* **40**, 414–435 (2013)
654. Mach, T., Pranić, M.S., Vandebril, R.: Computing approximate (block) rational Krylov subspaces without explicit inversion with extensions to symmetric matrices. *Electron. Trans. Numer. Anal.* **43**, 100–124 (2014)
655. Mansfield, L.: On the use of deflation to improve the convergence of conjugate gradient iteration. *Commun. Appl. Numer. Meth.* **4**(2), 151–156 (1988)

656. Manteuffel, T.A.: The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math.* **28**, 307–327 (1977)
657. Manteuffel, T.A.: Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration. *Numer. Math.* **31**, 183–208 (1978)
658. Manteuffel, T.A., Otto, J.S.: On the roots of the orthogonal polynomials and residual polynomials associated with a conjugate gradient method. *Numer. Linear Algebra Appl.* **1**(5), 449–475 (1994)
659. Manteuffel, T.A., Starke, G.: On hybrid iterative methods for nonsymmetric systems of linear equations. *Numer. Math.* **73**(4), 489–506 (1996)
660. Marchuk, G.I., Kuznetsov, Y.A.: On optimal iteration processes. *Soviet Math. Dokl.* **9**(4), 1041–1045 (1968)
661. Marchuk, G.I., Kuznetsov, Y.A.: Méthodes itératives et fonctionnelles quadratiques. In: Lions, J.L., Marchuk, G.I. (eds.) *Sur les méthodes numériques en sciences physiques et économiques*, pp. 1–117. Dunod (1974)
662. Marden, M.: Geometry of polynomials. The American Mathematical Society Surveys, vol. 3, 2nd edn. American Mathematical Society (1966)
663. Markham, G.: Conjugate gradient type methods for indefinite, asymmetric, and complex systems. *IMA J. Numer. Anal.* **10**(2), 155–170 (1990)
664. Marrero, J.A., Rachidi, M., Tomeo, V.: On new algorithms for inverting Hessenberg matrices. *J. Comput. Appl. Math.* **252**, 12–20 (2013)
665. Marrero, J.A., Tomeo, V.: Inverses of regular Hessenberg matrices. In: Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2010 (2010)
666. Marrero, J.A., Tomeo, V.: On the closed representation for the inverses of Hessenberg matrices. *J. Comput. Appl. Math.* **236**(12), 2962–2970 (2012)
667. McInnes, L., Smith, B., Zhang, H., Mills, R.T.: Hierarchical Krylov and nested Krylov methods for extreme-scale computing. *Parallel Comput.* **40**(1), 17–31 (2014)
668. Melman, A.: Generalizations of Gershgorin disks and polynomial zeros. *Proc. Am. Math. Soc.* **138**(7), 2349–2364 (2010)
669. Melman, A.: Modified Gershgorin disks for companion matrices. *SIAM Rev.* **54**(2), 355–373 (2012)
670. Meng, J., Zhu, P.Y., Li, H.B., Gu, X.M.: A deflated block flexible GMRES-DR method for linear systems with multiple right-hand sides. *Electron. Trans. Numer. Anal.* **41**, 478–496 (2014)
671. Meurant, G.: A review on the inverse of tridiagonal and block tridiagonal symmetric matrices. *SIAM J. Matrix Anal. Appl.* **13**(3), 707–728 (1992)
672. Meurant, G.: The computation of bounds for the norm of the error in the conjugate gradient algorithm. *Numer. Algorithms* **16**, 77–87 (1997)
673. Meurant, G.: Computer Solution of Large Linear Systems. North-Holland, Amsterdam (1999)
674. Meurant, G.: Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm. *Numer. Algorithms* **22**, 353–365 (1999)
675. Meurant, G.: Estimates of the ℓ_2 norm of the error in the conjugate gradient algorithm. *Numer. Algorithms* **40**(2), 157–169 (2005)
676. Meurant, G.: The Lanczos and Conjugate Gradient Algorithms, from Theory to Finite Precision Computations. SIAM, Philadelphia (2006)
677. Meurant, G.: Estimates of the norm of the error in solving linear systems with FOM and GMRES. *SIAM J. Sci. Comput.* **33**(5), 2686–2705 (2011)
678. Meurant, G.: On the residual norm in FOM and GMRES. *SIAM J. Matrix Anal. Appl.* **32**(2), 394–411 (2011)
679. Meurant, G.: The complete stagnation of GMRES for $n < 4$. *Electron. Trans. Numer. Anal.* **39**, 75–101 (2012)
680. Meurant, G.: The computation of isotropic vectors. *Numer. Algorithms* **60**(2), 193–204 (2012)

681. Meurant, G.: GMRES and the Arioli, Pták and Strakoš parametrization. *BIT Numer. Math.* **52**(3), 687–702 (2012)
682. Meurant, G.: Necessary and sufficient conditions for GMRES complete and partial stagnation. *Appl. Numer. Math.* **75**, 100–107 (2014)
683. Meurant, G.: On the location of the Ritz values in the Arnoldi process. *Electron. Trans. Numer. Anal.* **43**, 188–212 (2014–2015)
684. Meurant, G.: The coefficients of the FOM and GMRES residual polynomials. *SIAM J. Matrix Anal. Appl.* **38**(1), 96–117 (2017)
685. Meurant, G.: An optimal Q-OR Krylov subspace method for solving linear systems. *Electron. Trans. Numer. Anal.* **47**, 127–152 (2017)
686. Meurant, G., Duintjer Tebbens, J.: The role eigenvalues play in forming GMRES residual norms with non-normal matrices. *Numer. Algorithms* **68**(1), 143–165 (2015)
687. Meurant, G., Sommariva, A.: On the computation of sets of points with low Lebesgue constant on the unit disk. *J. Comput. Appl. Math.* **345**(1), 388–404 (2019)
688. Meurant, G., Strakoš, Z.: The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numer.* **15**, 471–542 (2006)
689. Meurant, G., Tichý, P.: Approximating the extreme Ritz values and upper bounds for the A-norm of the error in CG. *Numer. Algorithms* **82**, 937–968 (2018)
690. Meyer, C.D.: *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia (2000)
691. Meyerovich, L.M.: Professor Krylov's navy, memoir of Alexey N. Krylov. Magnet Publishing (2014). English translation of Krylov's memoirs
692. Meza, J.C.: A modification to the GMRES method for ill-conditioned linear systems. Tech. Rep. SAND95-8220, Scientific Computing Department, Sandia National Laboratories, Livermore, CA (1995)
693. Milovanović, G.V., Rassias, T.M.: Inequalities for polynomial zeros. In: *Survey on Classical Inequalities*, pp. 165–202. Springer, The Netherlands (2000)
694. Montagnac, M., Chesneaux, J.M.: Dynamic control of a bicgstab algorithm. *Appl. Numer. Math.* **32**(1), 103–117 (2000)
695. Morgan, R.B.: A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.* **16**(4), 1154–1171 (1995)
696. Morgan, R.B.: Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Anal. Appl.* **21**(4), 1112–1135 (2000)
697. Morgan, R.B.: GMRES with deflated restarting. *SIAM J. Sci. Comput.* **24**(1), 20–37 (2002)
698. Morgan, R.B.: Restarted block-GMRES with deflation of eigenvalues. *Appl. Numer. Math.* **54**(2), 222–236 (2005)
699. Morgan, R.B., Yang, Z., Zhong, B.: Pseudoeigenvector bases and deflated GMRES for highly nonnormal matrices. *Numer. Linear Algebra Appl.* **23**(6), 1032–1045 (2016)
700. Morgan, R.B., Zeng, M.: Harmonic projection methods for large non-symmetric eigenvalue problems. *Numer. Linear Algebra Appl.* **5**, 33–55 (1998)
701. Morgan, R.B., Zeng, M.: A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity. *Linear Algebra Appl.* **415**(1), 96–113 (2006)
702. Morikuni, K., Hayami, K.: Inner-iteration Krylov subspace methods for least squares problems. *SIAM J. Matrix Anal. Appl.* **34**(1), 1–22 (2013)
703. Morikuni, K., Hayami, K.: Convergence of inner-iteration GMRES methods for rank-deficient least squares problems. *SIAM J. Matrix Anal. Appl.* **36**(1), 225–250 (2015)
704. Morikuni, K., Reichel, L., Hayami, K.: FGMRES for linear discrete ill-posed problems. *Appl. Numer. Math.* **75**, 175–187 (2014)
705. Morikuni, K., Rozložník, M.: On GMRES for singular EP and GP systems. *SIAM J. Matrix Anal. Appl.* **39**(2), 1033–1048 (2018)
706. Moriya, K., Nodera, T.: Breakdown-free ML(k)BiCGStab algorithm for non-Hermitian linear systems. In: *International Conference on Computational Science and Its Applications*. Springer, Berlin, Heidelberg (2005)
707. Moriya, K., Nodera, T.: Usage of the convergence test of the residual norm in the Tsuno-Nodera version of the GMRES algorithm. *ANZIAM J.* **49**, 293–308 (2007)

708. Moufawad, S.: Enlarged Krylov subspace methods and preconditioners for avoiding communication. Ph.D. thesis, Université P. et M. Curie Paris VI, France (2015)
709. Muller, J.M., Brunie, N., de Dinechin, F., Jeannerod, C.P., Joldes, M., Lefèvre, V., Melquiond, G., Revol, N., Torres, S.: Handbook of Floating-Point Arithmetic. Birkhäuser (2018)
710. Nachtigal, N.M.: A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non Hermitian linear systems. Ph.D. thesis, MIT (1991). RIACS Tech. Report 91.19, NASA Ames
711. Nachtigal, N.M., Reddy, S.C., Trefethen, L.N.: How fast are nonsymmetric matrix iterations? SIAM J. Matrix Anal. Appl. **13**(3), 778–795 (1992)
712. Nachtigal, N.M., Reichel, L., Trefethen, L.N.: A hybrid GMRES algorithm for nonsymmetric linear systems. SIAM J. Matrix Anal. Appl. **13**(3), 796–825 (1992)
713. Nassif, N., Erhel, J., Philippe, B.: Introduction to Computational Linear Algebra. CRC Press (2015)
714. Neuenhofen, M.P.: A restarted GMRES-based implementation of IDR(s)stab(ℓ) to yield higher robustness. Master's thesis, RWTH Aachen University, Germany (2017)
715. Nevanlinna, O.: Convergence of iterations for linear equations, vol. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel (1993)
716. Neymeyr, K., Zhou, M.: Convergence analysis of restarted Krylov subspace eigensolvers. SIAM J. Matrix Anal. Appl. **37**(3), 955–975 (2016)
717. Nicolaides, R.A.: Deflation of conjugate gradients with application to boundary value problems. SIAM J. Numer. Anal. **24**(2), 355–365 (1987)
718. Nie, J., Demmel, J., Gu, M.: Global minimization of rational functions and the nearest GCDs. J. Global Optim. **40**(4), 697–718 (2008)
719. Niu, Q., Lu, L.Z.: Restarted GMRES method augmented with the combination of harmonic Ritz vectors and error approximations. Int. J. Math. Comput. Phys. Elec. Comput. Eng. **6**(8), 857–864 (2012)
720. Niu, Q., Lu, L.Z., Liu, G.: Accelerated GCRO-DR method for solving sequences of systems of linear equations. J. Comput. Appl. Math. **253**, 131–141 (2013)
721. Oettli, W., Prager, W.: Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. Numer. Math. **6**(1), 405–409 (1964)
722. O'Leary, D.P.: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. **29**, 293–322 (1980)
723. de Oliveira, G.: Matrices with prescribed characteristic polynomial and a prescribed submatrix. I. Pac. J. Math. **29**(3), 653–661 (1969)
724. de Oliveira, G.: Matrices with prescribed characteristic polynomial and a prescribed submatrix. II. Pac. J. Math. **29**(3), 663–667 (1969)
725. Onoue, Y., Fujino, S., Nakashima, N.: A difference between easy and profound preconditionings of IDR(s) method. Trans. JSCE Pap. **20080023**, (2008). (in Japanese)
726. Onoue, Y., Fujino, S., Nakashima, N.: Improved IDR(s) method for gaining very accurate solutions. World Acad. Sci. Eng. Technol. **55**, (2009)
727. Onoue, Y., Fujino, S., Nakashima, N.: An overview of a family of new iterative methods based on IDR theorem and its estimation. In: Proceedings of the International MultiConference of Engineers and Computer Scientists 2009, Vol II IMECS (2009)
728. Opfer, G., Schober, G.: Richardson's iteration for nonsymmetric matrices. Linear Algebra Appl. **58**, 343–361 (1984)
729. Overton, M.M.: Numerical Computing with IEEE Floating Point Arithmetic. SIAM, Philadelphia (2001)
730. Paige, C.C.: The computation of eigenvalues and eigenvectors of very large sparse matrices. Ph.D. thesis, University of London, UK (1971)
731. Paige, C.C.: A useful form of unitary matrix obtained from any sequence of unit 2-norm n -vectors. SIAM J. Matrix Anal. Appl. **31**(2), 565–583 (2009)
732. Paige, C.C.: An augmented stability result for the Lanczos Hermitian matrix tridiagonalization process. SIAM J. Matrix Anal. Appl. **31**(5), 2347–2359 (2010)

733. Paige, C.C., Panayotov, I.: Hessenberg matrix properties and Ritz vectors in the finite-precision Lanczos tridiagonalization process. *SIAM J. Matrix Anal. Appl.* **32**(4), 1079–1094 (2011)
734. Paige, C.C., Panayotov, I., Zemke, J.P.M.: An augmented analysis of the perturbed two-sided Lanczos tridiagonalization process. *Linear Algebra Appl.* **447**, 119–132 (2014)
735. Paige, C.C., Parlett, B.N., van der Vorst, H.: Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.* **2**(2), 115–133 (1995)
736. Paige, C.C., Rozložník, M., Strakoš, Z.: Modified Gram-Schmidt (MGS), least squares and backward stability of MGS-GMRES. *SIAM J. Matrix Anal. Appl.* **28**(1), 264–284 (2006)
737. Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**(4), 617–629 (1975)
738. Paige, C.C., Saunders, M.A.: Algorithm 583, LSQR: sparse linear equations and least squares problems. *ACM Trans. Math. Soft. (TOMS)* **8**, 195–209 (1982)
739. Paige, C.C., Saunders, M.A.: LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Soft. (TOMS)* **8**, 43–71 (1982)
740. Paige, C.C., Strakoš, Z.: Bounds for the least squares distance using scaled total least squares. *Numer. Math.* **91**, 93–115 (2002)
741. Paige, C.C., Strakoš, Z.: Residual and backward error bounds in minimum residual Krylov subspace methods. *SIAM J. Sci. Comput.* **23**(6), 1898–1923 (2002)
742. Paige, C.C., Van Dooren, P.: Sensitivity analysis of the Lanczos reduction. *Numer. Linear Algebra Appl.* **6**, 29–50 (1999)
743. Paige, C.C., Wülling, W.: Properties of a unitary matrix obtained from a sequence of normalized vectors. *SIAM J. Matrix Anal. Appl.* **35**(2), 526–545 (2014)
744. Papež, J., Liesen, J., Strakoš, Z.: Distribution of the discretization and algebraic error in numerical solution of partial differential equations. *Linear Algebra Appl.* **449**, 89–114 (2014)
745. Papež, J., Strakoš, Z.: On a residual-based a posteriori error estimator for the total error. *IMA J. Numer. Anal.* **38**, 1164–1184 (2018)
746. Papež, J., Strakoš, Z., Vohralík, M.: Estimating and localizing the algebraic and total numerical errors using flux reconstructions. *Numer. Math.* **138**, 681–721 (2018)
747. Parks, M.L., Soodhalter, K.M., Szyld, D.B.: A block Recycled GMRES method with investigations into aspects of solver performance. Tech. Rep. 16-04-04, Department of Mathematics, Temple University, USA (2016). Revised (2017)
748. Parks, M.L., de Sturler, E., Mackey, G., Johnson, D.D., Maiti, S.: Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.* **28**(5), 1651–1674 (2006)
749. Parlett, B.N.: Canonical decomposition of Hessenberg matrices. *Math. Comput.* **21**(98), 223–227 (1967)
750. Parlett, B.N.: Normal Hessenberg and moment matrices. *Linear Algebra Appl.* **6**, 37–43 (1973)
751. Parlett, B.N.: The Symmetric Eigenvalue Problem. Prentice Hall (1980). Reprinted by SIAM (1998)
752. Parlett, B.N.: Reduction to tridiagonal form and minimal realizations. *SIAM J. Matrix Anal. Appl.* **13**(2), 567–593 (1992)
753. Parlett, B.N., Barszcz, E.: Another orthogonal matrix. *Linear Algebra Appl.* **417**(2–3), 342–346 (2006)
754. Parlett, B.N., Strang, G.: Matrices with prescribed Ritz values. *Linear Algebra Appl.* **428**, 1725–1739 (2008)
755. Parlett, B.N., Taylor, D.R., Liu, Z.A.: A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comput.* **44**(169), 105–124 (1985)
756. Philippe, B., Reichel, L.: On the generation of Krylov subspace bases. *Appl. Numer. Math.* **62**(9), 1171–1186 (2012)
757. Pozza, S., Pranić, M.S., Strakoš, Z.: Gauss quadrature for quasi-definite linear functionals. *IMA J. Numer. Anal.* **37**, 1468–1495 (2017)

758. Pranić, M.S., Reichel, L., Rodriguez, G., Wang, Z., Yu, X.: A rational Arnoldi process with applications. *Numer. Linear Algebra Appl.* **23**, 1007–1022 (2016)
759. Pursell, L., Trimble, S.Y.: Gram-Schmidt orthogonalization by Gauss elimination. *Am. Math. Mon.* **98**(6), 544–549 (1991)
760. Radicati di Brozolo, G., Robert, Y.: Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor. *Parallel Comput.* **11**(2), 223–239 (1989)
761. Rehman, R., Ipsen, I.C.F.: Computing characteristic polynomials from eigenvalues. *SIAM J. Matrix Anal. Appl.* **32**(1), 90–114 (2011)
762. Rehman, R., Ipsen, I.C.F.: La Budde’s method for computing characteristic polynomials. arXiv preprint, [arXiv:1104.3769v1](https://arxiv.org/abs/1104.3769v1) (2011)
763. Reichel, L.: Newton interpolation at Leja points. *BIT* **30**, 332–346 (1990)
764. Reichel, L., Ye, Q.: Breakdown-free GMRES for singular systems. *SIAM J. Matrix Anal. Appl.* **26**(4), 1001–1021 (2005)
765. Reichel, L., Yu, X.: Tikhonov regularization via flexible Arnoldi reduction. *BIT* **55**, 1145–1168 (2015)
766. Reid, J.K.: On the method of conjugate gradients for the solution of large sparse systems of linear equations. In: Reid, J.K. (ed.) *Large Sparse Sets of Linear Equations*, pp. 231–254. Academic Press (1971)
767. Rendel, O., Rizvanolli, A., Zemke, J.P.M.: IDR: a new generation of Krylov subspace methods? *Linear Algebra Appl.* **439**(4), 1040–1061 (2013)
768. Ressel, K.J., Gutknecht, M.H.: QMR smoothing for Lanczos-type product methods based on three-term recurrences. *SIAM J. Sci. Comput.* **19**(1), 55–73 (1998)
769. Rice, J.R.: Experiments on Gram-Schmidt orthogonalization. *Math. Comput.* **20**(94), 325–328 (1966)
770. Rigal, J.L., Gaches, J.: On the compatibility of a given solution with the data of a linear system. *J. ACM* **14**, 543–548 (1967)
771. Robbé, M., Sadkane, M.: Exact and inexact breakdowns in the block GMRES method. *Linear Algebra Appl.* **419**(1), 265–285 (2006)
772. Röllin, S., Fichtner, W.: Improving the accuracy of GMRES with deflated restarting. *SIAM J. Sci. Comput.* **30**(1), 232–245 (2007)
773. Röllin, S., Gutknecht, M.H.: Variations of Zhang’s Lanczos-type product method. *Appl. Numer. Math.* **41**(1), 119–133 (2002)
774. Rozložník, M.: Numerical stability of the GMRES method. Ph.D. thesis, Czech Technical University, Prague (1996)
775. Rozložník, M., Strakoš, Z.: On the implementation of some residual minimizing Krylov space methods. In: *SOFSEM’95: Theory and Practice of Informatics*, pp. 449–454. Springer, Berlin, Heidelberg (1995)
776. Rozložník, M., Strakoš, Z.: Variants of residual minimizing Krylov subspace methods. In: Marek, I. (ed.) *Proceedings of the 6th Summer School Software and Algorithms of Numerical Mathematics*, pp. 208–225. University of West Bohemia, Pilsen (1995)
777. Rozložník, M., Strakoš, Z., Tůma, M.: On the role of orthogonality in the GMRES method. In: *SOFSEM’96: Theory and Practice of Informatics*, pp. 409–416. Springer, Berlin, Heidelberg (1996)
778. Rozložník, M., Tůma, M., Smoktunowicz, A., Kopal, J.: Numerical stability of orthogonalization methods with a non-standard inner product. *BIT Numer. Math.* **52**(4), 1035–1058 (2012)
779. Rozložník, M., Weiss, R.: On the stable implementation of the generalized minimal error method. *J. Comput. Appl. Math.* **98**, 49–62 (1998)
780. Ruge, J.W., Stüben, K.: Algebraic multigrid. In: MacCormick, S.F. (ed.) *Multigrid Methods*, pp. 73–130. SIAM (1987)
781. Ruhe, A.: Numerical aspects of Gram-Schmidt orthogonalization of vectors. *Linear algebra Appl.* **52**, 591–601 (1983)

782. Ruhe, A.: Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra Appl.* **58**, 391–405 (1984)
783. Ruhe, A.: The rational Krylov algorithm for nonsymmetric eigenvalue problems. III: complex shifts for real matrices. *BIT Numer. Math.* **34**(1), 165–176 (1994)
784. Ruhe, A.: Rational Krylov algorithms for nonsymmetric eigenvalue problems. II. Matrix pairs. *Linear Algebra Appl.* **197**, 283–295 (1994)
785. Ruhe, A.: Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils. *SIAM J. Sci. Comput.* **19**(5), 1535–1551 (1998)
786. Rutishauser, H.: Beiträge zur kenntnis des biorthogonalisierungs-algorithmus von Lanczos. *Z. Angew. Math. Phys.* **4**(1), 35–56 (1953)
787. Rutishauser, H.: Handbook for Automatic Computation: A Description of Algol 60. Springer (1967)
788. Saad, Y.: Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.* **34**, 269–295 (1980)
789. Saad, Y.: Krylov subspace methods for solving large nonsymmetric linear systems. *Math. Comput.* **37**, 105–126 (1981)
790. Saad, Y.: The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. *SIAM J. Numer. Anal.* **19**, 470–484 (1982)
791. Saad, Y.: Projection methods for solving large sparse eigenvalue problems. In: *Matrix Pencils*, Lecture Notes in Mathematics LNM 973, pp. 121–144. Springer (1983)
792. Saad, Y.: Practical use of some Krylov subspace methods for solving indefinite and non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.* **5**(1), 203–228 (1984)
793. Saad, Y.: Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems. *SIAM J. Numer. Anal.* **24**(1), 155–169 (1987)
794. Saad, Y.: *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press (1992). Revised edition, Classics in Applied Mathematics, SIAM (2011)
795. Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. *J. Sci. Comput.* **14**(2), 461–469 (1993)
796. Saad, Y.: Analysis of augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **18**(2), 435–449 (1997)
797. Saad, Y.: Further analysis of minimum residual iterations. *Numer. Linear Algebra Appl.* **7**(2), 67–93 (2000)
798. Saad, Y.: *Iterative Methods for Sparse Linear Systems*, 2nd edn. SIAM (2003)
799. Saad, Y., Schultz, M.H.: Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comput.* **44**(170), 417–424 (1985)
800. Saad, Y., Schultz, M.H.: GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. Tech. Rep. YALEU/DCS/RR-254, Yale University, USA (1985)
801. Saad, Y., Schultz, M.H.: GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **7**(3), 856–869 (1986)
802. Saad, Y., van der Vorst, H.A.: Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.* **123**(1), 1–33 (2000)
803. Saad, Y., Wu, K.: DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. *Numer. Linear Algebra Appl.* **3**(4), 329–343 (1996)
804. Saad, Y., Yeung, M., Erhel, J., Guyomarc'h, F.: A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.* **21**(5), 1909–1926 (2000)
805. Sadok, H.: CMRH: a new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm. *Numer. Algorithms* **20**(4), 303–321 (1999)
806. Sadok, H.: Analysis of the convergence of the minimal and the orthogonal residual methods. *Numer. Algorithms* **40**, 201–216 (2005)
807. Sadok, H., Szyld, D.B.: A new look at CMRH and its relation to GMRES. *BIT Numer. Math.* **52**(2), 485–501 (2012)
808. Saibel, E., Berger, W.J.: On finding the characteristic equation of a square matrix. *Math. Tables Other Aids Comput.* **7**(44), 228–236 (1953)

809. Saito, S., Tadano, H., Imakura, A.: Development of the block BiCGSTAB(ℓ) method for solving linear systems with multiple right hand sides. *JSIAM Lett.* **6**, 65–68 (2014)
810. Samuelson, P.A.: A method of determining explicitly the coefficients of the characteristic equation. *Ann. Math. Statist.* **13**(4), 424–429 (1942)
811. Sanan, P., Schnepf, S.M., May, D.A.: Pipelined, flexible Krylov subspace methods. *SIAM J. Sci. Comput.* **38**(5), C441–C470 (2016)
812. Saunders, M.A., Simon, H.D., Yip, E.L.: Two conjugate-gradient-type methods for unsymmetric linear equations. *SIAM J. Numer. Anal.* **25**(4), 927–940 (1988)
813. Sauren, M., Bücker, H.M.: On deriving the quasi-minimal residual method. *SIAM Rev.* **40**(4), 922–926 (1998)
814. Saylor, P.E.: Use of the singular value decomposition with the Manteuffel algorithm for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **1**(2), 210–222 (1980)
815. Saylor, P.E., Smolarski, D.C.: Addendum to: why Gaussian quadrature in the complex plane? *Numer. Algorithms* **27**, 215–217 (2001)
816. Saylor, P.E., Smolarski, D.C.: Why Gaussian quadrature in the complex plane? *Numer. Algorithms* **26**, 251–280 (2001)
817. Schneider, O.: Krylov subspace methods and their generalizations for solving singular linear operator equations with applications to continuous time Markov chains. Ph.D. thesis, Technischen Universität Bergakademie Freiberg, Freiberg, Germany (2006)
818. Schönauer, W.: Scientific Computing on Vector Computers, Special Topics in Supercomputing, vol. 2. North Holland (1987)
819. Schweitzer, M.: Any finite convergence curve is possible in the initial iterations of restarted FOM. *Electron. Trans. Numer. Anal.* **45**, 133–145 (2016)
820. Scott, D.S.: How to make the Lanczos algorithm converge slowly. *Math. Comput.* **33**(145), 239–247 (1979)
821. Sekimoto, T., Fujino, S.: A proposal of variants of BiCGSafe method for solving linear systems in realistic problems. In: Proceedings of the World Congress on Engineering 2012 Vol I WCE (2012)
822. Shaidurov, V.V.: Multigrid Methods for Finite Elements, vol. 318. Springer Science & Business Media (2013)
823. Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding in one element of a given matrix. *Ann. Math. Statist.* **21**(1), 124–127 (1950)
824. Sidje, R.B.: Alternatives for parallel Krylov subspace basis computation. *Numer. Linear Algebra Appl.* **4**(4), 305–331 (1997)
825. Simoncini, V.: Ritz and pseudo-Ritz values using matrix polynomials. *Linear Algebra Appl.* **241**, 787–801 (1996)
826. Simoncini, V.: A stabilized QMR version of block BICG. *SIAM J. Matrix Anal. Appl.* **18**, 419–434 (1997)
827. Simoncini, V.: A new variant of restarted GMRES. *Numer. Linear Algebra Appl.* **6**, 61–77 (1999)
828. Simoncini, V.: On the convergence of restarted Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **22**(2), 430–452 (2000)
829. Simoncini, V.: On a non-stagnation condition for GMRES and application to saddle point matrices. *Electron. Trans. Numer. Anal.* **37**, 202–213 (2010)
830. Simoncini, V., Gallopoulos, E.: An iterative method for nonsymmetric systems with multiple right-hand sides. *SIAM J. Sci. Comput.* **16**(4), 917–933 (1995)
831. Simoncini, V., Gallopoulos, E.: Convergence properties of block GMRES and matrix polynomials. *Linear Algebra Appl.* **247**, 97–119 (1996)
832. Simoncini, V., Gallopoulos, E.: A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides. *J. Comput. Appl. Math.* **66**(1–2), 457–469 (1996)
833. Simoncini, V., Szyld, D.B.: Flexible inner-outer Krylov subspace methods. *SIAM J. Numer. Anal.* **40**(6), 2219–2239 (2003)
834. Simoncini, V., Szyld, D.B.: Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.* **25**(2), 454–477 (2003)

835. Simoncini, V., Szyld, D.B.: The effect of non-optimal bases on the convergence of Krylov subspace methods. *Numer. Math.* **100**, 711–733 (2005)
836. Simoncini, V., Szyld, D.B.: On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Rev.* **74**(21), 247–272 (2005)
837. Simoncini, V., Szyld, D.B.: Recent computational developments in Krylov subspace methods for linear systems. *Numer. Linear Algebra Appl.* **14**(1), 1–59 (2007)
838. Simoncini, V., Szyld, D.B.: New conditions for non-stagnation of minimal residual methods. *Numer. Math.* **109**, 477–487 (2008)
839. Simoncini, V., Szyld, D.B.: Interpreting IDR as a Petrov-Galerkin method. *SIAM J. Sci. Comput.* **32**(4), 1898–1912 (2010)
840. Simoncini, V., Szyld, D.B.: On the superlinear convergence of MINRES. In: *Numerical Mathematics and Advanced Applications 2011*, pp. 733–740. Springer, Berlin, Heidelberg (2013)
841. Sleijpen, G.L.G., Fokkema, D.R.: BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.* **11**, 11–32 (1993)
842. Sleijpen, G.L.G., Sonneveld, P., van Gijzen, M.B.: Bi-CGSTAB as an induced dimension reduction method. *Appl. Numer. Math.* **60**(11), 1100–1114 (2010)
843. Sleijpen, G.L.G., van den Eshof, J., Smit, P.: Optimal a priori error bounds for the Rayleigh-Ritz method. *Math. Comput.* **72**(242), 677–684 (2003)
844. Sleijpen, G.L.G., van den Eshof, J., van Gijzen, M.B.: Restarted GMRES with inexact matrix-vector products. In: *International Conference on Numerical Analysis and its Applications*, pp. 494–502. Springer, Berlin, Heidelberg (2004)
845. Sleijpen, G.L.G., van der Vorst, H.: Krylov subspace methods for large linear systems of equations. Tech. Rep. 803, Mathematical Institute, University of Utrecht, Netherlands (1993)
846. Sleijpen, G.L.G., van der Vorst, H.: Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. *Numer. Algorithms* **10**, 203–223 (1995)
847. Sleijpen, G.L.G., van der Vorst, H.: An overview of approaches for the stable computation of hybrid BiCG methods. *Appl. Numer. Math.* **19**(3), 235–254 (1995)
848. Sleijpen, G.L.G., van der Vorst, H.: Reliable updated residuals in hybrid Bi-CG methods. *Computing* **56**(2), 141–163 (1996)
849. Sleijpen, G.L.G., van der Vorst, H., Fokkema, D.R.: BiCGstab(ℓ) and other hybrid Bi-CG methods. *Numer. Algorithms* **7**(1), 75–109 (1994)
850. Sleijpen, G.L.G., van Gijzen, M.B.: Exploiting BiCGstab(ℓ) strategies to induce dimension reduction. *SIAM J. Sci. Comput.* **32**, 2687–2709 (2010)
851. Smit, P.: Generating identical Ritz values. Tilburg University, The Netherlands, Tech. rep. (1995)
852. Smith, C.F., Peterson, A.F., Mittra, R.: A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields. *IEEE Trans. Antennas Propag.* **37**, 1490–1493 (1989)
853. Smoch, L.: Application des méthodes de Krylov à la résolution de systèmes singuliers. Ph.D. thesis, Université du Littoral, Calais, France (1999). (in French)
854. Smoch, L.: Some results about GMRES in the singular case. *Numer. Algorithms* **22**(2), 193–212 (1999)
855. Smoch, L.: Spectral behaviour of GMRES applied to singular systems. *Adv. Comput. Math.* **27**(2), 151–166 (2007)
856. Smoktunowicz, A., Barlow, J.L., Langou, J.: A note on the error analysis of classical Gram-Schmidt. *Numer. Math.* **105**(2), 299–313 (2006)
857. Smolarski, D.C., Saylor, P.E.: An optimum iterative method for solving any linear system with a square matrix. *BIT Numer. Math.* **28**(1), 163–178 (1988)
858. Smolarski, D.C., Saylor, P.E.: Implementation of an adaptive algorithm for Richardson's method. *Linear Algebra Appl.* **154**, 615–646 (1991)
859. Sogabe, T., Sugihara, M., Zhang, S.L.: An extension of the conjugate residual method to nonsymmetric linear systems. *J. Comput. Appl. Math.* **226**, 103–113 (2009)

860. Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. Tech. Rep. 84-16, Department of Mathematics and Informatics, Delft University of Technology, The Netherlands (1984)
861. Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **10**(1), 36–52 (1989)
862. Sonneveld, P.: AGS-IDR-CGS-BiCGSTAB-IDR(s): the circle closed. A case of serendipity. In: Proceedings of the International Kyoto Forum 2008 on Krylov subspace methods, pp. 1–14 (2008)
863. Sonneveld, P.: On the convergence behavior of IDR(s) and related methods. *SIAM J. Sci. Comput.* **34**(5), A2576–A2598 (2012)
864. Sonneveld, P.: A history of Krylov product methods - A case of serendipity -. Tech. Rep. 17-05, Institute of Applied Mathematics, Delft University of Technology, The Netherlands (2017)
865. Sonneveld, P., van Gijzen, M.B.: 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)
866. Soodhalter, K.M.: Block Krylov subspace recycling for shifted systems with unrelated right-hand sides. *SIAM J. Sci. Comput.* **38**(1), A302–A324 (2016)
867. Soodhalter, K.M.: Two recursive GMRES-type methods for shifted linear systems with general preconditioning. *Electron. Trans. Numer. Anal.* **45**, 499–523 (2016)
868. Soodhalter, K.M.: Stagnation of block GMRES and its relationship to block FOM. *Electron. Trans. Numer. Anal.* **46**, 162–189 (2017)
869. Soodhalter, K.M., Szyld, D.B., Xue, F.: Krylov subspace recycling for sequences of shifted linear systems. *Appl. Numer. Math.* **81**, 105–118 (2014)
870. Sorensen, D.C.: Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.* **13**(1), 357–385 (1992)
871. Sosonkina, M., Watson, L.T., Kapania, R.K., Walker, H.F.: A new adaptive GMRES algorithm for achieving high accuracy. *Numer. Linear Algebra Appl.* **5**, 275–297 (1998)
872. Spyropoulos, A.N., Boudouvis, A.G., Markatos, N.C.: An implementation of parallel preconditioned GMRES(m) on distributed memory computers. In: Computational Fluid Dynamics'98, pp. 478–483 (1998)
873. Staib, J.H.: An alternative to the Gram-Schmidt process. *Math. Mag.* **42**(4), 203–205 (1969)
874. Starke, G.: Field-of-values analysis of preconditioned iterative methods for nonsymmetric elliptic problems. *Numer. Math.* **78**(1), 103–117 (1997)
875. Starke, G., Varga, R.S.: A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations. *Numer. Math.* **64**(1), 213–240 (1993)
876. Stathopoulos, A., Wu, K.: A block orthogonalization procedure with constant synchronization requirements. *SIAM J. Sci. Comput.* **23**(6), 2165–2182 (2002)
877. Stephens, D., Howell, G.: The elementary residual method. *Contemp. Math.* **275**, 107–116 (2001)
878. Stewart, G.W.: Conjugate direction methods for solving systems of linear equations. *Numer. Math.* **21**(4), 285–297 (1973)
879. Stewart, G.W.: The triangular matrices of Gaussian elimination and related decompositions. *IMA J. Numer. Anal.* **17**(1), 7–16 (1997)
880. Stewart, G.W.: Matrix Algorithms, Volume I: Basic Decompositions. SIAM, Philadelphia (1998)
881. Stewart, G.W.: A generalization of Saad's theorem on Rayleigh-Ritz approximations. *Linear Algebra Appl.* **327**, 115–119 (2001)
882. Stewart, G.W.: Matrix Algorithms, Volume II: Eigensystems. SIAM, Philadelphia (2001)
883. Stewart, G.W.: Backward error bounds for approximate Krylov subspaces. *Linear Algebra Appl.* **340**, 81–86 (2002)
884. Stewart, G.W.: The Gram-Schmidt algorithm and its variations. Tech. Rep. TR-2004-84, Department of Mathematics, University of Maryland (2004)

885. Stewart, G.W.: Block Gram-Schmidt orthogonalization. *SIAM J. Sci. Comput.* **31**(1), 761–775 (2008)
886. Stewart, G.W.: On the semidefinite B-Arnoldi method. *SIAM J. Matrix Anal. Appl.* **31**(3), 1458–1468 (2009)
887. Stewart, G.W.: On the numerical analysis of oblique projectors. *SIAM J. Matrix Anal. Appl.* **32**(1), 309–348 (2011)
888. Stiefel, E.: Relaxationsmethoden bester strategie zur lösung linearer gleichungssysteme. *Comment. Math. Helv.* **29**(1), 157–179 (1955)
889. Stieltjes, T.J.: Quelques recherches sur la théorie des quadratures dites mécaniques. *Ann. Sci. Ecole Norm. Sup. Paris Sér. 3 tome 1*, 409–426 (1884). Also in *Oeuvres complètes*, vol. 1, pp. 377–396. Noordhoff, Groningen (1914)
890. Strakoš, Z.: On the real convergence rate of the conjugate gradient method. *Linear Algebra Appl.* **154–156**, 535–549 (1991)
891. Strakoš, Z.: Model reduction using the Vorobyev moment problem. *Numer. Algorithms* **51**(3), 363–379 (2008)
892. Strakoš, Z., Liesen, J.: On numerical stability in large scale linear algebraic computations. *Zamm.-Z. Aengew. Math. Me.* **85**(5), 307–325 (2005)
893. Strakoš, Z., Tichý, P.: On error estimates in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.* **13**, 56–80 (2002)
894. Strakoš, Z., Tichý, P.: Error estimation in preconditioned conjugate gradients. *BIT Numer. Math.* **45**, 789–817 (2005)
895. Strakoš, Z., Tichý, P.: On efficient numerical approximation of the bilinear form $c^* A^{-1} b$. *SIAM J. Sci. Comput.* **32**(2), 565–587 (2008)
896. Strikwerda, J.C., Stodder, S.C.: Convergence results for GMRES(m). Tech. Rep. 1280, Department of Computer Sciences, University of Wisconsin, USA (1995)
897. Sun, D.L., Huang, T.Z., Jing, Y.F., Carpentieri, B.: A block GMRES method with deflated restarting for solving linear systems with multiple shifts and multiple right-hand sides. *Numer. Linear Algebra Appl.* **25**(5) (2018)
898. Sun, D.L., Jing, Y.F., Carpentieri, B., Huang, T.Z.: Flexible and deflated variants of the block shifted GMRES method. *J. Comput. Appl. Math.* **345**, 168–183 (2019)
899. Sun, D.L., Jing, Y.F., Huang, T.Z., Carpentieri, B.: A quasi-minimal residual variant of the BiCORSTAB method for nonsymmetric linear systems. *Comput. Math. Appl.* **67**(10), 1743–1755 (2014)
900. Swanson, C.D., Chronopoulos, A.T.: Orthogonal s-step methods for nonsymmetric linear systems of equations. In: Proceedings of the 6th International Conference on Supercomputing, pp. 456–462. ACM (1992)
901. Swirydowicz, K., Langou, J., Ananthan, S., Yang, U., Thomas, S.: Low synchronization Gram-Schmidt and GMRES algorithms. arXiv preprint, [arXiv:1809.05805v1](https://arxiv.org/abs/1809.05805v1), to be published in Numerical Linear Algebra with Applications (2018)
902. Szász, O.: Az Hadamard-féle determinánstétel egy elemi bebizonyítása. *Mathematicai és Physikai Lapok* **19**, 221–227 (1910)
903. Szyld, D.B., Vogel, J.A.: FQMR: a flexible quasi-minimal residual method with inexact preconditioning. *SIAM J. Sci. Comput.* **23**(2), 363–380 (2001)
904. Tal-Ezer, H.: Polynomial approximation of functions of matrices and applications. *J. Sci. Comput.* **4**(1), 25–60 (1989)
905. Tanio, M., Sugihara, M.: GBi-CGSTAB(s, L): IDR(s) with higher-order stabilization polynomials. *J. Comput. Appl. Math.* **235**(3), 765–784 (2010)
906. Taussky, O., Zassenhaus, H.: On the similarity transformation between a matrix and its transpose. *Pac. J. Math.* **9**(3), 893–896 (1959)
907. Taylor, D.R.: Analysis of the look ahead Lanczos algorithm. Ph.D. thesis, University of California, Berkeley (1982)
908. Teng, Z., Wang, X.: Heavy ball restarted CMRH methods for linear systems. *Math. Comput. Appl.* **23**(1), 1–10 (2018)

909. Thompson, R.C.: Principal submatrices of normal and Hermitian matrices. *Ill. J. Math.* **10**(2), 296–308 (1966)
910. Tichý, P.: The shadow vector in the Lanczos method. In: Proceedings of the XIIIth Summer School Software and Algorithms of Numerical Mathematics Nectiny, pp. 309–320 (1999)
911. Tichý, P., Liesen, J., Faber, V.: On worst-case GMRES, ideal GMRES, and the polynomial numerical hull of a Jordan block. *Electron. Trans. Numer. Anal.* **26**, 453–473 (2007)
912. Tichý, P., Zítko, J.: Derivation of BiCG from the conditions defining Lanczos method for solving a system of linear equations. *Appl. Math.* **43**(5), 381–388 (1998)
913. Tisseur, F.: Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **22**(4), 1038–1057 (2001)
914. Titley-Peloquin, D., Pestana, J., Wathen, A.J.: GMRES convergence bounds that depend on the right-hand-side vector. *IMA J. Numer. Anal.* **34**(2), 462–479 (2014)
915. Toh, K.C.: GMRES vs. ideal GMRES. *SIAM J. Matrix Anal. Appl.* **18**(1), 30–36 (1997)
916. Toh, K.C., Todd, M.J., Tutuncu, R.H.: SDPT3 - a Matlab software package for semidefinite programming. *Optim. Methods Softw.* **11**, 545–581 (1999)
917. Toh, K.C., Trefethen, L.N.: The Chebyshev polynomials of a matrix. *SIAM J. Matrix Anal. Appl.* **20**, 400–419 (1998)
918. Tong, C.H.: A comparative study of preconditioned Lanczos methods for nonsymmetric linear system. Tech. Rep. SAND-91-8240B, Sandia National Laboratories, Livermore (1992)
919. Tong, C.H.: A family of quasi-minimal residual methods for nonsymmetric linear systems. *SIAM J. Sci. Comput.* **15**(1), 89–105 (1994)
920. Tong, C.H., Ye, Q.: A linear system solver based on a modified Krylov subspace method for breakdown recovery. *Numer. Algorithms* **12**(1), 233–251 (1996)
921. Tong, C.H., Ye, Q.: Analysis of the finite precision bi-conjugate gradient algorithm for nonsymmetric linear systems. *Math. Comput.* **69**(232), 1559–1575 (2000)
922. Torenbeek, R., Vuik, K.: The Arnoldi and Lanczos methods for approximating the eigenpairs of a matrix. Tech. Rep. 96-44, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1996)
923. Traviesas, E.: Sur le déploielement du champ spectral d'une matrice. Ph.D. thesis, University of Toulouse 1, France (2000). (in French)
924. Trefethen, L.N.: Approximation theory and numerical linear algebra. In: Mason, J.C., Cox, M.G. (eds.) *Algorithms for Approximation II*. Chapman and Hall, London (1990)
925. Trefethen, L.N.: Computation of pseudospectra. *Acta Numer.* **247–295**, (1999)
926. Trefethen, L.N., Embree, M.: *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press (2005)
927. Trottenberg, U., Oosterlee, C.W., Schuller, A.: *Multigrid*. Elsevier (2000)
928. Tsuno, N., Nodera, T.: The speedup of the GMRES(m) method using the early restarting procedure. *Trans. Inform. Process. Soc. Jpn.* **40**, 1760–1773 (1999)
929. Turing, A.M.: Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.* **1**, 287–308 (1948)
930. Turnbull, H.W., Aitken, A.C.: An Introduction to the Theory of Canonical Matrices. Blackie and Son (1932). Reprinted by Dover (2004)
931. Tyrtyshnikov, E.E.: How bad are Hankel matrices? *Numer. Math.* **67**, 261–269 (1994)
932. Uhlig, F.: Are the coefficients of a polynomial well-conditioned functions of its roots? *Numer. Math.* **61**(1), 383–393 (1992)
933. Uhlig, F.: An inverse field of values problem. *Inverse Prob.* **24**(5), 1–19 (2008)
934. Underwood, R.: An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. Tech. Rep. 496, Computer Science Department, Stanford University, USA (1975)
935. Van Barel, M., Vandebril, R., Van Dooren, P., Frederix, K.: Implicit double shift QR-algorithm for companion matrices. *Numer. Math.* **116**(2), 177–212 (2010)
936. van den Eshof, J., Sleijpen, G.L.G.: Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.* **26**(1), 125–153 (2004)

937. van den Eshof, J., Sleijpen, G.L.G., van Gijzen, M.B.: Relaxation strategies for nested Krylov methods. *J. Comput. Appl. Math.* **177**(2), 347–365 (2005)
938. van der Vorst, H.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. Department of Mathematics, University of Utrecht, The Netherlands, Tech. rep. (1990)
939. van der Vorst, H.: The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors. In: Axelsson, O., Kolotilina, L.Y. (eds.) *Preconditioned Conjugate Gradient Methods*, Lecture Notes in Mathematics, vol. 1457. Springer (1990)
940. van der Vorst, H.: BICGSTAB: a fast and smoothly converging variant of BI-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.* **12**(2), 631–644 (1992)
941. van der Vorst, H.: *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press (2003)
942. van der Vorst, H., Sonneveld, P.: CGSTAB, a more smoothly converging variant of CG-S. Department of Mathematics and Informatics, Delft University of Technology, Tech. rep. (1990)
943. van der Vorst, H., Vuik, C.: The superlinear convergence behavior of GMRES. *J. Comput. Appl. Math.* **48**, 327–341 (1993)
944. van der Vorst, H., Vuik, C.: GMRESR: a family of nested GMRES methods. *Numer. Linear Algebra Appl.* **1**(4), 369–386 (1994)
945. van der Vorst, H., Ye, Q.: Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. *SIAM J. Sci. Comput.* **22**(3), 835–852 (2000)
946. van Gijzen, M.B., Sleijpen, G.L.G., Zemke, J.P.M.: Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems. *Numer. Linear Algebra Appl.* **22**(1), 1–25 (2015)
947. van Gijzen, M.B., Sonneveld, P.: The IDR(s) method for solving nonsymmetric systems: a performance study for CFD problems. In: *High Performance Algorithms for Computational Science and Their Applications*, RIMS 2007 (2008)
948. van Gijzen, M.B., Sonneveld, P.: Algorithm 913: An elegant IDR(s) variant that efficiently exploits biorthogonality properties. *ACM Trans. Math. Soft. (TOMS)* **38**(1), 5.1–5.19 (2011)
949. van Rossum, H.: *A Theory of Orthogonal Polynomials Based on the Padé Table*, vol. 4. Van Gorcum (1953)
950. Varga, R.S.: *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N.J. (1962)
951. Vecharynski, E., Langou, J.: The cycle-convergence of restarted GMRES for normal matrices is sublinear. *SIAM J. Sci. Comput.* **32**(1), 186–196 (2010)
952. Vecharynski, E., Langou, J.: Any admissible cycle-convergence behavior is possible for restarted GMRES at its initial cycles. *Numer. Linear Algebra Appl.* **18**(3), 499–511 (2011)
953. Vinsome, P.K.W.: Orthomin, an iterative method for solving sparse sets of simultaneous linear equations. In: *Proceedings of the Fourth SPE Symposium on Numerical Simulation of Reservoir Performance*. Society of Petroleum Engineers of AIME (1976)
954. Vital, B.: Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur. Ph.D. thesis, Université de Rennes I, Rennes, France (1990). (in French)
955. Vogel, J.A.: Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems. *Appl. Math. Comput.* **188**(1), 226–233 (2007)
956. von Neumann, J., Goldstine, H.H.: Numerical inverting of matrices of high order. *Bull. Am. Math. Soc.* **53**, 1021–1099 (1947)
957. Vorobyev, Y.V.: *Methods of Moments in Applied Mathematics*. Gordon and Breach Science Publishers, New York (1965)
958. Vuik, C.: New insights in GMRES-like methods with variable preconditioners. *J. Comput. Appl. Math.* **61**(2), 189–204 (1995)
959. Vuik, C., Segal, A., Meijerink, J.A.: An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. *J. Comp. Phys.* **152**(1), 385–403 (1999)

960. Vuik, C., Sevink, A.G., Herman, G.C.: A preconditioned Krylov subspace method for the solution of least squares problems in inverse scattering. *J. Comp. Phys.* **123**(2), 330–340 (1996)
961. Vuik, C., van der Vorst, H.: A comparison of some GMRES-like methods. *Linear Algebra Appl.* **160**, 131–162 (1992)
962. Wakam, D.N., Erhel, J.: Parallelism and robustness in GMRES with a Newton basis and deflated restarting. *Electron. Trans. Numer. Anal.* **40**, 381–406 (2013)
963. Wakam, D.N., Erhel, J., Gropp, W.D.: Parallel adaptive deflated GMRES. In: *Domain Decomposition Methods in Science and Engineering XX*, pp. 631–638. Springer, Berlin, Heidelberg (2013)
964. Walker, H.F.: Implementation of the GMRES and Arnoldi methods using Householder transformations. *Tech. Rep. UCRL-93589*, Lawrence Livermore Laboratory (1985)
965. Walker, H.F.: Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Statist. Comput.* **9**(1), 152–163 (1988)
966. Walker, H.F.: Implementations of the GMRES method. *Comput. Phys. Comm.* **53**, 311–320 (1989)
967. Walker, H.F.: Residual smoothing and peak/plateau behavior in Krylov subspace methods. *Appl. Numer. Math.* **19**(3), 279–286 (1995)
968. Walker, H.F., Ni, P.: Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.* **49**(4), 1715–1735 (2011)
969. Walker, H.F., Zhou, L.: A simpler GMRES. *Numer. Linear Algebra Appl.* **1**(6), 571–581 (1994)
970. Wallis, J.R., Kendall, R.P., Little, T.E.: Constrained residual acceleration of conjugate residual methods. In: *SPE 13536, Reservoir Simulation Symposium*, Austin Texas. Society of Petroleum Engineers (1985)
971. Walsh, P.J.: Algorithm 127: Ortho. Commun. *ACM* **5**(10), 511–513 (1962)
972. Wang, J.M., Gu, T.X.: Flexible GPBi-CG method for nonsymmetric linear systems. *Appl. Math.* **3**(4), 331 (2012)
973. Wang, L.P., Yuan, J.: Conjugate decomposition and its applications. *J. Oper. Res. Soc. China* **1**(2), 199–215 (2013)
974. Wang, L.P., Zhu, Y.: Hybrid methods based on LCG and GMRES. *Comput. Appl. Math.* **35**(1), 301–319 (2016)
975. Watts, J.W.: A method for improving line successive overrelaxation in anisotropic problems—a theoretical analysis. *Soc. Pet. Eng. J.* **13**(2), 105–118 (1973)
976. Wei, Y.M., Wu, H.: Convergence properties of Krylov subspace methods for singular linear systems with arbitrary index. *J. Comput. Appl. Math.* **114**, 305–318 (2000)
977. Weiss, R.: Convergence behavior of generalized conjugate gradient methods. Ph.D. thesis, University of Karlsruhe, Germany (1990)
978. Weiss, R.: Error-minimizing Krylov subspace methods. *SIAM J. Sci. Comput.* **15**(3), 511–527 (1994)
979. Weiss, R.: Properties of generalized conjugate gradient methods. *Numer. Linear Algebra Appl.* **1**(1), 45–63 (1994)
980. Weiss, R.: A theoretical overview of Krylov subspace methods. *Appl. Numer. Math.* **19**(3), 207–233 (1995)
981. Wesseling, P., Sonneveld, P.: Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In: Rautmann, R. (ed.) *Approximation Methods for Navier-Stokes Problems*, Paderborn, Germany 1979, Lecture Notes in Mathematics. Springer (1980)
982. Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Oxford University Press (1965)
983. Włodkowiski, P.: Taming Poseidon's beast of uncertainty: the auspicious debut of marine engineer Aleksej Nikolaevich Krylov (1863–1945). *Int. J. Hist. Eng. Technol.* **85**(1), 140–158 (2015)
984. Wong, Y.K.: An application of orthogonalization process to the theory of least squares. *Ann. Math. Statist.* **6**(2), 53–75 (1935)

985. Woodbury, M.A.: Inverting modified matrices. Tech. Rep. Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, USA (1950)
986. Wu, G.: The convergence of harmonic Ritz vectors and harmonic Ritz values, revisited. *SIAM J. Matrix Anal. Appl.* **38**(1), 118–133 (2017)
987. Wu, G., Wei, Y., Jia, Z.G., Ling, S.T., Zhang, L.: Towards backward perturbation bounds for approximate dual Krylov subspaces. *BIT Numer. Math.* **53**(1), 225–239 (2013)
988. Wu, K.: Restarted variants of DQGMRES. In: Proceedings of 1997 International Workshop on Computational Science and Engineering (1997)
989. Wu, K., Simon, H.: Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **22**(2), 602–616 (2000)
990. Wu, Q., Bao, L., Lin, Y.: Residual-based simpler block GMRES for nonsymmetric linear systems with multiple right-hand sides. *Adv. Math. Phys.* ID **1369707**, (2018)
991. Wynn, P.: A general system of orthogonal polynomials. *Quart. J. Math. Oxford* **2**(18), 81–96 (1967)
992. Xie, G.: On convergence bounds of GMRES algorithm. In: Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003, pp. 750–753. IEEE (2003)
993. Xu, H.: Functions of a matrix and Krylov matrices. *Linear Algebra Appl.* **434**(1), 185–200 (2011)
994. Xu, J., Cai, X.C.: A preconditioned GMRES method for nonsymmetric or indefinite problems. *Math. Comput.* **59**(200), 311–319 (1992)
995. Yamamoto, Y., Nakatsukasa, Y., Yanagisawa, Y., Fukaya, T.: Roundoff error analysis of the CholeskyQR2 algorithm. *Electron. Trans. Numer. Anal.* **44**, 306–326 (2015)
996. Yamazaki, I., Abdelfattah, A., Ida, A., Ohshima, S., Tomov, S., Yokota, R., Dongarra, J.: Analyzing performance of BiCGStab with hierarchical matrix on GPU clusters. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE (2018)
997. Yamazaki, I., Anzt, H., Tomov, S., Hoemmen, M., Dongarra, J.: Improving the performance of CA-GMRES on multicores with multiple GPUs. In: Parallel and Distributed Processing IEEE symposium, pp. 382–391. IEEE (2014)
998. Yamazaki, I., Hoemmen, M., Luszczek, P., Dongarra, J.: Improving performance of GMRES by reducing communication and pipelining global collectives. In: Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1118–1127. IEEE (2017)
999. Yamazaki, I., Tomov, S., Dong, T., Dongarra, J.: Mixed-precision orthogonalization scheme and adaptive step size for improving the stability and performance of CA-GMRES on GPUs. In: International Conference on High Performance Computing for Computational Science, pp. 17–30. Springer (2014)
1000. Yamazaki, I., Tomov, S., Dongarra, J.: Deflation strategies to improve the convergence of communication-avoiding GMRES. In: Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, pp. 39–46. IEEE Press (2014)
1001. Yang, L.T., Brent, R.P.: The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In: Algorithms and Architectures for Parallel Processing, IEEE Conference, pp. 324–328. IEEE (2002)
1002. Yang, L.T., Brent, R.P.: Quantitative performance analysis of the improved quasi-minimal residual method on massively distributed memory computers. *Adv. Eng. Softw.* **33**(3), 169–177 (2002)
1003. Yang, L.T., Brent, R.P.: The improved Krylov subspace methods for large and sparse linear systems on bulk synchronous parallel architectures. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS03). IEEE (2003)
1004. Yang, L.T., Brent, R.P.: The improved parallel BiCG method for large and sparse unsymmetric linear systems on distributed memory architectures. In: Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 349–360. IEEE (2003)
1005. Ye, Q.: A convergence analysis for nonsymmetric Lanczos algorithms. *Math. Comput.* **56**(194), 677–691 (1991)

1006. Ye, Q.: A breakdown-free variation of the nonsymmetric Lanczos algorithms. *Math. Comput.* **62**(205), 179–207 (1994)
1007. Yeung, M.C.: On the solution of singular systems by Krylov subspace methods. In: Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science (DCABES). IEEE (2010)
1008. Yeung, M.C.: ML(n) BiCGSTAB: reformulation, analysis and implementation. *Numer. Math. Theor. Meth. Appl.* **5**(3), 447–492 (2012)
1009. Yeung, M.C.: ML(n)BiCGStab: a ML(n)BiCGStab variant with A-transpose. *J. Comput. Appl. Math.* **272**, 57–69 (2014)
1010. Yeung, M.C., Boley, D.: Transpose-free multiple Lanczos and its application in Padé approximation. *J. Comput. Appl. Math.* **177**(1), 101–127 (2005)
1011. Yeung, M.C., Chan, T.F.: ML(K)BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors. *SIAM J. Sci. Comput.* **21**(4), 1263–1290 (1999)
1012. Yeung, M.C., Tang, J., Vuik, C.: On the convergence of GMRES with invariant-subspace deflation. Tech. Rep. 10-14, Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands (2010)
1013. Young, D.M.: Iterative Solution of Large Linear Systems. Academic Press (1971)
1014. Young, D.M., Jea, K.C.: Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra Appl.* **34**, 159–194 (1980)
1015. Yuan, J.Y., Golub, G.H., Plemmons, R.J., Cecilio, W.A.G.: Semi-conjugate direction methods for real positive definite systems. *BIT Numer. Math.* **44**(1), 189–207 (2004)
1016. Zavorin, I.: Analysis of GMRES convergence by spectral factorization of the Krylov matrix. Ph.D. thesis, University of Maryland, USA (2001)
1017. Zavorin, I., O’Leary, D.P., Elman, H.C.: Complete stagnation of GMRES. *Linear Algebra Appl.* **367**, 165–183 (2003)
1018. Zemke, J.P.M.: Krylov subspace methods in finite precision: a unified approach. Ph.D. thesis, Technical University of Hamburg, Germany (2003)
1019. Zemke, J.P.M.: (Hessenberg) eigenvalue-eigenmatrix relations. *Linear Algebra Appl.* **414**(2–3), 589–606 (2006)
1020. Zemke, J.P.M.: Abstract perturbed Krylov methods. *Linear Algebra Appl.* **424**(2–3), 405–434 (2007)
1021. Zemke, J.P.M.: IDR(s) and IDR(s)Eig in parallel computing. *Supercomputing News, The University of Tokyo* **12**(2), 31–48 (2010)
1022. Zemke, J.P.M.: Variants of IDR with partial orthonormalization. *Electron. Trans. Numer. Anal.* **46**, 245–272 (2017)
1023. Zha, H., Zhang, Z.: The Arnoldi process, short recursions, and displacement ranks. *Linear Algebra Appl.* **249**(1–3), 169–188 (1996)
1024. Zhang, J., Dai, H.: A new quasi-minimal residual method based on a biconjugate A-orthonormalization procedure and coupled two-term recurrences. *Numer. Algorithms* **70**, 875–896 (2015)
1025. Zhang, J., Dai, H., Zhao, J.: Generalized global conjugate gradient squared algorithm. *Appl. Math. Comput.* **216**(12), 3694–3706 (2010)
1026. Zhang, J., Dai, H., Zhao, J.: A new family of global methods for linear systems with multiple right-hand sides. *J. Comput. Appl. Math.* **236**(6), 1562–1575 (2011)
1027. Zhang, K., Gu, C.: A flexible CMRH algorithm for nonsymmetric linear systems. *J. Appl. Math. Comput.* **45**(1–2), 43–61 (2014)
1028. Zhang, L., Nodera, T.: A new adaptive restart for GMRES(m) method. *ANZIAM J.* **46**, 409–425 (2005)
1029. Zhang, S.L.: GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.* **18**(2), 537–551 (1997)
1030. Zhang, S.L.: A class of product-type Krylov-subspace methods for solving nonsymmetric linear systems. *J. Comput. Appl. Math.* **149**(1), 297–305 (2002)
1031. Zhang, W.: GMRES on a tridiagonal Toeplitz linear system. Ph.D. thesis, University of Kentucky (2007)

1032. Zhong, B.: A product hybrid GMRES algorithm for nonsymmetric linear systems. *J. Comput. Math.* **83–92**, (2005)
1033. Zhong, B., Morgan, R.B.: Complementary cycles of restarted GMRES. *Numer. Linear Algebra Appl.* **15**(6), 559–571 (2008)
1034. Zhou, L., Walker, H.F.: Residual smoothing techniques for iterative methods. *SIAM J. Sci. Comput.* **15**, 297–312 (1994)
1035. Zítko, J.: Generalization of convergence conditions for a restarted GMRES. *Numer. Linear Algebra Appl.* **7**, 117–131 (2000)
1036. Zítko, J.: Convergence conditions for a restarted GMRES method augmented with eigenspaces. *Numer. Linear Algebra Appl.* **12**(4), 373–390 (2005)
1037. Zítko, J.: How residual bounds for restarted GMRES describe the real behaviour. In: PAMM: Proceedings in Applied Mathematics and Mechanics, vol. 8. Wiley-VCH Verlag (2008)
1038. Zítko, J.: Some remarks on the restarted and augmented GMRES method. *Electron. Trans. Numer. Anal.* **31**, 221–227 (2008)
1039. Zítko, J., Nádhra, D.: Behaviour of the augmented GMRES method. In: SNA'10, Nové Hrady. Charles University, Prague, Czech Republic (2010)

Index

A

- APS parametrization, 211, 212, 219, 220
Arnoldi-like relation, 54, 65, 67, 92, 99, 103, 127, 137, 139, 206, 341, 416, 419, 424, 425, 460, 468, 499
Arnoldi process, 100–102, 108, 109, 112, 133, 134, 137, 138, 147, 151, 153, 180, 181, 221, 242, 246–250, 253, 259, 263, 269, 311, 467, 468, 497, 498, 506, 520, 521, 535, 584, 585, 595, 601
Arnoldi relation, 77, 100, 252, 271, 274, 311, 500, 506, 509, 525, 537
Augmentation, 516, 518–521, 524

B

- Backward error, 45, 48–50, 276, 325
Backward stability, 24, 45, 167, 268, 274, 277, 278, 325, 349, 392
Breakdown, 115, 118, 119, 129, 149, 169, 171, 179, 307, 320, 327, 355, 357, 366, 367, 372, 374, 376, 377, 379, 380, 383, 385, 386, 389, 392, 411, 415, 424, 426, 433, 434, 463, 476, 519, 536

C

- Cauchy–Binet formula, 16, 86, 87, 89, 224, 233, 235, 236, 239
Cayley–Hamilton theorem, 19, 22, 26
Characteristic polynomial, 17–22, 26, 28–30, 40, 63, 66, 68–71, 73, 75, 85, 93, 232–235, 237–239, 326, 381
Cholesky factorization, 14, 55, 130, 145, 211, 213–215, 321, 360

© Springer Nature Switzerland AG 2020

G. Meurant and J. Duintjer Tebbens, *Krylov Methods for Nonsymmetric Linear Systems*, Springer Series in Computational Mathematics 57,
<https://doi.org/10.1007/978-3-030-55251-0>

Communication-avoiding, 46, 133, 147, 269, 389, 435

Companion matrix, 20–22, 26, 28, 30, 56, 57, 66, 69–71, 73, 77, 81, 82, 140, 205, 206, 209, 211, 267, 347, 379–381, 501, 502

Condition number, 14, 20, 42, 48, 49, 83, 84, 134, 151, 153, 154, 156, 157, 201, 276, 310, 311, 314–316, 321, 325, 347, 391, 478, 504, 602

Convergence curve, 77, 78, 205, 211, 218, 225, 245, 249, 288, 347, 348, 367, 379, 382, 430, 512

Cramer’s rule, 16, 32, 86

Crouzeix’s conjecture, 199

D

Deflation, 516, 518, 519, 533, 534, 545, 584, 585

Diagonalizable matrix, 18, 19, 40, 41, 83, 84, 86, 89, 198, 201–203, 223, 225, 229, 232, 235, 239, 240, 327, 382, 475, 504, 505

Dot product, 14, 24, 44, 46, 101, 102, 111, 115, 124, 131, 133, 145, 147, 153, 273, 304, 306, 319, 345, 352, 355, 357, 370, 374, 385, 391, 413, 415, 430, 432, 434, 435, 464, 502, 532, 585

E

Eigenvalue distribution, 84, 85, 241, 521

Euclidean norm, 14, 49, 198, 599

F

- Field of values, 198–200, 202, 217
 Formal orthogonal polynomial, 383, 432–
 434
 Frobenius norm, 14, 347, 585

G

- Gaussian elimination, 110, 167, 347
 Gershgorin disk, 129, 314
 Givens rotation, 35, 36, 45, 55, 56, 60–62,
 84, 194, 196, 204, 213, 246, 259, 269,
 277, 312, 320, 341, 368, 369, 471,
 475, 521, 523, 525, 538
 Global algorithm, 585
 Grade of a vector, 19, 40, 43, 53, 100, 147,
 197, 374, 415, 424, 456, 477, 498,
 584

H

- Hankel determinant, 383, 433
 Hankel matrix, 16, 42, 226, 227, 230, 356
 Harmonic Ritz value, 205, 209, 218, 232,
 237–240, 505, 506, 512, 516, 518,
 521–525, 530, 533, 547, 551
 Harmonic Ritz vector, 518, 521, 522, 524,
 525, 529, 534, 545, 551, 556, 557,
 563, 564, 570
 Hermitian matrix, 14, 94
 Hessenberg matrix, 22, 24, 26, 29, 33, 36,
 42, 43, 54, 56, 61, 73, 77, 81, 82, 84,
 90, 92, 103, 109, 113, 119, 124, 134,
 137, 139, 194, 212, 216, 220, 221,
 232, 241, 242, 245–249, 263, 267,
 269–271, 305, 311, 320, 379, 380,
 419, 467, 468, 475, 499, 501, 503,
 506, 508, 511, 512, 517, 520, 522,
 525, 536, 584, 590, 595
 Householder reflection, 33, 36, 102, 269,
 307, 316, 598

I

- Ideal GMRES, 202, 203, 290
 IEEE arithmetic standard, 43, 189
 Instability, 117, 311, 331, 599

J

- Jordan block, 18–20, 41, 200, 226, 229, 230
 Jordan canonical form, 18–21, 26, 41, 226,
 229, 519

K

- Krylov matrix, 41, 42, 56, 57, 85, 100, 109,
 139, 206, 211, 226, 231, 341, 347,
 349

L

- Lagrange polynomial, 134
 Least squares problem, 36–39, 55, 62, 107,
 108, 124, 145, 149, 152, 154, 194,
 263, 265, 268, 269, 275, 277, 283,
 307, 341, 368, 431, 471, 472, 474,
 523–525, 532, 584, 585, 590, 602
 Lebesgue constant, 134, 135
 Lebesgue function, 135
 Leja ordering, 134, 601
 Leja points, 134
 Level of orthogonality, 150, 152–154, 156,
 157, 281, 282

- Look-ahead strategy, 115, 376, 388
 Look-ahead technique, 149, 374, 376, 377,
 386, 388, 391, 396, 403, 411, 433,
 434
 Loss of orthogonality, 150, 151, 273, 275,
 280, 283, 394
 LU factorization, 14, 18, 22–24, 32, 55, 109,
 264, 341, 347, 349, 377, 426, 535,
 543, 594, 616

M

- Machine epsilon, 44–46, 152, 153, 275, 277,
 324, 376, 478
 Matrix–vector product, 40, 44, 46, 47, 49,
 112, 118, 130, 131, 147, 151, 171,
 269, 273, 274, 278, 304, 306, 345,
 367, 385, 402, 411, 415, 423, 427,
 430–433, 435, 440, 442, 447, 461,
 465, 474–477, 479, 481, 486, 488,
 520, 522, 539, 545, 583, 603, 614
 Maximum attainable accuracy, 153, 157,
 196, 273, 274, 276, 278, 281–283,
 286, 325, 327, 331, 334, 335, 349,
 368, 391, 394, 395, 398, 435, 436,
 439, 440, 443, 447, 469, 478, 481,
 483–485, 538, 540, 541, 545, 546,
 549, 571, 599, 600
 Minimal polynomial, 19, 20, 40, 82, 197,
 203, 584
 Moment matrix, 26, 57, 71, 74, 85, 226
 MR method, 56, 81, 82, 517

N

Nonderogatory matrix, 20–22, 56, 77, 82, 206, 380
 Nonsymmetric Lanczos algorithm, 112, 148, 149, 169, 356, 358, 368, 373, 374, 377, 379–381, 383, 389, 391–394, 421, 431, 433, 519
 1 -norm, 14, 376
 ℓ_2 norm, 14, 17, 48
 ∞ -norm, 14, 599

Normal equation, 55, 58, 108, 264, 265, 294, 428, 469, 522, 590, 601
 Normal matrix, 19, 26, 42, 78, 82, 84, 85, 134, 198, 203, 204, 216, 224, 225, 232, 237, 240, 241, 504, 505, 521

O

OR method, 77, 78, 81, 193
 Orthogonalization, 102, 137, 147, 149, 193, 269, 270, 272–274, 277, 315, 474, 528, 541, 582
 Orthogonal matrix, 14, 149, 219, 221, 275, 299, 602
 Orthogonal polynomial, 383

P

Padé approximant, 384, 388, 434
 Padé table, 383, 388
 Partial orthogonalization, 467, 475, 476, 478, 482, 483
 Partial pivoting, 23, 24, 110, 147, 167, 341, 345, 347, 349, 535, 536
 Peak–plateau phenomenon, 61, 74, 278, 280
 Perturbed Arnoldi relation, 151, 152, 275, 326, 436
 Pipelining, 273, 324, 435, 443, 599
 Polynomial numerical hull, 200
 Positive definite matrix, 14, 15, 42, 48, 120–123, 137, 199, 202, 213, 304, 306, 307, 503, 587
 Preconditioning, 47, 50, 402, 435, 456, 516, 532–534, 543, 545, 562, 569, 579, 580, 616, 622
 Principal submatrix, 14, 15, 20, 28, 36, 54, 90, 139, 227, 231, 273, 326, 378, 380, 508, 511
 Product-type method, 411, 432, 434
 Pseudospectrum, 200, 201, 290

Q

QR factorization, 14, 18, 42, 55, 99, 102, 133, 136, 147, 149, 151, 211, 212, 216, 243, 269, 274, 309, 312, 316, 524, 525, 536, 584, 594
 Quasi-residual, 56, 58–61, 63, 73, 74, 77, 82, 83, 85, 89, 92, 193, 346–349, 368, 378, 381, 382, 402, 419, 431, 503

R

Rank-one modification, 15, 91, 120–123, 203, 215, 254, 264
 Recycling, 585, 587, 588
 Regularization method, 594
 Reorthogonalization, 153, 272–274
 Residual polynomial, 65–68, 71, 75, 76, 78–81, 84, 85, 93, 197, 203, 232–234, 236–240, 298, 326, 415, 420, 421, 426, 428, 433, 436, 457, 464, 476, 504, 505, 516, 517, 532
 Ritz value, 134–136, 147, 180, 205–208, 218, 232–236, 240, 269, 288, 392, 506, 512, 516, 521, 601
 Ritz vector, 436, 534, 539
 RODDEC, 136, 147, 269
 Rounding error, 43, 45, 150, 151, 277, 281, 282, 324–326, 349, 392–394, 436, 437, 439, 478, 480, 481, 539, 598

S

Schur complement, 15, 24, 123, 253
 Schur parameter, 245–249
 Secular equation, 120, 121
 Shadow vector, 115, 367, 374, 379, 380, 396, 399, 414, 460, 476, 478
 Sherman–Morrison formula, 15, 58, 123, 254, 263, 264 377
 Sherman–Morrison–Woodbury formula, 15, 266
 Shift, 133, 134, 147, 181, 269, 523, 524, 588, 595
 Singular value, 20, 21, 99, 128, 129, 138, 152, 153, 156, 167, 171, 176, 177, 179, 181, 182, 281, 311, 314, 320, 325, 376, 377, 513, 582, 593
 Singular value decomposition, 20
 Singular vector, 20, 21
 Skew-symmetric matrix, 14, 19, 129, 336, 472
 Spectral factorization, 18, 85, 86, 120, 121, 223, 224, 232, 235, 237, 239 250, 382, 505, 597

Spectral radius, 17, 20
 Spectrum, 17, 198, 206, 210, 229, 242, 249, 380, 382, 476, 513, 519, 521, 524, 533, 534
 Spoke set, 134, 180, 181
 Stability, 35, 36, 45, 47, 101, 110, 111, 147, 194, 269, 273, 306, 324, 524, 525, 536
 Stagnation, 61, 65, 74, 82, 203, 207, 216–223, 251, 266, 289, 312, 334, 348, 382, 426, 501, 503, 505, 508, 515, 516, 518
 Stopping criterion, 48–50, 251, 286, 498, 515, 603, 614

T

Tall and skinny QR, 136, 147, 270
 Tiled QR factorization, 136, 147
 Toeplitz matrix, 16, 200, 204
 Triangular Hessenberg decomposition, 26, 56, 57, 79, 80, 90, 320, 511

Tridiagonal matrix, 31, 32, 42, 78, 113–115, 128–131, 136, 176, 204, 313, 355, 358, 368, 377, 379–381, 389, 393, 417–420, 431, 463, 595, 602

Truncation, 62, 138, 181, 497, 537, 582

Twice is enough, 274

U

Unitary matrix, 14, 15, 19, 20, 33, 35–37, 43, 77, 78, 81, 84, 89, 198, 205, 207, 209–213, 218–221, 224, 241–246, 248–251, 346, 348, 471, 506

Unit roundoff, 43, 150, 268, 273, 277, 599

V

Vandermonde matrix, 16, 41, 86, 87, 226, 232

W

Worst-case GMRES, 82, 202, 216