NUMERICAL METHODS FOR LARGE LYAPUNOV EQUATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Carl Christian Kjelgaard Mikkelsen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2009

Purdue University

West Lafayette, Indiana

To Birgitte

ACKNOWLEDGMENTS

I am grateful for the continued support of my advisor, Ahmed Sameh. Above all, Ahmed helped restore my confidence, when I had none of my own. Ahmed carefully read my manuscript and helped me improve the presentation.

I would like to thank Jie Shen, Robert Skeel, and Arshak Petrosyan for serving on my committee.

I would like to thank my two friends and colleagues Maxim Naumov and Murat Manguoglo. We have had many discussions which I have found very useful.

A number of people have assisted me in one way or another. Ole Østerby (U. of Aarhus, Denmark) and Zahari Zlatev (DMU, Denmark) made it possible for me to come to Purdue, when there was no funding to be found in Denmark. Peter Spelluci (TU, Darmstadt, Germany) and Robert Israel (UBC, Vancouver, Canada) have answered a couple of my questions on Gaussian elimination with partial pivoting and Lyapunov operators. Zvonimir Bothe (U. of Ljubljana, Slovenia), Marko Petkovsek (U. of Ljubljana, Slovenia) and Nicolas J. Higham (U. of Manchester, England) have answered some questions on Gaussian elimination and the Cholesky factorization. Petkovsek scanned and sent me several papers which could only be found in the library at Ljubljana. Bent Ørsted (U. of Aarhus, Denmark) provided me with an elegant proof for a theorem on the Cayley transform. Bernard Philippe (IRISA, France) and Paul van Dooren (CU, Louvain, Belgium) both suggested that I use the Lanczos biorthogonalization procedure in order to run BCG on a pair of Lyapunov matrix equations in the Kronecker product form. Philippe sent me several manuscripts on linear systems and Lyapunov equations. Imad Jaimoukha (IC, London, England) answered some specific questions on his GMRES method for Lyapunov equations. We ran several examples until my output was consistent with his. Y. Zhou (SMU, Dallas, Texas, US) kindly provided me a MATLAB version of D. C. Sorensen's solver for tall

Sylvester equations. A. S. Hodel (U. of. Auburn, Alabama, US) and B. Tenison have been very helpful answering some questions on their algorithms for solving Lyapunov equations. Marlis Hochbruck verified a couple of my questions on preconditioned Krylov subspace methods for Lyapunov equations in the Kronecker product form. David Drasin, Tzuong-Tsieng Moh, Jiu-Kang Yu, Ayhan Irfanoglu, all from Purdue University, helped me to solve a variety of problems.

Renate Mallus, Amy J. Ingram and William J. Gorman of the CS graduate office at Purdue have been very helpful and always had time to talk. Tammy S. Muthig of the CS business office handled my paperwork in a very competent and friendly manner.

Finally, I would like to thank my girlfriend Birgitte Brydsø without whom I would not have life and my work would not be possible.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Mikkelsen, Carl Christian Kjelgaard Ph.D., Purdue University, May, 2009. Numerical methods for Large Lyapunov equations. Major Professor: Ahmed Sameh.

Balanced truncation is a standard technique for model reduction of linear time invariant dynamical systems. The most expensive step is the numerical solution of a pair of Lyapunov matrix equations.

We consider the direct computation of the dominant invariant subspace of a symmetric positive semidefinite matrix, which is given implicitly as the solution of a Lyapunov matrix equation. We show how to apply subspace iteration with Ritz acceleration in this setting.

An n by n Lyapunov matrix equation is equivalent to a standard linear system with n^2 unknowns. Theoretically, it is possible to apply any Krylov subspace method to this linear system, but this option has not really been explored, because of the $O(n^2)$ flops and storage requirement. In this dissertation we show that it is possible to reduce these requirement to $O(n)$ for Lyapunov equations with a low rank inhomogeneous right-hand side. We show how to accomplish the reduction for a variety of methods including GMRES, CG, BCG and CGNR. In each case the key observation is a special relationship between certain Krylov subspaces in $\mathbb{R}^n$ and $\mathbb{R}^{n^2}$.

It is theoretically possible to precondition a Lyapunov matrix equation which is written as a standard linear system. However, our investigation has revealed that the choice of preconditioners is extremely limited, if we are to keep the storage and flops count at $O(n)$.

Above all we have found that while it is certainly possible to reduce the resource requirements to $O(n)$, the constants are too large to be competitive. The fundamental

problem is that Krylov subspace methods are not taking advantage of the low rank phenomenon for Lyapunov matrix equations.

Currently, the most successful Lyapunov matrix equations solver is the low rank cyclic Smith method. Central to this method is the automatic selection of certain shift parameters, preconditioners and the solution of certain linear systems. This is extremely difficult to accomplish in general, and the problem simplifies considerably in the special case in which the defining matrices can be reordered as narrow banded matrices. Currently, it is the solution of these narrow banded linear systems which is the bottleneck in an efficient parallel implementation of the low rank cyclic Smith method.

In the final part of this dissertation we consider the parallel solution of narrow-banded linear systems. We do an error analysis of the truncated SPIKE algorithm which applies to systems which are strictly diagonally dominant by rows. Above all, we establish bounds on the decay rate of the spikes and the truncation error which are tight. We explain why this analysis only carries partially to the general case. Our analysis of the truncated SPIKE algorithm has immediate implications for the overlapping partition method (OPM). Finally we consider the question of reducing the amount of interprocessor communication during the solve phase for a general narrow banded linear system. The final conclusion is that such a system is essentially block diagonal in a sense which can be made very precise.

1. Introduction

This is a dissertation on numerical methods for solving the Lyapunov matrix equation

$$AX + XA^T + Q = 0,$$

where A and Q are known real n by n matrices and X is a real n by n matrix, which is unknown.

Lyapunov matrix equations occur in a variety of situations, but we are primarily interested in model reduction by balanced truncation, which requires the solution of a pair of Lyapunov equations,

$$AP + PA^T + BB^T = 0,$$
$$A^TQ + QA + C^TC = 0,$$

where B and C^T are tall matrices, such that AB, and A^TC^T are defined.

Frequently, the solution of a Lyapunov matrix equation admits a good low rank approximation. This is the low rank phenomenon for Lyapunov equations. It is not universal, but occurs frequently enough to have serious practical implications. The central question for Lyapunov equations is how to compute a good low rank approximation of the solution, without forming the solution explicitly.

The dissertation is organized as follows: Chapters 2 through 7 contain the necessary background material. Our original contributions are concentrated in Chapters 8 through 12.

- Chapter 2 contains a list of elementary results on Lyapunov matrix equations, including existence and uniqueness theorems, solutions and their structures, as well as a brief account of the low rank phenomenon.

- Chapter 3 explains the importance of Lyapunov equations in the field of model reduction. We discuss stable linear systems, define the transfer function, introduce the concept of balancing and explain the relationship between a balancing transformation and certain eigenvalue problems.

- Chapter 4 contains a brief description of the standard subspace iteration and the Ritz acceleration scheme.

- Chapter 5 contains an elementary discussion of Krylov subspace methods for standard linear systems. However, this chapter contains a new result, Theorem 5.4.3, which is a simple extension of a recent result by Gamti and Philippe [12] on the convergence of GMRES.

- Chapter 6 surveys the current solvers for Lyapunov equations, including Bartels-Stewart's method, the Hessenberg-Schur method, the matrix sign function iteration, the Arnoldi and the GMRES method, as well as the ADI family of methods. We prove a new result, namely the fact that the Arnoldi method for Lyapunov equations can have an arbitrary positive residual history.

- Chapter 7 contains a description of the approximate power iteration by A. S. Hodel [24].

The main contributions are organized as follows:

- Chapter 8 describes how to apply the standard subspace iteration with Ritz acceleration to an operator which is given only implicitly as the solution to a Lyapunov matrix equation.

- Chapter 9 explores a consequence of Chapter 8 for the problem of computing the dominant eigenspace for the product PQ.

- Chapter 10 is concerned with the application of Krylov subspace methods to Lyapunov equations in Kronecker product form. The main result is Theorem

10.3.1 which establishes a simple relationship between a pair of Krylov subspaces in $\mathbb{R}^n$ and $\mathbb{R}^{n^2}$. This allows for a very compact representation of the necessary vectors in $\mathbb{R}^{n^2}$ and it is the key step in reducing the resource requirements from $O(n^2)$ to $O(n)$. Unfortunately, this representation is so specialized that it does not permit a general preconditioning strategy.

- In Chapter 11 we explain that it is possible to precondition Krylov subspace methods for Lyapunov equations in the Kronecker product form using $O(n)$ resources. However, because of size considerations, the standard preconditioners for linear systems can not be applied. The choice of preconditioners appear to be extremely limited.

- Chapter 12 deals with the SPIKE algorithms for solving narrow banded linear systems. Almost every solver of Lyapunov equations requires the solution of linear systems, and they are frequently either narrow banded or admit a good narrow banded preconditioner.

  Our analysis of the truncated SPIKE algorithm has immediate implications for another method for solving narrow banded linear systems, namely the overlapping partition method (OPM).

  We also consider the nature of a general narrow banded linear system. We show that it is possible to rearrange the calculations, so that the solve phase can be carried out with a single one to all communication.

- Chapter 13 contains a summary of our findings and research contributions.

  The dissertation can be read sequentially, beginning with Chapter 2, but there are also three distinct paths. The first consists of Chapters 2, 3, 4, 7, 8, and 9 and deals with the problem of computing a dominant subspace of P or of a product PQ, where P and Q are solutions of a pair of Lyapunov equations. The second path begins with Chapter 2 and continues with Chapters 5, 6, 10, and 11. The central theme is Krylov subspace methods for Lyapunov equations in

the Kronecker product form. The third and final path begins with Chapter 2, skips to Chapter 6, and concludes with Chapters 12. The theme is the solution of narrow banded linear systems, a topic which is interesting in its own right, but which is also essential for the solution of Lyapunov matrix equations.

2. Elementary theory of Lyapunov equations

2.1 Introduction

Let A and Q be real n by n matrices. The continuous time Lyapunov equation is given by

$$AX + XA^T + Q = 0, \tag{2.1}$$

and the discrete time Lyapunov equation is given by

$$AXA^T - A + Q = 0, \tag{2.2}$$

where the solution X is an n by n matrix.

This chapter contains a list of well known results on Lyapunov equations. We discuss the existence and uniqueness question for Lyapunov equations, different solution formulae, the structure of solutions, as well as the low rank phenomenon.

2.2 The Kronecker product

Let A be a real n by m matrix and let B be a real k by l matrix. The Kronecker product of A and B is the nk by ml matrix $A \otimes B$ given by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & & a_{1m}B \\ a_{21}B & a_{22}B & \ddots & & a_{2m}B \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \dots & \dots & a_{nm}B \end{bmatrix}. \tag{2.3}$$

If X is a real n by m matrix, then $\mathrm{vec}(X)$ is the vector in $\mathbb{R}^{nm}$ formed by stacking the columns of X on top of each other, specifically

$$\mathrm{vec}(X)_{i+(j-1)n} = x_{ij}, \quad i = 1, 2, \dots, n, \;\; j = 1, 2, \dots, m. \tag{2.4}$$

This is exactly the manner in which the programming language FORTRAN stores a dense matrix.

The Kronecker product satisfies the following proposition.

Proposition 2.2.1 *Let A, B, C, and D be real matrices.*

- *In general*
$$(A \otimes B)^T = A^T \otimes B^T.$$
- *If the products AC and BD are defined, then*
$$(A \otimes B)(C \otimes D) = AC \otimes BD.$$
- *If the product ABC is defined, then*
$$\mathrm{vec}(ABC) = (A \otimes C^T)\mathrm{vec}(B).$$
- *If A and B are nonsingular matrices, then $A \otimes B$ is nonsingular, and*
$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

Additional properties of the Kronecker product are given in Horn and Johnsen [26].

By applying the vec operator to both sides of the continuous time Lyapunov matrix equation we see that it is equivalent to the following standard linear system

$$\tilde{A}\tilde{x} + \tilde{q} = 0, \tag{2.5}$$

where

$$\tilde{A} = A \otimes I + I \otimes A, \quad \tilde{q} = \mathrm{vec}(Q), \tag{2.6}$$

and

$$\tilde{x} = \mathrm{vec}(X). \tag{2.7}$$

We emphasize that the real matrix $\tilde{A}$ is n^2 by n^2 and that both $\tilde{q}$ and $\tilde{x}$ are vectors in $\mathbb{R}^{n^2}$.

In this dissertation we are primarily interested in the special case where Q is a real symmetric positive semidefinite matrix with very low rank, i.e.

$$Q = BB^T, \tag{2.8}$$

where B is a real tall matrix with relatively few columns. This special case is very important in order reduction of dynamical systems [1, 68].

2.3 Existence and uniqueness theorems

Let A be a real n by n matrix, and let $\sigma(A) \subset \mathbb{C}$ be the set of eigenvalues of A. We have the following theorem.

Theorem 2.3.1 *The continuous time Lyapunov equation* (2.1) *has a unique solution* X *for every choice of the inhomogeneous term* Q *if and only if*

$$\lambda + \mu \neq 0, \tag{2.9}$$

for all $\lambda, \mu \in \sigma(A)$.

Proof The Lyapunov matrix equation is equivalent to the standard linear system (2.5). It therefore suffices to show that the matrix

$$\tilde{A} = I \otimes A + A \otimes I,$$

is nonsingular if and only if

$$\lambda + \mu \neq 0,$$

for all $\lambda, \mu \in \sigma(A)$.

By Schur's lemma there exists a unitary n by n matrix U such that

$$R = U^* A U,$$

is upper triangular and the diagonal entries of R are equal to the eigenvalues of A. Now, the matrix

$$\begin{aligned}(U \otimes U)^*(I \otimes A + A \otimes I)(U \otimes U) &= (I \otimes U^*AU + U^*AU \otimes I) \\ &= (I \otimes R + R \otimes I),\end{aligned}$$

is obviously similar to $\tilde{A} = I \otimes A + A \otimes I$, and is also upper triangular. We see that α is an eigenvalue of $\tilde{A}$ if and only if $\alpha = \lambda + \mu$, where λ and μ are eigenvalues of A. It follows that $\tilde{A}$ is nonsingular if and only if $\lambda + \mu \neq 0$ for all $\lambda, \mu \in \sigma(A)$. ■

Theorem 2.3.2 *The discrete time Lyapunov equation* (2.2) *has a unique solution* X *for every choice of the inhomogeneous term* Q *if and only if*

$$\lambda\mu \neq 1, \tag{2.10}$$

for all $\lambda, \mu \in \sigma(A)$.

Proof The proof is similar to that of Theorem 2.3.1 and is omitted. ■

2.4 The equivalence of the two classes of Lyapunov equations

It is not difficult to see that the two classes of Lyapunov equations are equivalent. Let A be a real square matrix and let $\sigma(A)$ denote the set of eigenvalues of A. Let $\mathcal{S}$ denote the set of matrices given by

$$A \in \mathcal{S} \Leftrightarrow 1 \notin \sigma(A).$$

Let $A \in (S)$, then the Cayley transform $\mathcal{C}(A)$ of A is given by

$$\mathcal{C}(A) = (A + I)(A - I)^{-1}.$$

The Cayley transform may be viewed as an extension of the complex function

$$\phi(z) = \frac{z+1}{z-1},$$

which maps $\mathbb{C} - \{1\}$ one to one and onto itself, with $\phi(\phi(z)) = z$ for all $z \in \mathbb{C} - \{1\}$.

Theorem 2.4.1 *The following statements are true.*

1. *The Cayley transform maps* $\mathcal{S}$ *one to one and onto itself with*

$$\mathcal{C}(\mathcal{C}(A)) = A.$$

2. *If* $A \in \mathcal{S}$, *then* $\lambda \in \sigma(A)$ *if and only if* $\phi(\lambda) \in \sigma(\mathcal{C}(A))$.

3. A *is stable if and only if* $\mathcal{C}(A)$ *is convergent.*

4. A *is negative definite if and only if* $\|\mathcal{C}(A)\|_2 < 1$.

Proof The proofs are given in appendix A. ■

The Cayley transform establishes the connection between continuous and discrete time Lyapunov equations. Specifically, we have the following theorem.

Theorem 2.4.2 *Let* $A \in \mathcal{S}$, *then* X *solves the continuous time Lyapunov equation*

$$AX + XA^T + Q = 0, \tag{2.11}$$

if and only if X *solves the discrete time Lyapunov equation*

$$\mathcal{C}(A)X\mathcal{C}(A)^T - X + Q_0 = 0, \tag{2.12}$$

where Q_0 *is given by*

$$Q_0 = 2(A - I)^{-1}Q(A - I)^{-T}. \tag{2.13}$$

Proof The proof is elementary and is omitted. ■

This theorem is important to us because it shows that the two classes of equations are equally difficult. We believe that it is simpler to explain the so called low rank phenomenon for Lyapunov equations in the discrete case. First, we give a pair of solution formulae and comment on the structure of solutions.

2.5 Solution formulae

Let A be a real n by n matrix and let $\sigma(A)$ denote the set of eigenvalues of A. We say that A is a stable matrix if and only if

$$\operatorname{Re}\lambda < 0, \quad \forall\, \lambda \in \sigma(A), \tag{2.14}$$

and we say that A is convergent if and only if

$$A^j \to 0, \quad j \to \infty \tag{2.15}$$

or equivalently

$$|\lambda| < 1, \quad \forall\, \lambda \in \sigma(A). \tag{2.16}$$

It is clear that if A is a real stable matrix, then

$$\lambda + \mu \neq 0,$$

for all $\lambda, \mu \in \sigma(A)$ and the continuous time Lyapunov equation has a unique solution for every choice of the inhomogeneous term. Similarly, it is clear that if A is a real convergent matrix, then

$$\lambda\mu \neq 1,$$

for all $\lambda, \mu \in \sigma(A)$ and the discrete time Lyapunov equation has a unique solution for every choice of the inhomogeneous term.

Theorem 2.5.1 *If A is a real stable matrix, then the solution of the continuous time Lyapunov equation* (2.1) *can be written in the form*

$$X = \int_0^\infty e^{tA} Q e^{tA^T} dt. \tag{2.17}$$

Proof The stability of A implies that the improper integral exists, and if $X(\tau)$ is given by

$$X(\tau) = \int_0^\tau e^{tA} Q e^{tA^T} dt,$$

then

$$X(\tau) \to X, \quad \tau \to \infty,$$

By continuity,

$$AX(\tau) + X(\tau)A^T + Q \to AX + XA^T + Q.$$

However, we also have

$$\begin{aligned} AX(\tau) + X(\tau)A^T + Q &= \int_0^\tau \left(Ae^{tA} Q e^{tA^T} + e^{tA} Q e^{tA^T} A^T \right) dt + Q \\ &= \int_0^\tau \frac{d}{dt} \left(e^{tA} Q e^{tA^T} \right) dt + Q = e^{\tau A} Q e^{\tau A^T} \to 0, \quad \tau \to \infty, \end{aligned}$$

from which it follows,

$$AX + XA^T + Q = 0,$$

which completes the proof. ■

Theorem 2.5.2 *If A is a real convergent matrix, then the solution of the discrete time Lyapunov equation* (2.2) *can be written in the form*

$$X = \sum_{j=0}^{\infty} A^j Q (A^T)^j. \tag{2.18}$$

Proof Let X_n be given by

$$X_n = \sum_{j=0}^{n-1} A^j Q (A^T)^j.$$

Then, it is clear that

$$AX_nA^T - X_n + Q = A^n Q (A^T)^n \to 0, \quad n \to 0,$$

however it remains to be seen that the sequences $\{X_n\}_{n=0}^{\infty}$ is convergent! It is enough to show that the series

$$\sum_{j=0}^{\infty} A^j Q (A^T)^j,$$

converges absolutely, i.e.

$$\sum_{j=0}^{\infty} \|A^j Q (A^T)^j\|_2 < \infty.$$

This is a standard result from the theory of normed linear spaces, see Proposition 6.3.5 [47]. If $\|A\| < 1$, then this would be simple, however we only know that A is convergent, i.e $\rho(A) < 1$. Now, in general

$$\|A^j\|_2^{\frac{1}{j}} \to \rho(A), \quad j \to \infty,$$

which implies there exists an integer N, such that

$$\rho - \frac{1-\rho}{2} < \|A^n\|_2^{\frac{1}{n}} < \rho + \frac{1-\rho}{2} < 1, \quad n \geq N.$$

This is a standard result from the spectral theory of unital Banach algebras, see Theorem 4.1.13 [39]. Let $\epsilon = \|A_N\|_2 < 1$ and let

$$C = \max\{\|A^r\|_2 \,:\, r = 0, 1, 2, \ldots, N-1\} < \infty.$$

Then we can estimate

$$\begin{aligned}\sum_{j=0}^{\infty} \|A^j Q (A^T)^j\|_2 &\leq \|Q\|_2 \sum_{j=0}^{\infty} \|A^j\|_2^2 \\ &= \|Q\|_2 \sum_{q=0}^{\infty} \sum_{r=0}^{N-1} \|A^{qN+r}\|_2^2 \leq \|Q\|_2 \sum_{q=0}^{\infty} \sum_{r=0}^{N-1} \epsilon^{2q} \|A^r\|_2^2 \\ &\leq C^2 N \|Q\|_2 \sum_{q=0}^{\infty} \epsilon^{2q} = C^2 N \|Q\|_2 \frac{1}{1-\epsilon^2} < \infty.\end{aligned}$$

We conclude that

$$X = \sum_{j=0}^{\infty} A^j Q (A^T)^j,$$

is well defined and satisfies

$$AXA^T - X + Q = 0.$$

■

2.6 The structure of solutions

Theorem 2.6.1 *Let A be a real square matrix such that $\lambda + \mu \neq 0$, for all $\lambda, \mu \in \sigma(A)$, and let X be the solution of the continuous time Lyapunov equation* (2.1)*. Then the following statements are true:*

- *If Q is symmetric, then X is symmetric.*
- *If A is stable, and if Q is symmetric positive (semi)definite, then X is symmetric positive (semi)definite,*

Proof By transposing the Lyapunov equation we discover that X and X^T satisfy the same equation. The uniqueness Theorem 2.3.1 now implies that $X = X^T$. The second statement follows by applying the solution formula (2.17). ■

Theorem 2.6.2 *Let A be a real square matrix such that $\lambda\mu \neq 1$, for all $\lambda, \mu \in \sigma(A)$, and let X be the solution of the discrete time Lyapunov equation* (2.2)*. Then the following statements are true:*

- *If Q is symmetric, then X is symmetric.*
- *If A is convergent, and if Q is symmetric positive (semi)definite, then X is symmetric positive (semi)definite.*

Proof By transposing the Lyapunov equation we discover that X and X^T satisfy the same equation. The uniqueness Theorem 2.5.2 now implies that $X = X^T$. The second statement follows by applying the solution formula (2.18). ■

In this dissertation we focus almost exclusively on the special case in which A is a real stable (convergent) matrix and

$$Q = BB^T,$$

where B is a real tall matrix. If A is any real square matrix such that AB is defined, then the Krylov subspace $K_l(A, B)$ is given by

$$K_l(A, B) = \text{Ran}\begin{bmatrix} B & AB & A^2B & \dots & A^{l-1}B \end{bmatrix} \subseteq \mathbb{R}^n, \quad l = 1, 2, \dots, \tag{2.19}$$

It is clear, that

$$K_l(A, B) \subseteq K_{l+1}(A, B)$$

for $l = 1, 2, \dots$, and there exists a smallest integer m such that

$$K_m(A, B) = K_l(A, B)$$

for all $l \geq m$. The number m is called the grade of B with respect to A. The Krylov subspace

$$K(A, B) = K_m(A, B), \tag{2.20}$$

is the smallest A invariant subspace which contains $\text{Ran}\, B$.

Theorem 2.6.3 *Let A be a real square matrix such that $\lambda+\mu \neq 0$ for all $\lambda, \mu \in \sigma(A)$, let B be a real tall matrix such that AB is defined, and let X be the solution of the continuous time Lyapunov equation*

$$AX + XA^T + BB^T = 0. \tag{2.21}$$

Let V be a matrix with orthonormal columns, such that $\mathrm{Ran}\, V = K(A, B)$. Then

$$X = VYV^T, \tag{2.22}$$

where Y is the unique solution of the reduced order equation

$$HY + YH^T + V^T BB^T V = 0, \tag{2.23}$$

where $H = V^T AV$.

The proof of this theorem is elementary, but since it contains several important ideas, we present it below.

Proof By definition $\mathrm{Ran}\, V = K(A, B)$ is A invariant, which implies

$$AV = VH,$$

where $H = V^T AV$. We claim that $\sigma(H) \subset \sigma(A)$, which in turn implies that the reduced order equation (2.23) has a unique solution Y. Let (α, x) be an eigenpair for H. Then $\alpha x = Hx$, which implies

$$\alpha Vx = VHx = AVx,$$

from which it is apparent that (α, Vx) is an eigenpair for A. Now, let Y be the unique solution of equation (2.23). Then we claim that VYV^T is a solution of the original Lyapunov equation (2.21). We have

$$\begin{aligned} AVYV^T + VYV^T A^T + BB^T &= VHYV^T + VYH^T V^T + VV^T BB^T VV^T \\ &= V\left(HY + YH^T + V^T BB^T V\right) V^T = 0, \end{aligned}$$

because $\mathrm{Ran}\, B \subseteq \mathrm{Ran}\, V$, and Y is a solution of equation (2.23). Finally, it follows from the uniqueness theorem, Theorem 2.3.1, that $X = VYV^T$. ■

Theorem 2.6.4 *Let A be a real stable matrix, let B be a real tall matrix such that AB is defined, and let X be the solution of*

$$AX + XA^T + BB^T = 0,$$

then

$$\operatorname{Ran} X = K(A, B).$$

Proof By Theorem 2.6.3 we have

$$\operatorname{Ran} X \subseteq K(A, B),$$

merely because $\lambda + \mu \neq 0$ for all λ and μ in $\sigma(A)$. However, since A is assumed stable, Theorem 2.5.1 applies, and we have

$$X = \int_0^\infty e^{tA} BB^T e^{tA^T} dt.$$

We must show

$$\operatorname{Ran} X \supseteq K(A, B),$$

or equivalently

$$\operatorname{Ker} X \subseteq K(A, B)^\perp.$$

If $Xv = 0$, then

$$0 = v^T X v = \int_0^\infty v^T e^{tA} BB^T e^{tA^T} v dt,$$

from which it follows

$$v^T e^{tA} B = 0, \quad t \geq 0.$$

By repeated differentiation with respect to t we discover that

$$v^T A^j e^{tA} B = 0, \quad t \geq 0, \quad j = 0, 1, 2, \ldots,$$

and by evaluating at $t = 0$ we find

$$v^T A^j B = 0, \quad j = 0, 1, 2, \ldots.$$

This is equivalent to $v \in K(A, B)^\perp$. ■

These two theorems and the techniques used in their proofs carry to the discrete case as well. Specifically, we have the following theorems.

Theorem 2.6.5 *Let A be a real square matrix such that $\lambda\mu \neq 1$ for all eigenvalues λ and μ of A. Let B be a real tall matrix such that AB is defined and let X be the solution of the discrete time Lyapunov equation*

$$AXA^T - X + BB^T = 0.$$

Let V be a real matrix with orthonormal columns, such that $RanV = K(A, B)$. Then

$$X = VYV^T,$$

where Y is the unique solution of the reduced order equation

$$HYH^T - Y + V^T BB^T V = 0,$$

where $H = V^T AV$.

Theorem 2.6.6 *Let A be a real convergent matrix, let B be a real tall matrix such that AB is defined and let X be the solution of the discrete time Lyapunov equation*

$$AXA^T - X + BB^T = 0.$$

Then

$$\operatorname{Ran} X = K(A, B).$$

It is extremely important to realize that the grade of B with respect to A need not be small and in the typical case the grade is comparable with n and the special case of

$$K(A, B) = \mathbb{R}^n$$

could be unavoidable. Nevertheless, these two theorems form the foundation for an important solver which we will discuss in detail in Chapter 6.

2.7 The low rank phenomenon

Let A be a stable n by n matrix, let B be a real tall matrix such that AB is defined, and consider the continuous time Lyapunov equation

$$AX + XA^T + BB^T = 0,$$

where, in general, the solution X is symmetric positive semidefinite (Theorem 2.6.1). Let

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0, \tag{2.24}$$

be the eigenvalues of X. For many practical application the eigenvalues of X decay extremely fast and since

$$\lambda_{k+1} = \min\{\|X - X_k\|_2 \ : \ X_k = X_k^T, \ \mathrm{rank}(X_k) \leq k\},$$

these are precisely the cases where X can be accurately approximated by a symmetric matrix with very low rank. This is the low rank phenomenon for Lyapunov matrix equations.

This phenomenon is most easily understood in term of the discrete time Lyapunov matrix equation,

$$AXA^T - X + BB^T = 0.$$

Consider the special case where $\|A\|_2 < 1$. Since A is convergent, the exact solution X is given by

$$X = \sum_{j=0}^{\infty} A^j BB^T (A^T)^j.$$

Let X_k be given by

$$X_k = \sum_{j=0}^{k-1} A^j BB^T (A^T)^j, \tag{2.25}$$

with rank at most k times the rank of B, and since $\|A\|_2 < 1$ we have

$$\|X - X_k\|_2 \leq \frac{\|A\|_2^{2k}}{1 - \|A\|_2^2} \|BB^T\|_2 \to 0, \quad k \to \infty$$

Thus, we conclude that if $\|A\|_2 \ll 1$ and B has very low rank, then X admits a good low rank approximation.

The bound given by inequality (2.7) is very simple and the eigenvalues for X may decay considerably faster than this bound would suggest.

Penzl [41] was the first to obtain a bound on the decay rate of the eigenvalues in the stable symmetric case. Zhou and Sorensen [69] obtained a bound in the case where A is stable and diagonalizable; see also the paper by Antoulas, Sorensen and Zhou [2].

It is important to understand that the low rank phenomenon is not universal, and that the decay of the eigenvalues does not depend exclusively on the spectrum of A, but also on the choice of B.

We illustrate this with an example where A is stable, B has rank 1, and $X = I$ is the solution of the corresponding continuous time Lyapunov equation. Let A be the n by n matrix given by $A = T - e_n e_n^T$, where T is the n by n skew symmetric tridiagonal Toeplitz matrix given by

$$T = \begin{bmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & & 1 \\ & & -1 & 0 \end{bmatrix}, \tag{2.26}$$

and $e_n = (0, 0, \ldots, 0, 1)^T$ is the last column in the n by n identity matrix. To see that A is stable we proceed as follows. Let λ be an eigenvalue of A and let $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{C}^n$ be the corresponding eigenvector. Then it follows by induction that $x_n \neq 0$, specifically $x_n = 0$ implies $x_{n-j} = 0$ for $j = 1, 2, \ldots, n-1$, in that order. In addition we have the following equality

$$\frac{x^* T x - |x_n|^2}{\|x\|^2} = \lambda.$$

Now, since T is normal the numerical range of T is the convex hull of its eigenvalues. If μ is an eigenvalue of T, then $\mathrm{Re}(\mu) = 0$, because T is skew symmetric. It follows that $\mathrm{Re}(x^* T x) = 0$. We conclude that A is stable. Finally, since

$$A + A^T = -2 e_n e_n^T,$$

we see that $X = I$ is the unique solution of the Lyapunov equation

$$AX + XA^T + BB^T = 0,$$

where $B = \sqrt{2}e_n$ has rank 1. Obviously, the eigenvalues for $X = I$ do not decay and there are no good low rank approximations to X. We shall return to this example in Chapter 6 where we discuss current Krylov subspace methods for solving Lyapunov equations.

The following general theorem is due to T. Penzl [41]. His proof is constructive and can be used to construct a Lyapunov equation for which a given symmetric positive definite matrix X is the solution.

Theorem 2.7.1 *Let $\tilde{A}$ be a real stable n by n matrix, and let $\tilde{B}$ be a tall matrix such that*

$$K(\tilde{A}, \tilde{B}) = \mathbb{R}^n.$$

Let X be any symmetric positive definite matrix, then there exists a nonsingular matrix T, such that X is the solution of the Lyapunov equation

$$AX + XA^T + BB^T = 0,$$

where $A = T\tilde{A}T^{-1}$, and $B = T\tilde{B}$.

The point of this section is that while low rank phenomenon occurs frequently, it is not universal. For discrete time Lyapunov equations the low rank phenomenon occurs with certainty when $\|A\|_2 \ll 1$, and the same occurs for continuous time Lyapunov when $\lambda_{\max}(A + A^T) \ll 0$. However, even in the general stable case, the phenomenon occurs frequently enough to have serious practical applications.

3. Linear time invariant dynamical systems

3.1 Introduction

In this chapter we justify our interest in investigating solution methods for Lyapunov matrix equations as well as certain related eigenvalue problems. We begin by introducing a few concepts from the theory of linear time invariant systems, before moving on to model reduction by balanced truncation. We explain the importance of the continuous time Lyapunov equation and demonstrate how to compute a balancing transformation by solving a very specific eigenvalue problem.

A thorough introduction to linear systems has been written by Zhou, Doyle, and Glover [68]. While Antoulas [1] also deals with this subject, he concentrates on techniques for model reduction.

We begin with two simple examples of linear dynamical systems which will be used throughout the dissertation.

Example 1 Consider the inhomogeneous heat equation

$$\phi_t - \phi_{xx} = b(x)u(t), \quad (x,t) \in (0,1) \times (0,\infty), \tag{3.1}$$

together with the initial condition

$$\phi(x,0) = g(x), \quad x \in (0,1), \tag{3.2}$$

and the homogeneous boundary conditions

$$\phi(0,t) = \phi(1,t) = 0, \quad t > 0. \tag{3.3}$$

Physically, ϕ represents the distribution of heat (energy/length) in a one dimensional rod occupying the unit interval. The boundary conditions correspond to the fact that the endpoints are kept at absolute zero, while the inhomogeneous term represents the input of heat from sources external to the rod.

The heat equation is discretized via

$$\dot{v}(t) = \frac{v_{j+1}(t) - 2v_j(t) + v_{j-1}(t)}{h^2} + b(x_j)u(t), \quad j = 1, 2, \ldots, N-1, \quad t > 0,$$

where

$$h = 1/N, \quad x_j = jh, \quad j = 0, 1, \ldots, N,$$

and $v_j(t)$ is an approximation of $\phi(x_j, t)$ with

$$v_j(0) = g(x_j), \quad j = 0, 1, 2, \ldots, N, \tag{3.4}$$

as well as

$$v_0(t) = v_N(t) = 0, \quad t > 0.$$

These equations can be expressed as

$$\dot{v}(t) = Av(t) + Bu(t),$$

where

$$v(t) = (v_1(t), v_2(t), \ldots, v_{N-1}(t))^T,$$

and A is the $N-1$ by $N-1$ Toeplitz matrix given by

$$A = h^{-2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & & 1 \\ & & 1 & -2 \end{bmatrix},$$

while

$$B = (b(x_1), b(x_2), \ldots, b(x_{N-1}))^T \in \mathbb{R}^{N-1}.$$

Frequently, we will not care for every single component of v, especially when N is very large. We might very well be more interested in a linear combination of the components, such as the sum

$$y(t) = h \sum_{j=1}^{N-1} v_j(t),$$

which is a very simple approximation of the thermal energy $E(t)$ stored in the rod at time t, i.e.

$$E(t) = \int_0^1 \phi(x,t)dx.$$

It is clear that this is merely a special case of a much more general situation, which we formalize with the following definition.

Definition 3.1.1 *A continuous time invariant linear dynamical system*

$$\mathbf{\Sigma} = (A, B, C, D),$$

is determined by a set of differential equations

$$\dot{x} = Ax + Bu, \quad x(0) = x_0,$$
$$y = Cx + Du,$$

where A, B, C, D are real matrices, while x, u, and y are real vectors. They are all assumed to be of compatible size. The vector $x(t)$ is called the system state, x_0 is the initial condition, $u(t)$ is called the system input, and $y(t)$ is called the system output.

Example 2 As in Example 1 we consider the initial boundary value problem

$$\phi_t - \phi_{xx} = b(x)u(t), \quad (x,t) \in (0,1) \times (0,\infty),$$

together with the initial condition

$$\phi(x,0) = g(x), \quad x \in (0,1),$$

and the homogeneous boundary conditions

$$\phi(0,t) = \phi(1,t) = 0, \quad t > 0.$$

In addition to the spatial discretization done in Example 1, we discretize the time by choosing $k > 0$ and defining

$$t_m = mk, \quad m = 0, 1, 2, \ldots$$

Now, let $\lambda = k/h^2$, then the standard Crank-Nicholson scheme for our problem is given by

$$w^{(0)} = (f(x_1), f(x_2), \ldots, f(x_{N-1}))^T,$$
$$A_1 w^{(m+1)} = A_2 w^{(m)} + kBu^{(m)}, \quad m = 0, 1, 2, \ldots,$$

where A_1 and A_2 are the $N-1$ by $N-1$ Toeplitz matrices determined by

$$A_1 = \begin{bmatrix} 1+\lambda & -\frac{\lambda}{2} & & \\ -\frac{\lambda}{2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\lambda}{2} \\ & & -\frac{\lambda}{2} & 1+\lambda \end{bmatrix},$$

and

$$A_2 = \begin{bmatrix} 1-\lambda & \frac{\lambda}{2} & & \\ \frac{\lambda}{2} & \ddots & \ddots & \\ & \ddots & \ddots & \frac{\lambda}{2} \\ & & \frac{\lambda}{2} & 1+\lambda \end{bmatrix},$$

while

$$B = (b(x_1), b(x_2), \ldots, b(x_{N-1}))^T,$$

and

$$u^{(m)} = u(t_m), \quad m = 0, 1, 2, \ldots.$$

The Crank-Nicholson scheme is second order accurate in both time and space, and it is stable for all values of $\lambda = k/h^2$. In this case stability means that $A_1^{-1}A_2$ is a convergent matrix.

As before, we might have good reasons to favor a weighted linear combination of the vector $w^{(m)}$ and the input $u^{(m)}$, i.e.

$$y^{(m)} = Cw^{(m)} + Du^{(m)}, \quad m = 0, 1, 2, \ldots,$$

over the potentially very large vector $w^{(m)} \in \mathbb{R}^{N-1}$. It is clear that we are dealing with a special case of a much more general situation. We make the following definition.

Definition 3.1.2 *A discrete time invariant linear system* $\mathbf{\Sigma} = (A, B, C, D)$ *is determined by an equation of the form*

$$x^{(m)} = Ax^{(m-1)} + Bu^{(m-1)}, \quad x^{(0)} = x_0,$$
$$y^{(m)} = Cx^{(m)} + Du^{(m)},$$

where A, B, C, *and* D *are real matrices, and* $x^{(m)}$, $u^{(m)}$ *and* $y^{(m)}$ *are real vectors. They are all assumed to be of compatible size. The vector* $x^{(m)}$ *is called the system state,* x_0 *is the initial condition,* $u^{(m)}$ *is called the system input and* $y^{(m)}$ *is called the system output.*

In this dissertation we shall focus almost exclusively on the case of continuous time, but there will be occasional references to the case of discrete time.

3.2 Elementary results and definitions

In this section we provide a list of definitions and elementary results relevant to continuous time invariant systems. Let $\mathbf{\Sigma} = (A, B, C, D)$ be a continuous time invariant linear system. Let $\phi(u; x_0; t)$ denote the solution of the state equation, then

$$\phi(u; x_0; t) = \int_0^t e^{(t-s)A)} Bu(s)ds + e^{tA}x_0.$$

Definition 3.2.1 *The state* $\bar{x} \in \mathbb{R}^n$ *is said to reachable from the zero state if there exists a* $t_1 > 0$ *and an input* $u \in C([0, t_1])$, *such that* $\bar{x} = \phi(u; 0; t_1)$.

The following theorem characterizes the set of reachable states.

Theorem 3.2.1 *The set of reachable states is the smallest* A*-invariant vector space which contains the range of* B, *i.e. the Krylov subspace* $K(A, B)$. *If every state is reachable, then the system is said to be reachable.*

Definition 3.2.2 *The state* $\bar{x} \in R^n$ *is said to be unobservable if and only if the output*

$$y(t) = C\phi(0; \bar{x}, t) = Ce^{tA}\bar{x},$$

corresponding to the input $u(t) = 0$*, satisfies*

$$y(t) = 0,$$

for all $t \geq 0$*.*

Let $\Omega \subseteq \mathbb{R}^n$ be any subset of $\mathbb{R}^n$. Then the orthogonal complement $\Omega^\perp$ to Ω is the vector space given by

$$\Omega^\perp = \{x \in \mathbb{R}^n \ : \ \forall y \in \Omega \ : (x, y) = 0\}. \tag{3.5}$$

From Definition 3.2.2 is clear that a state $\bar{x}$ is unobservable if and only if

$$\bar{x} \in K(A^T, C^T)^\perp,$$

and as result the zero state is always unobservable.

Definition 3.2.3 *If the zero state is the only unobservable state, then the system is said to be observable.*

We emphasize that a linear time invariant system $\mathbf{\Sigma} = (A, B, C, D)$ is reachable if and only if

$$K(A, B) = \mathbb{R}^n,$$

and it is observable if and only if

$$K(A^T, C^T) = \mathbb{R}^n.$$

The matrix D is not relevant to the reachability or observability of the system.

The Laplace transform is an important tool in the analysis of stable linear systems. If $x \in C([0, \infty))$ is a bounded function, then the Laplace transform X of x is given by

$$X(s) = \int_0^\infty e^{-st} x(t) dt,$$

where s is any complex number with strictly positive real part. If A is stable, and if the input is continuous and bounded independent of t, then both the state x and

the output y can be bounded independent of t and their Laplace transforms are well defined. There is a simple relationship between the Laplace transform of the input and the Laplace transform of the output, specifically

$$Y(s) = T(s)U(s),$$

where the transfer function T is given by

$$T(s) = C(sI - A)^{-1}B + D.$$

Let S be a nonsingular matrix and let $x(t)$ be the solution of the state equation, then the function $\hat{x}$ defined by

$$\hat{x}(t) = Sx(t),$$

satisfies the dynamical system

$$\begin{aligned} \dot{\hat{x}} &= SAS^{-1}\hat{x} + SBu, \quad \hat{x}(t_0) = Sx_0, \\ y &= CS^{-1}\hat{x} + Du. \end{aligned}$$

This observation motivates the following definition

Definition 3.2.4 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a stable linear time invariant system and let S be a nonsingular matrix. Then the similarity transform of $\mathbf{\Sigma}$ with respect to S is the stable linear time invariant system $\hat{\mathbf{\Sigma}} = (\hat{A}, \hat{B}, \hat{C}, \hat{D})$ given by*

$$(\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (SAS^{-1}, SA, CS^{-1}, D).$$

The following theorem follows directly from Definition 3.2.4

Theorem 3.2.2 *Let $\mathbf{\Sigma}$ be a stable linear time invariant system and let $\hat{\mathbf{\Sigma}}$ be the similarity transform of $\mathbf{\Sigma}$ with respect to a nonsingular matrix S. Then the two transfer functions are identical, i.e.*

$$T(s) = \hat{T}(s),$$

for all complex numbers s with positive real part.

3.3 Model reduction by balanced truncation

Given a linear time invariant system

$$\mathbf{\Sigma} = (A, B, C, D),$$

it is possible to integrate the differential equation and determine the state vector x as a function of the input u. However, this can be very expensive in terms of memory usage and number of arithmetic operations. In addition, the state vector x is frequently of secondary importance compared with the relationship between the input u and the output y. In the previous section we saw that the transfer function T links the Laplace transform U of the input together with the Laplace transform Y of the output, specifically

$$Y(s) = T(s)U(s).$$

However, this simple relationship is not as useful as it would appear to be. In reality we are merely trading one difficult problem for another. It is a nontrivial exercise to solve the necessary linear systems

$$(sI - A)Z = B, \tag{3.6}$$

for $Z = Z(s)$ in order to compute the transfer function,

$$T(s) = C(sI - A)^{-1}B + D.$$

If A is a general sparse matrix, then we may have to use a Krylov subspace method to solve the linear system (3.6) with a good preconditioner for every relevant choice of s. If A is a narrow banded matrix, then we can use Gaussian elimination with partial pivoting to solve our problem. However, a new LU factorization must be computed for every relevant choice of s.

This raises the question of whether it is possible to construct another system

$$\hat{\mathbf{\Sigma}} = (\hat{A}, \hat{B}, \hat{C}, \hat{D}),$$

of substantially smaller dimension, which can be used to simulate the original system $\mathbf{\Sigma}$. This is one of the central questions in model reduction.

There are many different algorithms for model reduction of dynamical systems, but it is desirable to preserve stability and to obtain a bound on the difference between the transfer functions for the reduced system and the original system. Model reduction by balanced truncation is a standard algorithm which can be used to accomplish this goal. We require the following definition.

Definition 3.3.1 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a stable linear time invariant system. Then the reachability Gramian P and the observability Gramiam Q are defined as the unique solutions of the Lyapunov equations*

$$AP + PA^T + BB^T = 0 \quad \textit{and} \quad A^TQ + QA + C^TC = 0.$$

The Gramians P, and Q satisfy

$$P = \int_0^\infty e^{tA}BB^Te^{tA^T}dt \quad \text{and} \quad Q = \int_0^\infty e^{tA^T}C^TCe^{tA}dt.$$

It is easy to show that P and Q are both symmetric positive semidefinite, and

$$\operatorname{Ran} P = K(A, B) \quad \text{and} \quad \operatorname{Ran} Q = K(A^T, C^T).$$

It follows immediately that the system $\mathbf{\Sigma}$ is reachable if and only if $P > 0$ and it is observable if only if $Q > 0$.

In general, $P \geq 0$ and $Q \geq 0$ and the matrices PQ and QP are merely similar to a symmetric positive semidefinite matrix. The Hankel singular values $\{\sigma_i\}_{i=1}^n$,

$$\sigma_1 \geq \sigma_2 \geq, \ldots, \geq \sigma_n \geq 0,$$

are defined as the eigenvalues of the products PQ and QP.

Let S be a nonsingular matrix and let $\hat{\mathbf{\Sigma}}$ be the corresponding similarity transform of the system $\mathbf{\Sigma}$. The Gramians of the transformed system satisfy

$$\hat{P} = SPS^T \quad \text{and} \quad \hat{Q} = S^{-T}QS^{-1}.$$

If S is an orthogonal matrix, then the eigenvalues of P and Q are preserved, but in general this is not the case. However, since

$$\hat{P}\hat{Q} = S(PQ)S^{-1} \quad \text{and} \quad \hat{Q}\hat{P} = S^{-T}(QP)S^{T},$$

we see that the Hankel singular values are preserved by any similarity transformation.

Since the concept of a balanced system is critical in model reduction we introduce the following definition and theorems. Notice, that it is very simple to extract a reduced model from a balanced one.

Definition 3.3.2 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a stable linear time invariant system, then $\mathbf{\Sigma}$ is said to be balanced if the Gramians are equal and diagonal, i.e.*

$$P = Q = \Sigma = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_n\}.$$

Theorem 3.3.1 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a stable linear time invariant system, then there exist a nonsingular matrix S such that the similarity transform of $\mathbf{\Sigma}$ with respect to S is balanced.*

Theorem 3.3.2 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a linear system which is stable and balanced. Let*

$$P = Q = \Sigma = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_n\},$$

be the system Gramians where

$$\sigma_1 \geq \sigma_2 \geq, \ldots, \geq \sigma_n \geq 0,$$

are the Hankel singular values. If $\sigma_r > \sigma_{r+1}$, then $\mathbf{\Sigma}$ is partitioned as

$$\Sigma = \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix},$$

where

$$\Sigma_1 = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\},$$
$$\Sigma_2 = \text{diag}\{\sigma_{r+1}, \sigma_{r+2}, \sigma_n\},$$

with A, B, and C partitioned conformally,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \textit{and} \quad C = \begin{bmatrix} C_1 & C_2 \end{bmatrix},$$

then the following statements are true:

1. *The reduced order system $(\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (A_{11}, B_1, C_1, D)$ is stable.*

2. *The transfer function of the reduced order system $\hat{T}$ given by*

$$\hat{T}(s) = C_1(sI_r - A_{11})^{-1}B,$$

satisfies

$$\sup\{|T(i\omega) - \hat{T}(i\omega)| \, : \, \omega \in \mathbb{R}, \, i^2 = -1\} \leq 2(\sigma_{r+1} + \sigma_{r+2} + \cdots + \sigma_n).$$

We see that if the Hankel singular values decay sufficiently fast, then it is possible to extract a stable reduced order model of substantially smaller dimension for which the transfer function is a good approximation of the transfer function for the original system.

We next consider the key question of computing a balancing transformation. We begin with the following lemma.

Lemma 3.3.1 *Let $\mathbf{\Sigma} = (A, B, C, D)$ be a stable system, and let $P \geq 0$ and $Q \geq 0$ be the system Gramians. Let Σ be an n by n diagonal matrix with nonnegative diagonal entries and let V, W be n by n matrices, such that*

$$PW = V\Sigma, \quad QV = W\Sigma,$$

and $V^TW = I$, then the system

$$\hat{\mathbf{\Sigma}} = (W^TAV, W^TB, CV, D),$$

is balanced and the transformed Gramians satisfy

$$\hat{P} = \hat{Q} = \Sigma.$$

Proof Let $S = W^T$, then $S^{-1} = V$ and we recognize the transformation as a similarity transformation. The transformed Gramians are

$$\hat{P} = SPS^T = W^T PW = W^T(PW) = W^T(V\Sigma) = (W^T V)\Sigma = \Sigma,$$

and

$$\hat{Q} = S^{-T}QS^{-1} = V^T QV = V^T(QV) = V^T(W\Sigma) = (V^T W)\Sigma = \Sigma.$$

■

But how do we compute matrices V, W, and a diagonal matrix Σ with nonnegative entries such that

$$PW = V\Sigma, \quad QV = W\Sigma,$$

and $V^T W = I$? It is clear that if such a factorization exists, then

$$PQV = P(W\Sigma) = V\Sigma^2, \quad QPW = Q(V\Sigma) = W\Sigma^2,$$

which explains why we must study the eigenvalue problem for the products PQ and QP.

3.4 The eigenvalue problem for the products PQ and QP

In this section we show how to compute a balancing transformation for a linear time invariant system. In the previous section we saw that this problem is intimately related to the eigenvalue problem for the operators PQ and QP. We begin with the following theorem.

Theorem 3.4.1 *Let P and Q be any pair of symmetric positive semidefinite matrices. Then there exist nonsingular matrices V, W and a diagonal matrix Σ with nonnegative entries, such that*

$$PQV = V\Sigma^2, \quad QPW = W\Sigma^2.$$

By reordering the rows and columns of Σ we may assume, that

$$\Sigma = \text{diag}\{\Sigma_1, \Sigma_2, \ldots, \Sigma_k\},$$

where $\Sigma_i = \sigma_i I_{i_r}$, and

$$\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_k^2 \geq 0,$$

are the distinct eigenvalues of PQ, and QP. Now partition V, and W conformally with Σ, i.e.

$$V = \begin{bmatrix} V_1, & V_2, & \ldots, V_k \end{bmatrix}, \quad W = \begin{bmatrix} W_1, & W_2, & \ldots, W_k \end{bmatrix}.$$

The matrices V and W given by Theorem 3.4.1 need not be mutually orthogonal. However, we have the following lemma.

Lemma 3.4.1 *Let $i \neq j$ and let v be a column of V_i and let w be a column of W_j, then v and w are orthogonal.*

Proof Recalling that P and Q are symmetric positive semidefinite, we have

$$\sigma_i^2(v, w) = (\sigma_i^2 v, w) = (PQv, w) = (v, QPw) = (v, \sigma_j^2 w) = \sigma_j^2(v, w),$$

or equivalently

$$(\sigma_i^2 - \sigma_j^2)(v, w) = 0,$$

which implies $(v, w) = 0$. ■

We conclude that the product $V^T W$ is block diagonal with respect to this partitioning. In particular, if the eigenvalues of PQ, and QP are distinct, then $V^T W$ is diagonal, but $V^T W$ is not necessarily the identity matrix. However, it is easy to ensure that $V^T W = I$. Since V and W are nonsingular and $V^T W$ is block diagonal, then $V_i^T W_i$ is nonsingular. Now suppose we replace W with $\tilde{W}$, where

$$\tilde{W} = \begin{bmatrix} W_1(V_1^T W_1)^{-1}, & W_2(V_2^T W_2)^{-1}, & \ldots, & W_k(V_k^T W_k)^{-1} \end{bmatrix},$$

then we force

$$V^T \tilde{W} = I_n.$$

In addition, since Σ_i^2 is a scaled identity matrix, then

$$PQ\tilde{W}_i = PQW_i(V_i^TW_i)^{-1} = W_i\Sigma_i^2(V_i^TW_i)^{-1} = W_i(V_i^TW_i)^{-1}\Sigma_i^2 = \tilde{W}_i\Sigma_i^2,$$

which implies

$$PQ\tilde{W} = \tilde{W}\Sigma^2.$$

We summarize the above in the following theorem.

Theorem 3.4.2 *Let (A, B, C, D) be a stable system, and let P and Q be the system Gramians. Then there exists a pair of matrices V and W, and a diagonal matrix Σ with nonnegative diagonal entries, such that*

$$PQV = V\Sigma^2, \quad QPW = W\Sigma^2,$$

and $V^TW = I$.

We claim that it is possible to further modify V, and W to achieve

$$PW = V\Sigma,$$
$$QV = W\Sigma,$$

while retaining $V^TW = I$, which settles the problem of computing a balancing transformation.

We begin by showing that if W^TPW and V^TQV are partitioned conformally with Σ, then they are block diagonal. This is a straightforward generalization of a previous lemma.

Lemma 3.4.2 *Let P, and Q be symmetric positive semidefinite matrices, let Σ be a diagonal matrix with nonnegative entries,*

$$\Sigma = \mathrm{diag}\{\Sigma_1, \Sigma_2, \ldots, \Sigma_k\},$$

where

$$\Sigma_i = \sigma_i I_{n_i}, \quad i = 1, 2, \ldots, k,$$

with

$$\sigma_1 > \sigma_2 > \cdots > \sigma_k \geq 0,$$

and let V *and* W *be matrices, such that*

$$PQV = V\Sigma^2, \quad QPW = W\Sigma^2,$$

and $V^TW = I$*. If* P *and* Q *are partitioned conformally with* Σ*, then they are block diagonal.*

Proof Partition V and W conformally with Σ, i.e.

$$V = \begin{bmatrix} V_1, & V_2, & \ldots, V_k \end{bmatrix}, \quad W = \begin{bmatrix} W_1, & W_2, & \ldots W_k, \end{bmatrix}.$$

Let $i \neq j$. It suffices to show that $W_i^T P W_j = 0$. The same technique can be applied to show that $V_i^T Q V_j = 0$. We have

$$\begin{aligned} \sigma_i^2(W_i^T P W_j) = (W_i\Sigma_i^2)^T P W_j = (QPW_i)^T P W_j \\ = W_i^T PQPW_j = W_i^T P W_j \Sigma_j^2 = \sigma_j^2(W_i^T P W_j), \end{aligned}$$

or equivalently

$$(\sigma_i^2 - \sigma_j^2)(W_i^T P W_j) = 0,$$

from which it follows, entry by entry, that $W_i^T P W_j = 0$. ■

We remark that the proof is identical to the proof that the eigenvectors corresponding to distinct eigenvalues of a Hermitian matrix are mutually orthogonal.

In addition, we have the following elementary identity

$$(W^TPW)(V^TQV) = (V^TQV)(W^TPW) = \Sigma^2.$$

or equivalently

$$(W_i^TPW_i)(V_i^TQV_i) = (V_i^TQV_i)(W_i^TPW_i) = \Sigma_i^2,$$

for $i = 1, 2, \ldots, k$. Now, let

$$P_i = W_i^T P W_i \quad \text{and} \quad Q_i = V_i^T Q V_i,$$

then

$$P_i Q_i = Q_i P_i = \Sigma_i^2,$$

for $i = 1, 2, \ldots, k$.

For the sake of simplicity we consider only the case in which P and Q are both positive definite, which implies that the Hankel singular values are all strictly positive, i.e. $\sigma_i > 0$, for $i = 1, 2, \ldots, k$. Let us drop the subscript i, and write

$$P = P_i \quad \text{and} \quad Q = Q_i,$$

so that

$$PQ = QP = \sigma^2 I.$$

We claim that it is possible to find V and W such that

$$W^T P W = \sigma I, \tag{3.7}$$

$$V^T Q V = \sigma I, \tag{3.8}$$

with $V^T W = I$. Now, since P is symmetric positive definite, there exists an orthogonal matrix U, such that

$$U^T P U = \Lambda,$$

is diagonal with positive diagonal entries. It follows that if

$$W = U \Lambda^{-1/2} \sigma^{1/2},$$

then

$$W^T P W = \sigma I.$$

Now, given W there is only one choice for V, because we want $V^T W = I$, namely

$$V = W^{-T} = U^{-T} \Lambda^{1/2} \sigma^{-1/2} = U \Lambda^{1/2} \sigma^{-1/2},$$

and we have

$$V^T Q V = W^{-1}\sigma^2 P^{-1} W^{-T} = \sigma^2 (W^T P W)^{-1} = \sigma I.$$

We summarize this entire discussion in Algorithm 1, which can be used to compute a balancing transformation.

Input: Gramians $P > 0$, $Q > 0$.

Output: Matrices V, and W such that $(W^T AV, W^T B, CV, D)$ is balanced and ready for truncation.

1: Solve the eigenvalue problems

$$PQV = V\Sigma^2, \quad QPW = W\Sigma^2,$$

where

$$V = \begin{bmatrix} V_1, & V_2, & \ldots, V_k \end{bmatrix}, \quad \text{and} \quad W = \begin{bmatrix} W_1, & W_2, & \ldots, W_k \end{bmatrix},$$

are partitioned conformally with

$$\Sigma = \operatorname{diag}\{\sigma_1 I_{n_1}, \sigma_2 I_{n_2}, \ldots, \sigma_k I_{n_k}\},$$

where

$$\sigma_1 > \sigma_2 > \cdots > \sigma_k > 0,$$

are the distinct Hankel singular values.

2: **for** i=1,2,...,k **do**

3: $W_i := W_i (V_i^T W_i)^{-1}$.

4: Set $P_i := W_i^T P W_i$.

5: Compute the spectral decomposition $U_i^T P_i U_i = \Lambda_i$.

6: Set $W_i := W_i U_i \Lambda_i^{-1/2} \sigma_i^{1/2}$.

7: Set $V_i := V_i U_i \Lambda_i^{1/2} \sigma_i^{-1/2}$.

8: **end for**

Algorithm 1: Computation of a balancing transformation

Now do we actually have to compute a balancing transformation, i.e. a full n by n nonsingular matrix S? The answer is no! This is the point where we appreciate the value of computing S^{-1} and S separately, as $S^{-1} = V$ and $S = W^T$, where the columns of V and W are eigenvectors for PQ and QP, respectively. Assuming that $\sigma_k > \sigma_{k+1}$, such that the rank k dominant eigenspaces for both PQ and QP are well defined, we partition V and W as follows

$$V = \begin{bmatrix} V_1, & V_2 \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} W_1, & W_2 \end{bmatrix},$$

where V_1 spans the rank k dominant eigenspace for PQ and W_1 spans the rank k dominant eigenspace for QP. The balanced system is

$$\hat{\mathbf{\Sigma}} = (SAS^{-1}, SB, CS^{-1}, D) = (W^T AV, W^T B, CV, D),$$

where

$$W^T AV = \begin{bmatrix} W_1^T AV_1 & W_1^T AV_2 \\ W_2^T AV_1 & W_2^T AV_2 \end{bmatrix}, \quad W^T B = \begin{bmatrix} W_1^T B \\ W_2^T B \end{bmatrix}, \quad \text{and} \quad CV = \begin{bmatrix} CV_1, & CV_2 \end{bmatrix}.$$

The truncation process discards the majority of this data, retaining only the upper left corner of $W^T AV$, the first block row of $W^T B$, as well as the first block column of CV. In other words, the reduced order model is given by the system

$$\hat{\mathbf{\Sigma}}_\mathbf{r} = (W_1^T AV_1, W_1^T B, CV_1, D),$$

where there is no reference to V_2 and W_2. Naturally we need an estimate of the truncation error, i.e. an upper bound for the sum of the neglected Hankel singular values, but there is no need to compute the columns of V_2 and W_2. Since we are only interested in the cases where the Hankel singular values decay rapidly, this significantly reduces the computational burden.

This is the reason why we are interested in finding the rank k dominant eigenspaces of the matrices PQ and QP.

How can this be achieved? Suppose for a moment that P, and Q were explicitly available, and the integer k was given to us in advance. Then, we might simply run two instances of the standard subspace iteration algorithm, as specified in Figure 3.1

1: $V_0^T V_0 = I_k$	1: $W_1^T W_1 = I_k$
2: **for** $j = 0, 1, \ldots,$ **do**	2: **for** $j = 1, 2, \ldots,$ **do**
3: Set $F_{j+1} := (PQ)V_j$	3: Set $G_{j+1} := (QP)W_j$
4: QR-factorization $V_{j+1}R_{j+1} = F_{j+1}$	4: QR-factorization $W_{j+1}S_{j+1} = G_{j+1}$
5: **end for**	5: **end for**

Figure 3.1. Two instances of the standard subspace iteration. The starting indices are different by deliberate choice.

The convergence of the standard subspace iteration as well as Ritz acceleration is discussed in Chapter 4.

The two distinct instances of the subspace iteration can be merged into a single iterative scheme, which we state as Algorithm 2. It is not difficult to see that

$$\operatorname{Ran} V_{j+1} = \operatorname{Ran}(PQV_j), \quad \operatorname{Ran} W_{j+2} = \operatorname{Ran}(QPW_{j+1}), \quad j = 0, 1, 2, \ldots,$$

and

$$\operatorname{Ran} \tilde{V}_{j+1} = \operatorname{Ran}(PQ\tilde{V}_j), \quad \text{and} \quad \operatorname{Ran} \tilde{W}_{j+2} = \operatorname{Ran}(QP\tilde{W}_{j+1}), \quad j = 0, 1, 2, \ldots,$$

while

$$\operatorname{Ran} \tilde{W}_1 = \operatorname{Ran} Q\tilde{V}_0.$$

Now, given a matrix $\tilde{V}_0$ with orthonormal columns, it follows that if we choose V_0 and W_1 such that

$$\operatorname{Ran} V_0 = \operatorname{Ran} \tilde{V}_0, \quad \operatorname{Ran} W_1 = \operatorname{Ran} Q\tilde{V}_0,$$

then the two procedures are equivalent in the sense that

$$\operatorname{Ran} \tilde{V}_j = \operatorname{Ran} V_j, \quad \operatorname{Ran} W_j = \operatorname{Ran} \tilde{W}_j,$$

for $j = 1, 2, \ldots$.

In either case, there is a fundamental problem which we have yet to address, namely that the Gramians P, and Q are not explicitly available to us, but are given

1: Chose V_0 such that $\tilde{V}_0^T \tilde{V}_0 = I_k$.

2: **for** $i = 0, 1, \ldots,$ **do**

3: QR-factorization: $\tilde{W}_{i+1} \tilde{S}_{i+1} := Q \tilde{V}_i$

4: QR-factorization: $\tilde{V}_{i+1} \tilde{T}_{i+1} := P \tilde{W}_{i+1}$

5: **end for**

Algorithm 2: Simultaneous subspace iteration for PQ, and QP

implicitly as the solution of the corresponding Lyapunov equations. It follows that we must use these equations to approximate the actions of P and Q.

Hodel [24] was the first to consider this approach. He developed the Approximate Power Iteration (API) which can be used to compute the dominant eigenspaces for P. This algorithm is discussed in Chapter 7. We have extended Hodel's algorithm, and in Chapter 8 we show how to apply subspace iteration with Ritz acceleration. Sorensen and Zhou applied Hodel's basic idea to the simultaneous iteration. We discuss this approach in Chapter 9.

It is clear that the two Lyapunov equations can be solved independently. However, we are not particularly interested in the individual Gramians P and Q, but rather in the dominant eigenspaces for the products PQ and QP. Now, there are at least two algorithms which occasionally can be used to compute a good low rank approximation of P and a good low rank approximation of Q. Suppose we have succeeded beyond our wildest imagination and extracted the optimal rank k approximation P_k for P and the optimal rank k approximation Q_k for Q. Now, is the matrix P_kQ_k necessarily a good approximation of PQ? The answer is no! We construct an example for which P and Q are both symmetric positive definite, but $P_kQ_k = 0$, for all k below a certain threshold.

Example 3 Consider the following very general situation, where A, B, and C are partitioned conformally

$$A = \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & \\ & B_{22} \end{bmatrix}, \quad \text{and} \quad C = \begin{bmatrix} C_{11} & \\ & C_{22} \end{bmatrix}. \tag{3.9}$$

The dimension of A_{ii} will be denoted by n_i. In this situation the Gramians P and Q will both be block diagonal

$$P = \begin{bmatrix} P_{11} & \\ & P_{22} \end{bmatrix}, \quad Q = \begin{bmatrix} Q_{11} & \\ & Q_{22} \end{bmatrix},$$

where

$$A_{ii}P_{ii} + P_{ii}A_{ii}^T + B_{ii}B_{ii}^T = 0,$$
$$A_{ii}^T Q_{ii} + Q_{ii}A_{ii} + C_{ii}^T C_{ii} = 0,$$

for $i = 1, 2$. Now, given any pair of stable matrices A_{11} and A_{22} we are free to selectively scale the individual diagonal blocks of B and C, in a manner which will ensure that the rank k dominant eigenspace for P satisfies

$$D_k(P) \subseteq \text{span}\{e_1, e_2, \ldots, e_{n_1}\},$$

while the rank k dominant eigenspace for Q satisfies

$$D_k(Q) \subseteq \text{span}\{e_{n_1+1}, e_{n_1+2}, \ldots, e_{n_1+n_2}\}.$$

It is clear that $P_k Q_k = 0$ for all $k \leq n_1$.

A small specific example of the phenomenon is given by the matrices

$$A = \begin{bmatrix} -2 & 1 & & \\ 1 & -2 & & \\ & & -2 & 1 \\ & & 1 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 100 & \\ 10 & \\ & 10 \\ & 1 \end{bmatrix}, \quad \text{and} \quad C = \begin{bmatrix} 10 & 1 & & \\ & & 100 & 10 \end{bmatrix}.$$

This is not an isolated example and it easy to generate an entire family. Let A, B, and C be as before, i.e. as in equation (3.9). Let U be any orthogonal matrix. Now we apply the similarity transform defined by U^T, i.e. we replace A with $U^T AU$, B with $U^T B$, and C with CU. The transformed Gramians satisfy

$$\hat{P} = U^T PU \quad \text{and} \quad \hat{Q} = U^T QU,$$

which implies the optimal rank k approximations are given by

$$\hat{P}_k = U^T P_k U \quad \text{and} \quad \hat{Q}_k = U^T Q_k U,$$

where P_k and Q_k are the optimal rank k approximations of P and Q. Therefore, we still have the critical property

$$\hat{P}_k \hat{Q}_k = 0, \quad k \leq n_1,$$

despite the fact that $\hat{P}\hat{Q} = U^T(PQ)U$ is nonsingular. We conclude that while it is certainly possible to solve the Lyapunov equations independently, it is not necessarily a sound strategy to do so. Nevertheless, there are algorithms for model reduction which rely on low rank approximations for P and Q which are computed independently, see Penzl [42].

4. Subspace iteration

4.1 Introduction

The standard subspace iteration can be used to compute the dominant invariant subspace of a general nonsymmetric matrix. The associated Ritz acceleration is a standard scheme, which can be used to dramatically improve the rate of convergence of subspace iteration. In what follows, we only state the results for the symmetric positive definite case. This chapter provides the necessary background for Chapter 7 where we discuss the Approximate Power Iteration by A. S. Hodel [24], as well as an improved version of his algorithm, which we present in Chapter 8.

The subspace iteration is also known as simultaneous iteration or orthogonal iteration. Additional details can be found in [48, 61, 62, 67].

Let A be any n by n diagonalizable matrix, and let $\{\lambda_i\}_{i=1}^n$ its eigenvalues

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

and let $\{u_i\}_{i=1}^n$ be the corresponding eigenvectors, i.e.,

$$Au_i = \lambda_i u_i, \quad i = 1, 2, \ldots, n.$$

If $|\lambda_k| > |\lambda_{k+1}|$, then

$$D_k(A) = \operatorname*{span}_{\mathbb{R}}\{u_1, u_2, \ldots, u_k\}$$

is the dominant invariant subspace of A of rank k.

4.2 The power method

Let A be an n by n diagonalizable matrix with a dominant eigenpair (λ_1, u_1). Then we may use the basic power method to compute an approximation of this dominant eigenpair, as shown in Algorithm 3.

Input: An n by n diagonalizable matrix A, $v^{(0)} \in \mathbb{R}^n$, $\|v^{(0)}\|_2 = 1$.

Output: An approximation $(\lambda^{(j)}, v^{(j)})$ of the dominant eigenpair of A.

1: **for** $j = 1, 2, \ldots$ **do**

2: $\quad y^{(j)} = Av^{(j-1)}$

3: $\quad v^{(j)} = y^{(j)}/\|y^{(j)}\|_2$

4: $\quad \lambda^{(j)} = y^{(j)}Ay^{(j)}$

5: **end for**

Algorithm 3: The basic power method

If the initial guess $v^{(0)}$ is not orthogonal to u_1 and if $|\lambda_1| > |\lambda_2|$, then $v^{(j)}$ converges to an eigenvector for A corresponding to the eigenvalue λ_1. We are primarily interested in the case where $A = P$ is a symmetric matrix. We have the following theorem.

Theorem 4.2.1 *Let P be a symmetric matrix, and let $\{\lambda_i\}_{i=1}^n$ be the eigenvalues listed in the order of decreasing magnitude,*

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

and let $\{u_i\}_{i=1}^n$ be the the corresponding eigenvectors,

$$Pu_i = \lambda_i u_i, \quad i = 1, 2, \ldots, n.$$

Define $\theta_j \in [0, \pi/2]$ by

$$\cos(\theta_j) = |u_1^T v^{(j)}|.$$

If $\cos(\theta_0) \neq 0$ then

$$|\sin(\theta_j)| \leq \tan(\theta_0) \left|\frac{\lambda_2}{\lambda_1}\right|^j,$$
$$|\lambda^{(j)} - \lambda_1| \leq |\lambda_1 - \lambda_n| \tan(\theta_0) \left|\frac{\lambda_2}{\lambda_1}\right|^{2j}.$$

Proof The proof is given in Golub [15] and is omitted. ■

In practice it does not matter if the initial guess $v^{(0)}$ is orthogonal to u_1, because roundoff errors will introduce a component in the direction of u_1, which is then amplified by subsequent iterations until it becomes the dominant component.

4.3 The subspace iteration

This is a straight forward generalization of the power method which can be used to compute the rank k dominant eigenspace of A. The subspace iteration is given as Algorithm 4.

Input: An n by n diagonalizable matrix A, an n by k matrix V_0 with $V_0^T V_0 = I_k$.

Output: An approximation of the rank k dominant subspace of A

1: **for** $j = 0, 1, 2, \ldots$ **do**

2: $\quad Y_{j+1} = PV_j$

3: $\quad QR$-factorization: $V_{j+1} R_{j+1} = Y_{j+1}$

4: **end for**

Algorithm 4: The basic subspace iteration

The distance between a pair of subspaces of equidimensional subspace E_1, E_2 is given by

$$\text{dist}(E_1, E_2) = \|\Pi_1 - \Pi_2\|_2,$$

where Π_i is the orthogonal projector onto E_i, $i = 1, 2$.

As before, we are primarily interested in the case where $A = P$ is symmetric. We have the following theorem.

Theorem 4.3.1 *Let P be a symmetric matrix, and with eigenvalues $\{\lambda_i\}_{i=1}^n$, such that*

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

Assume that $|\lambda_{k+1}| < |\lambda_k|$, and that the n by k matrices Y_j are given by Algorithm 4. Let $\theta \in [0, \pi/2]$ be given by

$$\cos(\theta) = \min \left\{ \frac{|u^T v|}{\|u\|_2 \|v\|_2} \ : \ u \in D_k(P), v \in \text{Ran}(Y_0) \right\}.$$

If $\theta < \pi/2$ then

$$\text{dist}(D_k(P), \text{Ran}\, Y_j) \leq \tan(\theta) \left| \frac{\lambda_{k+1}}{\lambda_k} \right|^j.$$

Proof The proof is given in Golub [15] and is omitted. ■

4.4 Ritz acceleration

The basic subspace iteration converges rapidly if $|\lambda_{k+1}| \ll |\lambda_k|$. However, given a matrix A it is a nontrivial problem to find such a k and it may not even exist. Ritz acceleration uses $r \geq k$ vectors in order to compute an improved approximation of $D_k(A)$. The procedure is given as Algorithm 5.

If $|\lambda_k| > |\lambda_{k+1}|$, so that $D_k(A)$ is well defined, then it can be shown that the range of the leading k columns of V_j converge to $D_k(A)$ essentially as fast as $|\lambda_r/\lambda_k|^j \to 0$, as $j \to \infty$.

Input: An n by n diagonalizable matrix A, an n by r matrix V_0 with orthonormal columns

1: **for** $j = 1, 2, \ldots$ **do**

2: $W_j := AV_{j-1}$.

3: $S_j := V_{j-1}^T W_j$.

4: Schur decomposition: $S_j = U_j T_j U_j^T$, where the diagonal entries of the triangular matrix T_j are sorted in the order of decreasing magnitude.

5: $W_j := W_j U_j$.

6: QR-factorization: $V_j R_j = W_j$.

7: **end for**

Algorithm 5: Subspace iteration with Ritz acceleration

5. Krylov subspace methods for standard linear equations

5.1 Introduction

In this chapter, we provide an elementary introduction to Krylov subspace methods for solving a standard linear system

$$Ax = f,$$

where A is a nonsingular n by n real matrix and f is a vector in $\mathbb{R}^n$. We cover the basic ideas and definitions, we review some standard algorithms including their convergence theory, and discuss the concept of preconditioning. With the possible exception of Theorem 5.4.3, which is a minor generalization of a recent result by Philipe and Gamti [12] , the material presented here is very well known. We shall make frequent references to this chapter, when we apply Krylov subspace methods directly to Lyapunov equations in the Kronecker product form.

5.2 Krylov subspaces

Let A be a nonsingular n by n matrix, let f be a vector in $\mathbb{R}^n$ and consider the problem of solving the standard linear system

$$Ax = f,$$

for$x$ in $\mathbb{R}^n$. We have the following lemma.

Lemma 5.2.1 *There exists a unique real monic polynomial p, such that*

$$p(A)f = 0, \quad \textit{and} \quad q(A)f \neq 0,$$

for all real nonzero polynomials q of degree strictly less than p.

Proof By Cayley's theorem there exists a real polynomial p of degree at most n such that $p(A) = 0$. It follows that the set

$$D = \{k \in \mathbb{N} \,:\, \exists p \in \mathbb{P}_k \,:\, p \neq 0,\, p(A)f = 0\},$$

is not empty and has a smallest element $m \in D$. Let q be any nonzero polynomial of degree strictly less than m. Then $q(A)f \neq 0$ by the minimality of m. Now let p_1 and p_2 be real monic polynomials of degree m such that $p_i(A)f = 0$. Since p_1 and p_2 are monic, the degree of $p_1 - p_2$ is strictly less than m, and since $[(p_1 - p_2)(A)]\, f = 0$, it follows that $p_1 - p_2 = 0$ or equivalently $p_1 = p_2$. ■

Definition 5.2.1 *The minimal polynomial for f with respect to A is the monic polynomial p of minimal degree, such that*

$$p(A)f = 0.$$

The degree of p is called the grade of f with respect to A.

Now, let p be the minimal polynomial for f with respect to A and let m be the degree of p. We have

$$p(t) = \sum_{j=0}^{m} \alpha_j t^j, \quad t \in \mathbb{R},$$

where $\alpha_j \in \mathbb{R}$, $j = 0, 1, 2, \ldots, m$, and $\alpha_m = 1$. We claim that $\alpha_0 \neq 0$. If $\alpha_0 = 0$, then

$$p(t) = tq(t), \quad t \in \mathbb{R},$$

where q has degree $m - 1$, and since A is nonsingular, it follows that

$$q(A)f = 0,$$

which violates the minimality of m. Therefore $\alpha_0 \neq 0$, and the solution to

$$Ax = f,$$

is given by

$$x = \sum_{j=0}^{m-1} \beta_j A^j f,$$

where $\beta_j = -\frac{\alpha_j+1}{\alpha_0}$ for $j = 0, 1, 2, \ldots, m-1$. This is the main reason why we study the subspaces named in the following definition.

Definition 5.2.2 *Let j be a positive integer. The Krylov subspace $K_j(A, f)$ is given by*

$$K_j(A, f) = \operatorname*{span}_{\mathbb{R}}\{f, Af, A^2f, \ldots, A^{j-1}f\}.$$

Lemma 5.2.2 *Let m be the grade of f with respect to A, then*

$$K_j(A, f) \subset K_{j+1}(A, f), \quad j < m,$$

and

$$K_j(A, f) = K_m(A, f), \quad j = m.$$

Proof It is clear from the definition that the Krylov subspaces form an increasing sequence,

$$K_j(A, f) \subseteq K_{j+1}(A, f).$$

Let p be the minimal polynomial for f with respect to A. Let $j \geq m$. We claim that $K_j(A, f) = K_m(A, f)$. It is enough to show that $K_j(A, f) \subseteq K_m(A, f)$. Let $y \in K_j(A, f)$ be given, then $y = w(A)f$, where w is a polynomial of degree strictly less than j. By factoring w as follows,

$$w(x) = g(x)p(x) + r(x),$$

where the degree of the remainder r is strictly less than m, we see that

$$y = w(A)f = g(A)p(A)f + r(A)f = r(A)f,$$

is really a member of $K_m(A, f)$, and we conclude that $K_j(A, f) = K_m(A, f)$. Now we claim that for $j < m$, we have

$$K_j(A, f) \subset K_{j+1}(A, f).$$

Let $j < m$. By definition $A^j f \in K_{j+1}(A, f)$, but we claim that $A^j f \notin K_j(A, f)$. Suppose, $A^j f \in K_j(A, f)$, then $A^j f = q(A)f$ for a polynomial of degree at most $j-1$. But, then $[A^j - q(A)]\, f = 0$, contradicting the definition of m. We conclude that $K_j(A, f) \subset K_{j+1}(A, f)$. ■

Corollary 5.2.1 *Let m be the grade of f with respect to A, then $K_m(A, f)$ is the smallest A invariant vector space.*

Now, given the fact that the solution of $Ax = f$ is a member of $K_m(A, f)$, it is only natural that we try to approximate x with $x_j \in K_j(A, f)$. This requires a reliable way of computing a basis for $K_j(A, f)$. By definition,

$$\{f, Af, A^2f, \ldots, A^{j-1}f\},$$

is a basis for $K_j(A, f)$, but this basis can be very ill conditioned. If A has a dominant eigenvalue, then the angle between $A^{j-1}f$ and the corresponding eigenvector will tend to zero as j tends to infinity, unless f is deficient in that particular direction. In practice, roundoff errors will introduce a component in the direction of the dominant eigenvector. It is clear that we require a systematic way of computing a well conditioned basis for $K_j(A, f)$.

5.3 The Arnoldi algorithm

The Arnoldi algorithm uses the Gram-Schmidt process to compute an orthonormal basis for $K_j(A, f)$. It is well known that the classical Gram-Schmidt algorithm is numerically unstable, and this problem can be cured partially by using the modified Gram-Schmidt method and/or reorthogonalization.

Mathematically, the four possible algorithms are equivalent. However, the modified Gram-Schmidt is much more reliable than the classical Gram-Schmidt process. Reorthogonalization improves the quality of the basis in either case and if reorthogonalization is applied, then there is virtually no difference between the classical and the modified scheme. The difference between the four variants have been investigated by Giroud, Langlou, and Rozlosnik [14].

The Arnoldi algorithm with the modified Gram-Schmidt method is stated as Algorithm 6.

1: $v_1 := f/\|f\|_2$
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $\quad w_j := Av_j$
4: $\quad$ **for** $i = 1, \ldots, j$ **do**
5: $\quad\quad h_{ij} := v_i^T w_j$
6: $\quad\quad w_j := w_j - v_i h_{ij}$
7: $\quad$ **end for**
8: $\quad h_{j,j+1} := \|w_j\|_2$
9: $\quad$ **if** $h_{j,j+1} = 0$ **then**
10: $\quad\quad v_{j+1} := 0$
11: $\quad\quad k := j$
12: $\quad\quad$ **exit**
13: $\quad$ **else**
14: $\quad\quad v_{j+1} := w/\tilde{h}_{j,j+1}$
15: $\quad$ **end if**
16: **end for**

Algorithm 6: Arnoldi algorithm with modified Gram-Schmidt (MGS)

Let m be the grade of f with respect to A. Let $k \geq m$, then the Arnoldi algorithm terminates after exactly m steps, and produces a factorization of the form

$$AV_m = V_m H_m,$$

where

$$V_m = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix},$$

is a matrix with orthonormal columns spanning $K_m(A, f)$, and

$$H_m = \begin{bmatrix} h_{11} & h_{12} & \dots & \dots & h_{1m} \\ h_{21} & h_{22} & \dots & \dots & h_{2m} \\ & h_{32} & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \end{bmatrix},$$

is an m by m upper Hessenberg matrix.

If $k < m$, then the Arnoldi algorithm terminates after exactly k steps, and produces a factorization of the form

$$AV_k = V_{k+1}\bar{H}_k,$$

where

$$V_k = \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix},$$

consists of the first k columns of V_m, which span $K_k(A, f)$, and

$$\bar{H}_k = \begin{bmatrix} h_{11} & h_{12} & \dots & \dots & h_{1k} \\ h_{21} & h_{22} & \dots & \dots & h_{2k} \\ & h_{32} & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & h_{k,k-1} & h_{kk} \\ & & & & h_{k+1,k} \end{bmatrix},$$

consists of the $k+1$ by k upper left corner of H_m.

5.4 Standard Krylov subspace methods

In this section we briefly review four standard Krylov subspace methods: GMRES, CG, CGNR, and BCG. We state the algorithms explicitly and comment on their convergence. We will refer to these algorithms in Chapter 10.

Saad [51] has written a good introduction to iterative methods for linear systems, which contains several chapters on Krylov subspace methods. Simoncini and Szyld [57] have also written a survey paper on such methods.

5.4.1 GMRES

The GMRES method is due to Saad and Schultz [49]. It can be used to solve a standard linear system

$$Ax = f.$$

Given an initial guess x_0, the GMRES algorithm forms the initial residual

$$r_0 = f - Ax_0$$

and constructs the Krylov subspaces

$$K_j(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{j-1}r_0\}.$$

For each j the algorithm computes an approximate solution

$$x_j \in x_0 + K_j(A, r_0)$$

such the 2-norm of the corresponding residual

$$r_j = f - Ax_j$$

is minimized, i.e.,

$$\|r_j\|_2 = \min\{\|f - Ax\|_2 \ : \ x \in x_0 + K_j(A, r_0)\}.$$

The basic procedure is given as Algorithm 7. For the sake of brevity we use the modified Gram-Schmidt orthogonalization scheme.

1: $r_0 = f - Ax_0,\ \beta := \|r_0\|_2,\ v_1 := r_0/\beta.$
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $w_j := Av_j$
4: **for** $i = 1, \ldots, j$ **do**
5: $h_{ij} := v_i^T w_j$
6: $w_j := w_j - v_i h_{ij}$
7: **end for**
8: $h_{j,j+1} := \|w_j\|_2$
9: **if** $h_{j,j+1} = 0$ **then**
10: k:=j
11: Goto 16
12: **else**
13: $v_{j+1} := w/h_{j,j+1}$
14: **end if**
15: **end for**
16: Solve the linear least squares problem $\bar{H}_k y_k = \beta e_1$ for $y_k \in \mathbb{R}^k$
17: Set $x_k = x_0 + V_k y_k$.

Algorithm 7: GMRES algorithm with modified Gram-Schmidt (MGS)

Let m be the grade of r_0 with respect to A. In exact arithmetic, the residuals returned by GMRES satisfy

$$\|r_1\|_2 \geq \|r_2\|_2 \geq \cdots \geq \|r_m\|_2 = 0,$$

because

$$K_1(A, r_0) \subseteq K_2(A, r_0) \subseteq, \ldots, \subseteq K_m(A, r_0),$$

and $x - x_0 = A^{-1}r_0 \in K_m(A, r_0)$.

A general convergence theory of GMRES is still lacking and it is the topic of current research. In particular, Greenbaum, Pták, and Strakoš [17] have shown that any non-increasing residual curve is possible, and Embree [11] discusses different types of bounds.

We need a few theorems, which we shall use later when we discuss the nature of good preconditioners.

Lemma 5.4.1 *The residuals produced by the GMRES algorithm satisfy*

$$\|r_j\|_2 = \min\{\|p(A)r_0\|_2 \ : \ p \in \mathbb{P}_j, \ p(0) = 1\},$$

where $\mathbb{P}_j$ denotes the set of polynomials of degree at most j.

Proof The proof follows directly from the definition of GMRES. ■

We now consider a very specific family of polynomials. Let A be any n by n matrix, such that $\|I - A\|_2 < 1$. Then A is nonsingular, and

$$A^{-1} = \sum_{j=0}^{\infty} (I - A)^j,$$

from which it follows, that

$$I - A \sum_{j=0}^{\infty} (I - A)^j = 0.$$

We now define ω_k to be the polynomial given by

$$\omega_k(x) = 1 - x \sum_{j=0}^{k-1} (1 - x)^j. \tag{5.1}$$

Then $\omega_k \in \mathbb{P}_k$ and $\omega_k(0) = 1$. In addition we have

$$0 = \underbrace{I - A\sum_{j=0}^{k-1}(I-A)^j}_{\omega_k(A)} - A\sum_{j=k}^{\infty}(I-A)^j,$$

which implies

$$\|\omega_k(A)\|_2 \leq \frac{\|A\|_2}{1-\|I-A\|_2}\|I-A\|_2^k.$$

If we define $\rho = \|I-A\|_2$, and estimate $\|A\|_2 \leq \|I\|_2 + \|I-A\|_2$, then

$$\|\omega_k(A)\|_2 \leq \frac{1+\rho}{1-\rho}\rho^k. \tag{5.2}$$

The following two theorems are both immediate consequences of this discussion.

Theorem 5.4.1 *If A is any matrix such that $\rho = \|I-A\|_2 < 1$, and if r_k is the residual after k steps of the GMRES algorithm, then*

$$\|r_k\|_2 \leq \frac{1+\rho}{1-\rho}\rho^k\|r_0\|_2.$$

Theorem 5.4.2 *Let A be a diagonalizable matrix, and let $AC = C\Lambda$ where C is nonsingular and $\Lambda = diag\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, is diagonal. If $\rho = \max|1-\lambda_j| < 1$, and if r_k is the residual after k steps of the GMRES algorithm, then*

$$\|r_k\|_2 \leq \kappa_2(C)\frac{1+\rho}{1-\rho}\rho^k\|r_0\|_2,$$

where $\kappa_2(C) = \|C\|_2\|C^{-1}\|_2$ is the condition number of C with respect to the 2-norm.

Proof Let p be any polynomial, then $p(A) = Cp(\lambda)C^{-1}$ from which it follows

$$\|p(A)\|_2 \leq \|C\|_2\|p(\Lambda)\|_2\|C^{-1}\|_2 = \kappa_2(C)\|p(\Lambda)\|_2,$$

which together with Lemma 5.4.1 implies

$$\|r_k\|_2 \leq \|\omega_k(A)r_0\|_2 \leq \|\omega_k(A)\|_2\|r_0\|_2 \leq \kappa_2(C)\|\omega_k(\Lambda)\|_2\|r_0\|_2,$$

and the proof is completed by using (5.2). ■

In addition we have the following lemma.

Lemma 5.4.2 *Let A be any n by n matrix and let $f \in \mathbb{R}^n$. If $I - A$ has rank k, then the grade of f with respect to A is at most $k + 1$.*

Proof If A is any n by n matrix and $f \in \mathbb{R}^n$ then

$$K_j(A, f) = K_j(I - A, f) \subseteq \underset{\mathbb{R}}{\text{span}}\{f\} + \text{Ran}(I - A),$$

for all $j = 1, 2, \ldots$. Now, let m be the grade of f with respect to A. Then

$$K_m(A, f) \subseteq \underset{\mathbb{R}}{\text{span}}\{f\} + \text{Ran}(I - A)$$

which implies that the dimension of $K_m(A, f)$ is at most $k + 1$. Since

$$\{A^j f : \ j = 0, 1, 2, \ldots, m - 1\}$$

is a basis for $K_m(A, f)$ we conclude that $m \leq k + 1$. ■

Corollary 5.4.1 *If A is a nonsingular matrix which is a rank k perturbation of the identity matrix, then GMRES will converge in at most k iterations.*

Proof Let $f \in \mathbb{R}^n$. By the previous lemma, the grade m of f with respect to A is at most $k + 1$. The GMRES algorithm therefore constructs an orthonormal basis for $K_m(A, f)$ and converges after at most k applications of the matrix A. ■

There is another case where the residual can be estimated. The case of $\rho = 1$ is due to Gamti and Philippe [12]. The following theorem is a straightforward generalization of their result.

Theorem 5.4.3 *Let A be a nonsingular and diagonalizable matrix with $AC = C\Lambda$, where C is nonsingular and $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is diagonal.*

Now suppose, that there exists an integer k and a $\rho < 1$, such that

$$|1 - \lambda_j| \geq \rho, \quad j = 1, 2, \ldots, k$$

and

$$|1 - \lambda_j| < \rho, \quad j = k+1, k+2, \ldots, n.$$

If r_m is the residual after $m \geq k$ steps of the GMRES algorithm, then

$$\|r_m\|_2 \leq \kappa_2(C)\|p_k(\Lambda)\|_2 \frac{1+\rho}{1-\rho}\rho^{m-k}\|r_0\|_2,$$

where p_k is the polynomial of degree k given by

$$p_k(\lambda_j) = 0, \quad j = 1, 2, \ldots, k$$

and $p_k(0) = 1$.

Proof First, we notice that the polynomial p_k is well defined, because A is nonsingular or equivalently $\lambda_j \neq 0$ for all j. Specifically, we have

$$p_k(x) = \prod_{j=1}^{k} \left(1 - \frac{x}{\lambda_j}\right).$$

Now, let $m \geq k$ and consider the polynomial

$$q_m(x) = p_k(x)\omega_{m-k}(x),$$

where ω_{m-k} is defined by equation (5.1). It is clear that $q_m(0) = 1$ and as in the proof of Theorem 5.4.2 we have

$$\|r_m\|_2 \leq \|q_m(A)r_0\|_2 \leq \kappa_2(C)\|q_m(\Lambda)\|_2\|r_0\|_2,$$

which is why we must derive an estimate for $\|q_m(\Lambda)\|_2$. We have

$$\|q_m(\Lambda)\|_2 = \max_{\lambda \in \sigma(A)} |q_m(\lambda)|.$$

However, by definition

$$q_m(\lambda_j) = p_k(\lambda_j)\omega_{m-k}(\lambda_j) = 0, \quad j = 1, 2, \ldots, k,$$

which implies

$$\|q_m(\Lambda)\|_2 = \max_{j>k} |q_m(\lambda_j)|.$$

It follows immediately, that

$$\|q_m(\Lambda)\|_2 \leq \left(\max_{j>k}|p_k(\lambda_j)|\right)\left(\max_{j>k}|\omega_{m-k}(\lambda_j)|\right).$$

Now, since $|1-\lambda_j| < \rho$ for $j > k$, we have

$$|\omega_{m-k}(\lambda_j)| \leq \left(\frac{1+\rho}{1-\rho}\right)\rho^{m-k}, \quad j > k.$$

Finally, we conclude

$$\|q_m(\Lambda)\|_2 \leq \|p_k(\Lambda)\|_2 \left(\frac{1+\rho}{1-\rho}\right)\rho^{m-k}$$

and as a result

$$\|r_m\|_2 \leq \kappa_2(C)\|p_k(\Lambda)\|_2 \left(\frac{1+\rho}{1-\rho}\right)\rho^{m-k}\|r_0\|_2,$$

which completes the proof. ■

Gamti and Philippe used the case of $\rho = 1$ to explain why GMRES converges even when some eigenvalues of $I - A$ lie outside the unit disk. Their results shows if $I - A$ has k eigenvalues outside the unit disk, then the convergence is delayed by k iterations, but the asymptotic rate is the same as if all the eigenvalues were inside the unit disk. We have merely generalized their result to a disk with radius $\rho < 1$.

We emphasize that these theorems do not imply that GMRES converges rapidly, because the condition number, $\kappa_2(C)$, can easily be so large that the bounds are worthless. However, if $\|I - A\|_2 < 1$ or if A is nonsingular and $I - A$ is of low rank, then GMRES converges rapidly to the solution.

5.4.2 CG

The conjugate gradient algorithm is due to Lanczos [31], as well as Hestenes and Stiefel [20]. The algorithm applies to

$$Ax = f,$$

where A is a symmetric positive definite matrix. The standard formulation of the method is given as Algorithm 8.

1: Compute $r_0 := f - Ax_0$, $p_0 = r_0$.
2: **for** $= 0, 1, 2, \ldots,$ until convergence **do**
3: $\quad \alpha_j := (r_j, r_j)/(Ap_j, p_j)$
4: $\quad x_{j+1} := x_j + \alpha_j Ap_j$
5: $\quad \beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$
6: $\quad p_{j+1} := r_{j+1} + \beta_j p_j$
7: **end for**

Algorithm 8: Standard CG for $Ax = f$

In order to discuss the convergence of CG it is convenient to introduce the A-norm, which is given by

$$\|x\|_A = \sqrt{(x, Ax)}.$$

If A is symmetric positive definite, then $\|\cdot\|_A$ is a norm, and

$$\lambda_{\min}\|x\|_2 \leq \|x\|_A \leq \lambda_{\max}\|x\|_2.$$

The following lemma characterizes the approximations obtained from the CG algorithm.

Lemma 5.4.3 (Saad [51]) *Let x_j be the approximate solution obtained after j steps of the CG algorithm, and let x be the exact solution, then x_m is of the form*

$$x_j = x_0 + q_j(A)r_0,$$

where q_j is a polynomial of degree $j-1$ such that

$$\|x - x_j\|_A = \|(I - Aq_j(A))(x - x_0)\|_A = \min_{q \in \mathbb{P}_{j-1}} \|(I - Aq(A))(x - x_0)\|_A$$

It follows immediately that if A has k distinct eigenvalues, then the CG algorithm converges using at most k iterations.

The following lemma can be used to estimate the A-norm of the error.

Lemma 5.4.4 (Saad [51]) *Let A be symmetric positive definite and consider the linear system $Ax = f$. If x_* is the true solution, and if x_j is the j'th approximation obtained from the CG algorithm, then*

$$\|x_* - x_j\|_A \leq 2\left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right]^j \|x_* - x_0\|_A.$$

We see that if A is a well conditioned symmetric positive definite matrix, then the CG algorithm will converge rapidly.

5.4.3 CGNR

The CGNR algorithm is given as Algorithm 9. It can be used to solve a general linear system

$$Ax = f,$$

where A is a nonsingular matrix with respect to x.

1: Compute $r_0 := f - Ax_0$, $z_0 := A^T r_0$, $p_0 := z_0$.
2: **for** $i = 0, 1, 2, \ldots,$ until convergence **do**
3: $\quad w_i = Ap_i$
4: $\quad \alpha_i = \|z_i\|_2^2 / \|w_i\|_2^2$
5: $\quad x_{i+1} = x_i + \alpha_i p_i$
6: $\quad z_{i+1} = A^T r_{i+1}$
7: $\quad \beta_i = \|z_{i+1}\|_2^2 / \|z_i\|_2^2$
8: $\quad p_{i+1} = z_{i+1} + \beta_i p_i$
9: **end for**

Algorithm 9: Standard CGNR for $Ax = f$

The approximation x_j is such that the corresponding residual

$$r_j = f - Ax_j,$$

satisfies

$$\|r_j\|_2 = \min\{\|f - Ax\|_2 \ : \ x \in x_0 + K_j(A^T A, A^T r_0)\}.$$

This expression is very similar to GMRES, the only difference is the subspace in which the residual is minimized.

The CGNR algorithm is essentially the CG algorithm applied to the normal equations,

$$A^T Ax = A^T f,$$

with the important difference that the CGNR algorithm calculates the true residual r_j, as well as the residual z_j for the normal equations. CGNR inherits its convergence properties from CG.

5.4.4 BCG

The biconjugate gradient algorithm, Algorithm 10, can be used to solve a pair of adjoint equations,

$$Ax = f, \tag{5.3}$$

$$A^T x = g. \tag{5.4}$$

Compute $r_0 := f - Ax_0$, $r_0^* = g - A^T x_0^*$
for $j = 0, 1, \ldots,$ until convergence **do**
  $\alpha_j := (r_j, r_j^*)/(Ap_j, p_j^*)$
  $x_{j+1} := x_j + \alpha_j p_j$
  $x_{j+1}^* := x_j^* + \alpha_j p_j^*$
  $r_{j+1} := r_j - \alpha_j A p_j$
  $r_{j+1}^* := r_j^* - \alpha_j A^T p_j^*$
  $\beta_j := (r_{j+1}, r_{j+1}^*)/(r_j, r_j^*)$
  $p_{j+1} := r_{j+1} + \beta_j p_j$
  $p_{j+1}^* := r_{j+1}^* + \beta_j p_j^*$
end for

Algorithm 10: BCG for solving a pair of adjoint equations.

The BCG algorithm reduces to the conjugate gradient algorithm in the special case where $A = A^T$ is a symmetric positive definite matrix and $f = g$.

The algorithm terminates prematurely if $(r_j, r_j^*) = 0$ for any value of j. There is very little information on the convergence of BCG. The algorithm does not necessarily converge, and the residual history may be very erratic, and it is absolutely necessary to find a good preconditioner.

The BCG algorithm caught our attention because we want to solve a pair of Lyapunov equations

$$AP + PA^T + BB^T = 0,$$
$$A^TQ + QA + C^TC = 0,$$

which are mathematically equivalent to a pair of standard linear equations, specifically

$$\tilde{A}\text{vec}(X) + \text{vec}(BB^T) = 0,$$
$$\tilde{A}^T\text{vec}(Y) + \text{vec}(C^TC) = 0,$$

where $\tilde{A} = I \otimes A + A \otimes I$. Properties of the Kronecker product and the vec operator are given in Chapter 2.

5.5 Preconditioning

Preconditioning is the attempt to replace the original linear system $Ax = f$ with an equivalent system

$$M^{-1}Ax = M^{-1}Ax,$$

such that the properties of the new matrix $M^{-1}A$ are more favorable than those of A. In the case of Krylov subspace methods, the ideal preconditioner is a linear map M such that

$$My = g,$$

can be solved efficiently for all g, and such that M^{-1} is a good approximation of A^{-1}. Good preconditioners are critical to the success of Krylov subspace methods.

In view of the previous section we prefer to obtain an M such that

$$\|I - M^{-1}A\|_2 \ll 1,$$

or $I - M^{-1}A$ is of low rank, because in these cases the GMRES algorithm will converge after only a few iterations.

Given a nonsingular matrix A it is a nontrivial exercise to compute a good preconditioner and there are many different techniques. A good introduction to the topic can be found in Saad [51].

We will treat the Lyapunov matrix equation

$$AX + XA^T + BB^T = 0,$$

as a standard linear equation

$$(I \otimes A + A \otimes I)\mathrm{vec}(X) + \mathrm{vec}(BB^T) = 0,$$

in n^2 variables. It is possible to overcome the $O(n^2)$ storage requirement, but most of the preconditioners for standard linear systems cannot be applied simply because we require at least n^2 words of memory in order to represent a general nonsingular linear map on $\mathbb{R}^{n^2}$.

6. Current numerical methods for the Lyapunov equation

6.1 Introduction

In this chapter we briefly review the standard methods for the solution of a single Lyapunov equation

$$AX + XA^T + Q = 0,$$

with special emphasis on the case $Q = BB^T$, where B is a tall matrix.

The algorithms are divided into two groups, namely those which are suitable for small dense problems, and those which are suitable for large sparse problems.

6.2 Methods for small dense problems

In this section we list the standard methods for small and dense Sylvester and Lyapunov equations.

6.2.1 Bartels-Stewart's method

Bartels-Stewart's method [3] can be used to solve a general Sylvester equation of the form

$$AX - XB = C,$$

where A is an n by n matrix, B is a m by m matrix, and C is an n by m matrix. If

$$\sigma(A) \cap \sigma(B) = \emptyset,$$

then there is a unique solution for every choice of C.

The basic idea is very simple. By Schur's lemma every matrix is unitarily similar to a triangular matrix, and we have unitary matrices U, and V, such that

$$A = URU^*,$$
$$B = VSV^*,$$

where R is upper triangular and S is lower triangular. Then

$$RY - YS = T,$$

where $T = U^*CV$. Now let y_j and t_j denote the j'th columns of Y and T respectively. Then the equation for y_j is of the form

$$Ry_j - \sum_{i=j}^{m} y_i s_{ij} = t_j, \quad j = 1, 2, \ldots, m,$$

or equivalently

$$(R - s_{jj}I)y_j = t_j + \sum_{i=j+1}^{m} y_i s_{ij}, \quad j = 1, 2, \ldots, m.$$

The matrices $R - s_{jj}I$ are all nonsingular and upper triangular, and we can compute y_j by substitution as soon as $y_m, y_{m-1}, y_{m-2}, \ldots, y_{j+1}$ have been obtained. Finally, the solution of the original problem is given by

$$X = UYV^*.$$

The serial algorithm has been improved by Sorensen and Zhou [59] and requires $O(n^3 + m^3)$ arithmetic operations. A parallel version of Bartels-Stewart's algorithm has been developed by Hodel [24].

There is a common variation of Bartels-Stewart's algorithm which can be used when A is a large sparse matrix, and B is a small dense matrix. In this case we only transform B to lower triangular form S, i.e.

$$B = VSV^*.$$

The new equation is

$$AY - YS = T,$$

where $T = CV$. The j'th column, y_j, of Y is given by

$$(A - s_{jj}I)y_j = t_j + \sum_{i=j+1}^{m} y_i s_{ij}, \quad j = 1, 2, \ldots, m,$$

but the situation is considerably more complicated than before because $A - s_{jj}I$ is not necessarily a triangular matrix. It is necessary to solve m linear systems each with a different coefficient matrix $A - s_{jj}I$. Even in the simple case in which A is a narrow banded matrix, we still have to perform m distinct LU factorizations.

6.2.2 The Hessenberg-Schur method

The Hessenberg-Schur method applies to the Sylvester equation

$$AX - XB = C,$$

where A is an n by n matrix, B is a m by m matrix, and C is an n by m matrix. It is designed to handle the situation where n is much larger than m. The main idea is to transform A to upper Hessenberg form H, and to transform B to lower triangular form S, i.e.

$$A = UHU^*, \quad B = VSV^*.$$

The new system is

$$HY - YS = T,$$

where $T = U^*CV$. Let y_j and t_j be the j'th column for Y and T respectively. The equation for the j'th column of Y is of the form

$$(H - s_{jj}I)y_j = t_j + \sum_{i=j+1}^{m} y_i s_{ij}, \quad j = 1, 2, \ldots, m.$$

These systems are solved one a at time using Gaussian elimination with partial pivoting, a process which is inexpensive for Hessenberg matrices. The number of arithmetic operations is $O(n^3 + m^3)$, which is the same as in Bartels-Stewart's algorithm, but the Hessenberg-Schur method requires less arithmetic, when $n \gg m$.

It is clear that reducing A to the upper Hessenberg form H destroys the sparsity of A and requires $O(n^2)$ memory locations to store H. This limits the method to relatively small problems.

6.2.3 Hammarling's method

Hammerling's method [19] applies exclusively to Lyapunov equations in which the inhomogeneous term is symmetric positive semidefinite, i.e. Lyapunov equations of the form

$$AX + XA^T + BB^T = 0.$$

The algorithm first transforms A to lower triangular form, and then solves recursively for the Cholesky factorization of the solution X. The algorithm return a lower triangular matrix G, such that $X = GG^*$. It can be shown that this is exactly what we need in model reduction by balanced truncation of a relatively small system. It is possible to avoid complex arithmetic by transforming A to real Schur form. In either case, the algorithm requires $O(n^3)$ arithmetic operations. The serial algorithm has been improved by Sorensen and Zhou [59]. Hammarling's method has been parallelized by Hodel [24].

6.2.4 Matrix sign function iteration

Let A be a diagonalizable matrix

$$A = V\Lambda V^{-1},$$

where $\Lambda = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. If A does not have any eigenvalues on the imaginary axis, then the matrix sign function of A is defined as follows

$$\mathrm{sign}(A) = VDV^{-1},$$

where $D = \text{diag}\{d_1, d_2, \ldots, d_n\}$ and

$$d_i = \begin{cases} 1 & \text{Re}(\lambda_i) > 0, \\ -1 & \text{Re}(\lambda_i) < 0. \end{cases}$$

The following iteration can be used to compute the matrix sign function,

$$Z_0 = A, \quad Z_{k+1} = \frac{1}{2}(Z_k + Z_k^{-1}), \quad k = 0, 1, \ldots.$$

It can be shown that $Z_k \to \text{sign}(A)$ as $k \to \infty$. This is of one of the topics covered in Chapter 5 of the recent book by Higham [22].

Roberts [46] used the matrix sign function iteration to solve the Sylvester equation

$$AX - XB = C,$$

where A is a stable n by n matrix and B is a stable m by m matrix. The key is to notice, that the matrices

$$\begin{bmatrix} A & C \\ & -B \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A & \\ & -B \end{bmatrix},$$

are similar. Specifically, we have

$$\begin{bmatrix} A & C \\ & -B \end{bmatrix} = \begin{bmatrix} I & X \\ & I \end{bmatrix} \begin{bmatrix} A & \\ & -B \end{bmatrix} \begin{bmatrix} I & -X \\ & X \end{bmatrix}.$$

It follows that

$$\text{sign}\left(\begin{bmatrix} A & C \\ & -B \end{bmatrix}\right) = \begin{bmatrix} I & X \\ & I \end{bmatrix} \text{sign}\left(\begin{bmatrix} A & \\ & -B \end{bmatrix}\right) \begin{bmatrix} I & -X \\ & X \end{bmatrix},$$

and since A and B are both stable matrices, we have

$$\text{sign}\left(\begin{bmatrix} A & \\ & -B \end{bmatrix}\right) = \begin{bmatrix} I_n & \\ & -I_m \end{bmatrix}.$$

Therefore

$$\text{sign}\left(\begin{bmatrix} A & C \\ & -B \end{bmatrix}\right) = \begin{bmatrix} I_n & 2X \\ & -I_m \end{bmatrix}.$$

Now if

$$Z_0 = \begin{bmatrix} A & C \\ & -B \end{bmatrix}, \quad \text{and} \quad Z_{k+1} = \frac{1}{2}(Z_k + Z_k^{-1}), \quad k = 0, 1, \ldots,$$

then we have

$$Z_k = \begin{bmatrix} A_k & C_k \\ & -B_k \end{bmatrix},$$

where

$$A_k = \frac{1}{2}(A_k + A_k^{-1}), \quad B_k = \frac{1}{2}(B_k + B_k^{-1}), \quad C_{k+1} = \frac{1}{2}(C_k + A_k^{-1} C_k B_k^{-1}).$$

It is important to notice that we are explicitly inverting both A_k and B_k. This drives the number of arithmetic operations up to $O(n^3)$ for each iteration, because even when $A_0 = A$ is a sparse matrix, A_k will almost certainly be a dense matrix for $k > 0$.

Benner [5] applied the matrix sign function iteration to the special case where the inhomogeneous term C has a low rank representation, i.e. $C = EF^T$. This allowed him to represent C_k in the form

$$C_k = E_k F_k^T,$$

where the number of columns in E_k and F_k was reduced by routinely performing a tall SVD. However, the expensive operation of inverting A_k and B_k remains.

Grasedyck, Hackbusch and Khoromskij [16] have implemented the matrix sign function iteration using hierarchical matrix arithmetic. This allows them to solve certain Lyapunov equations using $O(n \log^q n)$ arithmetic operations where $q > 0$, which is nearly optimal. However, the necessary hierarchical representation is limited to matrices which arise from the discretization of certain elliptic differential equations. A very thorough introduction to hierarchical matrix arithmetic has been written by Börm, Grasedyck and Hackbusch [7].

Baur and Benner [4] made the natural combination of these two ideas using both hierarchical matrix arithmetic and the low rank representation of C_k.

6.3 Methods for large sparse problems

In this section we list the standard methods for Lyapunov equations where A is a large sparse matrix. These methods are all iterative in nature.

We begin by reviewing the standard Krylov subspace methods for solving Lyapunov equations. It is necessary to introduce the block Arnoldi algorithm which is an extension of the basic Arnoldi algorithm discussed in Chapter 5.

6.3.1 The block Arnoldi algorithm

We begin with the following definition.

Definition 6.3.1 *Let A be an n by n matrix, let B be an n by p matrix, and let j be a positive integer. The block Krylov subspace $K_j(A, B)$ is defined by*

$$K_j(A, B) = \text{Ran}\begin{bmatrix} B & AB & A^2B & \dots & A^{j-1}B \end{bmatrix} \subseteq \mathbb{R}^n,$$

i.e. $K_j(A, B)$ is the range of the linear map $\phi : \mathbb{R}^{jp} \to \mathbb{R}^n$ given by

$$\phi(x) = \begin{bmatrix} B & AB & A^2B & \dots & A^{j-1}B \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{jp} \end{bmatrix}.$$

It is clear that the block Krylov subspaces form an increasing sequence

$$K_j(A, B) \subseteq K_{j+1}(A, B), \quad j \in \mathbb{N},$$

and there exists an m such that

$$K_j(A, B) = K_m(A, B)$$

for all $j \geq m$. This observation justifies the following definition.

Definition 6.3.2 *The smallest m such that $K_j(A, B) = K_m(A, B)$ for all $j \geq m$ is called the grade of B with respect to A.*

The block Arnoldi algorithm can be used to compute an orthonormal basis for the block Krylov subspaces $K_j(A, f)$. One version of the block Arnoldi algorithm is given as Algorithm 11. If we run $k \leq m$ steps of Algorithm 11, then we obtain a matrix V_k given by

$$V_k = \begin{bmatrix} Q_1 & Q_2 & \dots & Q_k \end{bmatrix},$$

with orthonormal columns spanning $K_k(A, B)$. If we define H_k as the upper block Hessenberg matrix given by

$$H_k = \begin{bmatrix} H_{11} & H_{12} & \dots & \dots & H_{1k} \\ H_{21} & H_{22} & \dots & \dots & \vdots \\ 0 & H_{32} & H_{33} & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H_{k,k-1} & H_{kk} \end{bmatrix},$$

then

$$AV_k = V_k H_k + Q_{k+1} H_{k+1,k} E_k^T, \quad k < m,$$

where E_k consists of the last p_k columns of the n_k by n_k identity matrix, and

$$AV_m = V_m H_m, \quad k = m.$$

The numbers p_k and n_k are computed by the algorithm in order to keep track of the partitioning of the matrices V_k and H_k. Specifically, V_k is an n by n_k matrix and H_{ij} is a p_i by p_j matrix.

If we attempt to run $k > m$ steps of Algorithm 11, then it terminates normally after $k = m$ steps.

The subdiagonal blocks $H_{i+1,i}$ are not necessarily upper triangular, but are column permutations of upper triangular matrices. It is important to notice that any rank degradation in B as well as the intermediate block vectors W_j is reflected in the block structure of V_m and H_m, specifically we have

$$p_1 = \text{rank}(B),$$

1: Compute

$$BP = \begin{bmatrix} \hat{Q}_1 & \hat{Q}_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}, \tag{6.1}$$

using a rank revealing QR algorithm with column pivoting.

2: Set $p_1 := \operatorname{rank} B \leq p$, and $Q_1 := \hat{Q}_1$.

3: **for** $j = 1, 2, \ldots, k$ **do**

4: Set $V_j := \begin{bmatrix} Q_1, Q_2, \ldots, Q_j \end{bmatrix}$, and $n_j := p_1 + p_2 + \cdots + p_j$.

5: $W_j := AQ_j$

6: **for** $i = 1, 2, \ldots, j$ **do**

7: $H_{ij} := Q_i^T W_j$

8: $W_j := W_j - Q_i H_{ij}$

9: **end for**

10: **if** $W_j = 0$ **then**

11: Set $Q_{j+1} := 0$, $n_{j+1} := n_j$, and $p_{j+1} = 0$.

12: $k := j$

13: **return**

14: **else**

15: Compute

$$W_j P_j = \begin{bmatrix} \hat{Q}_1^{(j+1)} & \hat{Q}_2^{(j+1)} \end{bmatrix} \begin{bmatrix} R_{11}^{(j+1)} & R_{12}^{(j+1)} \\ 0 & 0 \end{bmatrix},$$

using a rank revealing QR algorithm with column pivoting.

16: Set $p_{j+1} := \operatorname{rank} W_j$, $Q_{j+1} := \hat{Q}_1^{(j+1)}$, and

$$H_{j+1,j} := \begin{bmatrix} R_{11}^{(j+1)} & R_{12}^{j+1} \end{bmatrix} P_j^{-1}.$$

17: **end if**

18: **end for**

Algorithm 11: A block Arnoldi algorithm with column pivoting

and

$$p_{j+1} = \text{rank}(W_j)$$

for $j = 1, 2, 3, \ldots, m$. Finally, we emphasize that the matrices H_j satisfy

$$H_j = V_j^T A V_j$$

both for $j < m$ and for $j = m$.

If A is a sparse matrix, then the most expensive part of the block Arnoldi algorithm is the modified Gram-Schmidt process. The cost of running k iterations is $O(np^2k^2)$ arithmetic operations and we need $O(npk + p^2k^2)$ words of storage to execute the algorithm and hold the final result.

6.3.2 The Arnoldi method

The Arnoldi method for Lyapunov equations is due to Saad [50] who considered the case of $p = 1$, and Jaimoukha and Kasenally [28] who extended the method to the general case of $p > 1$. The convergence of the Arnoldi method has been studied by Simoncini and Druskin [56].

We now derive the algorithm and make several comments along the way regarding the stability and limitations of the method.

Let V_j, and H_j be the matrices generated by applying the block Arnoldi algorithm to the matrices A and B, i.e.

$$AV_j = V_j H_j + Q_{j+1} H_{j+1,j} E_j^T$$

for $j < m$, and

$$AV_m = V_m H_m.$$

We consider approximations of the form

$$X_j = V_j Y_j V_j^T,$$

where Y_j is any square matrix of the appropriate size, i.e., an n_j by n_j matrix. By Theorem 2.6.3 we will recover the true solution for $j = m$. Now, the residual corresponding to X_j is given by

$$R(X_j) = AV_jY_jV_j^T + V_jY_jV_j^TA^T + BB^T.$$

The Arnoldi method computes Y_j such that the Galerkin condition

$$V_j^TR(X_j)V_j = 0,$$

is satisfied. Jaimoukha and Kasenally proved the following theorem.

Theorem 6.3.1 *Suppose that $k < m$ steps of Algorithm 11 have been taken, then an approximation of the form*

$$X_k = V_kY_kV_k^T$$

satisfies the Galerkin condition

$$V_k^TR(X_k)V_k = 0,$$

if and only if Y_k is a solution of

$$H_kY_k + Y_kH_k^T + V_k^TBB^TV_k = 0,$$

in which case the Frobenius norm of the residual $R(X_k)$ is given by

$$\|R(X_k)\|_F = \sqrt{2}\|H_{k+1,k}E_k^TY_k\|_F.$$

If $k = m$ steps of Algorithm 11 have been taken, then the corresponding residual is zero and the exact solution is obtained.

Proof The proof is elementary and can be found in Jaimoukha and Kasenally's paper [28]. ■

The fundamental problem is to ensure that each of the reduced order equations

$$H_kY_k + Y_kH_k^T + V_k^TBB^TV_k = 0,$$

has a unique solution Y_k. If H is a real matrix, then the Lyapunov equation

$$HY + YH^T + Q = 0$$

has a unique solution Y for every choice of Q if and only if

$$\lambda + \mu \neq 0$$

for every pair of eigenvalues λ and μ for H. In our case the original matrix A is always stable, which ensures that the original equation has a unique solution. However, as illustrated in the following example, this is not enough to ensure that the reduced order equations are uniquely solvable.

Let A be the n by n matrix given by

$$A = T - e_n e_n^T, \tag{6.2}$$

where T is the skew-symmetric Toeplitz matrix given by

$$T = \begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 0 & 1 \\ & & & -1 & 0 \end{bmatrix}$$

and e_n is the last column of the n by n identity matrix. We studied the matrix A in Chapter 2, when we discussed the low rank phenomenon for Lyapunov equations. The matrix A is stable, but if $B = e_n$, then the reduced order equations

$$H_k Y_k + Y_k H_k^T + V_k^T B B^T V_k = 0$$

are singular for all k, except for $k = m$. To see why this is the case, note that the A is already in upper Hessenberg form and since $B = e_n$, the matrices H_k are given by

$$H_k = A(1:k, 1:k)$$

for $k = 1, 2, \ldots, m$. Now, for $k < m$

$$A(1:k, 1:k) = T(1:k, 1:k)$$

is skew-symmetric, and it is clear the corresponding reduced order Lyapunov equations are all singular. However, if A is negative definite, i.e.

$$A + A^T < 0,$$

then $V^T AV$ is stable for every n by p matrix V with orthonormal columns and $p \leq n$. This is easily seen by considering the numerical range of A, see Chapter 7. The matrix A given by (6.2) is not negative definite.

We are now ready to state the Arnoldi method as Algorithm 12. Solving the first k dense Lyapunov equations using Bartels-Stewart's method requires $O(k^4p^3)$ arithmetic operations and evaluating the first k residuals requires $O(k^3p^3)$ arithmetic operations. When n is large, these figures are insignificant compared with the cost of running k steps of the block Arnoldi algorithm, which is $O(np^2k^2)$.

Input: The matrices V_j and H_j generated by applying the block Arnoldi algorithm, Algorithm 11, to an n by n negative definite matrix A and an n by p matrix B.

Output: An approximate solution of the form $X_j = V_j Y_j V_j^T$.

1: **for** $j = 1, 2, \ldots,$ until convergence **do**

2: Solve the reduced order equation

$$H_j Y_j + Y_j H_j + V_j^T BB^T V_j = 0.$$

3: Compute the Frobenius norm

$$\|R(X_j)\|_F = \sqrt{2}\|H_{k+1,k} E_k^T Y_k\|_F,$$

where E_k consists of the last p_k columns of the n_k by n_k identity matrix.

4: **end for**

Algorithm 12: The basic Arnoldi method for continuous time Lyapunov equations

The convergence of the Arnoldi method has been studied by Simoncini and Druskin [56]. They consider Lyapunov equations of the form

$$AX + XA^T = BB^T, \quad A = A^T > 0.$$

Theorem 6.3.2 (Simoncini, Druskin [56]) *Let A be a symmetric positive definite matrix, and let $\lambda_{\min}$ be the smallest eigenvalue of A. Let $\hat{\lambda}_{\min}$, $\hat{\lambda}_{\max}$ be the extreme eigenvalues of $A + \lambda_{\min} I$ and $\hat{\kappa} = \hat{\lambda}_{max}/\lambda$. Then*

$$\|X - X_k\|_2 \leq \frac{\sqrt{\hat{\kappa}} + 1}{\hat{\lambda}_{\min}\sqrt{\hat{\kappa}}} \left(\frac{\sqrt{\hat{\kappa}} - 1}{\sqrt{\hat{\kappa}} + 1} \right)^k. \tag{6.3}$$

They also considered the nonsymmetric case and proved a set of theorems of which the following is the simplest.

Theorem 6.3.3 (Simoncini, Druskin [56]) *Assume that the field of values of the real matrix A is contained in an ellipse of center $(c, 0)$, $c > 0$, foci $(c \pm d, 0)$ and semi-axes a_1 and a_2, with $a_1 \geq a_2$, such that $d = \sqrt{a_1^2 - a_2^2}$. Then*

$$\|X - X_k\| \leq \frac{8}{\sqrt{(\alpha_{\min} + c)^2 - d^2}} \frac{r_2}{r_2 - 1} \left(\frac{1}{r_2} \right)^m, \tag{6.4}$$

where $r_2 = \frac{c + \alpha_{\min}}{2r} + \frac{1}{2r}\sqrt{(c + \alpha_{\min})^2 - d^2}$, and $r = \frac{a_1 + a_2}{2}$.

We mention these theorems to make a simple point, namely that the convergence rate, or more accurately the bound on the residuals, depends continuously on the entries of the matrix A. This will become important when we discuss the notion of preconditioning.

We now show that the Arnoldi method can have an arbitrary residual history: Given a positive integer n and a sequence of positive real numbers $\{r_j\}_{j=1}^{n-1}$ we show how to construct an n by n negative definite matrix A and a column vector $B \in \mathbb{R}^n$, such that the Arnoldi method returns a sequence of residuals $R(X_j)$, for which

$$\|R(X_j)\|_F = r_j, \quad j = 1, 2, \ldots, n-1.$$

To the best of our knowledge this is a new result.

The following lemma constructs a family of negative definite matrices which are unaffected by the basic Arnoldi algorithm, Algorithm 6.

Lemma 6.3.1 *Let A be the n by n real matrix given by*

$$A = \begin{bmatrix} -\epsilon_1 & -\alpha_1 & & & \\ \alpha_1 & -\epsilon_2 & -\alpha_2 & & \\ & \alpha_2 & \ddots & \ddots & \\ & & \ddots & -\epsilon_{n-1} & -\alpha_{n-1} \\ & & & \alpha_{n-1} & -\epsilon_n \end{bmatrix}, \tag{6.5}$$

where $\epsilon_i > 0$ for $i = 1, 2, \ldots, n$ and $\alpha_i > 0$ for $i = 1, 2 \ldots, n-1$. Then A is negative definite. If B is the first column vector of the n by n identity matrix, then the grade of B with respect to A is n, and the basic Arnoldi algorithm, Algorithm 6, returns the trivial factorization

$$AI_n = I_n A$$

when applied to the pair (A, B).

Proof It is clear that A is negative definite, because the subdiagonal elements have been choosen so that they cancel the corresponding superdiagonal elements, i.e.

$$\frac{1}{2}\left(A + A^T\right) = \text{diag}\{-\epsilon_1, -\epsilon_2, \ldots, -\epsilon_n\} < 0.$$

It is not important here, but we observe that we have complete control over the spectrum of $\frac{1}{2}\left(A + A^T\right)$. Since A is upper Hessenberg, and each $\alpha_i \neq 0$ for $i = 1, 2, \ldots, n-1$, it follows that the n vectors

$$A^j B, \quad j = 0, 1, \ldots, n-1$$

form an independent set, hence the grade of B with respect to A is n. The Arnoldi algorithm applied to the pair (A, B) will return an orthonormal basis V of $K(A, B) = \mathbb{R}^n$ as well as an upper Hessenberg matrix H, such that

$$AV = VH.$$

However, we also have the trivial factorization $AI_n = I_n A$, and since A is an upper Hessenberg matrix with positive subdiagonal entries and I_n has orthonormal columns, it follows that $V = I_n$ and $A = H$. This last conclusion is a special case of Theorem 2.4 [60]. ■

Now, let $\epsilon_i > 0$ for $i = 1, 2, \ldots, n$ and $\alpha_i > 0$ for $i = 1, 2 \ldots, n-1$. Let A be given by (6.5) and let B be the first column vector of the n by n identity matrix. Consider the Lyapunov equation

$$AX + XA^T + BB^T = 0.$$

The Arnoldi method applies, because A is negative definite. The Frobenius norm of the residual $R(X_j)$ is given by

$$\|R(X_j)\|_F = \sqrt{2}\|H_{j+1,j}E_j^T Y_j\|_F, \quad j = 1, 2 \ldots, n-1,$$

where E_j is the last column of the j by j identity matrix and Y_j is the solution of the reduced order equation

$$H_j Y_j + Y_j H_j^T + B_j B_j^T = 0.$$

Now, by design the matrix A is not affected by the Arnoldi process, and it follows that the reduced order equations are

$$A(1:j, 1:j)Y_j + Y_j A(1:j, 1:j)^T + B(1:j)B(1:j)^T = 0, \quad j = 1, 2, \ldots, n \quad (6.6)$$

and the Frobenius norm of the residual is given by

$$\|R(X_j)\|_F = \sqrt{2}\|\alpha_j Y_j(1:j, j)\|_F = \sqrt{2}\alpha_j\|Y_j(1:j, j)\|_F, \quad j = 1, 2, \ldots, n-1.$$

This last expression is the key. The real number α_j is the value of the $(j+1, j)$ entry of A, hence it is not involved in the calculation of Y_j. If the jth row of the Y_j is nonzero, then α_j can be chosen such that $\|R(X_j)\|_F$ assumes whatever value we desire.

Now, since the matrix Y_j is symmetric, it suffices to show that the jth column of the Y_j is nonzero. It is clear that $Y_j \neq 0$, because the inhomogenous terms $B_j B_j^T$ is nonzero. However, only the $(1, 1)$ entry of this matrix is nonzero, and we can use this structure to show that if $Y_j(:, j) = 0$, then $Y_j = 0$, which is a contradiction. Let $j > 1$, then by equating the jth columns of each side of equation (6.6) we see that

$$A(1:j, 1:j)Y_j(:, j) - Y_j(:, j)\epsilon_j + Y_j(:, j-1)\alpha_{j-1} = 0,$$

because A has only a single subdiagonal. If the jth column of Y_j should vanish, then this expression reduces to

$$Y_j(:, j-1)\alpha_{j-1} = 0$$

and since $\alpha_{j-1} > 0$ we see $Y_j(:, j-1) = 0$. Now, suppose that $1 < i < j$ and we have established that $Y_j(:, i) = Y_j(:, i+1) = \cdots = Y_j(:, j) = 0$, then we claim that $Y_j(:, i-1) = 0$. By equating the ith columns of both sides of equation (6.6) we see that

$$A(1:j, 1:j)Y_j(:, i) - Y_j(:, i+1)\alpha_i - Y_j(:, i)\epsilon_i + Y_j(:, i-1)\alpha_{i-1} = 0.$$

By assumption, each of the columns $Y_j(:, i+1)$ and $Y_j(:, i)$ are zero, and we are left with

$$Y_j(:, i-1)\alpha_{i-1} = 0$$

from which it follows $Y_j(:, i-1) = 0$. We conclude that if the last column of Y_j is zero, then Y_j is zero. This is a clear contradiction and it follows that the last column of Y_j is always nonzero.

Algorithm 13 constructs a negative definite Lyapunov equation for which the Arnoldi method has a given residual history. Using this algorithm we have constructed a matrix of dimension $n = 200$ for which the residual history is particularly nice, see Figure 6.1.

The Arnoldi method suffers from a fundamental problem which is not fully appreciated in the literature: It does not exploit the low rank phenomenon. We now construct a negative definite Lyapunov equation such that the solution admits a good low rank approximation and the Arnoldi method does not converge until the very last iteration. Let A be the n by n bidiagonal Toeplitz matrix given by

$$A = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}$$

Input: Integer $n > 0$, $\epsilon_j > 0$ for $i = j, 2, \ldots, n$, $r_j > 0$ for $j = 1, 2, \ldots, n-1$.

Output: $A \in \mathbb{R}^{n \times n}$ negative definite, $B \in \mathbb{R}^n$, such that the residuals returned by the Arnoldi method, Algorithm 12, satisfy

$$\|R(X_j)\|_F = r_j, \quad j = 1, 2, \ldots, n-1.$$

1: Set $A(i,i) := -\epsilon_i$ for $i = 1, 2, \ldots, n$ and $B := I_n(:, 1)$.

2: **for** j=1:n-1 **do**

3: Solve $A(1:j)Y_j + Y_j A(1:j, 1:j)^T + B(1:j)B(1:j)^T = 0$.

4: Set $\alpha_j := \frac{r_j}{\sqrt{2}\|Y_j(:,j)\|_F}$.

5: Set $A(j+1, j) := \alpha_j$, $A(j, j+1) := -\alpha_j$.

6: **end for**

Algorithm 13: Construction of a special Lyapunov equation

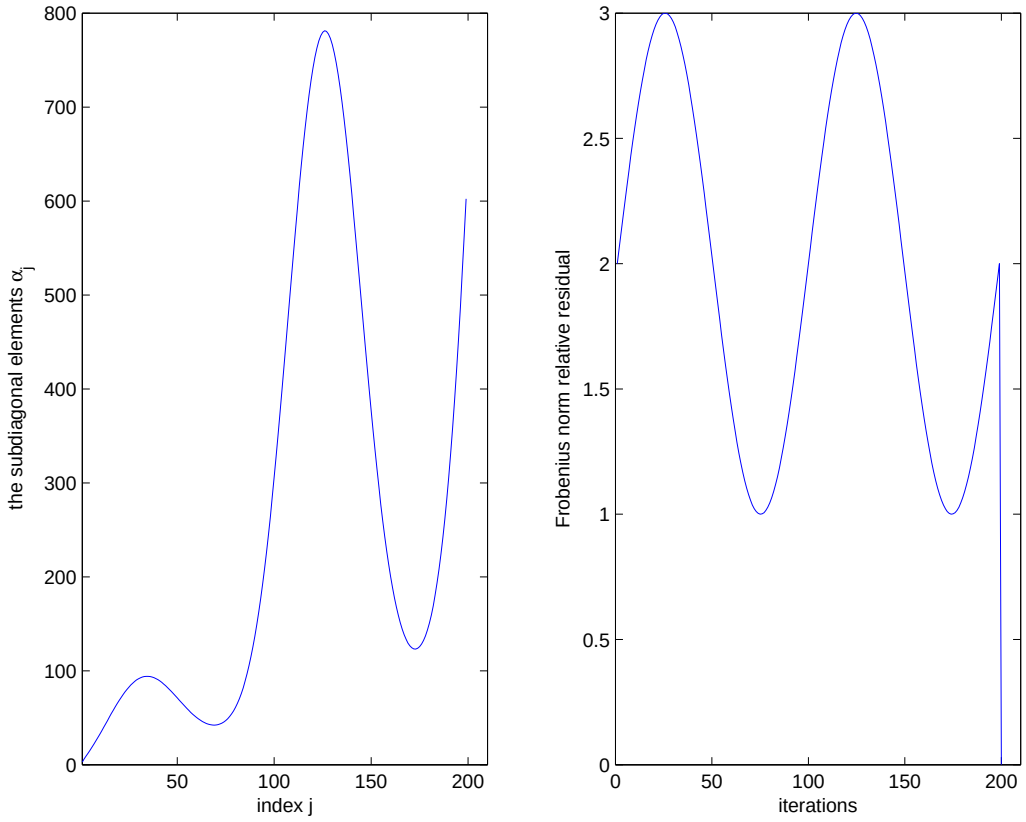

Figure 6.1. With $\epsilon_j = -1$ and a very particular choice of α_j we can force the Arnoldi method to have the given residual history.

and

$$B = (1, 1, \ldots, 1)^T.$$

We choose $n = 500$ and solve the corresponding Lyapunov equation

$$AX + XA^T + BB^T = 0 \tag{6.7}$$

using the MATLAB function "lyap". The singular values of the solution are approximated in Figure 6.2. It is clear that the exact solution can be accurately approximated by matrices of very low rank. We applied the Arnoldi method to the Lyapunov equation. The residual history can also be found in Figure 6.2. We would like to emphasize that the matrix A is negative definite, and

$$A + A^T = \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}.$$

The previous example realized the worst possible kind of behavior, but it does not correspond to any known "real-life" application that we are aware of. Consider the following heat equation

$$v_t = v_{xx} + g(x)u(t), \quad x \in (0, 1),\ t > 0,$$

together with the initial condition

$$v(x, 0) = f(x),$$

and the homogeneous boundary conditions

$$v(0, t) = v(1, t) = 0.$$

Now, choose an integer $N > 1$, set $h = 1/N$, and let

$$x_j = jh, \quad j = 0, 1, \ldots, N.$$

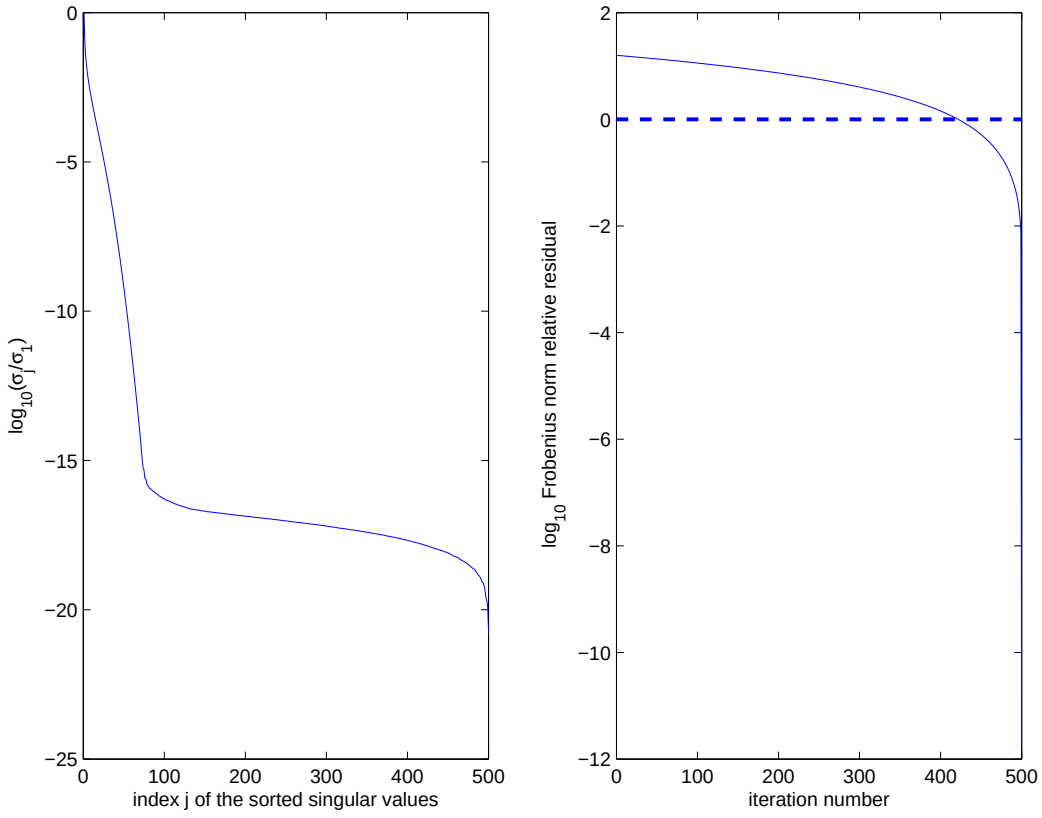

Figure 6.2. Singular value decay and residual history for the Lyapunov equation (6.7).

Let $w_j(t)$ be the approximation to $u(x_j, t)$, given by

$$\frac{dw_j}{dt} = \frac{w_{j+1} - 2w_j + w_{j-1}}{h^2} + g(x_j)u, \quad j = 1, 2, \ldots N-1,$$

and

$$w_0(t) = w_N(t) = 0$$

for all $t > 0$. This can be rewritten in matrix notation as

$$\dot{w} = Aw + Bu(t),$$

where A is the $N-1$ by $N-1$ matrix given by

$$A = h^{-2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}$$

and B is the vector given by

$$B = (g(x_1), g(x_2), \ldots, g(x_{N-1}))^T.$$

We now choose $g = 1$, which corresponds to

$$B = (1, 1, \ldots, 1)^T,$$

and consider the Lyapunov equation

$$AX + XA^T + BB^T = 0. \tag{6.8}$$

It is clear that the matrix A is stable, and even negative definite, and the Arnoldi method applies. We consider the case of $N = 501$. It is not difficult to see that

$$\dim K(A, B) = 250,$$

which implies that the true solution X of the Lyapunov equation has rank 250. The decay of the singular values for X is illustrated in Figure 6.3. We see that it is possible to find a good low rank approximation of X. The residual history for the Arnoldi method applied to this problem is given in Figure 6.3. Again, we see that the Arnoldi method does not converge until iteration 250. There is nothing special about $N = 501$ and we have seen the same type of behavior for different values of N.

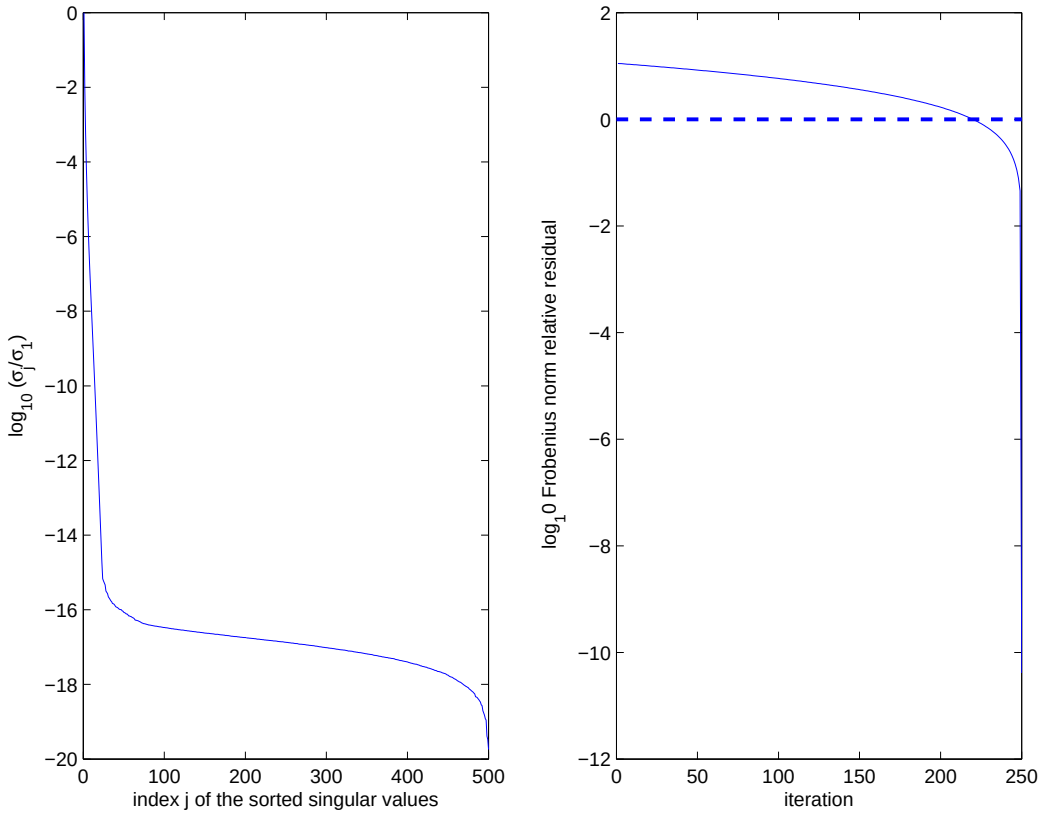

Figure 6.3. Singular value decay and residual history for the Lyapunov equation (6.8).

6.3.3 The GMRES method

The GMRES method, due to Jaimoukha and Kasenally [28], computes an approximation of the form

$$X_j = V_j Y_j V_j^T,$$

where the matrices V_j are computed by applying a block Arnoldi algorithm to the matrices A and B. The matrix Y_j is chosen such that the Frobenius norm of the residual

$$R(X_j) = AX_j + X_j A^T + BB^T$$

is minimized over all symmetric matrices Y_j, i.e.

$$\|R(X_j)\|_F = \min\{\|R(X)\|_F \ : \ X = V_j Y_j V_j^T, \, Y_j = Y_j^T\}.$$

The problem of computing the minimizer Y_j is a linear least squares problem with n_j^2 unknowns, where n_j denotes the number of columns in V_j:

$$\|R(X_j)\|_F = \|AV_j Y_j V_j^T + V_j Y_j V_j^T A^T + BB^T\|_2 = \|A(V_j)\mathrm{vec}(Y_j) + \tilde{b}\|_2,$$

where

$$A(V_j) = V_j \otimes AV_j + AV_j \otimes V_j,$$

is an n^2 by n_j^2 matrix, and $\tilde{b} = \mathrm{vec}(BB^T)$. When does this problem have a unique solution? If A is negative definite, then $A(V_j)$ has full rank, because

$$A(V_j)\mathrm{vec}(Y_j) = 0 \Rightarrow AV_j Y_j V_j^T + V_j Y_j V_j A^T = 0 \Rightarrow V_j^T AV_j Y_j + Y_j V_j^T A^T V_j = 0,$$

which in turn implies $Y_j = 0$, because $V_j^T AV_j$ is stable. Now, in general $n_j \leq jp$ and $n_j = jp$ is possible. If A is negative definite, then it is possible to compute Y_j using $O(j^6 p^6)$ arithmetic operations, by solving the normal equations

$$A(V_j)^T A(V_j)\mathrm{vec}(Y_j) + A(V_j)^T \tilde{b} = 0.$$

Jaimoukha and Kasenally [28] derived an algorithm, which exploits the special structure of the problem in order to create a sequence of smaller problems which can be solved independently. Their algorithm also requires $O(j^6 p^6)$ arithmetic operations. However, their approach has the following limitations:

- Their algorithm cannot handle column pivoting in the Arnoldi algorithm, a fact which is not stated explicitly in their original paper. However, in the proof of their central Theorem 3.3 [28], it is critical that the subdiagonal blocks of the upper block Hessenberg matrix H_j produced by the block Arnoldi algorithm are upper triangular. If the block Arnoldi algorithm uses column pivoting to handle rank degradation, then this structure is lost. We have confirmed this problem by communicating directly with Imad Jaimoukha.

- Each approximation is constructed as a linear combination of the solutions to certain specialized Lyapunov equations. The exact linear combination to use is given as the solution to a very special linear system. Unfortunately the condition number for this linear system has a tendency to grow exponentially with the iteration number to the point where it completely overwhelms the unit roundoff error of the machine, making it impossible to determine which linear combination to use.

In our numerical experiments we have tracked the condition number of these linear systems and observed algorithmic breakdown as these linear systems became singular to working precision.

In exact arithmetic, the GMRES method recovers the exact solution, when the Krylov subspace $K(A, B)$ has been constructed. To the best of our knowlegde there are no results on the convergence rate of the GMRES method, even in exact arithmetic. In their experiments Jaimoukha and Kasenally [28] observed that the iteration count for the Arnoldi method is slightly larger than the iteration count for the GMRES method. We would add that this is true only as long as their GMRES method has not broken down.

6.3.4 Other Krylov methods and variations

Jaimoukha and Kasenally [28] extended Saad's Arnoldi method [50] to the case of $p > 1$. We are aware of two distinct extensions, namely the work of Hu and

Reichel [27], as well as the work of Jbilou and Riquet. The method of Hu and Reichel is a Galerkin type method which requires the selection of certain parameters. The method of Jbilou and Riquet relies on their global Arnoldi method. It requires the solution of certain reduced order equations, which may or may not have unique solutions. Both methods have Saad's method as a special case.

Simoncini [55] has developed a variant of the Arnoldi method which requires the solution of linear systems with coefficient matrix A. There has not been a thorough comparison made betweeen this method and Sabino's version of the block cyclic Smith method [52] with his improved heuristics for the selection of shift parameters.

Attempts have been made to mitigate the effect of the slow convergence of the Arnoldi method by conserving memory. Kressner [30] uses a two-pass version of the Lanczos algorithm to reduce memory consumption to a very small number of vectors. Naturally, this procedure applies only to symmetric matrices A, and memory is saved at the cost of a two-fold increase in arithmetic.

6.3.5 ADI-methods

This is a class of algorithms which have been very successful for the solution of large sparse Lyapunov equations of the form

$$AX + XA^T + BB^T = 0,$$

where A is a stable matrix and B is a tall matrix with a relatively small number of columns.

Smith [58] recognized that the continuous time Lyapunov matrix equation

$$AX + XA^T + Q = 0,$$

is equivalent with the discrete time Lyapunov equation

$$X = UXU^T + W, \tag{6.9}$$

where

$$U = (A + pI)^{-1}(A - pI) \quad \text{and} \quad W = -2p(A + pI)^{-1}Q(A + pI)^{-T},$$

and p is any complex number with negative real part. This is not difficult to show. The key observation is the elementary identity

$$(A+pI)X(A+pI)^T = (A-pI)X(A-pI)^T + 2p(AX+XA^T),$$

which is valid for any complex number p. If p has negative real part, then $A+pI$ is nonsingular, because A is stable, and we may apply $(A+pI)^{-T}$ from the left and $(A+pI)^{-T}$ from the right.

Now, the solution of the discrete time Lyapunov equation (6.9) is given by

$$X = \sum_{j=0}^{\infty} U^j W (U^T)^j,$$

and if

$$\begin{aligned} X_0 &= 0, \\ X_{n+1} &= X_n + U X_n U^T, \quad n = 0, 1, 2, \ldots, \end{aligned} \tag{6.10}$$

then it is not difficult to see that $X_n \to X$, and the rate of convergence is linear. This is the standard Smith method. Smith [58] introduced the sequence

$$\begin{aligned} Y_0 &= 0, \\ Y_{n+1} &= Y_n + U^{2^n} W (U^T)^{2^n}, \quad n = 0, 1, 2, \ldots. \end{aligned}$$

An easy induction establishes

$$Y_n = \sum_{j=0}^{2^n-1} U^j W (U^T)^j,$$

from which it follows that $Y_n \to X$ for $n \to \infty$. Observing that,

$$U^{2^{n+1}} = \left(U^{2^n}\right)^2,$$

the iteration is given in Algorithm 14. This is the so-called squared Smith method which converges quadratically.

The fundamental problem is that neither the Smith method nor the squared Smith method exploits the sparsity of the matrix A. Even when A is very sparse the matrices

1: $Y := 0$

2: **for** $n = 0, 1, 2, \ldots$ **do**

3: $\quad Y := Y + UWU^T$,

4: $\quad U := U^2$,

5: **end for**

Algorithm 14: Squared Smith method

U will almost certainly be dense. This limits the Smith iteration and the squared Smith iteration to small dense problems.

Wachspress [66] recognized that the standard Smith iteration is equivalent to the basic ADI iteration

$$\begin{aligned} X_0 &= 0, \\ (A+pI)X_{i+1/2} &= -Q - X_i(A-pI)^T, \\ X_{i+1}(A+pI)^T &= -Q - (A-pI)X_{i+1/2}, \quad i = 0, 1, 2, \ldots, \end{aligned}$$

and introduced a more general iteration of the form

$$\begin{aligned} X_0 &= 0, \\ (A+p_iI)X_{i+1/2} &= -Q - X_i(A-p_iI)^T \\ X_{i+1}(A+p_iI)^T &= -Q - (A-p_iI)X_{i+1/2}, \quad i = 0, 1, 2, \ldots. \end{aligned}$$

This iteration is mathematically equivalent to

$$\begin{aligned} X_{i+1} = -2p_i(A+p_iI)^{-1}Q(A+p_iI)^{-T} \\ + (A+p_iI)^{-1}(A-p_iI)X_i(A-p_iI)^T(A+p_iI)^{-T}. \end{aligned} \quad (6.11)$$

The solution of the Lyapunov equation satisfies

$$\begin{aligned} X = -2p_i(A+p_iI)^{-1}Q(A+p_iI)^{-T} \\ + (A+p_iI)^{-1}(A-p_iI)X(A-p_iI)^T(A+p_iI)^{-T}, \end{aligned}$$

which implies that

$$(X - X_{i+1}) = (A+p_iI)^{-1}(A-p_iI)(X-X_i)(A-p_iI)^T(A+p_iI)^{-T},$$

or equivalently

$$(X - X_{i+1}) = (p_iI+A)^{-1}(p_iI-A)(X-X_i)(p_iI-A)^T(p_iI+A)^{-T}.$$

Now let r_l be the polynomial given by

$$r_l(t) = \prod_{i=1}^{l}(p_i - t),$$

then by induction we establish that

$$(X - X_{l+1}) = r_l(-A)^{-1} r_l(A)(X - X_0) r_l(A)^T r_l(-A)^{-T}.$$

It is clear that the rate of convergence is controlled by the spectral radius of the operator

$$D_l(A) = r_l(-A)^{-1} r_l(A).$$

The problem of minimizing the spectral radius as a function of the shift parameters is given by

$$\{p_1, p_2, \ldots, p_l\} = \underset{\{p_1,p_2,\ldots,p_l\}}{\arg\min} \max_{t \in \sigma(A)} \frac{|r_l(t)|}{|r_l(-t)|}.$$

This is the minimax problem for Lyapunov equations. It is a nontrivial problem to select the optimal shift parameters. However, the problem has been solved for matrices A with real spectrum. There have been many contributions to the complex case and a set of references can be found in Penzl's paper [40], as well as in Sabino's dissertation [52].

Penzl [40, 41] showed that it is possible to solve large sparse Lyapunov equations using $O(n)$ time and storage. He considered the ADI iteration in the special case in which the inhomogeneous term Q is a symmetric positive definite matrix with low rank, i.e. the case of

$$Q = BB^T,$$

where B is a tall matrix with a relatively small number of columns. Penzl recognized that the ADI matrices X_i given by equation (6.11) could be written as

$$X_i = Z_i Z_i^T, \quad i = 0, 1, 2 \ldots,$$

where

$$\begin{aligned} Z_0 &= 0, \quad \text{and} \\ Z_{i+1} &= \left[\sqrt{-2p_i}(A + p_i)^{-1} B \quad (A + p_i I)^{-1}(A - p_i I) Z_i \right], \quad i = 0, 1, 2, \ldots. \end{aligned} \tag{6.12}$$

This is the low rank ADI method for Lyapunov equations. Notice that each iteration consists of a linear solve with an increasing number of right hand sides.

Penzl showed experimentally that it was possible to obtain good results by using a limited number of shift parameters in a cyclic fashion, i.e.

$$p_{i+jl} = p_i,$$

for $i = 1, 2, \ldots, l$, and $j = 0, 1, 2 \ldots$. He also showed that it was possible is avoid complex arithmetic by using conjugate pairs of shift parameters.

Penzl introduced the low rank cyclic Smith method, which is equivalent to the above low rank ADI iteration, if the shift parameters are used in a cyclic fashion in the ADI iteration. The advantage of the low rank cyclic Smith method is that each iteration consists of a linear solve with a fixed, rather than an increasing number of right hand sides. Penzl developed a simple heuristic for selecting shift parameters based on the Arnoldi algorithm.

Gugercin, Sorensen, and Antoulas [18] developed and analyzed a modified version of the low rank ADI method and the low rank cyclic Smith method. They reduced the storage requirements by computing a truncated singular value decomposition of the iterates.

Sabino [52] developed a block version of the cyclic Smith method and made significant progress on the art of selecting good shift parameters. However, the problem of selecting optimal shift parameters for a general stable matrix remains open. Even when good shift parameters can be found it is a nontrivial exercise to solve the corresponding linear systems.

7. The Approximate Power Iteration

7.1 Introduction

In this chapter we review the approximate power iteration which was introduced by A. S. Hodel [24] in 1989. The algorithm computes an approximation of the rank k dominant invariant subspace U_k for the solution P to the Lyapunov equation

$$AP + PA^T + BB^T = 0, \tag{7.1}$$

by approximating the action of P, rather than computing P explicitly. We derive the algorithm, and present a few numerical experiments. We have used Hodel's main idea to develop a new algorithm which is faster and more accurate. The improved algorithm is presented in Chapter 8. Hodel's analysis of the API is extended in Theorem 7.5.2.

The problem of computing the dominant eigenspace of the implicitly given operator P is interesting in its own right, but it is also the first step toward computing the dominant eigenspace of a product PQ, where Q is given as the solution of the adjoint equation

$$A^TQ + QA^T + C^TC = 0$$

The importance of this problem is explained in detail in Chapter 3. In the special case of $A = A^T$ and $B = C^T$, the two equations are identical, $P = Q$, and the general problem reduces to the special case which we consider here and in Chapter 8.

7.2 The algorithm

If P were available explicitly, then we could have used subspace iteration to estimate U_k, see Chapter 4. Since P is not available explicitly, we must use equation

(7.1) to approximate the action of P. Let V be any n by k matrix with orthonormal columns. Our immediate goal is to approximate PV. By post-multiplying equation (7.1) by V it follows,

$$A(PV) + PA^TV + BB^TV = 0,$$

which we rewrite as

$$A(PV) + (PV)(V^TA^TV) + P(I - VV^T)A^TV + BB^TV = 0.$$

Now, let us for now disregard the term

$$P(I - VV^T)A^TV, \tag{7.2}$$

which may or may not be large, and consider instead the Sylvester equation

$$AY + Y(V^TA^TV) + BB^TV = 0. \tag{7.3}$$

This equation has a unique solution $Y = Y(V)$, since A being negative definite implies that both A and V^TA^TV are stable. This is not difficult to see. The numerical range $\mathrm{nr}(A)$ of an n by n matrix A is given by

$$\mathrm{nr}(A) = \{x^*Ax \,:\, x \in \mathbb{C}^n,\, x^*x = 1\} \subset \mathbb{C}.$$

For any square matrix A we have

$$\sigma(A) \subset \mathrm{nr}(A),$$

and if V is any matrix with orthonormal columns, then

$$\mathrm{nr}(V^TAV) \subseteq \mathrm{nr}(A).$$

If A is negative definite, then $\mathrm{nr}(A) \subset \mathbb{C}_-$. Additional properties of the numerical range of a matrix can be found in Shapiro [54], as well as Psarrakos and Tsatsomeros [45].

Now, if the special term (7.2) were to vanish, then we could recover PV as the unique solution to equation (7.3). The special term (7.2) need not be small, but it is possible to overcome this problem. Hodel proved the following theorem for the case of $p = 1$. The extension to $p > 1$ is trivial.

Theorem 7.2.1 *Let A be a real n by n negative definite matrix, and let B be a real n by p matrix and let P be the solution to the Lyapunov equation*

$$AP + PA^T + BB^T = 0.$$

Let V be any n by k matrix with orthonormal columns. Let the columns of W be the orthonormal basis for $K(A^T, V)$ constructed by using the block Arnoldi algorithm. Then

$$PV = Y \begin{bmatrix} I_k \\ 0 \end{bmatrix},$$

where Y is the unique solution to the Sylvester equation

$$AY + Y(W^T A^T W) + BB^T W = 0.$$

Proof Let V be any n by k matrix with orthonormal columns and let the columns of W be the orthonormal basis for $K(A^T, V)$ constructed by using the block Arnoldi algorithm. By post-multiplying equation (7.1) with W it follows

$$A(PW) + P(A^T W) + BB^T W = 0.$$

Since $\operatorname{Ran} W$ is A^T invariant, we have $A^T W = WW^T A^T W$, and

$$A(PW) + (PW)W^T A^T W + BB^T W = 0.$$

This equation has a unique solution $Y = PW$, because A being negative definite implies both A and $W^T A^T W$ are stable. By the block Arnoldi algorithm,

$$W = \begin{bmatrix} V, & Q \end{bmatrix},$$

where Q is a suitable matrix. It follows immediately that

$$PV = Y \begin{bmatrix} I_k \\ 0 \end{bmatrix}.$$

■

Now, the dimension of $K(A^T, V)$ need not be small and the worst case in which

$$K(A^T, V) = \mathbb{R}^n,$$

can be unavoidable. As a result, Hodel suggested to replace W with W_l, where the columns of W_l form an orthonormal basis for $K_l(A^T, V)$ and are computed using the block Arnoldi algorithm. He derived Algorithm 15 which is the original Approximate Power Iteration (API).

Input: An n by k matrix V_0 with orthonormal columns, a positive integer l.

Output: An approximation of the rank k dominant invariant eigenspace for P.

1: **for** $i = 0, 1, 2 \ldots$ **do**

2: Compute an orthonormal basis W_i of the Krylov subspace $K_l(A, V_i)$ using the block Arnoldi algorithm.

3: Solve $AY_i + Y_i W_i^T A^T W_i + BB^T W_i = 0$ for Y_i.

4: Compute V_{i+1}, an orthonormal basis for $Y_i \begin{bmatrix} I_k \\ 0 \end{bmatrix}$.

5: **end for**

Algorithm 15: The original API

Hodel pointed out that the size of the neglected term,

$$P(I - W_l W_l^T)A^T W_l,$$

in which W_l spans $K_l(A^T, V_i)$, is not necessarily a decreasing function of l. He gave an explicit example where the size of the term is a strictly increasing function of l until it finally vanishes, when

$$K_l(A^T, V_i) = K(A^T, V_i),$$

and an A^T invariant vector space has been constructed.

Hodel suggested to terminate the iteration when the distance between successive subspaces, i.e.

$$\|V_i V_i^T - V_{i-1} V_{i-1}^T\|_2 = \|V_i - V_{i-1} V_{i-1}^T V_i\|_2, \tag{7.4}$$

becomes sufficiently small. Hodel used the second stopping criterion, because the 2-norm can then be computed by examining a k by k matrix rather than a full n by n matrix. The equality is a consequence of Theorem 2.6.1 [15].

7.3 Numerical experiments

We illustrate the behavior of Algorithm 15 by applying it to the Lyapunov equation

$$AX + XA^T + BB^T = 0,$$

where A is the n by n Toeplitz matrix given by

$$A = \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix},$$

and

$$B = (1, 1, \ldots, 1)^T \in \mathbb{R}^n.$$

This is merely a special case of the problem treated in Example 1, Chapter 3. We choose $n = 500$ and compute approximations to the rank $k = 5$ dominant eigenspace for X. First we solve the Lyapunov equation using the MATLAB funtion "lyap" and extract an approximation U of the rank $k = 5$ dominant subspace using the MATLAB function "eig". This allows us to compute the distance between the current API iterate, V_i, and the target, U, i.e. the subspace identification error given by

$$\|UU^T - V_i V_i^T\|_2 = \|U - V_i V_i^T U\|_2, \quad i = 0, 1, 2, \ldots.$$

The results are given in Figure 7.1. We see that for this example a higher value of l ultimately leads to a smaller subspace identification error. The individual iterations become more expensive as l steps of the (block) Arnoldi algorithm requires $O(nl^2)$ arithmetic operations. In our example it is faster to use $l = 4$ to achieve an error of 10^{-4}. But if the largest acceptable error is 10^{-6}, then we must use $l = 8$, as the curve

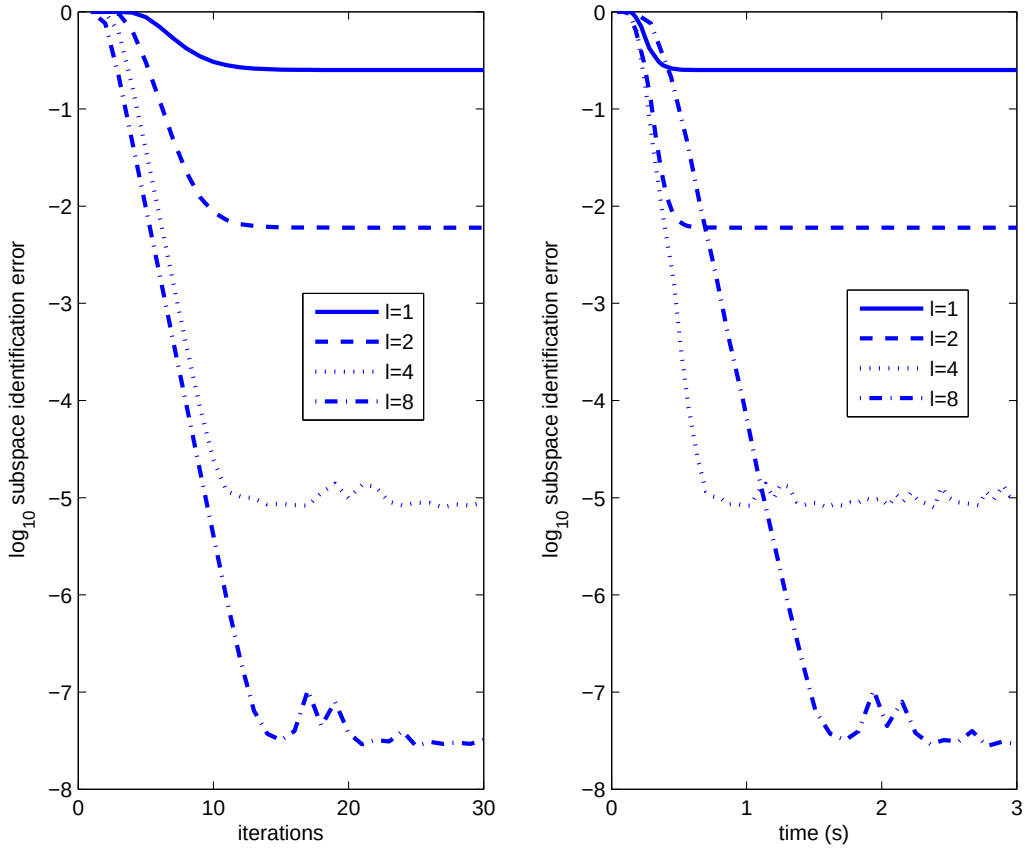

Figure 7.1. API: The subspace identification error for different values of the number of Arnoldi iterations.

corresponding to $l = 4$ levels off at 10^{-5}. We solve every relevant Sylvester equation using the variant of Bartels-Stewart's method described in Chapter 6.

7.4 Modifications of the original algorithm

Hodel, Tenison and Poolla [25] suggested that the block Arnoldi process in Algorithm 15 be run against $(A^T, \begin{bmatrix} V_i, & B \end{bmatrix})$, rather than the pair (A^T, V_i), which is what Hodel originally did. We have been unable to duplicate their exact experiments. The fundamental problem is that we do not know exactly how they initialized their

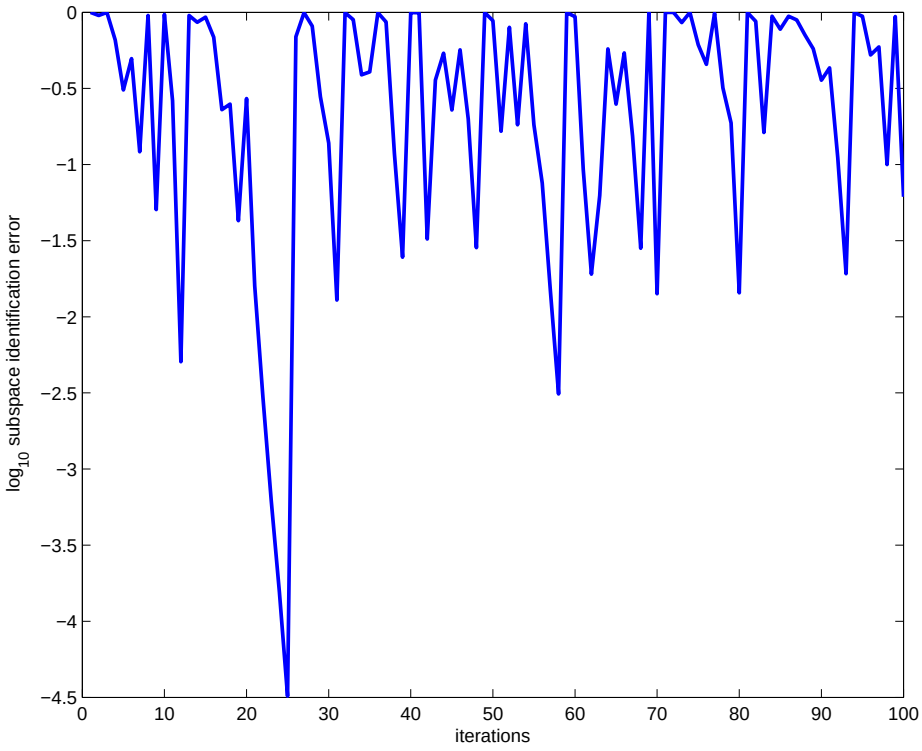

Figure 7.2. The effect of including B in the Arnoldi processes of Algorithm 15.

iterations. We have always used randomly generated starting points, and in our experience the inclusion of B does not improve the performance, but causes the subspace identification error to oscillate wildly. As an illustration we return to the example used in Section 7.3. Figure 7.2 shows the effect of appending B for the case of $l = 4$. We have not found a single case where there was any advantage to including B in the Arnoldi process.

7.5 Theoretical results

In his dissertation, Hodel [24] focused on extracting the dominant eigenpair (λ_1, v_1) $(k = 1)$ for the solution P. The analysis presented there dealt exclusively with the special case of $l = 1$. Let $v \in \mathbb{R}^n$ be any unit vector and let $y \in \mathbb{R}^n$ be the solution of the Sylvester equation

$$Ay + y\theta(v) + Qv = 0,$$

where

$$\theta(v) = v^T A^T v,$$

then

$$y = -(A + \theta(v)I)^{-1} v.$$

It is clear that y is not zero, because v is not zero and A is negative definite. We see that in the case of $k = 1$, $l = 1$, the API reduces to the simple iteration

$$v_{i+1} = \phi(v_i),$$

where v_0 is any unit vector and ϕ is given by

$$\phi(v) = \frac{-(A + \theta(v)I)^{-1} Qv}{\|(A + \theta(v)I)^{-1} Qv\|_2}.$$

This is the so-called scalar API algorithm. It is clear that $\phi : S \to S$, where

$$S = \{x \in \mathbb{R}^n \ : \ \|x\|_2 = 1\},$$

is a continuous function. Hodel [24] made a preliminary analysis of the existence, uniqueness and stability of fixed points for this map.

We are primarily interested in the following two results.

Proposition 7.5.1 (Hodel [24]) *Let A be a negative definite matrix and let X be the solution of*

$$AX + XA^T + BB^T = 0.$$

Let u_1 be the dominant eigenvector of X and let $\lambda_1 > \lambda_2$ be the dominant eigenvalues of X. Let $v \in R^n$ be a unit vector and let

$$\hat{y} = -(A + \theta(v)I)^{-1}v,$$

where

$$\theta(v) = v^T A^T v.$$

If $y = Xv$ then

$$\|y - \hat{y}\|_2 \leq \kappa_2(A)(\lambda_1 \|u_1 - v\|_2^2 + \lambda_2),$$

where $\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2$ is the condition number of A with respect to the 2-norm.

Theorem 7.5.1 (Hodel [24]) *Let A be negative definite, and let X be the solution of*

$$AX + XA^T + BB^T = 0.$$

Let u_1 be the dominant eigenvector of X and let $\lambda_1 > \lambda_2$ be the dominant eigenvalues of X. Let $\{v_i\}_{i=0}^{\infty}$ be given by

$$v_{i+1} = \phi(v_i), \quad i = 0, 1, 2, \ldots,$$

where v_0 is any unit vector. Suppose

$$\frac{\lambda_2}{\lambda_1} < \frac{1}{16(\kappa + 1)^2},$$

where $\kappa = \kappa_2(A) = \|A\|_2 \|A^{-1}\|_2$. Define

$$R = \frac{1}{\kappa + 1}\left(1 + \sqrt{1 - 16(\kappa + 1)^2 \frac{\lambda_2}{\lambda_1}}\right),$$
$$r = \frac{1}{\kappa + 1}\left(1 - \sqrt{1 - 16(\kappa + 1)^2 \frac{\lambda_2}{\lambda_1}}\right).$$

Then

1. *If $\|v_i - u_1\| \leq R$, then $\|v_{i+1} - u_1\| \leq R$.*

2. *If $\|v_0 - u_1\| \leq R$, then $\limsup \|v_i - u_1\|_2 \leq r$.*

3. *There exist a fixed point v_∞ for the scalar API such that $\|v_\infty - u_1\| \leq r$. The fixed point is not necessarily stable.*

Proposition 7.5.1 states that if v is not too far from the dominant eigenvector and if $\lambda_1 \gg \lambda_2$, then $\hat{y}$ is a fairly accurate approximation of $y = Xv$.

Theorem 7.5.1 states that if A is a well conditioned matrix and if $\lambda_1 \gg \lambda_2$, then there is a fixed point v_∞ which is fairly close to a dominant eigenvector u_1. If the initial separation is not too great, then the iterates will eventually lie in a small neighborhood surrounding u_1.

Hodel pointed out the special case of $\lambda_2 = 0$, in which $r = 0$, and the scalar API algorithm has a stable fixed point at the dominant eigenvector u_1 of X. We now strengthen Hodel's result by showing that if X has rank 1, then the API algorithm converges in a single iteration to a dominant eigenvector regardless of the initial guess.

Theorem 7.5.2 *Let A be a negative definite matrix, and let X be the solution of the Lyapunov equation*

$$AX + XA^T + BB^T = 0.$$

Let v be a unit vector. If X has rank 1, then $\phi(v)$ is a dominant eigenvector of X for every unit vector v.

Proof By assumption $y = Xv$, satisfies the Sylvester equation

$$Ay + y(v^T A^T v) + X(I - vv^T)A^T v + BB^T v = 0,$$

while $\hat{y} = \phi(v)$ is given as the solution of

$$A\hat{y} + \hat{y}(v^T A^T v) + BB^T v = 0.$$

It follows that $y - \hat{y}$ satisfies

$$A(y - \hat{y}) + (y - \hat{y})v^T A^T v + X(I - vv^T)A^T v = 0,$$

from which we deduce

$$(y - \hat{y}) = -(A + (v^T A^T v)I)^{-1} X(I - vv^T)A^T v.$$

By assumption, the symmetric positive semidefinite matrix X has rank 1, i.e.

$$X = \lambda_1 u_1 u_1^T,$$

where u_1 is a unit vector and $\lambda_1 > 0$. By Theorem 2.6.4

$$\operatorname{Ran} X = K(A, B),$$

from which we deduce that u_1 is an eigenvector for A, i.e.

$$Au_1 = \mu u_1.$$

Therefore

$$\begin{aligned}(y - \hat{y}) &= -(A + (v^T A^T v)I)^{-1} X (I - vv^T) A^T v \\ &= -\frac{1}{(\mu + v^T A^T v)} \lambda_1 u_1 u_1^T (I - vv^T) A^T v = -\frac{\lambda_1 u_1^T (I - vv^T) A^T v}{\mu + v^T A^T v} u_1.\end{aligned}$$

On the other hand, since $y = Xv = \lambda_1 (u_1^T v) u_1$, we deduce that

$$\hat{y} = y - (y - \hat{y}) = \left(\lambda_1 (u_1^T v) + \frac{\lambda_1 u_1^T (I - vv^T) A^T v}{\mu + v^T A^T v} \right) u_1.$$

In short, $\hat{y}$ is proportional to u_1. Since $\hat{y}$ is nonzero, it follows that $\phi(v) = \hat{y}/\|\hat{y}\|_2$ is a dominant eigenvector of X. ■

We would like to end this chapter by making an elementary point. In general, it is extremely unlikely that the API algorithm will converge to the dominant invariant eigenspace of P. Why is that? The entire algorithm is based on approximating the action of P, by replacing the original Lyapunov equation with a tall Sylvester equation. This introduces a truncation error and it is the size of this error which will determine the quality of the final approximation. The situation is similar to that when we attempt to solve a linear PDE using a finite difference method. We may solve the resulting linear system as accurately as we like, but we cannot escape the fact that we have already introduced a discretization error.

8. The Approximate Subspace Iteration

The approximate power iteration was introduced by Hodel [24]. The goal of the algorithm is to compute an approximation of the rank k dominant invariant subspace U_k of the solution P of the Lyapunov equation

$$AP + PA^T + BB^T = 0, \tag{8.1}$$

without computing P explicitly. If P were available explicitly, then we could have computed U_k by subspace iteration, see Chapter 4. In Chapter 7 we described how Hodel approximated the action of P and computed an approximation to U_k. In this chapter we reconsider the problem and show how to apply subspace iteration with Ritz's acceleration to the matrix P, without computing an approximation of P. Our algorithm is based on the same fundamental idea as the API, but we do not compute a basis of the Krylov subspaces $K_l(A^T, V_i)$. We still require that A be negative definite. We present numerical experiments which show that our algorithm can be faster and more accurate than the original scheme.

8.1 The new algorithm

Since P is not available explicitly, we must use the Lyapunov equation (8.1) to approximate the action of P. Let V be any n by r matrix with orthonormal columns. At this point the value of r is not important, but ultimately we will pick $r \geq k$, which is why we introduce this extra variable. Like Hodel, we post-multiply the Lyapunov equation (8.1) with V and find

$$A(PV) + (PV)(V^T A^T V) + P(I - VV^T)A^T V + BB^T V = 0.$$

As we pointed out in Chapter 7 the term

$$P(I - VV^T)A^T V,$$

need not be small, but we will use the Sylvester equation

$$AY + Y(V^T A^T V) + BB^T V = 0, \tag{8.2}$$

as an approximation and consider the truncation error in Section 8.2. Since A is negative definite, both A and $V^T A^T V$ are stable, which imply that this equation has a unique solution Y. Given V we shall use $Y = Y(V)$ as an approximation for PV. It is clear that $V^T Y(V)$ need not be a symmetric, while the target $V^T PV$ is always symmetric.

We can immediately write down the following approximate subspace iteration with Ritz acceleration: Algorithm 16. It clear that the symmetrization step 4 is never a bad idea: Suppose that T and Z are square matrices of the same size, and T is symmetric, then

$$\begin{aligned}\left\| T - \frac{1}{2}(Z + Z^T) \right\|_2 &= \left\| \frac{1}{2}(T - Z) + \frac{1}{2}(T - Z^T) \right\|_2 \\ &\leq \frac{1}{2}\|T - Z\|_2 + \frac{1}{2}\|T^T - Z^T\|_2 = \|T - Z\|_2.\end{aligned}$$

We see that the approximation error does not increase when we replace Z with its symmetric part, $\frac{1}{2}(Z + Z^T)$. It turns out that the symmetrization step is critical.

8.2 Analysis of the new algorithm

Algorithm 16 is based on the assumption that the term $P(I - VV^T)A^T V$ can eventually be ignored. The difference between our target, PV, and our approximation $Y = Y(V)$ satisfies the Sylvester equation

$$A(PV - Y) + (PV - Y)H + P(I - VV^T)A^T V = 0. \tag{8.3}$$

Since A is negative definite, we immediately have the following formula

$$PV - Y = \int_0^\infty e^{tA} P(I - VV^T)A^T V e^{tH} dt, \tag{8.4}$$

from which it follows that

$$\|PV - Y\|_2 \leq \frac{-1}{2\mu(A)} \|P(I - VV^T)\|_2 \|A\|_2, \tag{8.5}$$

Input: An n by r matrix V_0 with orthonormal columns, a positive integer k.

Output: An approximation of the rank k dominant invariant eigenspace of P.

1: **for** $i = 0, 1, 2 \ldots$ **do**

2: Solve $AY_j + Y_j V_j^T A^T V_j + BB^T V_j = 0$ for Y_j.

3: Set $S_j := V_j^T Y_j$.

4: Symmetrize: $S_j := (S_j + S_j^T)/2$.

5: Compute the eigenvalue decomposition $S_j = W_j \Lambda_j W_j^T$, where the eigenvalues of S_j are sorted by decreasing magnitude.

6: Set $Y_j := Y_j W_j$

7: QR factorization: $V_{j+1} R_{j+1} = Y_j$.

8: **end for**

Algorithm 16: Approximate Subspace Iteration with Ritz acceleration

where $\mu(A) = \lambda_{\max}(A + A^T) < 0$ is the log norm of A. The term

$$\|P(I - VV^T)\|_2,$$

measures the 2-norm of the restriction of P to the orthogonal complement of $\operatorname{Ran} V$. If V is sufficiently close to U_r, the rank r dominant eigenspace of P, then

$$\|P(I - VV^T)\|_2 \leq 2\lambda_{r+1}.$$

It follows that if λ_{r+1} is small and if V is sufficiently close to U_r, then $Y(V)$ is a good approximation of PV. However, this observation is not very useful, because we have no systematic way of selecting a good initial guess for U_r.

Hodel showed how to use the block Arnoldi algorithm to extend V to a matrix W with orthonormal columns, so that the corresponding truncation error

$$P(I - WW^T)A^TW \tag{8.6}$$

vanishes, see Theorem 7.2.1. In our case we have the following theorem.

Theorem 8.2.1 *Let A be a real n by n negative definite matrix and let B be a real n by p matrix. Let P be the solution of the the corresponding Lyapunov equation,*

$$AP + PA^T + BB^T = 0.$$

Let k denote the rank of P. Let V be any n by k matrix with orthonormal columns. Let Y be the solution of the Sylvester equation (8.2)*. If Y has full rank k, then $\operatorname{Ran} Y = \operatorname{Ran} P$.*

Proof Since P is symmetric positive semidefinite of rank k, it can be written in the form $P = U\Sigma U^T$, where U is an n by k matrix with orthonormal columns and Σ is a k by k diagonal matrix with positive diagonal entries. By Theorem 2.6.4 the range of P is exactly the Krylov subspace $K(A, B)$, which implies that

$$AU = UH,$$

for a suitable k by k matrix H. Now, let V be any n by k matrix with orthonormal columns. Then the difference between PV and Y is given by

$$\begin{aligned} PV - Y &= \int_0^\infty e^{At} P(I - VV^T) A^T V e^{V^T A^T V t} dt \\ &= \int_0^\infty e^{At} U \Sigma U^T (I - VV^T) A^T V e^{V^T A^T V t} dt \\ &= U \left(\int_0^\infty e^{tH} \Sigma U^T (I - VV^T) A^T V) e^{V^T A^T V t} dt \right). \end{aligned}$$

By writing Y as $PV - (Y - PV)$ we find,

$$Y = U \left(\Sigma U^T V - \int_0^\infty e^{tH} \Sigma U^T (I - VV^T) A^T V) e^{V^T A^T V t} dt \right),$$

from which it follows that if Y has full rank k, then $\operatorname{Ran} Y = \operatorname{Ran} P$. ■

This theorem shows that if P has rank k and if we choose any n by k matrix V with orthonormal columns, then $\operatorname{Ran} Y = \operatorname{Ran} P$, unless Y is rank deficient, and this condition is relatively easy to detect. The reader will notice that the theorem and the proof is an extension of the result derived in Chapter 7, specifically Theorem 7.5.2. The only difference is that we now have to entertain the possibility that Y might be rank deficient.

In the worst case P has rank n and it is out of the question to use $k = n$ vectors in V. However, it is well known that the nonzero eigenvalues of P will often decay very rapidly, and this knowledge motivated us to experiment with increasing values of k. We used the simple approximate subspace iteration, which is given as Algorithm 17. We found experimentally, that if k is chosen sufficiently large, then the size and the effect of the truncation errors will vanish after a few iterations. These observations lead to the application of Ritz-acceleration and the formulation of Algorithm 16.

8.3 Solving the Sylvester equation

The most time consuming step in Algorithms 16 and 17 is the solution of Sylvester equations of the type

$$AX + XH + M = 0, \tag{8.7}$$

Input: An n by k matrix V_0 with orthonormal columns, a positive integer k.

Output: An approximation of the rank k dominant invariant eigenspace of P.

1: **for** $i = 0, 1, 2 \ldots$ **do**

2: Solve $AY_j + Y_j V_j^T A^T V_j + BB^T V_j = 0$ for Y_j.

3: QR factorization: $V_{j+1} R_{j+1} = Y_j$.

4: **end for**

Algorithm 17: Approximate Subspace Iteration

where $H = V^T A^T V$ and $M = BB^T V$. Let n be the dimension of A and let m be the dimension of H.

Currently there is at least two different ways to solve such an equation. The first method is the variant of Bartels-Stewart's method, which we described in Chapter 6. The fundamental problem is that we have to solve a sequence of linear systems with different coefficient matrices. This can be extremely time consuming.

The second approach is due to Sorensen and is based on the implicit restarted Arnoldi (IRA) method which he pioneered. The principle is easy to explain. Suppose we have computed a partial Schur decomposition

$$\begin{bmatrix} A & M \\ & -H \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} R,$$

where $V_1^T V_1 + V_2^T V_2 = I_m$, and R is a real quasi upper triangular matrix. Now if V_2 is nonsingular, then $X = V_1 V_2^{-1}$ is the solution of (8.7) because

$$AV_1V_2^{-1} + M = V_1 R V_2^{-1} = -V_1 V_2^{-1} H.$$

Sorensen has shown that V_2 is nonsingular if and only if the eigenvalues of $-H$ and M are the same. It is clear that in general,

$$\sigma(R) \subseteq \sigma\left(\begin{bmatrix} A & M \\ & -H \end{bmatrix}\right) = \sigma(A) \cup \sigma(-H), \tag{8.8}$$

and since we want $\sigma(R) = \sigma(-H)$ we must locate those eigenvectors for $\begin{bmatrix} A & M \\ & -H \end{bmatrix}$, which correspond to eigenvalues with positive real part. The IRA method can accomplish this goal with varying degrees of success. Sorensen and his group are currently investigating ways of accelerating the IRA approach.

Both schemes suffer from the same fundamental difficulty, namely that there is a very large number of memory references compared with the number of arithmetic operations performed.

8.4 Practical stopping criteria

If the rank k dominant eigenspace $U = U_k$ of P is known, then we can directly monitor the size of the subspace identification error, i.e.

$$\|UU^T - V_i V_i^T\|_2 = \|U - V_i V_i^T U\|_2, \quad i = 1, 2, \ldots.$$

In practice, U is unknown and we cannot calculate the subspace identification error. Instead we can measure the distance between successive approximations,

$$\|V_i V_i^T - V_{i+1} V_{i+1}^T\|_2 = \|V_i - V_{i+1} V_{i+1}^T V_i\|_2, \tag{8.9}$$

which is exactly the approach taken by Hodel, Tenison and Poolla [25]. This is a very economical way to determine if the algorithm is stagnating.

8.5 Solving the corresponding Lyapunov equation

If U_1 spans the k'th dominant eigenspace and if U_2 spans $U_1^\perp$, then P can be written as

$$P = U_1 \Sigma_1 U_1^T + U_2 \Sigma_2 U_2^T, \tag{8.10}$$

where Σ_1, and Σ_2 are matrices of dimension k and $n - k$ respectively. They are not necessarily diagonal. It is easy to recover Σ_1 from U_1, because

$$0 = U_1^T(AP + PA^T + BB^T)U_1 = (U_1^T A U_1)\Sigma_1 + \Sigma_1(U_1^T A^T U_1) + (U_1^T B)(U_1^T B)^T, \tag{8.11}$$

is only a k by k Lyapunov equation in Σ_1.

It follows that we can turn Algorithm 16 into a solver of Lyapunov equations. Given an approximation V_i of the rank k dominant eigenspace U_1, we can solve the reduced order Lyapunov equation

$$A_i \Sigma^{(i)} + \Sigma^{(i)} A_i^T + B_i B_i^T = 0, \quad A_i = V_i^T A V_i, \tag{8.12}$$

for $\Sigma^{(i)}$ and use

$$P_i = V_i \Sigma^{(i)} V_i^T, \tag{8.13}$$

as an approximation to P. The corresponding residual R_i is given by

$$R_i = AV_i\Sigma^{(i)}V_i^T + V_i\Sigma^{(i)}V_i^TA^T + BB^T = E_iF_i^T, \tag{8.14}$$

where

$$E_i = \begin{bmatrix} AV_i\Sigma^{(i)}, & V_i, & B \end{bmatrix}, \quad \text{and} \quad F_i = \begin{bmatrix} V_i, & AV_i\Sigma^{(i)}, & B \end{bmatrix}. \tag{8.15}$$

Calculating the Frobenius norm of this residual in a manner which is both efficient and reliable is a nontrivial task, which we discuss in detail in another chapter.

If the residual stagnates at a level which is deemed unacceptable, then this can be taken as evidence that we have chosen a value of k which is too small, in the sense that P does not admit a good approximation of rank k, and we should instead increase k, while maintaining $k \leq r$.

8.6 Applications to the discrete time Lyapunov equation

Subspace iteration with Ritz acceleration applies equally well to discrete time Lyapunov equations,

$$P = APA^T + BB^T, \tag{8.16}$$

where A is a square matrix with $\|A\|_2 < 1$. Again, the only problem lies in approximating the action of P on any n by r matrix V with orthonormal columns. Hodel's idea is equally applicable in this case. We have

$$PV = A(PV)(V^TA^TV) + AP(I - VV^T)A^TV + BB^TV = 0,$$

and if Y is the solution of

$$Y = AY(V^TA^TV) + BB^TV,$$

then we can use Y as an approximation of PV. This is a Stein equation for Y, which is uniquely solvable, because $\|A\|_2 < 1$ implies $\|V^TA^TV\|_2 < 1$, and therefore $\lambda\mu \neq 1$ for all $\lambda \in \sigma(A)$ and $\mu \in \sigma(V^TA^TV)$. The Stein equation can be solved by orthogonal reduction of V^TA^TV to lower triangular form. It is clear that the difference between

what we want, namely PV, and what can easily be obtained, namely Y, satisfies the equation

$$PV - Y = A(PV - Y)V^T A^T V + AP(I - VV^T)A^T V,$$

which allows us to estimate the difference between PV and Y in a manner quite similar to the continuous time case. Specifically, we have

$$PV - Y = \sum_{j=0}^{\infty} A^j (AP(I - VV^T)A^T V)(V^T A^T V)^j.$$

Now, if P has rank r and V is an n by r matrix with orthonormal columns, and if $Y = Y(V)$ is not rank deficient, then

$$\operatorname{Ran} Y = \operatorname{Ran} P.$$

8.7 Numerical experiments

In this section we report on some numerical experiments which illustrate properties of the original API and our new algorithm, the approximate subspace iteration with Ritz acceleration. We have successfully solved every relevant Sylvester equation using the variant of Bartels-Stewart's method described in Chapter 6.

Let $n = 10^3$ and let A_1, and A_2 be the n by n Toeplitz matrices given by

$$A_1 = \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}, \quad \text{and} \quad A_2 = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}, \tag{8.17}$$

and let

$$B = (1, 1, \ldots, 1)^T \in \mathbb{R}^n.$$

It is clear that A_1, and A_2 are both negative definite. The matrices are chosen because they are nearly indefinite, and while A_1 is symmetric, A_2 cannot be diagonalized. While the standard Arnoldi method can be applied to these problems, they yield very poor results, see Chapter 6. Optimal ADI shift parameters can be found for

symmetric matrices, but there are no convergence results for the ADI applied to non-diagonalizable matrices, which is yet another reason to work with A_2.

Let X_i be the solution of the Lyapunov equation

$$A_i X_i + X_i A_i + BB^T = 0, \tag{8.18}$$

for $i = 1, 2$. Our goal is to compute approximations of the rank $k = 5$ dominant eigenspace $U_k^{(i)}$ of X_i. The matrices are very small and a direct solve is possible in MATLAB. We use the function 'lyap' to compute a pair of approximations $\hat{X}_i$ such that

$$\frac{\|A_1\hat{X}_1 + \hat{X}_1 A_1^T + BB^T\|_2}{\|BB^T\|_2} \approx 7.2 \cdot 10^{-10}, \tag{8.19}$$

and

$$\frac{\|A_2\hat{X}_2 + \hat{X}_2 A_2^T + BB^T\|_2}{\|BB^T\|_2} \approx 2.9 \cdot 10^{-15}. \tag{8.20}$$

We then compute the singular values for $\hat{X}_i$ using the function "svd". The results are given in Figure 8.1. We note that the singular values decay rapidly in both cases, but the decay is faster for X_1, than for X_2. Specifically, the numerical rank of $\hat{X}_i$ with respect to the unit roundoff error $u = 2^{-53}$ is 53 for $\hat{X}_1$ and 111 for $\hat{X}_2$.

To establish a baseline for what is possible when the X_i's are available explicitly, we apply subspace iteration with Ritz acceleration to the explicit matrices $\hat{X}_i$. The subspace identification error is shown in Figure 8.2. In all cases convergence is very fast, which is expected because of the rapid eigendecay. We note that the final subspace identification error is smaller than 10^{-10} for $\hat{X}_1$ and smaller than 10^{-13} for $\hat{X}_2$.

We then apply the original API algorithm using different values of l, the number of Arnoldi steps. The results for $l = 1, 2, 3, 4$ are shown in Figure 8.3. For X_1 we see that increasing l decreases the final subspace identification error which is reached after 15 iterations. For X_2 increasing l decreases the error, but for $l > 2$, the error does not stabilize at a fixed level, but starts to oscillate after 20 iterations.

The results for $l = 5, 10, 15, 20$ are shown in Figure 8.4. For X_1 increasing l continues to steadily decrease the error, while for X_2 the error continues to oscillate

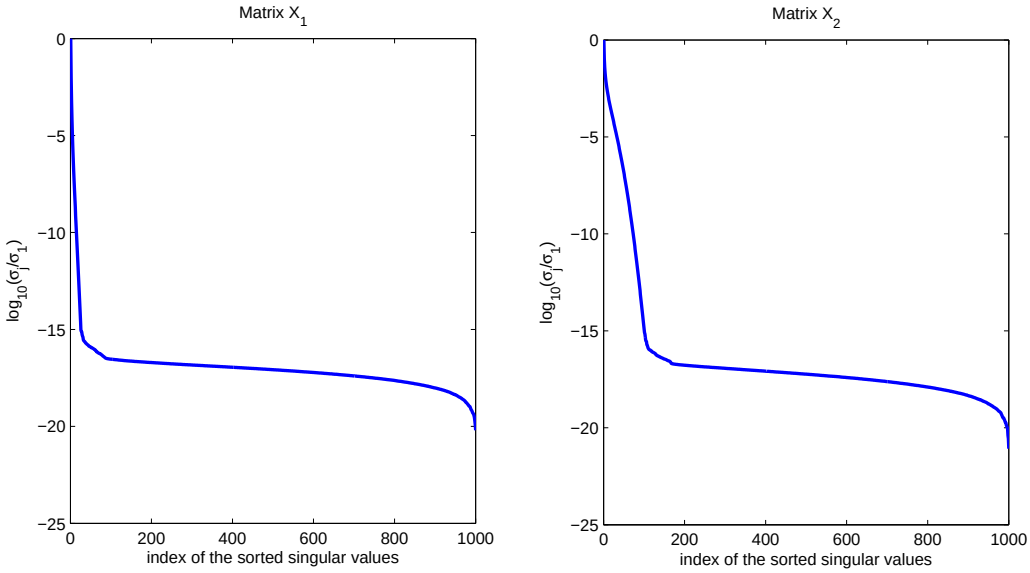

Figure 8.1. The decay of the singular values for the matrices $\hat{X}_1$ (left) and $\hat{X}_2$ (right).

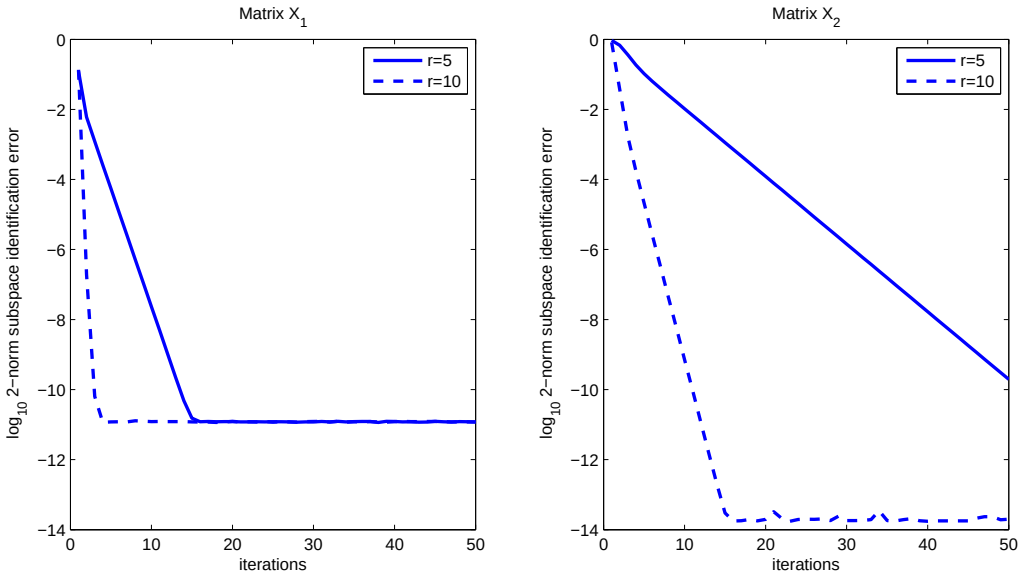

Figure 8.2. The results of applying subspace iteration directly to the matrices $\hat{X}_1$ (left), and $\hat{X}_2$ (right), which are explicitly available.

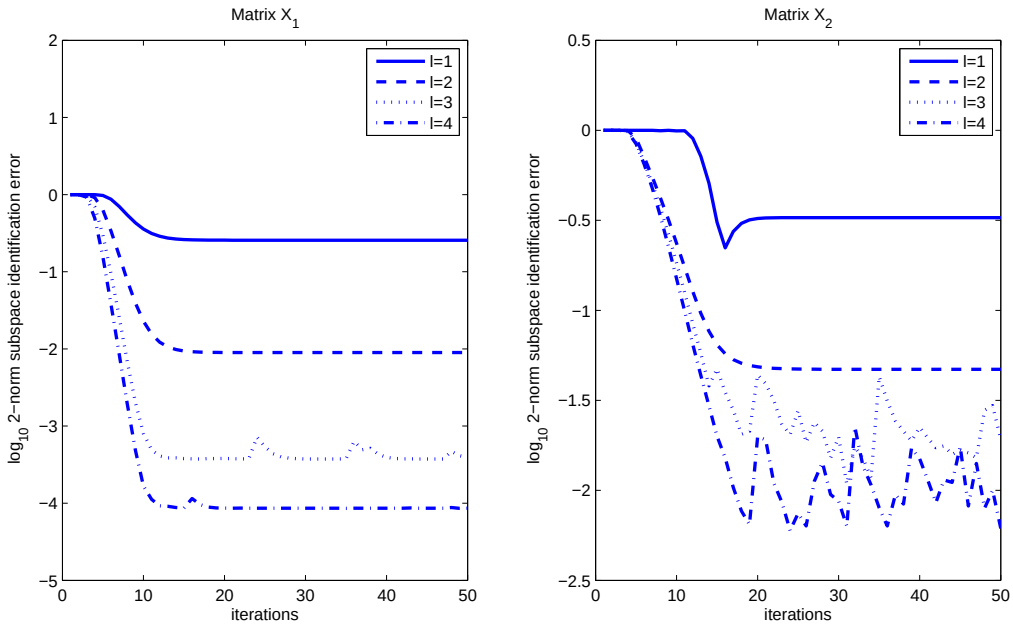

Figure 8.3. The results of applying Hodel's original algorithm to the equations defining X_1 (left) and X_2 (right).

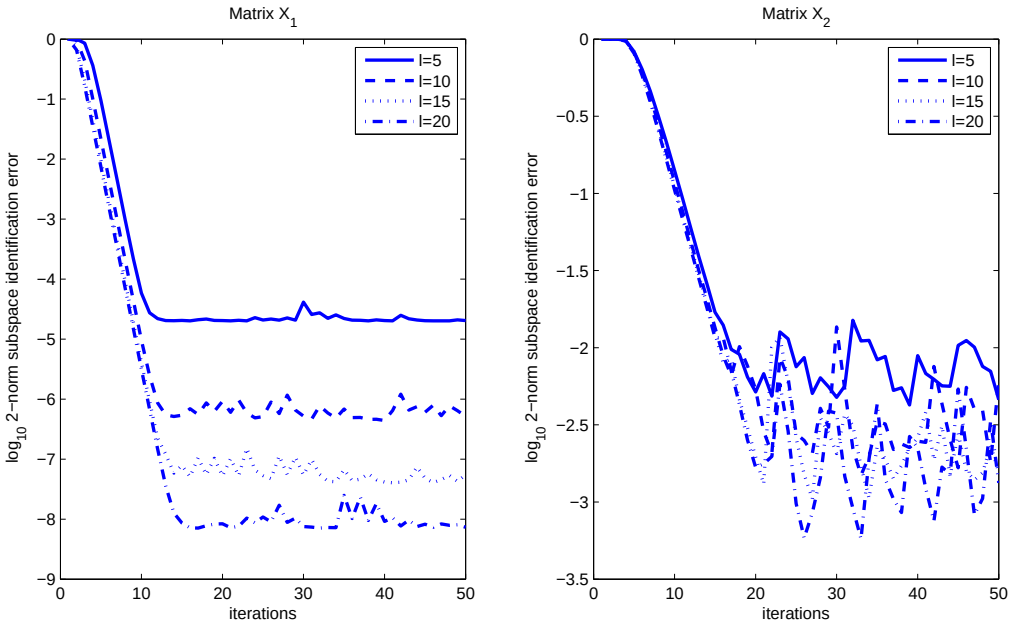

Figure 8.4. The results of applying the original API to the equations defining X_1 (left) and X_2 (right).

after 20 iterations, to the point where there is no clear advantage to using $l = 20$ over $l = 5$. These oscillations pose a serious problem. In practice, we cannot track the subspace identification error directly, because we do not know the target subspace, and we can only measure the difference between the successive iterates. Now, should the algorithm converge, then the difference between successive iterates will tend to zero, and this can be detected and we can stop the iteration. However, if the true error is oscillating violently, then it is unlikely that the difference between any pair of successive iterations will be small, and when the algorithm is eventually terminated, the final approximation need not be any good.

We now apply the approximate subspace iterations with Ritz acceleration to the same pair of problems. The algorithm differs from the standard subspace iterations in that it contains a symmetrization step which may appear somewhat arbitrary. We begin by running 20 iterations of our algorithm, using $r = 10, 20, 30, 40, 50$ vectors,

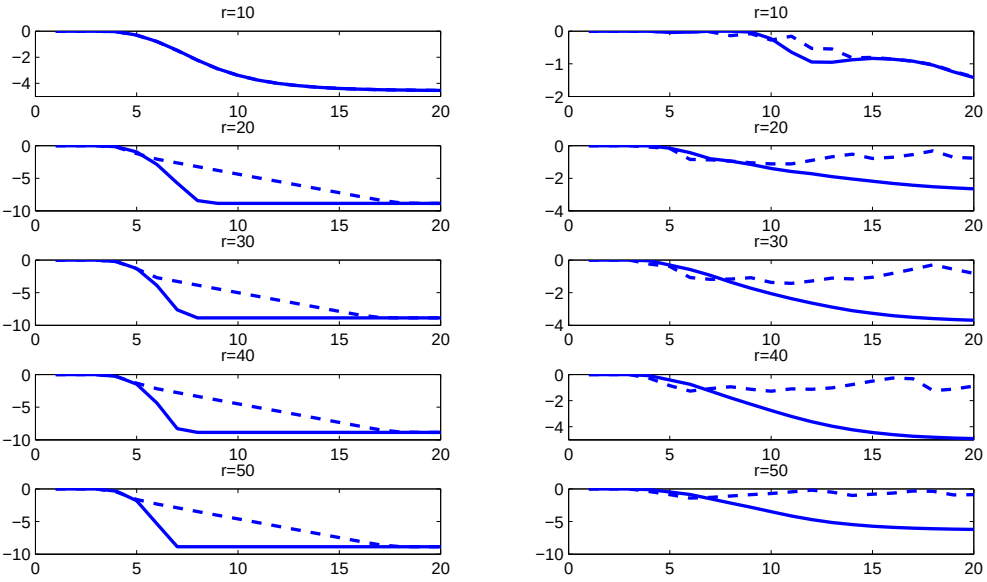

Figure 8.5. The effect of including the symmetrization step (solid lines), as opposed to no symmetrization (dashed curves). The logarithm (base 10) of the Frobenius norm of the subspace identification error is plotted against the number of iterations. The results for matrices $\hat{X}_1$ (on the left) and $\hat{X}_2$ (on the right). The number of vectors is indicated at the top of each diagram. It varies from $r = 10$ vectors (top row), to $r = 50$ vectors (bottom row).

with and without the symmetrization step. The subspace identification errors are depicted in Figure 8.5. In the case of X_1 we see that no symmetrization delays convergence, while for X_2 there is no convergence unless we symmetrize. We conclude that the inexpensive symmetrization step is absolutely necessary and will include it in all future experiments.

We now do a more detailed study of the dependence on r. For X_1 we used $r = 5, 7, 9, 11, 13, 15, 50$. The results are shown in Figure 8.6. In all cases the error has stabilized after at most 20 iterations, and there is a clear advantage of increasing r to a reasonable value, 15 in this case.

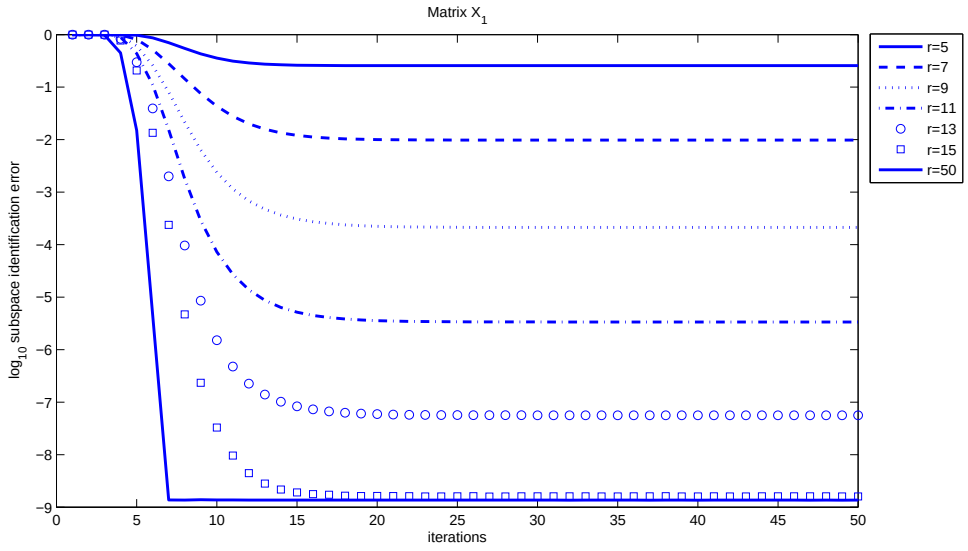

Figure 8.6. The performance of our algorithm applied to X_1 as a function of the number of vectors r.

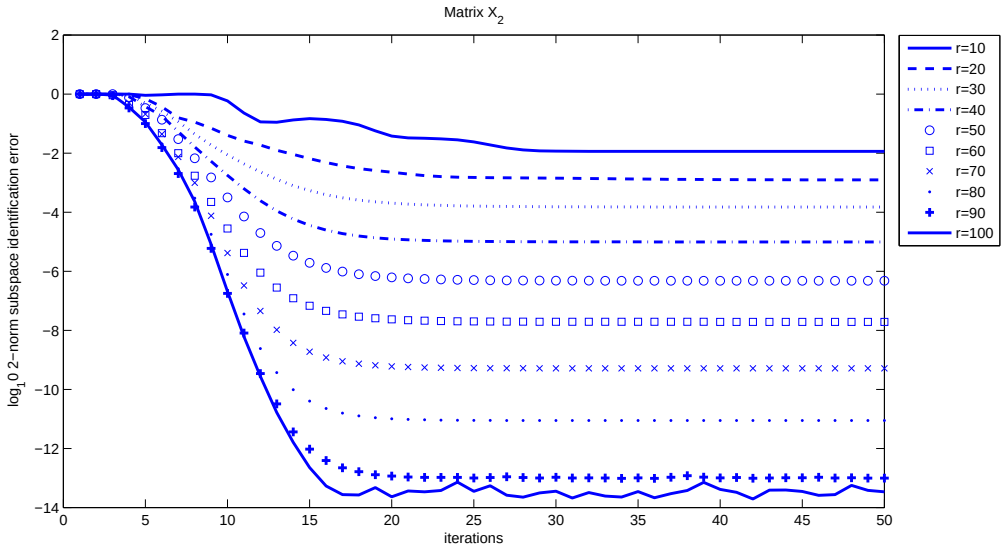

Figure 8.7. The performance of our algorithm applied to matrix X_2 as a function of the number of vectors r.

For X_2 we use $r = 10, 20, \ldots, 100$. The results are depicted in Figure 8.7. Again, we notice that the error stabilizes after 20 iterations, for $r > 10$. We find no clear advantage of using $r \geq 100$ and the error oscillates in this case. In view of our experiment with the explicit subspace iteration we cannot hope for a smaller subspace error, than the one achieved for $r = 100$.

We would like to make an obvious point. When we compute the subspace identification error, we use an approximation for the rank $k = 5$ dominant eigenspace of X_i obtained by applying "eigs" to $\hat{X}_i$. The matrix $\hat{X}_i$ is an approximation of X_i obtained using "lyap". The point is that we are not tracking the "exact" subspace error, only the difference between the iterates and a reasonable approximation of $U_k^{(i)}$.

However, in the case of X_1 there is room for improvement. We can obtain the spectral decomposition of A_1 and use it to compute an improved estimate for X_1. We find $\hat{X}_1'$ such that

$$\frac{\|A_1\hat{X}_1' + \hat{X}_1'A_1^T + BB^T\|_2}{\|BB^T\|_2} \approx 3.5 \cdot 10^{-11}. \tag{8.21}$$

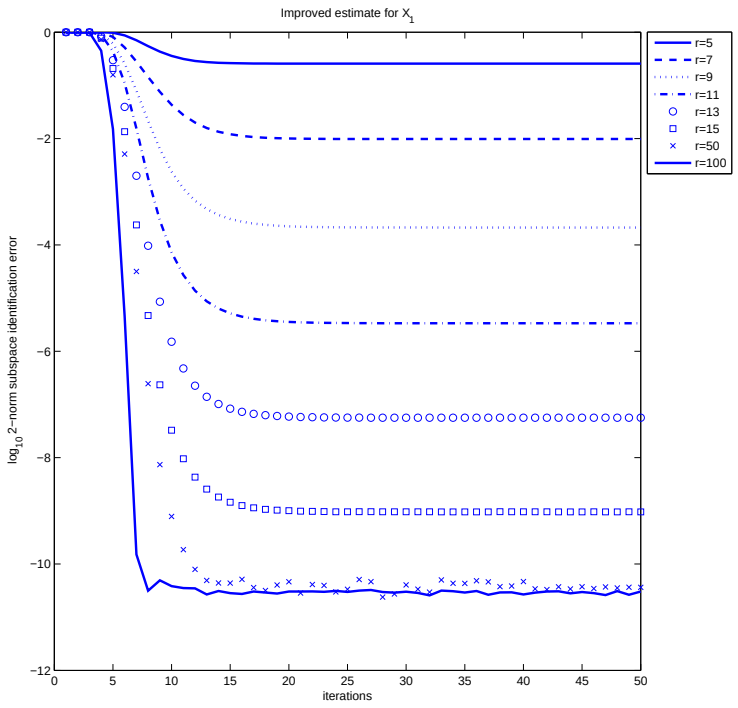

Figure 8.8. The performance of our algorithm applied to the matrix $\hat{X}_1'$, as a function of the number of vectors r. Matrix $\hat{X}_1'$ has a smaller residual than $\hat{X}_1$.

Iterative refinement can reduce the relative residual further, to about $9.6 \cdot 10^{-13}$, but $\hat{X}_1'$ is good enough to illustrate our point. We repeat the previous experiment using $\hat{X}_1'$ in the place of $\hat{X}_1$ to approximate the rank 5 dominant eigenspace for X_1. The subspace identification error is depicted in Figure 8.8. We see that no substantial advantage is realized by using more than $r = 19$ vectors. However, we are now recording subspace identification errors, which are well below 10^{-10}, and comparable with what can be achieved by applying subspace iteration with Ritz acceleration directly to $\hat{X}_1$.

	API			Our algorithm		
Matrix	l	iter	time (s)	r	iter	time (s)
X_1	10	12	0.79	13	11	0.34
X_2	failed			60	12	1.84

Figure 8.9. The parameters and the time needed to identify the rank $k = 5$ dominant eigenspace $U_k^{(i)}$ for X_i, $i = 1, 2$ with an error less than 10^{-6}.

name	dimension	symmetric	definite
bcsstk17	10974	yes	yes
bcsstk18	11948	yes	yes
memplus	17758	no	unknown
af23560	23560	no	unknown

Figure 8.10. Summary of matrices used from Matrix Market.

Table 8.9 offers a brief comparison of the API and our algorithm. Our algorithm is faster than the API, but more significantly, our algorithm succeeds in both cases, while the API fails for X_2.

We now apply our algorithm to a small set of real problems drawn from Matrix Market. A brief summary of the matrices are given in Figure 8.10. We continue to use an inhomogeneous term consisting of a matrix of ones, i.e. $B = (1, 1, \ldots, 1)^T$.

These problems are still quite small, but so large that it would be very expensive to solve the corresponding Lyapunov equations and compute the dominant subspaces by a direct method. Instead we can turn our algorithm into a solver for Lyapunov equations and monitor the relative residual.

For each of the four matrices we run 30 iterations of our method, using $r = 10, 20, \ldots, 100$ vectors. For each approximation to the dominant eigenspace of rank r, we extract the corresponding approximation for the exact solution of the Lyapunov matrix equation and compute the residual.

The results for matrix bcsstk17 are given in Figure 8.11. We have only rendered the results for $r = 10, 20, 30, 40$. We see that the relative residual is larger than 10^{-2} for $r = 10$. This indicates that the numerical rank of the true solution is probably larger than $r = 10$. We also note that the final relative residual continues to decrease as we increase the number of vectors.

The results for matrix bcsstk18 are given in Figure 8.12. As in the case of matrix bcsstk17 the residual curves are clearly determined by the number of vectors used in each iteration. We see that there is a clear advantage to increasing r when the current value is small, but that the gain decreases as we approach $r = 60$.

The results for matrices memplus and af23560 are similar to the previous two cases and are given in Figure 8.13 and 8.14.

We note that it is possible to obtain a relative residual of 10^{-6} in all four cases. The parameters and the runtimes are given in Figure 8.15.

Our runtimes are about 20 times larger than the ones achieved by Sabino [52], who solved the same problems using carefully selected shift parameters together with his own variant of the low rank cyclic Smith method on a comparable architecture. Since every ADI algorithm requires the selection of certain shift parameters, a nontrivial problem which has not been solved for arbitrary stable matrices, our algorithm has a clear advantage in terms of robustness and perhaps even efficiency. The advantage of our approach is that it does not require the selection of any parameters, and if the solution of the Lyapunov equation has low rank or if the singular values decay rapidly, then the dominant eigenspace can be recovered in a few iterations.

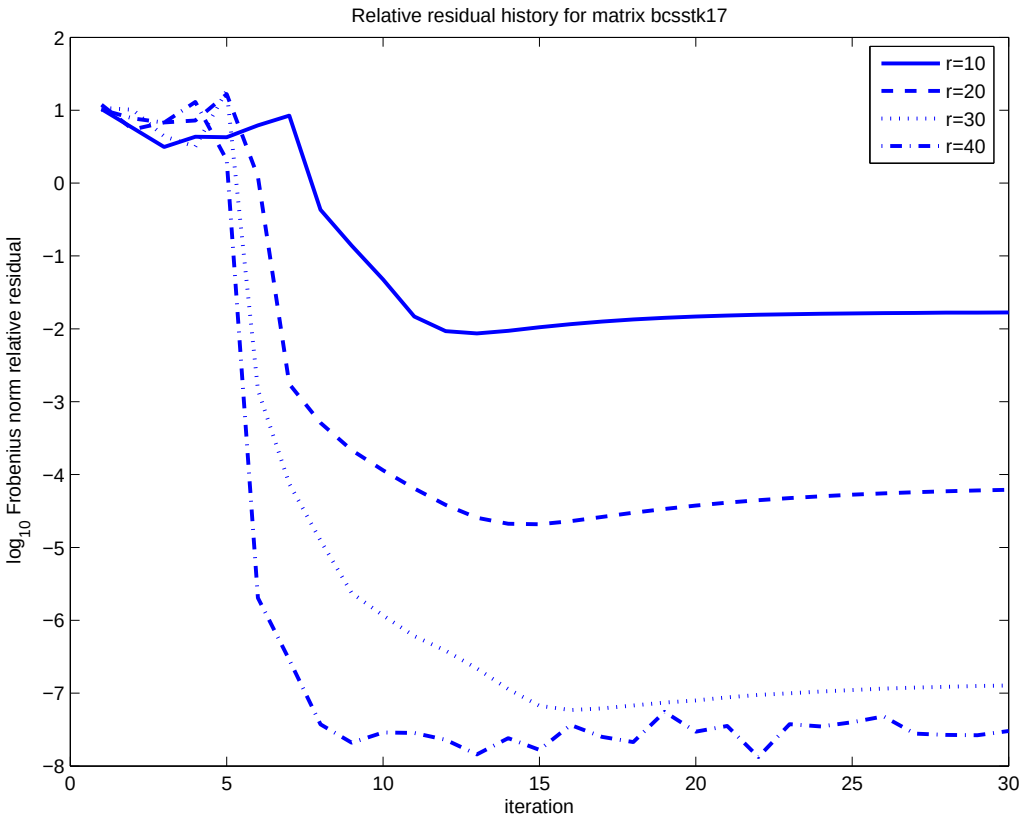

Figure 8.11. The results of applying our algorithm as a solver to the Lyapunov equation defined by matrix bcsttk17, and an inhomogeneous term consisting of ones.

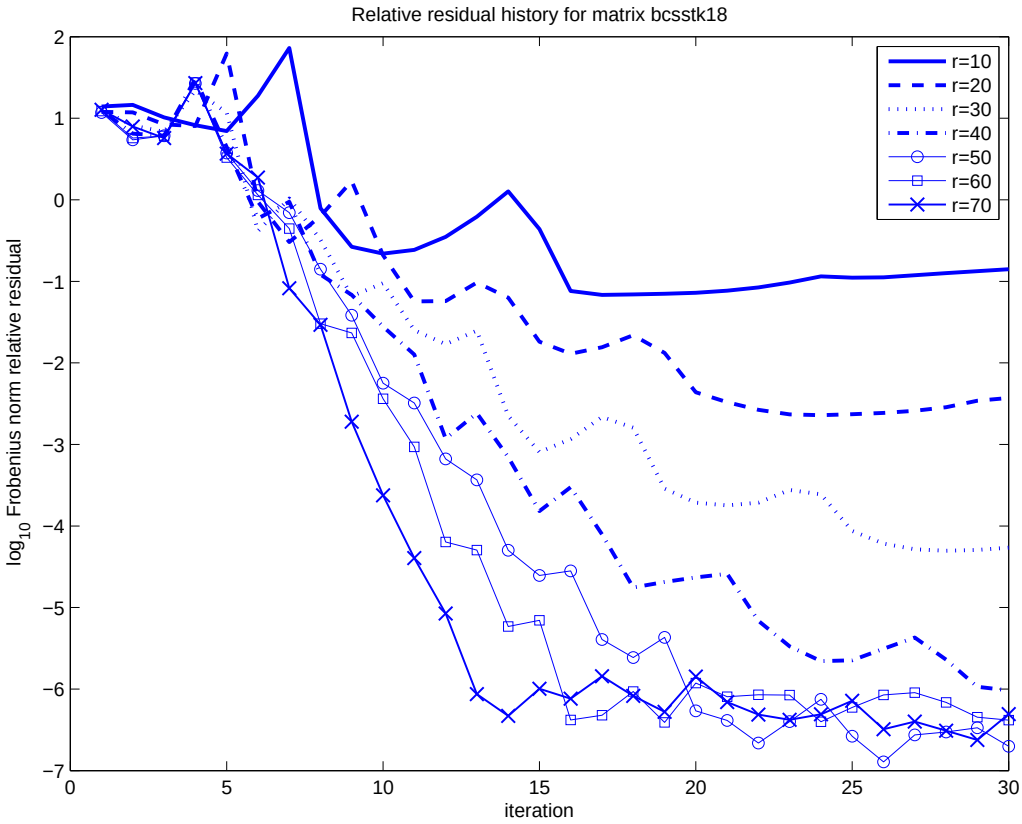

Figure 8.12. The results of applying our algorithm as a solver to the Lyapunov equation defined by matrix bcsttk18, and an inhomogeneous term consisting of ones.

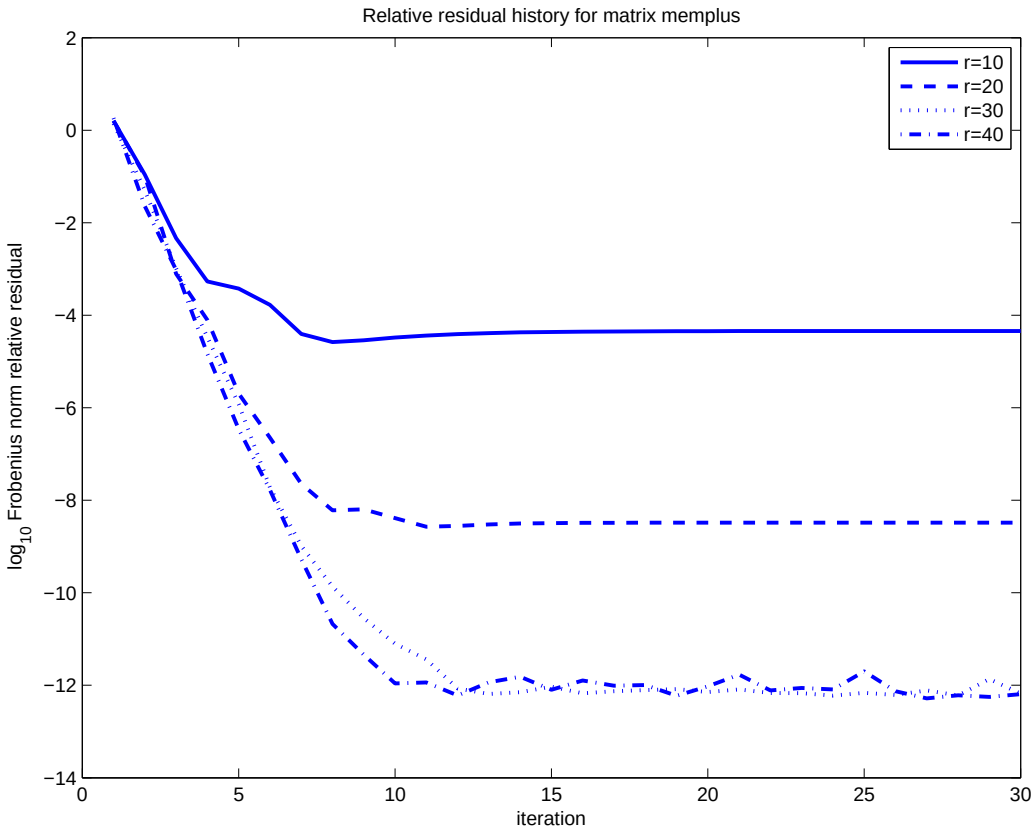

Figure 8.13. The results of applying our algorithm as a solver to the Lyapunov equation defined by matrix memplus, and an inhomogeneous term consisting of ones.

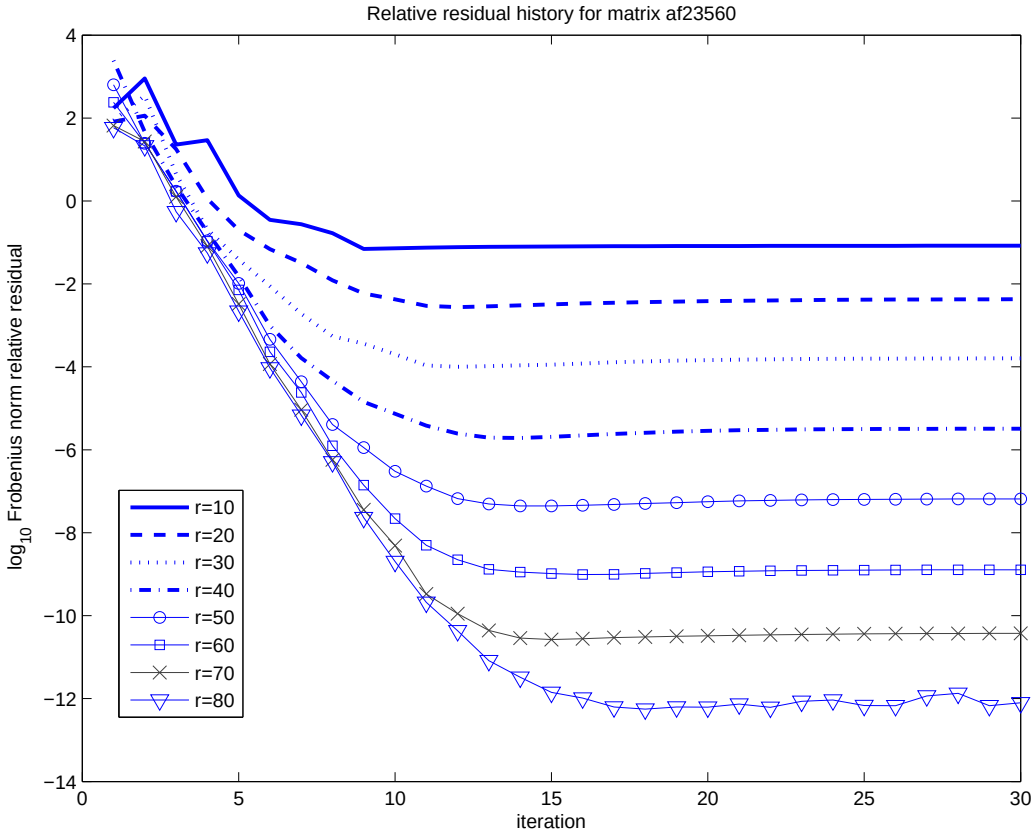

Figure 8.14. The results of applying our algorithm as a solver to the Lyapunov equation defined by matrix af23560, and an inhomogeneous term consisting of ones.

name	r	iter	time(s)	rel. res
bcsstk17	40	7	73	3e-7
bcsstk18	70	13	204	9e-7
memplus	20	6	45	2e-7
af23560	50	10	2513	3e-7

Figure 8.15. Summary of the results of applying our algorithm to the matrices used from Matrix Market.

8.8 Conclusion and future work

We have established our first real contribution to the problem of solving a Lyapunov equation. The main result is Theorem 8.2.1 which states that if P has rank r, then it is possible to recover the range of P by solving a single n by r Sylvester equation. In practice, P does not have low rank, but our experiments show that if the numerical rank is sufficiently low, then the approximate subspace iteration with Ritz acceleration converges in a small number of iterations.

The symmetrization step in our adaption of the Ritz acceleration procedure is reasonable because mathematically it cannot increase the error, but our experiments demonstrate the necessity of performing this step.

Above all we now understand that it is possible to accurately approximate the action of P by solving an n by r Sylvester equation as long as r is sufficiently large. A suitable value of r is difficult to determine beforehand, but there seems to be little gained from using r greater than the numerical rank of P with respect to the machine unit roundoff.

If speed is the only measure of success, then our approach would certainly not be competitive. However, ADI algorithms require the computation of certain shift parameters, and while Sabino [52] has made significant progress, the problem has not been solved for arbitrary stable matrices. The Arnoldi method is viable only for A negative definite, and its convergence can be arbitrary slow. Our algorithm requires A to be negative definite, but if the true solution has low numerical rank, then it converges after a few iterations without requiring any shift parameters.

In general, the block Arnoldi algorithm (ARPACK) requires fewer iterations than subspace iteration in order to identify the dominant eigenspace for an explicit matrix P. For future study, one can apply the ideas presented in this chapter to develop an approximate Arnoldi algorithm. The resulting scheme could be slightly better than our current approach, but the main problem of solving the tall Sylvester equations will remain.

9. The AISIAD algorithm

In this chapter we discuss aspects of the AISIAD algorithm due to Zhou and Sorensen [70]. The abbreviation is short for "Approximate Implicit Subspace Iteration Alternating Directions". The goal is to compute the dominant eigenspaces for the products PQ and QP where P and Q are given implicitly as the solution to a pair of Lyapunov equations

$$AP + PA^T + BB^T = 0,$$
$$A^TQ + QA + C^TC = 0.$$

We explained the importance of this problem in Chapter 3. Here, we consider approximating the actions of P and Q.

We present the algorithm, and using the material developed in Chapters 7 and 8, we describe a simple situation where AISIAD correctly identifies the range of PQ applied to a matrix V with orthonormal columns.

9.1 The algorithm

If P and Q were explicitly available, then it would have been a simple task to use the standard subspace iteration to compute the rank k dominant eigenspace for PQ and the rank k dominant eigenspace for QP. Specifically, we could have used Algorithm 2 from Chapter 3. However, since P and Q are not explicitly available, we must approximate their actions. Zhou and Sorensen [70] adopted Hodel's idea to address this situation and derived Algorithm 18, assuming that A is negative definite.

The actions of P and Q are approximated by solving a pair of tall Sylvester equations. Both equations have unique solutions, because A is negative definite. It

1: Chose V_0 such that $V_0^T V_0 = I_k$.
2: **for** $i = 0, 1, \ldots,$ **do**
3: Solve $A^T X_i + X_i V_i^T A^T V_i + C^T C V_i = 0$.
4: QR-factorization $W_{i+1} S_{i+1} := X_i$.
5: Solve $A Y_i + Y_i W_i^T A^T W_i + B B^T W_i = 0$.
6: QR-factorization $V_{i+1} T_{i+1} := Y_i$.
7: **end for**

Algorithm 18: Basic AISIAD

is clear, that if $A = A^T$ and $C^T = B$, such that $P = Q$, then the AISIAD algorithm reduces to the original API algorithm.

9.2 Elementary analysis

There has been very little progress in advancing the analysis of the AISIAD algorithm for the general case. It is clear that such an analysis is even more difficult than the analysis of Hodel's API algorithm for which even the scalar case (one vector and no Arnoldi-steps) is still not understood. In view of the truncation error made by approximating the actions of P and Q, the best we can hope for is that the AISIAD algorithm converges to a pair of subspaces, which are "close" to the dominant eigenspaces.

In this section we identify one situation where the range of the product PQ is computed exactly. Let Q have rank r, let V be an n by r matrix, and let X be the solution of the Sylvester equation

$$A^T X + X(VA^T V) + C^T CV = 0.$$

By Theorem 8.2.1, if X has full rank r, then $\operatorname{Ran} X = \operatorname{Ran} Q$. By Theorem 2.6.4

$$\operatorname{Ran} Q = K(A^T, C^T).$$

Now, suppose that X is not rank deficient. Then the QR-factorization of X produces an n by r matrix W with orthonormal columns, such that

$$\operatorname{Ran} W = K(A^T, C^T) = \operatorname{Ran} Q.$$

Then by post-multiplying the Lyapunov equation

$$AP + PA^T + BB^T = 0$$

by W we once again see that

$$A(PW) + (PW)A^T + BB^T W + P(I - WW^T)A^T W = 0.$$

However, the special term satisfies

$$P(I - WW^T)A^TW = 0,$$

because $\mathrm{Ran}\, W$ is an A^T invariant subspace. Thus, if X is the solution of the Sylvester equation

$$AX + XA^T + BB^TW = 0,$$

then $X = PW$. It follows that

$$\mathrm{Ran}X = \mathrm{Ran}(PQ)V,$$

and the algorithm correctly identifies the range of PQ applied to V.

In Chapter 8 we showed that if Q has rank r, and if V is any n by r matrix with orthonormal columns, and $Y = Y(V)$ is the solution of

$$A^TY + Y(V^TAV) + C^TC = 0,$$

then $\mathrm{Ran}\, Y = \mathrm{Ran}\, Q$, unless Y is rank deficient, which is easy to detect.

We now investigate the following two questions

1. Is it possible to extend this type of result to the AISAID algorithm?

2. Does the AISIAD algorithm converge in a single iteration provided that PQ has rank k and we use k vectors?

Assume that PQ has rank k, and let V be any n by k matrix with orthonormal columns. Then QV satisfies

$$A^T(QV) + (QV)V^TAV + Q(I - VV^T)AV + C^TCV = 0$$

Let $Y = Y(V)$ be the solution of

$$A^TY + Y(V^TAV) + C^TCV = 0.$$

Then the difference between Y and QV satisfies

$$A^T(QV - Y) + (QV - Y)(V^TAV) + Q(I - VV^T)AV = 0,$$

from which it follows

$$QV - Y = \int_0^\infty e^{A^T t} Q(I - VV^T) A V e^{V^T A V t} dt.$$

Now, since $\operatorname{Ran} Q$ is A^T invariant, it follows that

$$\operatorname{Ran}(QV - Y) \subseteq \operatorname{Ran} Q,$$

from which we conclude that

$$\operatorname{Ran} Y \subseteq \operatorname{Ran} Q.$$

Let $WS = Y$ is be a QR factorization of Y and let X be the solution of

$$AX + X(W^T A^T W) + BB^T W = 0.$$

Since PW satifies the equation

$$A(PW) + (PW)W^T A^T W + P(I - WW^T)A^T W + BB^T W = 0,$$

the difference between X and PW satisfies

$$A(PW - X) + (PW - X)(W^T A^T W) + P(I - WW^T)A^T W = 0,$$

from which it follows

$$PW - X = \int_0^\infty e^{tA} P(I - WW^T) A^T W e^{W^T A^T W} dt.$$

The question now reduces to wether one can conclude that

$$\operatorname{Ran} X = \operatorname{Ran}(PQ).$$

First, however, we make two observations: In view of the splitting

$$X = PW - (PW - X),$$

we would be able to conclude $\operatorname{Ran} X \subseteq \operatorname{Ran}(PQ)$ provided

$$\operatorname{Ran}(PW - X) \subseteq \operatorname{Ran}(PQ),$$

and by splitting the expression for $PW - X$,

$$PW - X = \underbrace{\int_0^\infty e^{tA} P A^T W e^{W^T A^T W} dt}_{\mathrm{Ran}(PA^TW) \subseteq \mathrm{Ran}(PQ)} - \underbrace{\int_0^\infty e^{tA} (PW) W^T A^T W e^{W^T A^T W} dt}_{\mathrm{Ran}(PW) \subseteq \mathrm{Ran}(PQ)},$$

we see that if

$$\mathrm{Ran}(PQ) \text{ is A invariant},$$

then

$$\mathrm{Ran}(PW - X) \subseteq \mathrm{Ran}(PQ),$$

and hence we would be able to conclude

$$\mathrm{Ran}\, X = \mathrm{Ran}(PQ),$$

whenever X had full rank r. Unfortunately, it is not true that $\mathrm{Ran}(PQ)$ is A invariant, except when $A = A^T$, $C^T = B$, and $P = Q$. We provide a simple example to illustrate this point. Let A be the matrix given by

$$A = \begin{bmatrix} -\frac{1}{2} & & -\frac{1}{2} \\ -1 & -1 & \\ & & -1 \end{bmatrix}.$$

It is easy to verify that

$$\lambda_{\max}(A + A^T) \approx -0.2753,$$

and A is negative definite. If B and C are given by

$$B = (1, 1, 0)^T, \quad C = \left(1, 0, \frac{1}{2}\right),$$

then the Gramians P and Q are given by

$$P = \mathrm{diag}\left\{1, \frac{1}{2}, 0\right\}, \quad Q = \mathrm{diag}\left\{1, 0, \frac{1}{8}\right\}.$$

We see that PQ has rank 1, and

$$\mathrm{Ran}(PQ) = \underset{\mathbb{R}}{\mathrm{span}}\,\{e_1\},$$

where $e_1 = (1, 0, 0)^T$. It is obvious that this vectorspace is not A invariant, because

$$A_{2,1} = -1 \neq 0.$$

In short, the range of PQ is not A invariant for general negative definite matrices A, and this particular approach to the analysis of the AISAID algorithm cannot succeed.

9.3 Conclusion

The analysis of the general AISAID algorithm continues to elude the research community. It is clear that the problem is even harder than the scalar case of Hodel's API algorithm, which is still not understood. We have identified a single very simple situation where the algorithm correctly identifies the range of PQ applied to a matrix V with orthonormal columns.

10. Lyapunov equations in Kronecker product form

10.1 Introduction

In this chapter we consider Krylov subspace methods for the Lyapunov matrix equation

$$AX + XA^T + BB^T = 0, \tag{10.1}$$

in Kronecker product form

$$\tilde{A}\text{vec}(X) + \tilde{b} = 0,$$

where

$$\tilde{A} = I \otimes A + A \otimes I$$

and

$$\tilde{b} = \text{vec}(BB^T).$$

Properties of the Kronecker product $\otimes$ and the vec operator are given in Chapter 2.

The Arnoldi algorithm can be used to compute an orthogonal basis for the Krylov subspace

$$K_k(\tilde{A}, \tilde{b}) = \text{Ran}\left[\tilde{b} \quad \tilde{A}\tilde{b} \quad \dots \quad \tilde{A}^{k-1}\tilde{b}\right]$$

using $O(n^2)$ memory and arithmetic operation. Our main goal is to reduce these requirements to $O(n)$. This procedure requires a compact representation of certain vectors in $\mathbb{R}^{n^2}$.

We show that it is possible to run the classical algorithms CG, CGNR, GMRES, and BCG using $O(n)$, rather than $O(n^2)$ memory and time.

We inherit the monotone residual history from GMRES, and we adapt the classical CG error estimate to our situation.

Our compact representation of vectors in $\mathbb{R}^{n^2}$ does not permit preconditioning and we leave the issue of preconditioning to Chapter 11.

10.2 The Arnoldi process for $\tilde{A}$

Let A be a real n by n matrix and let B be a real n by p matrix. Let

$$\tilde{A} = A \otimes I + I \otimes A,$$

and let

$$\tilde{b} = \mathrm{vec}(BB^T).$$

Consider the Krylov subspace $K_j(\tilde{A}, \tilde{b})$ defined by

$$K_j(\tilde{A}, \tilde{b}) = \mathrm{span}\{\tilde{b}, \tilde{A}\tilde{b}, \tilde{A}^2\tilde{b}, \ldots, \tilde{A}^{j-1}\tilde{b}\} \subseteq \mathbb{R}^{n^2}.$$

It is clear that

$$K_j(\tilde{A}, \tilde{b}) \subseteq K_{j+1}(\tilde{A}, \tilde{b}),$$

and there exist $\tilde{m}$ such that

$$K_j(\tilde{A}, \tilde{b}) = K_{\tilde{m}}(\tilde{A}, \tilde{b})$$

for all $j \geq \tilde{m}$. The smallest $\tilde{m}$ such that

$$K_j(\tilde{A}, \tilde{b}) = K_{\tilde{m}}(\tilde{A}, \tilde{b}), \quad j \geq \tilde{m},$$

is called the grade of $\tilde{b}$ with respect to $\tilde{A}$.

It is theoretically possible to compute an orthonormal basis for $K_k(\tilde{A}, \tilde{b})$ using the basic Arnoldi algorithm, which is restated as Algorithm 19. We need this algorithm for the statement and proof of the main result, Theorem 10.3.1. The tilde superscript is used to emphasize that we are computing in $\mathbb{R}^{n^2}$ rather than $\mathbb{R}^n$.

Algorithm 19 produces a factorization of the type

$$\tilde{A}\tilde{V}_k = \tilde{V}_k\tilde{H}_k + \tilde{h}_{k+1,k}\tilde{v}_{k+1}e_k^T,$$

where $\tilde{H}_k = \left[\tilde{h}_{ij}\right]$ is a k by k upper Hessenberg matrix and the columns $\tilde{v}_j$ of $\tilde{V}_k$ are orthonormal and span $K_k(\tilde{A}, \tilde{b})$. The vector e_k is the last column of the k by k identity matrix. If the initial value of k is greater than or equal to $\tilde{m}$, then the

1: $\tilde{v}_1 := \tilde{b}/\|\tilde{b}\|_2$
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $\quad \tilde{w}_j := \tilde{A}\tilde{v}_j$
4: $\quad$ **for** $i = 1, \ldots, j$ **do**
5: $\quad\quad \tilde{h}_{ij} := \tilde{v}_i^T \tilde{w}$
6: $\quad\quad \tilde{w}_j := \tilde{w}_j - \tilde{v}_i \tilde{h}_{ij}$
7: $\quad$ **end for**
8: $\quad \tilde{h}_{j,j+1} := \|\tilde{w}\|_2$
9: $\quad$ **if** $\tilde{h}_{j,j+1} = 0$ **then**
10: $\quad\quad \tilde{v}_{j+1} := 0$
11: $\quad\quad k := j$
12: $\quad\quad$ **exit**
13: $\quad$ **else**
14: $\quad\quad \tilde{v}_{j+1} := \tilde{w}/\tilde{h}_{j,j+1}$
15: $\quad$ **end if**
16: **end for**

Algorithm 19: Basic Arnoldi algorithm for the pair $(\tilde{A}, \tilde{b})$

algorithm terminates at step 12 after $\tilde{m}$ iterations, otherwise it terminates at step 16 after k iterations.

If $\mathrm{nz}(A)$ is $O(n)$, then running $k \leq \tilde{m}$ steps of Algorithm 19 requires $O(n^2k^2)$ flops and $O(n^2k + k^2)$ words of storage. This limits its application to small values of n. Our main result, Theorem 10.3.1, offers a shortcut which requires $O(np^2k^2 + p^3k^4)$ flops and $O(npk + p^2k^3)$ words of storage, and to this end we require a block Arnoldi algorithm.

In Chapter 6 we stated one version of the block Arnoldi algorithm (Algorithm 11). This algorithm can be used to compute an orthonormal basis for the block Krylov subspace

$$K_j(A, B) = \mathrm{span}\{B, AB, \ldots, A^{j-1}B\} \subseteq \mathbb{R}^n.$$

Let m denote the grade of B with respect to A. If we run $k \leq m$ steps of Algorithm 11, then we obtain a matrix V_k given by

$$V_k = \begin{bmatrix} Q_1 & Q_2 & \ldots & Q_k \end{bmatrix},$$

with orthonormal columns spanning $K_k(A, B)$. If we define H_k as the upper block Hessenberg matrix given by

$$H_k = \begin{bmatrix} H_{11} & H_{12} & \ldots & \ldots & H_{1k} \\ H_{21} & H_{22} & \ldots & \ldots & \vdots \\ 0 & H_{32} & H_{33} & \ldots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H_{k,k-1} & H_{kk} \end{bmatrix},$$

then

$$AV_k = V_kH_k + Q_{k+1}H_{k+1,k}E_k, \quad k < m,$$

where E_k consists of the last p_k columns of the n_k by n_k identity matrix, and

$$AV_m = V_mH_m, \quad k = m.$$

The parameters p_k and n_k are computed by the algorithm in order to keep track of the partitioning of the matrices V_k and H_k. Specifically, V_k is an n by n_k matrix and

H_{ij} is a p_i by p_j matrix. If we attempt to run $k > m$ steps of Algorithm 11, then it terminates normally after $k = m$ steps.

This is the traditional view of the block Arnoldi algorithm. We would like to adapt a slightly different perspective. We will use V_m and H_m to define a set of matrices $\{V_j\}_{j=1}^{\infty}$ and $\{\bar{H}_j\}_{j=1}^{\infty}$ as follows. Let

$$V_j = \begin{cases} V_m(:, 1 : n_j) & \text{for } j < m \\ V_m & \text{for } j \geq m \end{cases},$$

and

$$\bar{H}_j = \begin{cases} H_j(1 : n_{j+1}, 1 : n_j) & \text{for } j < m \\ H_m & \text{for } j \geq m \end{cases}.$$

Then,

$$AV_j = V_j \bar{H}_j, \quad j = 1, 2, \ldots.$$

The advantage is purely notational. This type of factorization allows us to do induction on j without concerning ourselves with whether $j < m$ or $j \geq m$. This simplifies the analysis of algorithms such as CG, where we might be interested in executing more than m iterations.

10.3 The main result

In this section we establish a simple relation between the Krylov subspaces

$$K_j(A, B) \subseteq \mathbb{R}^n$$

and

$$K_j(\tilde{A}, \tilde{b}) \subseteq \mathbb{R}^{n^2},$$

where $\tilde{A} = A \otimes I + I \otimes A$ and $\tilde{b} = \text{vec}(BB^T)$.

Theorem 10.3.1 *Let A, B, V_j, and n_j be as in section 10.2. Then there exist symmetric n_j by n_j matrices Π_j, such that the vectors $\tilde{v}_j$ produced by applying the*

basic Arnoldi algorithm to the matrix $\tilde{A} = I \otimes A + A \otimes I$*, and the vector* $\tilde{b} = \mathrm{vec}(BB^T)$ *satisfy*

$$\tilde{v}_j = \mathrm{vec}(V_j \Pi_j V_j^T).$$

Proof We begin by showing that the theorem is true for $i = 1$. Let $\beta = \|\tilde{b}\|_2$. Initially, the block Arnoldi algorithm does a rank revealing QR factorization of B,

$$BP = \begin{bmatrix} \hat{Q}_1 & \hat{Q}_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

where P is a permutation matrix, from which it follows

$$BB^T = \begin{bmatrix} \hat{Q}_1 & \hat{Q}_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} P^{-1} P^{-T} \begin{bmatrix} R_{11}^T & 0 \\ R_{12}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{Q}_1^T \\ \hat{Q}_2^T \end{bmatrix}$$
$$= \hat{Q}_1 (R_{11} R_{11}^T + R_{12} R_{12}^T) \hat{Q}_1^T.$$

Since $V_1 = \hat{Q}_1$ by definition, we see that

$$\tilde{v}_1 = \tilde{b}/\beta = \mathrm{vec}(BB^T)/\beta = \mathrm{vec}(V_1 \Pi_1 V_1^T),$$

provided we choose

$$\Pi_1 = (R_{11} R_{11}^T + R_{12} R_{12}^T)/\beta.$$

It is clear that Π_1 is a symmetric n_1 by n_1 matrix. Now, suppose that the statement is true for $i = 1, 2, \ldots j$. By the Arnoldi algorithms and the properties of the Kronecker product

$$\tilde{w}_j := \tilde{A}\tilde{v}_j = \mathrm{vec}(AV_j \Pi_j V_j^T + V_j \Pi_j V_j^T A^T) = \mathrm{vec}(V_{j+1} \bar{H}_j \Pi_j V_j^T + V_j \Pi_j \bar{H}_j^T V_{j+1}^T).$$

We claim that there exists a n_{j+1} by n_{j+1} symmetric matrix Γ_j such that

$$\tilde{w}_j = \mathrm{vec}\left(V_{j+1} \Gamma_j V_{j+1}^T\right). \tag{10.2}$$

The key to finding Γ_j is to use the fact that $V_{j+1} V_{j+1}^T$ is the orthogonal projection onto the column space of V_{j+1}, which includes the column space of V_j. Thus

$$V_j = V_{j+1} V_{j+1}^T V_j,$$

and we may write

$$V_{j+1}\bar{H}_j\Pi_j V_j^T + V_j\Pi_j\bar{H}_j^T V_{j+1}^T = V_{j+1}\left(\bar{H}_j\Pi_j V_j^T V_{j+1} + {V_{j+1}}^T V_j\Pi_j\bar{H}_j^T\right)V_{j+1}^T.$$

We see that

$$\Gamma_j := \bar{H}_j\Pi_j V_j^T V_{j+1} + V_{j+1}^T V_j\Pi_j\bar{H}_j^T, \tag{10.3}$$

is the choice which satisfies equation (10.2). Clearly, if Π_j is symmetric, then Γ_j is symmetric. The inner loop of the Arnoldi algorithm consists of the linear updates

$$\tilde{w}_j := \tilde{w}_j - \tilde{v}_i\tilde{h}_{ij},$$

where $\tilde{h}_{ij} = \tilde{v}_i^T\tilde{w}_j$, which correspond to the linear updates

$$\Gamma_j := \Gamma_j - {V_{j+1}}^T V_i\Pi_i V_i^T V_{j+1}\tilde{h}_{ij}. \tag{10.4}$$

This observation shows that the compact representation of $\tilde{w}_j$ can be maintained and if Γ_j is symmetric before the inner loop, then the final value of Γ_j will be symmetric as well.

After the inner loop has been completed, $\tilde{h}_{j+1,j} = \|\tilde{w}_j\|_2$ is computed and compared to zero. If $\tilde{h}_{j+1,j} = 0$, then the algorithm terminates, otherwise

$$\tilde{v}_{j+1} = \tilde{w}_j/\tilde{h}_{j+1,j},$$

which corresponds to

$$\Pi_{j+1} = \Gamma_j/\tilde{h}_{j+1,j},$$

and we have

$$\tilde{v}_{j+1} = \mathrm{vec}\left(V_{j+1}\Pi_{j+1}V_{j+1}^T\right).$$

This completes the proof. ■

In this proof we suppressed certain computational aspects of Theorem 10.3.1. Specifically, we used the fact that β and $\tilde{h}_{ij}$ are real numbers, and we gave a formula for Γ_j, but we did not show how to carry out these computations. We now address these questions.

By definition,

$$\beta = \|\tilde{b}\|_2 = \|\mathrm{vec}(BB^T)\|_2.$$

However, since we are using a block Arnoldi algorithm, we already have a QR factorization of B, $BP = QR$, where R is a p by p upper triangular matrix, from which it follows,

$$\beta = \|BB^T\|_F = \|BP(BP)^T\|_F = \|QRR^TQ^T\|_F = \|RR^T\|_F,$$

which is by far the most economical way of computing β.

At the beginning of the j'th iteration the matrix Γ_j is given by equation (10.3). The matrix $V_j^T V_{j+1}$ is merely the first n_j rows of the n_{j+1} by n_{j+1} identity matrix. As a result we see that the initial value of Γ_j can be evaluated using the following three steps.

1: $\Gamma_j := \mathrm{zeros}(n_{j+1}, n_{j+1})$
2: $\Gamma_j(1 : n_{j+1}, 1 : n_j) := \bar{H}_j \Pi_j$
3: $\Gamma_j(1 : n_j, n_{j+1}) := \Gamma_j(1 : n_j, 1 : n_{j+1}) + \Gamma_j(1 : n_{j+1}, 1 : n_j)^T$

The linear updates of Γ_j are performed using a similar relationship between V_i and V_{j+1}. The inner products $\tilde{h}_{ij}$ are given by

$$\tilde{h}_{ij} = \tilde{v}_i^T \tilde{w}_j = \mathrm{trace}\left(V_i \Pi_i V_i^T)^T (V_{j+1} \Gamma_j V_{j+1}^T\right) = \mathrm{trace}\left(\Pi_i^T (V_i^T V_{j+1} \Gamma_j V_{j+1}^T V_i)\right).$$

However, the matrix $V_i^T V_{j+1} \Gamma_j V_{j+1}^T V_i$ is just the upper left n_i by n_i corner of the matrix Γ_j, i.e.

$$V_i^T V_{j+1} \Gamma_j V_{j+1}^T V_i = \Gamma_j(1 : n_i, 1 : n_i),$$

which implies that $\tilde{h}_{ij}$ can be evaluated as

$$\tilde{h}_{ij} = \mathrm{trace}\left(\Pi_i^T \Gamma_j(1 : n_i, 1 : n_i)\right).$$

Note that

$$\|\tilde{w}_j\|_2^2 = \mathrm{trace}\left(\left(V_{j+1} \Gamma_j V_{j+1}^T\right)^T V_{j+1} \Gamma_j V_{j+1}^T\right) = \mathrm{trace}\left(\Gamma_j^T \Gamma_j\right) = \|\Gamma_j\|_F^2.$$

In short, if the result of the QR factorization of B, as well as the matrices $\bar{H}_j$ are available, then a compact representation of the $\tilde{v}_j$ can be computed using small and dense matrices exclusively. In particular there is no need to involve the tall and dense matrices V_j.

We can now state Algorithm 20, which uses the compact representation of Theorem 10.3.1 to compute an orthonormal basis for $K_k(\tilde{A}, \tilde{b})$.

1: Set $\beta := \|R_{11}R_{11}^T + R_{12}R_{12}^T\|_F$, and $\Pi_1 := (R_{11}R_{11}^T + R_{12}R_{12}^T)/\beta$.
2: **for** $j = 1, 2 \ldots, k$ **do**
3: $\quad \Gamma_j := \text{zeros}(n_{j+1}, n_{j+1})$
4: $\quad \Gamma_j(1 : n_{j+1}, 1 : n_j) := \bar{H}_j \Pi_j$
5: $\quad \Gamma_j(1 : n_j, n_{j+1}) := \Gamma_j(1 : n_j, 1 : n_{j+1}) + \Gamma_j(1 : n_{j+1}, 1 : n_j)^T$
6: $\quad$ **for** $i = 1, \ldots, j$ **do**
7: $\quad\quad \tilde{h}_{ij} := \text{trace}(\Pi_i^T \Gamma_j(1 : n_i, 1 : n_i))$
8: $\quad\quad \Gamma_j(1 : n_i, 1 : n_i) := \Gamma_j(1 : n_i, 1 : n_i) - \Pi_i \tilde{h}_{ij}$
9: $\quad$ **end for**
10: $\quad \tilde{h}_{j+1,j} := \|\Gamma_j\|_F$
11: $\quad$ **if** $\tilde{h}_{j+1,j} = 0$ **then**
12: $\quad\quad k := j$
13: $\quad\quad$ **exit**
14: $\quad$ **else**
15: $\quad\quad \Pi_{j+1} := \Gamma_j / \tilde{h}_{j+1,j}$
16: $\quad$ **end if**
17: **end for**

Algorithm 20: Arnoldi algorithm for the pair $(\tilde{A}, \tilde{b})$ using the compact representation

Algorithm 20 computes a compact representation of a factorization of the form

$$\tilde{A}\tilde{V}_k = \tilde{V}_k \tilde{H}_k + \tilde{h}_{k+1,k} \tilde{v}_{k+1} e_k^T,$$

where $\tilde{H}_k = \left[\tilde{h}_{ij}\right]$ is a k by k upper Hessenberg matrix and the columns $\tilde{v}_j$ of $\tilde{V}_k$ are orthonormal and span $K_k(\tilde{A}, \tilde{b})$. The vector e_k is the last column of the k by k identity matrix. The vectors $\tilde{v}_j$ are given by

$$\tilde{v}_j = \mathrm{vec}(V_j \Pi_j V_j^T),$$

where the matrices V_j are produced by the block Arnoldi algorithm, Algorithm 11.

The most expensive step in this algorithm is not the modified Gram-Schmidt loop, but the initial construction of each Γ_j. This is due to the fact that all the information we need about the large sparse matrix A has been condensed into the small and dense matrices $\bar{H}_j$. In the worst case there is no rank degradation and the flop count is $O(p^3k^4)$. If Γ_j is overwritten by the Π_{j+1}, then we only need to store the Π_j and the matrix $\tilde{H} = \left[\tilde{h}_{ij}\right]$. In any case the storage requirement is $O(p^2k^3)$. We need to run the block Arnoldi algorithm, Algorithm 11, either before or concurrently with Algorithm 20. This carries an additional cost of $O(np^2k^2)$ flops and $O(npk)$ words of storage. The total cost of computing a compact representation of $K_k(\tilde{A}, \tilde{b})$ is $O(np^2k^2 + p^3k^4)$ flops, and $O(npk + p^2k^3)$ words of storage.

In practice Algorithm 20 loses orthogonality, which can be cured partially by re-orthogonalization. When deriving algorithm 20 we used the fact that the columns of V_m are mutually orthonormal. In practice we lose orthogonality, which is why it is critical to apply re-orthogonalization during the block Arnoldi algorithm. Failure to do this limits the accuracy of any calculation, which is based on Algorithm 20. Re-orthogonalization increases the cost, but not the storage requirement.

10.4 GMRES

Given Algorithm 20 it is easy to apply GMRES for linear systems directly to the Kronecker product form of the Lyapunov matrix equation (10.1), as shown in Algorithm 21.

Algorithm 21 computes a symmetric n_k by n_k matrix Y_k such that $X_k = V_k Y_k V_k^T$ is an approximate solution of (10.1). The algorithm adds $O(p^2k^3 + k^2)$ to the flop count

1: Solve the $k+1$ by k LS problem $\tilde{H}_k y + \beta e_1 = 0$ for $y = (y_1, y_2, \ldots y_k)^T$.

2: $Y_k := \text{zeros}(n_k, n_k)$

3: **for** $i = 1, \ldots k$ **do**

4: $\quad Y_k(1 : n_i, 1 : n_i) := Y_k(1 : n_i, 1 : n_i) + \Pi_i y_i$

5: **end for**

Algorithm 21: GMRES for linear systems adapted to $\tilde{A}\text{vec}(X) + \tilde{b} = 0$

and $O(p^2k^2 + k)$ to the storage requirement. These costs are insignificant compared to the demands of Algorithms 11, and 20. If A is a sparse matrix, then the cost of applying $k \leq m$ steps of GMRES to the Kronecker product form of the Lyapunov matrix equation (10.1) using the compressed representation is $O(np^2k^2 + p^3k^4)$ flops and $O(npk + p^2k^3)$ words of storage.

We did not change the GMRES algorithm when we showed how to adapt it to a single Lyapunov equation in Kronecker form. We showed that the calculations could be carried out using $O(n)$ memory and flops. As a result, we inherit the monotone convergence from the classical theory. We note that if A is negative definite, then $\tilde{A} = I \otimes A + A \otimes I$ is negative definite.

In principle our adaptation of the GMRES algorithm may require $O(n^2)$ iterations before converging, however the block Arnoldi algorithm eventually finds a matrix V_m with orthonormal columns spanning the A-invariant subspace $K(A, B) = K_m(A, B)$ after at most n iterations. Then the exact solution X of (10.1) may be recovered by solving the 'reduced' order Lyapunov equation

$$(V_m^T A V_m)Y + Y(V_m^T A^T V_m) + V_m^T B B^T V_m = 0,$$

for Y, from which X can be computed as $X = V_m Y V_m^T$. In practice this distinction is irrelevant, as we can only afford to do $k \ll n$ iterations.

10.5 CG

In this subsection we show that the CG algorithm for linear systems can be adapted to the Lyapunov matrix equation

$$AX + XA^T = BB^T,$$

written in Kronecker product form

$$\tilde{A}\text{vec}(X) = \tilde{b},$$

where

$$\tilde{A} = (I \otimes A + A \otimes I),$$

and

$$\tilde{b} = \mathrm{vec}(BB^T).$$

It is clear that if A is symmetric positive definite, then $\tilde{A}$ is symmetric positive definite. However, while it is possible to apply the classical algorithm directly to the equivalent linear system it is not practical to do so, because of the time and memory requirements which are both $O(n^2)$. We state the classical algorithm as Algorithm 22 for the sole purpose of proving Theorem 10.5.1. The tilde superscript is used to emphasize that we are computing in $\mathbb{R}^{n^2}$ rather than $\mathbb{R}^n$.

1: $\tilde{r}_1 := \mathrm{vec}(BB^T)$, $\tilde{p}_1 := \tilde{r}_1$, $\tilde{x}_1 = 0$.
2: **for** $j = 1, 2 \ldots, k$ **do**
3: $\quad \alpha_j := (\tilde{r}_j, \tilde{r}_j)/(\tilde{A}\tilde{p}_j, \tilde{p}_j)$
4: $\quad \tilde{x}_{j+1} := \tilde{x}_j + \alpha_j \tilde{p}_j$
5: $\quad \tilde{r}_{j+1} := \tilde{r}_j - \alpha_j \tilde{A}\tilde{p}_j$
6: $\quad \beta_j := (\tilde{r}_{j+1}, \tilde{r}_{j+1})/(\tilde{r}_j, \tilde{r}_j)$
7: $\quad \tilde{p}_{j+1} := \tilde{r}_{j+1} + \beta_j \tilde{p}_j$
8: **end for**

Algorithm 22: Classical CG for solving $\tilde{A}\mathrm{vec}(X) = \tilde{b}$

Theorem 10.5.1 *Let A, B, n_j, and V_j be as in section 10.2. Then there exists n_j by n_j symmetric matrices R_j, P_j and X_j, such that the vectors computed by Algorithm 22 are given by*

$$\tilde{r}_j = \mathrm{vec}(V_j R_j V_j^T), \quad \tilde{p}_j = \mathrm{vec}(V_j P_j V_j^T), \quad \textit{and} \quad \tilde{x}_j = \mathrm{vec}(V_j X_j V_j^T).$$

Proof The proof is by induction on j and follows Algorithm 22 step by step. The proof is constructive and leads directly to Algorithm 23. It is clear that the statement of the theorem is true for $j = 1$. From the QR factorization of B we have

$$BB^T = \hat{Q}_1(R_{11}R_{11}^T + R_{12}R_{12}^T)\hat{Q}_1^T,$$

and we can write $R_1 = P_1 = (R_{11}R_{11}^T + R_{12}R_{12}^T)$ and $X_1 = \text{zeros}(n_1, n_1)$. Now, assume that the statement of the theorem is true for $j \geq 1$. Then,

$$\alpha_j = (\tilde{r}_j, \tilde{r}_j)/(\tilde{A}\tilde{p}_j, \tilde{p}_j) = \|R_j\|_F/\text{trace}(\Gamma(P_j)^T V_{j+1}^T V_j P_j V_j^T V_{j+1}),$$

where we have defined

$$\Gamma(P_j) = \bar{H}_j P_j V_j^T V_{j+1} + V_{j+1}^T V_j P_j \bar{H}_j^T.$$

Now,

$$\tilde{x}_{j+1} = \tilde{x}_j + \alpha_j \tilde{p}_j = \text{vec}(V_j X_j V_j^T) + \alpha_j \text{vec}(V_j P_j V_j^T) = \text{vec}(V_j(X_j + \alpha_j P_j)V_j^T),$$

and we have $X_{j+1} = V_{j+1}^T V_j (X_j + \alpha_j P_j) V_j^T V_{j+1}$. Similarly,

$$\begin{aligned}\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j \tilde{A}\tilde{p}_j = \text{vec}(V_j R_j V_j^T) - \alpha_j \,\text{vec}(AV_j P_j V_j^T + V_j P_j V_j^T A^T) \\ = \text{vec}(V_{j+1}(V_{j+1}^T V_j R_j V_j^T V_{j+1} - \alpha_j \Gamma(P_j) V_{j+1}^T),\end{aligned}$$

and we have $R_{j+1} = V_{j+1}^T V_j R_j V_j^T V_{j+1} - \alpha_j \Gamma(P_j)$. Finally, we have

$$\beta_j = (\tilde{r}_{j+1}, \tilde{r}_{j+1})/(\tilde{r}_j, \tilde{r}_j) = \|R_{j+1}\|_F / \|R_j\|_F,$$

and

$$\begin{aligned}\tilde{p}_{j+1} := \tilde{r}_{j+1} + \beta_j \tilde{p}_j = \text{vec}(V_{j+1} R_{j+1} V_{j+1}^T) + \beta_j \text{vec}(V_j P_j V_j^T) \\ \text{vec}(V_{j+1}(R_{j+1} + \beta_j V_{j+1}^T V_j P_j V_j^T V_{j+1}) V_{j+1}^T),\end{aligned}$$

which yields

$$P_{j+1} = R_{j+1} + \beta_j V_{j+1}^T V_j P_j V_j^T V_{j+1}.$$

We note, that if P_j is symmetric, then $\Gamma(P_j)$ is symmetric and the symmetry of R_{j+1}, P_{j+1}, and X_{j+1} follows from the symmetry of R_j, P_j, and X_j. We have shown that the statement of the theorem is true for $j + 1$, which completes the proof. ■

From the proof of Theorem 10.5.1 it is clear that once the matrices $\bar{H}_j$ have been computed by the block Arnoldi algorithm, then the rest of the computations can be

done with small and dense matrices. We formalize this statement in Algorithm 23. As in Algorithm 21 we need not compute the product $V_j^T V_{j+1}$ explicitly. Algorithm 23 requires $O(p^3k^4)$ arithmetic operations and $O(p^2k^3)$ words of storage, provided that the storage is recycled.

1: $R_1 = P_1 = V_1^T BB^T V_1$, $X_1 =$ the n_1 by n_1 zero matrix.
2: **for** $j = 1, 2, \ldots, k$ **do**
3: $\quad \Gamma(P_j) = \bar{H}_j P_j V_j^T V_{j+1} + V_{j+1}^T V_j P_j \bar{H}_j^T$
4: $\quad \alpha_j = \|R_j\|_F^2 / \mathrm{trace}(\Gamma(P_j)^T V_{j+1}^T V_j P_j V_j^T V_{j+1})$
5: $\quad X_{j+1} = V_{j+1}^T V_j (X_j + \alpha_j P_j) V_j^T V_{j+1}$
6: $\quad R_{j+1} = V_{j+1}^T V_j R_j V_j^T V_{j+1} - \alpha_j \Gamma(P_j)$
7: $\quad \beta_j := \|R_{j+1}\|_F^2 / \|R_j\|_F^2$
8: $\quad P_{j+1} = R_{j+1} + \beta_j V_{j+1}^T V_j P_j V_j^T V_{j+1}$
9: **end for**

Algorithm 23: CG adapted to $AX + XA^T = BB^T$

Because of the equivalence between the classical algorithm, Algorithm 22, and the adapted algorithm, Algorithm 23, we inherit the convergence results from the classical CG theory. In particular, we have the following theorem.

Theorem 10.5.2 *Let A be a symmetric positive definite matrix, $A = A^T > 0$. Then Algorithm 23 converges to the solution X of $AX + XA^T = BB^T$ and the matrices X_k satisfy*

$$\|X - V_k X_k V_k^T\|_F \leq 2\kappa(A) \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|X\|_F.$$

Proof From the classical theory [51] we have the estimate

$$\|\mathrm{vec}(X) - \mathrm{vec}(V_k X_k V_k^T)\|_{\tilde{A}} \leq 2 \left(\frac{\sqrt{\kappa(\tilde{A})} - 1}{\sqrt{\kappa(\tilde{A})} + 1} \right)^k \|\mathrm{vec}(X)\|_{\tilde{A}}, \tag{10.5}$$

where

$$\|\mathrm{vec}(X)\|_{\tilde{A}} = \mathrm{vec}(X)^T \tilde{A} \mathrm{vec}(X),$$

is the norm generated by the SPD matrix $\tilde{A}$. If $\lambda_{\min}(\tilde{A})$ is the smallest and $\lambda_{\max}(\tilde{A})$ is the largest eigenvalue for $\tilde{A}$, then

$$\lambda_{\min}(\tilde{A})\|\mathrm{vec}(X)\|_2 \leq \|\mathrm{vec}(X)\|_{\tilde{A}} \leq \lambda_{\max}(\tilde{A})\|\mathrm{vec}(X)\|_2. \tag{10.6}$$

In general the eigenvalues of $\tilde{A}$ are of the form $\lambda_i + \lambda_j$, where λ_i, and λ_j are eigenvalues of A. In particular, we have

$$\lambda_{\min}(\tilde{A}) = 2\lambda_{\min}(A), \quad \lambda_{\max}(\tilde{A}) = 2\lambda_{\max}(A) \quad \text{and} \quad \kappa(\tilde{A}) = \kappa(A). \tag{10.7}$$

The proof is completed by applying the information contained in (10.6) and (10.7) to the classical inequality (10.5). ■

We would like to draw attention to the fact that this result is very similar to one obtained by Simoncini and Druskin [56], Theorem 6.3.2. The two bounds are not immediately comparable, because Simoncini and Druskin bound the 2-norm of the absolute error for Saad's method [50], while we bound the Frobenius norm of the relative error of the CG applied to the Kronecker product form of a Lyapunov equation with low rank right hand side. However, we emphasize that their bound decays more rapidly than ours, and that the key quantity is $\kappa(A + \lambda_{\min}(A)I)$, rather than $\kappa(A)$.

Theorem 10.5.2 can be used to determine an upper bound on the number $k(\epsilon)$ of iterations needed in order to achieve a specific forward relative error ϵ. This allows us to estimate the runtime before starting the algorithm.

10.6 An experiment

At this junction we would like to insert an experiment which compares the Saad's Arnoldi method and the GMRES method by Jaimoukha and Kasenally, with our adaptations of the CG and GMRES algorithms for Lyapunov equations in the Kro-

necker product form. Let $n = 500$ and let A be the symmetric tridiagonal Toeplitz matrix given by

$$A = \begin{bmatrix} -3 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -3 \end{bmatrix}$$

and let

$$B = (1, 1, \ldots, 1)^T \in \mathbb{R}^n.$$

By Gershgorin's theorem A is negative definite, and each of the four algorithms applies to at least one of the two equivalent Lyapunov equations, namely

$$AX + XA^T + BB^T = 0,$$

or

$$(-A)X + X(-A)^T = BB^T.$$

The residual history for each of these four methods is recorded in Figure 10.1.

The GMRES method developed by Jaimoukha and Kasenally (JK GMRES) does not have a monotone residual history! As mentioned in Chapter 6 the algorithm hinges on the solution of a special linear system. Experimentally, the condition numbers for these linear systems grow monotonically with the iteration number, and in our case the condition number was greater than 10^{15}, after 21 iterations, and subsequent calculations became meaningless. This coincides with the point at which the residual norm starts to increase.

There is no data for iterations 1, 2, and 3 for JK GMRES: Each iteration is independent of the other iterations, but the first three iterations are special cases. It is not a trivial matter to implement JK GMRES, even in MATLAB, which is why we did not implement the first three iterations.

The basic Arnoldi method does not suffer from stability problems and compares well with JK GMRES, which is theoretically optimal for a given number steps with of the block Arnoldi algorithm.

Now, our CG algorithm and our GMRES algorithm are virtually indistinguishable from each other, until they reach stagnation, which, surprisingly, is at a lower level for our CG algorithm.

However, this is less significant compared with the following observation. Saad's Arnoldi method is faster than our CG algorithm. Asymptotically, both algorithms require the same number of arithmetic operations in order to carry out k iterations, specifically $O(np^2k^2 + p^3k^4)$. In both cases the underlying block Arnoldi algorithm requires $O(np^2k^2)$ flops, while the rest of the calculations are independent of n. Saad's Arnoldi method appear to be nearly optimal with respect to the number of Arnoldi steps, which is why it is faster than our method, especially for large values of n.

In general, the successful numerical scheme mimics the original problem as much as possible. Saad's Arnoldi method exploits the structure of the Lyapunov equations, specifically Theorem 2.6.3. We are only using the structure of the Lyapunov operator to reduce the storage requirement from $O(n^2)$ to $O(n)$. We are running the CG algorithm on a very large linear system, but our choice of method does not exploit the structure of the solution. This, we believe, is why we lose to the Arnoldi algorithm.

10.7 BCG

Consider a pair of Lyapunov equations

$$AX + XA^T = BB^T,$$
$$A^TY + YA = C^TC,$$

or the equivalent set of linear equations

$$\tilde{A}\text{vec}(X) = \tilde{b},$$
$$\tilde{A}^T\text{vec}(Y) = \tilde{c},$$

where $\tilde{A} = I \otimes A + A \otimes I$, $\tilde{b} = \text{vec}(BB^T)$, and $\tilde{c} = \text{vec}(C^TC)$. The importance of this problem has already been explained in Chapter 3.

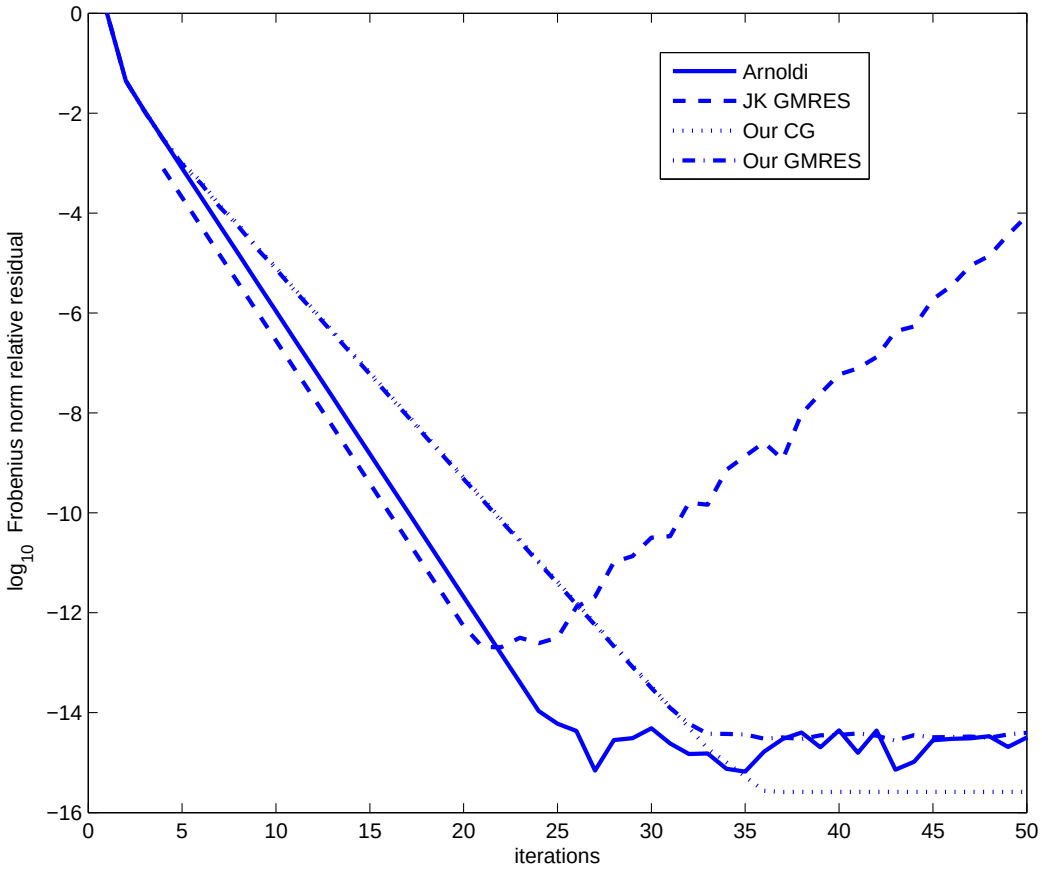

Figure 10.1. Residual history for different Krylov methods applied to equivalent Lyapunov equations

The BCG algorithm, which is stated as Algorithm 24, can be used to solve such as dual system. We continue to use the tilde superscript to emphasize that we are dealing with vectors in $\mathbb{R}^{n^2}$.

1: $\tilde{r}_1 := \tilde{b}$, $\tilde{r}_1^* := \tilde{c}$, $\tilde{p}_1 := \tilde{r}_1$, $\tilde{p}_1^* := \tilde{r}_1^*$
2: **for** $j = 1, \ldots,$ **do**
3: $\quad \alpha_j := (\tilde{r}_j, \tilde{r}_j^*)/(\tilde{A}\tilde{p}_j, \tilde{p}_j^*)$
4: $\quad \tilde{x}_{j+1} = \tilde{x}_j + \alpha_j \tilde{p}_j$
5: $\quad \tilde{x}_{j+1}^* = \tilde{x}_{j+1}^* + \alpha_j \tilde{p}_j^*$
6: $\quad \tilde{r}_j = \tilde{r}_j - \alpha_j \tilde{A}\tilde{p}_j$
7: $\quad \tilde{r}_{j+1}^* = \tilde{r}_j^* - \alpha_j \tilde{A}^T \tilde{p}_j^*$
8: $\quad \tilde{\beta}_j = (\tilde{r}_{j+1}, \tilde{r}_{j+1}^*)/(\tilde{r}_j, \tilde{r}_j^*)$
9: $\quad \tilde{p}_{j+1} = \tilde{r}_{j+1} + \tilde{\beta}_j \tilde{p}_j$
10: $\quad \tilde{p}_{j+1}^* = \tilde{r}_{j+1}^* + \tilde{\beta}_j \tilde{p}_j^*$
11: **end for**

Algorithm 24: BCG for a pair of linear equations

It is clear that the runtime and the storage requirements for Algorithm 24 are both $O(n^2)$, which limits this approach to small values of n.

For the sake of simplicity we consider only the case of $B = b \in \mathbb{R}^n$, and $C = c^T \in \mathbb{R}^n$, i.e. the pair of Lyapunov equations,

$$AX + XA^T = bb^T,$$
$$A^T Y + YA = c^T c.$$

Paul van Dooren suggested to us that we might use the Lanczos biorthogonalization procedure, Algorithm 25, to reduce the storage and runtime requirements to $O(n)$. The statement of Algorithm 25 is taken from Saad [51], with minor modifications in order to specify how the vectors v_1 and w_1 are constructed from b and c^T. We assume the inner product (b, c^T) is positive. If $(b, c^T) < 0$, then we simply replace b

with $-b$, which does not change the corresponding Lyapunov equation. If $(b, c^T) = 0$, then this entire approach cannot be applied.

1: $\gamma := (b, c^T) > 0$, $v_1 = b\gamma^{-1/2}$, $v_2 = c^T\gamma^{-1/2}$
2: Set $\beta_1 = \delta_1 = 0, w_0 = v_0 = 0$
3: **for** $j = 1, 2, \ldots, m$ **do**
4: $\quad \alpha_j = (Av_j, w_j)$
5: $\quad \hat{v}_{j+1} = Av_j - \alpha v_j - \beta_j v_{j-1}$
6: $\quad \hat{w}_{j+1} = A^T w_j - \alpha w_j - \delta_j w_{j-1}$
7: $\quad \delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1}|^{1/2}$. If $\delta_{j+1} = 0$ Stop.
8: $\quad \beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$
9: $\quad w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
10: $\quad v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
11: **end for**

Algorithm 25: Lanczos biorthogonalization procedure

Now, let T_m denote the matrix

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{bmatrix},$$

where the entries are defined by Algorithm 25.

It is well known, see Saad [51], that if the Lanczos algorithm does not break down before step m, then

$$V_m^T W_m = I_m, \quad j = 1, 2, \ldots, m,$$

and

$$K_m(A, b) = \operatorname{Ran} V_m, \quad K_m(A^T, c^T) = \operatorname{Ran} W_m,$$

and the following relations hold

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T, \tag{10.8}$$

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T, \tag{10.9}$$

as well as $W_m^T A V_m = T_m$.

We now define a sequence of matrices as follows

$$\bar{H}_j(1:j, 1:j) = T_j, \qquad \bar{H}_j(j+1, :) = \delta_{j+1} e_j^T,$$
$$\bar{K}_j(1:j, 1:j) = T_j^T, \qquad \bar{K}_j(j+1, :) = \beta_{j+1} e_j^T,$$

and

$$V_j = \begin{bmatrix} v_1 & v_2 & \dots & v_j \end{bmatrix} \quad \text{and} \quad W_j = \begin{bmatrix} w_1 & w_2 & \dots & w_j \end{bmatrix}$$

for $j = 1, 2, \dots, m-1$. We emphasize that the matrices $\bar{H}_j$, $\bar{K}_j$ are not square, but are $j+1$ by j matrices. These definitions allow us to rewrite equations (10.8), (10.9) as follows,

$$AV_j = V_{j+1} \bar{H}_j,$$
$$A^T W_j = W_{j+1} \bar{K}_j,$$

for $j = 1, 2, \dots m-1$. This completes our preparation for the following theorem.

Theorem 10.7.1 *If the unsymmetric Lanczos procedure does not break down before step m, then there exists symmetric j by j matrices X_j, R_j, P_j, X_j^*, R_j^*, and P_j^*, such that the vectors $\tilde{x}_j$, $\tilde{r}_j$, $\tilde{p}_j$, $\tilde{x}_j^*$, $\tilde{r}_j^*$, and $\tilde{p}_j^*$ produced by applying the BCG algorithm to the dual equations, satisfy*

$$\tilde{r}_j = \mathrm{vec}(V_j R_j V_j^T), \quad \tilde{p}_j = \mathrm{vec}(V_j P_j V_j^T), \quad \tilde{x}_j = \mathrm{vec}(V_j X_j V_j^T),$$

and

$$\tilde{r}_j^* = \mathrm{vec}(W_j R_j^* W_j^T), \quad \tilde{p}_j^* = \mathrm{vec}(W_j P_j^* W_j^T), \quad \tilde{x}_j^* = \mathrm{vec}(W_j X_j^* W_j^T).$$

Proof The proof is very similar to our analysis of the Arnoldi algorithm, and follows Algorithm 25 step by step. Initially we have

$$\tilde{x}_1 = 0, \quad \tilde{r}_1 = \tilde{p}_1 = \tilde{b} = \mathrm{vec}(bb^T) = \mathrm{vec}(\gamma v_1 v_1^T),$$

from which it follows that $X_1 = 0, R_1 = P_1 = \gamma$ (1 by 1 matrices) is the proper choice for $j = 1$. Similarly, we find that $X_1^* = 0$, and $R_1^* = P_1^* = \gamma$. Now, assume that the statement of the theorem is true for some value $j < m$. Then we have

$$\tilde{A}\tilde{p}_j = \tilde{A}\mathrm{vec}(V_j P_j V_j^T) = \mathrm{vec}(AV_j P_j V_j^T + V_j P_j V_j^T A^T) = \\ \mathrm{vec}(V_{j+1}\bar{H}_j P_j V_j^T + V_j P_j \bar{H}_j^T V_{j+1}^T).$$

We claim that there exists a matrix Γ_j such that

$$\tilde{A}\tilde{p}_j = \mathrm{vec}(V_{j+1}\Gamma_j V_{j+1}^T).$$

To this end we note that, $V_j^T W_{j+1} V_{j+1}^T = V_j^T$, from which is follows that

$$\Gamma_j = \bar{H}_j P_j V_j^T W_{j+1} + W_{j+1}^T V_j P_j \bar{H}_j^T$$

is the proper choice. Similarly, we see that

$$\Gamma_j^* = \bar{K}_j P_j^* W_j^T V_{j+1} + V_{j+1}^T W_j P_j^* \bar{K}_j^T,$$

satisfies

$$\tilde{A}^T \tilde{p}_j^* = \mathrm{vec}(W_{j+1}\Gamma_j^* W_{j+1}^T).$$

We now continue to the definition of α_j.

$$\alpha_j := (\tilde{r}_j, \tilde{r}_j^*)/(\tilde{A}\tilde{p}_j, \tilde{p}_j^*) = \mathrm{trace}(R_j^T R_j^*)/\mathrm{trace}(\Gamma_j^T(1:j, 1:j)P_j^*).$$

Then we have

$$\tilde{x}_{j+1} := \tilde{x}_j + \alpha_j \tilde{p}_j = \mathrm{vec}(V_j\{X_j + \alpha_j P_j\}V_j^T) \\ = \mathrm{vec}(V_{j+1}\{W_{j+1}^T V_j (X_j + \alpha_j P_j) V_j^T W_{j+1}\}V_{j+1}^T),$$

and

$$\tilde{r}_{j+1} := \tilde{r}_j - \alpha_j \tilde{A}\tilde{p}_j = \mathrm{vec}(V_j R_j V_j^T) - \alpha_j \mathrm{vec}(V_{j+1}\Gamma_j V_{j+1}^T)$$
$$= \mathrm{vec}(V_{j+1}\{W_{j+1}^T V_j R_j V_j^T W_{j+1} - \alpha_j \Gamma_j\} V_{j+1}^T),$$

from which it follows that

$$X_{j+1} = W_{j+1}^T V_j (X_j + \alpha_j P_j) V_j^T W_{j+1},$$
$$R_{j+1} = W_{j+1}^T V_j R_j V_j^T W_{j+1} - \alpha_j \Gamma_j.$$

Finally, we note that

$$\tilde{\beta}_j := (\tilde{r}_{j+1}, \tilde{r}_{j+1}^*)/(\tilde{r}_j, \tilde{r}_j^*) = \mathrm{trace}(R_{j+1}^T R_{j+1}^*)/\mathrm{trace}(R_j^T R_j^*),$$

and

$$\tilde{p}_{j+1} = \tilde{r}_{j+1} + \tilde{\beta}_j \tilde{p}_j = \mathrm{vec}(V_{j+1} R_{j+1} V_{j+1}^T) + \tilde{\beta}_j \mathrm{vec}(V_j P_j V_j^T)$$
$$= \mathrm{vec}(V_{j+1}\{R_{j+1} + \tilde{\beta}_j W_{j+1}^T V_j P_j V_j^T W_{j+1}\} V_{j+1}^T),$$

which implies that

$$P_{j+1} = R_{j+1} + \tilde{\beta}_j W_{j+1}^T V_j P_j V_j^T W_{j+1}.$$

It is clear that we can obtain expressions for X_{j+1}^*, R_{j+1}^*, and P_{j+1}^* in a similar fashion. Specifically, we find

$$X_{j+1}^* = V_{j+1}^T W_j (X_j^* + \alpha_j P_j^*) W_j^T V_{j+1},$$
$$R_{j+1}^* = V_{j+1}^T W_j R_j^* W_j^T V_{j+1} - \alpha_j \Gamma_j^*,$$
$$P_{j+1}^* = R_{j+1}^* + \tilde{\beta}_j V_{j+1}^T W_j P_j^* W_j^T V_{j+1}.$$

The symmetry of the matrices is established by induction, the key being the assumed symmetry of P_j (P_j^*), which leads to the symmetry of Γ_j (Γ_j^*). ■

It is now straight forward to adapt BCG the Kronecker product formulation. This is done in Algorithm 26. By construction, the matrices $W_{j+1}^T V_j$ and $V_{j+1}^T W_j$ are

1: $X_1 := 0$, $R_1 := \gamma$, $P_1 = \gamma$, $X_1 := 0$, $R_1 := \gamma$, $P_1 = \gamma$.

2: **for** $j = 1, 2, \ldots, m-1$ **do**

3: $\quad \Gamma_j := \text{zeros}(j+1, j+1)$, $\Gamma_j(1:j+1, 1:j) := \bar{H}_j P_j$, $\Gamma_j := \Gamma_j + \Gamma_j^T$

4: $\quad \Gamma_j^* := \text{zeros}(j+1, j+1)$, $\Gamma_j^*(1:j+1, 1:j) := \bar{K}_j P_j^*$, $\Gamma_j^* := \Gamma_j^* + \Gamma_j^{*T}$

5: $\quad \alpha_j := \text{trace}(R_j^T R_j^*)/\text{trace}(\Gamma_j(1:j, 1:j)^T P_j^*)$

6: $\quad X_{j+1} := \text{zeros}(j+1, j+1)$; $X_{j+1}(1:j, 1:j) = X_j + \alpha_j P_j$

7: $\quad X_{j+1}^* := \text{zeros}(j+1, j+1)$; $X_{j+1}^*(1:j, 1:j) = X_j^* + \alpha_j P_j^*$

8: $\quad R_{j+1} := -\alpha_j \Gamma_j$; $R_{j+1}(1:j, 1:j) := R_{j+1}(1:j, 1:j) + R_j$

9: $\quad R_{j+1}^* := -\alpha_j \Gamma_j^*$; $R_{j+1}^*(1:j, 1:j) := R_{j+1}^*(1:j, 1:j) + R_j^*$

10: $\quad P_{j+1} := R_{j+1}$; $P_{j+1}(1:j, 1:j) := P_{j+1}(1:j, 1:j) - \tilde{\beta}_j P_j$

11: $\quad P_{j+1}^* := R_{j+1}^*$; $P_{j+1}(1:j, 1:j)^* := P_{j+1}(1:j, 1:j)^* - \tilde{\beta}_j P_j^*$

12: **end for**

Algorithm 26: BCG for a pair of Lyapunov equations using the compact representation

nothing but the first j columns of the $j+1$ by $j+1$ identity matrix. This observation has been used to simplify the calculations.

First, we note that this is not our first choice of a solver. We inherit the erratic convergence of BCG, and the potential of breakdown from the Lanczos algorithm. In addition it is critical that the Lanczos vectors remain biorthogonal, i.e. $V_m^T W_m = I_m$, which forces us to reorthogonalize at every iteration of Algorithm 25. Our goal has been to emphasize that it is not the sheer size of the matrix $\tilde{A}$, which will prevent one from applying BCG to a pair of Lyapunov equations in the Kronecker product form.

It follows that we must find a way to limit the number of iterations, which makes it imperative that we discuss the problem of how to precondition Lyapunov equations. This is the topic of Chapter 11.

We finish this chapter by demonstrating how to apply CGNR to a Lyapunov equation, regardless of whether A is negative definite or even stable.

10.8 CGNR

Given a linear system $Ax = f$ and an initial guess x_0, the CGNR algorithm computes a sequences of approximations $\{x_j\}$ of the exact solution x_* such that x_j minimizes the 2-norm of the residual $r(x) = f - Ax$ over all vectors x in the affine Krylov subspace

$$x_0 + K_j(A^T A, A^T r_0) = x_0 + \text{span}\{A^T r_0, A^T A A^T r_0, \ldots, (A^T A)^{j-1} A^T r_0\},$$

where $r_0 = f - Ax_0$ and $j = 1, 2, \ldots$. If A is nonsingular, then the algorithm will converge to the solution x_* in at most n steps, if the calculations are carried out in exact arithmetic. The convergence of the algorithm is controlled by the condition number of $A^T A$ and/or the distribution of eigenvalues of $A^T A$. Specifically, if $A^T A$ is well conditioned or if the eigenvalues of $A^T A$ are tightly clustered, then x_j is a good approximation of x_*, even for modest values of j.

In this section we show how to adapt CGNR for linear systems to a Lyapunov equation

$$AX + XA^T + BB^T = 0,$$

in Kronecker product form

$$\tilde{A}\tilde{x} + \tilde{b} = 0,$$

where $\tilde{A} = I \otimes A + A \otimes I$ and $\tilde{b} = \text{vec}(BB^T)$. The advantage of this approach is that CGNR will converge if the matrix $\tilde{A}$ is nonsingular. Now, $\tilde{A}$ is nonsingular if the Lyapunov equation

$$AY + YA^T + Q = 0,$$

has a unique solution Y for every matrix Q or equivalently if the eigenvalues $\{\lambda_j\}$ of A satisfy

$$\forall\, i, j \;:\; \lambda_i + \lambda_j \neq 0. \tag{10.10}$$

Now, condition (10.10) is much weaker than that of requiring A to be negative definite, giving us a clear advantage over standard projection methods which require A to be negative definite. The low rank cyclic Smith method can also be applied to general problems which satisfy condition (10.10), but it is necessary to analyze the spectrum for A in order to determine the shift parameters p_j, and in general we must also compute good preconditioners for linear systems with coefficient matrices $(A + p_j I)$. No such analysis is necessary for CGNR.

As usual the main problem is to reduce the storage requirements and the number of arithmetic operations from $O(n^2)$ to $O(n)$. The compact representation of vectors in $\mathbb{R}^{n^2}$ which we developed in the previous chapters cannot be applied to CGNR. The problem is that while that representation is designed to respond well to multiplication by $\tilde{A}$, i.e.

$$\forall\, X_j\, \exists\, Y_{j+1} \;:\; \tilde{A}\text{vec}(V_j X_j V_j^T) = \text{vec}(V_{j+1} Y_{j+1} V_{j+1}^T),$$

we do not consider multiplication by $\tilde{A}^T$ and there is no guarantee that there exists a Z_{j+2}, such that

$$\tilde{A}^T\text{vec}(V_{j+1} Y_{j+1} V_{j+1}^T) = \text{vec}(V_{j+2} Z_{j+2} V_{j+2}^T).$$

The key to overcoming this problem is to replace the standard factorization

$$AV_j = V_{j+1}\bar{H}_j, \quad j = 1, 2 \ldots, 2k,$$

with a different factorization, specifically

$$AV_{2j-1} = V_{2j}\bar{H}_{2j-1}, \quad A^T V_{2j} = V_{2j+1}\bar{K}_{2j},$$

for $j = 1, 2 \ldots, k$.

We begin by stating our main result, Theorem 10.3.1, which explains why such a factorization is exactly what we need, then we give a short proof of its existence, and explain how to compute it in practice.

The CGNR algorithm for standard linear systems of the type $Ax = f$ is given as Algorithm 27.

1: $\tilde{r}_0 := f - Ax_0$, $z_0 := A^T r_0$, $p_0 := z_0$.
2: **for** $j = 0, 1, 2 \ldots,$ **do**
3: $\quad w_j := Ap_j$.
4: $\quad \alpha_j = \|z_j\|_2^2 / \|w_j\|_2^2$.
5: $\quad x_{j+1} := x_j + \alpha_j p_j$.
6: $\quad r_{j+1} := r_j - \alpha_j w_j$.
7: $\quad z_{j+1} := A^T r_{j+1}$.
8: $\quad \beta_j := \|z_{j+1}\|_2^2 / \|z_j\|_2^2$.
9: $\quad p_{j+1} := z_j + \beta_j p_j$.
10: **end for**

Algorithm 27: CGNR for $Ax = f$

We are interested in applying the algorithm to the linear system $\tilde{A}\tilde{x} + \tilde{b} = 0$, or equivalently $(-\tilde{A})\tilde{x} = \tilde{b}$. We limit ourselves to the initial guess $x_0 = 0$, because it is essential that the initial residual $\tilde{r}_0$ has the same structure as the inhomogeneous term $\tilde{b}$. As a result, it suffices to change the sign in the computation of x_{j+1}, i.e.

$$x_{j+1} := x_j - \alpha_j p_j.$$

Finally, it is convenient to start the iteration at $j = 1$, rather than $j = 0$. We state these minor modifications as Algorithm 28. We continue using the tilde superscript to emphasize that we are dealing with vectors in $\mathbb{R}^{n^2}$.

1: $\tilde{r}_1 := \tilde{b}$, $\tilde{z}_1 := \tilde{A}^T\tilde{r}_1$, $\tilde{p}_1 = \tilde{z}_1$.
2: **for** $j = 1, 2 \ldots,$ **do**
3: $\quad \tilde{w}_j := \tilde{A}\tilde{p}_j$.
4: $\quad \alpha_j = \|\tilde{z}_j\|_2^2 / \|\tilde{w}_j\|_2^2$.
5: $\quad \tilde{x}_{j+1} := \tilde{x}_j - \alpha_j \tilde{p}_j$.
6: $\quad \tilde{r}_{j+1} := \tilde{r}_j - \alpha_j \tilde{w}_j$.
7: $\quad \tilde{z}_{j+1} := \tilde{A}^T\tilde{r}_{j+1}$.
8: $\quad \beta_j := \|\tilde{z}_{j+1}\|_2^2 / \|\tilde{z}_j\|_2^2$.
9: $\quad \tilde{z}_{j+1} := \tilde{z}_j + \beta_j \tilde{p}_j$.
10: **end for**

Algorithm 28: CGNR for $\tilde{A}\tilde{x} + \tilde{b} = 0$ with initial guess $\tilde{x}_1 = 0$

It is clear that the storage requirement is $O(n^2)$ and that the runtime is at least $O(n^2 k)$. Simply storing a single vector explicitly requires $O(n^2)$ storage and computing a norm requires $O(n^2)$ arithmetic operations. If A is a general sparse matrix then matrix multiplication with $\tilde{A}$ requires $O(n^2)$ arithmetics operations, but this figure grows to $O(n^3)$ if A is a general dense matrix.

However, the following theorem illustrates that the vectors computed by Algorithm 28 have a very special structure, which can be used to reduce both the storage and the runtime requirements significantly.

Theorem 10.8.1 *Let k be a positive integer and suppose we are given matrices*

$$Q_1, Q_2, \ldots, Q_{2k+3}, \quad \bar{H}_1, \bar{H}_3, \ldots \bar{H}_{2k+1}, \quad \bar{K}_2, \bar{K}_4, \ldots, \bar{K}_{2k+2},$$

such that

$$V_j = \begin{bmatrix} Q_1 & Q_2 \ldots Q_j \end{bmatrix}, \quad j = 1, 2, \ldots 2k+3,$$

has orthonormal columns and

$$AV_{2j-1} = V_{2j}\bar{H}_{2j-1},$$
$$A^T V_{2j} = V_{2j+1}\bar{K}_{2j},$$

for $j = 1, 2, \ldots k+1$. *If* $Ran\, B = Ran\, V_1$, *then there exist matrices*

$$X_j,\ R_j,\ P_j, \quad \textit{and} \quad Z_j,$$

such that the vectors $\tilde{x}_j$, $\tilde{r}_j$, $\tilde{p_j}$, *and* $\tilde{z}_j$ *produced by doing* k *iterations of Algorithm 28, can be written as*

$$\tilde{x}_j = \mathrm{vec}(V_{2j+1}X_jV_{2j+1}^T), \qquad \tilde{r}_j = \mathrm{vec}(V_{2j}R_jV_{2j}^T),$$
$$\tilde{p}_j = \mathrm{vec}(V_{2j+1}P_jV_{2j+1}^T), \qquad \tilde{z}_j = \mathrm{vec}(V_{2j+1}Z_jV_{2j+1}^T),$$

for $j = 1, 2, \ldots, k+1$.

Proof The proof is by induction on j. First, let n_j denote the number of columns of V_j. Since the V_{j+1} is formed by augmenting V_j, we have

$$1 \le n_1 \le n_2 \le \cdots \le n_{2k+3}.$$

We begin by examining the initial values, i.e. the vectors $\tilde{x}_1$, $\tilde{r}_1$, $\tilde{p_1}$, and $\tilde{z}_1$. The initial guess is $\tilde{x_1} = 0$, so

$$X_1 = \mathrm{zeros}(1 : n_3, 1 : n_3),$$

will suffice. The initial residual is $\tilde{r}_1 = \tilde{b} = \mathrm{vec}(BB^T)$. By assumption $\mathrm{Ran}\, V_1 = \mathrm{Ran}\, B$, so $B = V_1S$ for a suitable matrix S, from which it follows $\tilde{r}_1 = \tilde{b} = \mathrm{vec}(V_1SS^TV_1)$, which we write as $\tilde{r}_1 = \mathrm{vec}(V_2R_1V_2^T)$, where

$$R_1 = \begin{bmatrix} SS^T & 0 \\ 0 & 0 \end{bmatrix}.$$

This trivial embedding of SS^TT in a n_2 by n_2 zero matrix is convenient because $\tilde{z}_1$ is computed by applying $\tilde{A}^T$ to $\tilde{r}_1$. We have

$$\tilde{z}_1 = \tilde{A}^T\tilde{r}_1 = \mathrm{vec}(A^TV_2R_2V_2^T + V_2R_2V_2^TA^T) = \mathrm{vec}(V_3Z_1V_3^T),$$

where

$$Z_1 = \bar{K}_2 R_2 V_2^T V_3 + V_3^T V_2 R_2 \bar{K}_2^T.$$

Finally we note that $\tilde{p}_1 = \tilde{z}_1$, so we should choose $P_1 = Z_1$.

Now, suppose the statement of the theorem is true for an integer j, $1 \leq j \leq k$. The auxiliary variable $\tilde{w}_j$ satisfies

$$\tilde{w}_j = \tilde{A}\tilde{p}_j = \tilde{A}\text{vec}(V_{2j+1} P_j V_{2j+1}^T) = \text{vec}(V_{2j+2} \Delta_j V_{2j+2}^T),$$

where

$$\Delta_j = \bar{H}_{2j+1} P_j V_{2j+1}^T V_{2j+2} + V_{2j+2}^T V_{2j+1} P_j \bar{H}_{j+1}^T.$$

The real number $\alpha_j := \|\tilde{z}_j\|_2^2 / \|\tilde{w}_j\|_2^2$ can be calculated as

$$\alpha_j = \|Z_j\|_F^2 / \|\Delta_j\|_F^2.$$

The vector $\tilde{x}_{j+1} := \tilde{x}_j - \alpha_j \tilde{p}_j$ satisfies

$$\tilde{x}_{j+1} = \text{vec}(V_{2j+1}\{X_j - \alpha_j P_j\} V_{2j+1}^T) = \text{vec}(V_{2j+3} X_{j+1} V_{2j+3}^T),$$

where

$$X_{j+1} = \begin{bmatrix} X_j - \alpha_j P_j & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Similarly, $\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j \tilde{w}_j$ satisfies

$$\tilde{r}_{j+1} = \text{vec}(V_{2j} R_j V_{2j}^T - \alpha_j V_{2j+2} \Delta_j V_{2j+2}^T) = \text{vec}(V_{2j+2} R_{j+1} V_{2j+2}^T),$$

where

$$R_{j+1} = \begin{bmatrix} R_j & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \alpha_j \Delta_j.$$

The real number $\beta_j := \|\tilde{z}_{j+1}\|_2^2 / \|\tilde{z}_j\|_2^2$ can be calculated as

$$\beta_j = \|Z_{j+1}\|_F^2 / \|Z_j\|_F^2,$$

and the vector $\tilde{z}_{j+1} = \tilde{A}^T \tilde{r}_{j+1}$ satisfies

$$\tilde{z}_{j+1} = \tilde{A}^T \mathrm{vec}(V_{2j+2} R_{j+1} V_{2j+2}^T) = \mathrm{vec}(V_{2j+3} Z_{j+1} V_{2j+3}^T),$$

where

$$Z_{j+1} = \bar{K}_{2j+2} R_{j+1} V_{2j+2}^T V_{2j+3} + V_{2j+3}^T V_{2j+2} R_{j+1} \bar{K}_{2j+2}^T.$$

Finally, we note that $\tilde{p}_{j+1} := \tilde{z}_{j+1} + \beta_j \tilde{p}_j$ satisfies

$$\tilde{p}_{j+1} = \mathrm{vec}(V_{2j+3} Z_{j+1} V_{2j+3}^T) + \beta_j \mathrm{vec}(V_{2j+1} P_j V_{2j+1}^T) = \mathrm{vec}(V_{2j+3} P_{j+1} V_{2j+3}^T),$$

where

$$P_{j+1} = Z_{j+1} + \beta_j \begin{bmatrix} P_j & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

■

We will now prove the existence of a factorization of the form which is required for Theorem 10.8.1. The key step is the following lemma, which we shall refer to as the single step extension lemma.

Lemma 10.8.1 *Given an n by n matrix A and an n by m matrix V such that $V^T V = I_m$, then there exist matrices Q, and H such that*

$$AV = \begin{bmatrix} V & Q \end{bmatrix} H$$

and $\begin{bmatrix} V & Q \end{bmatrix}$ has orthonormal columns.

Proof We begin with the elementary decomposition

$$AV = VV^T AV + (I - VV^T)AV.$$

Now, let Q be any matrix with orthonormal columns such that

$$\mathrm{Ran}\, Q = \mathrm{Ran}\, (I - VV^T)AV.$$

Then

$$AV = VV^TAV + QQ^T(I - VV^T)AV = \begin{bmatrix} V & Q \end{bmatrix} \begin{bmatrix} V^TAV \\ Q^T(I - VV^T)AV \end{bmatrix},$$

from which it follows that given a suitable matrix Q, we should pick

$$H = \begin{bmatrix} V^TAV \\ Q^T(I - VV^T)AV \end{bmatrix}.$$

How do we choose Q? Let

$$(I - VV^T)AVP = \begin{bmatrix} \hat{Q}_1 & \hat{Q}_2 \end{bmatrix} \begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & 0 \end{bmatrix}$$

be a rank revealing QR factorization of $(I - VV^T)AV$, where P is a permutation matrix. The matrix $\hat{Q}_1$ has p_1 columns, where p_1 is the rank of $(I - VV^T)W$. The choice of

$$Q = \hat{Q}_1,$$

will suffice. ■

The existence of a factorization of the form given above now follows by successive applications of the single step extension lemma, where we alternate between multiplication by A and A^T. The process can be continued until two successive iterations have failed to append vectors to the current value of V. In this case we have discovered a subspace V, which is invariant with respect to A as well as A^T. If the process does not break down before we have performed $2k + 2$ extensions of the initial matrix V_1, then we have found matrices $Q_1, Q_2, \ldots Q_{2k+3}$, $\bar{H}_1, \bar{H}_3, \ldots \bar{H}_{2k+1}$, and $\bar{K}_2, \bar{K}_4, \ldots, \bar{K}_{2k+2}$ such that

$$V_j = \begin{bmatrix} Q_1 & Q_2 \ldots Q_j \end{bmatrix}, \quad j = 1, 2, \ldots 2k + 3,$$

has orthonormal columns and satisfies

$$AV_{2j-1} = V_{2j}\bar{H}_{2j-1},$$
$$A^TV_{2j} = V_{2j+1}\bar{K}_{2j},$$

for $j = 1, 2, \ldots k+1$. This is exactly the assumption made in Theorem 10.8.1. Exactly how much memory is consumed constructing a factorization of this type? We are constructing an orthonormal basis for the vector-spaces

$$\begin{aligned}
K_1 &= \mathrm{Ran} B, \\
K_2 &= \mathrm{Ran} \begin{bmatrix} B & AB \end{bmatrix}, \\
K_3 &= \mathrm{Ran} \begin{bmatrix} B & AB & A^T \begin{bmatrix} B & AB \end{bmatrix} \end{bmatrix}, \\
K_4 &= \mathrm{Ran} \begin{bmatrix} B & AB & A^T B & A^T AB & A \begin{bmatrix} AB & A^T \begin{bmatrix} B & AB \end{bmatrix} \end{bmatrix} \end{bmatrix},
\end{aligned}$$

at which point the formulas become too large to write out in full. Let V be a matrix with orthonormal columns, such that

$$K_j = \mathrm{Ran} V(:, 1 : n_j)$$

for suitable integers n_j. Then

$$K_{j+1} = K_j \cup \mathrm{Ran}\left(AV(:, n_{j-1} + 1 : n_j)\right), \quad j = 3, 5, 7, \ldots$$

and

$$K_{j+1} = K_j \cup \mathrm{Ran}\left(A^T V(:, n_{j-1} + 1 : n_j)\right), \quad j = 4, 6, 8, \ldots$$

It follows, that if there is no rank degradation at all, then

$$n_1 = r, \quad n_2 = 2r, \quad n_3 = 4r,$$

where r denotes the rank of B, and in general

$$n_{j+1} = 2n_j - n_{j-2}, \quad j = 3, 4, 5, \ldots.$$

The general solution is

$$n_j = c_1 q_1^j + c_2 q_2^j + c_3 q_3^j,$$

where q_1, q_2, q_3 are the three roots of

$$z^3 = 2z^2 - 1,$$

which are

$$\left\{1, \frac{1 \pm \sqrt{5}}{2}\right\}.$$

The particular solution, which satisfies the initial conditions is

$$\frac{n_j}{r} = 1 + \left(\frac{1}{2} - \frac{3}{10}\sqrt{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^j + \left(\frac{1}{2} + \frac{3}{10}\sqrt{5}\right)\left(\frac{1+\sqrt{5}}{2}\right)^j.$$

For large values of j the expression is dominated by the last term, and we see that

$$n_j = O(q^j),$$

where q is the golden ratio,

$$q = \frac{1+\sqrt{5}}{2} \approx 1.61803.$$

This places a severe restriction on the Lyapunov equations which can be solved by this method. The corresponding linear equations must be well conditioned, or we run out of memory long before the residual has been significantly reduced.

This is the worst case. In the very best case A is symmetric, but not necessarily stable. In this case memory consumption grows only linearly. Unfortunately, we do not know any real-life applications which generate non-stable symmetric problems.

We wrote a MATLAB implementation of the single step extension algorithm and the CGNR algorithm for Lyapunov equations in Kronecker product form. The worst case behavior is realized with randomly generated matrices. A nice problem, which is not negative definite or even stable, is given by the following 500 by 500 matrix $A = [a_{ij}]$, where

$$a_{ij} = \begin{cases} -3 & i = j < 500 \\ 1 & i = j+1, i < 500 \\ -1 & i = j-1, i < 500, i \neq 5 \\ 6 & i = j = 500 \\ 10 & i = 5, j = 6. \end{cases}$$

We picked $B = (1, 1, \ldots, 1)^T \in \mathbb{R}^{500}$, and solved the corresponding Lyapunov equation

$$AX + XA^T + BB^T = 0$$

using our CGNR algorithm. The residual history as well as the memory consumption are given in Figure 10.2. We emphasize that each CGNR iteration contains two multiplications, one with A and one with A^T, because we must calculate the action of $\tilde{A}$ and of $\tilde{A}^T$. This is not an example which is in our favor. After only 12 CGNR iterations, we are using 52 vectors to hold the basis, which is hardly a small number compared with the dimension $n = 500$. At the same time the relative residual is almost $4 \cdot 10^{-3}$. The point of this experiment is to show that the CGNR algorithm can be applied using less than $O(n^2)$ memory and that the worst case behavior need not occur.

10.9 Conclusion

Let A be an n by n stable matrix, and let B be a tall matrix such that AB is defined. The Lyapunov matrix equation

$$AX + XA^T + BB^T = 0$$

is equivalent to a standard linear system with n^2 unknowns. Previously, Hochbruck and Starke [23] applied Krylov subspace methods directly to this linear system using $O(n^2)$ memory and time. We have shown that it is possible to reduce these requirements to $O(n)$ for a variety of methods including CG, GMRES, BCG and CGNR. The key is to exploit the relationship between certain Krylov subspaces.

Unfortunately, our CG and GMRES methods are slower than the Arnoldi method. The fundamental problem is that we do not exploit the structure of the solution, but are running algorithms which are designed for general linear systems.

Our BCG algorithm is built on the Lanczos biorthogonalization procedure, and may break down at any time. This procedure is not recommended.

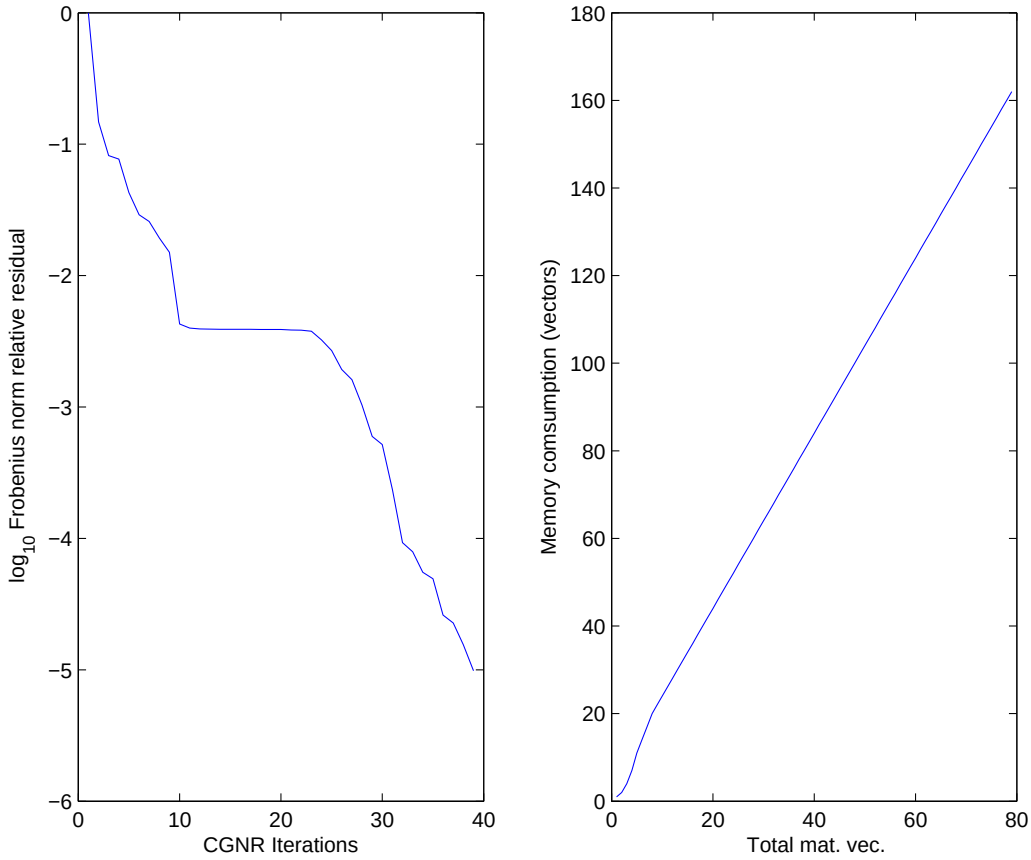

Figure 10.2. Residual history and memory consumption for CGNR applied to a non-stable Lyapunov equation.

In principle, our adaptation of the CGNR algorithm can solve any Lyapunov matrix equation, regardless of whether A is negative definite or even stable. Unfortunately, in the worst case memory consumption is $O(1.62^j n)$, where j is the number of iterations, and n is the dimension of the problem. This limits our approach to well conditioned problems.

11. Preconditioning

11.1 Introduction

Hochbruck and Starke [23] showed how to apply preconditioned Krylov subspace methods to Lyapunov matrix equations, however their method requires $O(n^2)$ memory and time, which limits their approach to all but the smallest values of n. In this chapter we explain how to reduce the storage and time requirements to $O(n)$.

The compact representations of vectors in $\mathbb{R}^{n^2}$ which we developed in the previous chapter does not permit preconditioning. The fundamental problem is that even if we are able to assemble and solve an equation of the type

$$\tilde{M}\text{vec}(Y) + \text{vec}(V_i X_i V_i^T),$$

there is no guarantee the the matrix Y can be written in the form

$$Y = \text{vec}(V_j Y_j V_j^T)$$

for any value of j. In short the very first application of a arbitrary preconditioner is likely to destroy our compact representation, which is why it necessary to rethink the entire approach.

11.2 Enabling preconditioning

In this section we explain how to enable preconditioning, without using $O(n^2)$ memory and time. We begin by illustrating why it is even possible.

We applied preconditioned CG to the equation

$$AX + XA^T = Q,$$

where $A = [a_{ij}]$, and

$$a_{ij} = \begin{cases} 3, & i = j \\ -1.4, & |i-j| = 1 \\ -0.1, & |i-j| = 10 \end{cases}$$

and $Q = \left[q_{ij}\right]$, where $q_{ij} = 1$, using the operator

$$\phi_M(X) = MX + XM^T$$

as preconditioner, where $M = [m_{ij}]$, and

$$m_{ij} = \begin{cases} 3, & i = j, \\ -1.4, & |i-j| = 1, \end{cases}$$

We recorded the residual history as well as the numerical rank of the iterates X_j with respect to the tolerance 10^{-6}. Our results can be found in Figure 11.1

As expected the residual is decaying rapidly and the relative residual is less than 10^{-6} after 72 iterations. The numerical rank of the iterates X_j grows almost linearly, until it peaks at 35 and falls down to 7, which is equal to the numerical rank of the exact solution with respect to the tolerance 10^{-6}. This experiment shows that it is at least theoretically possible to carry out the calculations using matrices with low rank.

We now show how to carry out the calculations in practice. We claim that it suffices to work with vectors $\tilde{x}$ which can be written in the form

$$\tilde{x} = \mathrm{vec}(EF^T)$$

where E and F are real n by r matrices, where $r \ll n$. It is clear that the inhomogeneous term is already in this format, but we must show that it is possible to continue using this format.

We begin by showing how to apply the matrix

$$\tilde{A} = (I \otimes A + A \otimes I)$$

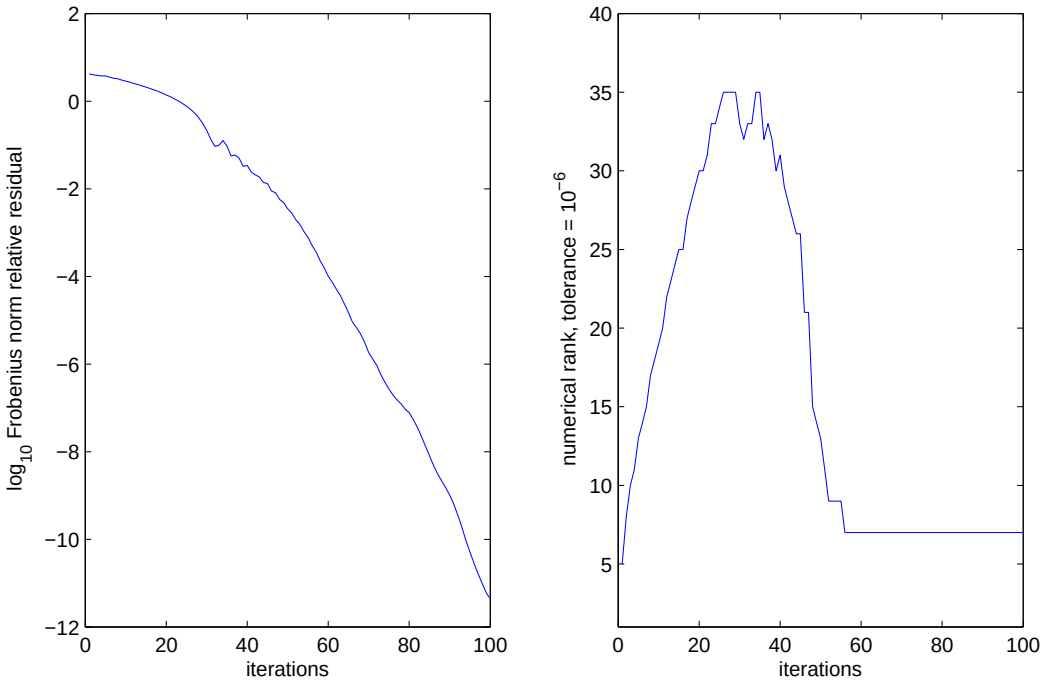

Figure 11.1. The residual history and the numerical rank of the approximate solutions for preconditioned CG applied to a simple Lyapunov equation

to a vector in this format. We have

$$\tilde{A}\tilde{x} = \text{vec}\left(AEF^T + EF^TA^T\right) = \text{vec}\left(\begin{bmatrix} AE & E \end{bmatrix}\begin{bmatrix} F & AF \end{bmatrix}\right).$$

This simple expression is actually very problematic. Each application of $\tilde{A}$ doubles the storage requirement. It is clear that we run out of memory, sooner rather than later, if $\tilde{A}$ is applied repeatedly. This growth can be kept in check by regularly performing tall SVD.

Now, a standard linear update is equally simple. If $\tilde{y} = \text{vec}(GH^T)$, then

$$\alpha\tilde{x} + \tilde{y} = \text{vec}\left(\alpha EF^T + GH^T\right) = \text{vec}\left(\begin{bmatrix} \alpha E & G \end{bmatrix}\begin{bmatrix} F & H \end{bmatrix}\right)$$

It is more complicated to compute the 2-norm of a vector $\tilde{x} = \text{vec}(EF^T)$. We have

$$\|\tilde{x}\|_2^2 = \|EF^T\|_F$$

Now, it is possible to form the n by n matrix EF^T and then calculate the Frobenius norm directly. This can be done one block at a time and does not necessarily require $O(n^2)$ memory, but $O(n^2)$ arithmetic operations, which limits this procedure to small values of n.

However, there is a shortcut, since

$$\|EF^T\|_F^2 = \text{trace}((EF^T)^TEF^T) = \text{trace}((FE^T)EF^T) = \text{trace}\left((E^TE)(F^TF)\right)$$

which allow us to calculate the norm using $O(np^2)$ arithmetic operations and only $O(p^2)$ memory. Unfortunately this procedure is not reliable, to the point where the even the sign can not be trusted. This situation can occur when the true norm is sufficiently small. Mathematically the problem is identical to the problem of calculating the inner product between a pair of vectors in $\mathbb{R}^n$. The standard algorithm for this problem can have a very large relative error, when the vectors are nearly orthogonal.

There is a way to get the sign right. Let $E = U\Sigma V^T$ be the SVD of E. Then

$$FE^TEF^T = FV\Sigma^2V^TF^T = (FV)\Sigma^2(FV)^T$$

from which it follows that

$$\text{trace}\left(FE^TEF^T\right) = \text{trace}\left((FV)\Sigma^2(FV)^T\right) = \Sigma_{j=1}^k \sigma_j^2 \|(FV)_j\|_2^2,$$

and this expression is guaranteed to return a nonnegative number even in the face of roundoff errors. Unfortunately this expression is also unreliable and problems occur when the true norm becomes sufficiently small. We believe that the problem lies within the SVD. With the current algorithm in LAPACK, the large singular values are determined with a small relative error, but the small singular values are determined with an large relative error. While the singular vectors are orthogonal almost to machine precision, their direction depends on the clustering of the singular values and in the above expression we depend on getting the singular vectors right.

Regardless, there is simple solution to our problem which does not involve a singular value decomposition. Let $E = QR$, $F = WS$ be the QR factorizations of E, and F. Then

$$FE^TEF^T = WSR^TQ^TQRS^TW^T = WSR^TRS^TW^T$$

from which it follows, that

$$\|EF\|_F^2 = \|(R^TR)(S^TS)\|_F^2.$$

We found that this expression is numerically reliable even when the true norm is very small.

Let $\tilde{y} = \text{vec}(GH^T)$, and consider the problem of calculation the inner product $\tilde{x}^T\tilde{y}$. We have

$$\tilde{x}^T\tilde{y} = \text{trace}\left((EF^T)^TGH^T\right) = \text{trace}(FE^TGH^T) = \text{trace}((E^TG)(H^TF))$$

and while this expression offers a computational shortcut, we already know it is unreliable in the special case of $E = G$, and $F = H$. Is it possible to avoid this problem? We have not been able to answer this question with any certainty, but such an algorithm would immediately allow us compute an regular inner product x^Ty with

a small relative error regardless of x and y being nearly orthogonal. This follows from the simple observation that if $\tilde{x} = \text{vec}(xe_1)^T$, and $\tilde{y} = \text{vec}(ye_1^T)$, then

$$\tilde{x}^T\tilde{y} = \text{trace}(e_1x^Tye_1^T) = \text{trace}(x^Tye_1^Te_1) = x^Ty.$$

Now, regardless of the choice of the preconditioner $\tilde{M}$ it is critical, that the solution $\tilde{y}$ of every linear system

$$\tilde{M}\tilde{y} = \text{vec}(EF^T).$$

admits a sufficiently good approximation of the same form, i.e.

$$\tilde{y} \approx \text{vec}(GH^T).$$

where G, and H are tall matrices. In addition we must be able to calculate G, and H directly, i.e. without forming the vector $\tilde{y}$ explicitly. If this can not be done, then we end up using $O(n^2)$ memory, which we can not afford.

This is a major difference between matrix equations and standard linear equations, where it suffices that $My = g$ can be solved efficiently and M^{-1} is a good approximation of A^{-1}.

Now are there any preconditioners which will satisfy this extra condition? We can precondition the original problem with another Lyapunov equation,

$$MX + XM^T + Q = 0$$

and use $(I \otimes M + M \otimes I)^{-1}$ as an approximation to $(I \otimes A + A \otimes I)^{-1}$. If M is a stable matrix with a tightly clustered spectrum and an eigenbasis with is well conditioned or if M is negative definite with $\lambda_{max}(M + M^T) \ll 0$, then we know that X will have a good low rank approximation as long as Q has low rank.

11.3 Preconditioners

In this section we consider different classes of preconditioners for Lyapunov equations in Kronecker product form.

Regardless of choice of the preconditioner $\tilde{M}$ it will be bijective linear map from $\mathbb{R}^{n^2}$ into $\mathbb{R}^{n^2}$. As a result we can not afford to form $\tilde{M}$ explicitly, because even the sparsest matrix still requires at least one entry per row in order to be nonsingular. It follows that many of the techniques used to preconditioning a standard linear systems

$$Ax = f$$

can not be applied in our case, because they would return a matrix with n^2 nonzero rows. In particular we can not apply the incomplete Cholesky, LU or QR factorization regardless of the drop tolerance and the level of fill.

11.3.1 Preconditioners based on splittings of A

Every splitting of $A = M - N$ introduces a splitting of $\tilde{A}$, into

$$\tilde{A} = \tilde{M} - \tilde{N}$$

where $\tilde{M} = I \otimes M + M \otimes I$, and $\tilde{N} = I \otimes N + N \otimes I$. Now consider $\tilde{M}$ as a preconditioner for $\tilde{A}$. We would like the eigenvalues of $\tilde{M}^{-1}\tilde{N} = I - \tilde{M}^{-1}\tilde{A}$ contained in a closed disk $D(0, \rho)$ given by

$$D(0, \rho) = \{z \in \mathcal{C} \, : \, |z| \leq \rho\}$$

If $\lambda \in \sigma(\tilde{M}^{-1}\tilde{N})$ then there exists a nonzero matrix X such that

$$\tilde{M}^{-1}\tilde{N}\text{vec}(X) = \lambda \text{vec}(X)$$

or equivalently

$$(N - \lambda M)X + X(N - \lambda M)^T = 0$$

It follows that in order to eliminate the possibility of $|\lambda| \geq \rho$ it suffices to ensure that every Lyapunov equation of the form

$$(N - \lambda M)X + X(N - \lambda M)^T = Q \tag{11.1}$$

has a unique solution.

There is at least one case where this is can be done. If M is symmetric positive definite and if N is symmetric, then $\tilde{M}^{-1}\tilde{N}$ is similar to a symmetric matrix and λ must be real. If N is bounded relative to M, i.e.

$$|x^T N x| \leq \rho\, x^T M x$$

for all $x \in \mathbb{R}^n$, then $(N - \lambda M)$ is stable for all $\lambda > \rho$ and anti stable for all $\lambda < -\rho$. It follows that equation (11.1) can not have a nontrivial solution for all $|\lambda| > \rho$, and we deduce $\sigma(I - \tilde{M}^{-1}\tilde{N}) \subseteq [-\rho, \rho]$.

11.3.2 Preconditioners based on structural simplification of A

In this subsection we consider preconditioners of the form

$$\tilde{M} = I \otimes M + M \otimes I,$$

where M is any matrix, such that $\tilde{M}$ is nonsingular. We show that if

$$\|I_{n^2} - \tilde{M}^{-1}\tilde{A}\|_2$$

is sufficiently small, then A is a good approximation of M, in the sense that

$$\frac{\|M - A\|_2}{\|M\|_2}$$

is small. We also show that if

$$I_{n^2} - \tilde{M}^{-1}\tilde{A}$$

has rank $k < n$, then $M = A$.

Let n denote the dimension of A and ϕ_A be given by Lyapunov operator given by

$$\phi_A(X) = AX + XA^T$$

for all real n by n matrices X.

Theorem 11.3.1 *The map $A \to \|\phi_A\|_F$ is a norm on the set of real n by n matrices.*

Proof Let A be an n by n matrix and let α be a real number. We must show

$$\|\phi_{\alpha A}\|_F = |\alpha|\|\phi_A\|_F$$

Let X be any n by n matrix. Then

$$\|\phi_{\alpha A}(X)\|_F = \|\alpha AX + X(\alpha A)^T\|_F = |\alpha|\|AX + XA^T\|_F = |\alpha|\|\phi_A(X)\|_F$$

which implies, that

$$\|\phi_{\alpha A}\|_F = |\alpha|\|\phi_A\|_F.$$

Let A_1, and A_2 be n by n matrices. We must show that

$$\|\phi_{A_1+A_2}\|_F \leq \|\phi_{A_1}\|_F + \|\phi_{A_2}\|_F.$$

Let X be any n by n matrix. Then

$$\begin{aligned}\phi_{A_1+A_2}(X) &= (A_1 + A_2)X + X(A_1 + A_2)^T \\ &= (A_1X + XA_1^T) + (A_2X + XA_2^T) = \phi_{A_1}(X) + \phi_{A_2}(X)\end{aligned}$$

which implies

$$\begin{aligned}\|\phi_{A_1+A_2}(X)\|_F &\leq \|\phi_{A_1}(X)\|_F + \|\phi_{A_2}(X)\|_F \\ &\leq \|\phi_{A_1}\|_F\|X\|_F + \|\phi_{A_2}\|_F\|X\|_F \quad (11.2)\end{aligned}$$

from which it follows, that

$$\|\phi_{A_1+A_2}\|_F \leq \|\phi_{A_1}\|_F + \|\phi_{A_2}\|_F$$

Finally, we must show that $\|\phi(A)\|_F = 0$ implies $A = 0$. Let A be a matrix such that $\|\phi(A)\|_F = 0$. Let X be any n by n matrix. Then

$$\|AX + XA^T\|_F = \|\phi_A(X)\|_F \leq \|\phi_A\|_F\|X\|_F = 0$$

which implies that

$$AX + XA^T = 0,$$

for all n by n matrices X. In particular, for $X = I$,

$$A + A^T = 0$$

It follows, that

$$AX = XA$$

for all matrices X, which implies $A = \alpha I$ for a suitable choice of $\alpha \in \mathbb{R}$. We claim that $\alpha = 0$ is the only possibility. However, $A + A^T = 0$ implies $2\alpha I = 0$, or equivalently $\alpha = 0$. ■

The following theorem is an immediate corollary.

Theorem 11.3.2 *There exists a pair of positive constants c_1 and c_2, such that*

$$c_1\|\phi_A\|_F \leq \|A\|_2 \leq c_2\|\phi A\|_F$$

for all real n by n matrices A.

We remark, that it is critical that we restrict ourselves to real matrices A, because the complex matrix $A = iI \neq 0$ is such that

$$\|AX + XA^*\|_F = 0$$

for all complex matrices X.

Now, if M is any real matrix such that $\tilde{M} = I \otimes M + M \otimes I$ is nonsingular, then

$$\frac{\|M - A\|_2}{\|M\|_2} \leq \frac{c_2}{c_1}\frac{\|\phi_{M-A}\|_F}{\|\phi_M\|_F} = \frac{c_2}{c_1}\frac{\tilde{M} - \tilde{A}\|_2}{\|\tilde{M}\|_2} \leq \frac{c_2}{c_1}\|I_{n^2} - \tilde{M}^{-1}\tilde{A}\|_2$$

Hodel [24] showed that we may use $c_1 = \frac{1}{2}$, but we have not been able to bound c_2 independent of n. However, we can still deduce that if $\tilde{M}$ is a sufficiently good preconditioner for $\tilde{A}$, in the sense that

$$\|I_{n^2} - \tilde{M}^{-1}\tilde{A}\|_2$$

is sufficiently small, then A is a good approximation of M, in the sense that

$$\frac{\|M - A\|_2}{\|M\|_2}$$

is small. This indicates why we have not been able to find good preconditioners of the type $\tilde{M} = I \otimes M + M \otimes I$ unless we choose M close to A.

Theorem 11.3.3 *Let A be any n by n matrix and let k denote the rank of*

$$\tilde{A} = I \otimes A + A \otimes I.$$

If $k < n$, then $A = 0$.

Proof Since M has rank k there exists a pair of n^2 by k matrices U, and V, such that $\tilde{A} = UV^T$. We partition U, and V into n blocks consisting of n rows,

$$U = \begin{bmatrix} U_1^T & U_2^T & \dots & U_n^T \end{bmatrix}^T$$
$$V = \begin{bmatrix} V_1^T & V_2^T & \dots & V_n^T \end{bmatrix}^T$$

and

$$\tilde{A} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \dots & \tilde{A}_{1n} \\ \tilde{A}_{21} & \ddots & & \tilde{A}_{2n} \\ \dots & & \ddots & \dots \\ \tilde{A}_{n1} & \tilde{A}_{n2} & \dots & \tilde{A}_{nn} \end{bmatrix}$$

Now consider an off-diagonal block $\tilde{A}_{ij}$, $i \neq j$. By definition

$$\tilde{A}_{ij} = a_{ji} I_n$$

where I_n is the n by n identity matrix. It is clear that $\tilde{A}_{ij}$ has rank n if and only if $a_{ji} \neq 0$. By assumption

$$\tilde{A}_{ij} = U_i V_j^T$$

from which it follows that the block M_{ij} has rank at most $k < n$. The only possibility is $a_{ji} = 0$. We conclude that A is diagonal.

Now, in general the rank of $\tilde{A}$ is determined by the spectrum of A, specifically

$$\text{rank}(M) = \#\{(i,j) \ : \ \lambda_i + \lambda_j \neq 0\}$$

and since we have deduced that A is diagonal, the eigenvalues are the diagonal entries. Now let r denote the number of nonzero diagonal entries of A. When we pair each of

these entries with itself or with one of the $n-r$ zeros, we generate a nonzero entry on the main diagonal of M. It follows, that

$$r(n-r+1) \leq k$$

Now, by assumption $k < n$, which implies

$$r(n-r+1) < n, \tag{11.3}$$

At this point we can immediately eliminate $r = 1$ and $r = n$. Let

$$r \in \{2, 3, \ldots, n-1\}.$$

Our goal is to derive a contradiction. By rearranging (11.3) and using $r \geq 2$ we find

$$(r-1)n < r^2 - r < r^2 - 1 = (r-1)(r+1),$$

from which it follows, that

$$n < r + 1 \leq n$$

which is impossible. We are left with $r = 0$ as the only choice and we conclude $A = 0$.

■

Now suppose that we have chosen M, such that

$$I \otimes M + M \otimes I \tag{11.4}$$

is nonsingular and

$$I_n^2 - (I \otimes M + M \otimes I)^{-1}(I \otimes A + A \otimes I) \tag{11.5}$$

has rank k. We claim that if $k < n$, then $M = A$. By assumption

$$(I \otimes M + M \otimes I) - (I \otimes A + A \otimes I) = (I \otimes (M - A) + (M - A) \otimes I) \tag{11.6}$$

has rank $k < n$. By applying the previous theorem to $M - A$ we conclude that $M = A$.

We conclude that it is impossible to find a preconditioner $\tilde{M}$ for $\tilde{A}$ of the form

$$\tilde{M} = I \otimes M + M \otimes I$$

such that $\tilde{M}$ is nonsingular and

$$I_n^2 - (I \otimes M + M \otimes I)^{-1}(I \otimes A + A \otimes I) \tag{11.7}$$

has rank $k < n$, unless we pick $M = A$, which is of no interest.

This is another significant difference between Lyapunov equations and standard linear equations. If A is almost block diagonal save for say $k \ll n$ nonzero entries, then one could solve $Ax = f$ using the main block diagonal D as preconditioner. This scheme would be successful, because $I - D^{-1}A$ would have rank $k \ll n$. However, for Lyapunov equations we simply will not experience this phenomenon, and if we converge after a few outer iterations, then it is not because

$$I_n^2 - (I \otimes M + M \otimes I)^{-1}(I \otimes A + A \otimes I)$$

had low rank.

11.4 Conclusion

In this chapter we have seen that it is theoretically possible to precondition Lyapunov equations in Kronecker product form using $O(n)$ memory and time. We saw that the compact representation developed in the previous chapter would not allow preconditioning to take place, and we showed that the calculations could be carried out using a set of specialized vectors $\tilde{x} \in \mathbb{R}^{n^2}$ of the form $\tilde{x} = \text{vec}(EF^T)$ were E and F are tall matrices. It became apparent that any preconditioner $\tilde{M}$ must respect the new format in the sense that the solution $\tilde{y}$ of the equation $M\tilde{y} = \text{vec}(EF^T)$ should admit a good low rank approximation $\tilde{y} \approx \text{vec}(GH^T)$ for every pair of tall matrices E, F. In addition, we must be able to extract the low rank approximation without using $O(n^2)$ memory. This is the fundamental difference between preconditioners for

standard linear systems and preconditioners for Lyapunov equations in Kronecker product form.

We can not use a general nonsingular n^2 by n^2 matrix $\tilde{M}$ as preconditioner, because we will run out of memory just trying to represent the map. Naturally, we considered preconditioners $\tilde{M}$ which could be written in the form

$$\tilde{M} = I \otimes M + M \otimes I$$

for a suitable matrix M. If M is negative definite matrix or if the eigenvalues of M are clustered around a few points, then it is possible to solve

$$MX + XM^T + EF^T = 0$$

efficiently using either the Arnoldi method or one of the ADI methods, and a good low rank approximation for the solution is readily available. We saw that if $\tilde{M}$ was a successful preconditioner in the sense that the number of outer iterations is guaranteed to be small, then A is a good approximation of M. Now, if Lyapunov equations with coefficient M can be solved quickly using either the Arnoldi method or an ADI methods, then by continuity the same holds for Lyapunov equations with coefficient matrix A. The cost of enabling preconditioning is high, because we must regularly perform tall SVDs in order to keep memory consumption down.

12. The SPIKE algorithms

12.1 Introduction

The SPIKE algorithms are designed to solve narrow banded linear systems

$$Ax = f \tag{12.1}$$

on parallel machines.

Narrow banded linear systems occur naturally in many different fields. There are sparse matrices who can either be reordered into narrow banded form or who admit a good narrow banded preconditioner after reordering.

Most numerical methods for Lyapunov equations rely on the solution of linear systems. The best example is the ADI family of methods which require the solution of linear systems with coefficient matrix $A+pI$ for certain values of the shift parameter p. It is difficult to automatically select a good preconditioner for $A + pI$, because the properties of the matrix may depend non-trivially on p. However, if A can be reordered as a narrow banded matrix, then the systems can be solved automatically using, say, Gaussian elimination with partial pivoting.

The Arnoldi method, the API, our approximate subspace iteration with Ritz acceleration and the AISAID algorithm all require the action of A on a vector or a block of vectors. Frequently, A is not given explicitly, and the action of A can only be computed by solving linear systems.

In this chapter we study the SPIKE algorithm for solving narrow banded linear systems. Our analysis has immediate implications for the overlapping partition method (OPM). Finally, our analysis of these methods raised some questions regarding the nature of narrow banded linear systems and their solution on a parallel machine.

We now turn to the SPIKE algorithms. The main idea was introduced by Sameh and Kuck [53] who considered the tridiagonal case and Chen, Kuck and Sameh [8] who studied the triangular case. Lawrie and Sameh [34] applied the algorithm to the symmetric positive definite systems, while Dongarra and Sameh [10] considered the strictly diagonally dominant case. Berry and Sameh [6] consider ideas central to the SPIKE algorithms. Variations of the SPIKE algorithms for tridiagonal systems were introduced by Sun, Zhang, and Ni [65], who also analyzed the truncation error for tridiagonal systems, which are evenly diagonally dominant. The truncation error for systems which are Toeplitz, tridiagonal, strictly diagonally dominant, and symmetric or skew symmetric was considered by Sun [64]. Another variation of the SPIKE algorithm for strictly diagonally dominant systems was studied by Larriba-Pey, Jorba and Navarro [32].

A large part of the material presented here has been drawn from Mikkelsen and Manguoglo [37], as well as Mikkelsen [36].

We assume for the sake of simplicity that the number of superdiagonals k equals the number of subdiagonals and that the number of processors p divides the n. Let the system be partitioned into the block diagonal form shown below

$$Ax = \begin{bmatrix} A_1 & \overline{B}_1 & & & \\ \overline{C}_2 & A_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \overline{B}_{p-1} \\ & & & \overline{C}_p & A_p \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_p \end{bmatrix}. \tag{12.2}$$

where $A_i, i = 1, 2, \ldots p$, is a banded matrix of order $\mu = n/p$, and bandwidth $(2k+1)$, which is inherited from A, and

$$\overline{B}_i = \begin{bmatrix} 0 & 0 \\ B_i & 0 \end{bmatrix}, \quad \text{and} \quad \overline{C}_{i+1} = \begin{bmatrix} 0 & C_{i+1} \\ 0 & 0 \end{bmatrix} \quad i = 1, 2, \ldots p-1,$$

in which B_i and C_{i+1} are lower and upper triangular matrices, respectively, each of order k.

In this chapter we assume that the main block diagonal D

$$D = \text{diag}\{A_1, A_2, \ldots A_p\} \tag{12.3}$$

is nonsingular. This is a nontrivial assumption and it is entirely possible for a well conditioned matrix to have a singular diagonal block. A specific example of a tridiagonal matrix A with this property is given by

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

for which the infinity norm condition number is

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = 15,$$

while the main block diagonal D corresponding to $p = 2$,

$$D = \text{diag}\left\{ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \right\}$$

is clearly a singular matrix of rank 4.

The explicit SPIKE algorithm applies to matrices for which the main block diagonal is nonsingular, while the truncated SPIKE algorithm applies to systems which are strictly diagonally dominant by rows.

A matrix $A = [a_{ij}]$ is diagonally dominant by rows if

$$\sum_{i \neq j} |a_{ij}| \leq |a_{ii}|, \tag{12.4}$$

for all i. If the inequality is sharp, then A is strictly diagonally dominant by rows.

Our main contribution is the analysis of the truncated SPIKE algorithm. The algorithm flows naturally from the explicit SPIKE algorithm.

12.2 The explicit SPIKE algorithm

The explicit SPIKE algorithm consists of four stages. Our presentation has been adapted from Dongarra and Sameh [10] with minor changes to the notation. The fundamental assumption is that the main diagonal blocks are nonsingular.

Stage 1 Compute the LU factorization

$$A_i P_i = L_i U_i, \quad i = 1, 2, \ldots p,$$

using Gaussian elimination with partial pivoting, one processor per factorization. Here L_i is unit lower triangular, U_i is a nonsingular upper triangular matrix, and P_i is a permutation matrix.

Stage 2 If we premultiply both sides of (12.2) by D^{-1} we obtain a system $Sx = g$ of the form

$$\begin{bmatrix} I_\mu & \overline{V}_1 & & & \\ \overline{W}_2 & I_\mu & \overline{V}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \overline{W}_{p-1} & I_\mu & \overline{V}_{p-1} \\ & & & \overline{W}_p & I_\mu \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{p-1} \\ x_p \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{p-1} \\ g_p \end{bmatrix} \tag{12.5}$$

where

$$\overline{V}_i = \begin{bmatrix} V_i, & 0 \end{bmatrix}, \quad \text{and} \quad \overline{W}_i = \begin{bmatrix} 0, & W_i \end{bmatrix}$$

in which V_i and W_i are matrices with k columns given by

$$V_i = A_i^{-1} \begin{bmatrix} 0 \\ B_i \end{bmatrix}, \quad \text{and} \quad W_i = A_i^{-1} \begin{bmatrix} C_i \\ 0 \end{bmatrix},$$

and will in general be full. In other words, V_i, W_i, and g_i are obtained by solving the linear systems

$$L_i U_i \begin{bmatrix} V_i, & W_i, & g_i \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ B_i \end{bmatrix}, & \begin{bmatrix} C_i \\ 0 \end{bmatrix}, & f_i \end{bmatrix} P_i^{-1}, \quad i = 1, 2, \ldots p,$$

where $C_1 = 0$, and $B_p = 0$.

Stage 3 Let the tall and narrow matrices V_i, and W_i be partitioned, in turn, as follows

$$V_i = \begin{bmatrix} V_i^{(t)} \\ V_i^{(m)} \\ V_i^{(b)} \end{bmatrix}, \quad \text{and} \quad W_i = \begin{bmatrix} W_i^{(t)} \\ W_i^{(m)} \\ W_i^{(b)} \end{bmatrix},$$

where the tips, i.e. the matrices $V_i^{(t)}$, $V_i^{(b)}$, $W_i^{(t)}$, and $W_i^{(b)}$ are small k by k blocks. The superscripts t, m, and b are abbreviations of the words top, middle, and bottom, respectively. Let x_i, and g_i be partitioned conformally, i.e.

$$x_i = \begin{bmatrix} x_i^{(t)} \\ x_i^{(m)} \\ x_i^{(b)} \end{bmatrix}, \quad \text{and} \quad g_i = \begin{bmatrix} g_i^{(t)} \\ g_i^{(m)} \\ g_i^{(b)} \end{bmatrix}.$$

As an illustration we show the system (12.5) for $p = 3$, where $\nu = \mu - 2k$,

$$\begin{bmatrix} I_k & & & V_1^{(t)} & & & & & \\ & I_\nu & & V_1^{(m)} & & & & & \\ & & I_k & V_1^{(b)} & & & & & \\ & & W_2^{(t)} & I_k & & & V_2^{(t)} & & \\ & & W_2^{(m)} & & I_\nu & & V_2^{(m)} & & \\ & & W_2^{(b)} & & & I_k & V_2^{(b)} & & \\ & & & & & W_3^{(t)} & I_k & & \\ & & & & & W_3^{(t)} & & I_\nu & \\ & & & & & W_3^{(t)} & & & I_k \end{bmatrix} \begin{bmatrix} x_1^{(t)} \\ x_1^{(m)} \\ x_1^{(b)} \\ x_2^{(t)} \\ x_2^{(m)} \\ x_2^{(b)} \\ x_3^{(t)} \\ x_3^{(m)} \\ x_3^{(b)} \end{bmatrix} = \begin{bmatrix} g_1^{(t)} \\ g_1^{(m)} \\ g_1^{(b)} \\ g_2^{(t)} \\ g_2^{(m)} \\ g_2^{(b)} \\ g_3^{(t)} \\ g_3^{(m)} \\ g_3^{(b)} \end{bmatrix}.$$

It is from these narrow block columns or spikes extending from the main diagonal that the algorithm has derived its name. We will frequently refer to the V_i as the superdiagonal spikes and to the W_i as the subdiagonal spikes, and call the matrix S the SPIKE matrix.

Observe that the union of the k equations above and the k equations below the $p-1$ partition lines form an independent subsystem of order $2k(p-1)$, which we shall refer to as the "reduced" system $Rx_r = g_r$, which is of the form

$$\begin{bmatrix} E_1 & F_1 & & & \\ G_2 & E_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & F_{p-2} \\ & & & G_{p-2} & E_{p-1} \end{bmatrix} \begin{bmatrix} x_{r,1} \\ x_{r,2} \\ \vdots \\ x_{r,p-1} \end{bmatrix} = \begin{bmatrix} g_{r,1} \\ g_{r,2} \\ \vdots \\ g_{r,p-1} \end{bmatrix}, \tag{12.6}$$

where

$$E_i = \begin{bmatrix} I_k & V_i^{(b)} \\ W_{i+1}^{(t)} & I_k \end{bmatrix}, \quad F_i = \begin{bmatrix} 0 & 0 \\ 0 & V_{i+1}^{(t)} \end{bmatrix}, \quad \text{and} \quad G_i = \begin{bmatrix} W_i^{(b)} & 0 \\ 0 & 0 \end{bmatrix},$$

and

$$x_{r,i} = \begin{bmatrix} x_i^{(b)} \\ x_{i+1}^{(t)} \end{bmatrix}, \quad \text{and} \quad g_{r,i} = \begin{bmatrix} g_i^{(b)} \\ g_{i+1}^{(t)} \end{bmatrix}.$$

The subscript r is an abbreviation of the word "reduced".

Stage 4 Once the reduced system has been solved, processor i computes

$$z_i = g_i - W_i^{(b)} x_{i-1}^{(b)} - V_i^{(t)} x_{i+1}^{(t)},$$

where x_0, and x_{p+1} are undefined and should be taken to zero in this equation. If the calculations are carried out using exact arithmetic, then z is the solution of $Ax = f$.

Solving the reduced system is the only part of the algorithm which requires communication. The reduced system is spread across the machine. The matrices $V_1^{(b)}$, and $g_1^{(b)}$ are calculated by processor 1, the matrices $V_i^{(t)}, V_i^{(b)}, W_i^{(t)}, W_i^{(b)}, g_i^{(t)}$, and $g_i^{(b)}$ are calculated by processor i, for $i = 2, 3, \ldots, p-1$, while the matrices $W_p^{(t)}$, and $g_p^{(t)}$ are calculated by processor p.

Dongarra and Sameh [10] noted that if A is strictly diagonally dominant by rows, then the SPIKE system and the reduced system are strictly diagonally dominant

by rows and they solved the reduced system using a parallel implementation of the Jacobi iteration. In Theorem 12.4.1 we show that the SPIKE matrix and the reduced system are strictly diagonally dominant by rows with a degree no less than A.

In general the reduced system is block tridiagonal. However, Polizzi and Sameh [43] noted that the off-diagonal block are often negligible and can be dropped, yielding a truncated reduced system $Tx_{tr} = g_r$, which is block diagonal,

$$\begin{bmatrix} E_1 & & & & \\ & E_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & E_{p-1} \end{bmatrix} \begin{bmatrix} x_{tr,1} \\ x_{tr,2} \\ \vdots \\ x_{tr,p-1} \end{bmatrix} = \begin{bmatrix} g_{r,1} \\ g_{r,2} \\ \vdots \\ g_{r,p-1} \end{bmatrix}. \tag{12.7}$$

The subscript tr is an abbreviation of the words "truncated" and "reduced". We will establish a tight upper bound on the size of the off-diagonal blocks in Section 12.4.2.

We have made some large drawings of the partitioning and the systems relevant to the SPIKE algorithms, see Appendix B.

Special attention has been given to matrices which are diagonally dominant. A tridiagonal matrix is A given by

$$\begin{bmatrix} a_1 & b_1 & & \\ c_2 & \ddots & \ddots & \\ & \ddots & & b_{n-1} \\ & & c_n & a_n \end{bmatrix}$$

is said to be evenly diagonally dominant, if

$$\frac{1}{2}|a_i| \geq \max_i\{|c_i|, |b_i|\},$$

where c_1 and b_n are undefined and should be treated as zero, and

$$b_i c_{i+1} \neq 0, \quad i = 1, 2, \ldots, n-1.$$

Sun, Zhang, and Ni [65] gave an upper bound on the truncation error for tridiagonal matrices which are evenly diagonally dominant. Sun [64] considered the case of tridiagonal Toeplitz matrices, which are strictly diagonally dominant as well as symmetric

or anti symmetric. Larriba-Pey, Jorba and Navarro [32] established an upper bound on the decay rate of the spikes in the case of a general banded matrix which is strictly diagonally dominant by rows. In Theorem 12.4.3 we establish a tight upper bound on the size of the off-diagonal blocks in terms of the degree of diagonal dominance of the original matrix and the size of the partitions.

Polizzi and Sameh [44] showed that it is possible to assemble the truncated reduced system directly without computing the entire matrix S. The key is to exploit the special structure of the right-hand sides. Let $\mathcal{A}$ denote one of the diagonal blocks and consider the problem of computing the bottom $\mathcal{V}^{(b)}$ of the corresponding spike $\mathcal{V}$, given by

$$\mathcal{A}\mathcal{V} = \begin{bmatrix} 0 \\ \mathcal{B} \end{bmatrix},$$

where $\mathcal{B}$ is k by k dense matrix. It is not important here that $\mathcal{B}$ is also lower triangular. We can exploit the remaining structure as follows. Let $\mathcal{A} = LU$ be the LU factorization of $\mathcal{A}$. First we solve

$$LY = \begin{bmatrix} 0 \\ \mathcal{B} \end{bmatrix}.$$

Partition L and Y conformally with the right-hand side,

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathcal{B} \end{bmatrix},$$

where L_{22} is a k by k lower unit triangular matrix. Since $L_{11}Y_1 = 0$ we have $Y_1 = 0$, and the problem reduces to solving $L_{22}Y_2 = \mathcal{B}$. Then we solve $U\mathcal{V} = Y$. Partition U and $\mathcal{V}$ conformally with the Y and the original right-hand side,

$$\begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} \mathcal{V}_1 \\ \mathcal{V}_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}.$$

Since U is upper triangular we can extract the bottom, $\mathcal{V}^{(b)} = \mathcal{V}_2 = U_{22}^{-1}Y_2$, without computing the rest of the spikes, i.e. the matrix $\mathcal{V}_1$. Similarly, it is possible to use

the UL factorization of a diagonal block to extract the top of the corresponding subdiagonal spike without computing the entire spike.

The original equation is equivalent to

$$A_i x = f_i - \overline{C}_i x_{i-1} - \overline{B}_i x_{i+1}, \quad i = 1, 2, \ldots p, \tag{12.8}$$

where $\overline{C}_1$, $\overline{B}_{p+1}$, x_0 and x_{p+1} are undefined and should be taken to zero in this equation.

These observations led to the truncated SPIKE algorithm, which is due to Polizzi and Sameh [43].

12.3 The truncated SPIKE algorithm

We assume that the matrix A is strictly diagonally dominant by rows. This implies that the diagonal blocks are nonsingular, and they have an LU factorization. The algorithm consist of four stages.

Stage 1 Processor i computes the LU/UL factorizations

$$A_i = L_i U_i \quad \text{and} \quad A_i = U_i' L_i',$$

for $i = 1, 2, \ldots p$.

Stage 2 Processor i solves

$$A_i g_i = f_i, \quad i = 1, 2, \ldots p,$$

using the LU factorization. Processor i computes $V_i^{(b)}$ using (L_i, U_i), $i = 1, 2, \ldots p-1$. Processor i computes $W_i^{(t)}$ using (U_i', L_i') , $i = 2, 3 \ldots p$.

Stage 3 Processor $i+1$ sends $W_{i+1}^{(t)}$, and $g_{i+1}^{(t)}$ to processor i, $i = 1, 2, \ldots p-1$. Processor i solves one block of the truncated reduced system, specifically

$$\begin{bmatrix} I_k & V_i^{(b)} \\ W_{i+1}^{(t)} & I_k \end{bmatrix} \begin{bmatrix} x_i^{(b)} \\ x_{i+1}^{(t)} \end{bmatrix} = \begin{bmatrix} g_i^{(b)} \\ g_{i+1}^{(t)} \end{bmatrix}, \quad i = 1, 2, \ldots p-1.$$

Theorem 12.4.1 shows that the truncated reduced system is diagonally dominant by rows with a degree no less than the original matrix, so Gaussian elimination without pivoting is the obvious choice here.

Stage 4 Processor i send $x_i^{(b)}$ to processor $i+1$, $i = 1, 2, \ldots p-1$. Then processor i solves

$$A_i y_i = f_i - C_i x_{i-1}^{(b)} - B_i x_{i+1}^{(t)},$$

using the LU factorization, where C_1, B_p, $x_0^{(b)}$, and $x_{p+1}^{(t)}$ are undefined and should be taken to zero in this equation. The vector y is an approximation of the solution to $Ax = f$.

In their experiments, Polizzi and Sameh [44] found that stage 1 and 2 of the truncated SPIKE algorithm requires much less time than stage 1 and 2 of the explicit SPIKE algorithm. This is true on machines where arithmetic operations require much less time than memory references. The primary reason is that the LU/UL strategy has greater data locality, computing the spikes is a BLAS 2 operation, while computing the LU/UL factorizations is a BLAS 3 operation.

12.4 The matrices S, R, and T

If A is nonsingular and diagonally dominant by rows, then the diagonal entries are non-zero and the (row-wise) dominance factor [13] ϵ is defined as follows

$$\epsilon = \max_i \left\{ \frac{\sum_{i \neq j} |a_{ij}|}{|a_{ii}|} \right\}. \tag{12.9}$$

If $\epsilon > 0$ then the degree of (row-wise) diagonal dominance d is given by

$$d = \epsilon^{-1}. \tag{12.10}$$

The degree of diagonal dominance is central to the analysis of the truncated SPIKE algorithm. In this section we prove that the matrices S, R, and T in (12.5), (12.6), and (12.7) are strictly diagonally dominant by rows with degree no less than the

original matrix A, and we establish an upper bound on their condition number. We bound the truncation error, i.e. the difference between R and T, and show that all our bounds are tight.

Lemma 12.4.1 *Let $n \leq m$ and let A be any n by m matrix which is strictly diagonally dominant by rows with degree $d > 1$. Let $A = LU$ be the LU factorization which is obtained by applying Gaussian elimination without pivoting to A. Then U is strictly diagonally dominant by rows with degree no less than d.*

Proof Gaussian elimination produces a chain of matrices $A^{(j)}$, where the first $j-1$ columns of $A^{(j)}$ are lower triangular, $A = A^{(1)}$ and $A^{(n)} = U$. Due to the recursive nature of Gaussian elimination it suffices to consider the transition from $A = A^{(1)}$ to $B = A^{(2)}$. Let $B = [b_{ij}]$. We must show the following equalities

$$|b_{kk}| \geq d \sum_{j \notin \{1,k\}} |b_{k,j}|, \quad k = 2, 3, \ldots, n$$

Now, since $d \geq 1$ and $|a_{11}| \geq d \sum_{j=2}^{n} |a_{1j}|$ we have

$$\begin{aligned} |a_{kk}| &\geq d \sum_{j \neq k} |a_{kj}| \geq |a_{k1}| + d \sum_{j \notin \{1,k\}} |a_{kj}| \\ &\geq |a_{k1}| \frac{d \sum_{j=2}^{n} |a_{1j}|}{|a_{11}|} + d \sum_{j \notin \{1,k\}} |a_{kj}| \\ &\geq |a_{k1}| \frac{|a_{1k}|}{|a_{11}|} + d \sum_{j \notin \{1,k\}} \left(|a_{kj}| + \frac{|a_{k1}|}{|a_{11}|} |a_{1j}| \right). \end{aligned}$$

Now, since

$$b_{ij} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}, \quad i, j = 2, 3, \ldots, n,$$

the previous inequality implies

$$|b_{kk}| \geq |a_{kk}| - \frac{|a_{k1}|}{|a_{11}|} |a_{1k}| \geq d \sum_{j \notin \{1,k\}} \left(|a_{kj}| + \frac{|a_{k1}|}{|a_{11}|} |a_{1j}| \right) \geq d \sum_{j \notin \{1,k\}} |b_{kj}|.$$

■

Corollary 12.4.1 *Let A be an n by n matrix, and let F be an n by m matrix. If the matrix $\begin{bmatrix} A, & F \end{bmatrix}$ is strictly diagonally dominant by rows with degree d, then the matrix $\begin{bmatrix} I, & A^{-1}F \end{bmatrix}$ is strictly diagonally dominant by rows with degree no less than d.*

Proof Let $\epsilon = d^{-1}$. We use Gaussian elimination with no pivoting to reduce the n by $n+m$ matrix $\begin{bmatrix} A, & F \end{bmatrix}$ to upper triangular form, $U = [u_{ij}]$. By Lemma 12.4.1, U is strictly diagonally dominant by rows with degree no less than d, and using back substitution we have a formula for the entries g_{ij} of the n by m matrix $G = A^{-1}F$, namely

$$g_{n-t,j} = \frac{1}{u_{n-t,n-t}} \left(u_{n-t,n+j} - \sum_{s=n-t+1}^{n} u_{n-t,s} g_{s,j} \right)$$

for $j = 1, 2, \ldots m$ and $t = 0, 1, 2 \ldots n-1$. Let $\Omega \subset \{0, 1, \ldots n-1\}$ be given by

$$t \in \Omega \Leftrightarrow \sum_{j=1}^{m} |g_{n-t,j}| \leq \epsilon.$$

We will prove that $\Omega = \{0, 1, \ldots n-1\}$. First, $0 \in \Omega$ by the diagonal dominance of U, and if $\{0, 1, 2, \ldots t-1\} \subset \Omega$ with $t < n$, then

$$\begin{aligned} \sum_{j=1}^{m} |g_{n-t,j}| &\leq \frac{1}{|u_{n-t,n-t}|} \sum_{j=1}^{m} \left(|u_{n-t,n+j}| + \sum_{s=n-t+1}^{n} |u_{n-t,s}||g_{s,j}| \right) \\ &= \frac{1}{|u_{n-t,n-t}|} \left(\sum_{j=1}^{m} |u_{n-t,n+j}| + \sum_{s=n-t+1}^{n} |u_{n-t,s}| \sum_{j=1}^{m} |g_{s,j}| \right) \\ &\leq \frac{1}{|u_{n-t,n-t}|} \left(\sum_{j=1}^{m} |u_{n-t,n+j}| + \sum_{s=n-t+1}^{n} |u_{n-t,s}| \epsilon \right) \leq \epsilon \end{aligned}$$

which implies $t \in \Omega$. Therefore, $\Omega = \{0, 1, 2 \ldots n-1\}$ and the proof is complete. ■

Theorem 12.4.1 *Let A be strictly diagonally dominant by rows with degree d. Then the matrices S, R, and T are strictly diagonally dominant by rows with degree no less than d, specifically*

$$d \leq d(S) \leq d(R) \leq d(T)$$

with equality possible. In addition the condition numbers share a common bound, namely

$$\max\{ \kappa_\infty(S), \kappa_\infty(R), \kappa_\infty(T)\} \leq \frac{d+1}{d-1},$$

with the possibility of

$$\kappa_\infty(S) = \kappa_\infty(R) = \kappa_\infty(T) = \frac{d+1}{d-1}.$$

Proof If S is strictly diagonally dominant by rows, then it is clear that T, and R are strictly diagonally dominant by rows and $d(T) \geq d(R) \geq d(S)$. By applying Lemma 12.4.1 to the matrices $\begin{bmatrix} A_i, & F_i \end{bmatrix}$, where

$$F_1 = \begin{bmatrix} 0 \\ B_1 \end{bmatrix}, \quad F_i = \left[\begin{bmatrix} 0 \\ B_i \end{bmatrix}, \begin{bmatrix} C_i \\ 0 \end{bmatrix} \right], \; i = 2, \ldots p-1, \quad \text{and} \quad F_p = \begin{bmatrix} C_p \\ 0 \end{bmatrix},$$

we see that S is strictly diagonally dominant by rows with degree no less than d. We estimate the condition number as follows. Since $S_{ii} = 1$, we have $\|S - I\|_\infty \leq \epsilon < 1$, which allows us to treat S as a small perturbation of the identity matrix and estimate

$$\|S^{-1}\|_\infty \leq \frac{1}{1-\epsilon},$$

which implies

$$\kappa_\infty(S) \leq \frac{1+\epsilon}{1-\epsilon} = \frac{d+1}{d-1},$$

and similarly for R, and T.

It remains to be seen that our bounds are tight. To this end, we consider a special matrix A given by

$$A = \begin{bmatrix} I_\mu & \overline{B}_1 & & \\ \overline{C}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \overline{B}_{p-1} \\ & & \overline{C}_p & I_\mu \end{bmatrix},$$

where the off-diagonal blocks are given by

$$\overline{B}_i = \begin{bmatrix} 0 & 0 \\ \epsilon J_k & 0 \end{bmatrix}, \quad \text{and} \quad \overline{C}_{i+1} = \begin{bmatrix} 0 & \epsilon J_k \\ 0 & 0 \end{bmatrix}, \quad i = 1, 2, \ldots p-1,$$

where J_k is the k by k anti diagonal identity matrix,

$$(J_k)_{ij} = \begin{cases} 1 & i = k - j + 1 \\ 0 & i \neq k - j + 1 \end{cases}$$

and $\epsilon \in (0,1)$. The matrix A is strictly diagonally dominant with degree $d = \epsilon^{-1}$. The upper and the lower bandwidths are equal to k. The main block diagonal is equal to the identity matrix, which implies $A = S$. The reduced system is block diagonal, which implies $T = R$. It follows that

$$d(T) = d(R) = d(S) = d.$$

Computing S^{-1} reduces to inverting the 2 by 2 matrix $\left[\begin{smallmatrix} 1 & \epsilon \\ \epsilon & 1 \end{smallmatrix}\right]$. Direct computation establishes that

$$\kappa_\infty(T) = \kappa_\infty(R) = \kappa_\infty(S) = \frac{1+\epsilon}{1-\epsilon} = \frac{d+1}{d-1}.$$

■

12.4.1 The truncation error

We now study the truncation error, i.e. $\|R - T\|_\infty$. Let $\mathcal{A}$ denote one of the diagonal blocks of A, and let $\mathcal{V}$ be the corresponding superdiagonal spike given by

$$\mathcal{A}\mathcal{V} = \begin{bmatrix} 0 \\ \mathcal{B} \end{bmatrix}.$$

We are especially interested in the size of the elements in the top of the spike, i.e. the submatrix $\mathcal{V}^{(t)}$, which is given by

$$\mathcal{V}^{(t)} = \mathcal{V}(1:k, 1:k).$$

There is no loss of generality in limiting the analysis to the first diagonal block, rather there is a slight notational advantage, because the numbering of elements of A and $\mathcal{A}$ coincide. We will use μ to denote the size of the first diagonal block.

We begin by estimating the size of the elements located in the bottom of $\mathcal{V}$, i.e. the submatrix $\mathcal{V}^{(b)}$ given by

$$\mathcal{V}^{(b)} = \mathcal{V}(\mu - k + 1 : \mu, 1 : k).$$

Lemma 12.4.2 *Let A be strictly diagonally dominant by rows with degree d. Let $\mathcal{V}$ be a superdiagonal spike. Then the submatrix $\mathcal{V}^{(b)}$ satisfies*

$$\|\mathcal{V}^{(b)}\|_\infty \leq \epsilon,$$

where $\epsilon = d^{-1}$.

Proof Reduce the first μ by n block row to upper triangular form U. Since Gaussian elimination with no pivoting preserves the upper bandwidth and does not decrease the degree of diagonal dominance, we have the following set of inequalities

$$\sum_{j=1}^{k} |u_{\mu-t,\mu-t+j}| \leq \epsilon |u_{\mu-t,\mu-t}|, \qquad t = 0, 1, \ldots k-1. \tag{12.11}$$

Our goal is to show that $\|\mathcal{V}^{(b)}\|_\infty \leq \epsilon$ or equivalently

$$\sum_{j=1}^{k} |v_{\mu-t,j}| \leq \epsilon, \quad t = 0, 1, \ldots k-1. \tag{12.12}$$

To this end we define the set $\Omega \subset \{0, 1, 2, \ldots, k-1\}$ by

$$t \in \Omega \Leftrightarrow \sum_{j=1}^{k} |v_{n-t,j}| \leq \epsilon.$$

We claim that $\Omega =\subset \{0, 1, 2, \ldots, k-1\}$. Clearly $0 \in \Omega$, because

$$\sum_{j=1}^{k} |u_{\mu,\mu+j}| \leq \epsilon |u_{\mu,\mu}|, \quad \text{and} \quad v_{\mu,j} = \frac{u_{\mu,\mu+j}}{u_{\mu,\mu}}, \quad j = 1, 2, \ldots k.$$

Now, suppose $\{0, 1, 2, \ldots t-1\} \subset \Omega$ with $t < k$. We wish to show that $t \in \Omega$. By back substitution we find that

$$v_{\mu-t,j} = \frac{1}{u_{\mu-t,\mu-t}} \left(u_{\mu-t,\mu+j} - \sum_{s=1}^{t} u_{\mu-t,\mu-t+s} v_{\mu-t+s,j} \right), \quad j = 1, 2, \ldots k-t,$$

and

$$v_{\mu-t,j} = -\frac{1}{u_{\mu-t,\mu-t}} \sum_{s=1}^{t} u_{\mu-t,\mu-t+s} v_{\mu-t+s,j}, \quad j = k-t+1, \ldots k.$$

It follows that

$$\begin{aligned}\sum_{j=1}^{k}|v_{\mu-t,j}| &\leq \frac{1}{|u_{\mu-t,\mu-t}|}\left(\sum_{j=1}^{k-t}|u_{\mu-t,\mu+j}|+\sum_{j=1}^{k}\sum_{s=1}^{t}|u_{\mu-t,\mu-t+s}v_{\mu-t+s,j}|\right)\\ &= \frac{1}{|u_{\mu-t,\mu-t}|}\left(\sum_{j=1}^{k-t}|u_{\mu-t,\mu+j}|+\sum_{s=1}^{t}|u_{\mu-t,\mu-t+s}|\sum_{j=1}^{k}|v_{\mu-t+s,j}|\right)\\ &\leq \frac{1}{|u_{\mu-t,\mu-t}|}\left(\sum_{j=1}^{k-t}|u_{\mu-t,\mu+j}|+\epsilon\sum_{s=1}^{t}|u_{\mu-t,\mu-t+s}|\right)\\ &\leq \frac{1}{|u_{\mu-t,\mu-t}|}\sum_{j=1}^{k}|u_{\mu-t,\mu-t+j}|\leq\epsilon,\end{aligned}$$

which implies $t \in \Omega$. It follows that $\Omega = \{0, 1, 2, \ldots k-1\}$ and $\|\mathcal{V}^{(b)}\|_\infty \leq \epsilon$. ■

We continue with the following lemma which relates the size of elements in a specific row of $\mathcal{V}$ to the infinity norm of the k by k submatrix which lies directly below the row.

Lemma 12.4.3 *Let μ denote the dimension of the diagonal block $\mathcal{A}$ and let $i \geq \mu - k$. Then,*

$$\sum_{j=1}^{k}|v_{i,j}| \leq \epsilon\|\mathcal{V}(i+1:i+k, 1:k)\|_\infty.$$

Proof We have

$$\mathcal{V} = \mathcal{A}^{-1}\begin{bmatrix}0\\ \mathcal{B}\end{bmatrix}$$

for the appropriate k by k matrix $\mathcal{B}$. We use Gaussian elimination with no pivoting to reduce the matrix

$$\begin{bmatrix}\mathcal{A}, & \begin{bmatrix}0\\ \mathcal{B}\end{bmatrix}\end{bmatrix}$$

to upper triangular form $U = [u_{ij}]$. By Lemma 12.4.1 U is strictly diagonally dominant by rows with degree no less than d. Since the original matrix A was banded and no pivoting was applied, it follows that $u_{ij} = 0$ for all i and j such that $j > \mu$ and $i \geq \mu - k$. It follows by back substitution that

$$v_{i,j} = -\frac{1}{u_{i,i}}\sum_{s=i+1}^{i+k}u_{i,s}v_{s,j},$$

which implies

$$\sum_{j=1}^{k} |v_{i,j}| \leq \frac{1}{|u_{i,i}|} \sum_{s=i+1}^{i+k} |u_{i,s}| \sum_{j=1}^{k} |v_{s,j}|.$$

By definition

$$\max_{s=i+1,\ldots i+k} \sum_{j=1}^{k} |v_{s,j}| = \|\mathcal{V}(i+1:i+k, 1:k)\|_\infty,$$

and since U is strictly diagonally dominant by rows with degree no less than d, we have

$$\frac{1}{|u_{i,i}|} \sum_{s=i+1}^{i+k} |u_{i,s}| \leq \epsilon,$$

which completes the proof. ■

The following corollary is an immediate consequence.

Corollary 12.4.2 *Let $\mathcal{V}'$ and $\mathcal{V}''$ be two k by k submatrices of the superdiagonal spike $\mathcal{V}$, such that $\mathcal{V}'$ lies directly on top of $\mathcal{V}''$. Then*

$$\|\mathcal{V}'\|_\infty \leq \epsilon \|\mathcal{V}''\|_\infty.$$

This corollary establishes a chain of inequalities leading from the bottom of the top of the spike which together with Lemma 12.4.2 implies the following theorem

Theorem 12.4.2 *Let d denote the degree of diagonal dominance of A, let μ denote the dimension of one of the diagonal blocks, and $q = \lfloor \mu/k \rfloor$. The elements in the top of the corresponding superdiagonal spike $\mathcal{V}$ satisfy the inequality*

$$\|\mathcal{V}^{(t)}\|_\infty \leq \epsilon^q.$$

Is this estimate for the decay rate of the spikes tight or not? Consider the upper triangular matrix A given by

$$a_{ij} = \begin{cases} 1 & \text{for } i = j, \\ \epsilon & \text{for } i = j - k, \\ 0 & \text{otherwise.} \end{cases}$$

Now consider a partition of a certain size μ. Write $\mu = qk + r$, where $q = \lfloor \mu/k \rfloor$ and the remainder r satisfies $0 \leq r < k$. By back substitution we find that the corresponding spike is given by

$$\mathcal{V} = \begin{bmatrix} \mathcal{V}_{q+1} \\ \mathcal{V}_q \\ \vdots \\ \mathcal{V}_1 \end{bmatrix},$$

where

$$V_j = (-1)^{j-1}\epsilon^j I_k \quad \text{for } j = 1, 2, \ldots, q,$$

and $V_{q+1} = (-1)^q \epsilon^{q+1} E_r$, where I_k is the k by k identity matrix and E_r consists of the last r rows of I_k. Regardless of the value of the remainder r, we have,

$$\|\mathcal{V}^{(t)}\|_\infty = \epsilon^q.$$

In short, if we limit ourselves to matrices A which are strictly diagonally dominant with degree d and upper bandwidth k, i.e.,

$$\max\{j - i \ : \ a_{ij} \neq 0\} = k,$$

then the estimate given in Theorem 12.4.2 is tight.

The following theorem is an immediate consequence of Theorem 12.4.2.

Theorem 12.4.3 *Let A be an n by n matrix which is narrow banded matrix with upper and lower bandwidth k, and strictly diagonally dominant by rows with degree d. Then the truncation error satisfies*

$$\|R - T\|_\infty \leq \max_{i=1,\ldots p} d^{-q_i},$$

where $q_i = \lfloor \mu_i/k \rfloor$ where μ_i is the size of the ith partition and k is its bandwidth.

A better bound exists in the special case in which A is a tridiagonal, evenly diagonally dominant matrix [65] or when A is a tridiagonal Toeplitz matrix, which is strictly diagonally dominant, as well as symmetric or anti symmetric.

What are the consequences of replacing the reduced system with the truncated reduced system? We have the following theorem.

Theorem 12.4.4 *Let A be an n by n matrix which is narrow banded with upper and lower bandwidth k, and strictly diagonally dominant by rows with degree d. Let y be the approximate solution of $Ax = f$, returned by the truncated SPIKE algorithm. Then y is the exact solution of*

$$A_T x = f, \tag{12.13}$$

where A_T satisfies the inequality

$$\|A - A_T\|_\infty \leq \max_{i=1,\ldots p} d^{-q_i} \|A\|_\infty, \tag{12.14}$$

where $q_i = \lfloor \mu_i / k \rfloor$, where μ_i is the size of the ith partition.

Proof Let S_T denote the matrix obtained by eliminating the tips of the spikes from the spike matrix S. Then the reduced system matrix for S_T is equal to T. The truncation error effectively replaces A with the matrix $A_T = DS_T$ for which we have

$$\|A - A_T\|_\infty \leq \|D\|_\infty \|S - S_T\|_\infty \leq \|A\|_\infty \|R - T\|_\infty \leq \max_{i=1,\ldots p} d^{-q_i} \|A\|_\infty. \tag{12.15}$$

■

In short, the effect of the truncation error is to introduce a small norm-wise relative backward error which is bounded by $\max_{i=1,\ldots p} d^{-q_i}$.

We have already seen that the estimate of Theorem 12.4.3 is tight, but which matrices exhibit the slowest possible decay rate? We can answer this question for tridiagonal matrices.

Theorem 12.4.5 *Let $\{(a_i, b_i, c_i)\}_{i=1}^n$ be a finite sequence, such that $a_i \neq 0$, and*

$$\max_{i=1,\ldots,n} \frac{|b_i| + |c_i|}{|a_i|} = \epsilon < 1.$$

If the vector x given by

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & & \\ c_2 & \ddots & \ddots & \\ & \ddots & & b_{n-1} \\ & & c_n & a_n \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ b_n \end{bmatrix},$$

exhibits the smallest possible decay rate, i.e. if

$$|x_1| = \epsilon^n, \tag{12.16}$$

then

$$c_i = 0, \quad \textit{and} \quad |b_i| = \epsilon |a_i|, \tag{12.17}$$

for $i = 1, 2, \ldots n$.

Proof We prove the theorem using Gaussian elimination without pivoting, i.e. the Thomas algorithm [63], which is designed to solve tridiagonal systems of the form

$$c_i x_{i-1} + a_i x_i + b_i x_{i+1} = f_i, \quad i = 1, 2, \ldots, n,$$

where x_0, and x_{n+1} are given in advance. If the strictly diagonally dominant system has degree $d = \epsilon^{-1} > 1$, then the solution can be computed as follows

$$x_i = p_i x_{i+1} + q_i, \quad i = 1, \ldots, n,$$

where the coefficient p_i and q_i are given by

$$p_0 = 0, \quad p_i = \frac{-b_i}{a_i + c_i p_{i-1}}, \quad i = 1, 2, \ldots, n,$$

and

$$q_0 = x_0, \quad q_i = \frac{f_i - c_i q_{i-1}}{a_i + c_i p_{i-1}}, \quad i = 1, 2, \ldots, n.$$

We claim that $|p_i| \leq \epsilon$, for $i = 0, 1, 2, \ldots, n$. If $b_i = 0$, then $p_i = 0$, and there is nothing to show. Assuming $|p_{i-1}| \leq \epsilon < 1$, and $b_i \neq 0$, we have

$$\begin{aligned} |p_i| \leq \frac{|b_i|}{|a_i + c_i p_{i-1}|} &\leq \frac{|b_i|}{|a_i| - |c_i|\epsilon} \\ &\leq \frac{|b_i|}{\epsilon^{-1}(||b_i| + |c_i|) - |c_i|\epsilon} = \frac{|b_i|}{\epsilon^{-1}|b_i| + (\epsilon^{-1} - \epsilon)|c_i|}. \end{aligned} \tag{12.18}$$

Now with $\epsilon \leq 1$, we have $(\epsilon^{-1} - \epsilon)|c_i| \geq 0$, which implies $|p_i| \leq \epsilon$.

In our case $x_0 = x_{n+1} = 0$, and $f_i = 0$ for $i = 1, 2, \ldots, n-1$, while $f_n = b_n$. It follows that

$$q_i = 0, \quad i = 0, 1, 2, \ldots, n-1,$$

while

$$q_n = \frac{b_n}{a_n + c_n p_{n-1}}, \quad \text{and} \quad |q_n| \leq \epsilon.$$

It follows that

$$x_n = q_n, \quad x_i = \left(\Pi_{j=i}^{n-1} p_i\right) q_n,$$

which implies that

$$|x_i| \leq \epsilon^{n-i+1}.$$

Now suppose $|x_1|$ assumes the largest possible value, namely

$$|x_1| = \epsilon^n,$$

then we must have

$$|p_i| = \epsilon, \quad i = 1, 2, \ldots, n-1, \quad \text{and} \quad |q_n| = \epsilon.$$

Now, we claim that this can only happen if $c_i = 0$, for $i = 1, 2, \ldots, n$. Using the same reasoning as in (12.18), we see that we actually have

$$\epsilon = \left| \frac{b_i}{a_i + c_i p_{i-1}} \right| \leq \frac{|b_i|}{|a_i| - |c_i|\epsilon} \leq \epsilon,$$

for $i = 1, 2, \ldots, n-1$, as well as $i = n$. It follows, that

$$\epsilon^2 |c_i| = \epsilon |a_i| - |b_i|.$$

However, $\epsilon |a_i| \geq |b_i| + |c_i|$, leaving us with

$$\epsilon^2 |c_i| = \epsilon |a_i| - |b_i| \geq |c_i|,$$

from which we deduce $|c_i| = 0$, because $\epsilon < 1$. ■

In short, if a tridiagonal matrix, which is also strictly diagonally dominant by rows, exhibits the slowest possible decay rate, then it is actually bidiagonal and the ratio $|b_i|/|a_i|$ is fixed. In our experience the spikes always decay much faster than the worst case.

12.4.2 The general case

The truncated SPIKE algorithm was developed for matrices which are strictly diagonally dominant by rows, but it can also be applied to matrices for which the diagonal blocks are well conditioned. In this section we examine the decay of the spikes in this more general setting, and explain why it is difficult to obtain estimates which are tight.

Demko, Moss and Smith [9] considered the decay rate of the entries of the inverse of band matrices. Given a nonsingular matrix A, they showed the existence of constants $C > 0$ and $\lambda \in (0, 1)$, such that

$$|A^{-1}(i, j)| \leq C\lambda^{|i-j|} \tag{12.19}$$

for all i and j. They gave elementary formulae for C and λ in terms of the smallest interval containing the singular values for A and the half bandwidth k. They showed that if A is narrow banded and well conditioned, then λ will be small and if the smallest singular value of A is not too small, then C will not be too large.

We can use an estimate of this type, i.e. (12.19), to bound the truncation error.

Theorem 12.4.6 *Let n_i denote the size of the i'th diagonal block. Assume $n_i > 2k$ and that constants $C_i > 0$ and $\lambda_i \in (0, 1)$ have been found such that an estimate of the type* (12.19) *is satisfied for every diagonal block A_i, then the difference between the truncated reduced system and the reduced system can be estimated by*

$$\|T - R\|_1 \leq \max_i \left\{ C_i \beta_i \lambda^{n_i - 2k + 1} \left(\frac{1 - \lambda_i^k}{1 - \lambda_i} \right)^2 \right\}, \tag{12.20}$$

where β_i must dominate every element on the i'th block row of A, which is not on the block diagonal of A.

Thus if the C_i are not too large, the λ_i are not too close to 1 and if the block sizes n_i are not too small, then the truncated reduced system will be a good approximation to the reduced system.

Proof The proof is elementary and is included only for the sake of completeness.

Let $\mathcal{A}$ be one the the diagonal blocks $A_1, A_2, \ldots A_{p-1}$ and let μ denote the dimension of $\mathcal{A}$. Consider the corresponding super diagonal spike

$$\mathcal{V} = \mathcal{A}^{-1} \begin{bmatrix} 0 \\ \mathcal{B} \end{bmatrix},$$

where $\mathcal{B} = [b_{ij}]$ is a lower triangular k by k matrix. By assumption constants $C > 0$, and $\lambda \in (0, 1)$ have been found, such that

$$|\mathcal{A}^{-1}(i, j)| \leq C\lambda^{|i-j|},$$

for $i, j = 1, 2, \ldots \mu$. The task at hand is to compute an upper bound for the norm of the very top of the spike, i.e. the square matrix

$$\mathcal{V}(1 : k, 1 : k),$$

in terms of C, and λ. Now, since $\mathcal{B}$ is lower triangular, we have

$$\mathcal{V}(:, t) = \sum_{j=0}^{k-t} b_{\mu-j,t}\mathcal{A}^{-1}e_{\mu-j}, \quad t = 1, 2 \ldots k.$$

Let $\beta = \max |b_{ij}|$. Then the entry $\mathcal{V}_{it}$ is bounded by

$$|\mathcal{V}_{it}| \leq C\beta \sum_{j=0}^{k-t} \lambda^{|i-(\mu-j)|}.$$

If $\mu \geq 2k - 1$ then

$$|i - (\mu - j)| = \mu - (i + j), \quad i = 1, 2, \ldots k, \quad j = 0, 1, \ldots k - 1,$$

and we can estimate further that

$$|\mathcal{V}_{it}| \leq C\beta \sum_{j=0}^{k-t} \lambda^{\mu-i-j} = C\beta\lambda^{\mu-i-(k-t)} \sum_{j=0}^{k-t} \lambda^{k-t-j} = C\beta\lambda^{\mu-i-k+t} \frac{1 - \lambda^{k-t+1}}{1 - \lambda},$$

from which it follows that

$$\sum_{i=1}^{k} |\mathcal{V}_{it}| \leq C\beta \frac{1 - \lambda^{k-t+1}}{1 - \lambda} \sum_{i=1}^{k} \lambda^{\mu-i-k+t} \leq C\beta \frac{1 - \lambda^{k-t+1}}{1 - \lambda} \lambda^{\mu-2k+t} \sum_{i=1}^{k} \lambda^{k-i}$$
$$= C\beta \frac{1 - \lambda^{k-t+1}}{1 - \lambda} \lambda^{\mu-2k+t} \frac{1 - \lambda^{k}}{1 - \lambda}.$$

Thus,

$$\|\mathcal{V}(1:k,1:k)\|_1 \leq C\beta\lambda^{\mu-2k+1}\left(\frac{1-\lambda^k}{1-\lambda}\right)^2,$$

provided that $\mu \geq 2k-1$. It is clear that a similar estimate holds for the spikes below the diagonal, which completes the proof. ■

Is it possible to derive a tight estimate on the decay rate of the spikes in the general case? The spikes are the solutions to some very special linear equations, where both the coefficient matrices and the right-hand sides are drawn from specific positions within the original matrix A. When A is diagonally dominant there is a very close relationship between the diagonal blocks and the corresponding off-diagonal blocks and we are able to exploit this relationship, when we estimate the norm of the bottom of a superdiagonal spike as well as the decay rate of the spikes. In the general case, there is no clear connection between the diagonal and the off-diagonal blocks, and it is even possible to have a well conditioned narrow banded matrix with a singular diagonal block. Diagonal dominance is a global property which ties the individual blocks together and this connection is lost in the general case. It is an open question if there are other classes of matrices for which it is possible to derive a tight bound on the decay rate.

12.4.3 The roundoff errors

In this section we consider an error analysis of the truncated SPIKE algorithm for systems which are strictly diagonally dominant by rows. We begin by deriving a few results on Gaussian elimination for systems which are diagonally dominant with degree $d > 1$, before turning to the truncated SPIKE algorithm.

A matrix which is diagonally dominant by rows with degree $d > 1$ need not be well conditioned, but it is only a row scaling away from being well conditioned with respect to the infinity norm. We now assume that the original problem has been

scaled such that $a_{ii} = 1$. Such a scaling preserves the degree of diagonal dominance, and it allows us to estimate

$$\|A_i\|_\infty \leq 1 + d^{-1}, \quad \|A_i^{-1}\|_\infty \leq \frac{1}{1 - d^{-1}}, \quad \text{and} \quad \kappa_\infty(A_i) \leq \frac{d+1}{d-1}.$$

Let u denote the unit roundoff error on the machine, and following Higham [21], we define

$$\gamma_j = \frac{ju}{1 - ju}, \tag{12.21}$$

when $ju < 1$. If A is any matrix, then $B = |A|$ is the matrix given by $b_{ij} = |a_{ij}|$. If A, B are matrices of the same dimension, then we write $A \leq B$, if $a_{ij} \leq b_{ij}$ for all i and j.

If A is a banded matrix with bandwidth $2k + 1$, which is diagonally dominant by rows, and if $Ax = f$ is solved using Gaussian elimination, then the computed solution $\hat{x}$ satisfies

$$(A + \Delta A)\hat{x} = f, \quad |\Delta A| \leq \gamma_{3k+2}|\hat{L}||\hat{U}|,$$

where $\hat{L}$ and $\hat{U}$ are the computed LU factors.

Now, how large is $\|\Delta A\|_\infty$? If A is any n by n matrix and if $A = LU$ is the exact LU-factorization, then

$$|L||U| = |AU^{-1}||U| \leq |A||U^{-1}||U|.$$

If U is diagonally dominant ($d = 1$), then by Lemma 8.8 [21]

$$\||U^{-1}||U|\|_\infty \leq (2n - 1). \tag{12.22}$$

This estimate is tight, as the following example shows.

Example 12.4.1 *Let U be the n by n bidiagonal Toeplitz matrix which has 1 on the main diagonal and -1 on the first superdiagonal,*

$$U = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix}, \quad \textit{and} \quad U^{-1} = \begin{bmatrix} 1 & \dots & \dots & 1 \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}.$$

It is clear that U is diagonally dominant and

$$|U||U^{-1}| = \begin{bmatrix} 1 & 2 & \dots & 2 \\ & \ddots & \ddots & \vdots \\ & & \ddots & 2 \\ & & & 1 \end{bmatrix},$$

which implies that $\||U||U^{-1}|\|_\infty = 2n-1$.

However, if A is strictly diagonally dominant by rows with degree $d > 1$ then a better estimate may exist. By Theorem 12.4.1 U is strictly diagonally dominant with degree no less d. Write $U = DV$, where D is the main diagonal of U, then

$$|U^{-1}||U| = |V^{-1}D^{-1}||DV| = |V^{-1}||V|.$$

Now, since $V = [v_{ij}]$ has $v_{ii} = 1$ and is strictly diagonally dominant with the same degree as U, we have

$$\|I - V\|_\infty \le d^{-1} < 1,$$

which allows us to write

$$V^{-1} = \sum_{j=0}^{\infty} (I - V)^j$$

and estimate

$$\|V^{-1}\|_\infty \le \frac{1}{1 - d^{-1}}$$

as well as $\|V\|_\infty \le 1 + d^{-1}$. It follows, that

$$\||U^{-1}||U|\|_\infty = \||V^{-1}||V|\|_\infty \le \frac{d+1}{d-1}. \tag{12.23}$$

The general bound (12.22) is superior when $d < d_0 = \frac{n}{n-1}$ and the specialized bound (12.23) is superior when $d > d_0$.

It is important to realize that neither (12.22) nor (12.23) need apply to the computed LU factorization, because while $\hat{L}\hat{U}$ is the exact LU factorization of the matrix $A + \Delta A$, this matrix need no be diagonally dominant! However, since $\hat{L} \to L$, and $\hat{U} \to U$ as the unit roundoff $u \to 0$, we see that if $d > 1$, then $A + \Delta A$ will be

diagonally dominant by rows with degree close to d, when u is sufficiently small, and then we may estimate

$$\|\Delta A\| \leq \gamma_{3k+2} \| |\hat{L}||\hat{U}| \|_\infty \lesssim \gamma_{3k+2} \frac{d+1}{d-1} \|A\|_\infty.$$

In the following we assume that we may estimate

$$\|\Delta A\| \leq \gamma_{3k+2} \frac{d+1}{d-1} \|A\|_\infty.$$

Now, what can be said about the solution $\hat{X}$ to the equation $AX = F$ where X and F have, say, m columns? We have

$$(A + \Delta A_j)\hat{x}_j = f_j, \quad |\Delta A_j| \leq \gamma_{3k+2} |\hat{L}||\hat{U}|, \quad j = 1, 2, \ldots m,$$

where the perturbations ΔA_j depend on j, but share a common bound which is independent of j. Now, if the unit roundoff error is sufficiently small, specifically if

$$\alpha = \gamma_{3k+2} \left(\frac{d+1}{d-1} \right)^2 < 1, \tag{12.24}$$

then $A^{-1}\Delta A_j$ and $\Delta A_j A^{-1}$ are both invertible and we may write

$$\hat{x}_j = \sum_{i=0}^{\infty} \left(-A^{-1}\Delta A_j \right)^i x_j = A^{-1} \sum_{i=0}^{\infty} \left(-(\Delta A_j)A^{-1} \right)^i f_j,$$

from which is follows immediately, that

$$|\hat{x}_j - x_j| \leq E_1 |x_j|, \quad E_1 = \sum_{i=1}^{\infty} (\gamma_{3k+2} |A^{-1}||\hat{L}||\hat{U}|)^i,$$
$$|A\hat{x}_j - f_j| \leq E_2 |f_j|, \quad E_2 = \sum_{i=1}^{\infty} (\gamma_{3k+2} |\hat{L}||\hat{U}||A^{-1}|)^i,$$

which implies

$$|\hat{X} - X| \leq E_1 |X|, \quad \text{and} \quad |A\hat{X} - F| \leq E_2 |F|.$$

The two operators, E_1 and E_2, share a common bound, namely

$$\|E_1\|_\infty \leq \frac{\alpha}{1-\alpha}, \quad \text{and} \quad \|E_2\|_\infty \leq \frac{\alpha}{1-\alpha},$$

where α is defined by (12.24). It follows that,

$$\|\hat{X} - X\|_\infty \leq \frac{\alpha}{1-\alpha}\|X\|_\infty, \quad \text{and} \quad \|A\hat{X} - F\|_\infty \leq \frac{\alpha}{1-\alpha}\|F\|_\infty. \tag{12.25}$$

Stage 1 Each matrix A_i has dimension μ and it is diagonally dominant by rows. The computed LU factorization satisfies

$$A_i + \Delta A_i = \hat{L}_i\hat{U}_i, \quad |\Delta A_i| \leq \gamma_{k+1}|\hat{L}_i||\hat{U}_i|.$$

where

$$\||\hat{L}_i||\hat{U}_i|\|_\infty \lesssim \frac{d+1}{d-1}\|A_i\|_\infty,$$

provided that the unit roundoff error u is sufficiently small. We have the same type of estimate for the computed UL factorizations.

Stage 2 In the truncated SPIKE algorithm, we do not compute the entire SPIKE matrix, but stop substituting, as soon as the truncated reduced system matrix has been computed. However, in order to estimate the error, it is convenient to consider the computation of the entire SPIKE matrix S.

By applying (12.25) repeatedly to the individual block rows we find

$$\|\hat{S} - S\|_\infty \leq \frac{2\alpha}{1-\alpha}\|S - I\|_\infty, \quad \|D\hat{S} - A\|_\infty \leq \frac{2\alpha}{1-\alpha}\|A - D\|_\infty$$

The extra factor of 2 is introduced, because we have to treat the superdiagonal and the subdiagonal spikes separately.

Similarly we find for the computation of the modified right-hand side that

$$\|\hat{g} - g\|_\infty \leq \frac{\alpha}{1-\alpha}\|g\|_\infty, \quad \text{and} \quad \|D\hat{g} - f\|_\infty \leq \frac{\alpha}{1-\alpha}\|f\|_\infty.$$

It is clear that since $\hat{T} - T$ is a submatrix of $\hat{S} - S$ we have

$$\|\hat{T} - T\|_\infty \leq \|\hat{S} - S\|_\infty \leq \frac{2\alpha}{1-\alpha}\|S - I\|_\infty \leq \frac{2\alpha}{1-\alpha}d^{-1}.$$

Stage 3 By Theorem 12.4.3 the truncated reduced system is a good approximation of the reduced system if d is not too close to 1 and if the partitions are not too small. By Theorem 12.4.1 the truncated reduced system is diagonally dominant by rows with

a degree no less than that of the original system. It consists of $p-1$ independent systems which are each of dimension $2k$. By Theorem 9.3 [21] it follows, that the computed solution $\hat{x}_{tr}$ of the computed truncated reduced system $\hat{T}x_{tr} = \hat{g}_r$ satisfies

$$(\hat{T} + \Delta\hat{T})\hat{x}_{tr} = \hat{g}_r, \quad |\Delta\hat{T}| \leq \gamma_{6k}|\hat{L}_t||\hat{U}_t|,$$

where $\hat{L}_t\hat{U}_t$ is the computed LU factorization of the computed truncated reduced system matrix $\hat{T}$. It follows that

$$\|\hat{x}_{tr} - x_{tr}\|_\infty \leq \frac{\beta}{1-\beta}\|x_{tr}\|_\infty \quad \text{and} \quad \|\hat{T}\hat{x}_{tr} - \hat{g}_r\| \leq \frac{\beta}{1-\beta}\|\hat{g}_r\|_\infty$$

provided the unit roundoff error is so small that

$$\beta = \gamma_{6k}\left(\frac{d+1}{d-1}\right)^2 < 1.$$

Stage 4 Adjusting the original right-hand side, i.e. computing

$$h_i = f_i - C_i x_{i-1,tr} - B_i x_{i+1,tr}$$

introduces a small forward error. Notice that C_i affects only the top of f_i and B_i affects only the bottom of f_i. The componentwise relative forward error satisfies

$$|\hat{h}_i - h_i| \leq \gamma_{k+1}\left(|f_i| + |C_i||x_{i-1,tr}| + |B_i||x_{i+1,tr}|\right),$$

regardless of the order in which the scalar product is evaluated. This is an overestimate, which does not take into account that the central components of f_i are not changed at all. The solution of the final set of linear equations is identical to stage 2 and generates a norm-wise relative residual of at most $\frac{\alpha}{1-\alpha}$, as well as a norm-wise relative forward error of at most $\frac{\alpha}{1-\alpha}$, cf. (12.25).

In short, if d is not too close to 1 and if the partitions are not too small, then the errors at every stage of the algorithm are small. We found that the simplest way to evaluate the overall error, is to calculate the residual and estimate

$$\|x - y\|_\infty \leq \|A^{-1}\|_\infty\|f - Ay\|_\infty \leq \frac{1}{1-d^{-1}}\|f - Ay\|_\infty,$$

which turned out to be fairly effective as long as d is not to close to 1.

12.5 Numerical experiments

We ran experiments to verify the main results as well as compare the accuracy of the truncated SPIKE algorithm with the algorithm implemented in ScaLAPACK.

12.5.1 The matrices S, R, and T

We wish to verify that the degree of diagonal dominance of the SPIKE matrix S is no less than that of the original matrix A. Towards this goal, we select two sequences of matrices with $(n, k_u, k_l) = (10^6, 5, 5)$:

$$A_{ij}^{(k)} = \begin{cases} 1 + 0.01k & \text{for } i = j \\ -0.1 & \text{for } 0 < |i-j| \leq 5, \\ 0 & \text{otherwise} \end{cases} \qquad B_{ij}^{(k)} = \begin{cases} 1 + 0.01k & \text{for } i = j \\ 0.1 & \text{for } 0 < |i-j| \leq 5, \\ 0 & \text{otherwise} \end{cases} \tag{12.26}$$

for $k = 1, 2, \ldots 100$. Selecting $p = 8$ partitions and a uniform block size of 125,000, we explicitly compute the entire SPIKE matrix S and the excess $\|S - I\|_\infty$ for each of these 200 matrices, from which we determine the degree of diagonal dominance as $d(S) = 1/\|S - I\|_\infty$. Our results are displayed in Figure 12.1. We find that not only is the degree of diagonal dominance preserved, i.e. $d(S) \geq d(A)$, but there can be a substantial increase in the degree of diagonal dominance as well.

We extract the truncated reduced system matrix T from each of the 200 matrices and compute the condition number in the infinity norm, by explicitly inverting T and calculating $\|T^{-1}\|_\infty$. We then plot the condition number of T as a function of the degree of diagonal dominance of A. The results are displayed in Figure 12.2. The theoretical upper bound is given by $\frac{d+1}{d-1}$, where $d = d(A)$ is the degree of diagonal dominance of A. We find that the truncated reduced system is even better conditioned than expected.

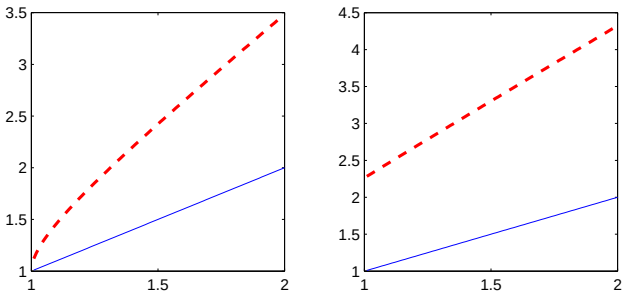

Figure 12.1. The degree of diagonal dominance for the matrix $S^{(k)}$ as a function of the degree of diagonal dominance of the original matrices: $A^{(k)}$ (left), and $B^{(k)}$ (right). The matrices are defined by equation (12.26). The red dotted line is the experimental result and the solid blue line is the theoretical lower bound.

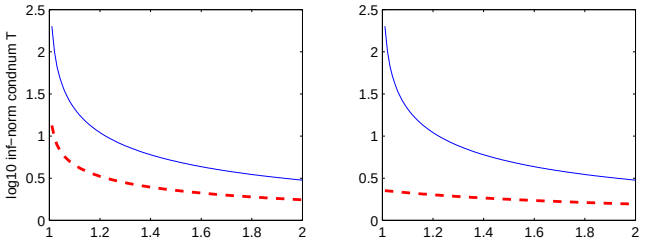

Figure 12.2. The condition number of the truncated reduced system as a function of the degree of diagonal dominance of the original system matrices: $A^{(k)}$ (left), and $B^{(k)}$ (right). The matrices are defined by equation (12.26). The dotted red line is the experimental result and the solid blue line is the theoretical upper bound.

12.5.2 The truncation error

Next, we want to investigate the size of the truncation error as a function of the degree of diagonal dominance of the original matrix A and the number p of partitions. We select a tridiagonal Toeplitz matrix with $n = 500,000$ and 1.01 on the main diagonal and 0.5 for the off-diagonal elements. We choose $p = 500j$, for $j = 1, 2, \ldots 10$ and compute the truncation error explicitly. The theoretical upper bound is given by d^{-q}, where $d = 1.01$ and $q = \lfloor (5e5/p \rfloor$. The results are displayed in Figure 12.3. The truncation error is much smaller than the theoretical upper bound and it is smaller than the unit roundoff error, $u = 2^{-53} \approx 1.1\text{e-}16$, as long as $p \leq 2000$.

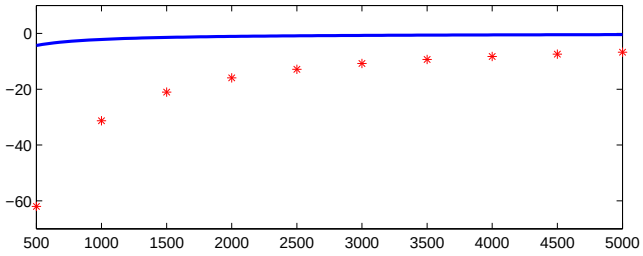

Figure 12.3. The infinity norm of the truncation error as a function of the number of partitions. The solid blue line is the theoretical upper bound, while red dots are experimental results. The matrix has degree of diagonal dominance $d = 1.01$ and is tridiagonal.

12.5.3 The roundoff errors

To verify the bounds presented in Section 12.4.3, we construct matrices which were diagonally dominant by different degrees and run them though our implementation of the truncated SPIKE algorithm. The matrices all have $(n, k_l, k_u) = (1e6, 10, 10)$, with every diagonal entry equal to 1. The nonzero, off-diagonal entries are positive and constant for each matrix, such that the degree of diagonal dominance ranges from 1.1

for the first matrix to 2.0 for the last matrix, with steps of 0.1. The right-hand side is generated from the solution, which is selected as $x = (1, 1, \ldots, 1)^T$. Our results are listed as Table 12.1, and Table 12.2. The bounds are computed as follows:

1. The modified right-hand side,
$$\|D\hat{g} - f\|_\infty \leq \frac{\alpha}{1-\alpha}\|f\|_\infty.$$
2. The SPIKE matrix,
$$\|D\hat{S} - A\|_\infty \leq 2\frac{\alpha}{1-\alpha}d^{-1}.$$
3. The computed truncated reduced system,
$$\|\hat{T}\hat{x}_{tr} - \hat{g}_r\| \leq \gamma_{6k}\frac{d+1}{d-1}\|\hat{x}_{tr}\|_\infty.$$
4. The overall error,
$$\|\hat{x} - x\|_\infty \leq \frac{1}{1-d^{-1}}\|A\hat{x} - f\|_\infty.$$

We see that the modified right-hand side g is computed with a small residual and that the bound becomes increasingly accurate as d becomes larger. The SPIKE matrix is computed with a very small residual compared to the bound which is between 10^3 and 10^5 times too large. The computed reduced system is solved with a very small residual and the bound is between 10^2 and 10^3 times larger. Finally, we see that using the residual to estimate the error is very reliable, leading to estimates that are accurate within one order of magnitude.

12.5.4 Comparisons with ScaLAPACK

We begin by comparing the errors in the truncated SPIKE algorithm to ScaLAPACK (PDDBTRF/PDDBTRS) for four different matrices with

$$(n, k_l, k_u) \in \{(2.0 \cdot 10^4, 10, 10), (10^5, 10, 10), (10^5, 50, 50), (10^6, 10, 10)\}.$$

Every diagonal entry is 1 and all the other entries within the band are 10^{-2}. The right hand side is constructed using an exact solution of $(1, 2, \ldots, n)^T$. The number

Table 12.1
A comparison of certain measurable quantities and their bounds for 10 different matrices distinguished by their degree of diagonal dominance

d	α	$\|D\hat{g} - f\|_\infty$		$\|D\hat{S} - A\|_\infty$	
		measured	bound	measured	bound
1.1	1.57e-12	9.58e-16	2.99e-12	6.94e-17	2.85e-12
1.2	4.30e-13	1.25e-15	7.88e-13	5.90e-17	7.16e-13
1.3	2.09e-13	1.57e-15	3.69e-13	7.05e-17	3.21e-13
1.4	1.28e-13	1.51e-15	2.19e-13	6.94e-17	1.83e-13
1.5	8.88e-14	1.64e-15	1.48e-13	5.11e-17	1.18e-13
1.6	6.67e-14	9.78e-16	1.08e-13	5.55e-17	8.34e-14
1.7	5.29e-14	1.51e-15	8.39e-14	2.93e-17	6.22e-14
1.8	4.35e-14	1.47e-15	6.77e-14	4.47e-17	4.84e-14
1.9	3.69e-14	1.75e-15	5.63e-14	4.27e-17	3.88e-14
2.0	3.20e-14	1.30e-15	4.80e-14	4.16e-17	3.20e-14

Table 12.2
A comparison of certain measurable quantities and their bounds for 10 different matrices distinguished by their degree of diagonal dominance

d	$\|\hat{T}\hat{x}_{tr} - g_r\|_\infty$		$\|A\hat{x} - f\|_\infty$	$\|\hat{x} - x\|_\infty$	
	measured	bound	measured	measured	bound
1.1	7.77e-16	1.40e-13	8.88e-16	8.88e-16	9.77e-15
1.2	7.77e-16	7.33e-14	1.33e-15	1.11e-15	7.99e-15
1.3	5.55e-16	5.11e-14	1.55e-15	1.33e-15	6.74e-15
1.4	8.88e-16	4.00e-14	1.55e-15	1.55e-15	5.44e-15
1.5	4.44e-16	3.33e-14	1.55e-15	1.55e-15	4.66e-15
1.6	7.77e-16	2.89e-14	8.88e-16	8.88e-16	2.37e-15
1.7	5.55e-16	2.57e-14	1.55e-15	1.22e-15	3.77e-15
1.8	5.55e-16	2.33e-14	1.55e-15	1.55e-15	3.50e-15
1.9	6.66e-16	2.15e-14	1.78e-15	1.55e-15	3.75e-15
2.0	5.55e-16	2.00e-14	1.33e-15	1.33e-15	2.66e-15

of partitions are 2, 4, 8, 16, 24, 32, 48, 64, 128, and 256. The calculations are carried out in IEEE double precision arithmetic, and we measure the 2-norm of the absolute error. Our results are displayed in Table 12.3. In our experiments ScaLAPACK yields slightly better results than the truncated SPIKE algorithm, but the difference between the two algorithms decrease, as the problems become larger. We would like to draw attention to the case of $p = 256$. In this case ScaLAPACK cannot be applied to the first matrix where $n = 20,000$, because the matrix is too small and the bandwidth is large compared to the number of partitions. The truncated SPIKE algorithm has a large error for the first and the third matrix. This is due to the fact that the infinity norm of the truncation error is very large: for the first matrix it is $1.62 \cdot 10^{-12}$, while for the third matrix it is $1.52 \cdot 10^{-7}$. In all other cases the infinity norm of the truncation error is either less than the machine ϵ or much smaller than the unit round off error u. The experiments with $p = 256$ emphasize the fact that the truncated SPIKE algorithm should not be applied to problems where the partitions are either too small or where the diagonal blocks are not diagonally dominant. The first matrix is diagonally dominant with degree $d = 5$, and for $p = 256$ the dimension of the smallest partition is 78. In this case Theorem 12.4.2 gives an upper bound for the infinity norm of the truncation error of $5^{-7} \approx 1.28 \cdot 10^{-5}$. In other words, we know in advance that the result might not be accurate. Theorem 12.4.2 does not apply to the third matrix, for which $d = 1$.

We consider nine matrices that are diagonally dominant in Matrix Market. They are all quite small, with dimensions no larger than 5000. Extracting narrow banded matrices from these benchmarks by choosing $k = \lceil 0.01n \rceil$, we run these examples through LAPACK (DGBTRF/DGBTRS), ScaLAPACK (PDDBTRF/PDDBTRS), our own implementation of the truncated SPIKE algorithm, as well as the SPIKE package itself (TU0). The matrices are scaled such that the main diagonals are 1 and the right-hand side is generated from the solution $x = (1, 1, \ldots, 1)^T$. We measured the 2-norm of the absolute error. Our results are listed in Table 12.4 and Table 12.5. We find no substantial difference in the accuracy of the four different routines.

Table 12.3
The 2-norm of the absolute error for ScaLAPACK (Sca) (PDDBTRF and PDDBTRS) and the truncated SPIKE (T.S.) algorithm for four different banded matrices and different numbers of partitions. The results from LAPACK (DGBTRF/DGBTRS) are listed at the bottom of the table.

	(n, k_l, k_u)							
	$(2e4, 10, 10)$		$(1e5, 10, 10)$		$(1e5, 50, 50)$		$(1e6, 10, 10)$	
p	Sca	T.S	Sca	T.S	Sca	T.S	Sca	T.S
2	4.98e-10	5.02e-10	5.33e-9	5.34e-9	1.33e-8	1.33e-8	2.10e-7	2.10e-7
4	4.98e-10	5.02e-10	5.33e-9	5.33e-9	1.33e-8	1.33e-8	2.10e-7	2.10e-7
8	4.97e-10	5.02e-10	5.32e-9	5.33e-9	1.33e-8	1.33e-8	2.10e-7	2.10e-7
12	4.97e-10	5.01e-10	5.32e-9	5.33e-9	1.33e-8	1.34e-8	2.10e-7	2.10e-7
16	4.97e-10	5.02e-10	5.32e-9	5.33e-9	1.33e-8	1.33e-8	2.10e-7	2.10e-7
24	4.95e-10	5.00e-10	5.32e-9	5.33e-9	1.33e-8	1.34e-8	2.10e-7	2.10e-7
32	4.95e-10	5.02e-10	5.32e-9	5.33e-9	1.32e-8	1.33e-8	2.10e-7	2.10e-7
48	4.90e-10	4.98e-10	5.31e-9	5.32e-9	1.33e-8	1.33e-8	2.10e-7	2.10e-7
64	4.88e-10	4.95e-10	5.30e-9	5.32e-9	1.32e-8	1.33e-8	2.10e-7	2.10e-7
128	4.81e-10	4.88e-10	5.28e-9	5.30e-9	1.33e-8	1.34e-8	2.10e-7	2.10e-7
256	N/A	1.43e-7	5.23e-9	5.26e-9	1.33e-8	7.64e-2	2.10e-7	2.10e-7
LA	4.99e-10		5.33e-9		1.33e-8		2.10e-7	

Table 12.4
The 2-norm of the absolute error for nine different matrices from Matrix Market. The results are given for LAPACK (dgbtrf/dgbtrs) and ScaLAPACK (PDDBTRF/PDDBTRS). The results are given for 2, 4, and 8 partitions.

matrix	n	LA	ScaLAPACK		
			2	4	8
dwb512	512	3.27e-15	3.14e-15	3.14e-15	3.14e-15
gr_30_30	900	0.00e+00	0.00e+00	1.57e-16	2.94e-16
jpwh_991	991	1.37e-15	2.04e-15	2.03e-15	2.01e-15
nos6	675	0.00e+00	3.05e-15	3.07e-15	3.10e-15
orsirr_1	1030	4.40e-15	4.44e-15	4.42e-15	4.36e-15
orsirr_2	886	4.11e-15	4.10e-15	4.11e-15	4.12e-15
orsreg_1	2205	7.08e-15	7.12e-15	7.30e-15	6.93e-15
sherman3	5005	1.92e-12	1.99e-12	1.99e-12	1.99e-12
sherman4	1104	2.53e-15	2.59e-15	2.58e-15	2.58e-15

Table 12.5
The 2-norm of the absolute error for nine different matrices from Matrix Market. The results are given for our implementation (T.S) of the truncated SPIKE algorithm, as well as the current implementation of the SPIKE package (TU0). The results are given for 2, 4, and 8 partitions.

matrix	T.S			TU0		
	2	4	8	2	4	8
dwb512	3.30e-15	3.30e-15	3.30e-15	3.04e-15	3.09e-15	3.11e-15
gr_30_30	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.11e-16	5.09e-16
jpwh_991	2.00e-15	1.97e-15	1.95e-15	2.03e-15	2.06e-15	2.01e-15
nos6	0.00e+00	0.00e+00	0.00e+00	3.03e-15	3.12e-15	3.01e-15
orsirr_1	4.39e-15	4.38e-15	4.31e-15	4.40e-15	4.26e-15	4.15e-15
orsirr_2	4.07e-15	4.10e-15	4.10e-15	4.16e-15	3.94e-15	4.06e-15
orsreg_1	7.16e-15	7.16e-15	7.00e-15	6.49e-15	6.79e-15	6.18e-15
sherman3	1.99e-12	1.99e-12	1.99e-12	1.98e-12	1.98e-12	1.98e-12
sherman4	2.49e-15	2.50e-15	2.51e-15	2.44e-15	2.41e-15	2.48e-15

12.6 Conclusions for SPIKE

We have discussed the explicit and the truncated SPIKE algorithm. Our main contribution is the theoretical analysis of the truncated SPIKE algorithm for systems which are strictly diagonally dominant by rows. We showed that the SPIKE matrix, the reduced system matrix, and the truncated reduced system matrix are all strictly diagonally dominant by rows with a degree no less than the original matrix. We established a tight upper bound on the decay rate of the spikes and the truncation error. This settled a question which has been open since the algorithm was introduced. The key to understanding the analysis is the fact that Gaussian elimination without pivoting does not decrease the degree of diagonal dominance. We offered a partial explanation as to why it is difficult to obtain tight estimates on the decay rate and the truncation error in the general case where the main block diagonal is a well conditioned matrix. We also did an elementary error analysis of the truncated SPIKE algorithm, which revealed that the error committed at each stage of the algorithm is very small. Our numerical experiments showed that our analysis is pessimistic, and we found no substantial difference in the accuracy of the solutions obtained via the truncated SPIKE algorithm and the corresponding algorithm from ScaLAPACK.

12.7 Extensions of the SPIKE algorithm

The truncated SPIKE algorithm is attractive because it is both fast and accurate and it scales very well on parallel machines. There is very little communication between the processors, especially during the solve phase where there are only two exchanges of data between neighboring processors. In this section we consider the solution of narrow banded linear systems from a theoretical point of view. We show that it is possible to modify the truncated SPIKE algorithm to the point where the reduced system is eliminate and we show that it is possible to solve a general narrow banded linear system on a parallel machine with a single one to all communication

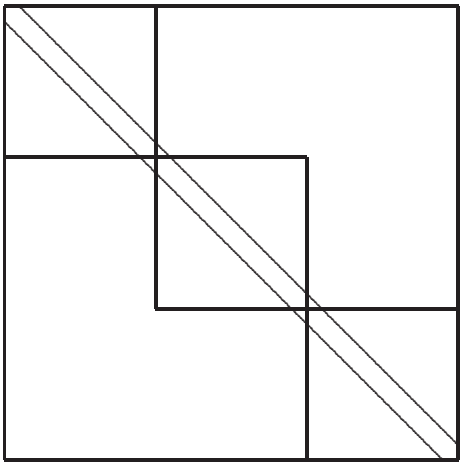

Figure 12.4. The original, non-overlapping partitioning of the narrow banded matrix.

during the solve phase. We will rely heavily on the notation and the results which we have established in this chapter.

12.7.1 Overlapping partition method

The idea of using overlapping partitions to solve a linear system originated in the masters thesis of G. Lou [35] and has been extended by Naumov and Sameh [38]. Larriba-Pey [29, 33] adapted the basic idea and introduced the overlapping partition method (OPM) for systems which are strictly diagonally dominant. In this section we analyze the error for this method using the theory we derived for the truncated SPIKE algorithms.

Our interest in the OPM is primarily theoretical and our main result, Theorem 12.7.1, is essentially a statement about the nature of such linear systems.

Let A be a narrow banded matrix with half bandwidth k. In the SPIKE algorithm the original matrix is partitioned by rows, and the case of $p = 3$ partitions is illustrated in Figure 12.4.

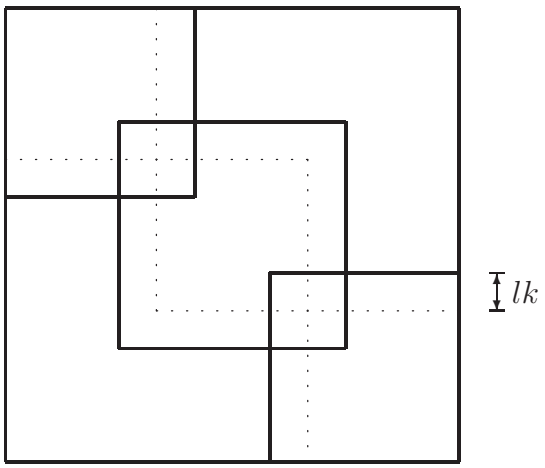

Figure 12.5. The overlapping partitioning of the matrix, with the original partitioning drawn using dotted lines. Each partition has been extended by lk in all possible directions.

Now, the elements of the second, central partition, satisfy

$$x_2 = g_2 - W_2 x_3^{(t)} - V_2 x_1^{(b)}, \tag{12.27}$$

where V_2 and W_2 are the corresponding spikes. We have already derived a tight upper bound on the decay rate of the spikes. The decay implies that if the partitions are sufficiently large, then the central components of x_2 are essentially equal to the corresponding components of g_2. This suggests that we should create a new partitioning by extending the original partitions in every direction, allowing them to overlap, see Figure 12.5. The non-overlapping partitions are extended by lk in every possible direction, where l is a nonnegative integer and k is the half bandwidth.

The overlapping partition method (OPM) is stated as Algorithms 29 and 30. For simplicity we continue to assume that the number of partitions, p, divides the dimension n of the matrix A, i.e. $\mu = \frac{n}{p}$ is an integer.

Theorem 12.7.1 *Let A be a narrow banded matrix with half bandwidth k. If A is diagonally dominant with degree d, and if y is the approximate solution obtained by*

1: LU factorization: $L_1U_1 = A(1 : \mu + lk, 1 : \mu + lk)$

2: **for** $i = 2 : p - 1$ **do**

3: $\quad s_i := (i-1)\mu + 1 - lk$, $e_i := s_i + \mu + 2lk = i\mu + lk$

4: $\quad$ LU factorization: $L_iU_i = A(s_i : e_i, s_i : e_i)$.

5: **end for**

6: LU factorization: $L_pU_p = A((p-1)\mu + 1 - lk : n, (p-1)\mu + 1 - lk : n)$.

Algorithm 29: Factorization phase of the OPM

1: Solve $L_1U_1y_1 = f(1 : \mu + k)$

2: **for** $i = 2 : p - 1$ **do**

3: $\quad s_i := (i-1)\mu + 1 - lk$, $e_i := s_i + \mu + 2lk = i\mu + lk$

4: $\quad$ Solve $L_iU_iy_i = f(s_i : e_i)$

5: **end for**

6: Solve $L_pU_py_p = f((p-1)\mu + 1 - lk : n)$

7: Assemble the approximate solution y from the y_i,

$$y = (y_1(1 : \mu)^T, y_2(lk + 1, \mu + lk + 1)^T, \ldots, y_p(lk + 1, \mu + lk + 1)^T)^T$$

Algorithm 30: Solve phase of the OPM

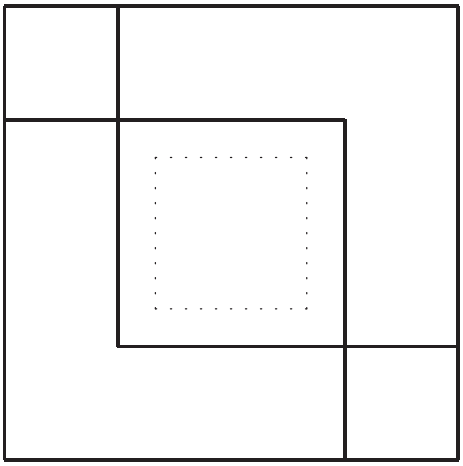

Figure 12.6. The auxiliary partitioning needed for the proof of Theorem 1. We are interested in the variables corresponding to the original diagonal block which is drawn using dotted lines.

applying the overlapping partitioning method to $Ax = f$ *with overlap* lk, $l = 0, 1, 2 \ldots$, *then*

$$\|x - y\|_\infty \leq 2d^{-(l+1)}\|x\|_\infty. \tag{12.28}$$

The theorem states that the overlapping partitioning method solves $Ax = f$ with a norm-wise relative forward error, which decreases as the overlap increases.

Proof The special case of $p = 3$ partitions contains all the details of the general case which is why we only treat this case. In addition, we focus on the second, central partition, which is drawn using dotted lines on Figure 12.6. In this figure we have also drawn an auxiliary, non-overlapping partitioning, using solid lines.

We have extended the central dotted partition by lk in all four directions and we have shrunk the two neighboring partitions just enough to create three non-overlapping partitions. Now let W_2' and V_2' be the central spikes generated by applying the SPIKE algorithm to the system in the auxiliary partitioning. We have

$$x_2' = g_2' - W_2'(x_3')^{(t)} - V_2'(x_1')^{(b)}. \tag{12.29}$$

Now let x_2 be the vector corresponding to the central dotted partition. Then, since x_2' was formed by extending x_2 with lk variables in either direction, it follows that

$$\|x_2 - y_2\|_\infty \leq 2d^{-(l+1)}\|x\|_\infty, \tag{12.30}$$

because $(x_3')^{(t)}$, and $(x_1')^{(b)}$ are also sub-vectors of the true solution x. We have used Theorem 12.4.2 to estimate the infinity norm of the central submatrices of V_2', and W_2', which correspond to x_2. In order to complete the case of $p = 3$ we note that

$$\|x_i - y_i\|_\infty \leq d^{-(l+1)}\|x\|_\infty \tag{12.31}$$

for $i = 1$ and $i = 3$, because only a single spike is involved in each of these two cases. It follows, that

$$\|x - y\|_\infty \leq 2d^{-(l+1)}\|x\|_\infty, \tag{12.32}$$

and the proof is complete. ■

It is possible to improve on this error bound slightly. We used our tight estimate for the decay rate of the spikes, but we suspect that the slowest decay rate is observed only for matrices which are also triangular, and we know this to be true for tridiagonal matrices. In short, while it might be possible to improve on the constant in Theorem 12.7.1, the asymptotic dependence on d and l is tight for the class of narrow banded matrices which are diagonally dominant by rows.

Now suppose that we are interested only in a particular component x_i of the entire solution x of $Ax = f$, but that we cannot afford to solve the entire system. In view of the proof of Theorem 12.7.1, it suffices to extract and solve

$$A(i - lk : i + lk, i - lk : i + lk)y = f(i - lk : i + lk) \tag{12.33}$$

for y, because y_{lk+1} will satisfy

$$|x_i - y_{lk+1}| \leq 2d^{-(l+1)}\|x\|_\infty \tag{12.34}$$

We see that if x_i and x_j are two distinct components of x, then they are essentially independent of each other if $|i-j|$ is sufficiently large. Our previous comments merely makes this notion precise.

The greatest difference between the OPM method and the truncated SPIKE algorithm is that the OPM does not require the solution of a reduced system. Both algorithms are suitable for a parallel computer. For the OPM it is natural to use a linear array of p processors, while for the truncated SPIKE algorithm a ring of $p-1$ processors is the natural choice, and in both cases we begin by splitting the data in disjoint block rows across the processors.

During the factorization phase the SPIKE algorithm does twice as much work as the OPM, but only a single k by k block is moved from each processor to its predecessor in the ring, while the OPM requires the exchange of lk rows of the matrix between each pair of neighboring processors.

During the solve phase, the SPIKE algorithm moves k rows of the modified right hand side from each processor to its immediate predecessor in the ring in order to set up the truncated reduced system. After the truncated reduced system has been solved, k rows of the solution are moved from each processor to its successor. For the OPM each pair of neighboring processors exchange lk rows of the right hand side matrix.

In summary, the SPIKE algorithm performs more arithmetic than the OPM during the factorization phase, but less data is moved during both phases.

12.7.2 Solving linear systems with a single communication

The SPIKE algorithms, the OPM and the solves implemented in SCALAPACK split naturally into two phases, a factorization phase where the coefficient matrix is analyzed and a solve phase where the right hand side is accessed and the solution is returned. The processors exchange multiple messages during both phases of all the algorithms.

We now explain why it is possible to solve a general narrow banded linear system using a single one to all communication between the processors during the solve phase.

If the linear system is sufficiently well conditioned, then this can be reduced to a single exchange between neighboring processors.

As in the SPIKE algorithms we split the linear system by block rows across the processors and we use D to denote the main block diagonal.

Let $f \in \mathbb{R}^n$, then by the SPIKE algorithm

$$Dx = f - (A - D)Qx_r \tag{12.35}$$

where Q is the natural injection of the reduced system space into $\mathbb{R}^n$. Now, if $x_r = 0$, then

$$Dx = f, \tag{12.36}$$

and it follows that the local segments of x can be computed with no communication, because $A_i x_i = f_i$. As a result of this observation we define the vectorspace G as follows,

$$G = \{f \in \mathbb{R}^n \ : \ PD^{-1}f = 0\} = \mathrm{Ker}(PD^{-1}), \tag{12.37}$$

where P is the orthogonal projection onto the reduced system space. This is the vectorspace of "good" right hand sides f for which the solution of $Ax = f$ does not require communication. We also define

$$B = G^{\perp} = \mathrm{Ran}\,(D^{-T}P), \tag{12.38}$$

which allows us to write

$$R^n = G \oplus B. \tag{12.39}$$

The vector space B is the set of "bad" right hand sides f, for which the solution of $Ax = f$ may require communication.

By assumption D is nonsingular, which implies that the dimension of B satisfies

$$\dim B = 2(p-1)k. \tag{12.40}$$

In general both k and p are much smaller than n, so the dimension of B is likely to be small. Now let $m = \dim B$. It is clear that B is spanned by the columns of the n by m matrix

$$E = \mathrm{diag}\{E_1, E_2, \ldots, E_p\}, \tag{12.41}$$

where

$$E_1 = A_1^{-T} \begin{bmatrix} 0 \\ 0 \\ I_k \end{bmatrix}, \quad E_i = A_i^{-T} \begin{bmatrix} I_k & 0 \\ 0 & 0 \\ 0 & I_k \end{bmatrix}, \quad 2 < i < p, \quad \text{and} \quad E_p = A_p^{-T} \begin{bmatrix} I_k \\ 0 \\ 0 \end{bmatrix} \tag{12.42}$$

and I_k denotes the k by k identity matrix. Let $E_i = Q_i R_i$ be the QR factorization of E_i, then the columns of

$$Q = \text{diag}\{Q_1, Q_2, \ldots, Q_p\} \tag{12.43}$$

form an orthonormal basis for B.

Now assume that we have solved the linear system

$$AY = Q \tag{12.44}$$

and that the solution Y has been split by block rows across the processors such that the i'th segment Y_i is stored in the local memory of processor i.

We are now given a new right hand side f and asked to solve $Ax = f$. First the right hand side is split across the processors. Then we compute the orthogonal projection f'' of f onto B,

$$f'' = QQ^T f. \tag{12.45}$$

This can be done locally with no communication, because Q is block diagonal, which implies that the local segments of f'' are given by

$$f_i'' = (QQ^T f)_i = Q_i(Q_i^T f_i). \tag{12.46}$$

We can now compute the orthogonal projection of f onto G, because

$$f' = (I - QQ^T)f = f - f'', \tag{12.47}$$

and the i'th processor merely computes

$$f_i' = f_i - f_i'', \tag{12.48}$$

which does not require any communication either. Now we solve $Ax' = f'$, and since $f' \in G$ this does not require any communication. The i'th processor merely solves

$$A_i x'_i = f'_i \tag{12.49}$$

for the local segment of x'. It remains to solve $Ax'' = f''$. However, we have already done most of the work, before we were even given f, because

$$x'' = A^{-1} f'' = A^{-1} Q Q^T f = (A^{-1} Q) Q^T f = Y(Q^T f). \tag{12.50}$$

We see that the local segments of x'' are given by

$$x''_i = Y_i (Q^T f), \tag{12.51}$$

where Y_i is the local segment of Y. Processor i already has access to its own segment of $(Q^T f)$, but it the entire vector $Q^T f \in \mathbb{R}^m$ is needed in order to complete the calculation of x''_i. At this point it is necessary to have a single one to all broadcast of the local segments of $Q^T f$. This is not a cheap operation, but it is necessary in the general case. However, there are cases were it is possible to reduce this demand to a single exchange of segments between neighboring processors.

Finally, processor i computes the local segment x_i of the solution of $Ax = f$, by simple addition, i.e.

$$x_i = x'_i + x''_i, \tag{12.52}$$

which completes the calculation.

We have summarized this discussion in Algorithms 31, 32, 33.

1: Processor i solves $A_i^T Z_i = E_i$ using any suitable algorithm.
2: Processor i computes $Q_i R_i = Z_i$, QR factorization.

Algorithm 31: Mincom: Assembling the auxiliary equations

Algorithm 33 contains a single one to all broadcast which will make the vector $Q^T f \in \mathbb{R}^m$ available to every processor. This is an expensive operation, because every processor must communicate data to every other processor. However, there are many

1: The linear system $AY = Q$ is solved using any suitable algorithm.

2: The i'th segment of Y_i is stored in the local memory of processor i.

Algorithm 32: Mincom: Solution of the auxiliary equations

1: Processor i computes $f_i'' = Q_i Q_i^T f_i$.

2: Processor i computes $f_i' = f_i - f_i''$.

3: Processor i solves $A_i x_i' = f_i'$.

4: In a single one to all broadcast, processor i sends $Q_i^T f_i$ to the other processors.

5: Processor i computes $x_i'' = Y_i(Q^T f)$.

6: Processor i computes $x_i = x_i' + x_i''$.

Algorithm 33: Mincom: Solution of a $Ax = f$

cases where this operation can be reduced to a single exchange of information between neighboring processors. The key to understanding this is to realize that the matrix $Y = A^{-1}Q$ has a very special structure. Let q be any column of Q. By definition, q consists mostly of zeros, and the nonzero entries are concentrated in a single segment corresponding to a particular processor, say, processor i. If A is a sufficiently well conditioned matrix, then it follows from the estimates developed by Demko, Moss, and Smith [9] that the entries of $y = A^{-1}q$ decay exponentially as we move away from the i'th segment and by the time we enter the segment corresponding to processor j, where $|i - j| > 1$, the entries may very well have decayed below the machine ϵ and they are effectively zero. This phenomenon is observed when A is well conditioned, and the partitions are large compared with the half bandwidth, i.e when $\mu/k \gg 1$.

13. Conclusion

13.1 Summary of contributions

We have considered three distinct, but related topics:

- The direct computation of a dominant eigenspace of a matrix X given implicitly as the solution of the Lyapunov equation, i.e.

$$AX + XA^T + BB^T = 0.$$

- The application of Krylov subspace methods to Lyapunov equations written in the Kronecker product form,

$$(I \otimes A + A \otimes I)\text{vec}(X) + \text{vec}(BB^T) = 0.$$

- The solution of general narrow banded linear systems, $Ax = f$.

Let A be an n by n negative definite matrix and let B be an n by p matrix. Let P be the solution of

$$AP + PA^T + BB^T = 0.$$

Also, let V be an n by k matrix with orthonormal columns. Hodel [24] approximated PV with Y, where Y is the solution of the tall Sylvester equation,

$$AY + Y(V^T A^T V) + BB^T V = 0.$$

Our main contribution is Theorem 8.2.1, which shows that if P has rank r, and if V has $k = r$ columns, then

$$\text{Ran}\, Y = \text{Ran}\, P,$$

unless Y is rank deficient, in which case

$$\text{Ran}\, Y \subset \text{Ran}\, P.$$

Our result is an extension of Hodel's analysis, which considered the case of $r = 1$. In practice, P has full rank $r = n$ and it is not practical to solve for Y. However, both the theoretical analysis and our experimental results suggest that the dominant eigenspaces of P can be computed accurately by choosing k sufficiently large.

We extended the analysis of the standard Arnoldi method [28,50,56] for Lyapunov equations, by showing how to construct a negative definite matrix A and a column vector B for which the method returns a predescribed residual history.

Hochbruck and Starke [23] showed that it is possible to apply Krylov subspace methods directly to the Lyapunov equation in the Kronecker product form. However, their approach requires $O(n^2)$ arithmetic operations and $O(n^2)$ words of storage. We have shown that if B is an n by p matrix, with $p \ll n$, then it is possible to reduce these requirements to $O(n)$. Our main result, Theorem 10.3.1, shows that it is possible to compute an orthonormal basis for the Krylov subspace $K_k(\tilde{A}, \tilde{b}) \subseteq \mathbb{R}^{n^2}$, using $O(npk + p^2k^3)$ words of storage and $O(np^2k^2 + p^3k^3)$ arithmetic operations. We used this and similar results to adapt GMRES, CG, BCG, and CGNR for standard linear systems to the Lyapunov equation in the Kronecker product form. Regrettably, our compact representation of the necessary vectors in $\mathbb{R}^{n^2}$ does not permit a general preconditioning strategy. However, preconditioning can be achieved by working with a representation of the form $\tilde{x} = \text{vec}(EF^T)$, where E and F are tall matrices with n rows. Unfortunately, it is necessary to compute tall singular value decompositions at regular intervals in order to keep the memory consumption down. In addition, it appears that the choice of preconditioners is extremely limited. A preconditioner $\tilde{M}$ must necessarily be an invertible map from $\mathbb{R}^{n^2}$ into $\mathbb{R}^{n^2}$, and representing $\tilde{M}$ explicitly would require at least n^2 words of memory. Further, every equation of the type

$$\tilde{M}\tilde{x} = \text{vec}(EF^T),$$

where E and F are tall matrices, must have a solution $\tilde{x}$, which admits an approximation of the form

$$\tilde{x} \approx \text{vec}(GH^T),$$

where G and H are tall matrices, and we must be able to compute G and H directly without forming $\tilde{x}$ explicitly. As a result, we investigated preconditioners of the type

$$\tilde{M} = I \otimes M + M \otimes I,$$

hoping to select M such that $\tilde{M}^{-1}$ could be considered a good approximation to $\tilde{A}^{-1}$, and such that solving

$$MX + XM + EF^T = 0$$

would be much faster than solving the original problem

$$AX + XA^T + BB^T = 0,$$

We found that out that if the n^2 by n^2 matrix

$$I_{n^2} - \tilde{M}^{-1}\tilde{A}$$

has rank strictly less than n, then surprisingly enough M must be equal to A (consequence of Theorem 11.3.3). We made the elementary observation (consequence of Theorem 11.3.2) that there exists a positive constant $C = C(n)$, such that

$$\frac{\|M - A\|_2}{\|M\|_2} \leq C(n)\|I_{n^2} - \tilde{M}^{-1}\tilde{A}\|_2,$$

which suggests that if $\tilde{M}$ is a good preconditioner for $\tilde{A}$ in the sense that

$$\|I_{n^2} - \tilde{M}^{-1}\tilde{A}\|_2 \ll 1$$

then is is likely that A is a good norm-wise approximation of M. Unfortunately we do not have a formula for $C(n)$.

Currently, we are only aware of two situations where an equation of the form

$$MX + XM + EF^T = 0$$

admits a good low rank approximation, namely when M is well conditioned and negative definite (the Arnoldi method) or when the spectrum of M is clustered around a few known points (the ADI family of methods). But if A is a good approximation

of M, then, by continuity, we can expect each of these methods to apply equally well to the original problem, eliminating the need for Krylov subspace methods and preconditioning in the first place.

Above all, simply treating the Lyapunov matrix equation as a standard linear system ignores the structure of the solution. We used the structure of the Lyapunov operator to reduce the storage and arithmetic requirements. Our investigation also revealed that the Arnoldi methods and its various extensions do not exploit the low rank phenomenon. Specifically, we exhibited an n by n negative definite matrix A for which the Arnoldi method did not converge until the n'th iteration, while the exact solution exhibited rapid eigendecay.

Almost every solver for Lyapunov matrix equations depends on the solution of standard linear systems. The ADI family of methods is a prominent example. Frequently, these systems are either narrow banded or admit narrow banded preconditioners. As a result, we spent time investigating the solution of narrow banded linear systems. We analyzed the SPIKE algorithms, with special emphasis on the truncated SPIKE algorithm, which applies to systems which are diagonally dominant by rows. We were able to settle several questions, which have been open since the algorithm was introduced. Specifically, we showed that the SPIKE matrix, the reduced system matrix, and the truncated reduced system matrix are all diagonally dominant with a degree no less than that of the original system. We estimate the decay rate of the spikes as well as the the truncation error. Our bounds are tight for matrices with a given half bandwidth and degree of diagonal dominance. The key to understanding all of these results is Lemma 12.4.1, which states that Gaussian elimination does not decrease the degree of diagonal dominance.

The analysis of the truncated SPIKE algorithm had immediate implications for the overlapping partition method. Let $x = (x_1, x_2, \ldots, x_n)^T$ be the solution of $Ax = f$, which is strictly diagonally dominant by rows, then the OPM can be used to explain why x_i is almost independent of x_j, provided that $|i - j|$ is large enough.

The explicit SPIKE algorithm and the truncated SPIKE algorithm, as well as the direct solvers implemented in SCALAPACK, can all be split into a preliminary factorization phase where the matrix is partitioned and factorized, and a solve phase where the systems are actually solved. We showed that it is possible to arrange the calculations such that the solve phase for a general narrow banded linear system can be carried out with a single one to all communication.

13.2 Future work

Our work on the approximate subspace iteration with Ritz acceleration as well as our adaption of the CGNR algorithm, were attempts to create methods which were parameter free. The approximate subspace iteration applies only to problems which are negative definite, and while the CGNR algorithm applies to any problem, the storage requirements are excessive in the worst case. Developing a parameter free method for general large and sparse Lyapunov equations is still an open question, which we would like to pursue.

We are intrigued by the possibility of reducing the interprocessor communication during the solve phase of the solution of a general narrow banded linear system. We expect this to be particularly relevant on a distributed memory machine. We have not done an error analysis of the algorithm that we have outlined, but it is clear that the rounding errors during the solution of the 'bad' systems, will necessarily pollute the solution of $Ax = f$ for an arbitrary right-hand side f. As a result we would only use our method as a preconditioner, and only when it is necessary to solve many systems with the same coefficient matrix, as in the ADI method.

LIST OF REFERENCES

LIST OF REFERENCES

[1] A. C. Antoulas. *Approximation of large-scale dynamical systems*. SIAM, first edition, 2005.

[2] A. C. Antoulas, D. C. Sorensen, and Y. Zhou. On the decay rate of Hankel singular values and related issues. *Linear Algebra and Applications*, 415(2-3):344–358, 2006.

[3] R. H. Bartels and G. W. Stewart. Solution of the matrix equation AX+XB=C. *Communication of the ACM*, 15(9):820–826, 1972.

[4] U. Baur and P. Benner. Factorized solution of the Lyapunov equation by using the hierarchical matrix arithmetic. *Computing*, 78(3):211–234, 2006.

[5] P. Benner. Factorized solution of the Sylvester equation with applications in control. In *Proceedings of the 16th Intl. Symp. on Mathematical Theory of Networks and Systems*, 2004.

[6] M. Berry and A. Sameh. Multiprocessor schemes for solving block tridiagonal linear systems. *Int. J. Supercomputing. Appl.*, 2(3):37–57, 1988.

[7] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Hierarchical matrices. Lecture notes no. 21, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2003.

[8] S. C. Chen, D. J. Kuck, and A. H. Sameh. Practical parallel band triangular solvers. *ACM Transactions on Mathematical Software*, 4:270–277, 1978.

[9] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Mathematics of Computation*, 43(168):491–499, 1984.

[10] J. J. Dongarra and A. H. Sameh. On some parallel banded system solvers. *Parallel computing*, 1:223–235, 1984.

[11] Mark Embree. How descriptive are GMRES convergence bounds. Technical research report, Oxford University Computing Laboratory, 1999.

[12] Nabil Gamti and Bernard Philippe. Comments on the GMRES convergence for preconditioned systems. In Jerzy Wasniewski Ivan Lirkov, Svetozar Margenov, editor, *Large scale scientific computing: 6th international conference, LSSC 2007, Sozopol, Bulgaria, June 5-9, 2007*, pages 41–51, New York, 2008. Springer.

[13] A. George and Kh. Ikramov. Gaussian elimination is stable for the inverse of a diagonally dominant matrix. *Math. Comp*, 73(246):653–657, 2003.

[14] Luc Giraud, Julien Langou, and Miroslav Rozložník. On the loss of orthogonality in the Gram-Schmidt orthogonalization process. Technical research report TR/PA/03/25, CERFACS, 2003.

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[16] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing*, 70(2):121–165, 2003.

[17] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš. Any non increasing convergence curve is possible for GMRES. *SIAM Journal of Matrix Analysis and Applications*, 17(3):465–469, 1996.

[18] S. Gugercin, D. C. Sorensen, and A.C. Antoulas. A modified low-rank Smith method for large-scale lyapunov equations. *Numerical Algorithms*, 32(1):27–55, 2003.

[19] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal of Numerical Analysis*, 2(3):303–323, 1982.

[20] M. R. Hestenes and E. L. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards, Section B*, 49:409–436, 1952.

[21] N. J. Higham. *Accuracy and stability of Numerical Algorithms*. SIAM, Philidelphia, PA, USA, second edition, 2002.

[22] N. J. Higham. *Functions of Matrices: Theory and computation*. SIAM, USA, first edition, 2008.

[23] M. Hochbruck and G. Starke. Preconditioned Krylov subspace methods for Lyapunov matrix equations. *SIAM J. Matrix. Anal. Appl.*, 16:156–171, 1995.

[24] A. S. Hodel. *Numerical Methods for the Solution of Large and Very Large, Sparse Lyapunov Equations*. PhD thesis, University of Illinois at Urbana-Champaign, 1989.

[25] A. S. Hodel, B. Tenison, and K. R. Poolla. Numerical solution of the Lyapunov equation by approximate power iteration. *Linear Algebra and its Applications*, 236:205–230, 1996.

[26] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, first edition, 1985.

[27] D. Y. Hu and L. Reichel. Krylov subspace methods for the Sylvester equation. *Linear Algebra Appl. 1992*, 172:283–313, 1992.

[28] I. Jaimoukha and E. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Matrix Anal*, 31:227–251, 1994.

[29] Angel Jorba, Josep-L. Larriba-Pey, and Juan J. Navarro. A proof for the accuracy of OPM. Technical research report 92-10, Departament de Matemàtica Aplicada I Universitat Politècnica de Catalunya, Barcelon, Spain, 1992.

[30] D. Kressner. Memory efficient Krylov subspace techniques for solving large-scale Lyapunov equations. In *IEEE International Conference on Computer Aided Control Systems*, pages 613–618, 2008.

[31] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49:33–53, 1952.

[32] Josep-L. Larriba-Pey and Juan J. Navarro Àngel Jorba. Spike algorithm with savings for strictly diagonal dominant tridiagonal systems. *Microprocessing and Microprogramming*, 39(2-5):125–128, 1993.

[33] Josep-L. Larriba-Pey, Angel Jorba, and Juan J. Navarro. OPM: a parallel method to solve banded systems of equations. Technical research report 92-09, Departament de Matemàtica Aplicada I Universitat Politècnica de Catalunya, Bercelon, Spain, 1992.

[34] D. H. Lawrie and A. H. Sameh. The computation and communication complexity of a parallel banded system solver. *ACM Transactions on Mathematical Software*, 10:185–195, 1984.

[35] Gang Lou. Parallel methods for solving linear systems via overlapping decomposition. Thesis (m.s.), University of Illinois at Urbana-Champaign, 1989.

[36] C. C. K. Mikkelsen. The decay rate of the solution to a tridiagonal system with a very special right hand side. Technical research report CSD TR #08-021, Department of Computer Science, Purdue University, West Lafayette, IN, USA, 2008.

[37] C. C. K. Mikkelsen and M. Manguoglu. Analysis of the truncated SPIKE algorithm. *SIAM Journal of Matrix Analysis and Applications, accepted*, 2008.

[38] M. Naumov and A. H. Sameh. A tearing based hybrid parallel banded solver. *Journal of Computational and Applied Mathematics, accepted*, 2008.

[39] G. K. Pedersen. *Analysis Now*. Springer Verlag, New York, second edition, 1995.

[40] T. Penzl. A cyclic low-rank Smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comp. 2000*, 21(4):1401–1418, 2000.

[41] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, 40(2):139–144, 2000.

[42] T. Penzl. Lyapack users' guide. `http://www.netlib.org/lyapack/guide.pdf`, 2000.

[43] E. Polizzi and A. H. Sameh. A parallel hybrid banded system solver: the SPIKE algorithm. *Parallel Computing*, 32:177–194, 2006.

[44] E. Polizzi and A. H. Sameh. SPIKE: A parallel enviroment for solving banded linear systems. *Computers & Fluids*, 36:113–120, 2007.

[45] P. J. Psarrakos and M. J. Tsatsomeros. Numerical range: (in) a matrix nutshell. Notes, National Technical University, Athens, Greece, 2004.

[46] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *International Journal of Control*, 32(4):677–687, 1980.

[47] H. L. Royden. *Real Analysis*. Prentice-Hall, New Jersey, third edition, 1988.

[48] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Halsted Press, New York, 1992.

[49] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, 7(3):856–869, 1986.

[50] Yousef Saad. Numerical solution of large Lyapunov equations. *Progr. Systems Control Theory*, 5:503–511, 1990.

[51] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, USA, second edition, 2003.

[52] J. Sabino. *Solution of large-scale Lyapunov equations via the block modified Smith method*. PhD thesis, University of Houston, Texas, 2006.

[53] A. H. Sameh and D. J. Kuck. On stable parallel linear system solvers. *Journal of the ACM*, 25:81–91, 1978.

[54] Joel H. Shapiro. Notes on the numerical range. Notes, Michigan State University, USA, 2004.

[55] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Scient. Computing*, 29(3):1268–1288, 2007.

[56] V. Simoncini and V. Druskin. Convergence analysis of projection methods for the numerical solution of large Lyapunov equations. `http://www.dm.unibo.it/~simoncini/list.html`, 2008.

[57] Valeria Simoncini and Daniel B. Szyld. Recent computational developments in Krylov subspace methods. *Numerical Linear Algebra and Applications*, 14(1):1–59, 2007.

[58] R. A. Smith. Matrix equation $XA + BX = C$. *SIAM Journal on Applied Mathematics*, 16(1):198–201, 1968.

[59] D. Sorensen and Y. Zhou. Direct methods for matrix Sylvester and Lyapunov equations. *Journal of Applied Mathematics*, 6:277–303, 2003.

[60] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[61] G. W. Stewart. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices. *Numerische Mathematik*, 25:123–136, 1976.

[62] G. W. Stewart. *Matrix Algorithms*. SIAM, first edition, 2001.

[63] J. C. Strikwerda. *Finite difference schemes and partial differential equations*. Wadsworth & Brooks/Cole, first edition, 1989.

[64] Xian-He Sun. Application and accuracy of the parallel diagonal dominant algorithm. *Parallel Computing*, 21:1241–1267, 1995.

[65] Xian-He Sun, Hong Zhang, and Lionel M. Ni. Efficient tridiagonal solvers on multicomputers. *IEEE Transactions on Computers*, 41(3):286–296, 1992.

[66] Eugene L. Wachspress. Iterative solution of the Lyapunov matrix equation. *Applied Mathematics Letters*, 1(1):87–90, 1988.

[67] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, first edition, 1965.

[68] K. Zhou, J. C. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall, New Yersey, 1996.

[69] Y. Zhou and D. C. Sorensen. Bounds on eigenvalue decay rates and sensitivity of solutions to Lyapunov equations. Technical research report TR-02-07, CACM Rice University, 2002.

[70] Y. Zhou and D.C. Sorensen. Approximate implicit subspace iteration with alternating directions for LTI system model reduction. *Numer. Linear Algebra Appl.*, to appear.

APPENDICES

A. The Cayley transform

A.1 Introduction

Let A be an n by n matrix and let $\sigma(A)$ denote the set of eigenvalues for A. Let $\mathcal{S}$ be the set of matrices, given by

$$\mathcal{S} = \{A \,:\, 1 \notin \sigma(A)\}.$$

The Cayley transform $\mathcal{C}(A)$ of A, given by

$$\mathcal{C}(A) = (A + I)(A - I)^{-1},$$

is well defined for all $A \in \mathcal{S}$.

The Cayley transform can be viewed as an extension of the function

$$\phi(z) = \frac{z+1}{z-1},$$

which maps $\mathbb{C} - \{1\}$ one to one and onto itself, and $\phi(\phi(z)) = z$ for all $z \in \mathbb{C} - \{1\}$.

We became interested in the Cayley transform because it provides a simple connection between discrete and continuous time Lyapunov equations. In this appendix we derive a list of simple results on the Cayley transform. We have been unable to locate a reference which contains the main result, Theorem A.2.3, which is why we give such detailed proofs.

A.2 Properties of the Cayley transform

Theorem A.2.1 *The Cayley transform maps $\mathcal{S}$ one to one and onto S, and*

$$\mathcal{C}(\mathcal{C}(A)) = A,$$

for all $A \in \mathcal{S}$.

Proof Let $A \in \mathcal{S}$. Then $\mathcal{C}(A)$ is well defined. We want to show that $\mathcal{C}(A) \in \mathcal{S}$. We must show that 1 is not an eigenvalue for $\mathcal{C}(A)$. Let (μ, w) be an eigenpair for $\mathcal{C}(A)$, then

$$(A + I)(A - I)^{-1}w = \mu w,$$

or equivalently

$$(A + I)w = \mu(A - I)w,$$

from which it follows

$$(1 - \mu)Aw = -(1 + \mu)w.$$

Now, if $\mu = 1$, then $w = 0$, which is impossible, because w is an eigenvector for $\mathcal{C}(A)$. It follows that $\mathcal{C}(A) \in \mathcal{S}$. We conclude that the Cayley transform maps $\mathcal{S}$ into itself. We now show that $\mathcal{C}(\mathcal{C}(A)) = A$. By definition

$$\mathcal{C}(A) = (A + I)(A - I)^{-1} = (A - I + 2I)(A - I)^{-1} = I + 2I(A - I)^{-1},$$

from which it follows, that

$$\begin{aligned}\mathcal{C}(\mathcal{C}(A)) = (\mathcal{C}(A) + I)(\mathcal{C}(A) - I)^{-1} &= 2(I + (A - I)^{-1})(2(A - I)^{-1})^{-1} \\ &= (I + (A - I)^{-1})(A - I) = (A - I) + I = A.\end{aligned}$$

We conclude that the Cayley transform maps $\mathcal{S}$ one to one and onto itself, and $\mathcal{C}(\mathcal{C}(A)) = A$ for all $A \in \mathcal{S}$. ■

Now, let $A \in \mathcal{S}$, and let (λ, v) be an eigenpair for A. By definition

$$Av = \lambda v,$$

which implies

$$(A \pm I)v = (\lambda \pm 1)v,$$

and

$$\mathcal{C}(A)v = (A + I)(A - I)^{-1}v = \phi(\lambda)v.$$

We conclude that $(\phi(\lambda), v)$ is an eigenpair for $\mathcal{C}(A)$. By Theorem A.2.1 it follows immediately that (λ, v) is an eigenpair for A if and only if $(\phi(\lambda), v)$ is an eigenpair for $\mathcal{C}(A)$.

It is well known that ϕ maps the open left hand complex plane one to one and onto the open unit disk. As a result we have the following theorem.

Theorem A.2.2 *A is a stable matrix if and only if the spectral radius of $\mathcal{C}(A)$ is strictly less than* 1.

We now prove the main result

Theorem A.2.3 *A is a negative definite matrix if and only if $\|\mathcal{C}(A)\|_2 < 1$.*

Proof We received this elegant proof from Prof. Bent Ørsted from the University of Aarhus, Denmark. By definition

$$\mathcal{C}(A)^T\mathcal{C}(A) - I = (A-I)^{-T}(A+I)^T(A+I)(A-I)^{-1} - I,$$

which we rewrite as

$$\begin{aligned}\mathcal{C}(A)^T\mathcal{C}(A) - I &= (A-I)^{-T}(A+I)^T(A+I) - (A-I)^T(A-I))(A-I)^{-1} \\ &= (A-I)^{-T}(2(A+A^T))(A-I)^{-1},\end{aligned}$$

or equivalently

$$I - \mathcal{C}(A)^T\mathcal{C}(A) = (A-I)^{-T}(-2(A+A^T))(A-I)^{-1}$$

Now if $\|\mathcal{C}(A)\| < 1$, then $I - \mathcal{C}(A)^T\mathcal{C}(A)$ is symmetric positive definite and admits a Cholesky factorization

$$I - \mathcal{C}(A)^T\mathcal{C}(A) = LL^T.$$

It follows immediately, that

$$-2(A+A^T) = (A-I)^T LL^T (A-I)$$

is also symmetric positive definite. Similarly if A is negative definite, then

$$I - \mathcal{C}(A)^T \mathcal{C}(A)$$

must necessarily be symmetric positive definite and it follows that

$$\mathcal{C}(A)^T \mathcal{C}(A) < I,$$

or equivalently

$$\|\mathcal{C}(A)\|_2 < 1.$$

■

Now suppose A is a negative definite matrix, then $\|\mathcal{C}(A)\|_2 < 1$. Let σ be a singular value for $\mathcal{C}(A)$, and let v be the corresponding right singular vector, then

$$\mathcal{C}(A)^T \mathcal{C}(A) v = \sigma^2 v$$

or equivalently

$$(A + I)^T (A + I) w = \sigma^2 (A - I)^T (A - I) w,$$

where we have introduced $w = (A - I)^{-1} v$. It follows that

$$(1 - \sigma^2)(A^T A + I) w = -(1 + \sigma^2)(A + A^T) w$$

and since $\sigma^2 < 1$, we have

$$(A^T A + I) w = -\frac{1 + \sigma^2}{1 - \sigma^2} (A + A^T) w = \phi(\sigma^2)(A + A^T) w$$

We see that the singular values of $\mathcal{C}(A)$ are given by the eigenvalues of a generalized eigenvalue problem involving symmetric positive definite matrices.

B. Drawings of the SPIKE partitioning

This appendix contains some large drawings of the partitioning used in the SPIKE algorithms. They contain information which has already been presented in the main chapter, but the drawings are clearer than the defining equations and we feel justified in presenting them here.

The case of $p = 3$ partitions is the simplest example which represents all aspect of the general case. In the case of $p = 2$ the reduced system is equal to the truncated reduced system and there is no truncation error.

We have illustrated the partitioning of the original matrix in Figure B.1, the layout of the SPIKE system in Figure B.2, emphasized the position of the reduced system in Figure B.3, and isolated the reduced system, as well as the truncated reduced system in Figure B.4, and Figure B.5.

It is assumed that the number of superdiagonals k is equal to the number of subdiagonals and that the number of processors n divides the dimension n of the matrix. The diagonal blocks is $\mu = n/p$.

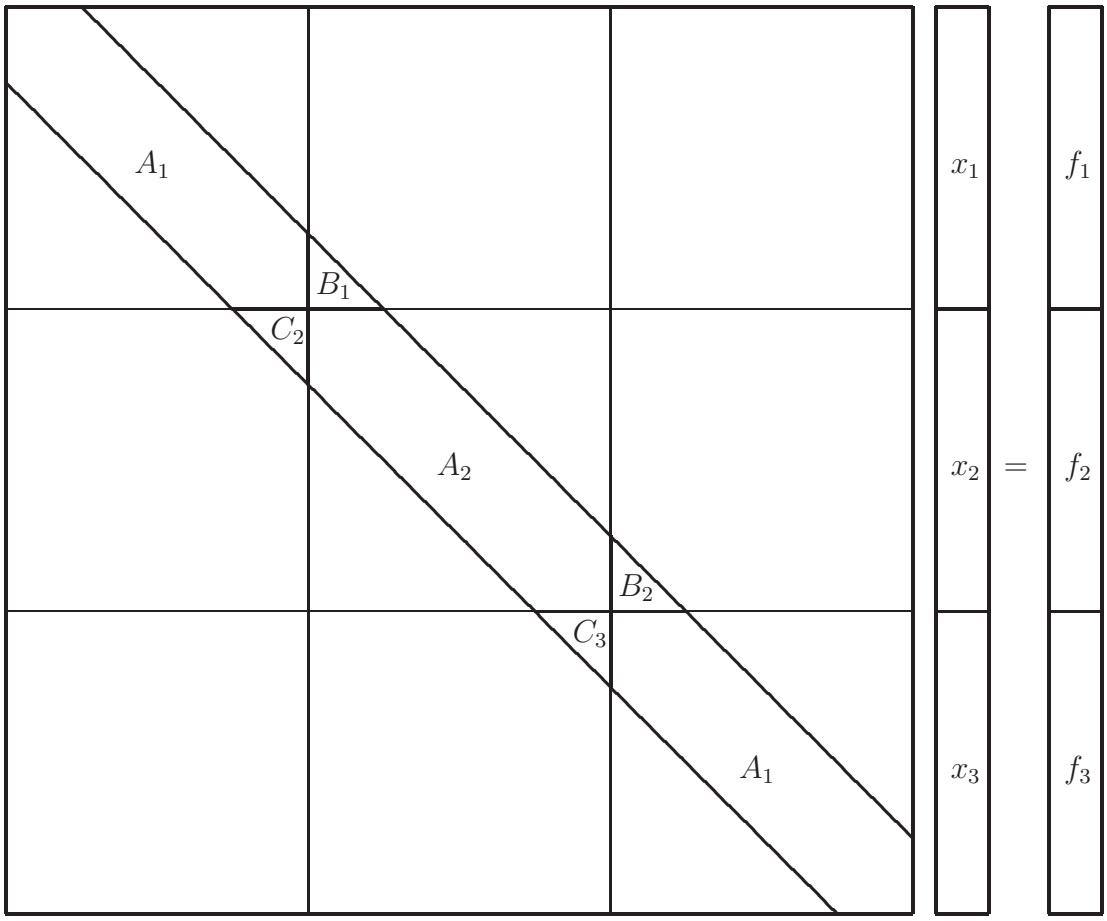

Figure B.1. The partitioning of the original system for the SPIKE algorithms, $p = 3$ partitions.

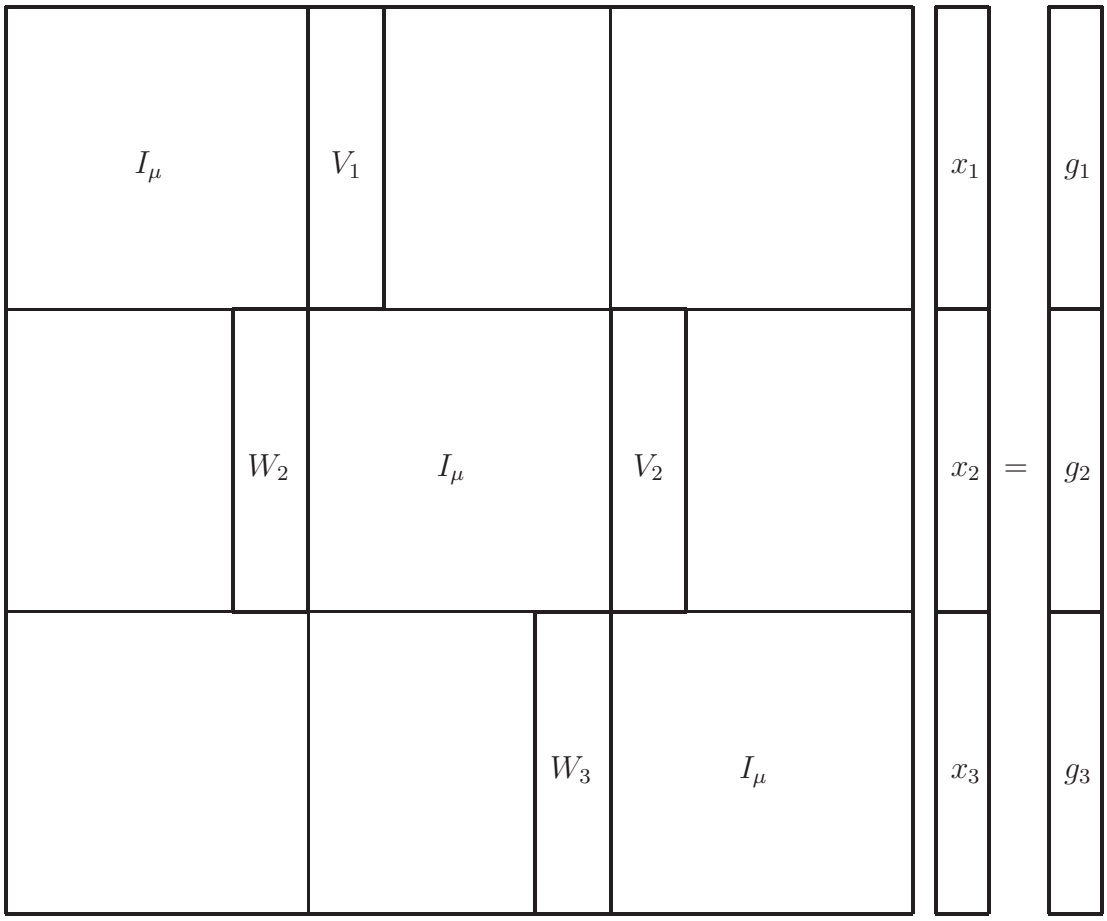

Figure B.2. The SPIKE system corresponding to $p = 3$ partitions.

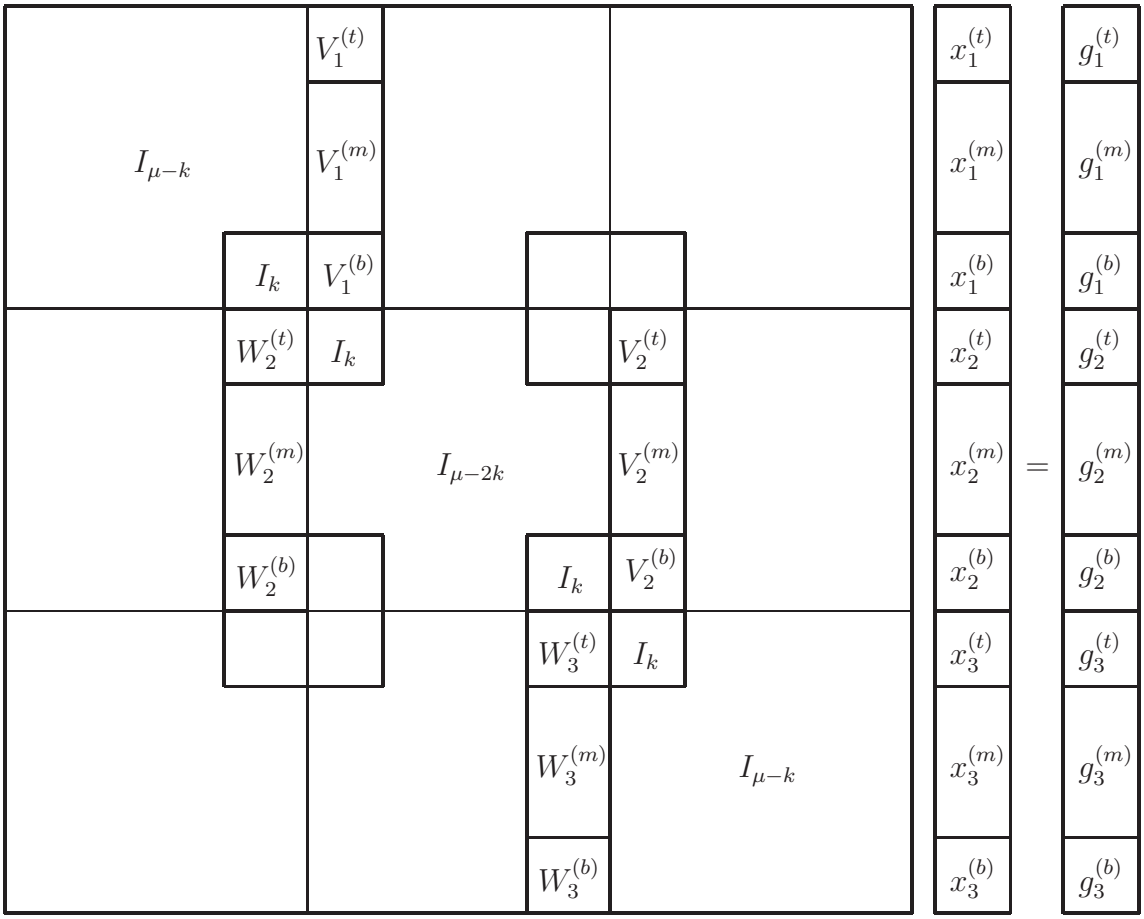

Figure B.3. The SPIKE system corresponding to $p = 3$ partitions with special emphasis on the reduced system.

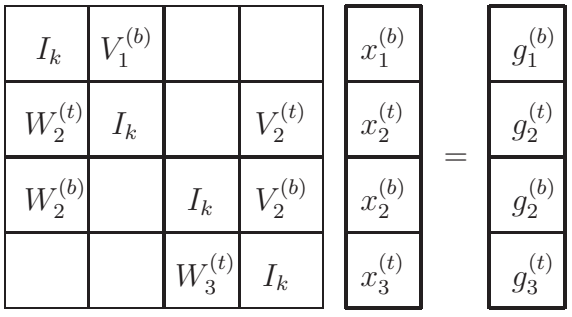

Figure B.4. The reduced system corresponding to $p = 3$ partitions. The reduced system matrix has dimension $4k$.

$$\begin{bmatrix} I_k & V_1^{(b)} & & \\ W_2^{(t)} & I_k & & \\ & & I_k & V_2^{(b)} \\ & & W_3^{(t)} & I_k \end{bmatrix} \begin{bmatrix} x_1^{(b)} \\ x_2^{(t)} \\ x_2^{(b)} \\ x_3^{(t)} \end{bmatrix} = \begin{bmatrix} g_1^{(b)} \\ g_2^{(t)} \\ g_2^{(b)} \\ g_3^{(t)} \end{bmatrix}$$

Figure B.5. The truncated reduced system responding to $p = 3$ partitions.

VITA

VITA

Carl Christian Kjelgaard Mikkelsen was born in Aarhus, Denmark, on the 3rd of February 1976. He received his masters degree in mathematics from the University of Aarhus in May 2003. He has been a graduate student at Purdue University since January 2004.