LOW-RANK SOLUTION METHODS FOR LARGE-SCALE LINEAR MATRIX EQUATIONS

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Stephen D. Shank
May, 2014

Examining Committee Members:

Daniel B. Szyld, Advisory Chair, Mathematics
Benjamin Seibold, Mathematics
Valeria Simoncini, External Member, University of Bologna, Mathematics
Wei-Shih Yang, Mathematics

ABSTRACT

LOW-RANK SOLUTION METHODS FOR LARGE-SCALE LINEAR MATRIX EQUATIONS

Stephen D. Shank
DOCTOR OF PHILOSOPHY

Temple University, May, 2014

Professor Daniel B. Szyld, Chair

We consider low-rank solution methods for certain classes of large-scale linear matrix equations. Our aim is to adapt existing low-rank solution methods based on standard, extended and rational Krylov subspaces to solve equations which may viewed as extensions of the classical Lyapunov and Sylvester equations. The first class of matrix equations that we consider are constrained Sylvester equations, which essentially consist of Sylvester's equation along with a constraint on the solution matrix. These therefore constitute a system of matrix equations. The second are generalized Lyapunov equations, which are Lyapunov equations with additional terms. Such equations arise as computational bottlenecks in model order reduction.

ACKNOWLEDGEMENTS

First, I would like to thank my advisors Daniel B. Szyld and Valeria Simoncini for all of their efforts towards my development as a mathematician. I thank Daniel for challenging me, for believing in me, for showing me a perspective of mathematics that I have come to appreciate and benefit from, and for pushing me harder than I have ever been pushed. Valeria was a fantastic mentor and host; the time spent in Bologna is something that I will always have fond memories of, and the hospitality is warmly acknowledged. Both deserve more thanks than I can give here for putting up with my antics, and for situating me to launch a successful career. I have come to appreciate Daniel's analogy of a mathematical family, and I hope we can stay in touch for years to come.

Boris Datskovsky was very influential on me throughout my undergraduate career at Temple; many of my encounters with him are what inspired me to pursue a career as a mathematician. Benjamin Seibold delivered inspiring lectures and talks, and these were the reason that I began a study of applied mathematics. Ray Tuminaro was a great person to work with and a great mentor. Many of my fellow graduate students deserve thanks for sharing ideas, enduring rants, and having drinks. These include Dong, Giulio, Kirk, Scott, Matt, Austin, Mattia, and Yiding.

My parents, Rachel and David, were an unending source of support and strength. They encouraged me to work hard and endure whenever it got tough, and were always there when I needed them. I hope I have made them proud.

Finally, one person deserves more thanks than I can ever put into words. She has been supportive and understanding, always willing to listen and to comfort. She has put up with weeks, even entire seasons apart, so that I could pursue my dreams while we pursue ours. Her love has made all the difference, without which I would be lost. For all of these reasons and more, Shana, this thesis is for you.

To Shana Album,

with love, now and always.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Linear matrix equations

This thesis is concerned with matrix equations and low-rank solution methods. A survey of techniques for linear matrix equations can be found in [69]. Another survey that gives an overview of low-rank solution methods and their applications may be found in [28]. Two standard references for required topics on numerical linear algebra are [26, 75], from which much notation and terminology are borrowed. A good general reference for linear algebra that is slightly more on the theoretical side is [23].

1.1.1 Existing results

We will primarily focus on the numerical solution of large-scale linear matrix equations. Many of these equations may be viewed as generalizations of *Sylvester's equation*

$$AX + XB + EF^T = 0, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{\ell\times\ell}$, $E \in \mathbb{R}^{n\times r}$, and $F \in \mathbb{R}^{\ell\times r}$ for the unknown $X \in \mathbb{R}^{n\times\ell}$. By linear, it is meant that the operator $\mathcal{S}(X) = AX + XB$ is a linear function of the unknown X. By large-scale, it is meant that $n, \ell \gg 1000$. We also assume A and B are sparse. We give a slightly imprecise definition of

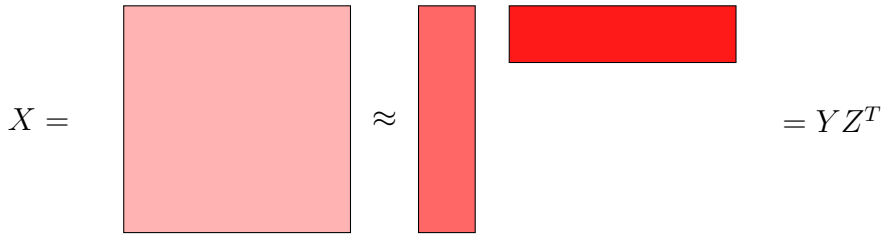


Figure 1.1: Depiction of a dense matrix X and a low-rank approximation YZ^T.

sparsity, saying that an $n \times n$ matrix is sparse if it has $\mathcal{O}(n)$ nonzero entries. In the next section we will see that the coefficient matrices are associated with some dynamical system, which may often be a two- or three-dimensional partial differential equation that has been semi-discretized in time.

A fundamental challenge when solving (1.1) in this large-scale setting is storing the solution matrix X, which is typically dense even when A and B are sparse. Practical applications arise where $r \ll n, \ell$. In fact, often $r \leq 10$ and one may even have $r = 1$, in which case $E = e$ and $F = f$ are column vectors. We will refer to this as having a low-rank right-hand side. In this case, the resulting storage requirements for the data A, B, E and F of (1.1) are $\mathcal{O}(n + \ell)$, which is comparable with what one expects when numerically solving a PDE. However, the solution matrix X, being dense, has $\mathcal{O}(n\ell)$ storage requirements. Thus we see that, in the large-scale case, even storing a solution to (1.1) is computationally challenging!

We shall soon see that in the large-scale, low-rank right-hand side scenario it is often the case that a solution may approximated as $X \approx YZ^T$ with Y and Z both having p columns, where $p \ll n, \ell$, and this is what is meant by a *low-rank approximation*. If one attempts to solve for the low-rank factors Y and Z instead of X, the cost for storing both the data and an approximate solution for (1.1) are both $\mathcal{O}(n + \ell)$. The problem therefore becomes computationally tractable, at least in theory. A depiction of a low-rank approximation to a dense matrix when $n = \ell$ is given in Figure 1.1.

1.1.2 Contributions of the thesis

The main contributions of this thesis are the development and discussion of numerical solvers for classes of large-scale linear matrix equations that may be viewed as a generalization of (1.1). We concentrate on numerical methods which maintain the optimal computational complexity of $\mathcal{O}(n+\ell)$. Our methods are based on extensions of existing low-rank solution techniques, typically based on Galerkin projection onto properly chosen subspaces and low-rank versions of popular linear solvers. Details of existing methods of this sort are given in Chapter 2. We provide what we believe to be either new or competitive solvers for such problems.

The first problem we consider is the so-called *constrained Sylvester equation*

$$A_1X + XA_2 - YC = 0 \tag{1.2}$$

$$XB = 0 \tag{1.3}$$

for the unknowns X and Y. This may be viewed as a Sylvester equation (1.2) with a constraint (1.3) on the solution matrix X. In [4], a numerical method is presented for solving (1.2)–(1.3) in the small-scale case, i.e., $n, \ell \ll 10000$[1]. We propose a modification of this approach that is suitable for the large-scale case and provide experimental evidence of its effectiveness on benchmark problems.

We also consider *generalized Lyapunov equations*

$$AX + XA^T + \sum_{j=1}^{k} N_jXN_j^T + BB^T = 0 \tag{1.4}$$

for the unknown X. Numerical solution methods based on extending existing ADI-based methods for the standard Lyapunov equation and low-rank preconditioned Krylov subspace methods were already considered in [7]. In this thesis we propose solution methods for (1.4) which utilize a variety of techniques found in the literature on Krylov subspace methods. We believe that what results is competitive with respect to the state-of-the-art in terms of

[1]The name "constrained Sylvester equation" is borrowed from that article.

computational and memory demands, as well number of subroutine calls such as solving a linear system with coefficient matrix A.

1.2 Applications

Though this thesis focuses primarily on numerical methods, we survey a short list of applications here to provide some motivation for developing numerical methods in the first place. For a survey of applications related to solving (1.1), see [12]. A reason to numerically solve the small-scale case of (1.2)–(1.3) was motivated by the design of reduced-order observers which achieve precise loop transfer recovery [46]. The large-scale case may be viewed as a generalization of the recovery design idea for a significantly larger number of degrees of freedom.

Consider a linear, time-invariant (henceforth LTI) dynamical system

$$\dot{x}(t) = Ax(t) + Bu(t), \;\; x(0) = x_0 \tag{1.5}$$

$$y(t) = Cx(t), \tag{1.6}$$

which may arise, for instance, from a semi-discretization in time of the heat equation

$$\begin{aligned} x_t - \Delta x &= 0 \text{ in } \Omega \\ x &= u \text{ on } \partial\Omega, \end{aligned}$$

as described in, e.g., [42]. The variables x, u, and y are referred to as the state, input, and output, respectively. Here the output may be the temperature at a single point in the room, in which case $C = e_k^T$, where e_k is a standard basis vector. A special case of the Sylvester equation (1.1) is the *Lyapunov equation*, where $B = A^T$ and $E = F$. One may wish to choose the input u to produce some desired behavior in the output, and formalizing this mathematically gives rise to what are known as linear-quadratic optimal control problems. For a discussion of such problems and the relevance of solving the Lyapunov equation, see [9].

We summarize some results on LTI systems; specifically, the notions of reachability, observability, and model order reduction by balanced truncation. A standard reference on these topics is [1]. Solving the Lyapunov equations

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

for P and Q (known as the reachability and observability Gramians) yield information about energies associated with inputs and outputs of the system (1.5)–(1.6). For instance, given some desired state x_d, one may seek an input function u and time t_d such that $x(t_d) = x_d$. If this is possible, the state x_d is called *reachable*. One can show that the set of all reachable states is given by range (P). Often one is concerned with the least amount of energy[2] required to drive the system to such a desired state. This is given by $x_d^T P^{-1} x_d$, which addresses the issues of numerical reachability and describes which states may be reached in practice.

Another relevant notion is that of observability; a state x_o will be called *unobservable* if the initial condition $x(0) = x_o$ and zero input function $u(t) \equiv 0$ yield $y(t) \equiv 0$. Therefore, an unobservable state x_o is indistinguishable from the zero state, at least as far as the output of the system is concerned. The set of all unobservable states can be shown to be null(Q). The maximum energy observed in the output from an arbitrary state x_o can be shown to be $x_o^T Q x_o$. A state whose energy is small in this sense is regarded as numerically unobservable; it is virtually indistinguishable from the zero state and any influence on the output may be regarded as negligible.

Intuitively, states that are hard to reach or difficult to observe should not be expected to influence the map of inputs to outputs, often referred to as the *system transfer*. Therefore, one might desire the ability to construct a smaller

[2]Energy is defined in the L_2 sense, i.e., as $\|u\|_{L_2([0,t])} = \int_0^t u^T(\tau)u(\tau)d\tau$.

dynamical system

$$\begin{aligned}\dot{\hat{x}} &= \hat{A}\hat{x}(t) + \hat{B}u(t)\\ \hat{y}(t) &= \hat{C}\hat{x}(t),\end{aligned}$$

where $\hat{A} \in \mathbb{R}^{\hat{n}\times\hat{n}}$ with $\hat{n} \ll n$, without any states that are negligible with respect to this transfer behavior. Ideally, such a system will capture the behavior of inputs with respect to outputs; formally, one would require $\|y - \hat{y}\| < \tau\|u\|$ for some threshold τ and an appropriate norm. Constructing such a smaller system while preserving properties of the original system is known as model order reduction. Cholesky factors (or more generally, square roots) of P and Q can be used to compute a transformation T so that the system

$$\begin{aligned}\dot{x}_T &= \left(TAT^{-1}\right)x_T(t) + TBu(t)\\ y(t) &= CT^{-1}x_T(t)\end{aligned}$$

has reachability and controllability Gramians $P_T = Q_T = \mathrm{diag}(\sigma_1, \ldots \sigma_n)$, which is known as a *balanced* system. Note that the above system produces the same output y from a given input u as the original system. The scalars σ_i are called the Hankel singular values, and one can show that a rapid decay is expected [2].

Once transformed, states in the new system that are difficult to reach are now also equally difficult to observe. Such states are discarded from the system, producing a smaller system. This procedure is known as balanced truncation, and a rich theory is available. For a given input, the resulting system (being much smaller) is now much less expensive to step in time. For large-scale systems, only approximate low-rank factors of P and Q are needed, and an *approximate* balanced truncation can be computed. For details, see [1, 30, 53].

There are analogues of reachability, observability, and balanced truncation

for *bilinear dynamical systems*

$$\dot{x}(t) = Ax(t) + \sum_{j=1}^{k} N_j x(t) u_j(t) + Bu(t), \;\; x(0) = x_0 \tag{1.7}$$

$$y(t) = Cx(t), \tag{1.8}$$

as well as stochastic systems

$$dx = Axdt + \sum_{j=1}^{k} N_j x(t) dw_j + Budt \tag{1.9}$$

$$y = Cx, \tag{1.10}$$

with each w_j denoting an independent zero mean Wiener process on a probability space $(\Omega, \mathcal{F}, \mu)$. Analogues of controllability and observability Gramians for (1.7)–(1.8) have been formulated and are shown to satisfy the generalized Lyapunov equation (1.4); see for instance [54]. As such, a notion of balanced truncation for bilinear systems can be formulated which requires solving two generalized Lyapunov equations, and the numerical solution of these equations are the computational bottleneck. Balanced truncation for bilinear systems is described in [8], where analogous results for stochastic systems (1.9)–(1.10) are also discussed. Moreover, these same equations play a role in stochastic control [15].

A survey on theory and applications of bilinear systems (1.7)–(1.8) may be found in [13]. One such bilinear system arises as a model in the steel industry. When trying to optimize the cooling of steel, one seeks to cool a recently processed beam as quickly as possible while preventing large temperature gradients that may lead to a degradation in the quality of the material. Modeling this physical process leads to a boundary control problem of the heat equation; see [64]. The stochastic system arises when trying to control the motion of a particle undergoing dragged Brownian motion; see [33].

1.3 Notation and preliminary definitions

In this section we gather some notation and preliminary definitions. The $n \times n$ identity matrix is denoted by I_n, the $n \times m$ zero matrix is denoted by $0_{n\times m}$, and $0_{n\times n} = 0_n$ denotes the square $n \times n$ zero matrix. The m^{th} standard unit basis vector will be denoted e_m, and its size should be clear from the surrounding context. A block of columns from the identity matrix will be denoted by $\mathbf{E}_m$, and the size of the identity matrix along with which columns are taken should again be clear from context. The transpose of a matrix X is denoted by X^T. By $\text{range}\,(X)$ we denote the column space of X, i.e., the linear span of the columns of X.

The Frobenius inner product of two matrices X and Y is denoted by $\langle X, Y\rangle_F = \text{trace}\left(X^T Y\right)$. The set of polynomials of degree m will be denoted by $\mathbb{P}_m$. The spectrum of a matrix will be denoted by $\Lambda(A)$, and the field of values of a matrix will be denoted by $W(A) = \{\frac{z^*Az}{z^*z} : z \in \mathbb{C}^n\}$. The complex plane is denoted by $\mathbb{C}$. The extended complex plane is given by $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$, and the left half of the complex plane shall be denoted $\mathbb{C}^-$. A matrix A is called *stable* if $W(A) \subset \mathbb{C}^-$.

Definition 1.1. The *vectorization operator* $\text{vec} : \mathbb{R}^{n\times\ell} \to \mathbb{R}^{n\ell}$ stacks the columns of a matrix X into a long column vector x; i.e., writing $X = [x_1, \ldots, x_\ell]$ in terms of columns, where $x_j \in \mathbb{R}^n$ for $j = 1, \ldots, \ell$, one has

$$\text{vec}(X) = \begin{bmatrix} x_1 \\ \vdots \\ x_\ell \end{bmatrix}.$$

Cleary the vectorization operator is linear, and in fact it is a vector space isomorphism. It is also an isometry in the sense that $\langle \text{vec}(A), \text{vec}(B)\rangle_2 = \langle A, B\rangle_F$.

Definition 1.2. Given matrices $S = (s_{ij}) \in \mathbb{R}^{m\times n}$ and $T \in \mathbb{R}^{\ell\times p}$, the *Kro-*

necker product $S \otimes T \in \mathbb{R}^{m\ell \times np}$ is defined as

$$S \otimes T = \begin{bmatrix} s_{11}T & \dots & s_{1n}T \\ \vdots & \ddots & \vdots \\ s_{m1}T & \dots & s_{mn}T \end{bmatrix}.$$

CHAPTER 2

PRELIMINARIES

We review material that is relevant for solving large-scale linear matrix equations that will be useful for subsequent purposes. We emphasize that all material found in this chapter corresponds to existing results, and are not an original contribution of the author.

2.1 Krylov subspace methods for linear systems

Among the most popular methods for solving large, sparse linear systems are what have collectively come to be known as Krylov subspace methods. Many ideas underlying these methods transfer naturally to several low-rank solution methods for linear matrix equations that are the subject of this thesis. We therefore review certain aspects of the theory as a means of developing subsequent material and fixing notation. Readers interested in further details can refer to the comprehensive books [29, 44, 63, 79] and the survey [72].

2.1.1 Standard Krylov subspaces

Consider the solution of the linear system

$$Ax = b \tag{2.1}$$

Algorithm 2.1 Arnoldi

Input: A, b, m

Output: Orthonormal basis $\{v_1, \ldots, v_m\}$ of $\mathbb{K}_m(A, b)$

```
β = ‖b‖_2, v_1 = b/β
for j = 1, ..., m do
    w = A v_j
    for i = 1, ..., j do
        h_ij = v_i^T w
        w = w − h_ij v_i
    end for
    h_{j+1,j} = ‖w‖_2
    v_{j+1} = w / h_{j+1,j}
end for
```

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular, sparse matrix, $x, b \in \mathbb{R}^n$, and n is large. As a first step towards solving (2.1), we proceed with a few definitions and algorithms that pertain to *Krylov subspaces*.

Definition 2.1. The m^{th} *Krylov subspace* is defined as

$$\begin{aligned}\mathbb{K}_m(A, b) :&= \operatorname{span}\{b, Ab, \ldots, A^{m-1}b\} \\ &= \{p(A)b : p \in \mathbb{P}_{m-1}\}.\end{aligned}$$

We will often write $\mathbb{K}_m$ when there is no ambiguity regarding A and b. An algorithm for building this space is the *Arnoldi algorithm*, which is described in Algorithm 2.1. Krylov subspaces can be shown to satisfy the "nested" properties

$$\mathbb{K}_m \subseteq \mathbb{K}_{m+1} \text{ and } A\mathbb{K}_m \subseteq \mathbb{K}_{m+1}. \tag{2.2}$$

The scalars h_{ij} are gathered in the upper-Hessenberg matrix $H_m = (h_{ij}) \in \mathbb{R}^{m\times m}$, and the vectors $v_1, \ldots, v_m$ are placed as columns in the "tall and skinny"

matrix V_m. One can show that the following *Arnoldi relation* holds:

$$AV_m = V_{m+1}\underline{H}_{m+1} \tag{2.3}$$

$$= V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{2.4}$$

where

$$\underline{H}_{m+1} := \begin{bmatrix} H_m \\ h_{m+1,m} e_m^T \end{bmatrix}.$$

We remark that relation (2.3) may be viewed as an expression of the nested property (2.2), since every column of AV_m is in $\mathbb{K}_{m+1}$, and the coefficients required to express them in this basis are gathered in the matrix $\underline{H}_{m+1}$. We shall see that having these coefficients available as a byproduct of the Arnoldi process is indeed desirable and will aid us when using Krylov subspaces to solve (2.1).

We note that the orthogonalization is done according to the modified Gram-Schmidt method [26, 75], which is known to be more stable than the standard Gram-Schmidt orthogonalization. Often one observes a loss of orthogonality which may negatively impact quantities of interest, such as approximations to (2.1). One can partially restore this loss of orthogonalization by *reorthogonalizing*. It is worth noting that one and only one extra orthogonalization is done in accordance with the "twice is enough" rule [24]. In the remainder of this thesis, all algorithms for building orthogonal bases are implemented with reorthogonalization. Algorithms with attention to such details are given in Appendix A, and are equivalent in exact arithmetic to those presented in the main body of this thesis.

An ubiquitous method in scientific computing for computing an approximate solution to a problem is known as *Galerkin projection*. Loosely speaking, it consists of three parts. First, a subspace where a decent approximate solution is expected to be found is defined, which we call the search space. Second, a residual[1] for any given approximate solution is formulated. Finally,

[1] A defining characteristic of a residual is that if a given residual is zero, the corresponding approximate solution is actually the true solution to the problem.

the approximate solution from the search space is selected by enforcing that its corresponding residual is orthogonal to the search space.

The *full orthogonalization method*, or FOM, is a method for approximating the solution to (2.1) by performing a Galerkin projection with search space $\mathbb{K}_m$. We consider this in some detail, as it contains seminal ideas for later developments. Given an approximate solution $x_m \in \mathbb{K}_m$, the corresponding residual is defined as

$$r_m = b - Ax_m,$$

and x_m is selected by enforcing

$$r_m \perp v \text{ for all } v \in \mathbb{K}_m.$$

Using Algorithm 2.1 to build $\mathbb{K}_m$, one can write $x_m = V_m y_m$. It is straightforward to show that (2.4) implies that

$$V_m^T A V_m = H_m$$

where $V_m^T A V_m$ is called a *Rayleigh Quotient matrix*. This may be exploited to derive a smaller linear system for y_m, as

$$r_m \perp \mathbb{K}_m \iff V_m^T r_m = 0 \iff H_m y_m = \beta e_1,$$

which is a small linear system for y_m, since we typically expect $m \ll n$. The coefficient matrix for this small system is precisely the Rayleigh quotient matrix and is explicitly available from the Arnoldi algorithm. It is often the case that one is justified in regarding the extra computation incurred when solving this small system as negligible. Moreover, the nested properties (2.2) imply that $r_m \in \mathbb{K}_{m+1}$. Since one possesses an orthonormal basis of this space at the m^{th} stage of the Arnoldi algorithm, it can be shown that $\|r_m\|_2$ is also available at negligible cost.

In the case of symmetric positive definite A, FOM (with a fancier implementation) reduces to the popular *conjugate gradients* method. Other popular methods may be described mathematically as *Petrov-Galerkin* methods.

In these methods two spaces are chosen: one to search for approximate solutions, and another to enforce orthogonality for the corresponding residual. They provide a high-level mathematical description of many popular methods for solving (2.1) such as the GMRES, MINRES, and BiCG methods[2]. With appropriate implementation details, these methods comprise a powerful class of methods for solving linear systems and are widely used in academia, government, and industry. Since the simpler notion of Galerkin projection will be more prominent in this thesis, we omit such details and refer the interested reader to the references at the beginning of the chapter.

2.1.2 Block Krylov subspaces

It is often the case that one needs to solve several, say k, linear systems

$$Ax_j = b_j \text{ for } j = 1, \ldots, k \tag{2.5}$$

for a fixed coefficient matrix A. We gather all right-hand sides b_j in the "tall and skinny" matrix $B = [b_1, \ldots, b_k]$.

Definition 2.2. The m^{th} *block Krylov subspace* is defined as

$$\mathbb{K}_m(A, B) = \mathbb{K}_m(A, b_1) + \cdots + \mathbb{K}_m(A, b_k) \tag{2.6}$$

where the sum denotes the sum vector spaces. An equivalent definition is

$$\mathbb{K}_m(A, B) = \text{range}\left([B, AB, \ldots, A^{m-1}B]\right).$$

We again write $\mathbb{K}_m$ when there is no ambiguity regarding b, b_j, or B, and let $\mathbf{V}_m$ denote a matrix whose columns form an orthonormal basis of (2.6). An algorithm analogous to Algorithm 2.1 for building an orthonormal basis of (2.6) is the block Arnoldi algorithm, described below. We note that the sum in (2.6) may not always be a direct one; i.e., there may be a loss of linear independence amongst the columns of $\mathbf{V}_m$. The difficulty is compounded in finite-precision

[2]Even BiCGStab may be viewed as a Petrov-Galerkin method, but with *rational Krylov subspaces* that shall be introduced in the next chapter; see [73].

Algorithm 2.2 Block Arnoldi

Input: A, B, m

Output: Matrix $\mathbf{V}_m = [\mathbf{V}_{(1)}, \ldots, \mathbf{V}_{(m)}]$ whose columns form an orthonormal basis of $\mathbb{K}_m(A, B)$

```
Compute the reduced QR decomposition V_(1) R_B = B
for j = 1, ..., m do
    W = A V_(j)
    for i = 1, ..., j do
        H_(ij) = V_(i)^T W
        W = W − V_(i) H_(ij)
    end for
    Compute the reduced QR decomposition V_(j+1) H_(j+1,j) = W
end for
```

arithmetic, where there may be a numerical loss of linear independence. As such, the above algorithm must be modified. Deflation strategies have been explored and may be required in these special circumstances. While it is good to be aware of such potential issues, in our numerical examples this did not pose a problem, and so we proceed under the assumption of no numerical loss of linear independence for ease of exposition and refer the reader to [31] for details on how to proceed when such issues arise.

The matrices $\mathbf{H}_{(ij)}$ are gathered in the block matrix $\mathbf{H}_m$, i.e.,

$$\mathbf{H}_m = \begin{bmatrix} \mathbf{H}_{(11)} & \ldots & \mathbf{H}_{(1m)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{(m1)} & \ldots & \mathbf{H}_{(mm)} \end{bmatrix}.$$

The block Krylov subspaces clearly also satisfy the nested properties

$$\mathbb{K}_m \subseteq \mathbb{K}_{m+1} \text{ and } A\mathbb{K}_m \subseteq \mathbb{K}_{m+1}, \tag{2.7}$$

and one has the *block Arnoldi relation*

$$A\mathbf{V}_m = \mathbf{V}_{m+1}\underline{\mathbf{H}}_m \tag{2.8}$$

$$= \mathbf{V}_m\mathbf{H}_m + \mathbf{V}_{(m+1)}\mathbf{H}_{(m+1,m)}\mathbf{E}_m^T, \tag{2.9}$$

where $\mathbf{E}_m$ denotes the last k columns of the $mk \times mk$ identity matrix. We again emphasize that the relation (2.8) follows in theory from the nested properties (2.7). We define the Rayleigh quotient matrix in an analogous fashion and see that it is again available as a byproduct of the block Arnoldi algorithm, i.e.,

$$\mathbf{V}_m^T A\mathbf{V}_m = \mathbf{H}_m.$$

It is worth noting that we can also gather all solutions into a matrix $X = [x_1, \ldots, x_k]$ which gives us our first example of a linear matrix equation

$$AX = B,$$

and a Galerkin projection strategy analogous to FOM could be pursued, as well as more sophisticated methods such as block variants of GMRES. As our primary interest is in different kinds of matrix equations, we do not pursue this further, and instead refer the interested reader to [31, 63] for more details on block methods and how they are used to treat (2.5).

2.2 Numerical solution of Sylvester equations

A canonical example of the type of matrix equation to be solved in this thesis is *Sylvester's equation*

$$AX + XB + EF^T = 0 \tag{2.10}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{\ell\times\ell}$ are nonsingular, sparse matrices, $E \in \mathbb{R}^{n\times k}$, $F \in \mathbb{R}^{\ell\times k}$ for the unknown $X \in \mathbb{R}^{n\times\ell}$. A classical sufficient condition for existence of a solution is $\Lambda(A) \cap \Lambda(-B) = \emptyset$ [23]. We postpone a discussion of the sizes of n, ℓ and k and instead focus on a plausible first attempt for solving

(2.10). We note that the operator $\mathcal{A}(X) = AX + XB$ is linear in the argument X, so that in theory one could convert (2.10) into a linear system. This can be made explicit by noting that the vectorization operator and Kronecker product (given by Definitions 1.1 and 1.2, respectively) satisfy the relation [35]

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \tag{2.11}$$

so that (2.10) is equivalent to

$$(I \otimes A + B^T \otimes I)\text{vec}(X) = -\text{vec}(EF^T). \tag{2.12}$$

which is a linear system with unknown $\text{vec}(X)$. The size of the coefficient matrix in (2.12) is $n\ell \times n\ell$.

First, assume that this is a small-scale problem. Solving (2.12) via a direct method such a Gaussian elimination would entail $\mathcal{O}\left((n\ell)^3\right)$ work. An obvious idea is to try take advantage of matrix decompositions of A and B, such as reduction to Schur or Hessenberg-Schur form. Computing any pair of these decompositions for both A and B would entail $\mathcal{O}\left(n^3 + \ell^3\right)$ work. As such, solvers were developed that maintain this complexity and are described in [5, 25]. We therefore see in the small-scale case that considering the matrix structure that is present in the problem is worthwhile.

For the large-scale problem, the seminal idea for a low-rank approximation may be found in [61] and was further studied and developed in [36, 37, 67]. They are based on truncating an analytic solution and using what results to motivate the definition of a subspace which is then used to perform a Galerkin projection. Specifically, under the assumption that A and B are both stable, the solution of (2.10) can be given by

$$X = \int_0^\infty e^{At} E F^T e^{Bt} dt. \tag{2.13}$$

Writing $e^z \approx p(z)$ for some $p \in \mathbb{P}_{m-1}$, replacing $e^{At} \approx p(At)$ in (2.13), and

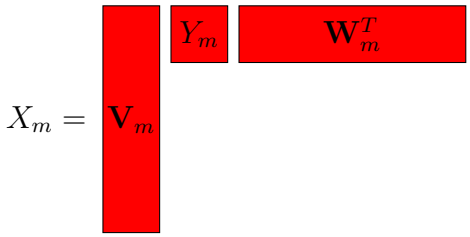


Figure 2.1: Depiction of an element of the space $\mathbb{S}_m$.

truncating the integral at some finite upper bound s yields

$$X = \int_0^\infty e^{At} E F^T e^{Bt} dt \tag{2.14}$$

$$\approx \int_0^s p(At) E F^T p(Bt) dt \tag{2.15}$$

$$= \mathbf{V}_m \tilde{X} \mathbf{W}_m^T \tag{2.16}$$

for an appropriate $\tilde{X}$, where

$$\text{range}\left(\mathbf{V}_m\right) = \mathbb{K}_m\left(A, E\right) \text{ and } \text{range}\left(\mathbf{W}_m\right) = \mathbb{K}_m\left(B^T, F\right). \tag{2.17}$$

The above calculation motivates a space to use for Galerkin projection. Given any two subspaces $\mathbb{V}$ and $\mathbb{W}$ with $\dim(\mathbb{V}) = r_V$ and $\dim(\mathbb{W}) = r_W$, we choose two matrices $\mathbf{V}$ and $\mathbf{W}$ whose columns satisfy $\text{range}\left(\mathbf{V}\right) = \mathbb{V}$ and $\text{range}\left(\mathbf{W}\right) = \mathbb{W}$. One then defines

$$\mathbb{S}\left(\mathbb{V}, \mathbb{W}\right) = \left\{\mathbf{V} Y \mathbf{W}^T : Y \in \mathbb{R}^{r_V \times r_W}\right\}.$$

Our previous calculations motivate us to consider

$$\mathbb{S}_m = \mathbb{S}\left(\mathbb{K}_m\left(A, E\right), \mathbb{K}_m\left(B^T, F\right)\right). \tag{2.18}$$

We emphasize that for $m \ll n$, an element of $\mathbb{S}_m$ entails $\mathcal{O}\left(n\right)$ storage. A depiction of a typical element is given in Figure 2.1. We use $\mathbb{S}_m$ as a space to perform a Galerkin projection, i.e., we consider the set of *all* matrices of the form (2.16). For $X_m \in \mathbb{S}_m$ the associated residual is defined as

$$R_m = A X_m + X_m B + E F^T$$

and $X_m = \mathbf{V}_m Y_m \mathbf{W}_m^T$ is chosen so that

$$R_m \perp_F S \text{ for all } S \in \mathbb{S}_m, \tag{2.19}$$

where we emphasize that the orthogonality is with respect to the Frobenius inner product. In full analogy with the FOM method for linear systems, this results in a smaller Sylvester equation for the matrix Y_m, since (2.19) can be shown to be equivalent to

$$\left(\mathbf{V}_m^T A \mathbf{V}_m\right) Y_m + Y_m \left(\mathbf{W}_m^T B \mathbf{W}_m\right) + \left(\mathbf{V}_m^T E\right) \left(\mathbf{W}_m^T F\right)^T = 0. \tag{2.20}$$

Note that A is stable and recall that for any orthonormal matrix $\mathbf{V}$ one has $\Lambda(\mathbf{V}^T A \mathbf{V}) \subseteq W(A) \subset \mathbb{C}^-$. This can be used to furnish a proof that stability of the coefficient matrices is a sufficient condition for existence and uniqueness of the solution to the projected equation (2.20).

As far as stopping criteria are concerned, a tolerance τ is chosen and the algorithm is stopped when the *backward error* falls below this tolerance, i.e., when

$$\rho_m := \frac{\|R_m\|_F}{(\|A\|_F + \|B\|_F)\|X_m\|_F + \|E\|_F \|F\|_F} \leq \tau. \tag{2.21}$$

We would like an inexpensive way to calculate the quantity on the left-hand side of the above inequality. Norms involving data are precomputed and cheap since A and B are sparse and E and F have few columns. Since the Frobenius norm is invariant with respect to multiplication by orthogonal matrices, one has $\|X_m\|_F = \|Y_m\|_F$. The nested properties of the block Krylov subspaces (2.7) extend naturally, as

$$\mathbb{S}_m \subseteq \mathbb{S}_{m+1} \text{ and } \mathcal{A}(\mathbb{S}_m) \subseteq \mathbb{S}_{m+1}$$

so that $R_m \in \mathbb{S}_{m+1}$, and thus calculating this quantity should again be available as a byproduct of building the spaces.

A quick calculation shows that

$$R_m = [\mathbf{V}_m, A\mathbf{V}_m] \begin{bmatrix} \mathbf{V}_m^T E (\mathbf{W}_m^T F)^T & Y_m \\ Y_m & 0 \end{bmatrix} \begin{bmatrix} \mathbf{W}_m^T \\ \mathbf{W}_m^T B \end{bmatrix}, \tag{2.22}$$

and it is worthwhile to remark that this calculation holds true for any $\mathbf{V}$ and $\mathbf{W}$. Defining the Rayleigh quotient matrices for each space as $\mathbf{H}_m^A = \mathbf{V}_m^T A \mathbf{V}_m$ and $\mathbf{H}_m^B = \mathbf{W}_m^T B^T \mathbf{W}_m$, the block Arnoldi relation (2.9) implies that

$$[\mathbf{V}_m, A\mathbf{V}_m] = [\mathbf{V}_m, \mathbf{V}_{(m+1)}] \begin{bmatrix} I & \mathbf{H}_m^A \\ 0 & \mathbf{H}_{(m+1,m)}^A \end{bmatrix}$$
$$[\mathbf{W}_m, B^T\mathbf{W}_m] = [\mathbf{W}_m, \mathbf{W}_{(m+1)}] \begin{bmatrix} I & \mathbf{H}_m^B \\ 0 & \mathbf{H}_{(m+1m)}^B \end{bmatrix}.$$

Upon plugging these relations into (2.22) and taking Frobenius norms, a short calculation shows that

$$\|R_m\|_F = \sqrt{\left\|\mathbf{H}_{(m+1,m)}^A \mathbf{E}_m^T Y_m\right\|_F^2 + \left\|\mathbf{H}_{(m+1,m)}^B \mathbf{E}_m^T Y_m\right\|_F^2}. \tag{2.23}$$

where the matrices under the square root are of small sizes. One can proceed by building the block Krylov spaces in (2.17) and solving (2.20) at every iterate in order to monitor the backward error (2.21). The required quantities to solve the projected equation are all available as a byproduct of building the block Krylov spaces.

A crucial component of what makes this method successful is the existence of low-rank solutions to (2.10) for $k \ll n, \ell$. We note that for a given $X_m \in \mathbb{S}_m \subseteq \mathbb{R}^{n\times\ell}$, the associated "core matrix" $Y_m \in \mathbb{R}^{mk\times mk}$. If we assume that this method converges in $m \ll n, \ell$ iterations, then the data required to build and store X_m will come from building $\mathbf{V}_m$ and $\mathbf{W}_m$, and solving and storing Y_m. Since we assume $m, k \ll n, \ell$, the main computational and storage expense will be in building the block spaces in (2.17), which entail $\mathcal{O}(n+\ell)$ work, thus achieving the optimal complexity described in Chapter 1.

Several papers provide theoretical evidence for the existence of low-rank approximations to X, the solution to (2.10) [2, 27, 52]. They proceed by proving bounds on the singular values of X. Without singular value decay, any hope for an accurate low-rank approximation is lost, since if $X = \sum_{i=1}^{\min(n,\ell)} \sigma_i u_i v_i^T$ is a singular value decomposition, a celebrated result (sometimes referred to as the Schmidt-Mirsky theorem) says that a best rank p approximate for X

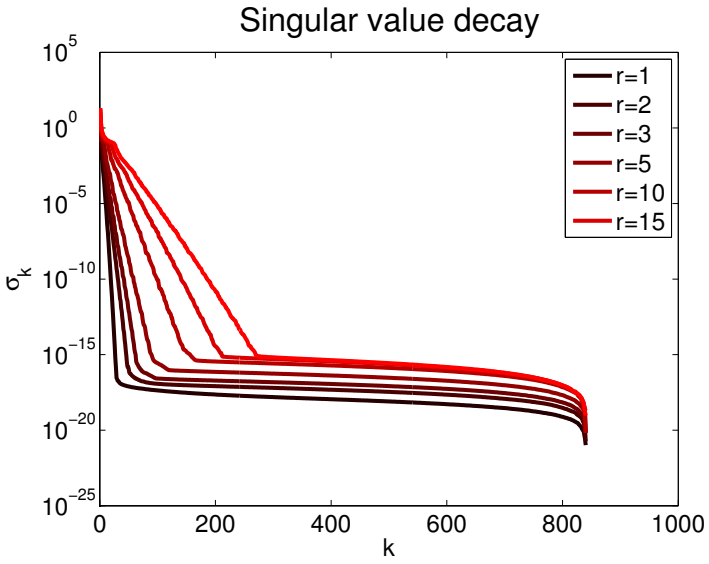


Figure 2.2: Singular value decay of the solution of a Lyapunov equation, with A a 2D discrete Laplacian on a 30×30 grid, and B the first r columns of the identity matrix for varying values of r.

is given by the truncation $X_{(p)} = \sum_{i=1}^{p} \sigma_i u_i v_i^T$ and $\|X - X_{(p)}\|_2 = \sigma_{p+1}$; see, e.g., [26]. We provide numerical evidence of the singular value decay of a small Lyapunov equation and the effect that the rank of the right-hand side plays on this decay in Figure 2.2.

Even with theoretical evidence for the existence of decent low-rank approximations due to singular value decay, a proper method is required to exploit it. The method described in the previous section, while circumventing the "curse of dimensionality"[3], suffers from a lack of robustness with respect to mesh width. That is to say, if A comes from the discretization of partial differential operator, the number of iterations required to achieve a given backward error increases as the mesh width h tends to 0. A convergence analysis for Lyapunov equations was done in [70]. Loosely speaking, it is based on polynomial

[3]We mean this very loosely, in the sense that an optimal linear solver for (2.12) would require $\mathcal{O}(n\ell)$ work, and under the assumptions presented the method requires $\mathcal{O}(n+\ell)$. However, there is a sense in which this approach avoids what is truly meant by the curse of dimensionality, as the same technique can extended naturally to solve linear systems of the form $I \otimes I \otimes A + I \otimes A \otimes I + A \otimes I \otimes I$ and their higher order analogues, as shown in [40].

approximation to the exponential occurring in (2.13) over the domain of integration. This sheds light on a shortcoming of the method, as polynomials are incapable of uniformly approximating the exponential function on the entire positive real axis.

Along with the lack of robustness with respect to conditioning of A, the Galerkin projection method described above suffers from growing memory demands at each iterate (even for symmetric A). Such phenomena have been observed when using Krylov subspace methods to solve linear systems of the form (2.1). For the linear system case, techniques that can be used to alleviate these negative effects are still an active area of research. Preconditioning is a crucial component of solving many challenging linear systems [10], and for certain problems preconditioners can be designed that deliver convergence in a number of iterations that is independent of the mesh width of the underlying discrete operator. To alleviate growing memory demands, a trivial method is to simply restart the iteration, and a more sophisticated version of this reuses information from the space built before restarting [50].

We assume that the reader is relatively familiar with these notions and provide short, heuristic arguments to show that neither extends trivially to the Galerkin projection procedure described above. If one can approximately invert A by performing a linear solve with a matrix M, it may seem natural to multiply (2.10) on the left by M^{-1}, since this is what is done for linear systems. This results in

$$M^{-1}AX + M^{-1}XB + M^{-1}EF^T = 0$$

which is no longer a Sylvester equation. We thus either require new algorithms for a new matrix equation or a restoration of the Sylvester structure. If we attempt the latter by changing to the new variable $\tilde{X} = M^{-1}X$, this results in

$$M^{-1}AM\tilde{X} + \tilde{X}B + M^{-1}EF^T = 0.$$

While this restores the Sylvester equation structure, unfortunately, this does not change the spectrum of the first coefficient matrix, as $M^{-1}AM$ and A

are similar! A standard goal in preconditioning a linear system is to have the spectrum of the preconditioned system cluster around a point, and the proposed approach does not change the spectrum of the first coefficient matrix. In fact, one can show that $\Lambda(I \otimes A + B^T \otimes I) = \Lambda(I \otimes M^{-1}AM + B^T \otimes I)$, so that upon viewing this equation at the linear system level, the spectrum is unchanged. Hence the system is not really preconditioned at all.

If one seeks to restart, suppose that there is some initial approximation $X_0 = U_0 Y_0^T$ for which one seeks a correction $\tilde{X}$ such that $X_0 + \tilde{X}$ is the true solution. Then $\tilde{X}$ solves the Sylvester equation

$$\begin{aligned} 0 &= A(X_0 + \tilde{X}) + (X_0 + \tilde{X})B + EF^T \\ &= A\tilde{X} + \tilde{X}B + [E, U_0, AU_0] \begin{bmatrix} F^T \\ Y_0 B \\ Y_0 \end{bmatrix} \end{aligned}$$

which shows that one would need to build (among other things) a block Krylov subspace of the form $\mathbb{K}_m(A, [E, U_0, AU_0])$. This is not feasible since U_0 potentially has many columns.

Though intriguing, this approach would not prove competitive with low-rank ADI based methods such as those described in [43] for large, ill-conditioned problems. Over fifteen years would pass until it was realized that methods based on Galerkin projection could be competitive, or even make the claim to be the state-of-the-art, if the spaces to project into were rich enough to produce a quality approximation.

2.3 Extended and rational Krylov subspace methods

We begin with a definition of the extended Krylov subspace, which shall play a large role in the results of this thesis.

Definition 2.3. The m^{th} *extended Krylov subspace* is defined as

$$\mathbb{EK}_m(A, E) = \mathbb{K}_m(A, E) + \mathbb{K}_m\left(A^{-1}, A^{-1}E\right). \tag{2.24}$$

An equivalent definition is

$$\mathbb{EK}_m(A, E) = \mathbb{K}_{2m}\left(A, A^{-m}E\right).$$

We denote an extended space by $\mathbb{EK}_m$ when A and E are clear from the context. The extended Krylov subspace (2.24) was originally introduced in [19] and was used to approximate functions of matrix applied to a vector, i.e., $f(A)b$. The equivalent definition shows that these are indeed Krylov subspaces, albeit with a modified right-hand side. The extended spaces also satisfy the nested properties

$$\mathbb{EK}_m \subseteq \mathbb{EK}_{m+1} \text{ and } A\mathbb{EK}_m \subseteq \mathbb{EK}_{m+1}. \tag{2.25}$$

As such, the approach described in the previous section is applicable and many desirable properties transfer over. One nontrivial detail that requires resolution is obtaining the Rayleigh quotient matrix. We briefly describe an approach for solving (2.10) with extended Krylov subspaces. This was originally done for the Lyapunov equation in [68], and details for applying this to the Sylvester equation can be found in [34]. We begin by presenting an algorithm that builds an orthonormal basis of (2.24), taken from [68].

We highlight a few differences between the standard block algorithm (Algorithm 2.2) and extended block Arnoldi algorithm (Algorithm 2.3). The initial reduced QR decomposition computes an orthonormal basis for $\mathbb{EK}_1$, and all blocks $\mathbf{V}_{(i)}$ now have $2k$ columns instead of k. To build the space in both directions, each block is split in two pieces with k columns as $\mathbf{V}_{(j)} = [\mathbf{V}_{(j)}^{[1]}, \mathbf{V}_{(j)}^{[2]}]$, and the space is grown by applying A and A^{-1} to the left and right pieces, respectively. We emphasize that the reader should not be troubled by the presence of A^{-1}, since it is not this (typically dense) matrix that is required, nor a factorization, but rather its action on tall matrices. Specifically, one only needs to compute quantities such as $A^{-1}E$ and $A^{-1}\mathbf{V}_{(j)}^{[2]}$, each of which has k

Algorithm 2.3 Extended Arnoldi

Input: A, E, m

Output: Matrix $\mathbf{V}_m = [\mathbf{V}_{(1)}, \ldots, \mathbf{V}_{(m)}]$ whose columns form an orthonormal basis of $\mathbb{EK}_m(A, E)$

```
Compute the reduced QR decomposition V_(1) R = [E, A^{-1}E]
for j = 1, ..., m do
    W = [A V_(j)^[1], A^{-1} V_(j)^[2]], where V_(j) = [V_(j)^[1], V_(j)^[2]]
    for i = 1, ..., j do
        H_(ij) = V_(i)^T W
        W = W − V_(i) H_(ij)
    end for
    Compute the reduced QR decomposition V_(j+1) H_(j+1,j) = W
end for
```

columns, and the computation may be done column by column. Hence all that is required is an efficient means of solving $Ax = b$, which would depend strongly on A. If an optimal solver such as multigrid is available [77], the stated algorithm maintains the desired optimal complexity. Our experiments consisted primarily of two-dimensional problems, so that sophisticated direct solvers such as those found in [17] were applicable, but in practice an optimal solver is enough to guarantee that (2.24) can be built efficiently.

For extended Krylov subspaces, the nested properties (2.25) guarantee an Arnoldi-type relation, which can be written explicitly as

$$A\mathbf{V}_m = \mathbf{V}_m\mathbf{T}_m + \mathbf{V}_{(m+1)}\mathbf{T}_{(m+1,m)}\mathbf{E}_m^T. \tag{2.26}$$

Here $\mathbf{E}_m$ denotes the last $2k$ columns of the $2mk \times 2mk$ identity matrix and $\mathbf{T}_m = \mathbf{V}_m^T A \mathbf{V}_m$ is again a Rayleigh quotient matrix. We emphasize that $\mathbf{T}_m \neq \mathbf{H}_m$, i.e., the Rayleigh quotient matrix is not immediately obtained from the orthogonalization coefficients. A derivation of how to derive the Rayleigh quotient matrix without any $\mathcal{O}(n)$ or $\mathcal{O}(\ell)$ computations is given in Appendix B, which in theory is possible due to (2.25).

We follow [34, 68] and perform Galerkin projection with the space

$$\mathbb{S}_m = \mathbb{S}\left(\mathbb{EK}_m(A, E), \mathbb{EK}_m(B^T, F)\right), \tag{2.27}$$

so that one needs to build both extended Krylov subspaces appearing in (2.27). If we denote the $(1,1)$ block of R appearing in Algorithm 2.3 as $R_{(11)}$, one can show that the equation that results from Galerkin projection of (2.10) onto the space defined (2.27) results in

$$\mathbf{T}_m^A Y_m + Y_m (\mathbf{T}_m^B)^T + \left(R_{(11)}^A (R_{(11)}^B)^T\right) \otimes (e_{2m} e_{2m}^T) = 0. \tag{2.28}$$

The space again satisfies $\mathcal{A}(\mathbb{S}_m) \subseteq \mathbb{S}_{m+1}$ due to (2.25), so we expect a representation for the residual in the bases that we have constructed. In fact, (2.22) follows for any bases $\mathbf{V}$ and $\mathbf{W}$, and the cheap calculation (2.23) follows from this expression and an Arnoldi relation (2.26), so that an analogous expression results and the residual may be cheaply calculated after solving the small equation (2.28).

This method has emerged as a competitor for the state-of-the-art for solving (2.10) in the large-scale case. Given the success of the Galerkin projection approach, a minimum residual method for Lyapunov equations is described in [45]. A convergence analysis was done in [38], which sheds light on how using powers of A^{-1} alleviate some of the difficulties with polynomial approximation, and this is due to superior properties of rational approximation. In fact, a similar derivation as was done in (2.14)–(2.16) can be used to motivate this approach and provides a first step towards a convergence analysis. Instead of using a polynomial of degree m, one uses a *Laurent polynomial* $q(z) = \sum_{k=-m}^{m-1} \alpha_k z^k$, and this provides a derivation of the method described.

As a viable alternative, *rational Krylov subspaces* also play a role in this thesis. They were originally introduced in [55] and further developed in [56, 57, 58, 59] for eigenvalue computations.

Definition 2.4. Given a vector of shifts $\mathbf{s} = [s_1, \ldots, s_{m-1}] \in \overline{\mathbb{C}}^{m-1}$, the m^{th} *rational Krylov subspace* is defined as

$$\mathbb{RK}_m(A, E, \mathbf{s}) = \text{range}\left([E, (A - s_1 I)^{-1} E, \ldots, (A - s_{m-1} I)^{-1} E]\right). \tag{2.29}$$

Algorithm 2.4 Rational Arnoldi

Input: A, b, m, vector of shifts $\mathbf{s}$

Output: Matrix $V_m = [v_1, \ldots, v_m]$ whose columns form an orthonormal basis of $\mathbb{RK}_m(A, b, \mathbf{s})$

```
1: β = ‖b‖_2, v_1 = b/β
2: for j = 1, ..., m do
3:     w = (I − A/s_j)^{-1} A v_j
4:     for i = 1, ..., j do
5:         h_ij = v_i^T w
6:         w = w − h_ij v_i
7:     end for
8:     h_{j+1,j} = ‖w‖_2
9:     v_{j+1} = w/h_{j+1,j}
10: end for
```

Defining $q_{m-1}(z) = \prod_{j=1}^{m-1}(1 - z/s_j)$, one can show that

$$\mathbb{RK}_m(A, E, \mathbf{s}) = q_{m-1}(A)^{-1}\mathbb{K}_m(A, E)$$

so we adopt the convention that an infinite shift $s_m = \infty$ increments the polynomial degree of the space but does not introduce any extra poles. In this case $q_{m-1}(z) = q_m(z)$ so that by the above

$$A\mathbb{RK}_m(A, E, \mathbf{s}) \subseteq \mathbb{RK}_{m+1}(A, E, \mathbf{s}),$$

i.e., by choosing an infinite pole we can recover the nested property and several of its desirable consequences.

For simplicity of exposition we present Algorithm 2.4 for building an orthonormal basis of (2.29) when $E = b$ is a column vector. An implementation for block spaces with reorthogonalization is described in Appendix A. It is again worth emphasizing that one only requires an efficient method of solving $(A - s_j I)x = v$, which may also be done by some sophisticated iterative method for the coefficient matrix A. Defining $D_m = \operatorname{diag}\left(1, s_1^{-1}, \ldots, s_{m-1}^{-1}\right)$,

as well as

$$\underline{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m} e_m^* \end{bmatrix} \text{ and } \underline{K}_m = \begin{bmatrix} I_m + H_m D_m \\ h_{m+1,m} \xi_m^{-1} e_m^T \end{bmatrix}$$

one can prove that the following *rational Arnoldi decomposition* holds

$$AV_{m+1}\underline{K}_m = V_{m+1}\underline{H}_m$$

and when $s_m = \infty$ it holds that

$$AV_m K_m = V_{m+1}\underline{H}_m. \tag{2.30}$$

Assuming that K_m is invertible, upon left multiplication of (2.30) by V_m^T one can recover the Rayleigh quotient matrix $V_m^T A V_m = H_m K_m^{-1}$ as a byproduct of the algorithm at negligible cost.

It was noticed that rational Krylov subspaces were powerful tools for approximating functions of a matrix applied to a vector, and a good historical account of the development of this theory along with relevant theoretical and algorithmic topics can be found in [32]. An adaptive shift selection strategy was proposed in [21] to approximate the matrix exponential of a symmetric matrix applied to a vector. This strategy was modified in [22] for non-symmetric matrices and shown to be effective for the Lyapunov equation. An analysis of this approach and comparison with ADI-based methods for Lyapunov equations was done in [20]. Convergence analysis for the Sylvester equation may be found in [6]. Rational Krylov subspaces constitute a fascinating topic that is very active at the moment of writing, and are good to be aware of when considering an alternative approach to extended Krylov spaces.

2.4 Low-rank classical and Krylov subspace methods

We next describe a method which pairs nicely with the methods described in the previous section. They are known as *low-rank Krylov subspace methods*

Algorithm 2.5 Preconditioned conjugate gradients

Input: S.p.d. matrix A, right-hand side b, preconditioner M

Otuput: Approximation solution u_m of $Ax = b$

1: Set $r_0 = b$, solve $Mz_0 = r_0$, set $p_0 = z_0$, $q_0 = Ap_0$
2: **for** $j = 0, 1, 2, \ldots$ **do**
3: $\quad \alpha_j = \langle r_j, z_j \rangle / \langle q_j, p_j \rangle$
4: $\quad u_{j+1} = u_j + \alpha_j p_j$
5: $\quad r_{j+1} = r_j - \alpha_j q_j$
6: $\quad$ Solve $Mz_{j+1} = r_{j+1}$
7: $\quad \beta_j = \langle r_{j+1}, z_{j+1} \rangle / \langle r_j, z_j \rangle$
8: $\quad p_{j+1} = z_{j+1} + \beta_j p_j$
9: $\quad q_{j+1} = Ap_{j+1}$
10: **end for**

and were originally described in [41] to treat parametrized linear systems. We give a description in the context of linear matrix equations, describing first the preconditioned conjugate gradients methods and then its low-rank modification. As a simple motivating example, we consider the numerical solution of the Poisson equation on the unit square with a separable source[4] that is symmetric in the x and y variables and homogeneous Dirichlet boundary conditions, i.e.,

$$\begin{aligned} -\Delta u &= f(x)f(y) \quad && \text{in } [0,1] \times [0,1] \\ u &= 0 && \text{on } \{0,1\} \times [0,1] \cup [0,1] \times \{0,1\}. \end{aligned}$$

We discretize using finite differences (see, e.g., [42]) and chose a uniform grid with n points in each direction, i.e., set $h = \frac{1}{n+1}$ and $x_j = y_j = jh$ for $j = 1, \ldots, n$. We seek to approximate $u(x_i, y_j)$ for each $i, j = 1, \ldots, n$ and gather these approximation in a matrix U such that $U_{ij} \approx u(x_i, y_j)$. The values of the source term at the grid points are gathered in the matrix F whose (i, j)

[4]The technique to be described can be modified to work for smooth sources that can be separably expanded as $f(x, y) = \sum_{i=0}^{\infty} f_i(x) g_i(y)$, perhaps by the `svd` function of Chebfun [76], and then truncated to desirable accuracy.

entry is $F_{ij} = f(x_i)f(y_j)$. We build the two-dimensional discrete Laplacian from its one-dimensional counterpart. The standard one-dimensional 3-point stencil results in the classic tridiagonal matrix

$$A_1 = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 2 \end{bmatrix}.$$

and the discrete system associated with the standard two-dimensional 5-point stencil has the Kronecker product structure

$$A = I_n \otimes A_1 + A_1 \otimes I_n. \tag{2.31}$$

The typical approach is to vectorize (or reshape[5]) U and F obtain a linear system

$$Au = b \tag{2.32}$$

where $u = \mathrm{vec}(U)$ and $b = \mathrm{vec}(F)$. This system is symmetric positive definite, so a basic iterative solver for (2.32) might consist of conjugate gradients preconditioned by Jacobi, i.e., using the preconditioner $M = \mathrm{diag}\,(A) = \frac{4}{h^2}I$. We state the preconditioned conjugate gradients as Algorithm 2.5 for convenience.

We shall see that the entire algorithm can be described at the matrix equation level, and with some slight modifications this perspective enables significant computational gains. Our point of departure from the classical method is an observation on the structure of F. Note that the separability of f implies that the matrix F is of rank one, since defining $\vec{f} = [f(x_1), \dots, f(x_n)]^T$ yields $F = \vec{f}\vec{f}^T$. Hence the dense matrix F, having n^2 entries, can be implicitly represented with $\mathcal{O}(n)$ storage. The preconditioner may also be described at the matrix level, since for any $X \in \mathbb{R}^{n\times n}$ one has $\mathcal{M}^{-1}(X) = \frac{h^2}{4}X$. Following the first line of the algorithm, if we define $R_0 = F$ and solve $\mathcal{M}(Z_0) = R_0$,

[5]We borrow this terminology from MATLAB; note $\mathrm{vec}(X) =$ `reshape(X,[],1)`.

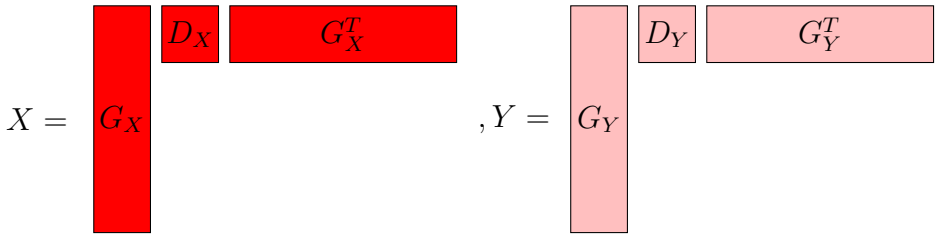


Figure 2.3: Depiction of two low-rank, symmetric matrices.

we see that the low-rank structure may be preserved, along with symmetry and positive semi-definiteness, since setting $Z_0 = (\frac{h}{2}\vec{f})(\frac{h}{2}\vec{f})^T$. We continue by setting $P_0 = Z_0$, though this is done implicitly, as we only store a low-rank factor of $P_0 = p_0 p_0^T$. Our experience with the Kronecker product and vectorization operator suggest to us that we use (2.11) to view (2.31) as a matrix equation

$$\mathcal{A}(U) = F,$$

where $\mathcal{A}(U) = A_1 U + U A_1$. The operator $\mathcal{A}$ also preserves low-rank structure and symmetry, though the rank now increases and definiteness is lost, since a quick calculation shows

$$Q_0 := \mathcal{A}(P_0) = [p_0, A_1 p_0] \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_0^T \\ p_0^T A_1^T \end{bmatrix}.$$

This motivates representing an arbitrary low-rank symmetric matrix X as $X = G_X D_X G_X^T$ for $G_X \in \mathbb{R}^{n\times k}$ and $D_X = D_X^T \in \mathbb{R}^{k\times k}$ with $k \ll n$. A graphical depiction is given in Figure 2.3. The operator $\mathcal{A}$ preserves such a structure, and the remainder of the algorithm consists of linear combinations of vectors and inner products. Linear combinations preserve this structure as well, since for scalars α, β and another low-rank symmetric matrix $Y = G_Y D_Y G_Y^T$, a quick calculation reveals

$$\alpha X + \beta Y = [G_X, G_Y] \begin{bmatrix} \alpha D_X & 0 \\ 0 & \beta D_Y \end{bmatrix} \begin{bmatrix} G_X^T \\ G_Y^T \end{bmatrix},$$

with a visualization of this operator given in Figure 2.4. Performing the stan-

$$\alpha X + \beta Y = [G_X, G_Y] \begin{bmatrix} \alpha D_X & 0 \\ 0 & \beta D_Y \end{bmatrix} \begin{bmatrix} G_X^T \\ G_Y^T \end{bmatrix} =$$

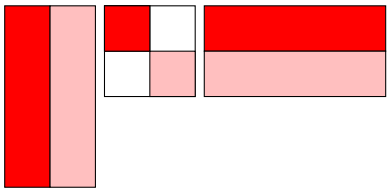
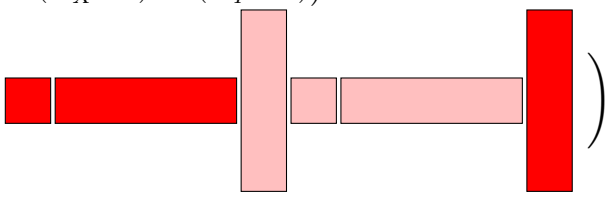


Figure 2.4: Depiction of taking a linear combination of two low-rank matrices while preserving low-rank structure.

$$\langle X, Y \rangle_F = \text{trace}\left(D_X(G_X^T G_Y) D_Y (G_Y^T G_X)\right)$$

$$= \text{trace}\Bigg($$



$$\Bigg)$$

Figure 2.5: Depiction of computing the Frobenius inner product of two low-rank matrices with $\mathcal{O}(n)$ computational complexity.

dard inner product on long vectors corresponds to performing the Frobenius inner product on low-rank representations, since $\langle \text{vec}(S), \text{vec}(T) \rangle_2 = \langle S, T \rangle_F$, and the relation $\text{trace}(ST) = \text{trace}(TS)$ can be used to exploit the low-rank structure and reduce the complexity of this operation. Specifically, note that

$$\begin{aligned} \langle X, Y \rangle_F &= \text{trace}(X^T Y) \\ &= \text{trace}\left((G_X D_X G_X^T)(G_Y D_Y G_Y^T)\right) \\ &= \text{trace}(D_X \underbrace{(G_X^T G_Y)}_{\mathcal{O}(n)} D_Y \underbrace{(G_Y^T G_X)}_{\mathcal{O}(n)}) \end{aligned}$$

with a visual depiction in Figure 2.5. Since the vectorization operator is a linear isometry, if one uses the Frobenius inner product on the low-rank representations and performs all calculations at the matrix level with care to preserve low-rank formats as described, it follows that one will be implementing the first few steps of PCG on (2.32) with $\mathcal{O}(n)$ computational and memory requirements.

An issue that arises is that all quantities incur a growth in rank as the iteration proceeds. This occurs whenever we apply the operator or take a linear

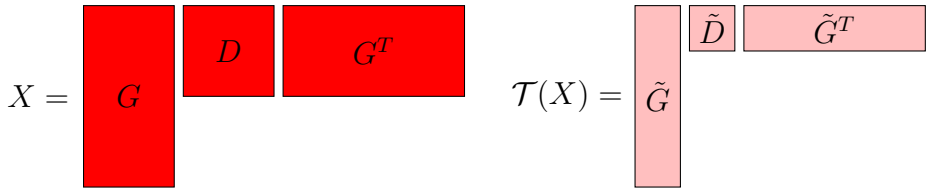


Figure 2.6: Depiction of the truncation operator, compressing the columns of a matrix X.

combination, and the rank of the output quantity will always be larger than the rank of the input quantity. After enough operations one will be storing large, dense matrices. This is resolved by the introduction of a *truncation operator*, denoted by $\mathcal{T}$. The truncation operator accepts the low-rank factors $G \in \mathbb{R}^{n\times k}$ and $D \in \mathbb{R}^{k\times k}$ of a matrix $X = GDG^T$ and returns low-rank factors $\tilde{G} \in \mathbb{R}^{n\times p}$ and $\tilde{D} \in \mathbb{R}^{p\times p}$ of a matrix $\mathcal{T}(X) = \tilde{G}\tilde{D}\tilde{G}^T$. The matrix $\mathcal{T}(X)$ is such that

$$\|X - \mathcal{T}(X)\|_F \leq \tau \|X\|_F \tag{2.33}$$

for some prescribed truncation tolerance τ, and the integer $p \leq k$ is as small as possible while still achieving this desired tolerance. The truncation operator is implemented by computing a reduced QR decomposition of $G = Q_G R_G$ and an eigenvalue decomposition of the small symmetric matrix $R_G D R_G^T = USU^T$. This can be converted to a singular value decomposition $U\Sigma V^T$ according to $\Sigma = S\text{sign}(S)$ and $V = U\text{sign}(S)$. This singular value decomposition is then truncated to $\tilde{U}\tilde{\Sigma}\tilde{V}^T$ so that $\sqrt{\sigma_{p+1} + \cdots + \sigma_k} \leq \tau\sqrt{\sigma_1 + \cdots + \sigma_k}$, where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_k)$ and p is the smallest integer that makes this inequality hold. One returns $\tilde{G} = G\tilde{U}$ and $\tilde{D} = \tilde{\Sigma}$, and by the invariance of the Frobenius norm under multiplication by unitary matrices, a quick calculation shows that (2.33) indeed holds. The columns of the matrix $\mathcal{T}(X)$ have been "compressed," and for this reason this operation is sometimes referred to as a column compression. The results of this procedure is depicted in Figure 2.6.

Algorithm 2.5 can now be performed at the matrix level, with truncations introduced to manage growth in ranks. The main expense in the truncation is the reduced QR decomposition, which is $\mathcal{O}(n)$, so provided there is no inordi-

Algorithm 2.6 Low-rank preconditioned conjugate gradients

Input: Operator $\mathcal{A}$, preconditioner $\mathcal{M}$, truncation operator $\mathcal{T}$, right-hand side B

Output: Low-rank factor Z_m of approximate solution $X_m = Z_m Z_m^T$ to $B - \mathcal{A}(X)$

Set $R_0 = B$, solve $\mathcal{M}(Z_0) = R_0$, set $P_0 = Z_0$, $Q_0 = \mathcal{A}(P_0)$, $\xi_0 = \langle P_0, Q_0 \rangle_F$
for $j = 0, 1, 2, \ldots$ **do**
  $\omega_j = \langle R_j, P_j \rangle_F / \xi_j$
  $X_{j+1} = X_j + \omega_j P_j$ $\qquad$ $X_{j+1} = \mathcal{T}(X_{j+1})$
  $R_{j+1} = B - \mathcal{A}(X_{j+1})$
  $Z_{j+1} = \mathcal{M}^{-1}(R_{j+1})$
  $\beta_j = -\langle Z_{j+1}, Q_j \rangle_F / \xi_j$
  $P_{j+1} = Z_{j+1} + \beta_j P_j$ $\qquad$ $P_{j+1} = \mathcal{T}(P_{j+1})$
  $Q_{j+1} = \mathcal{A}(P_{j+1})$
  $\xi_{j+1} = \langle P_{j+1}, Q_{j+1} \rangle_F$
end for

nate growth in the ranks, the entire procedure entails $\mathcal{O}(n)$ work per iterate. The low-rank CG algorithm is described in Algorithm 2.6, where we emphasize that all applications of the operator, solves with the preconditioner, linear combinations, and inner products are done in low-rank format as previously described.

The astute numerical analyst may object that the truncation introduces an error, so that any orthogonality or optimality properties of the conjugate gradient method that are due to recursions would be lost, and this is certainly a valid point. In fact, in the seminal paper describing these methods, the authors note that that residual must be explicitly calculated as $R_j = B - \mathcal{A}(X_j)$ for purposes of numerical stability, and this requires a modification of the derivation so that coefficients do not depend on any recursions for the residual. Despite such an objection, the method appears capable of delivering quality approximations while dramatically reducing memory requirements.

The main ingredients for low-rank PCG are a low-rank right-hand side, an operator and preconditioner that accept and output low-rank factors, a careful implementation that performs the required inner products and linear combinations while preserving low-rank structure, and a truncation operator. As such, low-rank versions of many classical and modern methods exist, including stationary iterations, steepest descent, MINRES, BiCGStab, and more. Any short-term recurrence method is a candidate for a low-rank alternative.

CHAPTER 3

CONSTRAINED SYLVESTER EQUATIONS

In this chapter, we present a low-rank solution method for large-scale constrained Sylvester equations. To our knowledge, this is the first use of such techniques on a system of matrix equations. We describe constrained Sylvester equations, review existing approaches for small problems and comment on their difficulties for large problems, present our proposed modification for large-scale problems, and survey some numerical results; see also [66].

3.1 Overview

We consider the following system of matrix equations,

$$A_1X + XA_2 - YC = 0 \tag{3.1}$$

$$XB = 0 \tag{3.2}$$

for given $A_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2 \in \mathbb{R}^{n_2 \times n_2}$, $B \in \mathbb{R}^{n_2 \times p}$ and $C \in \mathbb{R}^{m \times n_2}$, with unknowns $X \in \mathbb{R}^{n_1 \times n_2}$ and $Y \in \mathbb{R}^{n_1 \times m}$, where $p < m \ll \min\{n_1, n_2\}$. We assume that A_1 and A_2 are large, sparse, and nonsingular, and that B, C and CB have full rank. In the following, we refer to this system as a constrained Sylvester equation, with the second equation $XB = 0$ acting as a constraint.

A constrained Sylvester equation may also be viewed as a homogeneous system in the two unknown matrices X and Y, from which it readily follows that the matrix problem has a whole family of solutions. Indeed, by means of the relation between the Kronecker product and the vectorization operator (2.11), the system (3.1)–(3.2) can be rewritten as a very large homogeneous linear system, whose unknown vector contains the columns of X and Y, stacked one below the other. The resulting coefficient matrix for this linear system is of size $n_1(p + n_2) \times n_1(m + n_2)$, and therefore underdetermined; thus when a nontrivial solution exists, this is not unique.

In [4], the authors describe a direct method for solving (3.1)–(3.2) when n_1 and n_2 are small. The method essentially amounts to a change of variables and the formulation of an unconstrained equation of the form (2.10) for the new variable. However, the method requires a full QR factorization of the matrix B, which is computationally expensive for large n_2. To the best of our knowledge, we are unaware of any method in the literature for solving (3.1)–(3.2) in the large-scale setting, at a computational cost and memory requirements that grow only linearly with the problem dimensions n_1 and n_2.

Our approach is a modification of the existing small-scale method. We show that an unconstrained Sylvester equation can be formulated for the original unknown X whose solution automatically satisfies the constraint. The new formulation can be stated as a familiar unconstrained Sylvester equation. We describe how projection-type approaches based on Krylov subspaces can be adapted to effectively handle this new formulation when n_1 and n_2 are large. In particular, since one of the coefficient matrices of the transformed problem is singular and dense, a new strategy for using enriched subspaces (analogous to extended and rational Krylov subspaces) is devised. Our numerical experiments on benchmark problems seem to show that the new formulation can be effectively treated with powerful projection spaces, so that small approximation spaces are required to obtain a rather accurate solution.

3.2 Modification of existing results

In [4], the authors describe a direct method for solving (3.1)–(3.2) when the involved matrices have small dimensions. The proposed procedure transforms the original coupled equations into a single unconstrained Sylvester equation in one variable, which can be solved with available methods. We summarize their formulation in the following result. We include the proof because it is insightful for later developments.

Theorem 3.1. *[4] Let*

$$B = [U_1, U_2] \begin{bmatrix} R_B \\ 0 \end{bmatrix}$$

denote the full QR factorization of B, where $U_1 \in \mathbb{R}^{n_2 \times p}$, $U_2 \in \mathbb{R}^{n_2 \times (n_2 - p)}$, and $R_B \in \mathbb{R}^{p \times p}$, with $p < m$. Let also

$$CU_1 = [Q_1, Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

denote the QR factorization of CU_1, with R having full rank p. Then, a solution pair (X, Y) to (3.1)–(3.2) can be written as $Y = [\hat{Y}_1, \hat{Y}_2]Q^T$ with $\hat{Y}_2$ arbitrary, $X = ZU_2^T$, where $Z \in \mathbb{R}^{n_1 \times (n_2 - p)}$ solves the equation

$$A_1 Z + Z\left((U_2^T A_2 U_2) - (U_2^T A_2 U_1) R^{-1} Q_1^T C U_2\right) = \hat{Y}_2 Q_2^T C U_2, \tag{3.3}$$

and $\hat{Y}_1 = Z(U_2^T A_2 U_1)R^{-1}$.

Proof. The constraint (3.2) implies that any solution X can be written as $X = ZU_2^T$ for some matrix $Z \in \mathbb{R}^{n_1 \times (n_2 - p)}$. The equation (3.1) is then post-multiplied by U_1 and U_2, yielding

$$Z(U_2^T A_2 U_1) = YCU_1, \qquad A_1 Z + Z(U_2^T A_2 U_2) = YCU_2. \tag{3.4}$$

Define $[\hat{Y}_1, \hat{Y}_2] = YQ$ for $\hat{Y}_1 \in \mathbb{R}^{n_1 \times p}$ and $\hat{Y}_2 \in \mathbb{R}^{n_1 \times (m-p)}$. We can use the first equation in (3.4) to express part of Y in terms of Z; namely, since

$$Z(U_2^T A_2 U_1) = YCU_1 = YQQ^T CU_1 = [\hat{Y}_1, \hat{Y}_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = \hat{Y}_1 R,$$

it follows that $\hat{Y}_1 = Z(U_2^T A_2 U_1)R^{-1}$. With this expression, the second equation in (3.4) is used to formulate an unconstrained equation for Z. Since

$$\begin{aligned} YCU_2 = YQQ^TCU_2 &= [\hat{Y}_1, \hat{Y}_2] \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} CU_2 \\ &= \left(Z(U_2^T A_2 U_1)R^{-1}Q_1^T + \hat{Y}_2 Q_2^T \right) CU_2, \end{aligned}$$

it follows that Z must satisfy

$$A_1 Z + Z\left((U_2^T A_2 U_2) - (U_2^T A_2 U_1)R^{-1}Q_1^T CU_2\right) = \hat{Y}_2 Q_2^T CU_2. \tag{3.5}$$

The entries of $\hat{Y}_2$ are not forced by the procedure, and can be chosen arbitrarily. □

We remark that it can be readily shown that the result in Theorem 3.1 is in fact an equivalence; specifically, if $\hat{Y}_2 \in \mathbb{R}^{n_1 \times (m-p)}$ is arbitrary, Z solves (3.5), and $\hat{Y}_1$, X and Y are defined as stated, then (X, Y) satisfies the system (3.1)–(3.2). We also remark that the arbitrariness of $\hat{Y}_2$ reflects the number of degrees of freedom of the system that are a result of underdeterminedness.

A disadvantage of the approach described above is that the use of a full QR factorization of the matrix B is too costly for large n_2, thus making the whole procedure infeasible for large problems. However, in our setting $m, p \ll n_2$ with, in fact, often $m, p \leq 10$. Assuming such conditions on m, p, and n_2, we propose a modification of the above method, showing that the unknown X satisfies a different unconstrained equation, that can be solved numerically without relying on computing the expensive U_2 term. All that is required is a *reduced* QR decomposition of B, which is computationally feasible under our assumptions. It is worth mentioning that in our treatment a full QR factorization of the matrix $CU_1 \in \mathbb{R}^{m \times p}$ is still computed, but due to the extremely limited size of m and p, its cost remains very low.

Factoring out some terms in (3.3) enables one to rewrite this as

$$A_1 Z + ZU_2^T A_2 \left(I - U_1 R^{-1} Q_1^T C\right) U_2 = \hat{Y}_2 Q_2^T CU_2. \tag{3.6}$$

Right-multiplying by the full row rank matrix U_2^T and recalling that $X = ZU_2^T$, we obtain the equivalent equation

$$A_1X + XA_2\left(I - U_1R^{-1}Q_1^TC\right)\Pi = \hat{Y}_2Q_2^TC\Pi, \tag{3.7}$$

where $\Pi = U_2U_2^T$, which is well known to be an orthogonal projector onto null(B^T). Moreover, another right multiplication by U_2 converts (3.7) back to (3.6), so the two are indeed equivalent. This simple transformation enables one to rewrite the new equation in terms of the coefficient matrices of the original equation and projectors based on the problem data, and is summarized in the following theorem.

Theorem 3.2. *With the notation above, suppose (Z, Y) are as in Theorem 3.1. Then the matrix $P = U_1R^{-1}Q_1^TC$ is a projector onto range(B) and orthogonal to range(C^TCB). Moreover, the matrix $X = ZU_2^T$ solves the unconstrained Sylvester equation*

$$A_1X + XA_2(I - P)\Pi = \hat{Y}_2Q_2^TC\Pi, \tag{3.8}$$

and $\widehat{Y}_1 = XA_2U_1R^{-1}$.

Proof. Let $M = U_1R^{-1}$ and $N = C^TQ_1$, so that $P = MN^T$. Then range (M) = range (U_1) = range (B). Moreover, from $N = C^TCB(RR_B)^{-1}$ it also follows that range (N) = range $\left(C^TCB\right)$. Since $N^TM = Q_1^TCU_1R^{-1} = Q_1^TQ_1RR^{-1} = I$, it follows that $P = M(N^TM)^{-1}N^T$, i.e., P is a projector. Equation (3.8) coincides with (3.7) with P as defined. Finally, $\hat{Y}_1 = Z(U_2^TA_2U_1)R^{-1} = XA_2U_1R^{-1}$. □

A solution of the new unconstrained equation enjoys the property of automatically satisfying the constraint condition, as described in the following corollary.

Corollary 3.1. *Suppose that X solves* (3.8)*. Then $XB = 0$.*

Proof. Since $\Pi B = 0$, right-multiplication of (3.8) by B yields $A_1XB = 0$, so the result follows immediately from the nonsingularity of A_1. □

Moreover, notice that the resulting equation (3.8) can now be written as a familiar Sylvester equation

$$\mathbf{A}X + X\mathbf{B} + \mathbf{E}\mathbf{F}^T = 0$$

where we now use bold letters to distinguish constrained and unconstrained Sylvester equations. This is done by writing $\widehat{Y}_2 = \widehat{Y}_{2,1}\widehat{Y}_{2,2}^T$ and

$$\mathbf{A} = A_1, \quad \mathbf{B} = A_2(I-P)\Pi, \quad \mathbf{E} = -\hat{Y}_{2,1}, \quad \mathbf{F} = \Pi C^T Q_2 \widehat{Y}_{2,2}. \tag{3.9}$$

Other factorizations of $\hat{Y}_2 Q_2^T C\Pi = \mathbf{E}\mathbf{F}^T$ could be considered. We found that since $\widehat{Y}_2$ can be chosen arbitrarily, it may be computationally advantageous to choose a rank-one matrix. Indeed, with these choices both $\mathbf{E}$ and $\mathbf{F}$ will be vectors, with potential significant computational and memory savings in the solution of the Sylvester equation in the large-scale case.

We note that the method based on Galerkin projection and standard block Krylov subspaces described in Section 2.2 is immediately applicable to the newly formulated unconstrained Sylvester equation (3.8) for large-scale problems. From a computational standpoint, we note that in solving (3.8), the action of Π on a vector may be computed cheaply, as U_1 can be obtained with $\mathcal{O}(n_2)$ complexity; moreover, Π may be replaced by the complementary projector $I - U_1U_1^T$, whose action only involves $\mathcal{O}(n_2)$ computations. Similarly, the action of $I - P$ to a vector can exploit the low rank of P. The expensive full QR decomposition of B has been replaced with an affordable reduced QR decomposition, and the action of $\mathbf{B}^T$ on a vector can be applied with $\mathcal{O}(n_2)$ complexity since A_2 is sparse and the projectors P and Π are factored as low-rank matrices. Thus, the space $\mathbb{K}_m\left(\mathbf{B}^T, \mathbf{E}\right)$ can be built with a reasonable computational complexity. A nice consequence is that approximate solutions from such a space also enjoy the property of automatically satisfying the constraint.

Theorem 3.3. *Suppose that* $X_m \in \mathbb{S}\left(\mathbb{V}, \mathbb{K}_m\left(\mathbf{B}^T, \mathbf{F}\right)\right)$ *for some* m *and some subspace* $\mathbb{V}$. *Then* $X_m B = 0$.

Proof. Let $\mathbf{W}_m$ denote a matrix whose columns form a basis for $\mathbb{K}_m\left(\mathbf{B}^T, \mathbf{F}\right)$. We have that range $(\mathbf{W}_m) =$ range $\left([\mathbf{F}, \mathbf{B}^T\mathbf{F}, \ldots, (\mathbf{B}^T)^{m-1}\mathbf{F}]\right)$, so

$$\mathbf{W}_m = [\mathbf{F}, \mathbf{B}^T\mathbf{F}, \ldots, (\mathbf{B}^T)^{m-1}\mathbf{F}]W$$

for an appropriate W. One quickly checks that $\mathbf{F}^T B = 0$ and $\mathbf{B}B = 0$, so that we obtain

$$\mathbf{W}_m^T B = W^T \begin{bmatrix} \mathbf{F}^T \\ \mathbf{F}^T\mathbf{B} \\ \vdots \\ \mathbf{F}^T\mathbf{B}^{m-1} \end{bmatrix} B = W^T \begin{bmatrix} \mathbf{F}^T B \\ \mathbf{F}^T\mathbf{B}B \\ \vdots \\ \mathbf{F}^T\mathbf{B}^{m-1}B \end{bmatrix} = 0$$

so that $X_m B = \mathbf{V}\widetilde{X}\mathbf{W}_m^T B = 0$. □

3.3 Construction of enriched spaces

Though the projection method based on standard block Krylov spaces is immediately applicable, building $\mathbb{EK}_m(\mathbf{B}^T, \mathbf{E})$ or $\mathbb{RK}_m(\mathbf{B}^T, \mathbf{E}, \mathbf{s})$ to construct an enriched space for Galerkin projection poses some issues. It is not possible to build the former space due the the presence of the projectors, which makes the $\mathbf{B}$ term singular and hence not invertible. The latter space involves shifted solves which can be chosen to make the shifted matrix nonsingular, but difficulties arise as $\mathbf{B}$ is dense due to the presence of the projectors. It would be infeasible to form this dense matrix and invert it; and so we regard the matrix $\mathbf{B}$ as inaccessible. All that is accessible is the action of this matrix on a vector.

Nevertheless, we shall see that *approximating* a shifted solve with $\mathbf{B}^T$ is possible by use of the Sherman-Morrison-Woodbury formula, which we state for convenience [26]

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}.$$

We would like to compute the the shifted inverse

$$\mathbf{B}_\sigma := (\mathbf{B}^T + \sigma I)^{-1} = (\Pi(I - P^T)A_2^T + \sigma I)^{-1}.$$

The inverse can be explicitly computed by means of the Sherman-Morrison-Woodbury formula. Unfortunately, this procedure still involves computations with large and dense matrices, therefore it requires one further approximation step. We describe a plausible approximation $\widehat{\mathbf{B}}_\sigma \approx \mathbf{B}_\sigma$ as follows, recalling that $\Pi = U_2 U_2^T$. Using the Sherman-Morrison-Woodbury formula, one can write

$$(\mathbf{B}^T + \sigma I)^{-1} = \frac{1}{\sigma} I - \frac{1}{\sigma^2} U_2 \left[I + \frac{1}{\sigma} U_2^T (I - P^T) A_2^T U_2 \right]^{-1} U_2^T (I - P^T) A_2^T.$$

Let us set $P^T = P_1 P_2^T$ with $P_1 = C^T Q_1 R^{-T}$, $P_2^T = U_1^T$, and let $\widehat{P}_1 = (\sigma I + A_2^T)^{-1} P_1$. One more application of the Sherman-Morrison-Woodbury formula gives

$$\begin{aligned}
&\left[I + \frac{1}{\sigma} U_2^T (I - P^T) A_2^T U_2 \right]^{-1} = \\
&= \sigma \left[\sigma U_2^T U_2 + U_2^T (I - P^T) A_2^T U_2 \right]^{-1} = \sigma \left[U_2^T (\sigma I + (I - P^T) A_2^T) U_2 \right]^{-1} \\
&\approx \sigma U_2^T \left[\sigma I + (I - P^T) A_2^T \right]^{-1} U_2 = \sigma U_2^T \left[\sigma I + A_2^T - P^T A_2^T \right]^{-1} U_2 \\
&= \sigma U_2^T [(\sigma I + A_2^T)^{-1} + (\sigma I + A_2^T)^{-1} P_1 \\
&\qquad \cdot (I - P_2^T A_2^T (\sigma I + A_2^T)^{-1} P_1)^{-1} P_2^T A_2^T (\sigma I + A_2^T)^{-1}] U_2 \\
&= \sigma U_2^T \left[I + \widehat{P}_1 (I - P_2^T A_2^T \widehat{P}_1)^{-1} P_2^T A_2^T \right] (\sigma I + A_2^T)^{-1} U_2.
\end{aligned}$$

In general, the approximation in the third line of the above calculation could be very rough. However, since U_2 usually spans almost the whole space, we expect the approximation to be of good quality. Our numerical results seem to confirm this expectation. We therefore obtain

$$\begin{aligned}
(\mathbf{B}^T + \sigma I)^{-1} &\approx \tfrac{1}{\sigma} I - \tfrac{1}{\sigma} U_2 U_2^T [I + \widehat{P}_1 (I - P_2^T A_2^T \widehat{P}_1)^{-1} P_2^T A_2^T] \\
&\qquad \cdot (\sigma I + A_2^T)^{-1} U_2 U_2^T (I - P^T) A_2^T \\
&= \tfrac{1}{\sigma} I - \tfrac{1}{\sigma} \Pi \left[I + \widehat{P}_1 (I - P_2^T A_2^T \widehat{P}_1)^{-1} P_2^T A_2^T \right] \\
&\qquad \cdot (\sigma I + A_2^T)^{-1} \mathbf{B}^T =: \widehat{\mathbf{B}}_\sigma.
\end{aligned}$$

We note that the inner matrix $(I - P_2^T A_2^T \widehat{P}_1)$ is in general very small (namely of order $\mathcal{O}(1)$), so that its inversion is cheap, and could be explicitly formed

once and for all at the beginning of the procedure. Moreover, the application of $\widehat{\mathbf{B}}_\sigma$ requires solving a system with the large and sparse matrix $A_2 + \sigma I$ at each iteration, whose computational cost depends on the sparsity pattern of A_2. Such cost is comparable with what one would be willing to accept when dealing with a regular extended Krylov subspace built using A_2.

Given such a $\widehat{\mathbf{B}}_\sigma$, we then propose to build an approximation of a shifted extended Krylov subspace

$$\widehat{\mathbb{EK}}_m^\sigma := \mathbb{K}_m\left(\mathbf{B}^T, \mathbf{F}\right) + \mathbb{K}_m\left(\widehat{\mathbf{B}}_\sigma, \widehat{\mathbf{B}}_\sigma \mathbf{F}\right). \tag{3.10}$$

This space is formally not an *extended* Krylov subspace, as $\widehat{\mathbf{B}}_\sigma$ is not the inverse of $\mathbf{B}^T + \sigma I$; however, by our previous remarks we expect that it should approximate $\mathbb{EK}_m(\mathbf{B}^T + \sigma I, \mathbf{F})$ well. Our numerical experiments appear to confirm that it is sufficiently rich to produce quality approximations with low space dimensions. We shall refer to it as an augmented Krylov subspace.

We explicitly observe that if $x = \Pi x$, then $\widehat{\mathbf{B}}_\sigma x = \Pi \widehat{\mathbf{B}}_\sigma x$, that is, the application of $\widehat{\mathbf{B}}_\sigma$ preserves the space constraint. Therefore, by imposing the right basis $\mathbf{W}_m$ to be such that $\operatorname{range}(\mathbf{W}_m) = \widehat{\mathbb{EK}}_m^\sigma$, we readily obtain $\mathbf{W}_m^T B = 0$, so that the approximate solution X_m naturally satisfies the constraint also when using the enriched space.

We are left with the selection of the shift σ. A large body of literature is available on the computation of appropriate shifts for rational Krylov subspaces; see, e.g., the references in [22]. Here the situation is somewhat simplified, since a single shift is used. Our numerical experiments have shown that the geometric mean of the real "spectral interval" is particularly well suited for intervals spanning several orders of magnitude, namely

$$\sigma := -(\alpha_1 \alpha_n)^{\frac{1}{2}},$$

where $|\lambda_1| \geq \cdots \geq |\lambda_n|$ are the eigenvalues of A_2, arranged in decreasing order, and $\alpha_j = \operatorname{Re}(\lambda_j)$, $j = 1, \ldots, n$. An estimate of these quantities usually suffices.

Remark 3.1. An obvious generalization of our approach is to consider approximating a rational Krylov subspace. For instance, the space $\widehat{\mathbb{EK}}_m^\sigma$ with

our choice of $\widehat{\mathbf{B}}_\sigma$ can be generalized to an approximation of a regular *rational* Krylov subspace

$$\mathbb{RK}_m(\mathbf{B}^T, \mathbf{F}, \mathbf{s}),$$

where the shift σ_i, $i = 1, \ldots, k$ varies at each iteration, and can be either chosen a-priori, or selected adaptively by means of a greedy algorithm. We refer to [22] and references therein for a more detailed discussion. The associated computational cost may increase significantly, as a different shifted system needs to be solved at each iteration. On the other hand, convergence speed, in terms of number of iterations, may be significantly improved [6], making the trade-off problem dependent. To keep the treatment concise, and because of our satisfactory numerical experiments with $\widehat{\mathbb{EK}}_m^\sigma$, we decided not to pursue this generalization further.

An issue arises when trying to compute the norm of the residual for stopping criteria, since the space (3.10) does not possess an Arnoldi relation. We describe a procedure that incurs some additional overhead per iterate but provides a means of computing $\|R_m\|_F$ without forming this dense matrix. The relation for the residual (2.22) holds for arbitrary basis $\mathbf{V}$ and $\mathbf{W}$, and can be exploited by progressively computing QR decompositions of the outer matrices at each iterate. Specifically, if one computes $[\mathbf{V}_m, \mathbf{AV}_m] = Q_m^{\mathbf{A}} R_m^{\mathbf{A}}$ and $[\mathbf{W}_m, \mathbf{BW}_m] = Q_m^{\mathbf{B}} R_m^{\mathbf{B}}$, it readily follows that

$$\|R_m\|_F = \left\| R_m^{\mathbf{A}} \begin{bmatrix} \mathbf{V}_m^T E (\mathbf{W}_m^T F)^T & Y_m \\ Y_m & 0 \end{bmatrix} (R_m^{\mathbf{B}})^T \right\|_F .$$

In practice, there is a slight twist; since one wants to compute these decompositions as the iterations proceed, typically one computes the following QR decompositions:

$$[\mathbf{V}_{(1)}, \mathbf{AV}_{(1)}, \mathbf{V}_{(2)}, \mathbf{AV}_{(2)}, \cdots, \mathbf{V}_{(m)}, \mathbf{AV}_{(m)}] = Q_m^{\mathbf{A}} R_m^{\mathbf{A}}$$
$$[\mathbf{W}_{(1)}, \mathbf{B}^T\mathbf{W}_{(1)}, \mathbf{W}_{(2)}, \mathbf{B}^T\mathbf{W}_{(2)}, \cdots, \mathbf{W}_{(m)}, \mathbf{B}^T\mathbf{W}_{(m)}] = Q_m^{\mathbf{B}} R_m^{\mathbf{B}},$$

so that $[\mathbf{V}_m, \mathbf{AV}_m] = Q_k^{\mathbf{A}} R_k^{\mathbf{A}} \Omega_m^{(A)}$ and $[\mathbf{W}_m, \mathbf{BW}_m] = Q_m^{\mathbf{B}} R_m^{\mathbf{B}} \Omega_m^{(B)}$ for appropriate permutation matrices $\Omega_m^{(A)}$ and $\Omega_m^{(B)}$. This can be done progressively as

columns are added, so that the entire QR need not be computed at every step. Note that when only one space (in our case, $\mathbb{V}_m$) has an Arnoldi relation, a hybrid approach is possible; so that the progressive QR calculation need only be done for the space that does not possess an Arnoldi relation; this is in fact what was implemented.

Concerning stopping criteria, we observe that the term $\|\mathbf{B}\|_F$ used in (2.21) is not readily available. Therefore, we propose stopping when

$$\widehat{\rho}_m < \tau \qquad \text{with} \quad \widehat{\rho}_m := \frac{\|R_m\|_F}{\|X_m\|_F\|\mathbf{A}\|_F + \|X_m\mathbf{B}\|_F + \|\mathbf{E}\|_F\|\mathbf{F}\|_F} \tag{3.11}$$

for a prescribed tolerance τ; in our experiments we used $\tau = 10^{-12}$. Note that the term $\|X_m\mathbf{B}\|_F = \|\widetilde{X}_m\mathbf{W}_m\mathbf{B}^T\|_F$ can be computed at a reasonable cost and that $\rho_k \leq \widehat{\rho}_k$, so that using this criteria guarantees that the true backward error also falls below this threshold.

An important computational aspect concerns the choice of the matrix $\widehat{Y}_2$. Theorem 3.2 shows that $\widehat{Y}_2 = \widehat{Y}_{2,1}\widehat{Y}_{2,2}^T$ can be chosen freely. Moreover, from (3.9) we deduce that the number of columns of $\widehat{Y}_{2,1}$ and $\widehat{Y}_{2,2}$ determines the block size of the Krylov subspaces generated during the process. Therefore, as already mentioned, a computationally advantageous choice consists in selecting $\widehat{Y}_2$ to be of rank one, so that $\mathbf{E} = \widehat{Y}_{2,1}$ and $\mathbf{F} = \Pi C^T Q_2 \widehat{Y}_{2,2}$ both have a single column. We originally experimented with a larger number of columns for $\widehat{Y}_{2,1}$ and $\widehat{Y}_{2,2}$, however the computational cost of dealing with block methods was higher than the increase in convergence rate. Therefore, unless explicitly stated, in all of our experiments we defined $\widehat{Y}_{2,1}$ and $\widehat{Y}_{2,2}$ to be column vectors, and we chose them to have all components equal to one.

It is also worth mentioning once again that different space dimensions could be used for the right and left spaces. Without extra information on the spectral properties of the matrices, we did not find any special reason to exploit this extra feature, although the code could be easily adapted to handle this case as well. We also emphasize that in the generic case, the standard Krylov subspace after k iterations has dimension rk, whereas an extended or augmented Krylov subspace has dimension $2rk$, if started with a matrix with

r columns.

3.4 Numerical experiments

In this section we report on our numerical experience with the proposed formulation and methods. We experimented with the use of standard and augmented Krylov subspaces. For the sake of clarity we mainly only report the use of the same type of space as left and right spaces (either standard for both $\mathbf{A}$ and $\mathbf{B}^T$ or extended for $\mathbf{A}$ and augmented for $\mathbf{B}^T$). Nonetheless, a number of experiments were also performed with mixed choices, as the code implements all possibilities.

In all examples we defined the matrices A_1 and A_2 so as to have similar large size; unless stated otherwise, the matrices B and C were taken from the dataset associated with A_1 when available. In all plots the approximate backward error in (3.11) is displayed, versus the space dimension as iterations proceed.

Example 3.1. We consider a discretization of the Laplace operator for a variety of dimensions and signs. In all cases, B is the first column of the identity matrix, and C is given by the first five rows of the identity matrix. Let Δ_n be the $n \times n$ matrix stemming from the 5-point stencil finite difference discretization of the Laplace operator on the unit square. The first example (leftmost plot of Figure 3.1) considers $A_1 = n_1\Delta_{n_1}$ and $A_2 = -\Delta_{n_2}$, with $n_1 = 324$ and $n_2 = 400$. Note that the two matrices have eigenvalues on different sides of the complex plane, however the scaling of A_1 avoids any instability problems in the computation. Differences in the two approaches are much more pronounced for $n_1 = 2304$ and $n_2 = 2500$ (middle plot of Figure 3.1), for which the use of the enriched spaces significantly improves convergence, in terms of memory used. The computational cost of solving with a shifted version of A_2 is negligible. For larger matrix dimensions, the gap between the two curves substantially increases. Finally, the rightmost

plot of Figure 3.1 shows the performance for the larger case, where however now $A_1 = -n_1\Delta_{n_1}$, so that both A_1 and A_2 have the same sign. Once again, the augmented space is able to capture good spectral information soon during the iterations, leading to fast convergence. Although we shall not pursue this issue further, we observe that the final subspace dimension of the enriched space method seems to be rather insensitive to the problem dimension, and this is typical of shift-and-invert Krylov subspaces when applied to functionals such as the Laplace operator [78].

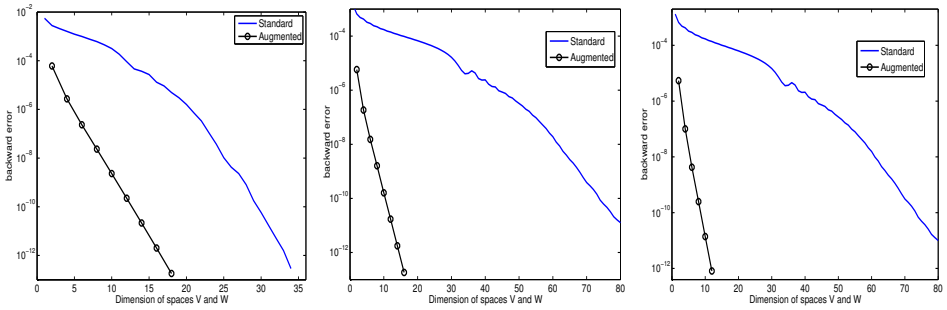


Figure 3.1: Example 3.1. Convergence history of standard and augmented Krylov subspace solvers using 2D discrete Laplacians of varying signs and mesh widths as problem data. In the left plot, $n_1 = 324$ and $n_2 = 400$, while in the middle and right plots, $n_1 = 2304$ and $n_2 = 2500$.

Example 3.2. We consider the solution of (3.8), where A_2, B and C stem from the non-symmetric CHIP dataset of the Oberwolfach collection [39]. The matrix A_1 was obtained as a finite difference discretization of the 2D Laplace operator, scaled so as to have the same Frobenius norm as A_2. The two matrices A_1 and A_2 have size $n_1 = 19881$ and $n_2 = 20082$, respectively. The input matrix B has a single column, whereas C has five rows. Figure 3.2 reports on our experiments comparing two choices of projection spaces: standard Krylov subspace for both $\mathbf{A}, \mathbf{B}^T$, and extended (augmented) Krylov subspaces for $\mathbf{A}$ ($\mathbf{B}^T$). This experiment fully confirms our findings on a more realistic problem than the one of the previous example.

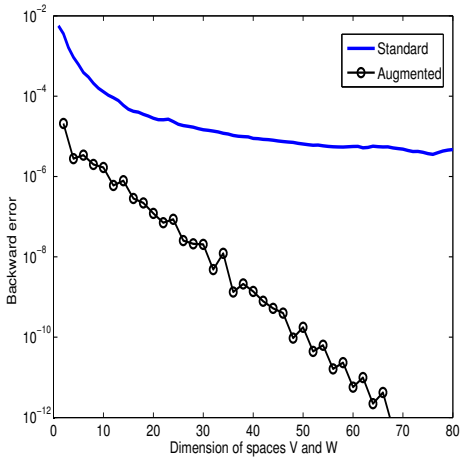


Figure 3.2: Example 3.2. Convergence history of standard/standard and extended/augmented Krylov subspace solvers on the CHIP problem from the Oberwolfach benchmark collection.

Example 3.3. We consider for A_2 the 9669×9669 non-symmetric and stable matrix FLOW from the Oberwolfach benchmark collection [39], with B having a single column, and C having five rows. A discretization of the Laplace operator Δu was used for A_1, giving rise to a 9604×9604 stable matrix. The performance of standard and enriched space methods in terms of space dimension is reported in the left plot of Figure 3.3. For this problem we further analyze the importance of selecting enriched spaces for *both* matrices $\mathbf{A}$ and $\mathbf{B}^T$. In the right plot of Figure 3.3 we report the convergence history for several different pairs of choices for the two spaces $\mathbb{V}_m$ and $\mathbb{W}_m$. Specifically, we experiment with extended/augmented (as before), as well as extended/standard and standard/augmented spaces. The combination of the two richer spaces provides the most effective approach. We also observe that although the use of the extended Krylov subspace for $\mathbf{A}$ seems to be very crucial to speed up convergence, the projection process becomes definitely competitive only when using the enriched (augmented) strategy also on $\mathbf{B}^T$.

Example 3.4. Here A_2 is the finite difference discretization of the operator $L(u) = (e^{-4xy}u_x)_x + (e^{4xy}u_y)_y$, $(x, y) \in (0, 1)^2$. The (scaled) symmetric matrix

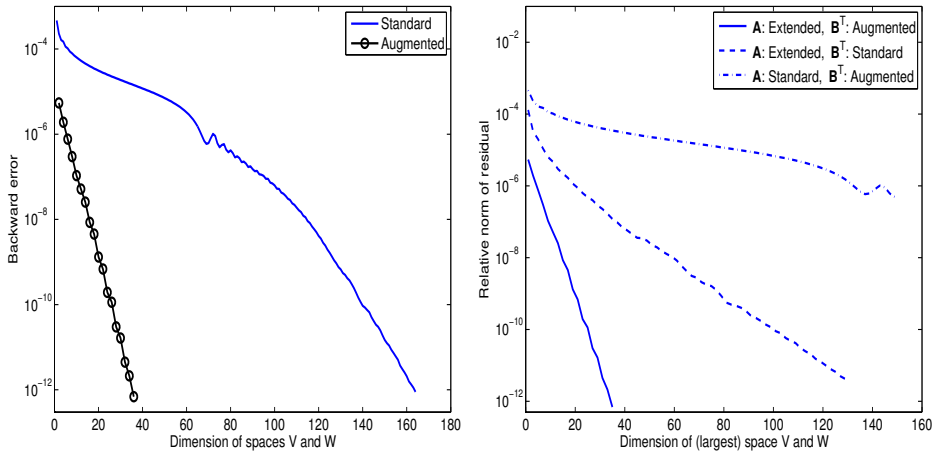


Figure 3.3: Example 3.3. Convergence history of various pairs of standard and augmented Krylov subspace solvers on the FLOW problem from the Oberwolfach benchmark collection.

A_2 of size $n_2 = 6400$ has eigenvalues in $[-1.7061 \cdot 10^2, -4.7543 \cdot 10^{-3}]$. We considered C with $m = 10$ rows and B with $p = 5$ columns, corresponding to the first m and p columns of the identity matrix, respectively. The matrix A_1 is the finite difference discretization of the negative Laplace operator in $(0, 1)^2$, with eigenvalues[1] in $[1.9779 \cdot 10^1, 5.1100 \cdot 10^4]$; here we used $n_1 = n_2 - m = 6390$, and the matrix was obtained by using 90 and 71 interior nodes in the two directions, respectively. The performance of the methods with the use of standard and augmented spaces is reported in Figure 3.4. For the left plot an a-priori computed rank-one matrix $\widehat{Y}_2 = \widehat{Y}_{2,1}\widehat{Y}_{2,2}^T$ with $\widehat{Y}_{2,1}, \widehat{Y}_{2,2}$ of all ones was used, whereas in the right plot a rank-p matrix with random entries for $\widehat{Y}_{2,1}, \widehat{Y}_{2,2}$ was employed. Defining the relative rank of a matrix X as the number of singular values greater than 10^{-12} times the largest singular value, we notice that the augmented method delivered an approximate solution matrix $\widetilde{X}_k$ with relative rank 6 and 22, respectively, for the two choices of $\widehat{Y}_2$. Therefore, the rank-one choice should be preferred, at least in terms of memory requirements for the approximation space (cf. Figure 3.4) as well as for the factors of the

[1]Only the 10 smallest in magnitude eigenvalues of $-A_1$ interlace those of A_2, with no apparent degradation in convergence.

solution matrix.

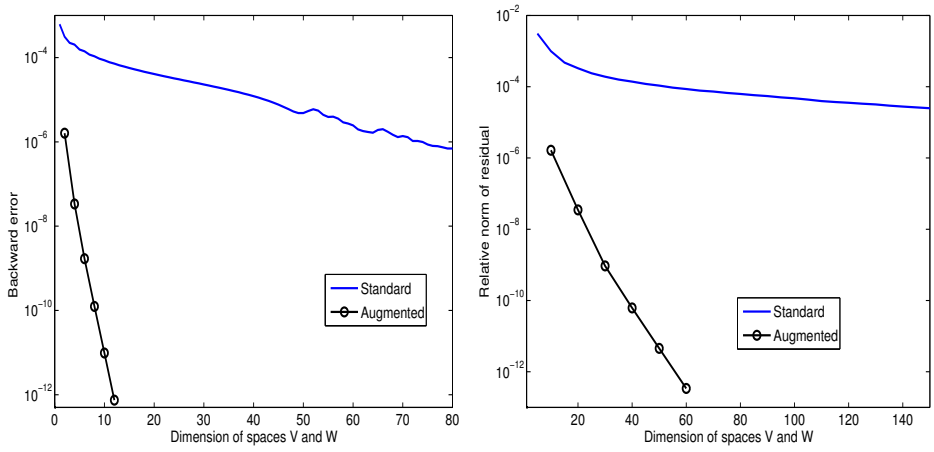


Figure 3.4: Example 3.4. Convergence history of standard and augmented Krylov subspace solvers for different ranks of $\widehat{Y}_2$. Left: rank-one $\widehat{Y}_2$. Right: rank-p $\widehat{Y}_2 = \widehat{Y}_{2,1}\widehat{Y}_{2,2}^T$ (random entries in $\widehat{Y}_{2,1}$, $\widehat{Y}_{2,2}$).

3.5 Chapter summary

We have devised a new formulation of a constrained Sylvester matrix equation, which allows one to solve large-scale problems by means of advanced Galerkin projection methods. To be able to efficiently solve the resulting Sylvester equation, we have introduced a new augmented space, which overcomes the singularity of the coefficient matrix $\mathbf{B}$. Though the implementation is completely analogous to that of more familiar extended Krylov subspaces, the use of such an approximation to an extended Krylov subspace as a means of enriching the standard space appears to be new. From a theoretical point of view, known results for rational Krylov subspace methods on the Sylvester equation can be applied to the "exact" augmented space $\mathbb{EK}_m(\mathbf{B}^T + \sigma I, \mathbf{F})$; see [6]. The space we actually adopt, $\widehat{\mathbb{EK}}_m^{\sigma}$, may somewhat differ from the exact case, depending on the properties of the matrix of inputs B.

Nonetheless, our promising numerical experiments seem to reinforce the intuition that the space generated with $\widehat{\mathbf{B}}_\sigma$ appropriately captures spectral infor-

mation that is otherwise missed in the standard Krylov subspace $\mathbb{K}_m\left(\mathbf{B}^T, \mathbf{F}\right)$ at an early stage. Finally, we expect that our approach may be of value also when attacking the non-homogeneous form of the equation (3.1)–(3.2), which is currently an open problem in the large-scale case.

CHAPTER 4

GENERALIZED LYAPUNOV EQUATIONS

In this chapter we present a numerical technique for solving generalized Lyapunov equations. We first state some known facts about using the extended Krylov subspace method for standard Lyapunov equations. We then survey some existing theory for generalized Lyapunov equations and comment on how this can be used to devise solution methods for the large-scale case. Our approach can be described as an inner-outer scheme where inner solves are Lyapunov solves with the extended Krylov subspace method performed in an inexact manner. We briefly describe some current problems in the literature and existing approaches to solve them, and compare CPU and memory costs with our proposed method. It can be appreciated that the proposed method is competitive with the current state-of-the-art.

4.1 Extended Krylov subspace methods for Lyapunov equations

We briefly comment on how the extended Krylov subspace method can be used to solve large-scale Lyapunov equations

$$AX + XA^T + BB^T = 0 \tag{4.1}$$

for large, sparse $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times r}$ with $r \ll n$. The method described in Section 2.3 extends naturally; in fact, historically, extended Krylov subspaces were used for Lyapunov equations before they were used for Sylvester equations. We essentially summarize the technique described in [68]. Building a basis of the space (2.18) as described in Section 2.3 now entails only building $\mathbb{EK}_m(A, B)$. Since transposing (4.1) yields

$$AX^T + X^TA^T + BB^T = 0$$

it follows that X^T solves the same Lyapunov equation. If this solution is unique we must have $X = X^T$, that is, X must be symmetric. This is also seen by examining the analytic solution (assuming A is stable)

$$X = \int_0^\infty e^{At}BB^Te^{A^Tt}dt \tag{4.2}$$

and in fact from this we may infer that $v^TXv \geq 0$ for all $v \in \mathbb{R}^n$, i.e., the solution to (4.1) is positive semi-definite.

A desirable property of the extended Krylov subspace method is that approximate solutions are also both symmetric and positive semi-definite. To demonstrate that this is indeed the case, note that an element X_m of the space

$$\mathbb{S}_m = \mathbb{S}\left(\mathbb{EK}_m(A, B), \mathbb{EK}_m(A, B)\right)$$

is of the form $X_m = \mathbf{V}_m Y_m \mathbf{V}_m^T$. The Rayleigh quotient matrices $\mathbf{T}_m = \mathbf{V}_m^T A \mathbf{V}_m$ inherit stability from A, since for any orthogonal matrix $\mathbf{V}$ one has

$$W(\mathbf{V}^TA\mathbf{V}) \subseteq W(A),$$

so that solution of the projected problem is given by

$$Y_m = \int_0^\infty e^{\mathbf{T}_m t}(\mathbf{V}_m^T B)(\mathbf{V}_m^T B)^T e^{\mathbf{T}_m^T t} dt$$

which is also symmetric and positive semi-definite for the same reasons as before. Thus X_m is clearly symmetric, and if one factors $Y_m = U_m U_m^T$, it follows that $X_m = Z_m Z_m^T$ where $Z_m = \mathbf{V}_m U_m$, so that X_m is also positive semi-definite. Conceptually, we think of the extended Krylov subspace method for Lyapunov equations as a way to efficiently solve for this single low-rank factor Z_m.

For positive semi-definite matrices we slightly abuse notation and write the truncation operator in terms of low-rank factors, i.e., writing $\tilde{Z} = \mathcal{T}(Z)$ means that

$$\left\| ZZ^T - \tilde{Z}\tilde{Z}^T \right\|_F \leq \tau \left\| ZZ^T \right\|_F$$

for some prescribed truncation tolerance τ, where the truncation is performed as described in Section 2.4. We note that this truncation can be performed at negligible cost for an approximate solution delivered by the extended Krylov subspace method. Since it is necessarily of the form $X_m = \mathbf{V}_m Y_m \mathbf{V}_m^T$ with $\mathbf{V}_m$ having orthonormal columns, the QR decomposition of the outer matrix can be skipped. In order to truncate, one needs to compute an eigenvalue decomposition of the small symmetric positive definite matrix Y_m, which is technically a singular value decomposition. This can be truncated as previously described.

4.2 Background on generalized Lyapunov equations

A *generalized Lyapunov equation*[1] is a linear matrix equation of the form

$$AX + XA^T + NXN^T + BB^T = 0 \tag{4.3}$$

[1]We note the name *generalized* Lyapunov equation often refers to an equation of the form $AX + XA^T + \sum_{j=1}^{\ell} N_j X N_j^T + BB^T = 0$ and that what we write is a special case of this.

for sparse $A, N \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times r}$ for the unknown $X \in \mathbb{R}^{n\times n}$. Again it is assumed that $r \ll n$. The matrix N may often be of low-rank and/or satisfy the property $\|N\|_F \ll \|A\|_F$. We write $\mathcal{A}(X) = AX + XA^T + NXN^T$ and $\mathcal{B} = -BB^T$, so that (4.3) can be written as

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

Our approach to solve (4.3) is inspired by Damm [16], where it is suggested that one use the splitting

$$\mathcal{A} = \mathcal{M} - \mathcal{N} \text{ with } \mathcal{M}(X) = AX + XA^T \text{ and } \mathcal{N}(X) = -NXN^T$$

and the associated classical iteration

$$\mathcal{M}(X_{k+1}) = \mathcal{N}(X_k) + \mathcal{B}. \tag{4.4}$$

It is well known that this iteration converges provided $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$; for general references on classical iterations and splittings we refer to [11, 80]. Damm then recalls an equivalence between positive semi-definiteness of the solution matrix X and convergence of the classical iteration (4.4), where the method of proof essentially follows from results in [65]. We remark that a positive semi-definite solution may often be expected as a result of interpreting X as a controllability or observability Gramian of some associated bilinear dynamical system. Based on this result, the above classical iteration and existing smale-scale Lyapunov solvers provide a viable numerical method in the small-scale case.

In [7], Benner and Breiten consider the large-scale, low-rank right-hand side case. They provide a theory for singular value decay in analogy to the standard Lyapunov equation. A well-known solver for standard Lyapunov equations is the low-rank alternating directions implicit (henceforth ADI) method, described in [43]. Benner and Breiten propose a modification of ADI that is suitable for the generalized Lyapunov equation, called Bilinear ADI, and use this as a stand-alone solver as well as a preconditioner for the low-rank Krylov subspace methods described in Section 2.4.

We shall propose a return to the method based on classical iterations of the form (4.4) and low-rank solvers for Lyapunov equations. Specifically, we use the classical iteration (4.4) and the fact that the extended Krylov subspace method returns a low-rank factor Z_k of the solution to reduce (4.3) to solving a sequence of equations of the form

$$AX_{k+1} + X_{k+1}A^T + [NZ_k, B] \begin{bmatrix} Z_k^T N^T \\ B^T \end{bmatrix} = 0, \tag{4.5}$$

that is, a sequence of large-scale Lyapunov equations with low-rank right-hand sides. Our perspective is essentially that the extended Krylov subspace method is such a powerful solver for large-scale Lyapunov equations such as those in (4.5) that a low-rank version of the classical iteration (4.4) can be a competitive modern solver for large-scale generalized Lyapunov equations.

We shall provide some theory for performing the required inner solves in an inexact manner. Moreover, given a splitting we shall propose a method for monitoring the norm of the residual associated with this method. What we present is especially beneficial in the low-rank case. We shall also describe a simple but advantageous perspective on using extended Krylov subspace methods to solve Lyapunov equations with a right-hand side that has a modest rank. With all of these ingredients, we shall provide numerical evidence that the proposed approach is competitive with respect to CPU time, storage requirements, and subroutine calls such as linear system solves with coefficient matrix A.

4.3 Inexact stationary iterations

4.3.1 Decreasing tolerances

We now provide some theory for performing an inexact version of the classical iteration (4.4) in such a manner that the outer iteration still converges. Namely, we consider non-stationary iterative methods that are of the form

$$Mx_{k+1} = Nx_k + b + e_{k+1}. \tag{4.6}$$

While the bounds we provide may be difficult to directly apply in certain settings, they provide theoretical evidence for several phenomena that were observed in our numerical experiments. One such phenomena was the use of decreasing inner tolerances while maintaing convergence of the outer iteration. A theoretical result aids understanding, since it is not clear a priori whether decreasing or increasing tolerances should be used. For instance, it is well known that for inexact Newton sequences the necessary Jacobian solves can initially be performed to a loose tolerance that is tightened as the iteration converges [18]. The case is exactly reversed in the case of Krylov subspace methods, where the accuracy of the matrix-vector product used to build a Krylov subspace must initially be very tight, and can eventually be relaxed as the iteration proceeds [71].

The following theorem motivates choosing a sequence of inner tolerances that is proportional to the relative residual while maintaining convergence of the iteration. It requires a slightly stronger condition for convergence of the exact stationary iteration, namely, that $\|M^{-1}N\| < 1$ in some computable norm.

Theorem 4.1. *Let $A = M - N$ with M nonsingular such that $\|M^{-1}N\| < 1$. Let $x_\star$ be the solution of $Ax = b$, and consider a nonstationary iteration of the form* (4.6)*, where we impose*

$$\|e_{k+1}\| \leq \tau \|r_k\| \tag{4.7}$$

for some positive real number τ. Then iteration (4.6) *converges linearly to the solution $x_\star$ with convergence factor no greater than $\gamma = \|M^{-1}N\| + \tau\|M^{-1}\|\|A\|$.*

Proof. Since $Mx_\star = Nx_\star + b$, we first write $x_{k+1} - x_\star = M^{-1}N(x_k - x_\star) +$

$M^{-1}e_{k+1}$. Taking norms yields

$$\begin{aligned}\|x_{k+1} - x_\star\| &\leq \|M^{-1}N\| \cdot \|x_k - x_\star\| + \|M^{-1}\| \cdot \|e_{k+1}\| \\ &\leq \|M^{-1}N\| \cdot \|x_k - x_\star\| + \tau\|M^{-1}\| \cdot \|r_k\| \\ &= \|M^{-1}N\| \cdot \|x_k - x_\star\| + \tau\|M^{-1}\| \cdot \|A(x_\star - x_k)\| \\ &\leq \left(\|M^{-1}N\| + \tau\|M^{-1}\| \cdot \|A\|\right) \|x_k - x_\star\|\end{aligned}$$

and the theorem follows. □

This theorem provides theoretical justification for another modification that proved to be beneficial, namely, the use of inexact right-hand sides. Solving (4.5) via the extended Krylov subspace method requires building $\mathbb{EK}_m(A, [NZ_k, B])$. In our experience, the matrix $[NZ_k, B]$ has too many columns to make this procedure computationally tractable. We therefore instead propose to compute $B_k = \mathcal{T}([NZ_k, B])$ for the same set of decreasing tolerances and instead solve

$$AX_{k+1} + X_{k+1}A^T + B_kB_k^T = 0. \tag{4.8}$$

This error can also be "hidden" in the inexactness term e_{k+1}, and building $\mathbb{EK}_m(A, B_k)$ is much more appealing since B_k now has far fewer columns as a result of column compression. We note that for the problems we considered, N was of low-rank, and this seemed to assist in reducing the number of columns of B_k.

4.3.2 Residual monitoring

As noted, solves with M will often be done according to some iterative procedure as well and thus performed to some tolerance μ, resulting in an iteration of the form (4.6) for some e_{k+1} that satisfies $\|e_{k+1}\| \leq \mu$. It would then be of interest to monitor $\|r_{k+1}\|$, where $r_{k+1} = b - Ax_{k+1}$ is the residual of the original system.

Theorem 4.2. *Let $A = M - N$ and consider an inexact stationary iteration of the form (4.6) where $\|e_{k+1}\| \leq \mu$ for some tolerance μ. Then*

$$\|r_{k+1}\| \leq \mu + \|N(x_{k+1} - x_k)\|. \tag{4.9}$$

Proof. We calculate

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} \\ &= b - Mx_{k+1} + Nx_{k+1} \\ &= b - Mx_{k+1} + Nx_k - Nx_k + Nx_{k+1} \\ &= -e_{k+1} + N(x_{k+1} - x_k) \end{aligned}$$

so that the desired inequality immediately follows from the triangle inequality. □

We propose the right-hand side of the inequality in (4.9) as a practical means for monitoring the norm of the residual and enforcing stopping criteria. At first it appears to be useless, since in the context of solving a linear system $Ax = b$ it provides an estimate for the residual norm at approximately the same cost of calculating the residual norm itself (one matrix vector product, one vector addition, and one vector norm). However, it does find utility in the context of low-rank solvers for linear matrix equations. This is because there are some subtleties associated with the ranks of the "vectors" involved, which are implicitly represented in some low-rank storage format.

Suppose we have just solved (4.8) to a tolerance μ for a low-rank factor Z that has p columns and we seek to estimate the norm of the residual when plugged into equation (4.3). Then the full residual is of the form

$$\mathcal{B} - \mathcal{A}(ZZ^T) = -[Z, AZ, NZ, B] \begin{bmatrix} 0_p & I_p & 0_p & 0_{p\times r} \\ I_p & 0_p & 0_p & 0_{p\times r} \\ 0_p & 0_p & I_p & 0_{p\times r} \\ 0_{r\times p} & 0_{r\times p} & 0_{r\times p} & I_r \end{bmatrix} \begin{bmatrix} Z^T \\ Z^TA^T \\ Z^TN^T \\ B^T \end{bmatrix} \tag{4.10}$$

where the low-rank factor of this residual has $3p+r$ columns. Note we typically expect $r \ll p$. If we examine the upper bound described in (4.9), where we

now use subscripts k and $k+1$ to differentiate low-rank factors and their ranks from successive iterates, we find that

$$\mathcal{N}(X_{k+1} - X_k) = [NZ_{k+1}, NZ_k] \begin{bmatrix} I_{p_{k+1}} & 0 \\ 0 & -I_{p_k} \end{bmatrix} \begin{bmatrix} Z_{k+1}^T N^T \\ Z_k^T N^T \end{bmatrix} \tag{4.11}$$

where the low-rank factor has only $p_{k+1} + p_k$ columns.

If we assume (for simplicity and argumentation) that $p = p_k = p_{k+1}$, then additional storage drops from $3p+r$ to $2p$. We then compute the residual norm $\|R_{k+1}\|_F$ as $\|R_{k+1}\|_F = \langle R_{k+1}, R_{k+1} \rangle_F^{1/2}$ according to the low-rank Frobenius inner product described in Section 2.4. Either method incurs some additional $\mathcal{O}(n)$ computation to compute or estimate the residual norm. But if storage is an issue, it can pay to bound the residual norm as described. At first glance it may seem puzzling; the estimate in the theorem may originally be discarded as not useful. But with low-rank methods, such subtle issues are common. Typically μ (the *inner* tolerance) is smaller than our outer tolerance, so that not much is lost in using the triangle inequality. In our experiments we found the approximation to be of very good quality.

4.4 Separate right-hand sides for Lyapunov equations

Even after truncating $B_k = \mathcal{T}([NZ_k, B])$, the cost of building $\mathbb{EK}_m(A, B_k)$ could still be dominated by orthogonalizations. We therefore consider solving Lyapunov equations with thick right-hand sides[2]. When solving a linear system

$$\mathbf{Ax} = \mathbf{b}$$

it may seem strange to write $\mathbf{b} = \mathbf{b}_1 + \cdots + \mathbf{b}_r$, solve $\mathbf{x}_i = \mathbf{A}^{-1}\mathbf{b}_i$ for $i = 1, \ldots r$, and return $\mathbf{x} = \mathbf{x}_1 + \cdots + \mathbf{x}_r$ as the solution. Nevertheless, this is what we

[2]Here thick means many columns, that is, $r > 10$.

shall consider for

$$\mathbf{A} = I \otimes A + A \otimes I \text{ and } \mathbf{b} = \text{vec}(-BB^T)$$

i.e., for the Lyapunov equation (4.1) where $\text{vec}(X) = \mathbf{x}$. We shall consider the case $r > 10$, which has shown up in several application areas outside of control theory. For instance, in control theory a SISO system guarantees that $r = 1$ so that the associated Lyapunov equation can be feasibly solved. However, a Lyapunov equation may arise as a subproblem of solving another problem, as in (4.4). In this case it can often have a thick right-hand side that is out of the user's control. The cost of building a block space may become infeasible.

As a possible way around this, write $B = [b_1, \ldots, b_r]$ and note that the interpretation of matrix-matrix products as a sum of outer products [26] yields

$$BB^T = b_1 b_1^T + \cdots + b_r b_r^T.$$

Since $\mathcal{M}$ is linear, so is $\mathcal{M}^{-1}$, thus

$$X = \mathcal{M}^{-1}(-b_1 b_1^T) + \cdots + \mathcal{M}^{-1}(-b_r b_r^T),$$

that is, we are in the above scenario with $\mathbf{b}_i = -b_i \otimes b_i$ for $i = 1, \ldots, r$. Individual solves $\mathbf{A}^{-1}\mathbf{b}_i$ correspond to rank-one Lyapunov equations

$$AX_i + X_i A^T + b_i b_i^T = 0$$

where $\text{vec}(X_i) = \mathbf{x}_i$. Noting that

$$\mathbb{EK}_m(A, b_i) \subseteq \mathbb{EK}_m(A, B) \text{ for all } i = 1, \ldots, r,$$

one should ask what is gained by building $\mathbb{EK}_m(A, B)$ versus $\mathbb{EK}_m(A, b_i)$ for all i, since the latter requires a lot less orthogonalization. Such issues have already been known to arise in standard block Krylov subspace methods [31].

However, a subtle issue with rank arises that can further justify the above approach. Upper bounds for the singular values of the solution X in the symmetric case are given in [52]. Let $\lambda_i(X)$ denoted the ith eigenvalue of

X non-increasingly ordered (so that these are the singular values since X is symmetric positive semi-definite). One can then show that the following upper bound holds

$$\frac{\lambda_{rk+1}(X)}{\lambda_1(X)} \leq \left(\prod_{j=0}^{k-1} \frac{\kappa^{(2j+1)/(2k)} - 1}{\kappa^{(2j+1)/(2k)} + 1} \right)^2,$$

where $\kappa = \kappa_2(A)$ is the 2-norm condition number of A. The main point in bringing up this bound is that the right-hand side of this inequality is independent of the rank r of problem data B. Hence the decay appears to scale with the rank r, and for higher r the above bound permits room for slower decay. This is reflected in Figure 2.2.

This is also reflected in other methods of proof for the existence of low-rank approximations. Specifically, in [27] an analytic solution based on a contour integral representation is approximated by chopping the contour into several pieces. On each piece, the integrand is expanded as a sum of separable functions, and the resulting integrals are then approximated by quadrature. Ultimately, this procedure results in a sum of a low-rank terms. At no point is the rank of the right-hand used to influence the procedure, and any bounds obtained on the quality of the separable approximations or quadrature method are independent of this rank. Yet the rank of the approximate solution scales exactly with the rank of the right-hand side. A simple way to see the rank of an approximate solution scaling with the number of columns of B is by using some quadrature rule on (4.2). Given quadrature nodes t_i and weights w_i for $i = 1, \ldots, k$, one has

$$\begin{aligned} X &= \int_0^\infty e^{At} B B^T e^{A^T t} dt \\ &\approx \sum_{i=1}^{k} w_i e^{At_i} B B^T e^{A^T t_i} \end{aligned}$$

whose rank is rk.

Taking these observations into account, a loose heuristic argument as to why one would build the individual spaces instead of the full block space goes as follows. Building $\mathbb{EK}_m(A, B)$ to solve (4.1) results in a basis that is "r

times as rich." However, when the problem is split in the proposed manner, individual problems become "r times easier" to solve since one expects more singular value decay from solutions with single column right-hand sides. If we make the simple but dubious assumption that these effects cancel one another out, there is a net gain in the new approach since the basis of the block space does not need to be fully orthogonalized nor fully stored.

If we denote X_i as the solution corresponding to right-hand side $b_i b_i^T$, we can guarantee that the approximate solution $X = X_1 + \ldots X_r$ satisfies a given relative residual tolerance

$$\frac{\left\|AX + XA^T + BB^T\right\|_F}{\left\|BB^T\right\|_F} \leq \tau$$

by solving each individual equation to tolerance τ/r

$$\frac{\left\|AX_i + X_iA^T + b_ib_i^T\right\|_F}{\left\|BB^T\right\|_F} \leq \frac{\tau}{r},$$

so that one must solve to a higher tolerance and thus incurs some additional computational overhead. We found it was well worth it to incur this cost as a tradeoff for not having to perform an inordinate amount of orthogonalization.

In our experiments we found truncating the low-rank factor $Z = [Z_1, \ldots, Z_r]$ as it is being built to be beneficial, where Z_i denotes the low-rank factor associated with the column vector b_i. Specifically, one first solves for Z_1 and sets the current approximation to the low-rank factor for Z as $Z^{(1)} = Z_1$. Then, since addition corresponds to concatenation of low-rank factors, one solves for Z_i and updates $Z^{(i)} = \mathcal{T}([Z^{(i-1)}, Z_i])$ for $i = 2, \ldots, r$ and sets the final low-rank factor of the approximate solution $Z = Z^{(r)}$. We shall refer to this procedure as *progressive truncation*. While initially introducing a minimal overhead in terms of CPU, this is eventually compensated by not having to perform a very thick truncation of a low-rank factor $[Z_1, \ldots, Z_r]$ that has not been progressively truncated. It also assisted with memory needs, in that storing $Z^{(r-1)}$ and a final basis $\mathbb{EK}_m(A, b_r)$ resulted in lower memory requirements when progressive truncation was implemented.

4.5 Storing the initial approximation

Note that with an iteration of the form

$$x_{k+1} = M^{-1}(Nx_k + b) \tag{4.12}$$

$$= M^{-1}Nx_k + M^{-1}b \tag{4.13}$$

a quick analysis shows that the costs of (4.12) and of the equivalent iteration (4.13) are about the same. Both entail one vector addition, one matrix vector product with N, and one linear solve with M per iterate. However, (4.13) requires storing the extra vector $M^{-1}b$ in order to compute the required update without recomputing this quantity at every iterate. If this additional vector can be stored (which is typically the case), one might expect the performance of either approach to be comparable. Nevertheless, for our purposes it was worthwhile to analyze the corresponding iterations at the matrix level

$$X_{k+1} = \mathcal{M}^{-1}\big(\mathcal{N}(X_k) + \mathcal{B}\big) \tag{4.14}$$

$$= \mathcal{M}^{-1}\big(\mathcal{N}(X_k)\big) + \mathcal{M}^{-1}(\mathcal{B}) \tag{4.15}$$

and consider the effect of all computations being performed implicitly on low-rank factors. In this context, applying $\mathcal{M}^{-1}$ entails performing a Lyapunov solve with the extended Krylov subspace method, and different right-hand sides entail different spaces being built. For instance, (4.14) requires building $\mathbb{EK}_m(A, [NZ_k, B])$, while (4.15) requires building $\mathbb{EK}_m(A, NZ_k)$. The initial block of the latter space has fewer columns and is therefore less expensive to build.

This observation prompted experimentation with both approaches. In order to implement (4.15), we store the initial low-rank factor $Z_1 = \mathcal{M}^{-1}(\mathcal{B})$. At each iterate, we then compute the intermediate low-rank factor

$$Z_{k+1/2} = \mathcal{M}^{-1}\big(\mathcal{N}(X_k)\big).$$

The two are then added to obtain the low-rank factor $Z_{k+1} = [Z_1, Z_{k+1/2}]$ of X_{k+1}. In both cases, the right-hand sides are truncated before building the

extended Krylov subspace, and we truncate to the same absolute tolerance in order to make a fair comparison.

4.6 Low-rank classical methods for generalized Lyapunov equations

We tie results from the previous sections together to describe our approach for solving generalized Lyapunov equations with extended Krylov subspaces. All truncation and inner tolerances will be some multiple of our current residual estimate as described in (4.7); this multiple shall be denoted τ^{inexact}. The quantity used to monitor the relative residual at the k^{th} outer iterate, based on (4.9), shall be denoted by τ_k^{bound}. All Lyapunov solves are done with the extended Krylov subspace method until the relative residual falls below a prescribed inner tolerance that is proportional to the accuracy of the outer iterate. Namely, at the k^{th} outer iterate we set our inner tolerance $\tau_k^{\text{inner}} = \tau^{\text{inexact}}\tau_{k-1}^{\text{bound}}$. These Lyapunov solves are done by separating the right-hand sides as described in Section 4.4. All truncations are performed to some specified truncation tolerance τ_k^{trunc}, which we take equal to the inner tolerance. The method terminates when the upper bound on the relative residual norm τ_k^{bound} falls below τ^{outer}, so that a relative tolerance τ^{outer} is guaranteed to have been achieved. The full method is described in Algorithm 4.1. We note that while we used the extended Krylov subspace method for Lyapunov equations described in Section 2.3, in theory any Lyapunov solver could be used. We chose EKSM because it is a strong contender for the state-of-the-art, especially for rank-one right-hand sides.

We perform a careful analysis of the storage demands that Algorithm 4.1 requires. There are two considerations that should be made. The first occurs when solving the sequence of rank-one Lyapunov equations. For a given outer iterate, we require the low-rank factor at the previous iterate in order to compute (4.11), whose norm will provide an upper bound for the residual norm

Algorithm 4.1 Low-rank classical iterations

Input: Problem data A, N and B, tolerances $\tau^{\text{inexact}}, \tau^{\text{outer}}$.

Set $\beta = \left\|BB^T\right\|_F$.
Solve $AX + XA^T + BB^T = 0$ to tolerance τ^{inexact} for Z_1.
Calculate $\tau_1^{\text{bound}} = \tau^{\text{inexact}} + \|\mathcal{N}(X_1)\|_F / \beta$
for $k = 2, 3, \ldots$ **do**
    Set tolerances $\tau_k^{\text{inner}} = \tau_k^{\text{trunc}} = \tau^{\text{inexact}} \tau_{k-1}^{\text{bound}}$
    Calculate the low-rank factor $B_k = \mathcal{T}([NZ_{k-1}, B])$
    Write $B_k = [b_1^{(k)}, \ldots, b_{\nu_k}^{(k)}]$
    for $i = 1, \ldots, \nu_k$ **do**
        Solve $AX + XA^T + b_i^{(k)}(b_i^{(k)})^T = 0$ to tolerance $\tau_k^{\text{inner}}/\nu_k$ for $\tilde{Z}_i$
        if $i = 1$ **then**
            $Z^{(1)} = \mathcal{T}(\tilde{Z}_1)$
        else
            $Z^{(i)} = \mathcal{T}([Z^{(i-1)}, \tilde{Z}_i])$
        end if
    end for
    Set $Z_k = Z^{(\nu_k)}$, calculate $\beta_k = \|B_k B_k^T\|_F$ and $\tilde{\beta}_k = \|\tilde{B}_k \tilde{B}_k^T\|_F$
    Set $\tau_k^{\text{bound}} = \left(\tilde{\beta}_k \tau_k^{\text{inner}} + \beta_k \tau_k^{\text{trunc}} + \|\mathcal{N}(X_k - X_{k-1})\|_F\right) / \beta$
    Stop if $\tau_k^{\text{bound}} \leq \tau^{\text{outer}}$
end for

at the current iterate. We are also progressively building a low-rank factor for the approximate solution that is being truncated at every step of the i loop. Finally, we shall also have a basis for the extended Krylov subspace associated with the next column vector $b_i^{(k)}$ that is used to continue assembling the low-rank factor. Thus, at this stage of a given outer iterate, our memory demands consist of vectors from

- the columns of the previous iterate Z_{k-1}
- the columns of the partially assembled Z_k

- the columns of an orthonormal basis for $\mathbb{EK}_\ell(A, b_i^{(k)})$

which counts all stored vectors that have length n. Therefore, our first consideration regarding memory is to record how many of the above vectors are stored at every outer iterate, with the maximum typically occurring at the last iterate.

At the next stage of an outer iteration, we store the fully truncated Z_k but discard the basis of the last extended space which is no longer needed. Computing $\|\mathcal{N}(X_k - X_{k-1})\|_F$ consists of computing of NZ_k and NZ_{k-1}, where the latter computation may be performed by applying N to each column of Z_{k-1} and overwriting them one column at a time since Z_{k-1} is no longer needed after this operation is performed. Hence at this stage our memory demands are

- the columns of the current iterate Z_k
- the columns of $[NZ_k, -NZ_{k-1}]$ used for estimating the residual norm

which again counts all vectors of length n that need to be stored. Therefore, for the memory demands of our proposed method, the number of vectors stored at both stages of each outer iterate are recorded, and we report the largest recorded number as the memory demands for the proposed method. We note that there was some competition between the two stages; at any given outer iterate one could be larger than the other.

4.7 Numerical experiments

We consider a symmetric problem arising in boundary control of the heat equation, described in [16, 64]. We also consider a non-symmetric problem which comes from the bilinearization of a nonlinear circuit model, described in [3]. First, we discuss the relative merits of different elements of our approach. Table 4.1 compares all possible combinations of four of our proposed features

Inexact solves	Inexact r.h.s.	Separate r.h.s.	Residual monitor	CPU	Memory	Solves	Rank
				66.56	405	1702	101
			✓	61.80	427	1702	101
		✓		39.72	421	1438	105
		✓	✓	35.79	225	1438	105
	✓			34.24	405	1077	101
	✓		✓	31.11	427	1077	101
	✓	✓		25.50	421	1020	105
	✓	✓	✓	22.22	225	1020	105
✓				41.81	405	1230	101
✓			✓	37.59	398	1230	101
✓		✓		24.82	421	722	105
✓		✓	✓	21.38	225	722	105
✓	✓			25.78	405	818	101
✓	✓		✓	22.61	398	818	101
✓	✓	✓		18.69	421	631	105
✓	✓	✓	✓	16.13	225	631	105

Table 4.1: Comparing various enhancements to overall performance of the method (symmetric problem, size $n = 10000$).

for the symmetric problem. A similar table is given for the non-symmetric problem in Table 4.2.

A checkmark in the column marked "Inexact solves" means that a decreasing sequence of inner tolerances was used; if empty, an inner tolerance of 10^{-11} was used. By "Inexact r.h.s." we mean that a decreasing sequence of truncation tolerances was used; otherwise, a truncation tolerance of 10^{-10} was used. The column entitled "Separate r.h.s." is used to denote that only Lyapunov equations with rank-one right-hand sides were solved, as described in Section 4.4. Finally, "Residual monitor" denotes whether the full residual (4.10) was formed, or the cheaper bound for the residual (4.11) was used. Where applicable, the quantity τ_{inexact} of Algorithm 4.1 was taken to be 10^{-3}.

We now analyze the effect of each feature. Decreasing inner and truncation tolerances both have the effect of savings in CPU time and number of linear solves performed, which is to be expected due to smaller spaces being built.

Inexact solves	Inexact r.h.s.	Separate r.h.s.	Residual monitor	CPU	Memory	Solves	Rank
				701.64	677	3692	169
			✓	690.60	603	3692	169
		✓		222.45	701	4690	175
		✓	✓	203.21	375	4690	175
	✓			616.84	697	3458	174
	✓		✓	541.97	671	3135	174
	✓	✓		193.23	721	4130	180
	✓	✓	✓	182.49	385	4130	180
✓				450.38	677	2934	169
✓			✓	447.87	592	2934	169
✓		✓		144.15	705	3024	176
✓		✓	✓	127.26	377	3019	176
✓	✓			505.74	697	3203	174
✓	✓		✓	353.05	622	2557	174
✓	✓	✓		97.79	709	2342	177
✓	✓	✓	✓	85.94	381	2342	177

Table 4.2: Comparing various enhancements to overall performance of the method (non-symmetric problem, size $n = 10100$).

However, neither assists in reducing memory demands, since by the final outer iterate the decreasing inner tolerances became approximately equal to the fixed inner tolerance, and performing an inner solve to this tolerance resulted in the equal storage demands. A similar phenomena occurred for truncation tolerances. Solving separate right-hand sides led to a large reduction in CPU time, but also a small reduction in memory demands, since the full block extended Krylov subspace is never formed. Residual monitoring, on the other hand, often led to a large reduction in memory demands, but also a small reduction in CPU time, since a full residual norm was more expensive to calculate. It is also worth remarking that residual monitoring was most effective when paired with separate right-hand sides, since building the full extended Krylov subspace could often be the dominating memory cost. The combination of all four proposed modifications provides the most effective approach; implementing one feature does not appear to negatively impact any other. Finally, we

observed the rank of approximate low-rank factors to be about the same with each method.

Next, we comment on the approach of storing the initial low-rank factor, described in Section 4.5. We compare the classical iterations (4.14) and (4.15) for the non-symmetric problem of size $n = 10100$. Applications of $\mathcal{M}^{-1}$ again entail Lyapunov solves with the extended Krylov subspace method. In both cases right-hand sides are truncated, so that the resulting extended Krylov subspace is feasible to build. The same absolute truncation tolerance was used, meaning that if $\mathcal{T}([NZ_k, B])$ is computed with truncation tolerance τ_1, to obtain the same absolute tolerance $\mathcal{T}(NZ_k)$ is computed with truncation tolerance

$$\tau_2 = \tau_1 \|\mathcal{N}(X_k) + \mathcal{B}\|_F / \|\mathcal{N}(X_k)\|_F.$$

No other modifications to the extended Krylov subspace method are considered. In this case, the unmodified method takes 703.8 seconds and requires storing 677 vectors. On the other hand, storing the initial low-rank factor entails building a smaller extended Krylov subspace. In terms of the number of columns of the initial block, we typically saw a reduction equal to the number of columns of B. This approach took 646 seconds and required 713 vectors, which resulted in an 9 % speedup with 5 % additional storage. This is reasonable, since each inner solve is cheaper to perform, while an additional low-rank factor needs to be stored.

Combining this feature with the others posed some difficulties. For instance, the quality of the approximate low-rank factor Z_1 affects the quality of all subsequent low-rank factors Z_k. Specifically, in order to satisfy a decreasing sequence of tolerances, the initial iterate Z_1 needed to be solved to a tolerance that was small enough to ensure that the final inner tolerance would be satisfied. Thus, additional columns of Z_1 needed to be stored, which made subsequent truncations more expensive. Overall, when combined with the other features we did not observe a speedup in our experiments and therefore decided to not pursue the issue further here. However, in our experiments B

also had only a single a column, and we believe the result could be of interest for B with several columns.

Finally, we compare our method with those proposed in [7], with special attention given to memory demands. In our experiments we found $\tau_{\text{inexact}} = 10^{-3}$ to be acceptable. All methods are stopped when the residual falls below $\tau_{\text{outer}} = 10^{-8}$. The truncation tolerance for the methods in [7] was $\tau_{\text{trunc}} = 10^{-10}$, and we adopt the same value here. For the low-rank Krylov subspace methods described in Section 2.4, we count the number of vectors of size n used to store of the low-rank matrices that implicitly represent a vector used in the Krylov subspace method after they have been truncated. For example, in assessing the memory demands of Algorithm 2.6 we count the number of vectors used to store X, R, Z, P and Q after they have been truncated, with an analogous counting scheme for low-rank BiCGStab.

The Bilinear ADI method updates a low-rank factor to an approximate solution according to

$$Z_{k+1} = [(A - \sigma_k I)^{-1}(A + \sigma_k I)Z_k, \sqrt{2\sigma_k}(A - \sigma_k I)^{-1}NZ_k, \sqrt{2\sigma_k}(A - \sigma_k I)^{-1}B]$$

for a series of shifts σ_k. This operation is done one shift at a time, and these shifts must be precomputed. The resulting low-rank factor is then truncated and the corresponding full residual (4.10) is formed. Thus, for the Bilinear ADI method as a stand-alone solver, we count memory costs as the number of vectors required to store this truncated approximate solution along with the number of columns of the associated full residual.

We present a summary of computational resources required for the different methods and problems. The abbreviation `EKSM-GenLyap`, which was described in Algorithm 4.1, consists of all of our proposed modifications in Tables 4.1 and 4.2. The abbreviations `ADI` and `BiADI` represent the alternating directions implicit method and its bilinear analogue. A prefix of `CG` or `BiCGStab` denotes a low-rank Krylov subspace method being used as an outer solver, with the following term denoting what was used as a preconditioner. For instance, `BiCGStab-BiADI` denotes that the low-rank BiCGStab method was used with

one step of the Bilinear ADI method as a preconditioner. For each problem, a bold number denotes superior performance in its respective category when compared against every other method.

Method	CPU (s)	Memory	Solves	Solution rank
EKSM-GenLyap	**16.1**	225	**631**	105
BiADI	100.0	**213**	3011	**53**
CG-BiADI	178.9	589	3934	54
CG-ADI	97.4	596	1545	54
EKSM-GenLyap	**35.7**	241	**581**	112
BiADI	224.6	**225**	3345	**56**
CG-BiADI	363.7	650	4868	57
CG-ADI	164.0	599	1557	**56**

Table 4.3: Comparison of performance of various methods for a symmetric generalized Lyapunov equation. Top half of the table is for a problem of size $n = 10000$, while the bottom half is for size $n = 22500$.

For the symmetric heat conduction problem, we consider problems of size $n = 10000$ and $n = 22500$, and in both cases $r = 1$. The coefficient matrices A and N are symmetric negative definite and symmetric negative semi-definite, respectively, so that the overall system is symmetric negative definite. A comparison of the performance of each method is given in Table 4.3. We note that the proposed method appears competitive with respect to memory demands, requiring only a few more vectors than then method with smallest storage requirements. As far as CPU timings and number of linear solves, the proposed method delivers the most competitive result, with a dramatically decreased CPU time. All elements of Algorithm 4.1 appear to pay off.

For the non-symmetric circuit problem, we consider two sizes $n = 10100$ and $n = 22650$, and in both cases $r = 1$. The coefficient matrix is non-symmetric and negative definite. A comparison of the performance of each method is given in Table 4.4. Here the proposed method gives the best performance with respect to CPU timings and memory, finishing in about a third of the time of the next closest method. The number of solves is also the best

Method	CPU (s)	Memory	Solves	Solution rank
EKSM-GenLyap	**85.9**	**381**	**2342**	177
BiADI	245.3	401	11654	100
BiCGStab-BiADI	818.2	3159	34300	**94**
BiCGStab-ADI	390.1	2848	13575	95
EKSM-GenLyap	**262.1**	**441**	**2729**	205
BiADI	743.9	445	14150	111
BiCGStab-BiADI	3195.8	3558	44048	108
BiCGStab-ADI	1413.7	3407	21702	**107**

Table 4.4: Comparison of performance of various methods for a non-symmetric generalized Lyapunov equation. Top half of the table is for a problem of size $n = 10100$, while the bottom half is for size $n = 22650$.

amongst methods tried, and is substantially less than the number of solves required by any of the low-rank Krylov or Bilinear ADI methods. While the rank of the approximate solution delivered by the proposed approach is larger than that of existing methods for both problems, we observed a favorable performance with respect to all other metrics.

CHAPTER 5

RITZ AUGMENTATION STRATEGIES

This chapter contains an alternative strategy for solving generalized Lyapunov equations that is based on Ritz augmentation methods. We draw heavily on the results of the preceding chapter; namely, that low-rank Lyapunov solvers may be used with stationary iterative methods to devise a solution method for large-scale generalized Lyapunov equations. Our method essentially hinges on the observation that for a convergent classical iteration

$$Mx_{k+1} = Nx_k + b$$

the right-hand sides $b_k := Nx_k + b$ must be converging as well. If solves with M are done according to some iterative method (again resulting in an inner-outer method) and this iterative method builds some basis (such as an extended Krylov subspace), the fact that $b_{k+1} \approx b_k$ can be exploited to reuse information from previous iterates.

The idea of reusing information is well known for standard linear systems and eigenvalue problems, and we shall see that the idea extends naturally to linear matrix equations. In fact, similar ideas had already been considered in [47]. Though we have assumed up until this point that we are in possession of optimal solvers for linear systems with coefficient matrix A, if no such

solver is available the construction of an extended Krylov subspace could be quite costly. We shall see that by using the method proposed in this chapter, performing linear solves with the matrix A can be stopped early in the outer iteration once enough information corresponding to these linear solves has been generated. In other words, the methods proposed in this chapter are designed to reduce the total number of linear solves with coefficient matrix A.

5.1 Background material and preliminaries

5.1.1 Augmented Krylov subspaces

Given a matrix $A \in \mathbb{R}^{n\times n}$, augmented Krylov subspaces are subspaces of the form

$$\mathbb{U}_m = \mathbb{W} + \mathbb{K}_m(A, b) \tag{5.1}$$

for some subspace $\mathbb{W} \subseteq \mathbb{R}^n$. They are described and analyzed in [14, 62]. For the solution of sequences of linear systems with the same coefficient matrix A, it may be possible to reuse information generated in the process of performing one linear solve to assist with the next. A simple example of this is the restarted GMRES method; at each inner iteration of this method, a basis for a Krylov subspace is built that is discarded upon the restart. It was realized that such discarded information could be reused to help aid convergence; see for instance [48], as well as [49] for an example pertaining to eigenvalue computations.

Perhaps the most straightforward method for building (5.1) is the one described in [62], where the augmented vectors are introduced at the end of the procedure after a Krylov subspace has been built. While this has the advantage of ease of implementation, it is not clear how to continue building the space if more vectors from the Krylov subspace are desired. Another possible approach consists of storing two sets of orthonormal bases; one for $\mathbb{U}_m$, and one for $\mathbb{K}_m(A, b)$. In this approach, one also builds the Krylov subspace $\mathbb{K}_m$ simultaneously while building $\mathbb{U}_m$. Every time a new Arnoldi vector of $\mathbb{K}_m$ is stored,

it is orthogonalized against the existing basis for $\mathbb{U}_m$ and then normalized to update the basis for the augmented space. While also easy to implement, this approach requires a doubling of storage and orthogonalizations, and it is not immediately clear how to obtain the Rayleigh quotient matrix required for a Galerkin projection at a negligible cost.

Finally, a method known as implicit restarting, originally described in [74], is applicable when the augmentation vectors are of a special form. In this case the starting vector used to build the next Krylov subspace can be modified in such a way that building a standard Krylov subspace with a new modified starting vector results in building the desired augmented space. This was used in [50] for solving linear systems. While elegant, the method is not trivial to implement. The method we shall propose is essentially based on a remarkable property of vectors known as Ritz vectors, which are described in the next section.

5.1.2 Ritz vectors

Ritz vectors and Ritz values are approximations to the eigenvalue problem

$$A\tilde{w} = \lambda\tilde{w}$$

from a Krylov subspace $\mathbb{K}_m(A, b)$. Such approximations can be extracted from $\mathbb{K}_m$ via Galerkin projection, and is sometimes referred to as the Rayleigh-Ritz procedure; details can be found in, e.g., [60]. We call an element $w \in \mathbb{K}_m$ a Ritz vector with corresponding Ritz value θ if

$$Aw - \theta w \perp v \text{ for all } v \in \mathbb{K}_m. \tag{5.2}$$

Given a Ritz vector w, the corresponding Ritz residual is $r = Aw - \theta w$. Letting the columns of V_m denote a basis for $\mathbb{K}_m$, we write $w = V_m g$ and enforce the Galerkin condition by writing $V_m^T r = 0$. In this manner, one can show that (5.2) is equivalent to

$$H_m g = \theta g,$$

that is, g is an eigenvalue of the Rayleigh quotient matrix $H_m = V_m^T A V_m$ with corresponding eigenvalue θ. From the nested properties (2.2) one can show that the Ritz residual $r \in \mathbb{K}_{m+1}$. Since r is in $\mathbb{K}_{m+1}$ but orthogonal to $\mathbb{K}_m$, the only direction in which it may have a nonzero component is v_{m+1}. This can be made explicit using the Arnoldi relation (2.4); specifically, it can be shown that

$$Aw - \theta w = f v_{m+1} \text{ for the scalar } f = h_{m+1,m} e_m^T g. \tag{5.3}$$

For large enough m, we expect the Ritz values that are largest in magnitude to produce good approximations to the eigenvalues of A that are of largest magnitude. Similarly, we expect the Ritz vectors in $\mathbb{K}_m$ corresponding to these Ritz values to approximate the corresponding eigenvectors.

If one possesses several, say q, Ritz vectors $w_1, \ldots, w_q$, these can be gathered as the columns of the matrix $W = [w_1, \ldots, w_q]$. The corresponding Ritz values $\theta_1, \ldots, \theta_q$ can be put in the diagonal matrix $\Theta = \text{diag}(\theta_1, \ldots, \theta_q)$ and the scalars $f_1, \ldots, f_q$ can be gathered in the column vector $\vec{f} = [f_1, \ldots, f_q]^T$. The relation (5.3) for each $i = 1, \ldots, q$ can then be succinctly summarized in the matrix equation

$$AW = W\Theta + v_{m+1}\vec{f}^T. \tag{5.4}$$

We remark that (5.4) expresses the fact that *all* eigenresiduals point in the same direction.

For our subsequent purposes, the relation (5.4) will be very useful. We seek an analogous structure for extended Krylov subspaces. Note that the nested properties are what guaranteed that the Ritz residual points in the direction of the last Arnoldi vector. The extended Krylov subspaces also possess an analogue of these properties; recall (2.25). We define a Ritz vector w from an extended Krylov subspace $\mathbb{EK}_m = \mathbb{EK}_m(A, B)$ in the same manner. Specifically, w is called a Ritz vector of $\mathbb{EK}_m$ with corresponding Ritz value θ if $Aw - \theta w \perp v$ for all $v \in \mathbb{EK}_m$. In exactly same manner, one can show that such Ritz values are eigenvalues of the Rayleigh quotient matrix $\mathbf{T}_m$ and that

the corresponding Ritz vector w satisfies

$$Aw = \theta w + \mathbf{V}_{(m+1)}\mathbf{f} \text{ for the column vector } \mathbf{f} := \mathbf{T}_{(m+1,m)}\mathbf{E}_m^T g,$$

where $\mathbf{V}_{(m+1)}$ denotes the last block of basis vectors generated by the extended Arnoldi algorithm and $\mathbf{T}_{(m+1,m)}$ denotes $(m+1 \times m)$ block of the block upper Hessenberg matrix $\underline{\mathbf{T}}_m$.

Assuming once again that we are in possession of q of these quantities that are gathered in W, Θ, and the matrix $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_q]$, it follows that

$$AW = W\Theta + \mathbf{V}_{(m+1)}\mathbf{F}. \tag{5.5}$$

However, note that a Ritz value, being an eigenvalue of $\mathbf{T}_m$, may be complex if A is non-symmetric. For subsequent purposes, we would like to avoid using complex arithmetic while maintaining a structure resembling (5.5) with quantities that are all purely real. It will also be desirable for W to have orthonormal columns.

To accomplish this task, given some Ritz vector w that we would like in $\mathbb{W}$, we first write the following quantities in terms of their real and imaginary parts

$$w = x + iy, \; g = s + it, \text{ and } \theta = \alpha + i\beta, \tag{5.6}$$

so that (5.3) is equivalent to

$$A[x, y] = [x, y] \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} + \mathbf{V}_{(m+1)}[\mathbf{f}_\alpha, \mathbf{f}_\beta], \tag{5.7}$$

where $\mathbf{f}_\alpha = \mathbf{T}_{(m+1,m)}\mathbf{E}_m^T s$ and $\mathbf{f}_\beta = \mathbf{T}_{(m+1,m)}\mathbf{E}_m^T t$. One can now construct real $\tilde{W}$, $\tilde{\Theta}$ and $\tilde{\mathbf{F}}$ satisfying (5.5). We first initialize these three matrices to have no columns. If we seek to augment with a real Ritz vector, we append this as the last column of $\tilde{W}$. We then place the corresponding Ritz value θ as the next diagonal entry of $\tilde{\Theta}$ and append $\mathbf{f}$ to the end of $\tilde{\mathbf{F}}$. If we instead want to augment with a complex Ritz vector, we split it into real and imaginary parts as in (5.6) and append these as two columns to $\tilde{W}$. We then place the real and

imaginary parts of the associated Ritz value as a 2×2 block on the diagonal of $\tilde{\Theta}$ resembling the 2×2 matrix in (5.7), and append the two column vectors $\mathbf{f}_\alpha$ and $\mathbf{f}_\beta$ as defined above to the end of $\tilde{\mathbf{F}}$. Doing this one Ritz vector at a time for every Ritz vector that we seek to put into $\mathbb{W}$ results in a relation of the form (5.5).

While this may suffice for some purposes, the columns of $\tilde{W}$ are not necessarily orthonormal. However, this can be obtained while preserving the desired structure at negligible cost in the following manner. Note that $\tilde{W} = \mathbf{V}_m \tilde{G}$ where the columns of $\tilde{G}$ consist of real and imaginary parts of each g. We compute the small QR decomposition $\tilde{G} = GR$. Upon defining $W = \mathbf{V}_m G$, $\Theta = R\tilde{\Theta}R^{-1}$, and $\mathbf{F} = R^{-T}\tilde{\mathbf{F}}$, one may show that W has orthonormal columns and (5.5) holds. Since W would automatically have orthonormal columns for a symmetric problem, we henceforth assume without lack of generality that W has orthonormal columns.

5.2 Ritz-augmented Arnoldi for $Ax = b$

We now describe a method for building an augmented Krylov subspace of the form (5.1) where the augmented subspace $\mathbb{W}$ consists of Ritz vectors. Though our eventual goal is to use this to solve matrix equations, we first treat the linear system case with single column right-hand sides for simplicity of exposition. The amount of computational work and storage is about the same as established Krylov subspace methods; for instance, using the basis to perform a Galerkin projection to solve a linear system entails about as much work and storage as FOM. To the best of our knowledge, the proposed method for building an augmented space is novel. Using Ritz vectors for $\mathbb{W}$ is not new; what is new is how to build the space and use it in an efficient manner.

We first assume that we have built some Krylov subspace $\mathbb{K}_\ell(A, b_0)$ for some column vector b_0 and have extracted q Ritz vectors $w_1, \ldots, w_q$. The subspace spanned by these Ritz vectors shall be denoted $\mathbb{W}$. We describe a

way to build an orthonormal basis of

$$\mathbb{U}_m = \mathbb{W} + \mathbb{K}_m\left(A, b\right),$$

and use information generated in the process to produce an approximation to the solution of a linear system $Ax = b$ efficiently via Galerkin projection with criteria for stopping the iteration at a small additional cost.

5.2.1 Building the augmented space

Let W be such that $W^TW = I$ and range $(W) = \mathbb{W}$. We seek an algorithm which will deliver vectors $u_1, \ldots, u_m$ such that

$$U_m = [W, u_1, \ldots, u_m] \text{ is orthonormal and range}\left(U_m\right) = \mathbb{U}_m.$$

We begin by orthogonalizing b with respect to W and setting u_1 to be the resulting vector after normalizing. This is accomplished by calculating $\beta = W^Tb$, orthogonalizing $\tilde{u} = b - W\beta$, calculating $\zeta = \|\tilde{u}\|_2$ and defining $u_1 = \tilde{u}/\zeta$. Mathematically speaking, since each u_j is a member of $\mathbb{U}_m$, it must necessarily be of the form

$$u_j = Wd_j + p_j(A)b \text{ for some } p_j \in \mathbb{P}_{j-1}, \tag{5.8}$$

where p_j is a $(j-1)^{\text{th}}$ degree polynomial and $d_j \in \mathbb{R}^q$. This is clearly true for $j = 1$. We seek a rule for obtaining an auxiliary vector $\tilde{u}$ such that upon orthogonalization with the columns of W and $u_1, \ldots, u_j$, the new u_{j+1} will be of the form required form (5.8).

To clarify what is meant by auxiliary vector, observe line 3 of Algorithm 2.1 to see this vector for the standard Arnoldi algorithm. When proving that the Arnoldi algorithm actually builds the desired Krylov subspace $\mathbb{K}_j$, one observes that a given basis vector is implicitly of the form $p_j(A)b$ for some polynomial $p_j \in \mathbb{P}_{j-1}$. Multiplication by A will result in the auxiliary vector implicitly being of the form $Ap_j(A)b =: q_j(A)b$ with $\deg(q_j) = j$, i.e., of one degree higher. This shows that the next Krylov subspace $\mathbb{K}_{j+1}$ is spanned by the resulting set of vectors. This auxiliary vector is then orthogonalized against

existing basis vectors and normalized to obtain the next basis vector, and the process is repeated.

Notice that if we attempt this same approach by proposing $\tilde{u} = Au_j$, then (5.4) and (5.8) show that

$$\tilde{u} = W\Theta d_j + \left(\vec{f}^T d_j\right) v_{\ell+1} + Ap_j(A)b. \tag{5.9}$$

where $v_{\ell+1}$ is the last Arnoldi vector of $\mathbb{K}_\ell(A, b_0)$. While the degree of the associated polynomial has been incremented, the presence of the $v_{\ell+1}$ shows that the resulting auxiliary vector has components in the direction of $v_{\ell+1}$, a vector that is not necessarily in $\mathbb{U}_{j+1}$. The property that the *all* eigenresiduals point in a known direction allows us to correct for components in this unwanted direction at a small cost. A slight modification of our auxiliary vector will ensure that that resulting algorithm builds the intended space.

Keeping (5.3) in mind, when building our augmented space we propose the following auxiliary vector

$$\tilde{u} = Au_j - \left(\vec{f}^T d_j\right) v_{\ell+1}. \tag{5.10}$$

Our reasoning is as follows. If one assumes that u_j is implicitly of the form (5.8), then by virtually the same calculation used to show (5.9), it follows that

$$Au_j - \left(\vec{f}^T d_j\right) v_{\ell+1} = W\Theta d_j + Ap_j(A)b.$$

The left-hand side of this equality is what we have proposed for $\tilde{u}$ and what may be calculated in practice, while the right hand side is an implicit representation of this same vector. This representation shows that the degree of the associated polynomial is incremented by one in full analogy with the Arnoldi method while remaining in the space $\mathbb{U}_{j+1}$.

For a fully working method, we note that computing (5.10) requires knowing d_j. Note that the construction of u_1 explicitly yields $d_1 = -\beta/\zeta$ as a byproduct of the operations performed. If u_j and $\tilde{u}$ are given as stated, we first orthogonalize $\tilde{u}$ with respect to W and denote the orthogonalization coefficients by h_{ij}^W. We next orthogonalize with respect to $u_1, \ldots, u_j$ and normalize

the resulting vector, and denote these coefficients coefficients by h_{ij}^U. The next basis vector will then be of the form

$$u_{j+1} = \frac{1}{h_{j+1,j}^U}\left(\tilde{u} - \sum_{i=1}^{q} h_{ij}^W w_i - \sum_{i=1}^{j} h_{ij}^U u_i\right). \tag{5.11}$$

We plug the implicit representations $u_i = Wd_i + p_i(A)b$ into (5.11) and note that $w_i = We_i$ to obtain

$$\begin{aligned} u_{j+1} = \frac{1}{h_{j+1,j}^U}\Bigg(W\Big(\Theta d_j - [h_{1j}^W, \ldots, h_{qj}^W]^T - \sum_{i=1}^{j} h_{ij}^U d_i\Big) \\ + Ap_j(A)b - \sum_{i=1}^{j} h_{ij}^U p_i(A)b\Bigg) \\ = Wd_{j+1} + p_{j+1}(A)b. \end{aligned}$$

This shows that the next basis vector is of the desired form and gives a recurrence relation for the next d_{j+1} in terms of the previous $d_i, i = 1, \ldots, j$. It is worth remarking that the only additional computation of $\mathcal{O}(n)$ is the vector update with $v_{\ell+1}$ to correct $\tilde{u}$, as the d_j are all vectors of size q, and in our applications $q \ll n$. We summarize all details as Algorithm 5.1, and the previous discussion is essentially an inductive proof that this algorithm builds an orthonormal basis of $\mathbb{U}_m$.

It is also worth remarking that a partial analogue of the Lanczos algorithm is possible. By this it is meant that for symmetric matrices, the coefficients required for orthogonalization are available for free. Unfortunately, they are nonzero, so that the vector updates to remove components still need to be performed, and method does not become a short-term recurrence method, but it still is very beneficial with regard to the amount of computation performed.

5.2.2 Performing Galerkin projection

We now describe how, given an orthonormal basis U_j of $\mathbb{U}_j$ built by Algorithm 5.1 and its byproducts, one may use these quantities to perform a

Algorithm 5.1 Ritz-augmented Arnoldi for standard Krylov subspaces

Input: Ritz vectors W, Ritz values Θ, $\vec{f}, v_{\ell+1}, A, b$.

Output: Vectors $u_1, \ldots, u_j$ such that $[W, u_1, \ldots, u_j]$ has orthonormal columns that span $\mathbb{U}_m$.

1: Set $d_1 = W^T b$, $\tilde{u} = b - W d_1$, $\zeta = \|\tilde{u}\|_2$, $u_1 = \tilde{u}/\zeta, d_1 = -d_1/\zeta$.
2: **for** $j = 1, ..., m$ **do**
3: $\quad \tilde{u} = Au_j - (\vec{f}^T d_j) v_{\ell+1}$
4: $\quad$ Orthogonalize $\tilde{u}$ with respect to $w_1, \ldots, w_q$, overwriting $\tilde{u}$ and storing coefficients as h_{ij}^W
5: $\quad$ Orthogonalize $\tilde{u}$ with respect to $u_1, \ldots, u_j$, overwriting $\tilde{u}$ and storing coefficients as h_{ij}^U
6: $\quad h_{j+1,j}^U = \|\tilde{u}\|_2$
7: $\quad u_{j+1} = \tilde{u}/h_{j+1,j}^U$
8: $\quad d_{j+1} = \frac{1}{h_{j+1,j}^U} \left(\Theta d_j - [h_{1j}^W, \ldots, h_{qj}^W]^T - \sum_{i=1}^{j} h_{ij}^U d_i \right)$
9: **end for**

Galerkin projection approximation to

$$Ax = b.$$

We proceed in a manner that is surely familiar by now; namely, writing our approximate solution $x_j = U_j y_j$ where y_j is chosen according to $U_j^T(b - Ax_j) = 0$, or

$$(U_j^T A U_j) y_j = U_j^T b.$$

A quick calculation shows that $U_j^T b = [\beta^T, \zeta e_j^T]$. For the remaining, the Rayleigh quotient matrix $T_j = U_j^T A U_j$ and the residual r_j, both entail an application of A to elements of the space $\mathbb{U}_j$. The calculation (5.9) shows that applying A to an element of this space results in a vector that is in

$$\mathbb{U}_{j+1} + \text{span}\{v_{\ell+1}\}.$$

We therefore expect to have to account for components of these quantities in the direction of $v_{\ell+1}$.

The following calculations make the remarks of the preceding paragraph explicit. We write $\widehat{U}_j = [u_1, \ldots, u_j]$ so that $U_j = [W, \widehat{U}_j]$. Furthermore, define $H_j^W \in \mathbb{R}^{q\times j}$ and $\underline{H}_j^U \in \mathbb{R}^{j+1\times j}$ as matrices with entires h_{ij}^W and h_{ij}^U, respectively. After each step of the j loop of Algorithm 5.1, it follows that

$$h_{j+1,j}^U u_{j+1} = Au_j - (\vec{f}^T d_j)v_{\ell+1} - \sum_{i=1}^{k} h_{ij}^W w_i - \sum_{i=1}^{j} h_{ij}^U u_i,$$

which is equivalent to

$$Au_j = v_{\ell+1}\vec{f}^T d_j + \sum_{i=1}^{k} h_{ij}^W w_i + \sum_{i=1}^{j+1} h_{ij}^U u_i.$$

Written in matrix form, this yields

$$A\widehat{U}_j = WH_j^W + \widehat{U}_{j+1}\underline{H}_j^U + v_{\ell+1}\vec{f}^T D_j,$$

where $D_j = [d_1, \ldots, d_j]$. This allows us to write

$$A[W, \widehat{U}_j] = [W, \widehat{U}_{j+1}, v_{\ell+1}] \begin{bmatrix} \Theta & H_j^W \\ 0 & \underline{H}_j^U \\ \vec{f}^T & \vec{f}^T D_j \end{bmatrix}.$$

The Rayleigh quotient matrix can be calculated by observing that

$$\begin{bmatrix} W^T \\ \widehat{U}_j^T \end{bmatrix} \begin{bmatrix} W & \widehat{U}_{j+1} & v_{\ell+1} \end{bmatrix} = \begin{bmatrix} I_q & 0_{q\times j+1} & 0_{q\times 1} \\ 0_{j\times q} & [I_j|0_{j\times 1}] & \widehat{U}_j^T v_{\ell+1} \end{bmatrix}$$

and therefore

$$\begin{aligned} T_j &= \begin{bmatrix} I_q & 0_{q\times j+1} & 0_{q\times 1} \\ 0_{j\times q} & [I_j|0_{j\times 1}] & \widehat{U}_j^T v_{\ell+1} \end{bmatrix} \begin{bmatrix} \Theta & H_j^W \\ 0 & \underline{H}_j^K \\ \vec{f}^T & \vec{f}^T D_j \end{bmatrix} \\ &= \begin{bmatrix} \Theta & H_j^W \\ \widehat{U}_j^T v_{\ell+1}\vec{f}^T & H_j^K + \widehat{U}_j^T v_{\ell+1}\vec{f}^T D_j \end{bmatrix} \end{aligned} \tag{5.12}$$

is the desired Rayleigh quotient matrix. The residual of this system satisfies

$$r_j := b - AU_j y_j = [W, \widehat{U}_{j+1}, v_{\ell+1}] \underbrace{\left(\begin{bmatrix} \beta \\ \zeta e_j \end{bmatrix} - \begin{bmatrix} \Theta & H_j^W \\ 0 & \underline{H}_j^K \\ \vec{f}^T & \vec{f}^T D_j \end{bmatrix} y_j \right)}_{=:s_j}$$

so that

$$\|r_j\|^2 = s_j^T \begin{bmatrix} W^T \\ \widehat{U}_{j+1}^T \\ v_{\ell+1} \end{bmatrix} [W, \widehat{U}_{j+1}, v_{\ell+1}] s_j.$$

Seeing some light at the end of the tunnel, we compute

$$\begin{bmatrix} W^T \\ \widehat{U}_{j+1}^T \\ v_{\ell+1}^T \end{bmatrix} [W, \widehat{U}_{j+1}, v_{\ell+1}] = \begin{bmatrix} I_q & 0 & 0 \\ 0 & I_{j+1} & \widehat{U}_{j+1}^T v_{\ell+1} \\ 0 & v_{\ell+1}^T \widehat{U}_{j+1} & 1 \end{bmatrix} \tag{5.13}$$

where $W^T v_{\ell+1} = 0$ since the columns of W are in $\mathbb{K}_\ell(A, b_0)$. The matrix on the right-hand of the above equality is small, as is s_j, so this provides a means of computing $\|x_j\|_2$ at a cost that we consider negligible, with one caveat. Upon further inspection, we notice that some additional $\mathcal{O}(n)$ computation is required; namely, the presence of the $\widehat{U}_{j+1}^T v_{\ell+1}$ that appear in the expression for the Rayleigh quotient (5.12) and the matrix needed for calculating the residual (5.13). Our intuition about the need to account for components in this additional direction $v_{\ell+1}$ has been confirmed. We thus propose to compute and store these at each iterate as $\phi_j = u_j^T v_{\ell+1}$ and $\Phi_j = \widehat{U}_{j+1}^T v_{\ell+1} = [\phi_1, \ldots, \phi_j]^T$. With this final ingredient, we are provided with a complete description of a method that solves $Ax = b$. A full pseudo-code implementation is given in Algorithm 5.2, where the updates for d_{j+1} have been incorporated into inner loops which perform the orthogonalization.

Our remark at the beginning of the chapter regarding comparable computational and storage requirements to FOM may be justified as follows. If FOM were to build a basis of the same dimension, the only additional $\mathcal{O}(n)$ computation incurred by Algorithm 5.2 are a dot product and vector update with $v_{\ell+1}$ at each iterate. Similarly, the only additional $\mathcal{O}(n)$ storage consists of the vector $v_{\ell+1}$. This is advantageous when compared to doubling storage and orthogonalization requirements of other approaches, as well as information required to perform Galerkin projection available at a small cost. Another benefit is that the vectors W that are being reused have already been orthogonalized and need not be orthogonalized again.

Algorithm 5.2 Ritz-augmented Arnoldi for $Ax = b$

Input: Ritz vectors W, Ritz values Θ, tolerance τ, $\vec{f}, v_{\ell+1}, A, b$.

Output: Approximate solution x_m to $Ax = b$ with $\|r_m\|_2/\|b\|_2 \leq \tau$

Set $d_1 = W^T b$, $\tilde{u} = b - W d_1$, $\zeta = \|\tilde{u}\|_2$, $u_1 = \tilde{u}/\zeta, d_1 = -d_1/\zeta$.

$\phi_1 = u_1^T v_{\ell+1}$

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

$\tilde{u} = Au_j - (\vec{f}^T d_j) v_{\ell+1}$

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

$h_{ij}^W = w_i^T \tilde{u}$

$\tilde{u} = \tilde{u} - h_{ij}^W w_i$

end for

$d_{j+1} = \Theta d_j - [h_{1j}^W, \ldots, h_{qj}^W]^T$

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

$h_{ij}^K = u_i^T \tilde{u}$

$\tilde{u} = \tilde{u} - h_{ij}^K u_i$

$d_{j+1} = d_{j+1} - h_{ij}^K d_i$

end for

$h_{j+1,j}^K = \|\tilde{u}\|_2$

$u_{j+1} = \tilde{u}/h_{j+1,j}^K$

$d_{j+1} = d_{j+1}/h_{j+1,j}^K$

$\phi_{j+1} = u_{j+1}^T v_{\ell+1}$

Solve $\begin{bmatrix} W & H_j^W \\ \Phi_j \vec{f}^T & H_j^K + \Phi_j \vec{f}^T D_j \end{bmatrix} y_j = \begin{bmatrix} \beta \\ \zeta e_j \end{bmatrix}$

Set $s = \begin{bmatrix} \beta \\ \zeta e_j \end{bmatrix} - \begin{bmatrix} \Theta & H_j^W \\ 0 & \underline{H}_j^K \\ \vec{f}^T & \vec{f}^T D_j \end{bmatrix} y_j$

If $\left(s^T \begin{bmatrix} I_q & 0 & 0 \\ 0 & I_{j+1} & \Phi_{j+1} \\ 0 & \Phi_{j+1}^T & 1 \end{bmatrix} s \right)^{\frac{1}{2}} \leq \tau \|b\|_2$ then stop.

end for

5.3 Modifications for solving generalized Lyapunov equations

With some minor modifications, the proposed augmentation scheme can be paired with the classical iterations described in Chapter 4 to dramatically reduce the number of linear solves with coefficient matrix A required for convergence to an approximate solution of the generalized Lyapunov equation (4.3). We again assume that we are solving

$$AX_{k+1} + X_{k+1}A^T + B_k B_k^T = 0, \tag{5.14}$$

for low-rank factors Z_k of X_k, where $B_k = \mathcal{T}([NZ_k, B])$; see equation (4.8) and the discussion contained around it.

We give a high level description here, and reserve technical details to Appendix C. Supposing that at outer iterate $k-1$ we have solved (5.14) by the extended Krylov subspace method, and the right-hand side B_{k-1} has converged enough so that reuse of information may be beneficial. Our approach to solve the Lyapunov equation that arises at the next outer iterate is to use the Galerkin projection strategy described in Section 2.2 with $\mathbb{S}_m = \mathbb{S}(\mathbb{U}_m, \mathbb{U}_m)$, where $\mathbb{U}_m$ is the augmented block Krylov subspace

$$\mathbb{U}_m = \mathbb{W} + \mathbb{K}_m(A, B_k). \tag{5.15}$$

Here the subspace $\mathbb{W}$ now consists of Ritz vectors from the extended Krylov subspace built at the previous iterate, and we emphasize that the Krylov subspace appearing in (5.15) is now a block Krylov subspace.

We note that the method in Section 5.2.2 used Ritz vectors from a standard Krylov subspace to augment a standard Krylov subspace and solve a linear system. Fortuantely, every step of this procedure extends in a natural way to take Ritz vectors from extended Krylov subspaces, build augmented block Krylov subspaces, and solve standard Lyapunov equations. Since extended Krylov subspaces possess the property (5.5), the auxiliary block in standard block Arnoldi (see line 3 of Algorithm 2.2) can be corrected in a

manner analogous to (5.10) to ensure that one builds the intended space. In order to calculate the Rayleigh quotient and residual norm efficiently, there are only a extra few directions for which one must account. These occur in the directions of the columns of $\mathbf{V}_{(\ell+1)}$, the matrix whose columns constitute the last extended Arnoldi vectors. The Rayleigh quotient matrix and residual norm may be computed at what is once again a small additional cost at each iterate. For details, see Appendix C.

One difficulty that arose was an instability in the calculation of the Rayleigh quotient matrix as described in Appendix B, since the Ritz vectors and values that we use for augmentation depend on this quantity. While this does not appear to affect the quality of Lyapunov solvers, we observed that for large block sizes the Ritz values that were delivered were of poor quality and not suitable for our purposes. Since rational Krylov subspaces were originally used for eigenvalue computations, we opted to use a block version of Ruhe's rational Arnoldi algorithm, given as Algorithm A.3. The Ritz vectors were of much better quality. To build an extended Krylov subspace we used poles which alternated between some shift σ and ∞ with $\sigma \neq 0$, so that the space built was technically

$$\mathbb{EK}_m^\sigma(A, B) = \mathbb{K}_m(A, B) + \mathbb{K}_{m-1}\left((A - \sigma I)^{-1}, (A - \sigma I)^{-1} B\right), \tag{5.16}$$

where we used the shift $\sigma = 1$.

We now comment on the selection of which Ritz vectors to use, and how many. We expect the Ritz values from the space (5.16) to give decent approximation to eigenvalues of A that are closest to 1 and largest in magnitude. When we build (5.15), no more information corresponding to $(A - \sigma I)^{-1}$ shall be generated. For this reason, we chose to keep a certain fixed percentage of Ritz vectors that were closest to 1. Our intuition is that since $B_{k+1} \approx B_k$, keeping the part of $\mathbb{EK}_m^\sigma(A, B_k)$ that is (loosely) associated with $\mathbb{K}_{m-1}((A - \sigma I)^{-1}, (A - \sigma I)^{-1} B_k)$ and using this to augment $\mathbb{K}_m(A, B_{k+1})$ will (hopefully) build a space that is close to $\mathbb{EK}_m^\sigma(A, B_{k+1})$, so we typically kept more than half of the basis.

The benefit is that half the basis will already be built, and shifted linear solves with A will no longer need to be performed. It is worth noting that this approach is not really based upon approximating any spectral information to enhance convergence, as in [48]. Instead, it is about the possibility of reusing an old basis to build a new basis and performing Galerkin projection in an efficient manner. In this sense, our approach is more like Krylov subspace recycling [51].

5.4 Numerical experiments

We again present a comparison with the methods described in [7] and the extended Krylov subspace method with no modifications other than truncated right-hand sides. For both tables in this section, `RA-KSM` denotes the Ritz-augmented Krylov subspace approach. Specifically, this denotes a classical iteration of the form (5.14), such that the first two outer iterates are performed by the usual extended Krylov subspace method (also with truncated right-hand sides). Our experience was that one needed a thick right-hand side $B_2 = \mathcal{T}([NZ_1, B])$ in order to build an extended Krylov subspace that had enough Ritz vectors to make the augmentation strategy successful. One this space has been built, we keep 75% of the Ritz vectors whose Ritz values were closest to 1 and use the augmentation strategy described in the previous section.

The memory for `RA-KSM` was comparable with `EKSM` (in fact, it often used slightly more) so we do not enter a detailed discussion of this. However, CPU times were competitive with the methods in [7], and since linear solves are stopped after the second outer iterate, the number of linear solves required to complete the outer iteration was dramatically lower for `RA-KSM`.

We first consider the same symmetric problem involving boundary control of the heat equation of size $n = 22500$. A comparison of CPU times and number of linear solves is given in Table 5.1. We note that `EKSM` was the second best solver for this problem, and the Lanczos analogue was required in order to make the `RA-KSM` method the most competitive. In Figure 5.1, we

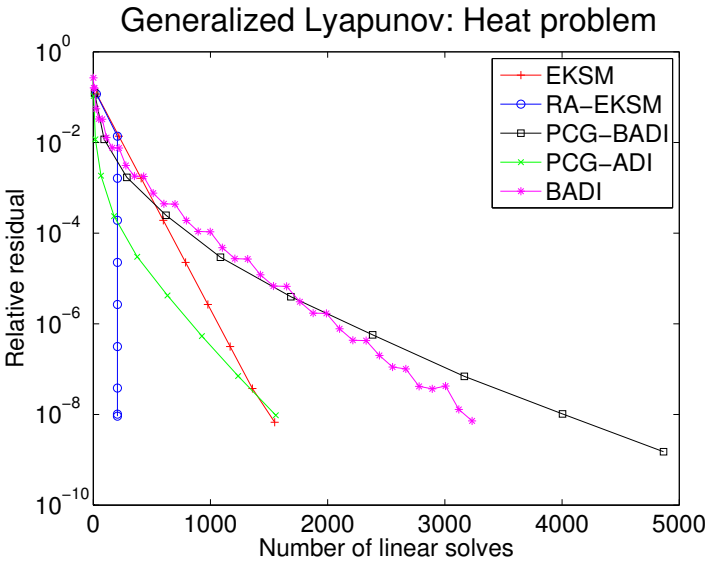


Figure 5.1: Number of linear solves required by different methods to solve a generalized Lyapunov equation with a symmetric coefficient matrix of size $n = 22500$. Norm of the relative residual plotted against the number of linear solves.

Method	**CPU (s)**	**Solves**
`EKSM`	139.2	1978
`RA-KSM`	**105.2**	**206**
`BiADI`	224.6	3345
`PCG-BADI`	363.7	4868
`PCG-ADI`	164.0	1557

Table 5.1: Performance comparison between several generalized Lyapunov solvers for a symmetric heat problem of size $n = 22500$.

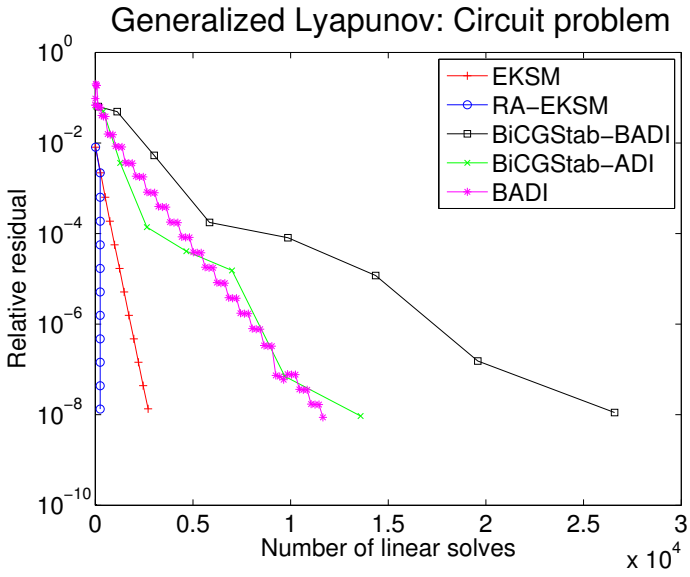


Figure 5.2: Number of linear solves required by different methods to solve a generalized Lyapunov equation with a non-symmetric coefficient matrix of size $n = 10100$. The norm of the relative residual is plotted against the number of linear solves.

plot the relative residual norm at a given outer iterate against the number of linear solves with coefficient matrix A that the method required up until that point.

It is pleasing to notice that after the second outer iterate, the plot for `RA-KSM` drops straight down. The Ritz vectors generated at outer iterate two produce augmented subspaces at successive outer iterates that are rich enough to obtain convergence. At later outer iterates, the only new information generated comes from matrix vector products with A, but the individual Lyapunov solvers do not suffer from any of the negative aspects of using a purely polynomial approximation. In other words, the Ritz vectors provide the necessary enrichment to alleviate such difficulties.

We next consider the non-symmetric circuit problem, again with $n = 10100$, with performance metrics presented in Table 5.2. Here `RA-KSM` obtains the best CPU time by a larger margin than in the symmetric case, and the number of required linear solves is again reduced dramatically when com-

Method	CPU time	# of solves
EKSM	701.6	3692
RA-KSM	**196.3**	**253**
BADI	245.3	11654
BiCGstab-BADI	818.2	34300
BiCGstab-ADI	390.1	13575

Table 5.2: Performance comparison between several generalized Lyapunov solvers for the non-symmetric circuit problem of size $n = 10100$.

pared with other existing methods. Figure 5.2 contains another plot of relative residual norms against total number of linear solves, where we see a similar effect as in the symmetric example.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary and conclusions

We have considered low-rank solution techniques for large-scale matrix equations, typically based around Galerkin projection onto enriched subspaces and low-rank analogues of classical iterative methods. We have extended these techniques to a certain class of systems of matrix equations, called a constrained Sylvester equation. This was done by reformulating existing methods for the small-scale case to be suitable for large-scale problems. A new subspace that approximates the well-known enriched spaces are a necessary part of the procedure, and numerical evidence for the effectiveness of our approach is provided for a variety of problems.

We have also proposed solvers for large-scale generalized Lyapunov equations, and compare them to existing approaches. We believe that what results is competitive with respect to several metrics, such as CPU usage, memory usage, and linear system solves. Our approach is essentially to convert the problem to a sequence of Lyapunov equations and leverage powerful existing Lyapunov solvers. In doing so, we provide theory for several kinds of inexactness that arise. We also propose some modifications to the well-known

extended Krylov subspace method that make it suitable for newer problems and utilize existing theory to justify our approach. In addition, we have designed what appears to be a novel algorithm for an alternative approach that can drastically reduce the required number of linear solves.

6.2 Future work

There are several interesting avenues for future research. Low-rank Krylov subspace solvers have recently been shown to be applicable to time-dependent PDE-constrained optimization, and forward problem solves consist of solving a Sylvester equation. The rank of the right-hand sides in these equations are almost certainly larger than what is typical in control theory, and it would be interesting to attempt separating the right-hand sides as was done for generalized Lyapunov equations. It would also be of interest to see how the Ritz-augmented Arnoldi algorithm fits into the literature on augmented and Krylov subspace recycling techniques, since it is applicable to linear systems.

REFERENCES

[1] Athanasios C. Antoulas. *Approximation of large-scale dynamical systems*, volume 6 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[2] Athanasios C. Antoulas, Danny C. Sorensen, and Yunkai Zhou. On the decay rate of Hankel singular values and related issues. *Systems Control Lett.*, 46(5):323–342, 2002.

[3] Zhaojun Bai and Daniel Skoogh. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra Appl.*, 415(2-3):406–425, 2006.

[4] Jewel B. Barlow, Moghen M. Monahemi, and Dianne P. O'Leary. Constrained matrix Sylvester equations. *SIAM J. Matrix Anal. Appl.*, 13(1):1–9, 1992.

[5] Richard H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Communications of the ACM*, 15(9):820–826, 1972.

[6] Bernhard Beckermann. An error analysis for rational Galerkin projection applied to the Sylvester equation. *SIAM J. Numer. Anal.*, 49(6):2430–2450, 2011.

[7] Peter Benner and Tobias Breiten. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numer. Math.*, 124(3):441–470, 2013.

[8] Peter Benner and Tobias Damm. Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems. *SIAM J. Control Optim.*, 49(2):686–711, 2011.

[9] Peter Benner, Jing-Rebecca Li, and Thilo Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numer. Linear Algebra Appl.*, 15(9):755–777, 2008.

[10] Michele Benzi. Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.*, 182(2):418–477, 2002.

[11] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*. Academic Press, New York-London, 1979.

[12] Rajendra Bhatia and Peter Rosenthal. How and why to solve the operator equation $AX - XB = Y$. *Bull. London Math. Soc.*, 29(1):1–21, 1997.

[13] Carlo Bruni, Gianni DiPillo, and Giorgio Koch. Bilinear systems: an appealing class of "nearly linear" systems in theory and applications. *IEEE Trans. Automatic Control*, AC-19:334–348, 1974.

[14] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl.*, 4(1):43–66, 1997.

[15] Tobias Damm. *Rational matrix equations in stochastic control*, volume 297 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, 2004.

[16] Tobias Damm. Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numer. Linear Algebra Appl.*, 15(9):853–871, 2008.

[17] Timothy A. Davis. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[18] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19(2):400–408, 1982.

[19] Vladimir Druskin and Leonid Knizhnerman. Extended Krylov subspaces: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771, 1998.

[20] Vladimir Druskin, Leonid Knizhnerman, and Valeria Simoncini. Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. *SIAM J. Numer. Anal.*, 49(5):1875–1898, 2011.

[21] Vladimir Druskin, Chad Lieberman, and Mikhail Zaslavsky. On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM J. Sci. Comput.*, 32(5):2485–2496, 2010.

[22] Vladimir Druskin and Valeria Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems Control Lett.*, 60(8):546–560, 2011.

[23] Harry Dym. *Linear algebra in action*, volume 78 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2007.

[24] Luc Giraud, Julien Langou, Miroslav Rozložník, and Jasper van den Eshof. Rounding error analysis of the classical Gram-Schmidt orthogonalization process. *Numer. Math.*, 101(1):87–100, 2005.

[25] G. H. Golub, S. Nash, and Charles F. Van Loan. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, 24(6):909–913, 1979.

[26] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[27] Lars Grasedyck. Existence of a low-rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation. *Numer. Linear Algebra Appl.*, 11(4):371–389, 2004.

[28] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitt.*, 36(1):53–78, 2013.

[29] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[30] Serkan Gugercin and Athanasios C. Antoulas. A survey of model reduction by balanced truncation and some new results. *Internat. J. Control*, 77(8):748–766, 2004.

[31] Martin H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: an introduction. In A.H. Siddiqi, I.S. Duff, and O. Christensen, editors, *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, pages 420–447. Anamaya, New Delhi, 2007.

[32] Stefan Güttel. *Rational Krylov methods for operator functions*. PhD thesis, Department of mathematics and computer science, TU Bergakademie Freiberg, Germany, 2010.

[33] Carsten Hartmann, Boris Schäfer-Bung, and Anastasia Thöns-Zueva. Balanced averaging of bilinear systems with applications to stochastic control. *SIAM J. Control Optim.*, 51(3):2356–2378, 2013.

[34] Mohammed Heyouni. Extended Arnoldi methods for large low-rank Sylvester matrix equations. *Appl. Numer. Math.*, 60(11):1171–1182, 2010.

[35] Roger A. Horn and Charles R. Johnson. *Topics in matrix analysis*. Cambridge University Press, Cambridge, 1991.

[36] Dan Y. Hu and Lothar Reichel. Krylov-subspace methods for the Sylvester equation. *Linear Algebra Appl.*, 172:283–313, 1992.

[37] Imad M. Jaimoukha and Ebrahim M. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Numer. Anal.*, 31(1):227–251, 1994.

[38] Leonid Knizhnerman and Valeria Simoncini. Convergence analysis of the extended Krylov subspace method for the Lyapunov equation. *Numer. Math.*, 118(3):567–586, 2011.

[39] Jan G. Korvink and Evgenii B. Rudnyi. Oberwolfach benchmark collection. In Peter Benner, Danny C. Sorensen, and Volker Mehrmann, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 311–315. Springer Berlin Heidelberg, 2005.

[40] Daniel Kressner and Christine Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM J. Matrix Anal. Appl.*, 31(4):1688–1714, 2009/10.

[41] Daniel Kressner and Christine Tobler. Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM J. Matrix Anal. Appl.*, 32(4):1288–1316, 2011.

[42] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2007.

[43] Jing-Rebecca Li and Jacob White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.

[44] Jörg Liesen and Zdenek Strakos. *Krylov subspace methods: principles and analysis*. Oxford University Press, Oxford, 2012.

[45] Yiding Lin and Valeria Simoncini. Minimal residual methods for large scale Lyapunov equations. *Appl. Numer. Math.*, 72:52–71, 2013.

[46] Moghen M. Monahemi, Jewel B. Barlow, and Dianne P. O'Leary. Design of reduced-order observers with precise loop transfer recovery. *Journal of guidance, control, and dynamics*, 15(6):1320–1326, 1992.

[47] Marlliny Monsalve and Daniel B. Szyld. Inexact Newton with Krylov projection and recycling for Riccati equations. Poster presented at the Conference on Numerical Linear Algebra: Perturbation, Performance, and Portability, Austin, Texas, 2010.

[48] Ronald B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16(4):1154–1171, 1995.

[49] Ronald B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Math. Comp.*, 65(215):1213–1230, 1996.

[50] Ronald B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Anal. Appl.*, 21(4):1112–1135, 2000.

[51] Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[52] Thilo Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, 40(2):139–144, 2000.

[53] Thilo Penzl. Algorithms for model reduction of large dynamical systems. *Linear Algebra Appl.*, 415(2-3):322–343, 2006.

[54] Antonio Ruberti, Alberto Isidori, and Paolo d'Alessandro. *Theory of bilinear dynamical systems*. Springer-Verlag, Vienna-New York, 1972. Course

held at the Department for Automation and Information, July 1972, International Centre for Mechanical Sciences, Udine. Courses and Lectures, No. 158.

[55] Axel Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra Appl.*, 58:391–405, 1984.

[56] Axel Ruhe. The rational Krylov algorithm for nonsymmetric eigenvalue problems. III. Complex shifts for real matrices. *BIT*, 34(1):165–176, 1994.

[57] Axel Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. In *Recent advances in iterative methods*, volume 60 of *IMA Vol. Math. Appl.*, pages 149–164. Springer, New York, 1994.

[58] Axel Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. II. Matrix pairs. *Linear Algebra Appl.*, 197/198:283–295, 1994.

[59] Axel Ruhe. Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils. *SIAM J. Sci. Comput.*, 19(5):1535–1551, 1998.

[60] Youcef Saad. *Numerical methods for large eigenvalue problems*. Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, Manchester; Halsted Press, New York, 1992.

[61] Yousef Saad. Numerical solution of large Lyapunov equations. In Marinus A. Kaashoek, Jan H. van Schuppen, and André C. Ran, editors, *Signal processing, scattering and operator theory, and numerical methods (Amsterdam, 1989)*, volume 5 of *Progr. Systems Control Theory*, pages 503–511. Birkhäuser, Boston, 1990.

[62] Yousef Saad. Analysis of augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.*, 18(2):435–449, 1997.

[63] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, PA, second edition, 2003.

[64] Jens Saak. *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*. PhD thesis, Department of Mathematics, TU Chemnitz, July 2009.

[65] Hans Schneider. Positive operators and an inertia theorem. *Numer. Math.*, 7:11–17, 1965.

[66] Stephen D. Shank and Valeria Simoncini. Krylov subspace methods for large-scale constrained Sylvester equations. *SIAM J. Matrix Anal. Appl.*, 34(4):1448–1463, 2013.

[67] Valeria Simoncini. On the numerical solution of $AX - XB = C$. *BIT*, 36(4):814–830, 1996.

[68] Valeria Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29(3):1268–1288, 2007.

[69] Valeria Simoncini. Computational methods for linear matrix equations. *Preprint*, 1, 2013.

[70] Valeria Simoncini and Vladimir Druskin. Convergence analysis of projection methods for the numerical solution of large Lyapunov equations. *SIAM J. Numer. Anal.*, 47(2):828–843, 2009.

[71] Valeria Simoncini and Daniel B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003.

[72] Valeria Simoncini and Daniel B. Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numer. Linear Algebra Appl.*, 14(1):1–59, 2007.

[73] Valeria Simoncini and Daniel B. Szyld. Interpreting IDR as a Petrov-Galerkin method. *SIAM J. Sci. Comput.*, 32(4):1898–1912, 2010.

[74] Danny C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[75] Lloyd N. Trefethen and David Bau, III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[76] Lloyd N. Trefethen et al. *Chebfun Version 4.2*. The Chebfun Development Team, 2011. `http://www.chebfun.org/`.

[77] Ulrich Trottenberg, Cornelius W. Oosterlee, and Anton Schüller. *Multigrid*. Academic Press, San Diego, CA, 2001.

[78] Jasper van den Eshof and Marlis Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27(4):1438–1457 (electronic), 2006.

[79] Henk A. van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2003.

[80] Richard S. Varga. *Matrix iterative analysis*, volume 27 of *Springer Series in Computational Mathematics*. Springer, Berlin, second edition, 2000.

APPENDIX A

Algorithms

In this appendix we collect several algorithms used to build various spaces described throughout the thesis. We emphasize that we focus on lower-level details that can have impacts on numerical stability. All algorithms are implemented with reorthogonalization to further ensure as much orthogonality as possible. We adopt the `+=` notation which is more common in texts on computer science, i.e., by `a += b` we mean `a = a + b`. Any value appearing in an assignment that has not been previously been assigned will be assumed to be zero.

Algorithm A.1 Block Arnoldi with reorthogonalization

Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times r}$, integer m

Output: Matrix $\mathbf{V}_m = [\mathbf{V}_{(1)}, \ldots, \mathbf{V}_{(m)}]$, with $\mathbf{V}_{(j)} = [v_{r(j-1)+1}, \ldots, v_{rj}]$ whose columns form an orthonormal basis of $\mathbb{K}_m(A, B)$. Block upper Hessenberg matrix $\underline{\mathbf{H}}_m$ such that $A\mathbf{V}_m = \mathbf{V}_{m+1}\underline{\mathbf{H}}_m$.

Compute the reduced QR decomposition $\mathbf{V}_{(1)}R = B$
for $j = 1, \ldots, m$ **do**
    $W = A\mathbf{V}_{(j)}$, write $W = [w_1, \ldots, w_r]$
    for $k = 1, \ldots, r$ **do**
        for reorth $= 1, 2$ **do**
            for $i = 1, \ldots, rj + k - 1$ **do**
                $h = w_k^T v_i$
                $\mathbf{H}_{i,r(j-1)+k} += h$
                $w_k = w_k - hv_i$
            end for
        end for
        $\mathbf{H}_{rj+k,r(j-1)+k} = \|w_k\|_2$
        $v_{rj+k} = w/\mathbf{H}_{rj+k,r(j-1)+k}$
    end for
end for

Algorithm A.2 Block extended Arnoldi with reorthogonalization

Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times r}$, integer m

Output: Matrix $\mathbf{V}_m = [\mathbf{V}_{(1)}, \ldots, \mathbf{V}_{(m)}]$, with $\mathbf{V}_{(j)} = [v_{2r(j-1)+1}, \ldots, v_{2rj}]$ whose columns form an orthonormal basis of $\mathbb{EK}_m(A, B)$.

Compute the reduced QR decomposition $\mathbf{V}_{(1)}R = [B, A^{-1}B]$
for $j = 1, \ldots, m$ **do**
    Partition $\mathbf{V}_{(j)} = \left[\mathbf{V}_{(j)}^{[1]}, \mathbf{V}_{(j)}^{[2]}\right]$ into two $n \times r$ blocks
    $W = \left[A\mathbf{V}_{(j)}^{[1]}, A^{-1}\mathbf{V}_{(j)}^{[2]}\right]$, write $W = [w_1, \ldots, w_{2r}]$
    for $k = 1, \ldots, 2r$ **do**
        for reorth $= 1, 2$ **do**
            for $i = 1, \ldots, 2rj + k - 1$ **do**
                $h = w_k^T v_i$
                $\mathbf{H}_{i,2r(j-1)+k} += h$
                $w_k = w_k - hv_i$
            end for
        end for
        $\mathbf{H}_{2rj+k,r(j-1)+k} = \|w_k\|_2$
        $v_{2rj+k} = w/\mathbf{H}_{2rj+k,r(j-1)+k}$
    end for
end for

Algorithm A.3 Block rational Arnoldi with reorthogonalization

Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times r}$, integer m, shifts $\mathbf{s} = (s_1, \dots s_m)$

Output: Matrix $\mathbf{V}_m = [\mathbf{V}_{(1)}, \dots, \mathbf{V}_{(m)}]$, with $\mathbf{V}_{(j)} = [v_{r(j-1)+1}, \dots, v_{rj}]$ whose columns form an orthonormal basis of $\mathbb{RK}_m(A, B, \mathbf{s})$.

Compute the reduced QR decomposition $\mathbf{V}_{(1)}R = B$
for $j = 1, \dots, m$ **do**
    $W = (I - A/s_j)^{-1}A\mathbf{V}_{(j)}$, write $W = [w_1, \dots, w_r]$
    for $k = 1, \dots, r$ **do**
        for reorth $= 1, 2$ **do**
            for $i = 1, \dots, rj + k - 1$ **do**
                $h = w_k^T v_i$
                $\mathbf{H}_{i,r(j-1)+k} += h$
                $w_k = w_k - hv_i$
            end for
        end for
        $\mathbf{H}_{rj+k,r(j-1)+k} = \|w_k\|_2$
        $v_{rj+k} = w/\mathbf{H}_{rj+k,r(j-1)+k}$
    end for
end for

APPENDIX B

Obtaining the Rayleigh quotient matrix for extended Krylov subspaces

In this appendix we describe an efficient numerical procedure for obtaining the Rayleigh quotient matrix as a byproduct of building an extended Krylov subspace. Our approach follows [68]; we provide the details for completeness, and also include details for block spaces.

After performing the orthogonalization, what results from performing the inner j loop is

$$\mathbf{V}_{(j+1)}\mathbf{H}_{(j+1,j)} = \left[A\mathbf{V}_{(j)}^{[1]}, A^{-1}\mathbf{V}_{(j)}^{[2]}\right] - \sum_{i=1}^{j} \mathbf{V}_{(i)}\mathbf{H}_{(ij)}. \tag{B.1}$$

We partition each $\mathbf{H}_{(ij)}$ as

$$H_{ij} = \begin{bmatrix}\mathbf{H}_{(ij)}^{(:,1)} & \mathbf{H}_{(ij)}^{(:,2)}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{(ij)}^{(11)} & \mathbf{H}_{(ij)}^{(12)} \\ \mathbf{H}_{(ij)}^{(21)} & \mathbf{H}_{(ij)}^{(22)}\end{bmatrix}$$

and we denote subblocks of the block matrix

$$\mathbf{H}_m = \begin{bmatrix}\mathbf{H}_{(11)} & \dots & \mathbf{H}_{(1m)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{(m1)} & \dots & \mathbf{H}_{(mm)}\end{bmatrix}$$

with a similar partitioning and notation for the Rayleigh quotient matrix $\mathbf{T}_m = \mathbf{V}_m^T A \mathbf{V}_m$. Looking at the first r columns of (B.1) implies that for $j = 1, \ldots, m$ we have

$$\begin{aligned}
\mathbf{V}_{(j+1)}\mathbf{H}^{(:,1)}_{(j+1,j)} &= A\mathbf{V}^{[1]}_{(j)} - \sum_{i=1}^{j} V_i \mathbf{H}^{(:,1)}_{(ij)} \\
\Rightarrow A\mathbf{V}^{[1]}_{(j)} &= \sum_{i=1}^{j+1} \mathbf{V}_{(i)}\mathbf{H}^{(:,1)}_{(ij)} \\
\Rightarrow \mathbf{V}^T_{(i)}\mathbf{A}\mathbf{V}^{[1]}_{(j)} &= H^{(:,1)}_{(ij)}
\end{aligned}$$

which yield the odd columns of $\mathbf{T}_m$. For the even columns, upon writing

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

note that inspecting the second r columns of the first QR decomposition implies

$$\begin{aligned}
A^{-1}B &= \mathbf{V}^{[1]}_{(1)} R_{12} + \mathbf{V}^{[2]}_{(1)} R_{22} \\
\Rightarrow B &= A\mathbf{V}^{[1]}_{(1)} R_{12} + A\mathbf{V}^{[2]}_{(1)} R_{22} \\
\Rightarrow \mathbf{V}_2^T B &= \mathbf{V}_2^T A\mathbf{V}^{[1]}_{(1)} R_{12} + \mathbf{V}_2^T A\mathbf{V}^{[2]}_{(1)} R_{22} \\
\Rightarrow \begin{bmatrix} R_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} &= \begin{bmatrix} \mathbf{T}^{(:,1)}_{(11)} \\ \mathbf{T}^{(:,1)}_{(21)} \end{bmatrix} R_{12} + \begin{bmatrix} \mathbf{T}^{(:,2)}_{(11)} \\ \mathbf{T}^{(:,2)}_{(21)} \end{bmatrix} R_{22}
\end{aligned}$$

which yields the first even column of $\mathbf{T}_m$. For the remaining even columns, looking at the second r columns of (B.1) implies that for $j = 1, \ldots, m$

$$\begin{aligned}
\mathbf{V}^{[1]}_{(j+1)}\mathbf{H}^{(12)}_{(j+1,j)} + \mathbf{V}^{[2]}_{(j+1)}\mathbf{H}^{(22)}_{(j+1,j)} &= A^{-1}\mathbf{V}^{[2]}_{(j)} - \sum_{i=1}^{j} \mathbf{V}_i \mathbf{H}^{(:,2)}_{(ij)} \\
\Rightarrow A\mathbf{V}^{[2]}_{(j+1)} &= \left(\mathbf{V}^{[2]}_{(j)} - \sum_{i=1}^{j} A\mathbf{V}_i \mathbf{H}^{(:,2)}_{(ij)} - A\mathbf{V}^{[1]}_{(j+1)}\mathbf{H}^{(12)}_{(j+1,j)} \right) \left(\mathbf{H}^{(22)}_{(j+1,j)} \right)^{-1}
\end{aligned}$$

so that upon reindexing j and left multiplication by $\mathbf{V}_{j+1}^T$ it follows that for $j = 2, \ldots, m+1$, one has that $\mathbf{V}_{j+1}^T A \mathbf{V}_{(j)}^{[2]}$ is equal to

$$\left(\mathbf{V}_{j+1}^T \mathbf{V}_{(j-1)}^{[2]} - \sum_{i=1}^{j-1} \mathbf{V}_{j+1}^T A \mathbf{V}_{(i)} \mathbf{H}_{(i,j-1)}^{(:,2)} - \mathbf{V}_{j+1}^T A \mathbf{V}_{(j)}^{[1]} \mathbf{H}_{(j,j-1)}^{(12)} \right) \left(\mathbf{H}_{(j,j-1)}^{(22)} \right)^{-1}.$$

Above is the block of columns that we desire; note that $\mathbf{V}_{j+1}^T \mathbf{V}_{(j)}^{[2]} = e_{2(j-1)}^{[2(j+1)]} \otimes I_r$ and

$$\begin{aligned} \mathbf{C}_j &:= \sum_{i=1}^{j-1} \mathbf{V}_{j+1}^* A \mathbf{V}_{(i)} \mathbf{H}_{(i,j-1)}^{(:,2)} + \mathbf{V}_{j+1}^T A \mathbf{V}_{(j)}^{[1]} \mathbf{H}_{(j,j-1)}^{(12)} \\ &= \begin{bmatrix} \mathbf{T}_{(11)} & \cdots & \mathbf{T}_{(1,j-1)} & \mathbf{T}_{(1,j)}^{(:,1)} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{T}_{(j+1,1)} & \cdots & \mathbf{T}_{(j+1,j-1)} & \mathbf{T}_{(j+1,j)}^{(:,1)} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{(1,j-1)}^{(:,2)} \\ \vdots \\ \mathbf{H}_{(j-1,j-1)}^{(:,2)} \\ \mathbf{H}_{(j,j-1)}^{(12)} \end{bmatrix} \end{aligned}$$

so that the desired update rule for the even columns (excluding the first) is

$$\mathbf{V}_{j+1}^T A \mathbf{V}_{(j)}^{[2]} = \left(e_{2(j-1)}^{[2(j+1)]} \otimes I_r - \mathbf{C}_j \right) \left(\mathbf{H}_{(j,j-1)}^{(22)} \right)^{-1}.$$

APPENDIX C

Details for Ritz-augmented Lyapunov solvers

Let $A \in \mathbb{R}^{n\times n}$ and $B_0 \in \mathbb{R}^{n\times r_0}$. First, note that the extended Arnoldi relation (2.26) implies that a Ritz vector w of $\mathbb{EK}_\ell(A, B_0)$ with Ritz value θ satisfies

$$Aw = \theta w + \mathbf{V}_{(\ell+1)} f \tag{C.1}$$

where f is the column vector $f = \mathbf{T}_{(\ell+1,\ell)}\mathbf{E}_\ell^T g$ and g is the eigenvector of the Rayleigh quotient matrix corresponding to θ. Suppose we are given q such Ritz vectors, denoted $w_1, \ldots, w_q$. We assemble matrices W, Θ, and $F \in \mathbb{R}^{2r_0\times q}$ such that range $(W) = \operatorname{span}\{w_1, \ldots, w_q\}$, W has orthonormal columns, and

$$AW = W\Theta + \mathbf{V}_{(\ell+1)} F$$

by using the relation (C.1) and the procedure described in Section 5.1.2.

We now describe an algorithm for building

$$\mathbb{U}_m = \mathbb{W} + \mathbb{K}_m(A, B). \tag{C.2}$$

which is given in Algorithm C.1. Here $\mathbb{W}$ consists of Ritz vectors $w_1, \ldots, w_q$ that come from the extended Krylov subspace $\mathbb{EK}_\ell(A, B_0)$, and we write the matrix $B = [b_1, \ldots, b_r] \in \mathbb{R}^{n\times r}$ in terms of columns. Compute $B_W = W^T B$ and a reduced QR decomposition $\mathbf{U}_{(1)} R = B - W B_W$, and set $D_1 = -B_W R^{-1}$.

Algorithm C.1 Ritz-augmented Arnoldi for block spaces

Input: A, B, Ritz vectors W and Ritz values Θ from $\mathbb{EK}_\ell(A, B_0)$, $\mathbf{V}_{(\ell+1)}$, F.

Output: Matrix $\widehat{\mathbf{U}}_m = [\mathbf{U}_{(1)}, \ldots, \mathbf{U}_{(m)}]$ such that the columns of $\mathbf{U}_m = [W, \widehat{\mathbf{U}}_m]$ form an orthonormal basis of $\mathbb{W} + \mathbb{K}_m(A, B)$.

Set $B_W = W^T B$
Compute a reduced QR decomposition $\mathbf{U}_{(1)} R_B = B - W B_W$
Set $D_1 = -B_W R_B^{-1}$
for $j = 1, \ldots, m$ **do**
    $\tilde{U} = A\mathbf{U}_{(j)} - \mathbf{V}_{(\ell+1)} F D_j$
    $\mathbf{H}^W_{(1j)} = W^T \tilde{U}$
    $\tilde{U} = \tilde{U} - W\mathbf{H}^W_{(1j)}$
    for $i = 1, \ldots, j$ **do**
        $\mathbf{H}^U_{(ij)} = \mathbf{U}^T_{(i)} \tilde{U}$
        $\tilde{U} = \tilde{U} - \mathbf{U}_{(i)} \mathbf{H}^U_{(ij)}$
    end for
    $\mathbf{U}_{(j+1)} \mathbf{H}^U_{(j+1,j)} = \tilde{U}$ (reduced QR decompositon)
    $D_{j+1} = \left(D_j - \mathbf{H}^W_{(j)} - \sum_{i=1}^{j} D_i \mathbf{H}^U_{(ij)} \right) (\mathbf{H}^U_{(j+1,j)})^{-1}$
end for

A block of vectors $\mathbf{U}_{(j)}$ such that $\mathbb{U}_{j-1} + \operatorname{range}\left(\mathbf{U}_{(j)}\right) = \mathbb{U}_j$ will necessarily be of the form $\mathbf{U}_{(j)} = W D_j + P_j$, where the i^{th} column of P_j is of the form $\sum_{k=1}^{r} p_k^{(j)}(A) b_i$. A rule for producing an auxiliary block that increments the degree of all such polynomials by one and keeps each column in the next space $\mathbb{U}_{j+1}$ is given by

$$\tilde{\mathbf{U}} = A\mathbf{U}_{(j)} - \mathbf{V}_{(\ell+1)} F D_j.$$

If we orthogonalize each column first with respect to the columns of W and then with respect to all previous blocks of vectors $\mathbf{U}_{(i)}$ for $i = 1, \ldots, j$ in a manner similar to block Arnoldi (see the inner loop of Algorithm 2.2) then what results is a relation of the form

$$\mathbf{U}_{(j+1)} \mathbf{H}^U_{(j+1,j)} = W D_j + P_j - W\mathbf{H}^W_{(j)} - \sum_{i=1}^{j} \mathbf{U}_{(i)} \mathbf{H}^U_{(ij)} \tag{C.3}$$

so that upon ignoring polynomial terms and gathering terms with W one obtains

$$D_{j+1} = \left(D_j - \mathbf{H}^W_{(j)} - \sum_{i=1}^{j} D_i \mathbf{H}^U_{(ij)} \right) (\mathbf{H}^U_{(j+1,j)})^{-1}.$$

We now sketch using $\mathbb{U}_m$ to approximate the solution to a Lyapunov equation via Galerkin projection using the appropriate additions to Algorithm C.1. Once in posession of an orthonormal basis for $\mathbf{U}_j$, this entails solving

$$\left(\mathbf{U}_j^T A \mathbf{U}_j\right) Y_m + Y_m \left(\mathbf{U}_j^T A^T \mathbf{U}_j\right) + \mathbf{U}_j^T B \left(\mathbf{U}_j B\right)^T = 0,$$

for which we require the Rayleigh quotient matrix $\mathbf{T}_j = \mathbf{U}_j^T A \mathbf{U}_j$, and computing the corresponding residual norm

$$\|R_j\|_F = \left\| [\mathbf{U}_j, A\mathbf{U}_j] \begin{bmatrix} \mathbf{U}_j^T B (\mathbf{U}_j^T B)^T & Y_m \\ Y_m & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_j^T \\ \mathbf{U}_j^T A \end{bmatrix} \right\|_F.$$

One computes

$$\mathbf{U}_j^T B = \begin{bmatrix} B_W \\ e_j \otimes R_B \end{bmatrix}.$$

Noting that (C.3) is equivalent to

$$A\mathbf{U}_{(j)} = \sum_{i=1}^{j} \mathbf{U}_{(i)} \mathbf{H}^U_{(ij)} + W \mathbf{H}^W_{(j)} + \mathbf{V}_{(\ell+1)} F \mathbf{D}_{(j)} \tag{C.4}$$

which can be expressed in the matrix form

$$A\widehat{\mathbf{U}}_j = \widehat{\mathbf{U}}_{j+1} \underline{\mathbf{H}}^U_j + W \mathbf{H}^W_j + \mathbf{V}_{(\ell+1)} F \mathbf{D}_j \tag{C.5}$$

so that following the derivations in Chapter 5 (and remarking that what must now be computed and stored is $\widehat{\mathbf{U}}_j^T \mathbf{V}_{(\ell+1)}$) gives the desired quantities of interest.