A dissertation submitted to
Kent State University in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Xuebo Yu

August, 2014
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................ v

LIST OF TABLES ............................................................................................................. ix

Acknowledgements ....................................................................................................... xi

1 Preliminary ..................................................................................................................... 1

1.1 Notation .................................................................................................................... 1

1.2 SVD and GSVD ....................................................................................................... 3

1.3 Krylov subspace ..................................................................................................... 4

1.4 Arnoldi and flexible Arnoldi iteration .................................................................... 5

2 Reduction methods and application to Tikhonov regularization ................................. 8

2.1 Introduction ............................................................................................................ 8

2.1.1 Ill-posed problems .......................................................................................... 8

2.1.2 Tikhonov regularization .................................................................................. 10

2.1.3 A generalized Krylov subspace method for Tikhonov regularization .............. 12

2.2 Flexible Arnoldi reduction ..................................................................................... 14

2.2.1 A reduction method for matrix tuplets ............................................................ 19

2.2.2 One-parameter Tikhonov regularization .......................................................... 20

2.2.3 Multi-parameter Tikhonov regularization ...................................................... 23

2.2.4 Numerical examples ....................................................................................... 24
LIST OF FIGURES

1  Example 2.2.6. Approximate solution $x_{31,\mu_{31}}$ determined by FATR with $\rho = 1/5$ for the test problem \textbf{baart} with noise level $10^{-3}$ (red solid curve), approximate solution determined with GSVD (blue dashed curve), and desired solution $\hat{x}$ (black dash-dotted curve) . . . . . . . . . . . . . . . . 27

2  Example 2.2.6. (a) $\|Bx_{\ell,\mu_\ell}\|$ as a function of $\ell$ for the test problem \textbf{baart} with $x_{\ell,\mu_\ell}$ computed by FATR with $\rho = 1/5$ for $\ell = 13, 14, \ldots, 50$. (b) $\|x_{\ell,\mu_\ell} - \hat{x}\|/\|\hat{x}\|$ as a function of $\ell$. . . . . . . . . . . . . . . . . . . . . . . 28

3  Example 2.2.7. Computed approximate solutions for the test problem \textbf{deriv2} with noise-level $10^{-3}$. The red solid curve shows the approximate solution $x_{11,\mu_{11}}$ determined by FATR without augmentation and $\rho = \infty$, the blue dashed curve displays $x_{19,\mu_{19}}$ computed by FATR without augmentation and $\rho = 1/5$, and the black dash-dotted curve shows the $\hat{x}$. . . 29

4  Example 2.2.8. Computed approximate solutions for the problem \textbf{i_laplace} with noise level $10^{-3}$. The FATR method with $\rho = 1/50$ yields an accurate approximate solution $x_{92,\mu_{92}}$ (red solid curve). Carrying out more steps yields a worse approximation $x_{117,\mu_{117}}$ (blue dashed curve). The desired solution $\hat{x}$ is also shown (black dash-dotted curve). . . . . . . . . . . . . . . . . . . . . . . 31

5  Example 2.2.8: FATR with $\rho = 1/50$ applied to \textbf{i_laplace} with noise level $10^{-3}$. (a) Relative error $\|x_{\ell,\mu_\ell} - \hat{x}\|/\|\hat{x}\|$ as a function of $\ell$. (b) $\|Bx_{\ell,\mu_\ell}\|$ as function of $\ell$ for $\ell = 50, 51, \ldots, 117$. . . . . . . . . . . . . . . . . . . . . . . 32
Example 2.2.9. (Image restoration) (a) original image. (b) blur- and noise-contaminated image. .............................................. 35

Example 2.2.9. (Image restoration) Image restored by the FATR method with the regularization matrix $B$ a discretization of the Laplacian 50.

(a) Restored image $x_{2,\mu_2}$ obtained with $\rho = \infty$, (b) restored image $x_{2,\mu_2}$ determined with $\rho = 1$, and (c) restored image $x_{3,\mu_3}$ obtained with $\rho = 1/2$. 36

Example 2.2.9. (Image restoration) (a) Restored image $x_{4,\mu_4}$ by FATR with Perona–Malik operator 51 with $\rho = \infty$, (b) edges map for the restoration in (a). ................................................................. 37

Example 2.3.9. Computed approximate solutions for the test problem deriv2 with noise-level $10^{-3}$. (a) $B$ is the bidiagonal matrix $L_1$ in (1). (b) $B$ is the tridiagonal matrix $L_2$ in (2). The curves determined with the GSVD are on top of the curves obtained with GGKTR($\rho = 1$). ................. 54

Example 2.3.9. Relative error in computed approximate solutions $x_{\ell,\mu_\ell}$ as a function of $\ell \geq 27$ for the test problem deriv2 with noise-level $10^{-3}$ and the tridiagonal regularization matrix (2). ................................. 55

Example 2.3.10. Computed approximate solutions for the modified phillips test problem with noise-level $10^{-2}$. (a) Tikhonov solution $x_{95,\mu_95}$ with $B$ defined by (2). The curve determined with GSVD is on top of the curve obtained with GGKTR($\rho = 1$). (b) RRGMRES solution $x_8$. .......... 56
Example 2.3.10. Computed approximate solutions for the modified phillips test problem with noise-level $10^{-2}$ and the regularization matrix (2). The figure differs from Figure 11(a) in that it shows the GGKTR solution $\mathbf{x}_{27,\mu_{27}}$ determined after only $\ell = 27$ steps with GGKTR($\rho = 1$). 

Blur- and noise-free images for Examples 2.3.11 and 2.3.12.

Example 2.3.11. (a) Blurred and noisy image. (b) Image restored with $B = I$. (c) Image restored with $B$ being the discretized Laplacian. (d) Image restored with $B$ being the discretized Perona–Malik operator.

Example 2.3.12. (a) Blurred and noisy image. (b) Image restored with $B$ being the discretized Laplacian. (c) Image restored with $B$ being the discretized Perona–Malik operator. (d) Edge map for the restoration in (c).

Example 2.3.13. The red solid curve displays the computed approximate solution $\mathbf{x}_{13,\mu_{13}}$ determined by GGKTR with $\rho = 1/2$, the blue dashed curve shows the solution computed via GSVD, and the black dash-dotted curve depicts the desired solution $\hat{\mathbf{x}}$.

Example 2.4.1. Approximate solution $\mathbf{x}_{26,\mu_{26}}$ determined by Algorithm 2.4.1 with $\rho = 1/2$ with noise level $10^{-3}$ (red solid curve), approximate solution computed with GSVD (blue dashed curve), and desired solution $\hat{\mathbf{x}}$ (black dash-dotted curve).

Example 2.4.2. Exact image.

Example 2.4.2. Blur- and noise-contaminated image.

Example 2.4.2. Restored image $\mathbf{x}_{22,\mu_{22}}$ obtained by Algorithm 2.4.1 with $\rho = 0.1$ and regularization matrix (70).
Example 3.5.1. $\varepsilon$-pseudospectra determined from $A$ and by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$. 90

Example 3.5.2. $\varepsilon$-pseudospectra determined from $A$ and by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$. 93

Example 3.5.3. $\varepsilon$-pseudospectra determined by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$. . . . . 94

Example 3.5.4. $\varepsilon$-pseudospectra determined by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$. . . . . 96
LIST OF TABLES

1. Example 2.2.6. Relative errors and \( \| Bx_{\ell,\mu} \| \) in computed approximate solutions for noise-level \( 10^{-3} \). The parameter \( \ell \) denotes the number of steps with FATR. .................................................. 26

2. Example 2.2.7. Relative errors and \( \| Bx_{\ell,\mu} \| \) in computed approximate solutions for the test problem deriv2 with noise level \( 10^{-3} \). .................. 30

3. Example 2.2.8. Relative errors and \( \| Bx_{\ell,\mu} \| \) in computed solutions for noise level \( 10^{-3} \)................................................................. 32

4. Example 2.2.6. PSNR-values of restorations determined by FATR with the Laplacian regularization operator. ........................................... 35

5. Example 2.3.9. Relative errors in computed solutions for noise-level \( 10^{-3} \). The parameter \( \ell \) denotes the number of steps with GGKTR(\( \rho = 1 \)) and RRGMRES. ............................................................. 54

6. Example 2.3.10. Relative error in computed solutions for noise-level \( 10^{-2} \). The parameter \( \ell \) denotes the number of steps with GGKTR(\( \rho = 1 \)) and RRGMRES. ............................................................. 56

7. Example 2.3.10. Relative error in computed solutions for noise-level \( 10^{-3} \). The parameter \( \ell \) denotes the number of steps with GGKTR(\( \rho = 1 \)) and RRGMRES. ............................................................. 57

8. Example 2.3.13. Relative errors in computed approximate solutions for the noise level \( 10^{-1} \) ................................................................. 63
Example 2.4.1. Relative errors in computed approximate solutions for the noise level $10^{-3}$. The parameter $\ell$ denotes the number of steps with the Algorithms 2.4.1 and with FATR. ........................................ 67

Example 2.4.2. PSNR-values of restorations computed by Algorithm 2.4.1 with $B$ defined by (70). ......................................................... 69

Example 3.5.1. Comparison of CPU times in seconds. .......................... 91

Example 3.5.2. Comparison of CPU times in seconds. .......................... 92

Example 3.5.3. Comparison of CPU times in seconds. .......................... 95

Example 3.5.4. Comparison of CPU times in seconds. .......................... 95
Acknowledgements

I would like to express my special appreciation and thanks to my advisor Professor Dr. Lothar Reichel, you have been a tremendous mentor for me. I would like to thank you for encouraging my research.

A special thanks to my family. Words can not express how grateful I am to my wife Ran, my daughter Rana and my son Aiden. My completion of this could not have been accomplished without your voice and smile.
CHAPTER 1

Preliminary

1.1 Notation

\( I_n \) : \( n \times n \) identity matrix.

\( \| \cdot \| \) : Euclidean vector norm.

\( \langle \cdot, \cdot \rangle \) : The standard inner product of vectors in \( \mathbb{C}^n \).

\( M^* \) : The conjugate transpose of \( M \).

\( M^\dagger \) : Pseudoinverse (or Moore-Penrose generalized inverse) of matrix \( M \).

\[ A^\dagger = \sum_{i=1}^{\text{rank}(A)} v_i \sigma_i^{-1} u_i^T, \text{ where } v_i, \sigma_i \text{ and } u_i \text{ are defined in (3)}. \]

\( M_A^\dagger \) : \( A \)-weighted generalized inverse. See (20).

\( M_i \) : The submatrix formed by first \( i \) columns of matrix \( M \).

\( M_{i,j} \) : The leading principal \( i \times j \) submatrix of \( M \).

\( m_{i,j} \) : The entry in the \( i \)-th row and \( j \)-th column of matrix \( M \).

\( m_i \) : The \( i \)-th column of matrix \( M \).

\( b_i \) : The \( i \)-th element of vector \( b \).

\( x_{\text{GSVD}} \) : The solution of of (14) via GSVD in full space.

\( \mathcal{R}(M) \) : The range of matrix \( M \).

\( \mathcal{N}(M) \) : The null space of matrix \( M \).

\( e \) : Error vector.

\( e_i \) : The \( i \)-th column of identity matrix \( I \).
**SVD**: Singular value decomposition. See (3).

**GSVD**: Generalized singular value decomposition. See (4).

**MGS**: Modified Gram-Schmidt orthogonalization.

**PSNR**: Peak signal-to-noise ratio. It is commonly used to measure the quality of a restored image. See (49).

**$L_1$**: Bidiagonal regularization matrix, the scaled finite difference approximations of the first derivative operator with first row removed,

$$L_1 = \begin{bmatrix}
1 & -1 & \\
 & 1 & -1 \\
 & & \ddots & \ddots \\
 & & & 1 & -1
\end{bmatrix} \in \mathbb{R}^{(n-1) \times n}$$

(1)

**$L_2$**: Tridiagonal regularization matrix, the scaled finite difference approximations of the second derivative operator with first and last row removed,

$$L_2 = \begin{bmatrix}
-1 & 2 & -1 & \\
 & -1 & 2 & -1 \\
 & & \ddots & \ddots \\
 & & & -1 & 2 & -1
\end{bmatrix} \in \mathbb{R}^{(n-2) \times n}.$$
1.2 SVD and GSVD

**Definition 1.2.1.** Let $A \in \mathbb{R}^{m \times n}$ and assume $m \geq n$. The singular value decomposition (SVD) of $A$ is a decomposition of the form

$$A = U \Sigma V^* = \sum_{i=1}^{n} u_i \sigma_i v_i^*,$$

where $U = [u_1, u_2, \ldots, u_n] \in \mathbb{R}^{m \times n}$ and $V = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{n \times n}$ are matrices with orthonormal columns, $U^* U = V^* V = I$, and where the diagonal matrix $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ has nonnegative diagonal elements in non-increasing order such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

The numbers $\sigma_i$ are called singular values of $A$ while the vectors $u_i$ and $v_i$ are the left and right singular vectors of $A$. The matrix $I$ denotes the identity.

**Definition 1.2.2.** Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$ satisfy $m \geq n \geq p$ and $\mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}$, where $\mathcal{N}(M)$ denotes the null space of the matrix $M$. The generalized singular value decomposition (GSVD) is a decomposition of $A$ and $B$ of the form

$$A = U \begin{pmatrix} \Sigma & 0 \\ 0 & I_{n-p} \end{pmatrix} W^{-1}, \quad B = V \begin{pmatrix} M & 0 \end{pmatrix} W^{-1}.$$  

The matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{p \times p}$ have orthonormal columns, $U^* U = I_m$ and $V^* V = I_p$; $W \in \mathbb{R}^{n \times n}$ is nonsingular; $\Sigma$ and $M$ are $p \times p$ diagonal matrices: $\Sigma = \text{diag}(\rho_1, \ldots, \rho_p)$, $M = \text{diag}(\tau_1, \ldots, \tau_p)$. Moreover, the diagonal entries of $\Sigma$ and $M$ are nonnegative and ordered such that

$$\rho_1 \geq \cdots \geq \rho_p \geq 0, \quad 0 \leq \tau_1 \leq \cdots \leq \tau_p,$$

and they are normalized such that

$$\rho_i^2 + \tau_i^2 = 1, \quad i = 1, \ldots, p.$$  

The generalized singular values $\sigma_i$ of $\{A, B\}$ are defined as the ratios

$$\sigma_i = \rho_i / \tau_i,$$

in non-increasing order.

1.3 Krylov subspace

In linear algebra, the order-$r$ Krylov subspace generated by a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$ is the linear subspace spanned by the images of $b$ under the first $r$ powers of $A$ (starting from $A^0 = I$), that is,

$$K_r(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{r-1}b\}.$$

To construct the Krylov subspace, starting with a vector $b$, one computes $Ab$ then multiplies that vector by $A$ to get $A^2b$ and so on. Methods that use Krylov subspaces are referred to as Krylov subspace methods. We refer a method as a generalized Krylov subspace method if it uses a subspace generated by more than one matrix.

Let $K_r = [b, Ab, A^2b, \ldots, A^{r-1}b]$. Then $K_r(A, b)$ is the range of $K_r$. The column vectors of $K_r$ are not orthogonal and usually become almost linearly dependent already for fairly small values of $r$ and many matrices $A$ due to the properties of power iteration. One will want to determine an orthonormal basis for $K_r(A, b)$, because a matrix with orthonormal columns is easy to deal with. The most common orthogonalization schemes are Arnoldi iteration for general matrices and Lanczos iteration for Hermitian matrices. We will discuss Arnoldi iteration in the next section. The algorithms in this work are Arnoldi-type methods.
1.4 Arnoldi and flexible Arnoldi iteration

The Arnoldi iteration is defined as follows. It is described, e.g., in [1]. The norm $\| \cdot \|$ denotes the Euclidean vector norm or the induced matrix norm throughout this thesis.

**Algorithm 1.4.1. Arnoldi iteration**

1. Start with an arbitrary vector $b$,
2. $v_1 = b/\|b\|$, $h_{1,1} = \|b\|
3. for $k = 1, 2, \ldots, \ell$
4. \hspace{1em} $w = Av_k$
5. \hspace{1em} for $i = 1, 2, \ldots, k$
6. \hspace{2em} $h_{i,k} = v_i^* w$
7. \hspace{2em} $w = w - h_{i,k} v_i$
8. \hspace{1em} end
9. $h_{k+1,k} = \|w\|$ Assume $h_{k+1,k} \neq 0$
10. $v_{k+1} = w / h_{k+1,k}$
11. end

Application of $\ell < n$ steps of Algorithm 1.4.1 to the matrix $A$ with given initial vector $b$ yields the following decomposition:

$$AV_\ell = V_{\ell+1} H_{\ell+1,\ell},$$

where $H_{\ell+1,\ell} = [h_{i,j}]$ is an upper Hessenberg matrix and the matrix $V_{\ell+1} = [v_1, v_2, \ldots, v_{\ell+1}]$ has orthonormal columns. The columns $\{v_1, v_2, \ldots, v_{\ell+1}\}$ form an orthonormal basis of $K_\ell(A, b)$.

The Arnoldi iteration breaks down if $h_{k+1,k} = 0$ in line 10 and yields the decomposition $AV_k = V_k H_{k,k}$ instead of (8). In this case, the $\mathcal{R}(V_k)$ is the smallest invariant subspace
of $A$ that contains the vector $b$. We say the subspace $V = \text{span}\{v_1, v_2, \ldots, v_m\}$ is an invariant subspace of $A \in \mathbb{R}^{n \times n}$ if for any vector $y \in V$ we have $Ay \in V$.

The flexible GMRES (FGMRES) iterative method was introduced by Saad [2] for the solution of large linear systems of equations with a nonsingular matrix $A$. The pseudocode for flexible Arnoldi is as following:

**Algorithm 1.4.2. Flexible Arnoldi iteration**

1. Start with an arbitrary vector $b$,
2. $u_1 = v_1 = b/\|b\|$, $h_{1,1} = \|b\|$,
3. for $k = 1, 2, \ldots, \ell$
4. $w = Av_k$
6. for $i = 1, 2, \ldots, k$
7. $h_{i,k} = u_i^T w$
8. $w = w - h_{i,k} u_i$
9. end
10. $h_{k+1,k} = \|w\|$ Assume $h_{k+1,k} \neq 0$
11. $u_{k+1} = w/h_{k+1,k}$
12. determine $v_{k+1}$ for next iteration.
13. end

Application of $\ell < n$ steps of Algorithm 1.4.2 to matrix $A$ gives the following decomposition:

$$AV_\ell = U_{\ell+1}H_{\ell+1,\ell},$$

where $V_\ell = [v_1, v_2, \ldots, v_\ell]$ and $U_{\ell+1} = [u_1, u_2, \ldots, u_{\ell+1}]$ has orthonormal columns and $H_{\ell+1,\ell} = [h_{i,j}]$ is an upper Hessenberg matrix. The $\mathcal{R}(V_\ell)$ is the solution subspace for FGMRES. Saad’s interest in FGMRES stems from the fact that the method allows the
application of a sequence of different preconditioners while building up the solution subspace. An application of FGMRES to the solution of large linear discrete ill-posed problems has recently been described by Morikuni et al. [3].
CHAPTER 2

Reduction methods and application to Tikhonov regularization

The Tikhonov regularization method is a popular method to solve linear discrete ill-posed problems. The regularized problems can be solved with the aid of the generalized singular value decomposition (GSVD) when the problem is of small to medium size. This decomposition is not practical to use when the problem is of large size since the computation of the GSVD then is too expensive. The idea is to construct a solution subspace of small size with the aid of a generalized Krylov subspace method and find a solution in the solution subspace as an approximation to the solution in the full space. We refer to this as a reduction method. Several reduction methods for solving large Tikhonov regularization problems have been developed and are discussed in [4–6]. In this chapter we will add three novel reduction methods to this family. Our methods can give approximate solutions of higher accuracy than the GSVD and, therefore are attractive alternatives to the GSVD also when the matrices are small enough for the latter to be computed.

2.1 Introduction

2.1.1 Ill-posed problems

Consider linear systems of equations

\[ Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad x, b \in \mathbb{R}^n, \]  

(10)

The right-hand side \( b \) of linear discrete ill-posed problems that arise in applications
in science and engineering represents available data and typically is contaminated by a measurement error \( e \in \mathbb{R}^n \), which we will refer to as “noise.” Let \( \hat{b} \) denote the (unknown) noise-free vector associated with \( b \), i.e.,

\[
\mathbf{b} = \hat{\mathbf{b}} + e,
\]

and assume that the unavailable linear system of equations with the noise-free right-hand side,

\[
A\mathbf{x} = \hat{\mathbf{b}},
\]

is consistent. Denote the solution of (12) of minimal Euclidean norm by \( \hat{x} \). We would like to determine an approximation of \( \hat{x} \) by computing an approximate solution of (10).

Straightforward solution of (10) generally does not yield a meaningful approximation of \( \hat{x} \) due to the ill-conditioning of \( A \) and the error \( e \) in \( b \). To see the difficulty, consider the mapping \( A\mathbf{x} \) of an arbitrary vector \( \mathbf{x} \). Using the SVD, we get

\[
\mathbf{x} = \sum_{i=1}^{n}(v_i^*\mathbf{x})v_i \quad \text{and} \quad A\mathbf{x} = \sum_{i=1}^{n}\sigma_i(v_i^*\mathbf{x})u_i.
\]

This shows that for the direct problem (compute \( A\mathbf{x} \) with \( \mathbf{x} \) is known), the components of \( \mathbf{x} \), \( v_i^*\mathbf{x} \), corresponding to small singular values are more damped in \( A\mathbf{x} \) than the components corresponding to large singular values. For the inverse problem, computing \( \mathbf{x} \) from (10) or \( \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\| \), we have the opposite effect: the components of the right-hand side \( \mathbf{b} \) corresponding to small singular values are amplified. The least squares solution of minimal norm \( \mathbf{x}_{LS} \) of the least squares problem \( \min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\| \) is given by

\[
\mathbf{x}_{LS} = A^\dagger\mathbf{b} = \sum_{i=1}^{\text{rank}(A)} \frac{u_i^*\mathbf{b}}{\sigma_i}v_i.
\]
We see that the error $e$ in $b$ is divided by the small singular values in (13). Therefore, the solution $x_{LS}$ has large norm and is not an acceptable approximation of $\hat{x}$.

2.1.2 Tikhonov regularization

To find a useful and stable solution to the discrete ill-posed problems (10) we incorporate further information of the desired solution. This is known as regularization. The Tikhonov regularization method may be the most popular regularization method. It was developed independently by Phillips and Tikhonov. The method is designed to be able to incorporate a priori assumptions of the size and smoothness of the desired solution.

It has the *general form*

$$\min_{x \in \mathbb{R}^n} \{ \| Ax - b \|^2 + \mu \| Bx \|^2 \},$$

(14)

where matrix $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times n}$ with $m \geq n$ and $p \geq 1$ arbitrary. The *regularization parameter* $\mu > 0$ controls the weight given to minimization of the regularization term, relative to the minimization of the residual norm. The *regularization operator* $B$ determines how the size of the computed solution is measured. We say that Tikhonov regularization is in *standard form* if $B = I$ in (14).

We assume that $B$ is chosen so that

$$\mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}.\tag{15}$$

Then (14) has the unique solution

$$x_\mu = (A^*A + \mu B^*B)^{-1} A^*b$$

(16)

for all $\mu > 0$. 
Substituting the GSVD (4) into (16) makes it possible to express the Tikhonov approximation $x_\mu$ in the form

$$x_\mu = \sum_{i=1}^{p} \phi^T_\mu(\sigma^2_i) \frac{u_i^* b}{\rho_i} w_i + \sum_{i=p+1}^{n} (u_i^* b) w_i,$$

where $\phi^T_\mu(\lambda) = \lambda/((\lambda+\mu)$ is known as the Tikhonov filter function. We denote the second sum of right-hand side of (17) as

$$x_* = \sum_{i=p+1}^{n} (u_i^* b) w_i.$$  \hspace{1cm} (18)

We observe that $x_*$ is the component of the solution in the null space of $B$ and is not affected by regularization. The regularization occurs via the first term in (17). If we choose $\mu$ such that $\mu \ll \sigma^2_k$, then the filter factor $\phi^T_\mu(\sigma^2_k)$ is close to 1 for $i \leq k$, i.e., there is little regularization of the first $k$ terms. Based on the amount of noise, we may vary $\mu$ to control the amount of regularization in approximate solutions.

It is important to determine a suitable regularization parameter $\mu$. It can be determined via the discrepancy principle, the L-curve criterion, or the generalized cross validation method. In this work, the regularization parameter is determined by the discrepancy principle, i.e., one should choose the regularization parameter $\mu$ such that the residual norm, $\|b - Ax_\mu\|$, equals the priori given norm of the error $e$:

$$\phi(\mu) \equiv \|b - Ax_\mu\|^2 = \|e\|^2.$$ \hspace{1cm} (19)

This nonlinear equation for $\mu$ can be solved by, e.g., Newton’s method. This requires that (14) be solved for several $\mu$ values. Also see Section 2.2.2. More discussions of Tikhonov regularization, GSVD and the discrepancy principle can be found in [7,8].
2.1.3 A generalized Krylov subspace method for Tikhonov regularization

When \( A \) and \( B \) are small to moderately sized matrices, the computation effort required for solving (14) for several \( \mu \)-values can be reduced by first computing the GSVD of the matrix pair \( \{A, B\} \); see, e.g., [7] for discussions of this approach. However, the evaluation of the GSVD of a pair of large matrices is expensive. Hence, we rarely use the GSVD for the practical solution of large-scale regularization problems.

Another approach that for certain regularization matrices \( B \) may reduce the cost of repeated solution of (14) is to first transform the problem to standard form. This is achieved by substituting \( y = Bx \) into (14). The matrix \( A \) then is replaced by \( AB_A^\dagger \) in (14), where

\[
B_A^\dagger = (I - (A(I - B^\dagger B))^\dagger A)^\dagger \in \mathbb{R}^{n \times p}
\]  

is known as the \( A \)-weighted generalized inverse of \( B \). Here \( M^\dagger \) denotes the Moore–Penrose pseudoinverse of the matrix \( M \), and \( I \) is the identity matrix; see e.g., Eldén [9] or Hansen [7, Section 2.3] for details. This transformation is attractive to use when \( B \) is a banded matrix with small bandwidth or an orthogonal projection; see [10] for the latter. For general large matrices \( B \), the evaluation of matrix-vector products with the matrices \( AB_A^\dagger \) and \( B_A^\dagger \) is prohibitively expensive. Kilmer et al. [11] describe an inner-outer iterative method for the solution of (14) with a general matrix \( B \in \mathbb{R}^{p \times n} \), where \( p \) is allowed to be different from \( n \). The inner iterations implement a scheme by Zha [12] for determining approximations of generalized singular vectors associated with the largest generalized singular values of the matrix pair \( \{A, B\} \). This method produces nice results but can be expensive when many inner iterations are required.
We therefore are interested in solving large-scale Tikhonov minimization problems (14) via a generalized Krylov subspace method, which provide a reduction decomposition for the matrix pair \{A, B\}:

\[
AV = U H^{(A)} \tag{21}
\]

\[
BV = W H^{(B)}, \tag{22}
\]

where the matrices \(U\) and \(W\) have orthonormal columns, and the matrices \(H^{(A)}\) and \(H^{(B)}\) are of small size compared with \(A\) and \(B\). The numbers of columns of \(U\) and \(W\), and the properties of \(H^{(A)}\) and \(H^{(B)}\) depend on the reduction methods. The first column of the matrix \(U\) is defined by

\[
u_1 = b / \|b\|. \tag{23}\]

The \(R(V)\) is the solution subspace, i.e., \(x\) is a linear combination of columns of \(V\). It has following form:

\[
x = V y. \tag{24}\]

Substituting the decompositions (21), (22) and (24) into (14), and using (23), yields the reduced Tikhonov minimization problem

\[
\min_y \left\{ \|H^{(A)}y - e_1\|b\| \|^2 + \mu \|H^{(B)}y\|^2 \right\}, \tag{25}\]

where \(e_1\) denotes the first column of identity matrix. The solution \(y_\mu\) of (25) gives the approximate solution

\[
x_\mu = V y_\mu \tag{26}\]

of (14) and (10). We will discuss the solution of the reduced problem (25) in Section 2.2.2.
The reduction method in Section 2.2 is inspired by the generalized Arnoldi process described by Li and Ye [6] for the reduction of a pair of square matrices. Their approach yields the decompositions

\[ AV_\ell = V_{2\ell} H_{2\ell,\ell}^{(A)}, \quad BV_\ell = V_{2\ell+1} H_{2\ell+1,\ell}^{(B)}, \]  

(27)

where the matrix \( V_{2\ell+1} \in \mathbb{R}^{n \times (2\ell+1)} \) has orthonormal columns, the first of which is prescribed, and the matrices \( V_\ell \) and \( V_{2\ell} \) are made up of the \( \ell \) and \( 2\ell \) first columns of \( V_{2\ell+1} \), respectively. Moreover, \( H_{2\ell,\ell}^{(A)} \in \mathbb{R}^{2\ell \times \ell} \) and \( H_{2\ell+1,\ell}^{(B)} \in \mathbb{R}^{(2\ell+1) \times \ell} \) are rectangular matrices with a Hessenberg-type structure. Application of the decompositions (27) to Tikhonov regularization is described in [5]. We are interested in using the decompositions (21) and (22) because they allow larger flexibility in the choice of the matrix \( V_\ell \), and the reduced matrices \( \{ H^{(A)}, H^{(B)} \} \) obtained with the method of the present in Section 2.2 are smaller for fixed \( \ell \) than the reduced matrices \( \{ H_{2\ell,\ell}^{(A)}, H_{2\ell+1,\ell}^{(B)} \} \) in (27). Both the reduction methods by Li and Ye [6] and the one in Section 2.2 require the matrices \( A \) and \( B \) to be square. Both square (see [13–16]) and rectangular (see [7, 11]) regularization matrix are commonly used in the literature. The reduction method presented in Section 2.3 allows both \( A \) and \( B \) to be rectangular; Section 2.4 is for the situation when \( A \) is square and \( B \) is rectangular.

2.2 Flexible Arnoldi reduction

This section describes an algorithm for constructing the matrix \( V_\ell \) in (21) and (22) via generalized Krylov subspaces and their application as solution subspaces. These spaces contain information about both the matrices \( A \) and \( B \), which is gathered by evaluating
matrix-vector products. In iteration $j$ with Algorithm 2.2.1 below, one computes matrix-vector products $A\mathbf{v}_j$ and $B\mathbf{v}_j$, where $\mathbf{v}_j$ is the $j$th column of $V_\ell$. The vector $A\mathbf{v}_j$ is orthogonalized against the columns of $U_j = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_j]$ and normalized to yield the last column $\mathbf{u}_{j+1}$ of the matrix $U_{j+1}$ and, similarly, $B\mathbf{v}_j$ is orthogonalized against the columns of $W_{j-1} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{j-1}]$ and normalized to yield the last column $\mathbf{w}_j$ of the matrix $W_j$. The solution subspace is then expanded by the last column of $U_{j+1}$ or $W_j$ that has not already been used to expand the solution subspace. The parameter $\rho \geq 0$ in the algorithm specifies the ratio of number of columns from $U_{\ell+1}$ and $W_\ell$ that are used to expand the solution subspace. Thus, the value $\rho = 1$ indicates that every other vector used to expand the solution subspace is a column of the matrix $U_{\ell+1}$. When $\rho = 5$ the solution subspace is expanded by five columns from $U_{\ell+1}$ for every column from $W_\ell$. We have found that for certain Tikhonov minimization problems (14), it is beneficial to choose $\rho \neq 1$. The counters $N(\mathbf{u})$ and $N(\mathbf{w})$ in Algorithm 2.2.1 keep track of how many columns from the matrices $U_{\ell+1}$ and $W_\ell$, respectively, have been included in the solution subspace at step $j$ of the algorithm. We discuss properties of Algorithm 2.2.1 below.
Algorithm 2.2.1. Flexible Arnoldi reduction of a matrix pair \( \{A, B\} \).

1. Input: matrices \( A, B \in \mathbb{R}^{n \times n} \), vector \( b \in \mathbb{R}^n \), ratio \( \rho \geq 0 \), and number of steps \( \ell \)

2. \( h_{1,1} := \|b\|; \quad u_1 := b/h_{1,1}; \quad v_1 := u_1 \);

3. \( N(u) := 1; \quad N(w) := 1 \)

4. for \( j = 1, 2, \ldots, \ell \) do

5. \( \tilde{u} := Av_j \) new \( u \)-vector

6. for \( i = 1, 2, \ldots, j \) do

7. \( h_{i,j}^{(A)} := u_i^* \tilde{u}; \quad \tilde{u} := \tilde{u} - u_i h_{i,j}^{(A)} \)

8. end for

9. \( h_{j+1,j}^{(A)} := \|\tilde{u}\| \)

10. \( u_{j+1} := \tilde{u}/h_{j+1,j}^{(A)} \) if \( h_{j+1,j}^{(A)} = 0 \): see text

11. \( \tilde{w} := Bv_j \) new \( w \)-vector

12. for \( i = 1, 2, \ldots, j - 1 \) do

13. \( h_{i,j}^{(B)} := w_i^* \tilde{w}; \quad \tilde{w} := \tilde{w} - w_i h_{i,j}^{(B)} \)

14. end for

15. \( h_{j,j}^{(B)} := \|\tilde{w}\| \)

16. \( w_j := \tilde{w}/h_{j,j}^{(B)} \) if \( h_{j,j}^{(B)} = 0 \): see text

17. if \( N(w)/N(u) > 1/\rho \)

18. \( N(u) := N(u) + 1; \quad v := u_{N(u)} \)

19. else

20. \( v := w_{N(u)}; \quad N(w) := N(w) + 1 \)

21. end

22. for \( i = 1, 2, \ldots, j \) do

23. \( v := v - (v_i^* v)v_i; \)

24. end for

25. \( \alpha_j := \|v\| \); new \( v \)-vector

26. \( v_{j+1} := v/\alpha_j \)

27. end for
Algorithm 2.2.1 determines the orthonormal columns $u_j$, $v_j$, and $w_j$ of the matrices $U_{\ell+1}$, $V_\ell$, and $W_\ell$, respectively, in the decompositions (28) and (29). In addition, the algorithm computes the last column of $V_{\ell+1}$. The elements $h_{i,j}^{(A)}$ and $h_{i,j}^{(B)}$ determined by Algorithm 2.2.1 are the nontrivial entries of the upper Hessenberg matrix $H_{\ell+1,\ell}^{(A)}$ in (28) and of the upper triangular matrix $H_{\ell,\ell}^{(B)}$ in (29), respectively. Lines 5-10 of the algorithm describe one step of the flexible Arnoldi process introduced by Saad [2]. This process is said to break down at step $j$ if $h_{j+1,j}^{(A)}$ vanishes in line 9. Breakdown may indicate that the solution subspace contains the solution of (10); see Proposition 2.2.2 below.

The computations can be continued after breakdown by letting $u_{j+1}$ in line 10 be an arbitrary unit vector that is orthogonal to the already generated vectors $u_1, u_2, \ldots, u_j$.

Lines 12-16 update the QR factorization of the matrix $BV_j$ to yield a QR factorization of $BV_{j-1}$, where $V_k = [v_1, v_2, \ldots, v_k]$ for $k \in \{j, j+1\}$. Daniel et al. [17] provide detailed discussions on updating and downdating of QR factorizations. The updating process is said to breakdown if $h_{j,j}^{(B)}$ vanishes in line 15. The computations can be continued after breakdown by letting $w_j$ on line 16 be an arbitrary unit vector that is orthogonal to the already generated vectors $w_1, w_2, \ldots, w_{j-1}$. We remark that the occurrence of breakdown is rare unless the vectors $v_j$ are chosen to be in $\mathcal{N}(B)$.

Lines 22-26 orthogonalize the new vector for the solution subspace against the already available orthonormal solution subspace basis. Breakdown occurs when $\alpha_j$ vanishes in line 25. In this situation, the vector $v_{j+1}$ in line 26 can be chosen to be a unit vector that is orthogonal to the already available orthonormal vectors $v_1, v_2, \ldots, v_j$.

**Proposition 2.2.1.** Let $A, B \in \mathbb{R}^{n \times n}$. Then application of $\ell$ steps of Algorithm 2.2.1 to the matrix pair $\{A, B\}$ with breakdowns handled as described above gives a flexible
Arnoldi decomposition of $A$ and a QR factorization of $BV\ell$ for any $1 \leq \ell \leq n$. It has following form:

$$AV\ell = U_{\ell+1}H^{(A)}_{\ell+1,\ell},$$

$$BV\ell = W_{\ell}H^{(B)}_{\ell,\ell},$$

where the matrices $V_{\ell}, U_{\ell+1}$ and $W_{\ell}$ have orthonormal columns, $H^{(A)}_{\ell+1,\ell}$ is an upper Hessenberg matrix and $H^{(B)}_{\ell,\ell}$ is an upper triangular matrix.

When the matrix $A$ is nonsingular, breakdown in line 9 of Algorithm 2.2.1 may signal that an approximation of $\hat{x}$ can be computed from the flexible Arnoldi decomposition, only, without Tikhonov regularization.

**Proposition 2.2.2.** Let the matrix $A$ be nonsingular and assume that breakdown in line 9 of Algorithm 2.2.1 occurs for the first time at iteration $k \leq n$. Let $H^{(A)}_{k}$ denote the leading $k \times k$ submatrix of the upper Hessenberg matrix $H^{(A)}_{k+1,k} \in \mathbb{R}^{(k+1)\times k}$ available at breakdown. Then the solution of (10) is given by

$$x_{k} = V_{k}y_{k}, \quad y_{k} = (H^{(A)}_{k})^{-1}e_{1}\|b\|.$$  

(30)

**Proof.** The results is an application of [3, Proposition 2.1]. The nonsingularity of $H_{k}$ follows from the facts that $A$ is nonsingular and the matrix $V_{k}$ is of rank $k$.

Whether the solution $x_{k}$ of Proposition 2.2.2 is a useful approximation of the desired solution $\hat{x}$ of (12) depends on the amount of noise $e$ in the right-hand side $b$ in (10) and on how ill-conditioned the matrix $H^{(A)}_{k}$ is. Since the matrix $H^{(A)}_{k+1,k} \in \mathbb{R}^{(j+1)\times (j+1)}$ generated after $j-1$ steps of Algorithm 2.2.1 is a submatrix of $H^{(A)}_{j}$ for $j = 2, 3, \ldots, k$, the condition number of the matrices $H^{(A)}_{j}$ increases (generally strictly) with $j$. Therefore, unless $k$ is very small at breakdown or there is very little noise in $b$, Tikhonov regularization may yield a better approximation of $\hat{x}$ than (30).
The following results illustrate the role of the parameter $\rho$ in Algorithm 2.2.1. The proposition can be shown by direct computation.

**Proposition 2.2.3.** Assume that no breakdown occurs in Algorithm 2.2.1. Let $\rho = \infty$. Then the solution subspace is given by

$$\mathcal{R}(V_\ell) = K_\ell(A, b).$$

If, instead, $\rho = 0$, then we have

$$\mathcal{R}(V_\ell) = K_\ell(B, b).$$

When $0 < \rho < \infty$, the solution subspace is a generalized Krylov subspace when its dimension $\ell$ is sufficiently large. In particular,

$$\rho = 1 : \quad \mathcal{R}(V_\ell) = \text{span}\{b, Bb, Ab, B^2b, ABB, BAb, A^2b, B^3b, \ldots \},$$

$$\rho = 5 : \quad \mathcal{R}(V_\ell) = \text{span}\{b, Ab, A^2b, A^3b, A^4b, Bb, A^5b, ABB, A^6b, \ldots \},$$

$$\rho = \frac{1}{5} : \quad \mathcal{R}(V_\ell) = \text{span}\{b, Bb, B^2b, B^3b, B^4b, B^5b, Ab, B^6b, BAb, \ldots \}.$$
It is straightforward to develop an extension of Algorithm 2.2.1 that determines the decompositions

\[ AV_\ell = U_{\ell+1} H_{\ell+1,\ell}^{(A)}, \]

\[ B^{(i)} V_\ell = W^{(i)} H_{\ell}^{(B)}, \quad i = 1, 2, \ldots, q - 1, \]

where the matrices \( U_{\ell+1} \in \mathbb{R}^{n \times (\ell+1)} \) and \( V_\ell, W^{(i)}_\ell \in \mathbb{R}^{n \times \ell} \) have orthonormal columns. The first column of \( U_{\ell+1} \) is a specified unit vector such as \( u_1 = b/\|b\| \). The columns of \( V_\ell \) can be determined in a similar fashion as in Algorithm 2.2.1 or be chosen to be arbitrary linearly independent vectors. The matrix \( H_{\ell+1,\ell}^{(A)} \in \mathbb{R}^{(\ell+1)\times \ell} \) is of upper Hessenberg form and the matrices \( H_{\ell}^{(B)} \in \mathbb{R}^{\ell \times \ell} \) are upper triangular. The relation (31) is a flexible Arnoldi decomposition, and the relations (32) are QR factorizations. Analogues of Propositions 2.2.1-2.2.2 can be shown.

### 2.2.2 One-parameter Tikhonov regularization

Substituting the decompositions (28) and (29) into (14), and letting \( x = V_\ell y \), yields the small minimization problem

\[
\min_{y \in \mathbb{R}^\ell} \left\{ \| H_{\ell+1,\ell}^{(A)} y - e_1 \| b \|^2 + \mu \| H_{\ell,\ell}^{(B)} y \|^2 \right\}.
\]

This problem can be solved in a variety of ways for several values of \( \mu > 0 \). One approach is to determine the solution \( y_{\ell,\mu} \) of (33) with the aid of the GSVD of the matrix pair \( \{ H_{\ell+1,\ell}^{(A)}, H_{\ell,\ell}^{(B)} \} \). An approximate solution of (14) is then furnished by

\[
x_{\ell,\mu} = V_\ell y_{\ell,\mu}
\]

It may be faster to compute the solution \( y_{\ell,\mu} \) of (33) by using the decomposition described in [18]. Finally, when the upper triangular matrix \( H_{\ell,\ell}^{(B)} \) in (33) is not very ill-conditioned,
i.e., when its condition number \( \kappa(H_{\ell,\ell}^{(B)}) = \|H_{\ell,\ell}^{(B)}\|\|(H_{\ell,\ell}^{(B)})^{-1}\| \) is not very large, we can transform the Tikhonov minimization problem (33) to standard form

\[
\min_{z \in \mathbb{R}^r} \left\{ \|\tilde{H}_{\ell+1,\ell} z - \|b\|e_1\|^2 + \mu \|z\|^2 \right\}, \quad z = H_{\ell,\ell}^{(B)} y, \tag{35}
\]

where

\[
\tilde{H}_{\ell+1,\ell} = H_{\ell+1,\ell}^{(A)} (H_{\ell,\ell}^{(B)})^{-1}. \tag{36}
\]

The minimization problem (35) can be solved in several ways, for instance, by first determining the entries of \( \tilde{H}_{\ell+1,\ell} \) by solving the system of equations \( \tilde{H}_{\ell+1,\ell} H_{\ell,\ell}^{(B)} = H_{\ell+1,\ell}^{(A)} \) column by column and then computing the singular value decomposition (SVD) of the matrix (36). Eldén [19] describes how, instead of computing the SVD, one can reduce the computational effort by only reducing \( \tilde{H}_{\ell+1,\ell} \) to bidiagonal form.

In the numerical examples of this chapter, we assume a bound \( \varepsilon \) for the norm of the error \( e \) in \( b \) to be available. This allows us to determine the regularization parameter \( \mu \) by the discrepancy principle. Thus, we seek to compute a value \( \mu > 0 \) so that the solution \( z_{\ell,\mu} \) of (35) satisfies

\[
\|\tilde{H}_{\ell+1,\ell} z_{\ell,\mu} - \|b\|e_1\| = \eta \varepsilon, \tag{37}
\]

where \( \eta \geq 1 \) is a user-specified constant independent of \( \varepsilon \). Denote this value of \( \mu \) by \( \mu_\ell \) and the associated solution of (35) by \( z_{\ell,\mu_\ell} \). The following result shows that generally it is possible to determine \( \mu_\ell \) and \( z_{\ell,\mu_\ell} \) with the desired properties.

**Proposition 2.2.4.** Let the matrix \( \tilde{H}_{\ell+1,\ell} \) be defined by (36) and assume that \( \tilde{H}_{\ell+1,\ell}^* e_1 \neq 0 \). Let \( P_{N(\tilde{H}_{\ell+1,\ell}^*)} \) denote the orthogonal projector onto \( N(\tilde{H}_{\ell+1,\ell}^*) \). Then the function

\[
\varphi(\nu) = \|\tilde{H}_{\ell+1,\ell} z_{\ell,1/\nu} - \|b\|e_1\|^2.
\]
is strictly decreasing, convex, and

\[ \varphi(0) = \|b\|^2, \quad \lim_{\nu \to \infty} \varphi(\nu) = \|P_N(\tilde{H}_{\ell+1}\nu) e_1\|^2 \|b\|^2. \]

**Proof.** The results can be shown by using the representation

\[ \varphi(\nu) = \|b\|^2 e_1^*(\nu \tilde{H}_{\ell+1} \tilde{H}_{\ell+1}^* + I)^{-2} e_1, \quad \nu \geq 0 \]

and the SVD of \( \tilde{H}_{\ell+1} \). An analogous result is established in [20, Theorem 2.1], where a detailed proof is provided.

Typically,

\[ \|P_N(\tilde{H}_{\ell+1}^* e_1)\| \|b\| < \eta \varepsilon < \|b\|. \quad (38) \]

Then, by Proposition 2.2.4, the equation

\[ \varphi(\nu) = \eta^2 \varepsilon^2 \quad (39) \]

has a unique solution \( \nu_\ell \) and \( \mu_\ell \) defined above is given by \( \mu_\ell = 1/\nu_\ell \). The solution \( y_{\ell, \mu_\ell} \) of (33) yields the approximate solution \( x_{\ell, \mu_\ell} = V_\ell y_{\ell, \mu_\ell} \) of (14) and (10). It follows that \( x_{\ell, \mu_\ell} \) satisfies

\[ \|A x_{\ell, \mu_\ell} - b\| = \eta \varepsilon. \]

Many zero-finders can be used to solve (39). The monotonic decrease and convexity of \( \varphi(\nu) \) secure that Newton’s method with an initial iterate smaller than \( \nu_\ell \), such as zero, converges monotonically and quadratically.

Let \( x_{\ell, \mu_\ell} \) solve (14) with \( \mu = \mu_\ell \). Our computed examples indicate that it is important that \( \|B x_{\ell, \mu_\ell}\| \) be fairly small. We next show that this quantity is a decreasing function of \( \ell \).
Proposition 2.2.5. Assume that (38) holds and let \( \ell \) be large enough so that the equation

\[
\|Ax - b\| = \eta \varepsilon
\]  

(40)

has the solutions \( x_{p,\mu} \in \mathcal{R}(V_p) \) for \( p \in \{\ell, \ell + 1\} \). Then

\[
\|Bx_{\ell+1,\mu_{\ell+1}}\| \leq \|Bx_{\ell,\mu_{\ell}}\|. 
\]  

(41)

Proof. We have \( x_{p,\mu} = V_p y_{p,\mu} \) for some \( y_{p,\mu} \in \mathbb{R}^p \) for \( p \in \{\ell, \ell + 1\} \). Then

\[
\|AV_{\ell+1}y_{\ell+1,\mu_{\ell+1}} - b\|^2 + \mu_{\ell+1}\|BV_{\ell+1}y_{\ell+1,\mu_{\ell+1}}\|^2 \\
= \min_{y \in \mathbb{R}^{\ell+1}} \left\{ \|AV_{\ell+1}y - b\|^2 + \mu_{\ell+1}\|BV_{\ell+1}y\|^2 \right\} \\
\leq \min_{y \in \mathbb{R}^{\ell}} \left\{ \|AV_{\ell}y - b\|^2 + \mu_{\ell+1}\|BV_{\ell}y\|^2 \right\} \\
\leq \|AV_{\ell}y_{\ell,\mu_{\ell}} - b\|^2 + \mu_{\ell+1}\|BV_{\ell}y_{\ell,\mu_{\ell}}\|^2. 
\]  

(42)

It follows from (40) that

\[
\|AV_{\ell+1}y_{\ell+1,\mu_{\ell+1}} - b\|^2 = \|AV_{\ell}y_{\ell,\mu_{\ell}} - b\|^2 = \eta^2 \varepsilon^2, 
\]

and substituting into (42) shows (41).

We have observed that the norm \( \|Bx_{\ell,\mu_{\ell}}\| \) typically decreases faster during the initial steps of Algorithm 2.2.1 than during later steps. A possible stopping criterion for the algorithm therefore is to terminate the computations when \( \|Bx_{\ell,\mu_{\ell}}\| \) stops decreasing rapidly. We illustrate this stopping criterion in Section 2.2.4.

2.2.3 Multi-parameter Tikhonov regularization

We briefly consider multi-parameter Tikhonov regularization

\[
\min_{x \in \mathbb{R}^n} \left\{ \|Ax - b\|^2 + \sum_{i=1}^{q-1} \mu_i \|B^{(i)}x\|^2 \right\}, 
\]  

(43)
where the $B^{(i)}$ are regularization matrices and the scalars $\mu_i \geq 0$ regularization parameters. The parameter $q$ is assumed to be strictly larger than three. An insightful discussion on multi-parameter Tikhonov regularization for small to moderately sized problems (43) is provided by Brezinski et al. [21]. More recent discussions are given by Gazzola and Novati [22,23], and Lu and Pereverzyev [24].

Letting $x = V_\ell y$ and applying the decompositions (31) and (32), we proceed similarly as in the derivation of (33) and obtain that the minimization problem (43) over the space $\mathcal{R}(V_\ell)$ is equivalent to the reduced minimization problem

$$
\min_{y \in \mathbb{R}^\ell} \left\{ \| H_{\ell+1,\ell} y - \| b \| e_1 \|^2 + \sum_{i=1}^{q-1} \mu_i \| R_\ell^{(i)} y \|^2 \right\}.
$$

Methods for determining suitable regularization parameters for this minimization problem are described in [21–24].

2.2.4 Numerical examples

We present a few examples that illustrate the performance of one-parameter Tikhonov regularization based on Algorithm 2.2.1 for various values of the parameter $\rho$. We refer to this method as the flexible Arnoldi Tikhonov regularization (FATR) method. Our implementation first evaluates one matrix-vector product with the matrix $A$ before matrix-vector products with $B$ are computed, even when $\rho < 1$ in Algorithm 2.2.1.

The “noise vector” $e$ has normally distributed pseudorandom entries with mean zero in all examples. The vector is scaled to correspond to a chosen noise-level

$$
\delta = \| e \| / \| \hat{b} \|,
$$

where $\hat{b}$ denotes the noise-free right-hand side vector in (10). Thus, the parameter $\varepsilon = \delta \| \hat{b} \|$; cf. (37). The regularization parameter $\mu$ is in all example determined with the
aid of the discrepancy principle with \( \eta = 1 \) in (37). This set up is valid throughout this chapter unless otherwise specified.

The first three examples are test problems from Regularization Tools [25], while the last example shows an application to the restoration of an image that has been contaminated by blur and noise. The regularization matrix in the latter example is such that its \( A \)-weighted pseudoinverse is unattractive to use in computations.

We compare the quality of the computed approximations of the desired solution \( \hat{x} \) obtained with the FATR method of this section with that achieved with the generalized Krylov subspace method described in [5]. The latter method is referred to as generalized Arnoldi Tikhonov regularization (GATR) in the examples below. Similarly as in [5], we determine regularization parameter for GATR with the discrepancy principle and use the user-chosen parameter \( \eta = 1 \). This parameter is analogous to \( \eta \) in (37).

The first three examples are small enough to allow the computation of the GSVD of the matrix pair \( \{A, B\} \). We therefore are able to compare the quality of the computed approximations of \( \hat{x} \) achieved with FATR with those determined by GSVD. The matrix \( B \) in these examples is the tridiagonal square matrix

\[
L = \begin{bmatrix}
0 & 0 & \ldots & 0 \\
-1 & 2 & -1 \\
1 & 2 & -1 \\
\vdots & \vdots & \ddots & \ddots \\
-1 & 2 & -1 \\
0 & \ldots & 0 & 0 
\end{bmatrix} \in \mathbb{R}^{n \times n}.
\] (44)
It is a scaled finite difference approximation of the second derivative operator in one space-dimension with vanishing first and last rows. Its properties are discussed in [15,16]. The relative error is computed according to \( \| x_{\ell,\mu} - \hat{x} \| / \| \hat{x} \| \), where \( \hat{x} \) is the minimal-norm solution of (12). All computations were carried out in MATLAB with about 15 significant decimal digits.

**Example 2.2.6.** The Fredholm integral equation of the first kind,

\[
\int_{0}^{\pi/2} \kappa(\sigma, \tau)x(\sigma)d\sigma = b(\tau), \quad 0 \leq \tau \leq \pi,
\]

with \( \kappa(\sigma, \tau) = \exp(\sigma \cos(\tau)) \), \( b(\tau) = 2 \sinh(\tau)/\tau \), and solution \( x(\tau) = \sin(\tau) \) is discussed by Baart [26]. We use the MATLAB function `baart` from [25] to discretize (45) by a Galerkin method with 500 orthonormal box functions as test and trial functions. The function `baart` produces a nonsymmetric matrix \( A \in \mathbb{R}^{500 \times 500} \) and a scaled discrete approximation \( \hat{x} \in \mathbb{R}^{500} \) of \( x(\tau) \), with which we compute the error-free right-hand side \( \hat{b} = A\hat{x} \). The error vector \( e \in \mathbb{R}^{500} \) corresponds to the noise level \( \delta = 1 \cdot 10^{-3} \). The right-hand side \( b \) in the system (10) is obtained from (11).

<table>
<thead>
<tr>
<th>Method</th>
<th>( \rho )</th>
<th>( \ell )</th>
<th>( | x_{\ell,\mu} - \hat{x} | / | \hat{x} | )</th>
<th>( | Bx_{\ell,\mu} | )</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATR</td>
<td>( \infty )</td>
<td>7</td>
<td>( 2.76 \cdot 10^{-2} )</td>
<td>( 4.84 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>FATR</td>
<td>1</td>
<td>11</td>
<td>( 1.20 \cdot 10^{-2} )</td>
<td>( 4.88 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>FATR</td>
<td>( 1/5 )</td>
<td>31</td>
<td>( 8.18 \cdot 10^{-3} )</td>
<td>( 4.93 \cdot 10^{-5} )</td>
</tr>
<tr>
<td>GSVD</td>
<td></td>
<td></td>
<td>( 3.19 \cdot 10^{-2} )</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example 2.2.6. Relative errors and \( \| Bx_{\ell,\mu} \| \) in computed approximate solutions for noise-level \( 10^{-3} \). The parameter \( \ell \) denotes the number of steps with FATR.

Figure 1 displays the approximate solution \( x_{31,\mu31} \) (red solid curve) computed by the FATR method with \( \rho = 1/5 \) and with GSVD (blue dashed curve). The figure also shows the desired solution \( \hat{x} \) (black dash-dotted curve). It is clear that FATR with \( \rho = 1/5 \) delivers the better approximation of \( \hat{x} \). Table 1 shows numerical results. The choice of the ratio \( \rho \) is seen to be important for the accuracy. The value \( \rho = \infty \), which corresponds to
Figure 1: Example 2.2.6. Approximate solution $x_{31,\mu}$ determined by FATR with $\rho = 1/5$ for the test problem `baart` with noise level $10^{-3}$ (red solid curve), approximate solution determined with GSVD (blue dashed curve), and desired solution $\hat{x}$ (black dash-dotted curve).

letting the solution subspace be a standard Krylov subspace, yields worse approximations of $\hat{x}$ than smaller values of $\rho$. We recall that $\rho = 1$ and $\rho = 1/5$ give solution subspaces that are generalized Krylov subspaces; cf. Proposition 2.2.3. The number of steps $\ell$ of FATR is chosen to yield approximate solutions $x_{\ell,\mu}$ of highest accuracy for a fixed ratio $\rho$. This example shows that FATR can give better approximations of $\hat{x}$ than Tikhonov regularization based on the GSVD of $\{A, B\}$. Moreover, the error $\|x_{\ell,\mu} - \hat{x}\|$ generally does not change much with $\ell$. It is, therefore, not crucial for the good performance of FATR that the number of iterations $\ell$ that yields the smallest error be available.

Figure 2(a) shows the values of $\|Bx_{\ell,\mu}\|$ as a function of the number of steps $\ell$ for $\rho = 1/5$. The values of $\|Bx_{\ell,\mu}\|$ for $\ell < 13$ are very large and therefore not included in the graph. The norms $\|Bx_{\ell,\mu}\|$ do not change much with $\ell$ for $\ell \geq 31$. Generally, this indicates that an accurate approximation of $\hat{x}$ has been found. This is the case for the present example. Figure 2(b) shows the relative errors $\|x_{\ell,\mu} - \hat{x}\|/\|\hat{x}\|$ as a function of $\ell$. We observe a drop in relative error at $\ell = 31$ and a jump at $\ell = 43$. The jump is
stems from the fact that the solution $x_{\ell,\mu}$ has converged to the solution $x_{GSVD}$ obtained with GSVD when $\ell \geq 43$. For many linear discrete ill-posed problems, FATR determines more accurate approximations of $\hat{x}$ than $x_{GSVD}$ before converging to the latter. □

![Figure 2: Example 2.2.6. (a) $\|Bx_{\ell,\mu}\|$ as a function of $\ell$ for the test problem \texttt{baart} with $x_{\ell,\mu}$ computed by FATR with $\rho = 1/5$ for $\ell = 13, 14, \cdots, 50$. (b) $\|x_{\ell,\mu} - \hat{x}\|/\|\hat{x}\|$ as a function of $\ell$.](image)

**Example 2.2.7.** Consider the Fredholm integral equation of the first kind

$$
\int_0^1 k(s,t)x(t)dt = \exp(s) + (1 - e)s - 1, \quad 0 \leq s \leq 1,
$$

(46)

where

$$
k(s,t) = \begin{cases} 
s(t-1), & s < t, \\
t(s-1), & s \geq t. \end{cases}
$$

We discretize the integral equation by a Galerkin method with orthonormal box functions as test and trial functions using the MATLAB function \texttt{deriv2} from [25]. This yields the matrix $A \in \mathbb{R}^{500 \times 500}$. The function \texttt{deriv2} also produces a scaled discrete approximation $\hat{x} \in \mathbb{R}^{500}$ of the solution $x(t) = \exp(t)$ of (46). The error vector $e \in \mathbb{R}^{500}$ corresponds to the noise level $10^{-3}$.

It is well known that augmentation of Krylov or generalized Krylov subspaces by suitable vectors can enhance the quality of the computed approximate solutions; see, e.g.,
Figure 3: Example 2.2.7. Computed approximate solutions for the test problem deriv2 with noise-level $10^{-3}$. The red solid curve shows the approximate solution $x_{11,\mu_{11}}$ determined by FATR without augmentation and $\rho = \infty$, the blue dashed curve displays $x_{19,\mu_{19}}$ computed by FATR without augmentation and $\rho = 1/5$, and the black dash-dotted curve shows the $\hat{x}$.

[3,5,27] for examples. For the present example, it is beneficial to include an orthonormal basis for the span of the vectors

$$n_1 = [1, 1, \ldots, 1]^*, \quad n_2 = [1, 2, \ldots, n]^*$$

in the solution subspace. However, for many linear discrete ill-posed problems it is not clear which vectors should be included in the solution subspace to improve the quality of the computed solution. It is therefore desirable that solution methods perform well also in this situation. The present example compares the performance of FATR with and without the inclusion of the span of the vectors (47) in the solution subspace.

Figure 3 shows the approximate solution $x_{11,\mu_{11}}$ (red solid curve) computed by FATR with $\rho = \infty$ and without augmentation (i.e., without inclusion of the vectors (47) in the solution subspace), the approximate solution $x_{19,\mu_{19}}$ (blue dashed curve) computed by FATR with $\rho = 1/5$ and without augmentation, and the desired solution $\hat{x}$ (black
Table 2: Example 2.2.7. Relative errors and $\|Bx_{\ell,\mu}\|$ in computed approximate solutions for the test problem deriv2 with noise level $10^{-3}$.

dash-dotted curve). The relative errors and $\|Bx_{\ell,\mu}\|$ are reported in Table 2. There we also show results obtained by FATR and GATR when the solution subspace is augmented by an orthonormal basis for span\{$n_1, n_2$\}. For the former method augmentation means that the first two vectors $v_1$ and $v_2$ form an orthonormal basis for span\{$n_1, n_2$\}. The table shows that FATR for $\rho = 1/5$ and without augmentation yields a more accurate approximation of $\hat{x}$ than the augmented FATR method with $\rho = \infty$. This is important, because it can be difficult to choose suitable vectors for augmentation. FATR with $\rho = 1$ and $\rho = 1/5$ also is seen to yield better approximations of $\hat{x}$ than GSVD and the GATR method with solution subspace augmented by the span of the vectors (47). □

Example 2.2.8. Regard the inverse Laplace transform

$$\int_0^\infty \exp(-st)f(t)dt = \frac{1}{s + 1/2}, \quad 0 \leq s < \infty, \quad (48)$$

whose solution is $f(t) = \exp(-t/2)$. This problem is discussed, e.g., by Varah [28]. We use the function i_laplace from [25] to determine a discretization $A \in \mathbb{R}^{500 \times 500}$ of the integral operator and a discretized scaled solution $\hat{x} \in \mathbb{R}^{500}$. The error vector $e \in \mathbb{R}^{500}$ has noise level $10^{-3}$. We let $b = A\hat{x} + e$.

When applying the FATR method with $\rho = \infty$ or $\rho = 1$ to compute an approximate solution of (10), we obtain quite accurate approximations of the GSVD solution $x_{GSVD}$. However, $x_{GSVD}$ is a poor approximation of $\hat{x}$. A much better approximation of $\hat{x}$ can be achieved by letting $\rho$ be smaller than unity. Figure 4 depicts the approximation $x_{92,92}$.
Figure 4: Example 2.2.8. Computed approximate solutions for the problem \( i_{\text{laplace}} \) with noise level \( 10^{-3} \). The FATR method with \( \rho = 1/50 \) yields an accurate approximate solution \( x_{92,\mu_{92}} \) (red solid curve). Carrying out more steps yields a worse approximation \( x_{117,\mu_{117}} \) (blue dashed curve). The desired solution \( \hat{x} \) is also shown (black dash-dotted curve).

obtained by FATR with \( \rho = 1/50 \) (red solid curve). Carrying out 25 further iteration steps yields the computed solution \( x_{117,\mu_{117}} \) (blue dashed curve). This approximate solution is close to \( x_{GSVD} \). Table 3 shows the relative errors in \( x_{92,\mu_{92}}, x_{117,\mu_{117}}, \) and \( x_{GSVD} \). Moreover, \( \|Bx_{GSVD}\| = 1.13 \cdot 10^{-2} \).

Figure 5(a) shows the relative error in the approximate solutions \( x_{\ell,\mu_\ell} \) computed by FATR with \( \rho = 1/50 \) as a function of \( \ell \) for \( \ell = 50, 51, \ldots, 117 \). The error is small for \( 70 \leq \ell \leq 100 \). The figure illustrates that \( x_{\ell,\mu_\ell} \) furnishes an accurate approximation of \( \hat{x} \) for \( \ell \) sufficiently large, but not so large that \( x_{\ell,\mu_\ell} \) is close to \( x_{GSVD} \). □

Our last example in this subsection illustrates the performance of FATR when applied to the restoration of a two-dimensional gray-scale image that has been contaminated by blur and noise. The gray-scale images are represented by arrays of \( m \times m \) pixels, with each pixel stored as an 8-bit unsigned integer, with a value in the interval \([0, 255]\). The pixels
are ordered row-wise and stored in a vector of dimension $n = m^2$. Let $\hat{x} \in \mathbb{R}^n$ represent a blur- and noise-free image. We generate an associated blurred and noise-free image $\hat{b}$ by multiplying $\hat{x}$ with a Gaussian blurring operator $A \in \mathbb{R}^{n \times n}$. This matrix is generated with the MATLAB function `blur` from [25]. The matrix $A$ so obtained depends on the parameters `band` (the half-bandwidth of the Toeplitz blocks) and `sigma` (the variance of the Gaussian point spread function). Increasing `sigma` means more blur, while increasing `band` means that more storage and computational work is needed for the matrix-vector product evaluations. The blur- and noise-contaminated image $b \in \mathbb{R}^n$ is obtained by adding a “noise-vector” $e \in \mathbb{R}^n$ to $\hat{b}$; cf. (11). We assume the blurring matrix $A$, the
contaminated image $b \in \mathbb{R}^n$, and the noise-level $\delta$ to be available, and we would like to determine a restoration, which accurately approximates the blur- and noise-free image $\hat{x}$, which is assumed not to be available. We will generate the image restoration examples the same way in Sections 2.3.3 and 2.4.2.

The peak signal-to-noise ratio (PSNR) is commonly used to measure the quality of our restored image $x_{\ell,\mu}$. It is defined as

$$\text{PSNR}(x_{\ell,\mu}, \hat{x}) = 20 \log_{10} \left( \frac{255}{\|x_{\ell,\mu} - \hat{x}\|} \right),$$

where the numerator, 255, is the largest pixel-value of the image. Although a larger PSNR generally indicates that the restoration $x_{\ell,\mu}$ is of higher quality, in some cases the PSNR value might not agree with visual judgment. We therefore also display the restored images.

We use two regularization matrices: the standard discretization of the Laplacian using a five-point stencil

$$B(x) = \Delta x$$

and a discretization of the Perona–Malik operator

$$B(x) = \text{div}(g(|\nabla x|^2) \nabla x),$$

where $\nabla x$ denotes the gradient of $x$ considered as a real-valued function defined in $\mathbb{R}^2$. The diffusivity is given by

$$g(s) = \frac{\rho}{\rho + s}$$

with $\rho > 0$ a small positive constant; see [29] for discussions on this choice of diffusivity; we use the value $\rho = 10^{-5}$. 
We discretize (51) by finite differences. This gives a matrix $L(x)$ with, generically, five nonvanishing entries in each row. The entries in the row associated with pixel $(i, j)$ are determined by the values of the image $x$ at pixel $(i, j)$ and at the four adjacent pixels in the horizontal and vertical directions, denoted by $(N, S, E, W)$. The row of $L(x)$ associated with pixel $(i, j)$ is generically of the form

$$\{l_{i,j,S,0}, \ldots, 0, l_{i,j,E}, -(l_{i,j,S} + l_{i,j,E} + l_{i,j,W} + l_{i,j,N}), l_{i,j,W}, 0, \ldots, 0, l_{i,j,N}\},$$

with elements

$$l_{i,j,E} = \frac{g_{i,j} + g_{i+1,j}}{2}, \quad l_{i,j,W} = \frac{g_{i,j} + g_{i-1,j}}{2},$$

where $g_{i,j}$ represents the value of the discretization of the diffusivity $g(|\nabla x|^2)$ at pixel $(i, j)$. Partial derivatives are approximated by central finite differences, giving

$$g_{i,j} = g \left( \frac{(x_{i+1,j} - x_{i-1,j})}{2} \right)^2 + \left( \frac{x_{i,j+1} - x_{i,j-1}}{2} \right)^2,$$

where $x_{i,j}$ denotes the value of $x$ at pixel $(i, j)$. Expressions for $l_{i,j,S}(x)$ and $l_{i,j,N}(x)$ can be derived similarly; see [30] for details. Alternative discretizations are discussed in [31].

**Example 2.2.9.** We illustrate the performance of the FATR method when applied to the restoration of a blurred and noisy version of the $412 \times 412$-pixel image shown in Figure 6(a). This image has many fine details and well-defined edges. Contamination by severe blur, determined by the parameters $\text{band} = 9$ and $\text{sigma} = 3$ of the function $\text{blur}$ from [25], and by 30% additive Gaussian noise, yields the image displayed in Figure 6(b).

Figure 7 shows the restored images determined by FATR with the regularization matrix $B$ chosen to be the standard five-point discretization of the Laplace operator. The PSNR-values for the restoration are listed in Table 4. When $\rho = \infty$, the best restoration
Figure 6: Example 2.2.9. (Image restoration) (a) original image. (b) blur- and noise-contaminated image.

Table 4: Example 2.2.6. PSNR-values of restorations determined by FATR with the Laplacian regularization operator.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\ell$</th>
<th>PSNR</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\infty$</td>
<td>2</td>
<td>12.11</td>
<td>Figure 7 (a)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>12.47</td>
<td>Figure 7 (b)</td>
</tr>
<tr>
<td>$1/2$</td>
<td>3</td>
<td>12.55</td>
<td>Figure 7 (c)</td>
</tr>
</tbody>
</table>

is obtained after $\ell = 2$ steps; the restored image is shown in Figure 7(a), which is the most blurry restoration of the three images of Figure 7. It has PSNR=12.11. Letting $\rho = 1$, we again obtain the best restoration after only $\ell = 2$ steps. This restoration is displayed in Figure 7(b); its PSNR-value is 12.47. This image has less blurred edges than the image of Figure 7(a). Decreasing $\rho$ to $1/2$, we obtain, after three steps, the restoration with the largest PSNR-value: 12.55. It is depicted in Figure 7(c).

Finally, Figure 8(a) shows the restored image $x_{4,\mu_4}$ determined by the FATR method with the discretized Perona–Malik regularization operator (51) and $\rho = \infty$. The Perona–Malik operator is linearized by first carrying out two steps of the range restricted GMRES method [32]. This yields a better approximation $\hat{x}$ of $\hat{x}$ than $b$. We use $x_2$ to determine the discretized Perona–Malik operator, which then is kept fixed during subsequent
Figure 7: Example 2.2.9. (Image restoration) Image restored by the FATR method with the regularization matrix $B$ a discretization of the Laplacian 50. (a) Restored image $x_{2,\mu_2}$ obtained with $\rho = \infty$, (b) restored image $x_{2,\mu_2}$ determined with $\rho = 1$, and (c) restored image $x_{3,\mu_3}$ obtained with $\rho = 1/2$. Iterations. This linearization gives a more accurate restoration than linearizing the regularization operator at $b$.

The PSNR-value for $x_{4,\mu_4}$ is 13.53. This is larger than the PSNR-values of restorations computed with the discretized Laplace operator. The image has sharp edges. Figure 8(b) shows the edge map for the restoration of Figure 8(a). It is determined with the edge detector of gimp, a public domain software tool for image processing. The edge map is seen to be accurate despite the severe noise and blur in the image that we restored.

Thus, FATR with $\rho = 1/2$ gives the best restoration when the regularization matrix is a discrete Laplacian, while $\rho = \infty$ yields the best restoration when the regularization
Figure 8: Example 2.2.9. (Image restoration) (a) Restored image $x_{A,B}$ by FATR with Perona–Malik operator 51 with $\rho = \infty$, (b) edges map for the restoration in (a).

matrix is a discretization of the Perona–Malik operator. This illustrates the value of a method with a parameter $\rho$ that specifies the ratio of matrix-vector product evaluations with $A$ and with $B$. We are presently investigating how a suitable value of $\rho$ can be determined during the computations. □
2.3 A generalized Golub–Kahan-type decomposition method for matrix pairs

2.3.1 Standard Golub–Kahan bidiagonalization

We recall some properties of partial Golub–Kahan bidiagonalization of a matrix $A \in \mathbb{R}^{m \times n}$; pseudocode is given in Algorithm 2.3.1 below. The choice of initial vector $u_1 \in \mathbb{R}^m$ determines the partial bidiagonalization. The parameter $\ell$ specifies the number of steps.

For now, we assume $\ell$ to be small enough so that breakdown is avoided. Breakdown is discussed below. It occurs when certain vectors $\hat{u}$ or $\hat{v}$ in the algorithm vanish.

**Algorithm 2.3.1. Partial Golub–Kahan bidiagonalization of $A$.**

1. **Input:** matrix $A \in \mathbb{R}^{m \times n}$, initial unit vector $u_1$, and number of steps $\ell$
2. $v_0 := 0$
3. for $j = 1, 2, \ldots, \ell$ do
4.   $\hat{v} := A^* u_j - h_{j,j-1} v_{j-1}$
5.   $h_{j,j} := \|\hat{v}\|$  
6.   $v_j := \hat{v} / h_{j,j}$ if $h_{j,j} = 0$ : see text
7.   $\hat{u} := A v_j - h_{j,j} u_j$
8.   $h_{j+1,j} := \|\hat{u}\|$  
9.   $u_{j+1} := \hat{v} / h_{j+1,j}$ if $h_{j+1,j} = 0$ : see text
10. end for

The following aspects of the above algorithm are essential. An initial unit vector $u_1$ is provided and the algorithm alternatingly multiplies a vector $u_j$ by $A^*$ followed by orthogonalization to the most recently generated vector $v_{j-1}$, and alternatingly multiplies a vector $v_j$ by $A$ followed by orthogonalization to the most recently generated vector $u_j$. This secures that in exact arithmetic all computed vectors $v_1, \ldots, v_\ell$ are orthonormal and all computed vectors $u_1, \ldots, u_{\ell+1}$ are orthonormal. After $\ell \leq \min\{m, n\}$ steps,
Algorithm 2.3.1 determines the partial Golub–Kahan decompositions

$$AV_\ell = U_{\ell+1}H_{\ell+1,\ell}, \quad A^*U_\ell = V_\ell H_{\ell,\ell}^*,$$

where the matrices $U_{\ell+1} = [u_1, \ldots, u_{\ell+1}] \in \mathbb{R}^{m \times (\ell+1)}$ and $V_\ell = [v_1, \ldots, v_\ell] \in \mathbb{R}^{n \times \ell}$ have orthonormal columns, $U_\ell \in \mathbb{R}^{m \times \ell}$ is made up of the first $\ell$ columns of $U_{\ell+1}$, and the matrix $H_{\ell+1,\ell} \in \mathbb{R}^{(\ell+1) \times \ell}$ is lower bidiagonal; its nontrivial entries are the scalars $h_{j,k}$ determined by the algorithm. Finally, $H_{\ell,\ell}$ is the leading $\ell \times \ell$ submatrix of $H_{\ell+1,\ell}$.

Breakdown of the computations with Algorithm 2.3.1 occurs when either $h_{j,j} = 0$ in line 6 or $h_{j+1,j} = 0$ in line 9. In some applications it is then appropriate to terminate the computations, while in others we may choose to continue the method. For instance, if breakdown occurs in line 6, then we can continue the computations by letting $v_j$ be an arbitrary unit vector that is orthogonal to the already generated vectors $\{v_1, \ldots, v_{j-1}\}$.

Breakdown in line 9 can be handled similarly. By carrying out $\ell = \min\{m, n\}$ steps with Algorithm 2.3.1 in this manner, we see that any matrix can be bidiagonalized.

**Proposition 2.3.1.** (a) For an arbitrary unit vector $u_1 \in \mathbb{R}^m$, there are orthogonal matrices $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$ and $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$ such that

$$H = U^*AV$$

is a lower bidiagonal matrix with nonnegative entries.

(b) The matrix $H$ is uniquely determined if all diagonal and subdiagonal elements are nonzero (and hence positive).

(c) Given $u_1$, the matrices $U$ and $V$ are uniquely determined if $A$ is square and all diagonal and subdiagonal elements of $H$ are nonzero.

We conclude this subsection with two remarks about Proposition 2.3.1: First, if $m > n$ or $n < m$, then the last vectors $u_{n+1}, \ldots, u_m$ or $v_{m+1}, \ldots, v_n$ are not determined.
uniquely. Moreover, if some nontrivial element of the bidiagonal matrix $H$ vanishes, then the next $u$-vector or $v$-vector is not determined uniquely. Second, we note that the Golub–Kahan bidiagonalization process is a particularly efficient way of determining the leading entries of $H$ when the matrix $A$ is large and sparse, because then the required matrix-vector products with $A$ and $A^*$ are fairly inexpensive to evaluate.

2.3.2 Generalized Golub–Kahan-type decomposition method for matrix pairs

In Section 2.2 we introduced the flexible Arnoldi method. Algorithm 2.2.1 constructs the solution space by either choosing a column of $U$ or a column of $W$. Special choices of the vector $\hat{u}$ and $\hat{w}$ in Algorithm 2.2.1, yield the Golub–Kahan bidiagonalization procedure. This subsection is dedicated to a generalized Golub–Kahan-type decomposition method for matrix pairs and its application to Tikhonov regularization. We first discuss the standard Golub–Kahan method for one matrix $A \in \mathbb{R}^{m \times n}$, then generalize to matrix pairs $\{A, B\}$. Algorithm 2.3.2, which yields the decomposition of matrix pair $\{A, B\}$

$$AV_{\ell} = U_{\ell+1}H^{(A)}_{\ell+1, \ell}, \quad (52)$$

$$BV_{\ell} = W_{\ell}H^{(B)}_{\ell, \ell}, \quad (53)$$

is given after the theorems. Proposition 2.3.2 to Theorem 2.3.6 explore the structure of the matrices $H^{(A)}_{\ell+1, \ell}$ and $H^{(B)}_{\ell, \ell}$ above.

**Proposition 2.3.2.** Let $A \in \mathbb{R}^{m \times n}$ and let $u_1 \in \mathbb{R}^m$ be a unit vector. Then $k$ steps of Golub–Kahan bidiagonalization applied to $A$ with initial vector $u_1$ yields the decompositions

$$AV_k = U_{k+1}H_{k+1,k}, \quad (54)$$

$$A^*U_{k+1} = V_{k+1}K_{k+1,k+1}, \quad (55)$$
where the columns of the matrices \( U_{k+1} = [u_1, u_2, \ldots, u_{k+1}] \in \mathbb{R}^{m \times (k+1)} \) and \( V_{k+1} = [v_1, v_2, \ldots, v_{k+1}] \in \mathbb{R}^{n \times (k+1)} \) are orthonormal, the matrix \( H_{k+1,k} \in \mathbb{R}^{(k+1) \times k} \) is lower bidiagonal and the leading \( k \times (k+1) \) submatrix of \( K_{k+1,k+1} \in \mathbb{R}^{(k+1) \times (k+1)} \) satisfies

\[
K_{k,k+1} = H_{k+1,k}^*.
\]

The initial column \( v_1 \) of \( V_{k+1} \) is determined by (55) with \( k = 0 \) so that \( k_{1,1} > 0 \). Generically, the diagonal and subdiagonal entries of \( H_{k+1,k} \) can be chosen to be positive. The decompositions (54) and (55) then are uniquely determined.

Proof. The columns \( u_2, v_2, u_3, v_3, \ldots \) of \( U_{k+1} \) and \( V_{k+1} \) are generated, in order, by alternatingly using equations (54) and (55) for increasing values of \( k \). Thus, the column \( u_2 \) is determined by requiring \( u_2 \) be of unit length, be orthogonal to \( u_1 \), and to satisfy (54) for \( k = 1 \) and a positive subdiagonal entry \( h_{2,1} \) of \( H_{2,1} \), where we note that \( h_{1,1} = k_{1,1} \). This determines both \( u_2 \) and \( h_{2,1} \). The column \( v_2 \) of \( V_2 \) is now defined by equation (55) for \( k = 2 \). The column is uniquely determined by the requirements that \( v_2 \) be orthogonal to \( v_1 \), of unit length, and such that the last diagonal entry of \( K_{2,2} \) is positive. This entry equals \( h_{2,2} \). The next vector to be evaluated is \( u_3 \). Generically, the computations can be continued in the manner indicated until the decompositions (54) and (55) have been computed.

In rare situations, the computations cannot be completed as described, because the first, say \( j \), generated columns \( v_1, v_2, \ldots, v_j \) of \( V_{k+1} \) span an invariant subspace of \( A^*A \). This situation is referred to as breakdown. The computations can be continued by letting the next column, \( v_{j+1} \), be an arbitrary unit vector that is orthogonal to \( \text{span}\{v_1, v_2, \ldots, v_j\} \). The situation when the first \( j \) generated columns of \( U_{k+1} \) span an invariant subspace of \( AA^* \) can be handled analogously and also is referred to as breakdown. In case of breakdown, suitable entries of \( H_{k+1,k} \) and \( K_{k+1,k+1} \) are set to zero so that the decompositions (54) and (55) are valid. The decompositions are not unique when breakdown occurs.
In applications of partial Golub–Kahan bidiagonalization to the solution of least-squares problems \( \min_x \|Ax - b\| \), one generally chooses the initial vector \( u_1 = b/\|b\| \).

Available descriptions of the Golub–Kahan bidiagonalization use that \( K_{k,k+1} \) can be expressed in terms of \( H_{k+1,k} \), see (56), and do not explicitly use the matrix \( K_{k+1,k+1} \). It is convenient for our discussion below to distinguish between the matrices \( H_{k+1,k} \) and \( K_{k,k+1} \).

We now turn to a modification of Golub–Kahan bidiagonalization that allows the choice of a fairly arbitrary column \( v_{i+1} \) in addition to the column \( u_1 \). The matrices \( U_{k+1} \) and \( V_{k+1} \) generated will have orthonormal columns, similarly as in the decompositions (54) and (55), but the structure of the matrices analogous to \( H_{k+1,k} \) and \( K_{k+1,k+1} \) in (54) and (55) will be different. We assume for notational simplicity the generic situation that no breakdown takes place. Let the decompositions (54) and (55) be available for \( k = i - 1 \), i.e., we have

\[
AV_{i-1} = U_i H_{i,i-1}, \tag{57}
\]

\[
A^*U_i = V_i K_{i,i}, \tag{58}
\]

with \( K_{i-1,i-1} = H_{i-1,i-1}^* \). Determine the column \( u_{i+1} \) of \( U_{k+1} \) from (54) with \( k = i \). This defines the entry \( h_{i+1,i} > 0 \) of \( H_{i+1,i} \). Now let the column \( v_{i+1} \) of \( V_{k+1} \) be an arbitrary unit vector such that

\[
v_{i+1} \perp \text{span}\{v_1, v_2, \ldots, v_i\}. \tag{59}
\]

We proceed to compute the column \( u_{i+2} \) of \( U_{k+1} \) by using (54) with \( k = i + 1 \). This determines the last column of \( H_{i+2,i+1} \). We will show below that all entries above the diagonal in column \( H_{i,i+1} \) vanish. The column \( v_{i+2} \) of \( V_{i+2} \) is chosen to be of unit length,
orthogonal to the preceding columns of $V_{i+2}$, and such that the relation

$$A^*U_{i+1} = V_{i+2}K_{i+2,i+1}$$

holds. We then compute the columns $u_{i+3}, v_{i+3}, \ldots, u_{k+1}, v_{k+1}$, in order, from decompositions of the form

$$AV_j = U_{j+1}H_{j+1,j}, \quad (60)$$

$$A^*U_j = V_{j+1}K_{j+1,j}, \quad (61)$$

for $j = i + 2, i + 3, \ldots, k$. The following theorem describes the structure of the matrices $H_{k+1,k}$ and $K_{k+1,k}$.

**Theorem 2.3.3.** Let the decompositions (60) and (61) for $j = k$ be generated as described above and assume that no breakdown occurs. Then the columns of $U_{k+1} \in \mathbb{R}^{m \times (k+1)}$ and $V_{k+1} \in \mathbb{R}^{n \times (k+1)}$ are orthonormal and the matrix $H_{k+1,k} \in \mathbb{R}^{(k+1) \times k}$ has the structure

$$H_{k+1,k} = \begin{bmatrix}
h_{1,1} & O \\
h_{2,1} & h_{2,2} & \ \\
 & \ddots & \ddots & \ddots \\
 & & h_{i+1,i+1} & h_{i+1,i+2} & \ddots \\
 & & & h_{i+2,i+1} & h_{i+2,i+2} & \ddots \\
 & & & & h_{i+3,i+2} & \ddots & h_{k-1,k} \\
 & & & & & \ddots & h_{k,k} \\
O & & & & & h_{k+1,k}
\end{bmatrix}.$$

Thus, the leading principal $(i + 2) \times (i + 1)$ submatrix is lower bidiagonal and the matrix $H_{k+1,k}$ is tridiagonal. Furthermore, $K_{k,k} = H_{k,k}^*$. 

**Proof.** Let the decompositions (57) and (58) be available. The matrix $H_{i,i-1}$ in (57)
is lower bidiagonal by Proposition 2.3.2. The next step in the Golub–Kahan bidiagonalization method is to replace \( V_{i-1} \) by \( V_i \) in (57) and define the matrix \( U_{i+1} \) by appending a suitable column \( u_{i+1} \) to \( U_i \). Append a zero row to \( H_{i,i-1} \) and the column 
\[ [h_{1,i}, h_{2,i}, \ldots, h_{i+1,i}]^* \]
to the matrix so obtained. This gives a decomposition of the form (57) with \( i \) replaced by \( i + 1 \). The entries \( h_{j,i} \) are defined by
\[
Av_i = \sum_{j=1}^{i+1} h_{j,i}u_j,
\]
where we choose \( h_{i+1,i} > 0 \) so that \( u_{i+1} \) is a unit vector that is orthogonal to \( u_1, u_2, \ldots, u_i \).

It follows from (58) and the fact that \( K_{i,i} \) is upper triangular that
\[
A^*u_j \in \text{span}\{v_1, v_2, \ldots, v_j\}, \quad j = 1, 2, \ldots, i.
\]

Therefore,
\[
h_{j,i} = u_j^*Av_i = v_i^*(A^*u_j) = 0, \quad j = 1, 2, \ldots, i - 1.
\]
The last diagonal entry of \( H_{i+1,i} \) is determined by (58), i.e., by
\[
h_{i,i} = u_i^*Av_i = k_{i,i}.
\]

Thus, we have obtained the desired decomposition
\[
AV_i = U_{i+1}H_{i+1,i},
\]
where the columns of \( U_{i+1} \) are orthonormal and \( H_{i+1,i} \) is lower bidiagonal.

Let \( v_{i+1} \) be a unit vector that satisfies (59). We proceed as above to determine a new unit vector \( u_{i+2} \) that we append to \( U_{i+1} \) to determine the matrix \( U_{i+2} \) with orthonormal columns. Append a zero row to \( H_{i+1,i} \) and the column 
\[ [h_{1,i+1}, h_{2,i+1}, \ldots, h_{i+2,i+1}]^* \]
to the matrix so obtained. Our aim is to determine a decomposition of the form (57) with \( i \) replaced by \( i + 1 \). Therefore, analogously to (62), we let
\[
Av_{i+1} = \sum_{j=1}^{i+2} h_{j,i+1}u_j
\]
and choose $h_{i+2,i+1} > 0$ so that $u_{i+2}$ is a unit vector that is orthogonal to the vectors $u_1, u_2, \ldots, u_{i+1}$. It follows from (63) that
\[ h_{j,i+1} = u_j^* A v_{i+1} = v_{i+1}^* (A^* u_j) = 0, \quad 1 \leq j \leq i. \]

The remaining entry of the last column of $H_{i+1}$ is defined by $h_{i+1,i+1} = u_{i+1}^* A v_{i+1}$. Thus, we have determined the decomposition
\[ AV_{i+1} = U_{i+2} H_{i+2,i+1}, \]
where the columns of $U_{i+2}$ and $V_{i+1}$ are orthonormal and $H_{i+2,i+1}$ is lower bidiagonal.

To proceed, let $v_{i+2}$ be a unit vector that is orthogonal to $\{ v_1, v_2, \ldots, v_{i+1} \}$ and satisfies
\[ A^* u_{i+1} = \sum_{j=1}^{i+2} k_{j,i+1} v_j \]
with $k_{i+2,i+1} > 0$. It follows from (64) and the structure of $H_{i+2,i+1}$ that $k_{j,i+1} = 0$ for $1 \leq j < i$. We first append two zero rows to the matrix $K_i$ and then the column $[k_{1,i+1}, k_{2,i+1}, \ldots, k_{i+2,i+1}]^*$ to the matrix so obtained. This defines the matrix $K_{i+2,i+1}$. By construction, it satisfies
\[ A^* U_{i+1} = V_{i+2} K_{i+2,i+1}. \]
Hence, the matrix $V_{i+2}$ has orthonormal columns, the last column of $K_{i+2,i+1}$ has at most three nonvanishing entries, and the submatrix $K_{i+1,i+1}$ is upper bidiagonal.

We continue to define the column $u_{i+3}$ of the matrix $U_{i+3}$ with the aim of obtaining a decomposition of the form
\[ AV_{i+2} = U_{i+3} H_{i+3,i+2}. \]
Specifically, we let $u_{i+3}$ be of unit length, orthogonal to $u_1, u_2, \ldots, u_{i+2}$, and such that
\[ Av_{i+2} = \sum_{j=1}^{i+3} h_{j,i+2} u_j \]
with $h_{i+3,i+2} > 0$. It follows from (65) and the structure of $K_{i+2,i+1}$ that
\[ h_{j,i+2} = u_j^* A v_{i+2} = 0, \quad 1 \leq j \leq i. \]
Hence, only the last three entries of the vector $[h_{1,i+2}, h_{2,i+2}, \ldots, h_{i+3,i+2}]^*$, which is the last column of the matrix $H_{i+3,i+2}$, may be nonvanishing.

We proceed by defining new columns of the matrices $U_{k+1}$ and $V_{k+1}$ in this manner until the decompositions (60) and (61) have been determined for $j = k$. 

Results analogous to Theorem 2.3.3 can be obtained by letting the column $u_{i+1}$ of $U_{k+1}$ be an arbitrary unit vector that is orthogonal to the preceding columns $u_1, u_2, \ldots, u_i$. We will not dwell on this situation, since it is of little interest for our numerical method for Tikhonov regularization.

The special case of Theorem 2.3.3 when both the initial columns of $U_{k+1}$ and $V_{k+1}$ are chosen to be arbitrary unit vectors is described by the following corollary. It has previously been discussed in [33,34].

**Corollary 2.3.4.** Let the initial columns of the matrices $U_{k+1} \in \mathbb{R}^{m \times (k+1)}$ and $V_{k+1} \in \mathbb{R}^{n \times (k+1)}$ be arbitrary unit vectors. Determine the remaining columns similarly as in Theorem 2.3.3 and assume that no breakdown occurs. Then the matrices $U_{k+1}$ and $V_{k+1}$ satisfy the relations

$$AV_k = U_{k+1}H_{k+1,k},$$

$$A^*U_k = V_{k+1}K_{k+1,k},$$

where $H_{k+1,k} \in \mathbb{R}^{(k+1)\times k}$ is tridiagonal and $K_{k,k} = H_{k,k}^*$. 

**Proof.** The result is a consequence of Theorem 2.3.3. Breakdown of the recursions is discussed in [33]. 

We can extend Theorem 2.3.3 to allow the inclusion of several arbitrary orthonormal columns in the matrix $V_{k+1}$.
Theorem 2.3.5. Let the indices $i_j$ be ordered so that $1 \leq i_1 < i_2 < \ldots < i_s \leq k$, and let $v_{i_1}, v_{i_2}, \ldots, v_{i_s}$ be arbitrary unit vectors such that $v_{i_\ell}$ is orthogonal to all preceding columns $v_1, v_2, \ldots, v_{i_\ell-1}$ of $V_{k+1}$ for $1 \leq \ell \leq s$. Introducing these columns similarly as the column $v_{i+1}$ in Theorem 2.3.3 yields the decompositions

$$AV_k = U_{k+1}H_{k+1,k},$$
$$A^*U_{k+1-s} = V_{k+1}K_{k+1,k+1-s},$$

where $U_{k+1} \in \mathbb{R}^{m \times (k+1)}$ and $V_{k+1} \in \mathbb{R}^{n \times (k+1)}$ have orthonormal columns. The matrices $H_{k+1,k} \in \mathbb{R}^{(k+1) \times k}$ and $K_{k+1,k+1-s} \in \mathbb{R}^{(k+1) \times (k+1-s)}$ are banded and satisfy $(H_{k+1-s,k})^* = K_{k+1,k+1-s}$. Moreover, $H_{k+1,k}$ is upper Hessenberg and such that

- all entries but $h_{j+1,j}$ and $h_{j,j}$ of the column $H_{i,j}$ vanish for $j \leq i_1$,
- all entries but $h_{j+1,j}, h_{j,j}, \ldots, h_{j-t,j}$ of the column $H_{i,j}$ vanish for $i_t < j \leq i_{t+1}$, where $1 \leq t \leq s-1$.
- all entries but $h_{j+1,j}, h_{j,j}, \ldots, h_{j-s,j}$ of the column $H_{i,j}$ vanish for $j > i_s$.

Proof. Theorem 2.3.3 shows that when introducing an arbitrary unit vector $v_{i_1}$ that is orthogonal to the preceding vectors $v_1, v_2, \ldots, v_{i_1-1}$, the upper bandwidth of the matrix $H_{k+1,k}$ increases by one starting at column $i_1 + 1$. A slight modification of the proof of Theorem 2.3.3 shows that if a new arbitrary unit vector $v_{i_2}$, that is orthogonal to the preceding vectors $v_1, v_2, \ldots, v_{i_2-1}$, is introduced, then the upper bandwidth of $H_{k+1,k}$ is increased by one starting at column $i_2 + 1$. Repeating this process for all vectors $v_{i_1}, v_{i_2}, \ldots, v_{i_s}$ shows the theorem.

The above theorem forms the basis for our generalized Golub–Kahan reduction method for matrix pairs $\{A, B\}$. We first present an outline of this method. A detailed algorithm is presented below. Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$, and let the first column $u_1$ of the matrix $U = [u_1, u_2, \ldots]$ be an arbitrary unit vector. Define the first column of the
matrix $V = [v_1, v_2, \ldots]$ by $v_1 = A^*u_1/\|A^*u_1\|$. Let the matrix $U_j$ consist of the first $j$ columns of $U$; similar notation is used for the matrices $V$ and $W$. Further, $H_{j+1,j}^{(A)}$ denotes the leading principal $(j+1) \times j$ submatrix of $H^{(A)}$. We use the same notation for the matrices $H^{(B)}$, $K^{(A)}$, and $K^{(B)}$. Let the $(1,1)$–entries of $H^{(A)}$ and $K^{(A)}$ be given by $h_{1,1}^{(A)} = k_{1,1}^{(A)} = \|A^*u_1\|$. We generate successive columns of the matrices $U$, $V$, $W$, $H^{(A)}$, $H^{(B)}$, $K^{(A)}$, and $K^{(B)}$ in the following manner.

**Algorithm Outline 2.3.1.**

*Initialization:* $s_A = 1$; $s_B = 0$; $\mathbb{P}_A = \{1\}$; $\mathbb{P}_B = \emptyset$;

*Iteration:*

for $j = 1, 2, 3, \ldots$ :

- **Determine the new columns $u_{j+1}$ and $w_j$ of the matrices $U$ and $W$ by equating**

  $$AV_j = U_{j+1}H_{j+1,j}^{(A)},$$

  $$BV_j = W_jH_{j,j}^{(B)},$$

  so that the matrices $U_{j+1}$ and $W_j$ have orthonormal columns, $H_{j+1,j}^{(A)} \in \mathbb{R}^{(j+1) \times j}$ is upper Hessenberg, and $H_{j,j}^{(B)} \in \mathbb{R}^{j \times j}$ is upper triangular.

- **Determine the new column $v_{j+1}$ of the matrix $V$ by equating one of the formulas and carrying out the other required computations**

  (i): $A^*U_{j+1-s_B} = V_{j+1}K_{j+1,j+1-s_B}^{(A)}$; $s_A = s_A + 1$; $\mathbb{P}_A = \mathbb{P}_A \cup \{j + 1\}$;

  (ii): $B^*W_{j+1-s_A} = V_{j+1}K_{j+1,j+1-s_A}^{(B)}$; $s_B = s_B + 1$; $\mathbb{P}_B = \mathbb{P}_B \cup \{j + 1\}$;

so that $V_{j+1}$ has orthonormal columns. The matrices $K_{j+1,j+1-s_B}^{(A)} \in \mathbb{R}^{(j+1) \times (j+1-s_B)}$ and $K_{j+1,j+1-s_A}^{(B)} \in \mathbb{R}^{(j+1) \times (j+1-s_A)}$ have zero entries below the diagonal. The indices $s_A$ and $s_B$ count the number of columns of $V$ that have been determined.
by equating (i) and (ii), respectively. Thus, $s_A + s_B = j$. The index sets $\mathcal{P}_A$ and $\mathcal{P}_B$ are used in Theorem 2.3.6 below.

\text{end j-loop}

\textbf{Theorem 2.3.6.} Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times n}$, and let the first columns of the matrices $U$ and $V$ be defined as in the Algorithm Outline 2.3.1. Then, assuming that no breakdown occurs, $k$ iteration steps described by Algorithm Outline 2.3.1 yield the decompositions

\begin{align*}
AV_k &= U_{k+1}H^{(A)}_{k+1,k}, \\
BV_k &= W_kH^{(B)}_{k,k}, \\
A^*U_{k+1-s_B} &= V_{k+1}K^{(A)}_{k+1,k+1-s_B}, \\
B^*W_{k+1-s_A} &= V_{k+1}K^{(B)}_{k+1,k+1-s_A}.
\end{align*}

(66)

(67)

(68)

Assume that the vectors $v_{i_1}, v_{i_2}, \ldots, v_{i_{sB}}$ are generated by using equation (ii). Then $i_1 > 1$ and $H^{(A)}$ has the structure described by Theorem 2.3.5 with the indices $i_1 < i_2 < \ldots < i_s$ defined by $\mathcal{P}_B = \{i_j\}_{j=1}^{s}$. Let the indices $k_1 < k_2 < \ldots < k_t$ be defined by $\mathcal{P}_A = \{k_j\}_{j=1}^{t}$. Then $k_1 = 1$ and the matrix $H^{(B)}$ is upper triangular and such that the column $H^{(B)}_{:,j}$ has at most $1+q$ nonzero entries $h^{(B)}_{j,j}, h^{(B)}_{j+1,j}, \ldots, h^{(B)}_{j+q,j}$ for $i_q < j \leq i_{q+1}$.

\textit{Proof.} The structure of $H^{(A)}$ is obtained from Theorem 2.3.5 by replacing $s$ by $s_B$ and by letting the vectors $v_{i_1}, v_{i_2}, \ldots, v_{i_{sA}}$ of Theorem 2.3.5 be $v_{i_1}, v_{i_2}, \ldots, v_{i_{sB}}$ of the present theorem.

The structure of $H^{(B)}$ can be shown similarly as the structure of $H^{(A)}$ as follows. Consider the two-part iteration (66) and (68) to generate the first $i - 1$ columns of $V$. This is Golub–Kahan bidiagonalization with initial vector $v_i$. The matrix $H^{(B)}$ determined is upper bidiagonal. Now let $v_i$ be determined by (67) for $i = \ell_1, \ell_2, \ldots, \ell_{s_A}$. 
We apply Theorem 2.3.3 repeatedly, similarly as in the proof of Theorem 2.3.5, to show the structure of $H(B)$.

Algorithm 2.3.2 below describes a particular implementation of the Algorithm Outline 2.3.1, in which a parameter $\rho > 0$ determines whether step (i) or step (ii) should be executed to determine a new column of $V$. The value of $\rho$ affects the solution subspace $\mathcal{R}(V_k)$ generated by the algorithm. This is illustrated below. Computed examples in Section 2.3.3 show that it may be beneficial to apply step (i) a different number of times than step (ii).

The counters $N(u)$ and $N(w)$ in Algorithm 2.3.2 are indices used when generating the next column of $V$. 

Algorithm 2.3.2. Generalized Golub–Kahan-type reduction to matrix pairs \( \{ A, B \} \).

1. Input: matrices \( A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{p \times n} \), unit vector \( \mathbf{u}_1 \in \mathbb{R}^n \), ratio \( \rho \geq 0 \), and number of steps \( \ell \)

2. \( \hat{\mathbf{v}} := A^* \mathbf{u}_1; \ h_{1,1}^{(A)} := \| \hat{\mathbf{v}} \|; \ v_1 := \hat{\mathbf{v}} / h_{1,1}^{(A)} \)

3. \( N(\mathbf{u}) := 1; \ N(\mathbf{w}) := 1 \)

4. for \( j = 1, 2, \ldots, \ell \) do

5. \( \hat{\mathbf{u}} := A \mathbf{v}_j \)

6. for \( i = 1, 2, \ldots, j \) do

7. \( h_{i,j}^{(A)} := \mathbf{u}_i^* \hat{\mathbf{u}}; \ \hat{\mathbf{u}} := \hat{\mathbf{u}} - \mathbf{u}_i h_{i,j}^{(A)} \)

8. end for

9. \( h_{j+1,j}^{(A)} := \| \hat{\mathbf{u}} \| \)

10. \( \mathbf{u}_{j+1} := \hat{\mathbf{u}} / h_{j+1,j}^{(A)} \) \( \quad \text{if} \ h_{j+1,j}^{(A)} = 0 : \text{see text} \)

11. \( \hat{\mathbf{w}} := B \mathbf{v}_j \)

12. for \( i = 1, 2, \ldots, j - 1 \) do

13. \( h_{i,j}^{(B)} := \mathbf{w}_i^* \hat{\mathbf{w}}; \ \hat{\mathbf{w}} := \hat{\mathbf{w}} - \mathbf{w}_i h_{i,j}^{(B)} \)

14. end for

15. \( h_{j,j}^{(B)} := \| \hat{\mathbf{w}} \| \)

16. \( \mathbf{w}_j := \hat{\mathbf{w}} / h_{j,j}^{(B)} \) \( \quad \text{if} \ h_{j,j}^{(B)} = 0 : \text{see text} \)

17. if \( N(\mathbf{w}) / N(\mathbf{u}) > 1 / \rho \)

18. \( N(\mathbf{u}) := N(\mathbf{u}) + 1; \ v := A^* \mathbf{u}_{N(\mathbf{u})} \)

19. else

20. \( v := B^* \mathbf{w}_{N(\mathbf{w})}; \ N(\mathbf{w}) := N(\mathbf{w}) + 1 \)

21. end

22. for \( i = 1, 2, \ldots, j \) do

23. \( v := v - (\mathbf{v}_i^* v) \mathbf{v}_i \)

24. end for

25. \( \alpha_j := \| v \| \)

26. \( \mathbf{v}_{j+1} := v / \alpha_j \) \( \quad \text{new} \ v \text{-vector, if} \ h_{j+1,j} = 0 : \text{see text} \)

27. end for
Example 2.3.7. Let $\rho = 1$. Then $\ell$ steps with Algorithm 2.3.2 generates the matrix $V_\ell$ with range

$$\mathcal{R}(V_\ell) = \text{span}\{A^*b, B^*BA^*b, A^*AA^*b, (B^*B)^2A^*b, A^*AB^*BA^*b, B^*BA^*AA^*b, (A^*A)^2A^*b, \ldots \}. \quad \square$$

Example 2.3.8. Let $\rho = 1/2$. Then each application of step (i) is followed by two applications of step (ii). This yields a subspace of the form

$$\mathcal{R}(V_\ell) = \text{span}\{A^*b, B^*BA^*b, (B^*B)^2A^*b, A^*AA^*b, (B^*B)^3A^*b, B^*BA^*AA^*b, A^*AB^*BA^*b, \ldots \}. $$

The computation of the matrix $V_\ell$ in this example requires more matrix-vector product evaluations with the matrix $B^*$ than with $A^*$ than the determination of the corresponding matrix $V_\ell$ of Example 2.3.7. In many applications of Tikhonov regularization, $B^*$ represents a discretization of a differential operator and is sparse. Typically, the evaluation of matrix-vector products with $B^*$ is cheaper than with $A^*$. Therefore, the computations of a solution subspace of dimension $\ell$ generally is cheaper when $\rho = 1/2$ than when $\ell = 1$. Moreover, Example 2.3.13 of Section 2.3.3 shows that $\rho < 1$ can yield more accurate approximations of the desired solution $\hat{x}$ than $\rho = 1$. \quad \square

Algorithm 2.3.2 is said to break down if one of the coefficients $h_{j+1,j}^{(A)}$, $h_{j,j}^{(B)}$, or $\alpha_j$ in lines 10, 16, or 26 vanishes. Breakdown is very unusual in our application to Tikhonov regularization. If breakdown occurs in lines 10 or 16, then we may terminate the computations with the algorithm and solve the available reduced problem. When breakdown takes place in line 26, we ignore the computed vector $v$ and generate a new vector $v$ via either line 18 or line 20. Breakdown also could be handled in other ways. The occurrence of breakdown may affect the structure of the matrices $H^{(A)}$ and $H^{(B)}$. We apply Algorithm 2.3.2 with input matrix $A$, $B$ and vector $b$ in (14). This yields the decomposition
(52) and (53), which forms the downsized Tikhonov regularization that can be solved in the way described as in Section 2.2.

2.3.3 Numerical examples

We present several numerical examples with the one-parameter Tikhonov regularization method discussed in Section 2.2.2, while the matrix $H^{(A)}$ and $H^{(B)}$ are generated by Algorithm 2.3.2. To illustrate the importance of the use of a user-chosen regularization matrix, we also report results achieved with the range restricted GMRES (RRGMRES) iterative method. This method requires that the matrix $A$ be square. The $\ell$th iterate, $x_\ell$, determined by RRGMRES when applied to the approximate solution of $\min_x \|Ax - b\|$ satisfies

$$\|Ax_\ell - b\| = \min_{x \in K_\ell(A, Ab)} \|Ax - b\|, \quad x_\ell \in K_\ell(A, Ab).$$

An implementation of this method based on the Arnoldi process is described in [32]. Examples 2.3.9 and 2.3.10 illustrate the performance when generalized Golub–Kahan Tikhonov regularization (GGKTR) method is used for the solution of linear discrete ill-posed problems from Regularization Tools [25] with rectangular regularization matrices. Examples 2.3.11 and 2.3.12 consider the restoration of blurred and noisy 2D-images. The last example illustrate that GGKTR performs better when $\rho < 1$. The starting vector for Algorithm 2.3.2 is $u_1 = b/\|b\|$ for all examples.

Example 2.3.9. Consider the test problem deriv2 as in the Example 2.2.7. We discretize the integral equation (46) using MATLAB code deriv2 from [25]. This yields the matrix
Figure 9: Example 2.3.9. Computed approximate solutions for the test problem deriv2 with noise-level $10^{-3}$. (a) $B$ is the bidiagonal matrix $L_1$ in (1). (b) $B$ is the tridiagonal matrix $L_2$ in (2). The curves determined with the GSVD are on top of the curves obtained with GGKTR($\rho = 1$).

<table>
<thead>
<tr>
<th>$B$</th>
<th>GGKTR($\rho = 1$)</th>
<th>GSVD</th>
<th>RRGMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidiagonal (1)</td>
<td>$1.17 \cdot 10^{-2}$ ($\ell = 18$)</td>
<td>$1.23 \cdot 10^{-2}$</td>
<td>$1.43 \cdot 10^{-1}$ ($\ell = 11$)</td>
</tr>
<tr>
<td>Tridiagonal (2)</td>
<td>$3.38 \cdot 10^{-3}$ ($\ell = 37$)</td>
<td>$3.41 \cdot 10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Example 2.3.9. Relative errors in computed solutions for noise-level $10^{-3}$. The parameter $\ell$ denotes the number of steps with GGKTR($\rho = 1$) and RRGMRES.

$A \in \mathbb{R}^{1000 \times 1000}$ and the “exact” solution $\hat{x} \in \mathbb{R}^{1000}$. We determine the “noise-free” vector $\hat{b} := A\hat{x}$ to which we add an error vector $e \in \mathbb{R}^{1000}$ that corresponds to the noise-level $10^{-3}$. The bidiagonal regularization matrix $L_1$ (1) and the tridiagonal regularization matrix $L_2$ (2) are used.

We apply GGKTR with $\rho = 1$ to the matrix pair $\{A, B\}$ and determine the regularization parameter with the aid of the discrepancy principle. Figure 9 displays the approximate solutions $x_{18,\mu_{18}}$ and $x_{37,\mu_{37}}$ of (10) determined by GGKTR($\rho = 1$) (red continuous curves), the approximate solutions computed by using the GSVD of the matrix pair $\{A, B\}$ (blue continuous curves), and the desired solution $\hat{x}$ (black dash-dotted curves). The regularization parameter for the GSVD approximate solutions is determined by the discrepancy principle using the MATLAB function discrep from [25]. Clearly, the
tridiagonal regularization matrix (2) gives the best approximation of \( \hat{x} \).

Table 5 shows numerical values for the errors in the computed approximate solutions. The number of steps \( \ell \) with GGKTR(\( \rho = 1 \)) is chosen to give the best approximation of the desired solution \( \hat{x} \). Table 5 shows the approximations of \( \hat{x} \) determined in this manner to be slightly more accurate approximations of \( \hat{x} \) than the approximation solutions determined by GSVD. The table also shows the error in the most accurate approximation of \( \hat{x} \) computed by the RRGMRES iterative method. The poor approximation achieved with RRGMRES illustrates the benefit of using the regularization matrices (1) and (2) for the problem at hand.

Figure 10: Example 2.3.9. Relative error in computed approximate solutions \( \mathbf{x}_{\ell,\mu\ell} \) as a function of \( \ell \geq 27 \) for the test problem deriv2 with noise-level \( 10^{-3} \) and the tridiagonal regularization matrix (2).

Figure 10 shows the error in the computed approximate solutions \( \mathbf{x}_{\ell,\mu\ell} \) as a function of the number of steps \( \ell \) when the tridiagonal regularization matrix (2) is used. The regularization parameter \( \mu\ell \) is determined by the discrepancy principle. It can not be satisfied for small values of \( \ell \). Approximate solutions \( \mathbf{x}_{\ell,\mu\ell} \) that satisfy the discrepancy
principle can be computed for $\ell \geq 27$. Increasing $\ell$, Figure 10 shows the error in $x_{\ell,\mu}$ to drop significantly when $\ell$ reaches 37 and then stays about the same when $\ell$ is increased further. We conclude that the exact choice of the number of steps $\ell$ is not very critical for the quality of the computed solution as long as it is not too small.

![Figure 10](image)

**Figure 10:** Example 2.3.10. Computed approximate solutions for the modified phillips test problem with noise-level $10^{-2}$. (a) Tikhonov solution $x_{95,\mu_{95}}$ with $B$ defined by (2). The curve determined with GSVD is on top of the curve obtained with GGKTR($\rho = 1$). (b) RRGMRES solution $x_{8}$.

<table>
<thead>
<tr>
<th>$B$</th>
<th>GGKTR($\rho = 1$)</th>
<th>GSVD</th>
<th>RRGMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidagonal</td>
<td>$6.80 \cdot 10^{-3}$ ($\ell = 57$)</td>
<td>$6.81 \cdot 10^{-3}$</td>
<td>$4.30 \cdot 10^{-2}$ ($\ell = 8$)</td>
</tr>
<tr>
<td>Tridiagonal</td>
<td>$4.64 \cdot 10^{-3}$ ($\ell = 95$)</td>
<td>$4.86 \cdot 10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Example 2.3.10. Relative error in computed solutions for noise-level $10^{-2}$. The parameter $\ell$ denotes the number of steps with GGKTR($\rho = 1$) and RRGMRES.

**Example 2.3.10.** Consider the Fredholm integral equation of the first kind

$$
\int_{-6}^{6} k(s, t) x(t) \, dt = g(s), \quad -6 \leq s \leq 6,
$$

(69)
discussed by Phillips [35]. Define the function

$$
\phi(x) = \begin{cases} 
1 + \cos\left(\frac{\pi x}{3}\right), & |s| < 3, \\
0, & |s| \geq 3.
\end{cases}
$$
The kernel $k$, the solution $x$, and the right-hand side $g$ are given by

\begin{align*}
k(s, t) &= \phi(s - t), \\
x(t) &= \phi(t), \\
g(s) &= (6 - |s|)(1 + \frac{1}{2} \cos(\frac{\pi s}{3})) + \frac{9}{2\pi} \sin(\frac{\pi |s|}{3}).
\end{align*}

We use the MATLAB code \texttt{phillips} from [25] to discretize (69). This yields the matrix $A \in \mathbb{R}^{1000 \times 1000}$ and a discretization of a scaled solution $x_0 \in \mathbb{R}^{1000}$. We add a discretization of the function $1 + \exp(\frac{(t+6)}{12})$, $t \in [-6, 6]$, to $x_0$, to obtain a vector $\hat{x}$ which represents a slowly oscillatory and increasing solution. The vector $\hat{b} := A\hat{x}$ yields “error-free” data to which we add a noise-vector $e$. The vector $e$ is chosen to correspond to the noise-levels $10^{-2}$ or $10^{-3}$. We apply GGKTR with $\rho = 1$ to determine approximations of the desired vector $\hat{x}$ using the regularization matrices (1) and (2). The number of steps is chosen to yield the best approximation of $\hat{x}$ and the regularization parameter is determined by the discrepancy principle. The results achieved are displayed by Tables 6 and 7 for the noise-levels $10^{-2}$ and $10^{-3}$, respectively. GGKTR($\rho = 1$) is seen to determine approximations of $\hat{x}$ that are of about the same quality as those obtained by GSVD. The regularization parameter for the latter method also is determined by the discrepancy principle. Figure 11(a) shows the computed solution for the noise-level $10^{-2}$ and the tridiagonal regularization matrix (2). The best approximation of $\hat{x}$ determined by RRGMRES is shown in Figure 11(b). This figure as well as the error in the best RRGMRES solution reported in Tables 6 and 7 illustrate the advantage of using Tikhonov regularization with the regularization matrix (2).

<table>
<thead>
<tr>
<th>$B$</th>
<th>GGKTR($\rho = 1$) $10^{-3}$</th>
<th>GSVD $10^{-3}$</th>
<th>RRGMRES $10^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidagonal</td>
<td>$3.65 \cdot 10^{-3}$ ((\ell = 84))</td>
<td>$3.64 \cdot 10^{-3}$</td>
<td>$7.62 \cdot 10^{-3}$ ((\ell = 11))</td>
</tr>
<tr>
<td>Tridiagonal</td>
<td>$1.12 \cdot 10^{-3}$ ((\ell = 100))</td>
<td>$1.13 \cdot 10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Example 2.3.10. Relative error in computed solutions for noise-level $10^{-3}$. The parameter $\ell$ denotes the number of steps with GGKTR($\rho = 1$) and RRGMRES.
Figure 12: Example 2.3.10. Computed approximate solutions for the modified phillips test problem with noise-level $10^{-2}$ and the regularization matrix (2). The figure differs from Figure 11(a) in that it shows the GGKTR solution $x_{27,\mu_{27}}$ determined after only $\ell = 27$ steps with GGKTR($\rho = 1$).

The number of steps $\ell$ carried out with GGKTR($\rho = 1$) for the results reported in Tables 6 and 7 is fairly large. However, it is not necessary to execute so many steps to achieve a useful approximation of $\hat{x}$. This is illustrated by Figure 12, which shows the error in the computed approximate solution $x_{27,\mu_{27}}$ computed by taking only $\ell = 27$ steps with GGKTR($\rho = 1$). The regularization parameter $\mu = \mu_{27}$ is determined by the discrepancy principle. The approximate solution $x_{27,\mu_{27}}$ is not as close to the exact solution $\hat{x}$ as the approximate solution $x_{95,\mu_{95}}$ of Figure 11(a), but is probably still useful in many contexts, and is faster to compute. $\Box$

The next two examples illustrate the use of GGKTR to the restoration of 2D gray-scale pictures which contain blur and noise.

**Example 2.3.11.** Consider the restoration of a blurred and noisy $384 \times 384$-pixel image. The desired noise- and blur-free image is displayed in Figure 13(a). We compare three regularization matrices $B$: the identity, a discretization of the Laplacian (50), and a
discretization of the Perona–Malik operator (51).

The regularization matrix $B$ in (14) plays an important role. The discretization of the Perona–Malik operator (51) often provides better restorations with sharper edges than the discretization of the Laplacian (50), which usually yields over-smoothed restorations. We determine the discretized Perona–Malik operator $B$ from the available image.

An image with blur and noise is generated with the MATLAB function `blur` from [25] with $\text{band} = 7$ and $\text{sigma} = 5$. The noise-level is $\delta = 10^{-1}$; see Figure 14(a). The best restorations for the regularization matrices $B = I$ and $B$ determined by (50) are generated with $\ell = 6$ and $\ell = 7$ steps of GGKTR($\rho = 1$) and have PSNR-values 25.82 and 25.68, respectively; see Figures 14(b) and (c). The best restoration with $B$ being a discretization of the Perona–Malik operator (51) is derived with $\ell = 15$ steps of GGKTR($\rho = 1$). It has PSNR-value 28.32 and is shown in Figure 14(d). The higher PSNR-value of the latter restoration is in agreement with visual perception. While the evaluation of matrix-vector products with $B$ is straightforward, the use of the inverse of $B$ is not. This makes it attractive to apply the GGKTR method. □

**Example 2.3.12.** We apply the regularization matrices obtained by discretizing the operators (50) and (51) to the $412 \times 412$-pixel image shown in Figure 13(b), the same
original image as in Example 2.2.9. We severely blur the image using \texttt{band} = 9 and \texttt{sigma} = 3 in the function \texttt{blur} from [25], and add 30\% Gaussian noise; see Figure 15(a).

The discretizations of the Laplace and Perona–Malik operators (50) and (51) are the same as in Example 2.3.11. For $B$ being the discrete Laplacian, the best restoration is obtained after $\ell = 5$ iterations with GGKTR($\rho = 1$); for $B$ being the discretized Perona–Malik operator, the most accurate restoration is obtained after $\ell = 7$ iterations. We get the PSNR-values 11.77 and 13.54, respectively, for the restored images; see Figures 15(b) and (c). We see the over-smoothing caused by the discretized Laplacian. Figure 15(d) shows an edge map for the image restored with $B$ being the discretized Perona–Malik operator.
Figure 15: Example 2.3.12. (a) Blurred and noisy image. (b) Image restored with $B$ being the discretized Laplacian. (c) Image restored with $B$ being the discretized Perona–Malik operator. (d) Edge map for the restoration in (c).

operator. The edge map determined with gimp is accurate also in the presence of the severe noise and blur present in the image. □

Example 2.3.13. Consider the test problem inverse Laplace transform as in Example 2.2.8. We use the MATLAB function iLaplace from [25] to determine a discretization $A \in \mathbb{R}^{1000 \times 1000}$ of the integral operator and a discretized scaled solution $\hat{x} \in \mathbb{R}^{1000}$. The error vector $e \in \mathbb{R}^{1000}$ has noise level $10^{-1}$. The error-contaminated data vector $b$ in (10) is defined by (11).
We will use the regularization matrix $B$ defined by

$$B = \begin{bmatrix} 0.5L_1 \\ 0.25L_2 \end{bmatrix},$$

where $L_1$ and $L_2$ are defined in (1) and (2). The matrix $B$ damps finite differences that approximate both the first and second derivative in the computed approximate solution.

We apply GGKTR with $\rho = 1$ and $\rho = 0.5$. The results are reported in Table 8. When $\rho = 1$, the best approximation of $\hat{x}$ is achieved after $\ell = 20$ iterations. If instead $\rho = 0.5$, then the best approximation of $\hat{x}$ is obtained after only $\ell = 13$ iterations. This example illustrates that GGKTR can yield approximate solutions of higher quality and require less storage and computational work when $\rho < 1$.

We do not at present have a complete understanding of how $\rho$ should be chosen, but note that it is important to choose a regularization matrix $B$ with an appropriate null space, because components of the solution $\mathbf{x}_\mu$ of (14) in $\mathcal{N}(B)$ are not damped. Illustrative examples on the importance of $\mathcal{N}(B)$ can be found in [15,16]. Letting $\rho < 1$ may yield a solution subspace that is closer to $\mathcal{N}(B)$ than when $\rho \geq 1$. This issue requires further investigation.

We also solve the Tikhonov minimization problem of this example with the aid of the GSVD in the following manner. First we determine the QR factorization $B = QR$, where $Q \in \mathbb{R}^{(2n-3)\times n}$ has orthonormal columns and $R \in \mathbb{R}^{n\times n}$ is upper triangular, and then we compute the GSVD of the matrix pair $\{A, R\}$. Table 8 shows this approach to yield the least accurate approximation of $\hat{x}$. Thus, it may be appropriate to use GGKTR also for problems that are small enough to allow the application of the GSVD. Figure 16 shows the desired solution $\hat{x}$ (black dash-dotted curve) and the approximation $\mathbf{x}_{13,\mu_{13}}$ computed by Algorithm GGKTR with $\rho = 0.5$ (red solid curve). They are very close. The figure also displays the approximate solution determined by the GSVD (blue dashed curve). □
Table 8: Example 2.3.13. Relative errors in computed approximate solutions for the noise level $10^{-1}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\rho$</th>
<th>$\ell$</th>
<th>$|\mathbf{x}_{\ell,\mu_1} - \hat{\mathbf{x}}|/|\hat{\mathbf{x}}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGKTR</td>
<td>1</td>
<td>20</td>
<td>$3.71 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>GGKTR</td>
<td>0.5</td>
<td>13</td>
<td>$3.16 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>GSVD</td>
<td></td>
<td></td>
<td>$1.16 \cdot 10^{-1}$</td>
</tr>
</tbody>
</table>

Figure 16: Example 2.3.13. The red solid curve displays the computed approximate solution $\mathbf{x}_{13,\mu_{13}}$ determined by GGKTR with $\rho = 1/2$, the blue dashed curve shows the solution computed via GSVD, and the black dash-dotted curve depicts the desired solution $\hat{\mathbf{x}}$. 
2.4 A decomposition method based on flexible Arnoldi reduction

Many commonly used regularization matrices $B \in \mathbb{R}^{p \times n}$ are rectangular with either $p < n$ or $p > n$. The reduction method for matrix pairs $\{A, B\}$ described in Section 2.2, which is based on the flexible Arnoldi method, requires both the matrix $A$ and $B$ to be square. This section describes a simple modification of Algorithm 2.2.1 that allows $B$ to be rectangular. Differently from the method in Algorithm 2.2.1, the method in this section requires the evaluation of matrix-vector products with $B$ and $B^*$. 
2.4.1 Algorithm based on Arnoldi reduction

Algorithm 2.4.1. Reduction of matrix pair \(\{A, B\} \); \(A\) square, \(B\) rectangular.

1. Input: \(A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{p \times n}, b \in \mathbb{R}^n\), ratio \(\rho \geq 0\), and number of steps \(\ell\)

2. \(h_{1,1}^{(A)} := \|b\|; \ u_1 := b/h_{1,1}^{(A)}; \ v_1 := u_1\)

3. \(N(u) := 1; \ N(w) := 1\)

4. for \(j = 1, 2, \ldots, \ell\) do

5. \(\tilde{u} := Av_j\)

6. for \(i = 1, 2, \ldots, j\) do

7. \(h_{i,j}^{(A)} := u_i^T \tilde{u}; \ \tilde{u} := \tilde{u} - u_i h_{i,j}^{(A)}\)

8. end for

9. \(h_{j+1,j}^{(A)} := \|\tilde{u}\|\)

10. \(u_{j+1} := \tilde{u}/h_{j+1,j}\)

11. \(\tilde{w} := Bv_j\)

12. for \(i = 1, 2, \ldots, j - 1\) do

13. \(h_{i,j}^{(B)} := w_i^T \tilde{w}; \ \tilde{w} := \tilde{w} - w_i h_{i,j}^{(B)}\)

14. end for

15. \(h_{j,j}^{(B)} := \|\tilde{w}\|\)

16. \(w_j := \tilde{w}/h_{j,j}^{(B)}\)

17. if \(N(w)/N(u) > 1/\rho\)

18. \(N(u) := N(u) + 1; \ v := u_{N(u)}\)

19. else

20. \(v := B^* w_{N(u)}; \ N(w) := N(w) + 1\)

21. end

22. for \(i = 1, 2, \ldots, j\) do

23. \(v := v - (v_i^T v)v_i;\)

24. end for

25. \(\alpha_j := \|v\|;\)

26. \(v_{j+1} := v/\alpha_j;\)

27. end for
The elements $h^{(A)}_{i,j}$ and $h^{(B)}_{i,j}$ in the above algorithm are the nontrivial entries of the matrices $H^{(A)}_{\ell+1,\ell}$ and $H^{(B)}_{\ell,\ell}$ determined by the algorithm. Algorithm 2.4.1 differs from the flexible Arnoldi reduction Algorithm 2.2.1 only in that line 20 has been changed from $v := w_{N(w)}$ to $v := B^* w_{N(w)}$. Most of the properties of Algorithm 2.2.1 carry over to Algorithm 2.4.1. The structure of the matrix $H^{(B)}$ determined by Algorithm 2.4.1 is similar to the structure of the matrix $H^{(B)}$ computed by Algorithm 2.3.2. Moreover, if the matrix $A$ in Algorithm 2.4.1 is symmetric, then the structure of the computed matrix $H^{(A)}$ is analogous to the structure of the matrix $H^{(A)}$ determined by Algorithm 2.3.2. These properties about the structure of $H^{(A)}$ and $H^{(B)}$ can be shown similarly as the results for the matrices $H^{(A)}$ and $H^{(B)}$ in Section 2.3.

2.4.2 Numerical examples

Example 2.4.1. Consider the test problem baart as in Example 2.2.6. We use the MATLAB function baart from [25] to discretize (45). This yields the matrix $A \in \mathbb{R}^{1000 \times 1000}$ and the “exact” solution $\hat{x} \in \mathbb{R}^{1000}$, with which we compute the error-free right-hand side $\hat{b} := A\hat{x}$. The error vector $e \in \mathbb{R}^{1000}$ corresponds to the noise level $\delta = 1 \cdot 10^{-2}$. The data vector $b$ in (10) is obtained from (11).

We seek to determine an approximation of $\hat{x}$ by using a decomposition determined by Algorithm 2.4.1. The regularization matrix $L_2$ is defined by (2). This approach is compared to Algorithm 2.2.1. The latter algorithm requires the regularization matrix to be square. We therefore pad the regularization matrix (2) with two zero rows, see (44). We also compute an approximate solution by using the GSVD of the matrix pair $\{A, L_2\}$.

The results are listed in Table 9. Both FATR and Algorithm 2.4.1 yield better approximations of $\hat{x}$ than the GSVD. The best approximation of $\hat{x}$ is determined by Algorithm
2.4.1 with $\rho = 0.5$; the relative error is $6.58 \cdot 10^{-3}$. This approximate solution is shown in Figure 17 (red solid curve). The figure also displays $\hat{x}$ (black dash-dotted curve), and the GSVD solution (blue solid curve). Both Algorithm 2.4.1 and FATR yield approximations of $\hat{x}$ of higher quality when $\rho = 1/2$ than when $\rho = 1$. We therefore only show results for $\rho = 1/2$. □

![Figure 17: Example 2.4.1. Approximate solution $x_{26,\mu_{26}}$ determined by Algorithm 2.4.1 with $\rho = 1/2$ with noise level $10^{-3}$ (red solid curve), approximate solution computed with GSVD (blue dashed curve), and desired solution $\hat{x}$ (black dash-dotted curve)](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>$\rho$</th>
<th>$\ell$</th>
<th>$|x_{\ell,\mu_{\ell}} - \hat{x}|/|\hat{x}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATR</td>
<td>0.5</td>
<td>16</td>
<td>$9.44 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>Algorithm 2.4.1</td>
<td>0.5</td>
<td>26</td>
<td>$6.58 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>GSVD</td>
<td></td>
<td></td>
<td>$2.76 \cdot 10^{-2}$</td>
</tr>
</tbody>
</table>

Table 9: Example 2.4.1. Relative errors in computed approximate solutions for the noise level $10^{-3}$. The parameter $\ell$ denotes the number of steps with the Algorithms 2.4.1 and with FATR.

**Example 2.4.2.** Our last example illustrates the performance of Algorithm 2.4.1 when
applied to the restoration of a two-dimensional gray-scale image that has been contaminated by blur and noise. The gray-scale image *rice* from MATLAB’s Image Processing Toolbox is represented by an array of 256 × 256 pixels. We generate the blur and noise image $b$ as described in Section 2.2.4 with noise-level $\delta = 10^{-2}$. The parameters for the function *blur* are $\text{band} = 9$ and $\text{sigma} = 2$. Our task is to restore the image $b$. The desired image $\hat{x}$ and the blur- and noise-contaminated image $b$ are shown in Figures 18 and 19. We assume the blurring matrix $A$, the contaminated image $b \in \mathbb{R}^n$, and the noise level $\delta$ to be available. Our task is to determine an accurate approximation of the blur- and noise-free image $\hat{x}$.

Figure 18: Example 2.4.2. Exact image.

Let the regularization matrix $B$ be defined by

$$ B = \begin{bmatrix} I \otimes L_1 \\ L_1 \otimes I \end{bmatrix}, $$

(70)

where $L_1$ is given by (1) with $n = 256$. The matrix $B \in \mathbb{R}^{130560 \times 65536}$ has about twice the number of rows as columns. Therefore, we cannot use FATR. This regularization matrix also is used in [11].
Figure 19: Example 2.4.2. Blur- and noise-contaminated image.

Table 10 reports results achieved with Algorithm 2.4.1 for several values of the parameter $\rho$. For each $\rho$, we carry out 30 iterations and select the approximation in the 30-dimensional solution subspace with the largest PSNR value. The restoration with the largest PSNR value is determined by Algorithm 2.4.1 with $\rho = 0.1$ and is displayed by Figure 20. We see that the best restoration is achieved with the smallest number of iterations.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\rho$</th>
<th>$\ell$</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 2.4.1</td>
<td>1</td>
<td>25</td>
<td>28.213</td>
</tr>
<tr>
<td>Algorithm 2.4.1</td>
<td>0.5</td>
<td>27</td>
<td>28.222</td>
</tr>
<tr>
<td>Algorithm 2.4.1</td>
<td>0.2</td>
<td>22</td>
<td>28.233</td>
</tr>
<tr>
<td>Algorithm 2.4.1</td>
<td>0.1</td>
<td>22</td>
<td>28.297</td>
</tr>
</tbody>
</table>

Table 10: Example 2.4.2. PSNR-values of restorations computed by Algorithm 2.4.1 with $B$ defined by (70).

This example illustrates that Algorithm 2.4.1 with $B$ given by (70) can yield quite accurate restorations with only 3 matrix-vector product evaluations with the matrix $A$. 
2.5 Conclusion

We described three generalized Krylov subspace methods for reducing a pair large matrices to a pair of small matrices. The large matrices are accessed via matrix-vector product evaluations only. An extension of these methods for the reduction of a \((q + 1)\)-tuplet of large matrices to a \((q + 1)\)-tuplet of small matrices also is presented.

The relation between the large and reduced matrices makes our reduction methods well suited for application to one-parameter and multi-parameter Tikhonov regularization of linear discrete ill-posed problems with large, possibly rectangular, matrices. The main reason for developing the one-parameter Tikhonov regularization method of the present paper is to obtain a new alternative to the GSVD when the problems are so large that the latter is expensive or infeasible to apply. We note that the numerical examples of one-parameter Tikhonov regularization show that for problems that are small enough
to allow the evaluation of the GSVD of the matrix pair, our reduction methods and the GSVD yield computed approximate solutions of the same or higher accuracy. Our reduction methods therefore are attractive alternatives to GSVD also when the matrices are small enough for the latter to be applicable.
CHAPTER 3

Rational Arnoldi and pseudospectrum computation

The rational Arnoldi process is a popular method for the computation of a few eigenvalues of a large non-Hermitian matrix $A \in \mathbb{C}^{n \times n}$ and for the approximation of matrix functions. The method is particularly attractive when the rational functions that determine the process have few distinct poles $z_j \in \mathbb{C}$, because then only few factorizations of matrices of the form $A - z_j I$ have to be computed. We discuss recursion relations for orthogonal bases of rational Krylov subspaces determined by a few distinct poles. An application of the rational Arnoldi process to the computation of the pseudospectrum of $A$ is described. In this application, the poles are chosen in regions of the complex plane where we would like to determine the pseudospectrum. We discuss how the computed pseudospectrum relates to the pseudospectrum of $A$.

3.1 Introduction

Let $A \in \mathbb{C}^{n \times n}$ be a large, possibly sparse, non-Hermitian matrix, and let $v \in \mathbb{C}^n$ be a unit vector. Throughout this chapter, except for in Section 3.2, we use the Euclidean vector norm, which we denote by $\| \cdot \|$. The standard inner product of vectors in $\mathbb{C}^n$ is written as $\langle \cdot, \cdot \rangle$. Thus, $\| w \| = \langle w, w \rangle^{1/2}$ for $w \in \mathbb{C}^n$. The rational Arnoldi process determines orthonormal rational functions with prescribed poles with respect to the sesquilinear form

$$\langle f, g \rangle := v^*(f(A))^*g(A)v,$$  (71)
where the superscript * denotes transposition and, if applicable, complex conjugation. The expression (71) is an inner product of functions $f$ and $g$, when the functions live in a suitably restricted set. We are interested in the situation when they are rational functions with fixed poles of low enough order so that (71) is an inner product. The exact order allowed depends on the matrix $A$ and vector $v$. Throughout this paper, the latter will be of unit Euclidean norm. We note that the inner product (71) also can be represented as

$$\langle f, g \rangle = \frac{1}{4\pi^2} \int_{\Gamma} \int_{\Gamma} \overline{f(z_1)g(z_2)}v^*(\overline{z_1}I - A^*)^{-1}(z_2I - A)^{-1}v d\overline{z}_1 dz_2,$$

(72)

where the contour of integration $\Gamma$ contains the spectrum of $A$ in its interior and the bar denotes complex conjugation; see, e.g., [36] for discussions of related representations.

The rational Arnoldi process was introduced by Ruhe [37] for the computation of selected eigenvalues and eigenvectors of a large matrix; see also [38–40]. Applications of the rational Arnoldi or Lanczos processes to the approximation of matrix functions are described in [36, 41–46].
Algorithm 3.1.1. THE RATIONAL ARNOLDI PROCESS.

Input: $A \in \mathbb{C}^{n \times n}$, $v \in \mathbb{C}^n$ initial unit vector, shifts $z_i \in \mathbb{C}$, $i = 1, 2, \ldots, m$.

for $i = 1, 2, \ldots, m$ do

Apply one step of rational Arnoldi with shift $z_i$. This gives

$$w := (A - z_i I)^{-1} v - \sum_{j=1}^i h_{j,i} v_j;$$

$$h_{i+1,i} := ||w||; \quad v_{i+1} := w / h_{i+1,i};$$

where the coefficients $h_{j,i}$ are chosen so that the vector $w$ is orthogonal to $v_j$ for $1 \leq j \leq i$. When $z_i = \infty$, we replace the matrix $(A - z_i I)^{-1}$ by $A$.

end

We assume that the coefficients $h_{i+1,i} > 0$ in (73) for $i = 1, 2, \ldots, m$. This is the generic situation. The computations simplify in the rare case when some $h_{i+1,i}$ vanishes. We will not dwell on this situation. Define the matrices $V_m = [v_1, v_2, \ldots, v_m] \in \mathbb{C}^{n \times m}$ and $V_{m+1} = [V_m, v_{m+1}] \in \mathbb{C}^{n \times (m+1)}$, as well as the upper Hessenberg matrix $H_{m+1,m} = [h_{i,j}] \in \mathbb{C}^{(m+1) \times m}$. Thus, the columns of $V_m$ form an orthonormal basis for the rational Krylov subspace

$$\text{span}\{v, (A - z_1 I)^{-1} v, (A - z_2 I)^{-1}(A - z_1 I)^{-1} v, \ldots, \prod_{i=1}^{m-1} (A - z_i I)^{-1} v\},$$

where each factor $(A - z_i I)^{-1}$ with $z_i = \infty$ is replaced by $A$.

When all the shifts $z_i$ are the same, the recursion (73) simplifies to the standard Arnoldi process for the matrix $(A - z_m I)^{-1}$ and can be expressed in the form

$$(A - z_m I)^{-1} V_m = V_{m+1} H_{m+1,m}.$$  \hspace{1cm} (74)

Ruhe has shown how the decomposition (73) can be updated when not all shifts $z_i$ are the same, so that also in this situation we obtain after $m$ steps a relation of the form (74).
with an upper Hessenberg matrix $H_{m+1,m} \in \mathbb{C}^{(m+1)\times m}$. The entries of this matrix differ from the elements $h_{j,i}$ determined in (73); see, e.g., [40] for details. The application of Ruhe’s updating method is particularly attractive when many consecutive shifts are the same.

We will discuss the structure of the matrix analogous to $H_{m+1,m}$ in (74) when the updating formulas by Ruhe are not applied. The matrix, therefore, is not upper Hessenberg. Consecutive shifts do not have to be the same; however, the structure of the matrix analogous to $H_{m+1,m}$ is most useful when there are only few distinct shifts. We show that the analogue of $H_{m+1,m}$ in (74) is upper Hessenberg-like with a left half-bandwidth, whose size depends on the number of distinct shifts and on the order in which the shifts are applied. We are particularly interested in the situation when the sequence of shifts $z_1, z_2, \ldots, z_m$ is a cyclic enumeration of a few distinct points. The structure of the matrix analogous to $H_{m+1,m}$ is discussed in Section 3.2.

We consider the application of the rational Arnoldi method to the computation of the $\varepsilon$-pseudospectrum of a large matrix. Let $\varepsilon > 0$ be fixed. The $\varepsilon$-pseudospectrum of the matrix $A$ is the set

$$\Lambda_\varepsilon(A) := \{ z \in \mathbb{C} : \sigma_{\min}(A - zI) \leq \varepsilon \},$$

where $\sigma_{\min}(M)$ denotes the smallest singular value of the matrix $M$. Thus, the spectrum of $A$ is a subset of $\Lambda_\varepsilon(A)$. An insightful discussion of properties and many applications of the $\varepsilon$-pseudospectrum is presented by Trefethen and Embree [47].

The computation of the $\varepsilon$-pseudospectrum for large a matrix is very expensive. Therefore Toh, Trefethen, and Wright [48–50] discuss how the standard Arnoldi process and
ARPACK [51], which is based on this process, can be used to reduce the matrix $A$ to a small Hessenberg matrix $H$ and an approximation of the $\varepsilon$-pseudospectrum of $A$ can be determined by computing the $\varepsilon$-pseudospectrum of $H$. This approach may require the evaluation of a fairly large number of matrix-vector products with $A$ in order to be able to determine the $\varepsilon$-pseudospectrum of $A$ in selected parts of the complex plane with desired accuracy. It follows that this approach to computing an approximation of the $\varepsilon$-pseudospectrum of $A$ also can be expensive. We are therefore interested in investigating the performance of the rational Arnoldi method in this application.

This chapter is organized as follows. Section 3.2 describes the rational Arnoldi method when applied to the orthonormalization of a sequence of rational functions with a few distinct poles. In Section 3.3, we discuss the rational Arnoldi method in a linear algebra context. A detailed algorithm is presented and the structure of the matrix with recursion coefficients is further explored. In particular, we show that the subdiagonal part of this matrix has a block structure. Section 3.4 describes an application of the rational Arnoldi process to the computation of the $\varepsilon$-pseudospectrum of a large matrix $A$ in selected regions of the complex plane. This process is able to determine an accurate approximation of an invariant subspace associated with eigenvalues close to the selected poles $z_i$. A few computed examples are presented in Section 3.5 and concluding remarks can be found in Section 3.6.

3.2 A rational Arnoldi method for function approximation

This section discusses the recursion formulas for the Arnoldi method when applied to the orthonormalization of a sequence of rational functions with a few fixed distinct poles.
We proceed similarly as in [52], where the situation when the inner product is defined by a nonnegative measure on the real axis and the poles are real or appear in complex conjugate pairs is considered. Thus, let $d\mu$ be a nonnegative measure in $\mathbb{C}$ with infinitely many points of support. In this section, we use the inner product

$$\langle f, g \rangle = \int \overline{f(z)}g(z)d\mu(z),$$

and the induced norm $\|f\| := \langle f, f \rangle^{1/2}$. The measure is supposed to be such that the inner product exists for all polynomial $f$ and $g$. Let $\mathcal{P}$ denote the space of all polynomials and introduce the linear space of rational functions with finite poles $z_1, z_2, z_3, \ldots$ bounded away from the support of $d\mu$,

$$Q = \text{span} \left\{ \frac{1}{(z-z_k)^j} : j \in \mathbb{N}, z_k \in \mathbb{C}, |z_k| < \infty \right\},$$

where as usual $\mathbb{N}$ denotes the set of positive integers. Define the linear space

$$\mathcal{P} + Q = \text{span}\{1, z^j, \frac{1}{(z-z_k)^j} : j \in \mathbb{N}, z_k \in \mathbb{C}, |z_k| < \infty \}.$$

Let $\Psi = \{\psi_0, \psi_1, \psi_2, \ldots \}$ denote an elementary basis for this space, i.e., $\psi_0(z) = 1$ and each $\psi_\ell(z)$ for $\ell = 1, 2, \ldots$ is one of the functions

$$z^j, \quad \frac{1}{(z-z_k)^j},$$

for some positive integers $j$ and $k$. Application of the Gram–Schmidt process with respect to the inner product (75) to the basis $\Psi$ yields a basis of orthonormal rational functions with the prescribed finite poles $z_1, z_2, z_3, \ldots$,

$$\Phi = \{\phi_0, \phi_1, \phi_2, \ldots \}.$$
We here assume that no pole $z_k$ is an eigenvalue of $A$ and that (72) is an inner product for the functions considered.

The recursion relations for the $\phi_j$ depend on the ordering of the basis functions $\psi_j$ of $\Psi$. We write $\psi_j \prec \psi_k$ if the basis function $\psi_j$ comes before $\psi_k$. The ordering of the basis $\Psi$ is said to be natural if it satisfies the following conditions:

1. $z^j \prec z^{j+1}$ for all integers $j \geq 0$.
2. $\frac{1}{(z-z_k)^j} \prec \frac{1}{(z-z_k)^{j+1}}$ for all positive integers $j$ and every pole $z_k$.

We tacitly assume that there are no missing powers, i.e., if $z^{j+1}$ is an elementary basis function, then so is $z^j$. Similarly, if $(z-z_k)^{-j-1}$ is an elementary basis function, then so is $(z-z_k)^{-j}$.

We now will show that orthonormal rational functions with prescribed poles corresponding to any natural ordering satisfy two types of recurrence relations.

**Theorem 3.2.1.** Let the basis $\Psi = \{\psi_0, \psi_1, \psi_2, \ldots \}$ satisfy the requirements of natural ordering and assume that every sequence of $m_1$ consecutive basis functions $\psi_k, \psi_{k+1}, \ldots, \psi_{k+m_1-1}$ contains at least one power $z^\ell$. Then the orthonormal rational functions $\phi_0, \phi_1, \phi_2, \ldots$ with prescribed poles satisfy a recurrence relation of the form

$$z\phi_k(z) = \sum_{i=0}^{k+m_1} c_{k,i} \phi_i(z), \quad k = 0, 1, 2, \ldots .$$

(76)

**Proof.** It follows from the ordering that

$$z\phi_k(z) \in \text{span}\{\phi_0, \phi_1, \ldots, \phi_{k+m_1-1}, \phi_{k+m_1}\}, \quad k = 0, 1, 2, \ldots .$$

This shows (76).

**Remark 3.2.2.** If $Q$ is empty, i.e., if there are no finite poles, then $m_1 = 1$, and we obtain the standard Arnoldi process. □
Remark 3.2.3. If \( Q \) is not empty, i.e., if there are finite poles, then we can order the basis \( \Psi \) to get the smallest possible value of \( m_1 \). If there are sufficiently many positive powers \( z^j, j \in \mathbb{N} \), among the elementary basis functions, then the elementary basis functions can be ordered so that \( m_1 = 2 \). □

**Theorem 3.2.4.** Let the basis \( \Psi = \{\psi_0, \psi_1, \psi_2, \ldots \} \) satisfy the requirements of natural ordering and let \( z_{\ell} \) be a finite pole of this basis. Assume that every sequence of \( m_2 \) consecutive basis functions \( \psi_k, \psi_{k+1}, \ldots, \psi_{k+m_2-1} \) contains at least one power \( (z - z_{\ell})^{-t} \), \( t \geq 1 \). Then the basis of orthonormal rational functions \( \phi_0, \phi_1, \phi_2, \ldots \) with prescribed poles satisfy a recurrence relation of the form

\[
\frac{1}{z - z_{\ell}} \phi_k(z) = \sum_{i=0}^{k+m_2} c_{k,i}^{(t)} \phi_i(z), \quad k = 0, 1, 2, \ldots.
\]

**Proof.** The result can be shown similarly as Theorem 3.2.1. □

We conclude this section with an Arnoldi-type algorithm for generating an orthonormal basis \( \phi_0, \phi_1, \phi_2, \ldots \) for a subspace for \( \Psi = \{\psi_0, \psi_1, \psi_2, \ldots \} \). We say that the orthonormal function \( \phi_m \) contains the elementary basis function \( \psi_j \) if \( \psi_j \in \text{span}\{\phi_0, \phi_1, \ldots, \phi_m\} \).

**Algorithm 3.2.1. An Arnoldi-type procedure.**

\[
\phi_0 := \psi_0/\|\psi_0\|; \quad k := 1;
\]

while \( k \leq n \) do

if \( \psi_k(z) = z^s \) for some \( s \in \mathbb{N} \) then

if \( s = 1 \) then

\( p := -1; m := 0; \)

else

let \( m \leq k - 1 \) be (the smallest) integer such that \( \phi_m \) contains \( z^{s-1} \);

let \( p \) be (the largest) integer such that

\( \forall j \leq p, \quad z\phi_j(z) \in \text{span}\{\phi_0, \ldots, \phi_{m-1}\}; \)

end if
\begin{align*}
g(z) &:= z\phi_m(z); \\
\text{for } i = p + 1, \ldots, k - 1 \text{ do} \\
c_{k-1,i} &:= \langle g, \phi_i \rangle; \quad g := g - c_{k-1,i}\phi_i; \\
\text{end for} \\
\phi_k &:= g/\|g\|; \quad k := k + 1.; \\
\end{align*}

if \psi_k(z) = (z - z_\ell)^{-s} for some \ s \in \mathbb{N} then

if \ s = 1 \text{ then} \\
p := -1; \quad m := 0; \\
else \\
\text{let } m \leq k - 1 \text{ be (the smallest) integer such that } \phi_m \text{ contains} \\
(z - z_\ell)^{-(s-1)}; \\
\text{let } p \text{ be (the largest) integer such that} \\
\forall j \leq p, \ (z - z_\ell)^{-1}\phi_j(z) \in \text{span}\{\phi_0, \ldots, \phi_{m-1}\}; \\
\text{end if} \\
g(z) := (z - z_\ell)^{-1}\phi_m(z); \\
\text{for } i = p + 1, \ldots, k - 1 \text{ do} \\
c_{k-1,i} &:= \langle g, \phi_i \rangle; \quad g := g - c_{k-1,i}\phi_i; \\
\text{end for} \\
\phi_k &:= g/\|g\|; \quad k := k + 1.; \\
\text{end if}
\end{align*}

3.3 The rational Arnoldi method in $\mathbb{C}^n$

This section discusses in detail the structure of the matrix $H_{m+1,m}$ in the decomposition (74). In this section $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{C}^n$ and $\| \cdot \|$ stands for the associated vector norm. Given a matrix $A \in \mathbb{C}^{n \times n}$, an initial unit vector $v \in \mathbb{C}^n$, two finite poles $z_1, z_2 \in \mathbb{C}$, and a desired dimension $m + 1 = 3p + 2$ of the rational
Krylov subspace, Algorithm 3.3.1 below generates a decomposition of the form (74). The columns of the matrix \( V_{m+1} = [v_1, v_2, \ldots, v_{m+1}] \in \mathbb{C}^{n \times (m+1)} \) form an orthonormal basis for the subspace

\[
\text{span}\{v, Av, (A - z_1 I)^{-1}v, (A - z_2 I)^{-1}v, A^2b, (A - z_1 I)^{-2}v, (A - z_2 I)^{-2}v, \cdots\} \quad (77)
\]
determined by rational functions in \( A \) with poles in the set \( \{\infty, z_1, z_2\} \) times the vector \( v \).

The nontrivial entries of the Hessenberg-type matrix \( H_{m+1,m} \in \mathbb{C}^{(m+1) \times m} \) are recursion coefficients for the orthonormal rational functions in \( A \) that form a basis for the subspace (77). Algorithm 3.3.1 easily can be modified to allow any number of prescribed poles. We let \( m_i \) denote the \( i \)th column of the matrix \( M \) with entries \( m_{i,j} \), \( M_i \) stands for the sub-matrix made up of first \( i \) columns of \( M \), and \( M_{i,j} \) denotes the leading principal \( i \times j \) submatrix of \( M \). Similarly, \( w_i \) denotes the \( i \)th element of the vector \( w \). Finally, the range of the matrix \( M \) is denoted by \( \mathcal{R}(M) \).

**Algorithm 3.3.1. A three-pole rational Arnoldi method.**

**Input:** \( A \in \mathbb{C}^{n \times n}, \ v \in \mathbb{C}^n \) initial unit vector, finite shifts \( z_1, z_2 \), termination index \( m = 3p + 1 \).

**Output:** \( V_{m+1} \in \mathbb{C}^{n \times (m+1)} \) matrix with orthonormal columns that span the rational Krylov subspace, \( H_{m+1,m} \in \mathbb{C}^{(m+1) \times m} \) Hessenberg-type matrix.

1. Generate orthonormal basis for rational Krylov subspace

\[
\text{span}\{v, Av, (A - z_1 I)^{-1}v, (A - z_2 I)^{-1}v\}.
\]

\[
v_1 := v; \ w := Av_1; \ h_{1,1} := (v_1, w);
\]

\[
w := w - h_{1,1}v_1; \ h_{2,1} := \|w\|; \ v_2 := w/h_{2,1};
\]

for \( i = 1, 2 \) do

\[
w := (A - z_i I)^{-1}v; \quad \text{% Solve linear system of equations for } w.
\]

for \( j = 1, \ldots, i + 1 \) do

\[
w := w - (v_j, w)v_j;
\]

end for
\[ v_{i+2} := w/\|w\|; \]

end for

2. Augment rational Krylov subspace by a new positive power of \( A \), new negative powers of \( A - z_1 I \) and \( A - z_2 I \), and a new positive power of \( A \) in order.

\[ k := 2; \]

while \( k < m \)

\[ a) \ Add \ new \ vector \ Av_k \ to \ rational \ Krylov \ subspace. \]

\[ w := Av_k; \]

for \( j = 1, 2, \ldots, k + 2 \) do

\[ h_{j,k} := \langle v_j, w \rangle; w := w - h_{j,k}v_j; \]

end for

\[ h_{k+3,k} := \|w\|; v_{k+3} := w/h_{k+3,k}; \]

\[ b) \ Only \ update \ the \ matrix \ H \ with \ recursion \ coefficients. \]

for \( i = 1, 2 \) do

\[ w := Av_{k+i}; \]

for \( j = 1, 2, \ldots, k + 3 \) do

\[ h_{j,k+i} := \langle v_j, w \rangle; w := w - h_{j,k+i}v_j; \]

end for

end for

\[ c) \ Augment \ rational \ Krylov \ subspace \ by \ vectors \ (A - z_i I)^{-1}v_{k+i} \ for \ i = 1, 2. \]

for \( i = 1, 2 \) do

\[ w := (A - z_i I)^{-1}v_{k+i}; \quad \% \ Solve \ linear \ system \ of \ equations \ for \ w. \]

for \( j = 1, 2, \ldots, k + i + 1 \) do

\[ w := w - \langle v_j, w \rangle v_j; \]

end for

\[ v_{k+i+2} := w/\|w\|; \]

end for

\[ k := k + 3; \]

end while
Algorithm 3.3.1 is an application of Theorem 3.2.1. We comment on some details:

Step 1: The algorithm generates the matrix \( V_4 = [v_1, v_2, v_3, v_4] \), whose columns form an orthonormal basis for
\[
\text{span}\{v, Av, (A - z_1I)^{-1}v, (A - z_2I)^{-1}v\}.
\]
Also the matrix \( H_{2,1} = \begin{bmatrix} h_{1,1} \\ h_{1,2} \end{bmatrix} \) is determined. It satisfies
\[
AV_1 = V_2H_{2,1}.
\]
The last two columns of \( V_4 \) are not part of this decomposition. The column \( v_3 \) is generated by orthogonalizing \((A - z_1I)^{-1}v\) against \( v_1 \) and \( v_2 \). Therefore \( v_3 \) is a linear combination of \( v_1, v_2, \) and \((A - z_1I)^{-1}v\). We have
\[
v_3 = V_2[\begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}] + \gamma_3(A - z_1I)^{-1}v \tag{78}
\]
for certain coefficients \( \gamma_i \). We assume here the generic situation that
\((A - z_1I)^{-1}v \notin \text{span}\{v_1, v_2\} \). Similarly, \( v_4 \) is determined by orthogonalizing \((A - z_2I)^{-1}v\) against \( v_1, v_2, v_3 \) and normalizing the residual obtained.

Step 2: We discuss the computations during the first pass through the while-loop, i.e., when \( k = 2 \). The algorithm augments the matrix \( V_4 \) with three columns to obtain an orthonormal basis for the rational Krylov subspace
\[
\text{span}\{v, Av, (A - z_1I)^{-1}v, (A - z_2I)^{-1}v, A^2v, (A - z_1I)^{-2}v, (A - z_2I)^{-2}v\}
\]
and recursion coefficients stored in the matrix \( H \). We carry out the following computations:
a) Evaluate $A\mathbf{v}_2$ and orthogonalize against the columns of $V_1$. Assign the normalized residual to $\mathbf{v}_5$. The coefficients generated during orthogonalization and normalization are stored in the second column of the matrix $H_{5,2}$. This gives the decomposition

$$AV_2 = V_5 H_{5,2}, \quad H_{5,2} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \\ 0 & h_{3,2} \\ 0 & h_{4,2} \\ 0 & h_{5,2} \end{bmatrix}.$$ 

Thus,

$$\mathcal{R}(AV_2) \subset \mathcal{R}(V_5) = \text{span}\{\mathbf{v}, A\mathbf{v}, (A - z_1I)^{-1}\mathbf{v}, (A - z_2I)^{-1}\mathbf{v}, A^2\mathbf{v}\}.$$ 

b) Evaluate $A\mathbf{v}_3$ and orthogonalize against the columns of $V_5$. The following discussion shows that the orthogonalization process will break down and, therefore, no new columns of the matrix $V_5$ will be generated. It follows from (78) that

$$A\mathbf{v}_3 = AV_2[\gamma_2] + \gamma_3 A(A - z_1I)^{-1}\mathbf{v}, \quad (79)$$

where, by the discussion above, the first term in the right-hand side lives in $\mathcal{R}(V_5)$. The vector $A(A - z_1I)^{-1}\mathbf{v}$ is a linear combination of $\mathbf{v}$ and $(A - z_1I)^{-1}\mathbf{v}$. Therefore the second term in the right-hand side of (79) lives in $\mathcal{R}(V_3)$. In particular, $A\mathbf{v}_3 \in \mathcal{R}(V_5)$. A similar argument shows that $A\mathbf{v}_4 \in \mathcal{R}(V_5)$. When part b) of Step 2 of Algorithm 3.3.1 with $k = 2$ is completed, we have the decomposition

$$AV_4 = V_5 H_{5,4}, \quad H_{5,4} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} \\ 0 & h_{4,2} & h_{4,3} & h_{4,4} \\ 0 & h_{5,2} & h_{5,3} & h_{5,4} \end{bmatrix}.$$ 

(80)
c) Evaluate \((A - z_1 I)^{-1}v_3\) and orthogonalize against the columns of \(V_5\). Assign the normalized residual to \(v_6\). This defines \(V_6\). Similarly, compute \((A - z_2 I)^{-1}v_3\) and orthogonalize against the columns of \(V_6\). Store the normalized residual in \(v_7\). This yields \(V_7\). We have to show that

\[
\mathcal{R}(V_6) = \text{span}\{v, Av, (A - z_1 I)^{-1}v, (A - z_2 I)^{-1}v, A^2v, (A - z_1 I)^{-2}v\}.
\]

By (78), we have

\[
(A - z_1 I)^{-1}v_3 = (A - z_1 I)^{-1}V_2[\gamma_1 \gamma_2] + \gamma_3(A - z_1 I)^{-2}v.
\] (81)

The first term on the right-hand side of (81) is in \(\mathcal{R}(V_5)\). This shows that the subspace \(\mathcal{R}(V_6) = \mathcal{R}(V_5) \cup \{(A - z_1 I)^{-1}v_3\}\) can be expressed as \(\mathcal{R}(V_5) \cup \{(A - z_1 I)^{-2}v\}\). One can show similarly that

\[
\mathcal{R}(V_7) = \mathcal{R}(V_6) \cup \text{span}\{(A - z_2 I)^{-2}v\}.
\]

The vectors \(v_6\) and \(v_7\) will be part of the decomposition determined when \(k = 5\); see below.

We briefly comment on the second pass through the while-loop of Algorithm 3.3.1. At the beginning of the execution of the loop, the decomposition (80) is available as well as the matrix \(V_7 = [v_1, v_2, \ldots, v_7]\) with orthonormal columns. In part a) of Step 2, we express \(Av_5\) as a linear combination of the columns of \(V_7\) and assign the normalized residual to \(v_8\). This yields the decomposition

\[
AV_5 = V_8 H_{8,5}, \quad H_{8,5} = \begin{bmatrix}
  h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} \\
  h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} \\
  0 & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} \\
  0 & h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} \\
  0 & h_{5,2} & h_{5,3} & h_{5,4} & h_{5,5} \\
  0 & 0 & 0 & 0 & h_{6,5} \\
  0 & 0 & 0 & 0 & h_{7,5} \\
  0 & 0 & 0 & 0 & h_{8,5}
\end{bmatrix}.
\]
In Step b), we determine columns 6 and 7 of $H_{8,7}$. Due to breakdown in the orthogonalization process, we obtain the decomposition

$$AV_7 = V_8 H_{8,7}, \quad H_{8,7} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} & h_{1,6} & h_{1,7} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} & h_{2,6} & h_{2,7} \\ 0 & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} & h_{3,6} & h_{3,7} \\ 0 & h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} & h_{4,7} \\ 0 & h_{5,2} & h_{5,3} & h_{5,4} & h_{5,5} & h_{5,6} & h_{5,7} \\ 0 & 0 & 0 & h_{6,5} & h_{6,6} & h_{6,7} \\ 0 & 0 & 0 & h_{7,5} & h_{7,6} & h_{7,7} \\ 0 & 0 & 0 & h_{8,5} & h_{8,6} & h_{8,7} \end{bmatrix}. \quad (82)$$

It is straightforward to adapt Algorithm 3.3.1 to the situation when the rational Krylov subspace is defined by only one finite pole and a pole at infinity. Then the three $i$-loops are only executed for $i = 1$ instead of for $i = 1$ and $i = 2$. We conclude this section with some comments on how the algorithm can be modified to handle the situation when the rational Krylov subspace is determined by two finite distinct poles $z_1$ and $z_2$, without a pole at infinity. Consider

$$(A - z_2 I)^{-1} = ((A - z_1 I) - (z_2 - z_1)I)^{-1} = \frac{1}{z_1 - z_2} (A - z_1 I)^{-1}((A - z_1 I)^{-1} - \frac{1}{z_2 - z_1} I)^{-1}$$

and let

$$S = (A - z_1 I)^{-1} \quad (83)$$

and $\gamma = \frac{1}{z_2 - z_1}$. Then

$$(A - z_2 I)^{-1} = (\gamma) S(S - \gamma I)^{-1}.$$ 

This change of variables allows us to apply the adaption of Algorithm 3.3.1 to the situation when the rational Krylov subspace is defined by only one finite pole and a pole at infinity.
3.4 Application to pseudospectrum computation

We seek to determine an invariant subspace associated with eigenvalues of $A$ of interest with the rational Arnoldi process. The following result shows that these subspaces determine subsets of pseudospectra of $A$.

**Proposition 3.4.1.** Let the columns of the matrix $W \in \mathbb{C}^{n \times \ell}$ with $\ell \leq n$ form an orthonormal basis for an invariant subspace of the matrix $A \in \mathbb{C}^{n \times n}$. Then $AW = WM$ for some matrix $M \in \mathbb{R}^{\ell \times \ell}$ and

$$\Lambda_\varepsilon(M) \subseteq \Lambda_\varepsilon(A)$$

for all $\varepsilon \geq 0$.

**Proof.** The relation (84) follows from

$$\sigma_{\min}(A-zI) \leq \sigma_{\min}((A-zI)W) = \sigma_{\min}(W(M-zI)) = \sigma_{\min}(M-zI).$$

$\square$

The following special case of the above proposition is of particular interest.

**Corollary 3.4.2.** Let $\lambda$ be an eigenvalue of $A$. The disk

$$D_\varepsilon(\lambda) = \{z \in \mathbb{C} : |z - \lambda| \leq \varepsilon\}$$

is a subset of $\Lambda_\varepsilon(A)$ for any $\varepsilon \geq 0$.

**Proof.** Let $w \in \mathbb{C}^n$ be an eigenvector of $A$ of unit length associated with the eigenvalue $\lambda$ and assume that $z \in D_\varepsilon(\lambda)$. Then we obtain similarly as in the proof of Proposition 3.4.1 that $\sigma_{\min}(A-zI) \leq |z - \lambda|$ and the desired result follows. $\square$

We turn to the application of Algorithm 3.3.1 to the computation of pseudospectra of the matrix $A$. When a standard Arnoldi decomposition

$$AV_m = V_{m+1}H_{m+1,m}$$
is available, where the orthonormal columns of $V_{m+1}$ form a basis for the (standard) Krylov subspace $K_{m+1}(A, v) = \text{span}\{v, Av, \ldots, A^m v\}$ and $H_{m+1,m} \in \mathbb{C}^{(m+1)\times m}$ is an upper Hessenberg matrix, then Wright and Trefethen [49] drop the last row of $H_{m+1,m}$ to obtain the square matrix $H_{m,m} \in \mathbb{C}^{m\times m}$ and use the $\varepsilon$-pseudospectrum of $H_{m,m}$ as an approximation of the $\varepsilon$-pseudospectrum of $A$; see also [47] for discussions on the approximation of pseudospectra of a large matrix.

We apply decompositions determined with Algorithm 3.3.1 in an analogous fashion. Thus, assume that the decomposition (82) is available. We then drop the last row of $H_{8,7}$ to obtain the square matrix $H_{7,7} \in \mathbb{C}^{7\times 7}$ and use the $\varepsilon$-pseudospectrum of $H_{7,7}$ as an approximation of the $\varepsilon$-pseudospectrum of $A$.

When we only have one finite pole and one pole at infinity, and apply the transformation (83), we have to transform the problem back before computing the $\varepsilon$-pseudospectrum. Specifically, we have, for some $m > 0$,

$$SV_m = V_{m+1}H_{m+1,m} \approx V_mH_{m,m}.$$ 

Substituting (83) into the left-hand side yields after some manipulations

$$AV_m \approx V_m(H_{m,m}^{-1} + z_1 I).$$ 

We therefore use the $\varepsilon$-pseudospectra of $H_{m,m}^{-1} + z_1 I$ as approximations of the $\varepsilon$-pseudospectra of $A$.

3.5 Numerical Examples

This section presents several numerical results that compare the application of the standard Arnoldi method with Algorithm 3.3.1, and this algorithm modified to only
have two finite distinct poles and no pole at infinity when applied to approximate $\varepsilon$-pseudospectra of a given matrix $A$. Since we would like to compare the approximations of $\varepsilon$-pseudospectra to the exact ones, the matrix $A$ in some examples is not large. The initial vector $v$ in the standard and rational Krylov subspaces has in all examples normally distributed entries with mean zero. The vector is normalized to have unit length. All computations were carried out in MATLAB with machine epsilon about $2 \cdot 10^{-16}$ on an Acer computer with a 64-bit operating system, an i5-3210M CPU, and 4GB of memory.

Example 3.5.1. Let

$$A = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 \\
-1 & 1 & 1 & 1 & 1 \\
-1 & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
-1 & 1 & 1 & 1 & 1 \\
0 & \cdots & \cdots & \cdots & -1 \\
\end{bmatrix} \in \mathbb{R}^{100 \times 100}$$

be a Grcar matrix. Grcar matrices are banded nonsymmetric Toeplitz matrices with $-1$ on the subdiagonal and 1 on the diagonal and three superdiagonals. We choose the finite complex poles $z_1 = 0.0862 + 2.5i$ and $z_2 = 1.8 + 0.65i$, where $i := \sqrt{-1}$. Standard and rational Krylov subspaces of dimension 30 are computed. This yields Hessenberg or Hessenberg-like matrices $H_{30,30}$ as described above. We evaluate $\sigma_{\min}(H_{30,30} - zI)$ on a $64 \times 64$ Cartesian grid with equidistant grid points. The level curves of Figure 21 show the $\varepsilon$-pseudospectra for several levels $\varepsilon$. The power of 10 of the level is marked on the level curve. Table 11 displays the CPU time in seconds for the different approaches.
Figure 21: Example 3.5.1. $\varepsilon$-pseudospectra determined from $A$ and by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$.

Figure 21(a) shows $\varepsilon$-pseudospectra determined by evaluating $\sigma_{\min}(A - zI)$ at the grid points. We consider these pseudospectra exact and compare them to the pseudospectra determined by standard and rational Arnoldi methods. Figure 21(b) displays the pseudospectra determined by projecting the problem by the standard Arnoldi method. The graphs show the $\varepsilon$-pseudospectrum to be inaccurate for small values of $\varepsilon$. We found that the standard Arnoldi method requires Krylov subspaces of high dimension to determine level curves for $\varepsilon$-pseudospectra for small values of $\varepsilon$. The computation of a high-dimensional Krylov subspace may not be attractive when the matrix $A$ is large.
<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact $\varepsilon$-pseudospectra</td>
<td>17.0</td>
</tr>
<tr>
<td>Standard Arnoldi</td>
<td>1.40</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $\infty$, $z_1$, $z_2$</td>
<td>1.68</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $z_1$, $z_2$</td>
<td>1.80</td>
</tr>
</tbody>
</table>

Table 11: Example 3.5.1. Comparison of CPU times in seconds.

However, the level curves of $\varepsilon$-pseudospectra of $A$ for larger values of $\varepsilon$ can be determined accurately with a few steps of the standard Arnoldi method; see, for instance, the 0-level curve in Figure 21(b). Figure 21(c) is determined with Algorithm 3.3.1 with poles at $\infty$, $z_2 = 0.0862 + 2.5i$, and $z_1 = 1.8 + 0.65i$. The levels obtained when only the two finite poles $z_1$ and $z_2$ are used in the rational Arnoldi method are shown in Figure 21(d). Figures 21(c) and (d) illustrate that rational Arnoldi decompositions may determine the $\varepsilon$-pseudospectra accurately in a vicinity of the poles. Comparing Figures 21(c) and (d), we see that the rational Arnoldi method with only two finite poles (and no pole at infinity) yields more accurate $\varepsilon$-pseudospectra in the vicinity of the finite poles than Algorithm 3.3.1, while the latter determines more accurate level curves for $\varepsilon$-pseudospectra for larger values of $\varepsilon$, but these levels curves are not as accurate as when the standard Arnoldi method is used. The CPU time (in seconds) is shown in Table 11. For this example, the major part of the CPU time is spent on computing the smallest singular value of the matrices $A - zI$ for Figure 21(a) or of the matrices $H_{30,30} - zI$ for the standard and rational Arnoldi methods when $z$ traverses the grid points. The difference in timings for the full and projected problems would be even larger if $A$ were of larger dimensions.

The advantage of projecting the matrix $A$ onto the smaller matrices $H_{30,30}$ is apparent, in particular when one only requires knowledge of $\varepsilon$-pseudospectra around specific points in the complex plane. Table 11 shows the use of two finite distinct poles to be more expensive than when using these poles and the pole at infinity. The reason for this
is that the former approach requires more solutions of linear systems of equations to determine a rational Krylov subspace of desired dimension than the latter, in which some system solves are replaced by matrix-vector product evaluations with $A$.

**Example 3.5.2.** We apply standard and rational Arnoldi methods to a medium-sized Grcar matrix $A \in \mathbb{R}^{300 \times 300}$. Let the unit random vector $v \in \mathbb{R}^{300}$ be constructed similarly as in Example 3.5.1. Choose the finite poles $z_1 = -0.15 + 2.6i$ and $z_2 = 2.2 + 1i$ for Algorithm 3.3.1 and for the rational Arnoldi method with only two poles. We determine projected matrices $H_{50,50}$ by these rational Krylov methods and evaluate $\sigma_{\min}(A - zI)$ and $\sigma_{\min}(H_{50,50} - zI)$ at the grid points of a Cartesian grid with $32 \times 32$ grid points.

Comparing with Example 3.5.1, we find that the level curves for $\varepsilon$-pseudospectra of $A$ for corresponding levels $\varepsilon$ are smaller in this example. To determine these levels with rational Arnoldi methods, we had to choose the poles close enough to the eigenvalues of $A$ to achieve desired accuracy, but far away enough from the eigenvalues to avoid to get matrices $H_{50,50}$ of numerical very low rank. If $H_{50,50}$ is of very low rank, say of rank one, then this matrix does not provide much useful information about the pseudospectra of $A$. We propose to first determine approximations of the $\varepsilon$-pseudospectra of $A$ by reducing $A$ by the standard Arnoldi method, and then allocating poles in regions in the complex plane that are deemed to be interesting, and apply a rational Arnoldi method with these poles to gain further insight into $\varepsilon$-pseudospectra of $A$. Table 12 displays CPU times in seconds for the methods in our comparison.

Table 12: Example 3.5.2. Comparison of CPU times in seconds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact $\varepsilon$-pseudospectra</td>
<td>56.0</td>
</tr>
<tr>
<td>Standard Arnoldi</td>
<td>1.02</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at (\infty, z_1, z_2)</td>
<td>1.11</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $z_1, z_2$</td>
<td>1.42</td>
</tr>
</tbody>
</table>

The following two examples illustrate the performance of the rational Arnoldi method
(a) Exact $\varepsilon$-pseudospectra.

(b) Standard Arnoldi.

(c) Rational Arnoldi with three distinct poles at $\infty$, $z_1$, and $z_2$.

(d) Rational Arnoldi with two distinct poles at $z_1$ and $z_2$.

Figure 22: Example 3.5.2. $\varepsilon$-pseudospectra determined from $A$ and by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$.

when applied to large matrices.

**Example 3.5.3.** Let $A \in \mathbb{R}^{2000 \times 2000}$ be a Grcar matrix and choose the finite poles $z_1 = 2.789 + 1i$ and $z_2 = 2.789 + 0.8i$. We generate Hessenberg and Hessenberg-like matrices $H_{50,50}$ with the standard and and rational Arnoldi methods and evaluate $\sigma_{\text{min}}(H_{50,50} - zI)$ at the grid points of a $64 \times 64$ Cartesian grid. Comparing the level curves determined by applying Algorithm 3.3.1 with the poles $\infty$, $z_1$, $z_2$ and the variant of the algorithm using
only the poles $z_1$ and $z_2$ shows the latter to determine more reasonable level curves. We omit the computation of exact $\varepsilon$-pseudospectra of $A$, because this is too time consuming.

\[
\begin{array}{c}
\text{(a) Standard Arnoldi.} \\
\text{(b) Rational Arnoldi with three distinct poles at } \infty, z_1, \text{ and } z_2. \\
\text{(c) Rational Arnoldi with two distinct poles } z_1 \text{ and } z_2.
\end{array}
\]

Figure 23: Example 3.5.3. $\varepsilon$-pseudospectra determined by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of $A$. 
<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Arnoldi</td>
<td>4.29</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $\infty, z_1, z_2$</td>
<td>8.89</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $z_1, z_2$</td>
<td>14.2</td>
</tr>
</tbody>
</table>

**Example 3.5.4.** Let $A$ be the tridiagonal nonsymmetric Toeplitz matrix

$$
A = \begin{bmatrix}
0 & 2 & 0 & \cdots & \cdots & \cdots \\
\frac{1}{4} & 0 & 2 & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{1}{4} & 0 & 2 & \cdots & \cdots & \cdots \\
0 & \frac{1}{4} & 0 & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix} \in \mathbb{R}^{2000 \times 2000}.
$$

Many properties of tridiagonal Toeplitz matrices, including their spectra and pseudospectra, are discussed in [53]. We choose the finite poles $z_1 = -2 + 0.8i$ and $z_2 = -1.9 + 0.9i$ and generate Hessenberg and Hessenberg-type matrices $H_{50,50}$ by the standard Arnoldi method, Algorithm 3.3.1, and the modification of the latter that allows the use of only two finite poles and no pole at infinity. The smallest singular value of the matrices $H_{50,50} - zI$ are computed for $z$ at the grid points of a $64 \times 64$ Cartesian grid. Computed $\varepsilon$-pseudospectra are displayed in Figure 24, which is analogous to Figure 23. Timings are reported in Table 14. □

**Table 14:** Example 3.5.4. Comparison of CPU times in seconds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Arnoldi</td>
<td>4.33</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $\infty, z_1, z_2$</td>
<td>8.62</td>
</tr>
<tr>
<td>Rational Arnoldi, distinct poles at $z_1, z_2$</td>
<td>14.0</td>
</tr>
</tbody>
</table>
Figure 24: Example 3.5.4. \( \varepsilon \)-pseudospectra determined by standard and rational Arnoldi methods. The red dots in the figure are eigenvalues of \( A \).

3.6 Conclusion

We explore the structure of the Hessenberg-type matrices generated by the rational Arnoldi method with a few distinct poles in the extended complex plane. This results in a new implementation that allows the use of the poles in a cyclic fashion in an efficient manner. An application to the computation of \( \varepsilon \)-pseudospectra of a large matrix in regions in the complex plane is described.
CHAPTER 4

Conclusion

In this work, we develop several methods for reducing computational problems with large matrices to problem with small matrices. The reduction methods depend on the problem to be solved. We discuss the solution of large-scale Tikhonov regularization problems as well as the computation of the $\varepsilon$-pseudospectrum of a large matrix. Several reduction methods for Tikhonov regularization problems are presented. Some methods are suitable for the situation when the given large matrices are square, other methods can be applied to the reduction of large rectangular matrices. The reduction methods proposed are generalized Krylov subspace methods. The methods extend available Krylov subspace methods for the reduction of one square or rectangular matrix. In the context of $\varepsilon$-pseudospectrum computations, we propose a new rational Arnoldi method that is well suited for the situation when the rational functions involved have few distinct poles that are applied in a cyclic fashion.
BIBLIOGRAPHY


