

A STUDY OF THE S-STEP BICONJUGATE GRADIENT METHOD

CARLOTTA BOI

Bachelor's thesis
2022:K14



LUND UNIVERSITY

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

LUND UNIVERSITY

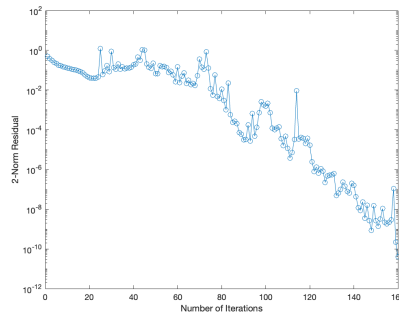
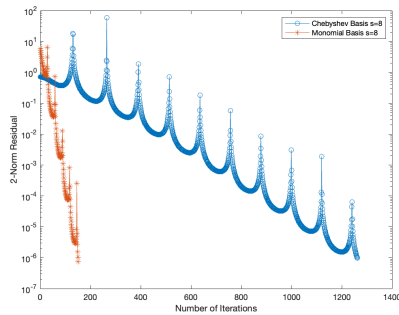
NUMK11

BACHELOR'S PROJECT IN NUMERICAL ANALYSIS

A study of the s-step biconjugate gradient method

Author: Carlotta BOI

Supervisor: Philipp BIRKEN



0.1 Abstract

In this thesis we will examine how to solve linear systems using the s-step biconjugate gradient algorithm, which is an iterative method based on the Krylov subspaces. It is useful especially when we have a large and sparse matrix. We begin looking over the biconjugate gradient algorithm (BiCG), in order to understand how to construct the s-step BiCG algorithm. We will go through some numerical examples to see which method can give a better numerical solution and which one is able to converge. At the end we will talk about finite precision arithmetic and study roundoff errors of the s-step BiCG method.

0.2 Popular scientific abstract

Iterative algorithms are important methods to make of solutions for systems of linear equations. They do it by creating a succession of approximate solutions which can drive the user to a solution that can be closer to the exact one. These methods are valuable in different fields of science, for instance materials science and statistics. One of the most known iterative techniques are the Krylov subspace methods (KSMs). This thesis focuses on an algorithm based on the KSMs, named s-step biconjugate method, which is very useful especially for decreasing the communication costs caused by exchanging information among different levels of computer storage and among different devices. But this comes with a price: as we increment the s number for minimizing the price of transferring information, we can experience side effects like the decrease of precision of the solution computed by the algorithm, or the increase of the number of iterations for arriving at a solution. In this thesis we will explore these side effects and compare our results to another iterative technique named biconjugate gradient method, which is the technique used for building the s-step method.

0.3 Keywords

biconjugate gradient methods, s-step biconjugate gradient methods, nonsymmetric linear systems, sparse matrices.

0.4 Acknowledgment

I'm really thankful to my supervisor Philipp Birken for guiding me through this project and for the time he spent on helping me.

Contents

0.1	Abstract	2
0.2	Popular scientific abstract	2
0.3	Keywords	2
0.4	Acknowledgment	2
1	Introduction	5
2	BiCG and s-step BiCG algorithms	7
2.1	BiCG algorithm	7
2.2	From BiCG method to s-step BiCG method	8
2.3	S-step BiCG method	16
2.4	Bases	17
3	Numerical experiments	21
4	S-step BiCG technique and finite precision	33
4.1	Revise of s-step BiCG technique	33
4.2	Finite precision arithmetic	37
5	Summary	45
6	MATLAB codes	47
	References	71

Chapter 1

Introduction

Linear systems with large matrices are solved making use of iterative methods. These methods start with an initial guess solution named x_0 and improve it until some conditions given by the user are reached [2]. One iterative method which is also one of the “Top 10 Algorithms” of the 20th century is the Krylov subspaces method [6]. Consider a matrix A and a vector v , it is possible to define the Krylov subspace as:

$$K_{s+1} = \text{span}(v, Av, \dots, A^s v). \quad (1.1)$$

Here $s \in \mathbb{Z}^+$ [9]. Krylov subspaces methods (KSMS) generate bases for Krylov subspaces through the multiplications between the matrix and the vector [9]. The KSMS can have communication costs, i.e. costs of “the movement of data between levels of memory hierarchy or between processors over a network” [2]. In order to decrease them, it is possible to divide the loop of the KSMS and create two loops: an inside loop and an outside loop. In the inside loop the algorithm will perform s iterations at once, while the outside loop moves from s iterations to other s iterations until some conditions are met [2]. These algorithms are called either “ s -step KSMS”, or “communication-avoiding KSMS” [2]. At first, only the Krylov (monomial) bases were used, but as s becomes a large value, it was noticed that it was possible that the methods could not converge [2]. In order to find a solution to that problem, Joubert and Carey, with the use of the Chebyshev polynomials, constructed a basis that it was possible to utilize also for larger s [10], [2]. One iterative algorithm built on the KSMS is the biconjugate gradient (BiCG) method, it was first discovered by Lanczos in 1952 and after more than 20 years, in 1976, it was utilized again by Fletcher [14]. Starting from the BiCG algorithm, it is possible to create the s -step BiCG algorithm, which is an iterative algorithm to make of solutions for “nonsymmetric linear systems” [2], based on the s -step KSMS. As the user increment the value of s , for decreasing the cost of transferring information, it is possible to experience some side effects of the s -step technique, as for instance the increase of the number of iterations for reaching convergence or the decrease of precision of the results [2].

We will investigate these side effects and look over if the biconjugate gradient algorithm can give better results.

The thesis concentrates on the s -step biconjugate gradient algorithm. Chapter 2 starts with a review of the biconjugate gradient method and after that it focuses on how to construct the s -step BiCG algorithm starting from the BiCG one and using the Krylov subspaces. In the s -step BiCG method it is possible to use different bases, which are based on the Krylov subspaces. The ones that we will analyse are the monomial basis and the Chebyshev basis. We will study how to construct the Chebyshev basis using Chebyshev polynomials and through the help of the spectrum of a matrix A of a linear system and an ellipse. Chapter 3 is focused on numerical experiments. Its are presented two examples, where both bases are used for the s -step BiCG method. We will use sparse matrices, which are matrices that have the majority of values equal to zero [17]. What we will discover is that the monomial basis is a good option when s is small value, but as s becomes bigger the Chebyshev basis seems to be a better choice. To compare the results of the BiCG and the s -step BiCG methods we will look at how many iterations are needed to meet the conditions given by the user. The number of iterations is affected by the “round-off error in finite precision” [2], these errors are the difference between the exact value of a number and the value computed by the computer [11]. Because of that in the last chapter we will look over “the s -step biconjugate gradient algorithm in finite precision arithmetic” [4] and see that the computation of the Krylov bases can generate roundoff errors.

This thesis is based on the technical report written by Erin Carson and James Demmel: “Analysis of the finite precision s -step biconjugate gradient method” [4], on the research paper “Avoiding Communication In Nonsymmetric Lanczos - Based Krylov Subspace methods” [5] written by Erin Carson, Nicholas Knight, James Demmel, and on chapters 1,2,3,4,5 of the PhD thesis: “Communication-Avoiding Krylov Subspace Methods in Theory and Practice” [2], written by Erin Carson.

Chapter 2

BiCG and s-step BiCG algorithms

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a sparse, nonsymmetric matrix and \mathbf{b} a n -dimensional vector. We want to solve a nonsymmetric linear system of equations $Ax = b$ making use of iterative methods. In this chapter we will study the BiCG and the s-step BiCG algorithms, with the use of the Krylov basis and the Chebyshev basis.

2.1 BiCG algorithm

This section is based on the research paper written by Charles H. Tong and Qiang Ye [13] and on lecture 38 of the book [14]. Consider a nonsymmetric matrix $A \in \mathbb{R}^{n \times n}$ and a vector b of length n . The biconjugate gradient method (BiCG) is an iterative technique to construct solutions for “nonsymmetric linear systems” $Ax = b$ [9]. It starts by initializing a vector x_0 , which is our initial guess solution, then we define the vectors r_0, \tilde{r}_0 which are named residuals and p_0, \tilde{p}_0 which are called search directions. We create a loop, with index $m \in \mathbb{Z}$, $m > 0$, which constructs a succession of vectors $\{x_m\}, \{r_m\}, \{\tilde{r}_m\}, \{p_m\}, \{\tilde{p}_m\}$. The m loop works until the conditions given by the user are met.

The following algorithm is taken from the work by Tong and Ye [13], it is nearly the same, we change just some notation and the for loop becomes a while loop for us.

BiCG Algorithm

Our inputs are x_0, A, b

1. $x_0 = [0, \dots, 0]$, $m = 1$
2. $r_0 = p_0 = \tilde{r}_0 = \tilde{p}_0 = b - Ax_0$
3. $\rho_0 = \tilde{r}_0^T r_0$
4. *while m until convergence*
5. $\sigma_{m-1} = \tilde{p}_{m-1}^T A p_{m-1}$
6. $\alpha_{m-1} = \rho_{m-1} / \sigma_{m-1}$
7. $r_m = r_{m-1} - \alpha_{m-1} A p_{m-1}$

8. $x_m = x_{m-1} + \alpha_{m-1}p_{m-1}$
9. $\tilde{r}_m = \tilde{r}_{m-1} - \alpha_{m-1}A^T\tilde{p}_{m-1}$
10. $\rho_m = \tilde{r}_m^T r_m$
11. $\beta_m = \rho_m / \rho_{m-1}$
12. $p_m = r_m + \beta_m p_{m-1}$
13. $\tilde{p}_m = \tilde{r}_m + \beta_m \tilde{p}_{m-1}$
14. $m=m+1$
18. *end while*

The residuals are biorthogonals, which means that satisfy the following property, for $n \neq m$:

$$\tilde{r}_m^T r_n = 0. \quad (2.1)$$

The search directions are A-biconjugates, that means that satisfy the following property, for $n \neq m$:

$$\tilde{p}_m^T A p_n = 0.$$

From (2.1) it follows that r_m is perpendicular to the following Krylov subspace:

$$K_m(A^T, r_0) = \text{span}\{r_0, A^T r_0, \dots, (A^{m-1})^T r_0\}.$$

The method introduced in this section will experience a collapse if one of the two following situations happens [13]:

- (1) $\sigma_m = 0$, “pivotal breakdown”,
- (2) $\rho_m = 0$, “breakdown in the underlying Lanczos process”.

2.2 From BiCG method to s-step BiCG method

This section is based on the research paper written by Carson, Knight and Demmel [5] and on [2].

Our goal is to construct the s-step BiCG method beginning from the BiCG method. In order to create the s-step BiCG algorithm we need two loops: an outside loop, for which we will use the index k , and an inside loop, which goes from $j = 1$ to $j = s$. We will utilize the index m for indicating $m = sk + j$ [3].

Lemma 1. Assume that $A \in \mathbb{R}^{n \times n}$ is a matrix and assume that $r_m, p_m, x_m, \tilde{r}_m, \tilde{p}_m$ are the vectors in the BiCG algorithm. Then, they can be written as a linear combination of the Krylov basis [5]:

$$\begin{aligned}
 p_m &\in K_{j+1}(A, p_{sk}) + K_j(A, r_{sk}), \\
 r_m &\in K_{j+1}(A, p_{sk}) + K_j(A, r_{sk}), \\
 \tilde{r}_m &\in K_{j+1}(A^T, \tilde{p}_{sk}) + K_j(A^T, \tilde{r}_{sk}), \\
 \tilde{p}_m &\in K_{j+1}(A^T, \tilde{p}_{sk}) + K_j(A^T, \tilde{r}_{sk}), \\
 x_m - x_{sk} &\in K_j(A, p_{sk}) + K_{j-1}(A, r_{sk}).
 \end{aligned} \quad (2.2)$$

This can be proven by induction on rows $\{7, 8, 9, 12, 13\}$ of the BiCG algorithm.

We will show the proof for p_m , it is very similar for the other vectors.

Proof. Basis step. For $m = 1$, ($m = sk + j$, $sk = 0$ and $j = 1$), line (12), $p_1 = r_1 + \beta_1 p_0$ should satisfy:

$$p_1 \in K_2(A, p_0) + K_1(A, r_0). \quad (2.3)$$

Using line (12) and (7) of the BiCG algorithm we see:

$$\begin{aligned} p_1 &= r_1 + \beta_1 p_0 \\ &= (r_0 - \alpha_0 A p_0) + \beta_1 p_0 \\ &= \beta_1 p_0 - \alpha_0 A p_0 + r_0. \end{aligned}$$

Following the definition of Krylov subspaces, we can write p_1 in this way:

$$\begin{aligned} p_1 &\in K_2(A, p_0) + K_1(A, r_0) \\ &= a_0 p_0 + a_1 A p_0 + b_0 r_0. \end{aligned}$$

Here $a_0, a_1, b_0 \in \mathbb{R}$. So for $a_0 = \beta_1$, $a_1 = -\alpha_0$ and $b_0 = 1$, we can affirm that line (12) satisfies (2.3).

Hypothesis step. We assume that it is true for $sk = n - 1$, $j = 1$, so $m = n$, and $p_n = r_n + \beta_n p_{n-1}$ should satisfy:

$$p_n \in K_2(A, p_{n-1}) + K_1(A, r_{n-1}). \quad (2.4)$$

Considering line (12) and (7), it follows that it possible to write p_n as:

$$\begin{aligned} p_n &= r_n + \beta_n p_{n-1} \\ &= (r_{n-1} - \alpha_{n-1} A p_{n-1}) + \beta_n p_{n-1} \\ &= \beta_n p_{n-1} - \alpha_{n-1} A p_{n-1} + r_{n-1}. \end{aligned}$$

Using the definition of Krylov subspaces, we can write p_n as:

$$\begin{aligned} p_n &\in K_2(A, p_{n-1}) + K_1(A, r_{n-1}) \\ &= u_0 p_{n-1} + u_1 A p_{n-1} + v_0 r_{n-1}. \end{aligned}$$

Here $u_0, u_1, v_0 \in \mathbb{R}$. For $u_0 = \beta_n$, $u_1 = -\alpha_{n-1}$ and $v_0 = 1$, we assume that line (12), with $m=n$, satisfies (2.4).

Inductive Step. For $sk = n$, $j = 1$, so $m = n + 1$, it follows that: $p_{n+1} = r_{n+1} + \beta_{n+1} p_n$ should satisfy:

$$p_{n+1} \in K_2(A, p_n) + K_1(A, r_n). \quad (2.5)$$

Using line (12) and (7) and by the hypothesis step, we can write p_{n+1} as:

$$\begin{aligned} p_{n+1} &= r_{n+1} + \beta_{n+1} p_n \\ &= (r_n - \alpha_n A p_n) + \beta_{n+1} p_n \\ &= \beta_{n+1} p_n - \alpha_n A p_n + r_n. \end{aligned}$$

By the definition of Krylov subspaces, we can write p_{n+1} also as:

$$\begin{aligned} p_{n+1} &\in K_2(A, p_n) + K_1(A, r_n) \\ &= w_0 p_n + w_1 A p_n + z_0 r_n. \end{aligned}$$

Here $w_0, w_1, z_0 \in \mathbb{R}$. For $w_0 = \beta_{n+1}$, $w_1 = -\alpha_n$, $z_0 = 1$ and using the hypothesis step, line (12) satisfies (2.5). \square

Lemma 2. Assume that $A \in \mathbb{R}^{n \times n}$ is a matrix and that $p_m, r_m, \tilde{r}_m, \tilde{p}_m, x_m$ are the vectors given in the BiCG algorithm. Assume $s > 0$ and $j \leq s$. Then the vectors satisfy [5]:

$$\begin{aligned} p_{sk+j}, r_{sk+j} &\in K_{s+1}(A, p_{sk}) + K_s(A, r_{sk}), \\ \tilde{p}_{sk+j}, \tilde{r}_{sk+j} &\in K_{s+1}(A^T, \tilde{p}_{sk}) + K_s(A^T, \tilde{r}_{sk}) \\ x_{sk+j} - x_{sk} &\in K_s(A, p_{sk}) + K_{s-1}(A, r_{sk}). \end{aligned}$$

We present the proof by induction, using (2.2) and using the propriety of the Krylov subspaces: $K_1(A, v) \subseteq K_2(A, v) \subseteq \dots \subseteq K_{s+1}(A, v)$, where v is a vector. We will prove it for p_{sk+j} as the proof for the other vectors is very similar.

Proof. Basis case. For $sk = 0$ and $j = 1$, $p_1 = r_1 + \beta_1 p_0$ should satisfy:

$$p_1 \in K_2(A, p_0) + K_1(A, r_0). \quad (2.6)$$

Consider line (12) and (7) of the BiCG algorithm, so we can write p_1 as:

$$\begin{aligned} p_1 &= r_1 + \beta_1 p_0 \\ &= (r_0 - \alpha_0 A p_0) + \beta_1 p_0 \\ &= \beta_1 p_0 - \alpha_0 A p_0 + r_0. \end{aligned}$$

Using the definition of Krylov subspace we write p_1 as:

$$\begin{aligned} p_1 &\in K_2(A, p_0) + K_1(A, r_0) \\ &= a_0 p_0 + a_1 A p_0 + b_0 r_0. \end{aligned}$$

Here $a_0, a_1, b_0 \in \mathbb{R}$. For $a_0 = \beta_1$, $a_1 = -\alpha_0$ and $b_0 = 1$, we can affirm that line (12) satisfies (2.6).

Hypothesis step. We assume that for $sk = 0$, $j = s - 1$:

$$p_{s-1} = r_{s-1} + \beta_{s-1} p_{s-2}.$$

satisfies

$$\begin{aligned} p_{s-1} &\in K_s(A, p_0) + K_{s-1}(A, r_0) \\ &= c_0 p_0 + c_1 A p_0 + c_2 A A p_0 + \dots + c_{s-1} A^{s-1} p_0 + d_0 r_0 + d_1 A r_0 + \dots + d_{s-2} A^{s-2} r_0, \end{aligned}$$

so we can write:

$$\begin{aligned}
p_{s-1} &= r_{s-1} + \beta_{s-1}p_{s-2} \\
&= r_{s-2} - \alpha_{s-2}Ap_{s-2} + \beta_{s-1}p_{s-2} \\
&= \dots = d_0r_0 + d_1Ar_0 + d_2AAr_0 + \dots d_{s-2}A^{s-2}r_0 + c_0p_0 + c_1Ap_0 + \dots + c_{s-1}A^{s-1}p_0 \\
&\in K_s(A, p_0) + K_{s-1}(A, r_0).
\end{aligned}$$

Here $d_0, \dots, d_{s-2}, c_0, \dots, c_{s-1} \in \mathbb{R}$.

Inductive Step. For $sk = 0, j = s$ we have to show that

$$p_s = r_s + \beta_s p_{s-1}$$

satisfies

$$p_s \in K_{s+1}(A, p_0) + K_s(A, r_0). \quad (2.7)$$

Using the hypothesis step, it follows that:

$$\begin{aligned}
p_s &= r_s + \beta_s p_{s-1} \\
&= r_{s-1} - \alpha_{s-1}Ap_{s-1} + \beta_s p_{s-1} \\
&= d_0r_0 + d_1Ar_0 + d_2AAr_0 + \dots + d_{s-1}A^{s-1}r_0 + c_0p_0 + c_1Ap_0 + \dots + c_sA^s p_0 \in K_{s+1}(A, p_0) + K_s(A, r_0).
\end{aligned}$$

Here $d_0, \dots, d_{s-1}, c_0, \dots, c_s \in \mathbb{R}$. So line (12) satisfies (2.7). \square

To construct the vectors that iterate from $sk+1$ to $sk+s$, i.e. with $j = 1, \dots, s$, in the s-step BiCG algorithm, we create the following Krylov matrices [4],[5],[2]:

$$\begin{aligned}
V_k^p &= [v_{k,0}^p, v_{k,1}^p, \dots, v_{k,s}^p], & \text{span}(V_k^p) &= K_{j+1}(A, p_{sk}), \\
V_k^r &= [v_{k,0}^r, \dots, v_{k,s-1}^r], & \text{span}(V_k^r) &= K_j(A, p_{sk}), \\
V_k^{\tilde{p}} &= [v_{k,0}^{\tilde{p}}, v_{k,1}^{\tilde{p}}, \dots, v_{k,s}^{\tilde{p}}], & \text{span}(V_k^{\tilde{p}}) &= K_{j+1}(A, \tilde{p}_{sk}), \\
V_k^{\tilde{r}} &= [v_{k,0}^{\tilde{r}}, v_{k,1}^{\tilde{r}}, \dots, v_{k,s-1}^{\tilde{r}}], & \text{span}(V_k^{\tilde{r}}) &= K_j(A, \tilde{r}_{sk}).
\end{aligned} \quad (2.8)$$

We start with $v_{k,0}^p = p_{sk}, v_{k,0}^r = r_{sk}, v_{k,0}^{\tilde{p}} = \tilde{p}_{sk}, v_{k,0}^{\tilde{r}} = \tilde{r}_{sk}$ and then we use these three-term vectors [4] for $i \in \{0, \dots, s-1\}$:

$$\begin{aligned}
v_{k,i+1}^p &= \frac{1}{\gamma_i}(A - a_i I)v_{k,i}^p - \frac{\beta_{i-1}}{\gamma_i}v_{k,i-1}^p, \\
v_{k,i+1}^r &= \frac{1}{\gamma_i}(A - a_i I)v_{k,i}^r - \frac{\beta_{i-1}}{\gamma_i}v_{k,i-1}^r, \\
v_{k,i+1}^{\tilde{p}} &= \frac{1}{\gamma_i}(A^T - a_i I)v_{k,i}^{\tilde{p}} - \frac{\beta_{i-1}}{\gamma_i}v_{k,i-1}^{\tilde{p}}, \\
v_{k,i+1}^{\tilde{r}} &= \frac{1}{\gamma_i}(A^T - a_i I)v_{k,i}^{\tilde{r}} - \frac{\beta_{i-1}}{\gamma_i}v_{k,i-1}^{\tilde{r}},
\end{aligned} \quad (2.9)$$

here I is the identity matrix and $\gamma_i, a_i, \beta_i \in \mathbb{C}$. We will explain in section 2.4 how to calculate them. The reader should be aware of the fact that in our numerical examples (chapter 3) only real values will be used.

Let $V_k = [V_k^p, V_k^r]$ and $\tilde{V}_k = [V_k^{\tilde{p}}, V_k^{\tilde{r}}]$ be two matrices, and $p'_{k,j}, \tilde{p}'_{k,j}, r'_{k,j}, \tilde{r}'_{k,j}, e_{k,j}$ vectors of length $2s+1$. It follows from (2.2) and (2.8) that the vectors $p_m, r_m, \tilde{p}_m, \tilde{r}_m, x_m - x_{sk}$ from the BiCG algorithm can be written using the Krylov basis. The vectors $p'_{k,j}, \tilde{p}'_{k,j}, r'_{k,j}, \tilde{r}'_{k,j}, e_{k,j}$ describe the vectors of the BiCG algorithm in the following way [5],[2]:

$$\begin{aligned}
p_{sk+j} &= V_k p'_{k,j}, \\
r_{sk+j} &= V_k r'_{k,j}, \\
\tilde{p}_{sk+j} &= \tilde{V}_k \tilde{p}'_{k,j}, \\
\tilde{r}_{sk+j} &= \tilde{V}_k \tilde{r}'_{k,j}, \\
x_{sk+j} - x_{sk} &= V_k e_{k,j}.
\end{aligned} \tag{2.10}$$

When $j = 0$ the coefficients vector $p'_{k,j}, \tilde{p}'_{k,j}, r'_{k,j}, \tilde{r}'_{k,j}, e_{k,j}$ are initialize as [2]:

$$p'_{k,0} = \tilde{p}'_{k,0} = [1, 0_{1,2s}]^T, \quad r'_{k,0} = \tilde{r}'_{k,0} = [0_{1,s+1}, 1, 0_{1,s-1}]^T, \quad e_{k,0} = 0_{2s+1,1}. \tag{2.11}$$

Here $0_{l,i}$ indicates a zero matrix which dimension is $l \times i$, with l rows and i columns [4]. Consider two tridiagonal matrices $C_{k,s+1} \in \mathbb{C}^{s+1 \times s}$, $C_{k,s} \in \mathbb{C}^{s \times s-1}$:

$$C_{k,s+1} = \begin{pmatrix} a_0 & \beta_0 & 0 & 0 & 0 & \dots & 0 \\ \gamma_0 & a_1 & \beta_1 & 0 & 0 & \dots & 0 \\ 0 & \gamma_1 & a_2 & \beta_2 & 0 & \dots & 0 \\ 0 & 0 & \gamma_2 & a_3 & \beta_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \beta_{s-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & a_s \\ 0 & 0 & 0 & 0 & 0 & \dots & \gamma_s \end{pmatrix},$$

$$C_{k,s} = \begin{pmatrix} a_0 & \beta_0 & 0 & 0 & 0 & \dots & 0 \\ \gamma_0 & a_1 & \beta_1 & 0 & 0 & \dots & 0 \\ 0 & \gamma_1 & a_2 & \beta_2 & 0 & \dots & 0 \\ 0 & 0 & \gamma_2 & a_3 & \beta_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \beta_{s-2} \\ \dots & \dots & \dots & \dots & \dots & \dots & a_{s-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & \gamma_{s-1} \end{pmatrix}.$$

We use $C_{k,s+1}$ and $C_{k,s}$ to define the matrix B_k [4]:

$$B_k = \begin{pmatrix} [C_{k,s+1} \ 0_{s+1,1}] \\ [C_{k,s} \ 0_{s,1}] \end{pmatrix}.$$

Here $0_{s+1,1}$ represents a matrix of dimension $s+1 \times 1$, with $s+1$ rows and 1 column, and $0_{s,1}$ is a matrix of dimension $s \times 1$.

We can express B_k as follows:

$$B_k = \begin{pmatrix} a_0 & \beta_0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \gamma_0 & a_1 & \beta_1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & \gamma_1 & a_2 & \beta_2 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \gamma_2 & a_3 & \beta_3 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \beta_{s-1} & 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & a_s & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & \gamma_s & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & a_0 & \beta_0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \gamma_0 & a_1 & \beta_1 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \gamma_1 & a_2 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \gamma_2 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \gamma_{s-1} & \dots \end{pmatrix}.$$

It is possible to describe the product of A and A^T in the rows $\{7, 9\}$ of the BiCG algorithm using the Krylov bases. We start noticing that if we use (2.9) we can write the following [5]:

$$\begin{aligned} AV_{k,j}^p &= V_{k,j+1}^p C_{k,j+1}, & AV_{k,j-1}^r &= V_{k,j}^r C_{k,j}, \\ A^T V_{k,j}^{\bar{p}} &= V_{k,j+1}^{\bar{p}} C_{k,j+1}, & A^T V_{k,j-1}^{\bar{r}} &= V_{k,j}^{\bar{r}} C_{k,j}. \end{aligned}$$

Here $V_{k,j}^p, V_{k,j}^r, V_{k,j}^{\bar{p}}, V_{k,j}^{\bar{r}}$ are defined as:

$$\begin{aligned} V_{k,j}^p &= [v_{k,0}^p, v_{k,1}^p, \dots, v_{k,j}^p], \\ V_{k,j}^r &= [v_{k,0}^r, v_{k,1}^r, \dots, v_{k,j}^r], \\ V_{k,j}^{\bar{p}} &= [v_{k,0}^{\bar{p}}, v_{k,1}^{\bar{p}}, \dots, v_{k,j}^{\bar{p}}], \\ V_{k,j}^{\bar{r}} &= [v_{k,0}^{\bar{r}}, v_{k,1}^{\bar{r}}, \dots, v_{k,j}^{\bar{r}}]. \end{aligned}$$

$V_{k,j}^p$ and $V_{k,j+1}^p$ have size $1 \times j$, $1 \times j + 1$ respectively and $V_{k,j}^r, V_{k,j-1}^r$ indicate the basis matrices of size $1 \times j$ and $1 \times j - 1$ respectively. It's the same for $V_{k,j+1}^{\bar{p}}, V_{k,j}^{\bar{p}}, V_{k,j}^{\bar{r}}, V_{k,j-1}^{\bar{r}}$. We show the case $AV_{k,j}^p = V_{k,j+1}^p C_{k,j+1}$, for $j = 2$, as the other ones are similar. Define $V_{k,3}^p$ as:

$$V_{k,3}^p = [v_{k,0}^p, v_{k,1}^p, v_{k,2}^p],$$

here $v_{k,0}^p, v_{k,1}^p, v_{k,2}^p$ are:

$$\begin{aligned} v_{k,0}^p &= p_{sk}, \\ v_{k,1}^p &= \frac{1}{\gamma_0}(A - a_0 I)v_{k,0}^p, \\ v_{k,2}^p &= \frac{1}{\gamma_1}(A - a_1 I)v_{k,1}^p - \frac{\beta_0}{\gamma_1}v_{k,0}^p. \end{aligned}$$

We have that:

$$AV_{k,2}^p = A[v_{k,0}^p, v_{k,1}^p]$$

and:

$$V_{k,3}^p C_{k,3} = [v_{k,0}^p, v_{k,1}^p, v_{k,2}^p] \begin{bmatrix} a_0 & \beta_0 \\ \gamma_0 & a_1 \\ 0 & \gamma_1 \end{bmatrix} = [v_{k,0}^p a_0 + v_{k,1}^p \gamma_0, v_{k,0}^p \beta_0 + v_{k,1}^p a_1 + v_{k,2}^p \gamma_1].$$

It follows from (2.9) that:

$$v_{k,0}^p a_0 + v_{k,1}^p \gamma_0 = p_{sk} a_0 + \left(\frac{1}{\gamma_0}(A - a_0 I)p_{sk}\right) \gamma_0 = A p_{sk},$$

$$v_{k,0}^p \beta_0 + v_{k,1}^p a_1 + v_{k,2}^p \gamma_1 = p_{sk} \beta_0 + \left(\frac{1}{\gamma_0}(A - a_0 I)p_{sk}\right) a_1 + \left(\frac{1}{\gamma_1}(A - a_1 I)v_{k,1}^p - \frac{\beta_0}{\gamma_1}p_{sk}\right) \gamma_1,$$

after computation, we arrive at:

$$= A \left(\frac{1}{\gamma_0}(A - a_0 I)\right) p_{sk}.$$

It follows that:

$$AV_{k,2}^p = V_{k,3}^p C_{k,3}.$$

Consider the basis matrices $\bar{V}_k^p, \bar{V}_k^r, \bar{V}_k^{\tilde{p}}, \bar{V}_k^{\tilde{r}}$, which are equal to $V_k^p, V_k^r, V_k^{\tilde{p}}, V_k^{\tilde{r}}$ but their last column is equal to a zero column, and define $\bar{V}_k = [\bar{V}_k^p, \bar{V}_k^r]$, $\tilde{\bar{V}}_k = [\bar{V}_k^{\tilde{p}}, \bar{V}_k^{\tilde{r}}]$, by (2.8) we can write:

$$\begin{aligned} A\bar{V}_k &= A[V_{k,s}^p, 0_{n,1}, V_{k,s-1}^r, 0_{n,1}] \\ &= A[\bar{V}_{k,s+1}^p, \bar{V}_{k,s}^r] = [V_k^p, V_k^r]B_k = V_k B_k. \end{aligned}$$

So we have:

$$A\tilde{\bar{V}}_k = V_k B_k.$$

In a similar way we have: $A^T \tilde{\bar{V}}_k = \tilde{V}_k B_k$. We show the case for $A\bar{V}_k = V_k B_k$, for $s = 2$:

$$\begin{aligned} A\bar{V}_k &= A[v_{k,0}^p, v_{k,1}^p, 0, v_{k,0}^r, 0] \\ &= [v_{k,0}^p, v_{k,1}^p, v_{k,2}^p, v_{k,0}^r, v_{k,1}^r]B_k. \end{aligned}$$

Since:

$$\begin{aligned} [v_{k,0}^p, v_{k,1}^p, v_{k,2}^p, v_{k,0}^r, v_{k,1}^r]B_k &= [v_{k,0}^p, v_{k,1}^p, v_{k,2}^p, v_{k,0}^r, v_{k,1}^r] \begin{bmatrix} \mathbf{a}_0 & \beta_0 & 0 & 0 & 0 \\ \gamma_0 & \mathbf{a}_1 & 0 & 0 & 0 \\ 0 & \gamma_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{a}_0 & 0 \\ 0 & 0 & 0 & \gamma_0 & 0 \end{bmatrix} \\ &= [v_{k,0}^p \mathbf{a}_0 + v_{k,1}^p \gamma_0, v_{k,0}^p \beta_0 + v_{k,1}^p \mathbf{a}_1 + v_{k,2}^p \gamma_1, 0, v_{k,0}^r \mathbf{a}_0 + v_{k,1}^r \gamma_0, 0] \\ &= A[v_{k,0}^p, v_{k,1}^p, 0, v_{k,0}^r, 0]. \end{aligned}$$

Here in the last equal we use (2.9). So we have: $A\bar{V}_k = V_k B_k$. For the other cases is similar.

It follows that the product of A in row (7) can be written as follow:

$$\begin{aligned} Ap_{sk+j-1} &= AV_k p'_{k,j-1} && \text{by (2.10)} \\ &= A[V_{k,s+1}^p, V_{k,s}^r] p'_{k,j-1} \\ &= A[V_{k,s}^p, 0_{n,1}, V_{k,s-1}^r, 0_{n,1}] p'_{k,j-1} \\ &= A[\bar{V}_k^p, \bar{V}_k^r] p'_{k,j-1} \\ &= A\bar{V}_k p'_{k,j-1} \\ &= V_k B_k p'_{k,j-1}. \end{aligned} \tag{2.12}$$

The product of A^T in row (9) of the BiCG algorithm is similar. Consider the rows {7, 8, 9, 12, 13} of the BiCG algorithm and replace the vectors $r_m, p_m, x_m, \tilde{r}_m, \tilde{p}_m$ with (2.10) and using (2.12), we have that lines {7, 9, 12, 13, 8}, in this order, become:

$$\begin{aligned} V_k r'_{k,j} &= V_k r'_{k,j-1} - \alpha_{m-1} AV_k p'_{k,j-1} \\ &= V_k r'_{k,j-1} - \alpha_{m-1} V_k B_k p'_{k,j-1}, \end{aligned} \tag{2.13}$$

$$\begin{aligned}\tilde{V}_k \tilde{r}'_{k,j} &= \tilde{V}_k \tilde{r}'_{k,j-1} - \alpha_{m-1} A^T \tilde{V}_k \tilde{p}'_{k,j-1} \\ &= \tilde{V}_k \tilde{r}'_{k,j-1} - \alpha_{m-1} \tilde{V}_k B_k \tilde{p}'_{k,j-1},\end{aligned}\quad (2.14)$$

$$V_k p'_{k,j} = V_k r'_{k,j} + \beta_m V_k p'_{k,j-1}, \quad (2.15)$$

$$\tilde{V}_k \tilde{p}'_{k,j} = \tilde{V}_k \tilde{r}'_{k,j} + \beta_m \tilde{V}_k \tilde{p}'_{k,j-1}, \quad (2.16)$$

$$V_k e_{k,j} = V_k e_{k,j-1} + \alpha_{m-1} V_k p'_{k,j-1}. \quad (2.17)$$

We define $G_k = \tilde{V}_k^T V_k$ and, using $C_{k,s+1}$, $C_{k,s}$ and B_k , the scalar products:

$$\langle \tilde{r}_{sk+j}, r_{sk+j} \rangle, \langle \tilde{p}_{sk+j-1}, A p_{sk+j-1} \rangle,$$

which were denoted as ρ_m and σ_{m-1} (lines (10) and (5) respectively), in the BiCG algorithm, can be expressed, using the Krylov bases and (2.10), in the following way [5]:

$$\begin{aligned}\langle \tilde{r}_m, r_m \rangle &= \tilde{r}_m^T r_m \\ &= (\tilde{V}_k \tilde{r}'_{k,j})^T (V_k r'_{k,j}) \\ &= \tilde{r}'_{k,j}{}^T G_k r'_{k,j}, \\ \langle \tilde{p}_{m-1}, A p_{m-1} \rangle &= \tilde{p}_{m-1}^T A p_{m-1} \\ &= (\tilde{V}_k \tilde{p}'_{k,j-1})^T (A V_k p'_{k,j-1}) \\ &= \tilde{p}'_{k,j-1}{}^T \tilde{V}_k^T A V_k p'_{k,j-1}, \\ &\text{it follows from (2.12) that} \\ &= \tilde{p}'_{k,j-1}{}^T \tilde{V}_k^T V_k B_k p'_{k,j-1}, \\ &\text{because } G_k = \tilde{V}_k^T V_k, \text{ we have} \\ &= \tilde{p}'_{k,j-1}{}^T G_k B_k p'_{k,j-1}.\end{aligned}\quad (2.18)$$

If we collect the equations (2.13)-(2.19) we are able to create the inside loop from $j=1$ to $j=s$ of the s -step BiCG algorithm [2]. While if we collect the bases from (2.8), equations (2.11) and (2.10), the matrix B_k , the product G_k , and V_k , \tilde{V}_k we can create the outside loop with index k . In the following section we present the s -step BiCG algorithm.

2.3 S-step BiCG method

The s -step BiCG method is an iterative algorithm utilized to make of solutions for “nonsymmetric linear systems $Ax=b$ ” [2]. It is used especially with large matrices.

S-step BiCG algorithm

The following algorithm is taken from the technical report by Carson and Demmel in [4] and from the PhD thesis by Carson [2]. It's nearly the same, we changed just some notation.

1. $x_0 = [0, \dots, 0]$,

2. $r_0 = p_0 = \tilde{r}_0 = \tilde{p}_0 = b - Ax_0$, $k = 0$
3. *while* k *until* convergence
4. $V_k^p = [v_{k,0}^p, \dots, v_{k,s}^p]$
5. $V_k^r = [v_{k,0}^r, \dots, v_{k,s-1}^r]$
6. $V_k^{\tilde{p}} = [v_{k,0}^{\tilde{p}}, \dots, v_{k,s}^{\tilde{p}}]$
7. $V_k^{\tilde{r}} = [v_{k,0}^{\tilde{r}}, \dots, v_{k,s-1}^{\tilde{r}}]$
8. $V_k = [V_k^p, V_k^r]$
9. $\tilde{V}_k = [V_k^{\tilde{p}}, V_k^{\tilde{r}}]$
10. *Compute* the matrix B_k
11. $G_k = \tilde{V}_k^T V_k$
12. $p'_{k,0} = [1, 0_{1,2s}]^T$
13. $r'_{k,0} = [0_{1,s+1}, 1, 0_{1,s-1}]^T$
14. $\tilde{r}'_{k,0} = r'_{k,0}$, $\tilde{p}'_{k,0} = p'_{k,0}$
15. $e_{k,0} = [0_{2s+1}]$
16. *for* $j = 1, \dots, s$
17. $\delta_{m-1} = \tilde{r}'_{k,j-1}{}^T G_k r'_{k,j-1}$
18. $\alpha_{m-1} = \delta_{m-1} / \tilde{p}'_{k,j-1}{}^T G_k B_k p'_{k,j-1}$
19. $e_{k,j} = e_{k,j-1} + \alpha_{m-1} p'_{k,j-1}{}^T$
20. $r'_{k,j} = r'_{k,j-1} - \alpha_{m-1} B_k p'_{k,j-1}$
21. $\tilde{r}'_{k,j} = \tilde{r}'_{k,j-1} - \alpha_{m-1} B_k \tilde{p}'_{k,j-1}$
22. $\delta_m = \tilde{r}'_{k,j}{}^T G_k r'_{k,j}$
23. $\beta_m = \delta_m / \delta_{m-1}$
24. $p'_{k,j} = r'_{k,j} + \beta_m p'_{k,j-1}$
25. $\tilde{p}'_{k,j} = \tilde{r}'_{k,j} + \beta_m \tilde{p}'_{k,j-1}$
26. *end for*
27. $x_{sk+s} = V_k e_{k,s}^T + x_{sk}$
28. $r_{sk+s} = V_k r'_{k,s}$
29. $p_{sk+s} = V_k p'_{k,s}$
30. $\tilde{r}_{sk+s} = \tilde{V}_k \tilde{r}'_{k,s}$
31. $\tilde{p}_{sk+s} = \tilde{V}_k \tilde{p}'_{k,s}$
32. $k=k+1$
33. *end while*
34. *return* x_{sk} .

2.4 Bases

In this section we will study two bases that we use in the s -step BiCG algorithm. Particularly, we will see how to calculate (2.9). In chapter 3 we will compare them in some numerical examples.

2.4.1 Krylov or Monomial basis

Consider a matrix A and a vector p_{sk} . The Krylov subspace created by A and p_{sk} is the following:

$$K_{s+1} = \text{span}(p_{sk}, Ap_{sk}, \dots, A^s p_{sk}).$$

The three-term vectors (2.9), become:

$$\begin{aligned} v_{k,i+1}^p &= Av_{k,i}^p, & v_{k,i+1}^r &= Av_{k,i}^r, \\ v_{k,i+1}^{\bar{p}} &= Av_{k,i}^{\bar{p}}, & v_{k,i+1}^{\bar{r}} &= Av_{k,i}^{\bar{r}}, \end{aligned}$$

with $\gamma_i = 1$, $\alpha_i = 0$, $\beta_i = 0$.

So for instance, for $i=0$, we will have:

$$v_{k,1}^p = Av_{k,0}^p = Ap_{sk},$$

for $i=1$:

$$v_{k,2}^p = Av_{k,1}^p = A(Ap_{sk}),$$

and so on, until we reach the following matrix:

$$V_k^p = [v_{k,0}^p, \dots, v_{k,s}^p] = [p_{sk}, Ap_{sk}, \dots, A^s p_{sk}].$$

which is called monomial or Krylov basis and we can affirm that (2.8) is satisfied:

$$\text{span}(V_k^p) = K_{j+1}(A, p_{sk}).$$

In a similar way we obtain: $V_k^r, V_k^{\bar{p}}, V_k^{\bar{r}}$.

2.4.2 Chebyshev Polynomials

We start by reviewing Chebyshev polynomials. This paragraph and figure (2.1) are based on the research paper written by Manteuffel [12]. In figure (2.1) we use just one line. Figure (2.2) is based on [16]. Let $z \in \mathbb{C}$, $z = x + iy$, $x, y \in \mathbb{R}$ and $n \in \mathbb{R}$. The Chebyshev polynomials are [5]:

$$\begin{aligned} \tau_0(z) &= 1, \\ \tau_1(z) &= z, \\ \tau_n(z) &= 2z\tau_{n-1}(z) - \tau_{n-2}(z), \quad \text{for } n > 1. \end{aligned} \tag{2.20}$$

We can express $\tau_n(z)$ in the following way: $\tau_n(z) = \cosh(n \cosh^{-1}(z))$ [12]. Suppose that $x = a$, $a \in \mathbb{R}$, is a line, then the function $\cosh(z)$ maps x onto an ellipse (figure 2.1) [12]. If we consider the formula for the hyperbolic cosine for z , then we have [12]:

$$\cosh(z) = \cosh(x + iy) = \cosh(x)\cos(y) + i\sinh(x)\sin(y) = u + iv,$$

where $\cosh(x)\cos(y) = u$ and $\sinh(x)\sin(y) = v$. We notice that [12]:

$$\frac{u^2}{\cosh^2(x)} + \frac{v^2}{\sinh^2(x)} = 1,$$

since:

$$\frac{u^2}{\cosh^2(x)} + \frac{v^2}{\sinh^2(x)} = \frac{\cosh^2(x)\cos^2(y)}{\cosh^2(x)} + \frac{\sinh^2(x)\sin^2(y)}{\sinh^2(x)} = 1.$$

2.4.3 Chebyshev Basis

To construct the Chebyshev basis we need an ellipse and the spectrum of the matrix A of a linear system $Ax = b_1$. Consider an ellipse delimited by the rectangle [10]:

$$\{z = x+iy : d-a \leq x \leq d+a, -b \leq y \leq b \mid a, b, d \in \mathbb{R}, a \geq 0, b \geq 0\}, \quad (2.21)$$

with (d, v) the center of the ellipse and $c = \sqrt{a^2 - b^2}$, which means that the foci are at $d+c$ and $d-c$ [10], [5]. It is supposed that the set of the eigenvalues of the matrix A is delimited by (2.21) [5]. “The scaled, shifted and rotated Chebyshev polynomials” are [5]:

$$\begin{aligned} \tilde{\tau}_0(z) &= 1, \\ \tilde{\tau}_1(z) &= \frac{\sigma_0(d-z)}{c\sigma_1} \\ \tilde{\tau}_j(z) &= \frac{2\sigma_{j-1}(d-z)\tilde{\tau}_{j-1}(z)}{c\sigma_j} - \frac{\sigma_{j-2}\tilde{\tau}_{j-2}(z)}{\sigma_j}, \quad j > 1, \\ \sigma_j &= \tau_j(d/c). \end{aligned} \quad (2.22)$$

Consider the following constants that we will use in the matrices $C_{k,s+1}, C_{k,s}, B_k$ and in the three-term vectors (2.9) [5]:

$$\begin{aligned} a_j &= d, & \beta_j &= -c \frac{\sigma_j}{2\sigma_{j+1}}, \\ \gamma_0 &= -c \frac{\sigma_1}{\sigma_0}, & \gamma_j &= -c \frac{\sigma_{j+1}}{2\sigma_j}, \quad j > 0. \end{aligned} \quad (2.23)$$

If A is a real matrix, [5] and [10] provide the following constants:

$$\begin{aligned} a_j &= d, & \beta_j &= \frac{c^2}{4g}, \\ \gamma_0 &= 2g, & \gamma_j &= g, \quad j > 0, \end{aligned}$$

here $g = \max(a, b)$. These values are the one that we will use in the next chapter for the numerical examples. The Krylov matrices (2.8) created in section 2.2 become:

$$\begin{aligned} V_k^p &= [v_{k,0}^p, \dots, v_{k,s}^p] = [\tilde{\tau}_0(A)p_{sk}, \tilde{\tau}_1(A)p_{sk}, \dots, \tilde{\tau}_s(A)p_{sk}], \\ V_k^r &= [v_{k,0}^r, \dots, v_{k,s-1}^r] = [\tilde{\tau}_0(A)r_{sk}, \tilde{\tau}_1(A)r_{sk}, \dots, \tilde{\tau}_{s-1}(A)r_{sk}], \\ V_k^{\tilde{p}} &= [v_{k,0}^{\tilde{p}}, \dots, v_{k,s}^{\tilde{p}}] = [\tilde{\tau}_0(A^T)\tilde{p}_{sk}, \tilde{\tau}_1(A^T)\tilde{p}_{sk}, \dots, \tilde{\tau}_s(A^T)\tilde{p}_{sk}], \\ V_k^{\tilde{r}} &= [v_{k,0}^{\tilde{r}}, \dots, v_{k,s-1}^{\tilde{r}}] = [\tilde{\tau}_0(A^T)\tilde{r}_{sk}, \tilde{\tau}_1(A^T)\tilde{r}_{sk}, \dots, \tilde{\tau}_{s-1}(A^T)\tilde{r}_{sk}], \end{aligned}$$

which are called Chebyshev bases. Using (2.22), (2.23) and by $\tilde{\tau}_0(A) = 1$, for $i = 0$, and $i = 1$ for instance, it is possible to see that the three-term vectors

(2.9) become:

$$\begin{aligned}
 v_{k,0}^p &= p_{sk} = \tilde{\tau}_0(A)p_{sk}, \\
 v_{k,1}^p &= \frac{1}{\gamma_0}(A - a_0I)v_{k,0} \\
 &= \frac{1}{\gamma_0}(A - a_0I)p_{sk} \\
 &= -\frac{\sigma_0}{c\sigma_1}(A - dI)p_{sk} \\
 &= \frac{\sigma_0(d - A)}{c\sigma_1}p_{sk} \\
 &= \tilde{\tau}_1(A)p_{sk}.
 \end{aligned}$$

The other three-term vectors are obtained in a similar way.

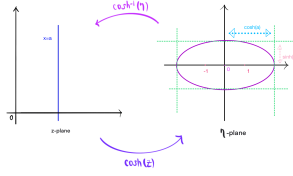


Figure 2.1: - From a line to an ellipse.

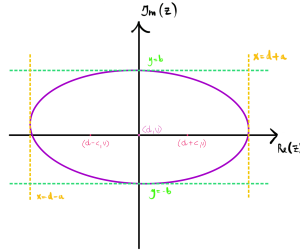


Figure 2.2: - Ellipse.

Chapter 3

Numerical experiments

In this chapter we will show some numerical examples for the BiCG and the s-step BiCG methods. The criteria for stopping the algorithm is that what we call 2-norm residual ($\frac{\|r_m\|_2}{\|b_1\|_2}$) reaches the tolerance, which is 10^{-6} or 10^{-10} in these examples, and here r_m is the computed residual. We call true residual the following: $\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2}$ [5]. The software that has been used is MATLAB, and the number of iterations starts at $k = 1$.

1) *Example* We look first at a small matrix to see how the algorithms work. Let $A \in \mathbb{R}^{4 \times 4}$ be a sparse and nonsymmetric matrix.

Let $\delta_x = n + 1 = 5$, where $n = 4$.

Let $b_1 = [1, 0, 1, 0]^T$.

We can write A as:

$$A = \frac{1}{\delta_x} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

We want to solve the linear system $Ax = b_1$, which exact solution is: $x = [5, 5, 10, 10]^T$.

BiCG method.

We will show what we get if we use the BiCG algorithm. The condition for stopping the algorithm is that the 2-norm residual, $\frac{\|r_m\|_2}{\|b_1\|_2}$, reaches the tolerance of 10^{-6} . The algorithm stopped at the maximum number of iterations, which is 4. The user can decide the maximum number of iterations, in this case after the fourth iteration the algorithm diverges. The final solution is $x_4 = [5, 5, 10, 5]^T$, which is not near to the exact solution. The 2-norm residual in the last iteration is $r = \frac{\|r_m\|_2}{\|b_1\|_2} = 0.707106781186547$. In this case the true residual is the same. Figure 3.1 shows the plot of the 2-norm residual during the different iterations using the BiCG algorithm.

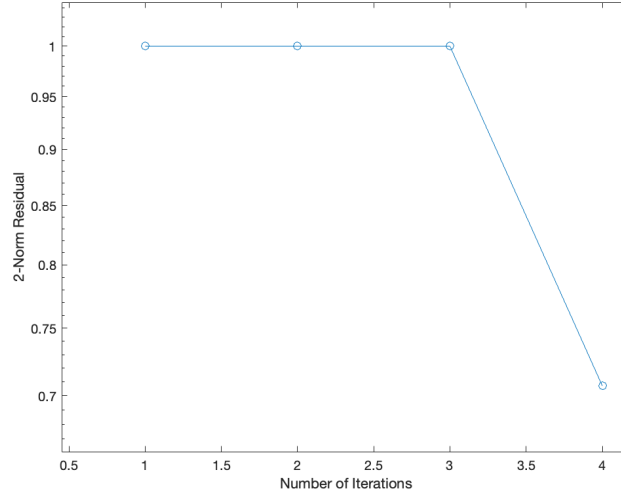


Figure 3.1: plot of the 2-norm residual using BiCG method, first example.

What happens in this case is what we introduced as “breakdown in the underlying Lanczos process” in chapter 2 [13], which means that at $k=4$ the code collapses. To avoid this problem, we changed the initial \tilde{r}_1 and as suggested in the algorithm presented in [5], we chose it. Our choice is $\tilde{r}_1 = [1, 1, 1, 1]^T$. Changing the maximum number of iterations to $n = 89$, we experienced a convergence at $k = 88$. The final solution that we achieved is:

$$x_m = \begin{bmatrix} 4.999999997764300 \\ 5.000000024547736 \\ 9.999999781395340 \\ 10.000001443133058 \end{bmatrix}.$$

The 2-norm residual is now $\frac{\|r_m\|_2}{\|b_1\|_2} = 2.375381075345510e - 07$, and the true residual is: $\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 2.375381099159145e - 07$. Figure 3.2 shows the plot of the 2-norm residual during the different iterations using the BiCG algorithm with the new \tilde{r}_1 .

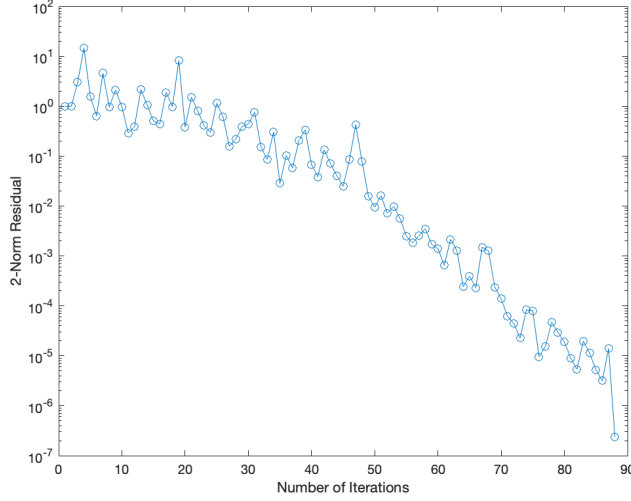


Figure 3.2: plot of the 2-norm residual using BiCG method, first example.

Now we will show if and how the results change if we use s -step BiCG algorithm instead of the BiCG algorithm, we will use the Chebyshev and the monomial basis.

s-step BiCG method.

Chebyshev basis. For using the Chebyshev basis we need an ellipse and the spectrum of A . First of all, we need to find the eigenvalues of the matrix A . Here the eigenvalues are all 0.20. Let $z = 0.20$ and consider an ellipse bounded by the rectangle (2.21). Let $a = 7.9888$, $b = 0.010$, $d = 7.98$, $c = \sqrt{a^2 - b^2}$, the center of the ellipse is $(d, v) = (7.98, 0)$, and $g = \max(a, b)$. The constants in the matrices $C_{k,s+1}, C_{k,s}, B_k$ are: $\alpha_j = d$, $\beta_j = \frac{c^2}{4g}$, $\gamma_0 = 2g$ and $\gamma_{j+1} = g$, for $j = 0, \dots, s$.

For $s=4$, the maximum number of iterations that we choose is 2513 ($628 \times s + 1$) and the tolerance as before is 10^{-6} . The level of tolerance is reached at 1291st iteration of the k -loop. The final 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 9.785531951826592e - 07,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 9.785522663754157e - 07.$$

The solution is:

$$x_m = \begin{bmatrix} 5.000000009269464 \\ 5.000000009379033 \\ 10.000000010773732 \\ 9.999993091370650 \end{bmatrix}.$$

For s=8, the tolerance is reached at 1263rd iteration, the 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 9.687621446093399e - 07,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 9.687776137392549e - 07.$$

The solution is:

$$x_m = \begin{bmatrix} 4.999999958769584 \\ 4.999999958966497 \\ 9.999999927688089 \\ 9.999993077591380 \end{bmatrix}.$$

For s=16, we have convergence at 145th iteration. The 2-norm residual is:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 7.961976029306281e-07, \text{ the true residual is: } \frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 7.962429652349543e-07.$$

The solution is:

$$\begin{bmatrix} 4.999999953345904 \\ 4.999999953017598 \\ 9.999999925339091 \\ 10.000005555365750 \end{bmatrix}.$$

Monomial Basis. We now do the same process with the monomial basis, what changes is that $d = 0$, $\gamma_i = 1$, $\beta_i = 0$. For s=4, we reached convergence at 115th iteration. The 2-norm residual and the true residual, after the iterations, are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 7.008830181987705e - 07,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 7.008830373062005e - 07.$$

While the solution is:

$$x_m = \begin{bmatrix} 5.000000000000000 \\ 5.000000000000001 \\ 9.999999999999998 \\ 10.000004955991484 \end{bmatrix}.$$

For $s=8$, the tolerance is reached at 152nd iteration. The 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 7.392758276606094e - 07,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 7.392758187707963e - 07.$$

The solution is:

$$x_m = \begin{bmatrix} 5.000000000000012 \\ 4.999999999999986 \\ 9.999999999999963 \\ 9.999994772530517 \end{bmatrix}.$$

For $s=16$, the tolerance is obtained at 73rd iteration, the 2-norm residual and the true residual are:

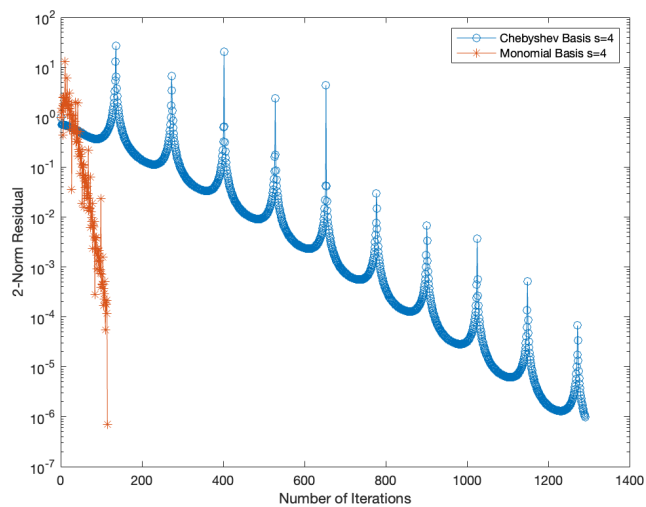
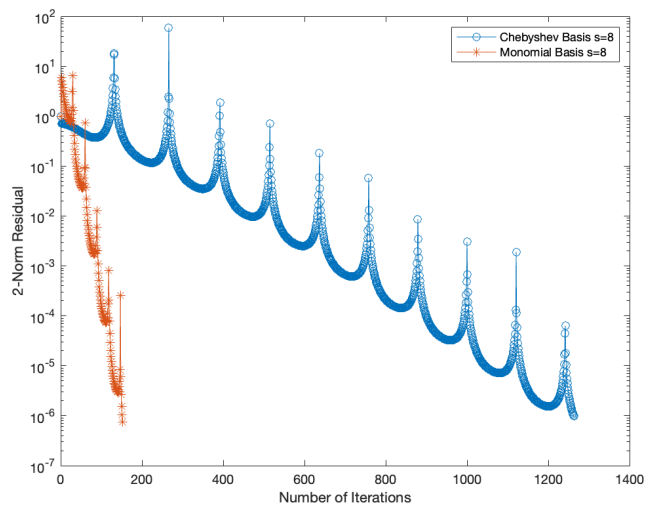
$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 4.183973937685572e - 07,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 4.183973952504021e - 07.$$

The solution is:

$$x_m = \begin{bmatrix} 5.000000000000004 \\ 5.000000000000039 \\ 9.999999999999964 \\ 10.000002958516317 \end{bmatrix}.$$

Figures 3.3, 3.4, 3.5 show the 2-norm residual during the number of iterations using the Chebyshev basis and the monomial basis for $s = \{4, 8, 16\}$.

Figure 3.3: plot of the 2-norm residual using the s-step BiCG method, $s=4$ Figure 3.4: plot of the 2-norm residual using the s-step BiCG method, $s=8$

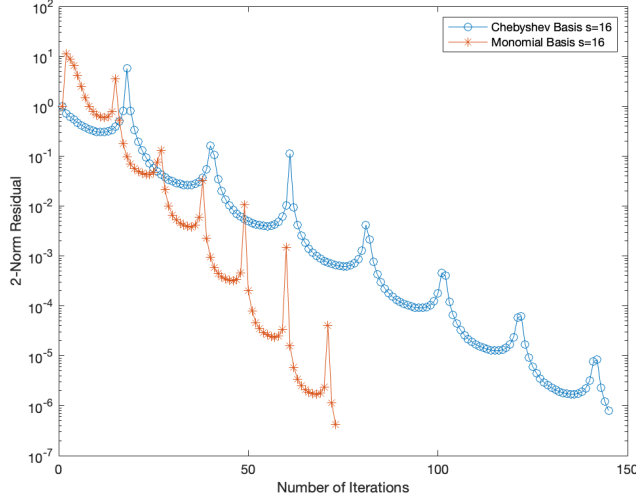


Figure 3.5: plot of the 2-norm residual using the s-step BiCG method, s=16

2) Example - cdde1

The next example that we will show is from “the constant-coefficient convection diffusion equation” [5]. The matrix that we are about to use is called “cdde1” and it is from [1]. The problem is the following:

$$\begin{aligned} -\Delta u + 2p_1 u_x + 2p_2 u_y - p_3 u_y &= f \quad \text{in } [0, 1]^2, \\ u &= g \quad \text{on } \partial[0, 1]^2. \end{aligned}$$

The matrix is unsymmetric and has dimension 961×961 and $(p_1, p_2, p_3) = (1, 2, 30)$. We create the vector b_1 in the following way: $b_1 = \frac{AU}{\sqrt{n}}$, where U is a vector of size 961×1 which components are ones and $n = 961$. We use a tolerance of 10^{-10} . We will use the BiCG and the s-step BiCG methods with the monomial basis and the Chebyshev basis, for $s = \{4, 8, 16\}$.

BiCG method. We reached the tolerance at $k=161$, with a 2-norm residual of $\frac{\|r_m\|_2}{\|b_1\|_2} = 4.075713485829166e - 11$. The true residual is: $\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 4.075688135301425e - 11$. The following figure shows the 2-norm residual for the BiCG method.

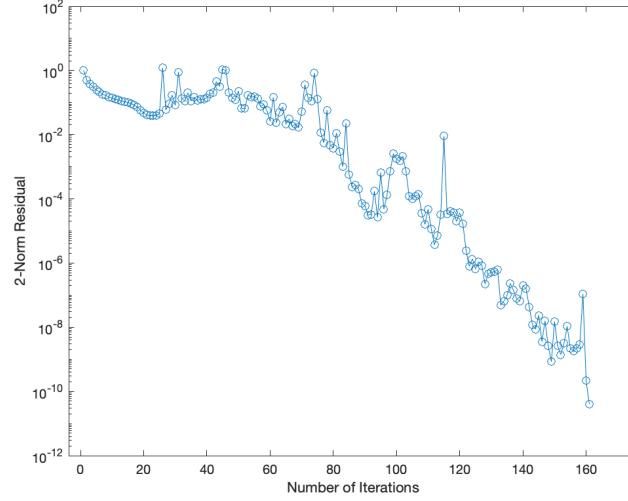


Figure 3.6: plot of the 2-Norm residual using the BiCG method

s-step BiCG. In order to use the Chebyshev basis we have to construct an ellipse and we need the spectrum of the matrix A . The ellipse is delimited by the rectangle (2.21). To construct the rectangle and the ellipse we need the maximum and the minimum eigenvalues, since it is supposed that the set of the eigenvalues is delimited by (2.21) [5]. To find them, we can use “eigs(A)” in MATLAB, for the maximum eigenvalue, and “eigs(A,1,'smallestab’)", for the minimum one. The maximum one is: $\lambda_1 = 7.9466$, while the minimum one is: $\lambda_2 = -0.0052$. We assign: $a = 8$, $b = 0.010$, $d = 7$.

Monomial Basis. For $s=4$, the tolerance is reached at iteration number 41. The 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|}{\|b_1\|} = 3.461371396068399e - 11$$

$$\frac{\|b_1 - Ax_m\|}{\|b_1\|} = 3.461400328811182e - 11.$$

For $s=8$, when we use the monomial basis, we reached the tolerance at 30th iteration, the 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 3.911032297540012e - 11$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 3.922603323237105e - 11.$$

For $s=16$, the tolerance is never reached, and the 2-norm residual and the true

residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 4.029813877452058e + 126$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 4.029813877452023e + 126.$$

Chebyshev Basis. For s=4, the 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 0.004871241595892,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 0.004871241604220.$$

The tolerance is never reached, so the algorithm stops at the maximum number of iterations, which is 2513 in this case. For s=8, the tolerance is reached at k=3364, the 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 4.413627013589070e - 11,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 0.030766839185844.$$

For s=16 the tolerance is reached at 335th iteration, the 2-norm residual and the true residual are:

$$\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 6.766967663643521e - 11,$$

$$\frac{\|b_1 - Ax_m\|_2}{\|b_1\|_2} = 5.085361350595155e - 08.$$

Figures 3.7, 3.8, 3.9 show the 2-norm residual for s=4, s=8 and s=16. The number of k that has been used is 628. This number has been chosen by the author.

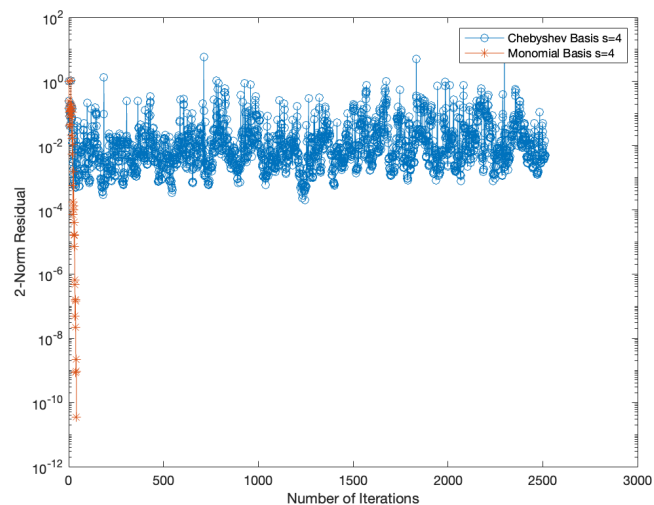


Figure 3.7: plot of the 2-norm residual using the s -step BiCG, $s=4$, Monomial and Chebyshev bases

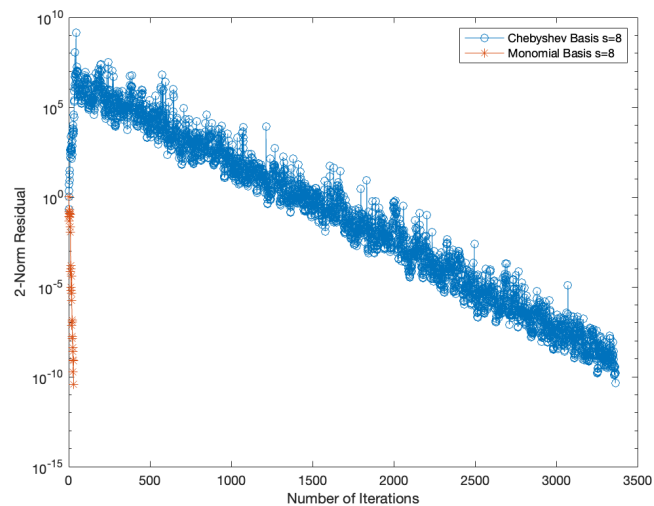


Figure 3.8: plot of the 2-norm residual using the s -step BiCG method, $s=8$, Monomial and Chebyshev bases

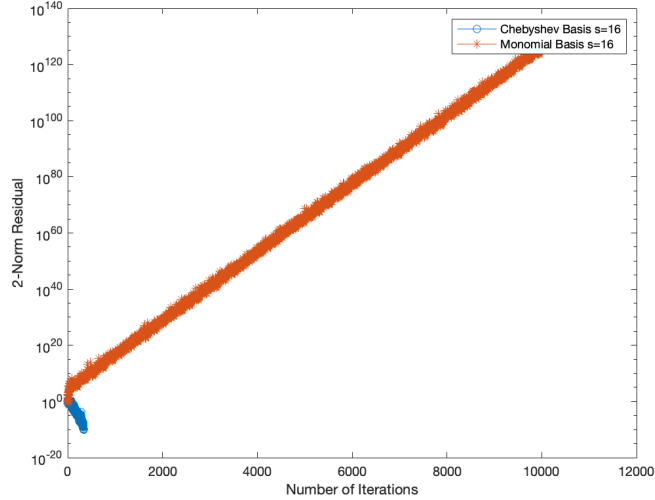


Figure 3.9: plot of the 2-norm residual using the s -step BiCG method, $s=16$, Monomial and Chebyshev bases

Comments about the choice of basis and about the BiCG and the s -step BiCG methods. When we use large and sparse matrices, as in the second example, for $s=4$, the monomial basis is a better choice. But as s becomes a bigger number, for $s=16$ for instance, we have seen that the Chebyshev basis reaches the tolerance, but the monomial basis does not. So we can say that for large values of s , the Chebyshev basis is a better choice than the monomial one. In the first example we see that when we use the BiCG algorithm we are able to reach a computed solution which is close to the exact one at 88th iteration. Meanwhile, when we use the s -step BiCG method, the number of iterations used for achieving the convergence making use of the Chebyshev basis is bigger than the one utilized for the BiCG algorithm. When we use the monomial basis, just for $s = 16$ the convergence is achieved after a smaller number of iterations with respect to the BiCG method. In the second example the BiCG algorithm reaches the tolerance before arriving at the maximum number of iterations and with a smaller number of iterations with respect to the s -step BiCG algorithm when we use the Chebyshev basis, or for $s = 16$ when we use the monomial basis. The large number of iterations that the s -step BiCG method made to reach the tolerance is caused by roundoff errors made while computing the bases and changing the bases [2]. Another effect of the roundoff errors can be seen in the different values of the 2-norm residual, where we used the residual updated by the algorithms, and the true residual, where we used the solution computed and updated by the algorithms [2].

Chapter 4

S-step BiCG technique and finite precision

This chapter is based on the research paper [13], on the technical report [4] and on chapter 5 of the PhD thesis [2]. The theorems and the proofs written in this chapter are the same as those given by Tong and Ye in [13], by Carson and Demmel in [4] and by Carson in [2]. We added just some explanations in some parts and they are adapted to our s-step BiCG algorithm. In this chapter we discuss about roundoff errors that the s-step BiCG algorithm encounters. These errors affect the values computed by the algorithm, and as a consequence, as we saw in chapter 3, these values differ from the one in exact arithmetic.

4.1 Revise of s-step BiCG technique

This section is the same as section (2.3) of [4] and (5.2.1) of [2].

In chapter (2), section 2.3, we present the s-step BiCG algorithm. In this section we see how it is possible to write lines (20) and (24) of the s-step BiCG algorithm using matrices.

Theorem 1. Let $r'_{k,j}$, $p'_{k,j}$, $e_{k,j}$, α_m , $p_{k,s}$, $r_{k,s}$, β_m , V_k and B_k be the quantities in the s-step BiCG algorithm. Then:

$$AV_k \mathcal{R}'_{k,j} = \mathcal{V}_k \mathcal{R}'_{k,j} \mathcal{T}_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'_{sk+j+1T},$$

here $\mathcal{V}_k, \mathcal{R}'_{k,j}$ and $\mathcal{T}_{k,j}$ are defined as:

$$\mathcal{V}_k = [\bar{V}_0, \dots, \bar{V}_k],$$

$$\mathcal{R}'_{k,j} = \begin{bmatrix} R'_{0,s-1} & & & & \\ & R'_{1,s-1} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & R'_{k,j} \end{bmatrix},$$

$$\mathcal{T}_{k,j} = \begin{bmatrix} \frac{1}{\alpha_0} & -\frac{\beta_1}{\alpha_0} & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_1}{\alpha_0} & \ddots & & \\ & \ddots & \ddots & & \\ & & & \frac{\beta_m}{\alpha_{m-1}} & \\ & & & -\frac{1}{\alpha_{m-1}} & \frac{1}{\alpha_m} + \frac{\beta_m}{\alpha_{m-1}} \end{bmatrix}.$$

Proof. The lines (20) and (24) of the s-step BiCG algorithm are the following:

$$r'_{k,j} = r'_{k,j-1} - \alpha_{m-1} B_k p'_{k,j-1}, \quad (4.1)$$

$$p'_{k,j} = r'_{k,j} + \beta_m p'_{k,j-1}, \quad (4.2)$$

for $j = 1, \dots, s$. We can write (4.1) in the following way:

$$r'_{k,j+1} = r'_{k,j} - \alpha_{sk+j} B_k p'_{k,j},$$

$$\text{so we have:} \quad (4.3)$$

$$B_k p'_{k,j} = \frac{1}{\alpha_{sk+j}} (r'_{k,j} - r'_{k,j+1}),$$

and (4.2) as:

$$p'_{k,j+1} = r'_{k,j+1} + \beta_{sk+j+1} p'_{k,j},$$

equation (4.3) and the last equation are valid for $j = 0, \dots, s-1$. We can write (4.2) as follows:

$$r'_{k,j} = p'_{k,j} - \beta_{sk+j} p'_{k,j-1} \quad (4.4)$$

If we left-multiply (4.4) by V_k , and we utilize:

$$V_k[r'_{k,j}, p'_{k,j}] = \bar{V}_k[r'_{k,j}, p'_{k,j}],$$

for $j = 0, \dots, s-1$, we get:

$$\bar{V}_k r'_{k,j} = \bar{V}_k p'_{k,j} - \beta_m \bar{V}_k p'_{k,j-1}, \quad (4.5)$$

which is true from $j = 1$, because $p'_{k,-1}$ is not stated. We want to write an expression for $r'_{k,0}$, we have:

$$\begin{aligned} \bar{V}_k r'_{k,0} &= V_{k-1} r'_{k-1,s} \\ &= V_{k-1} p'_{k-1,s} - \beta_{sk} \bar{V}_{k-1} p'_{k-1,s-1} \\ &= \bar{V}_k p'_{k,0} - \beta_{sk} \bar{V}_{k-1} p'_{k-1,s-1}. \end{aligned} \quad (4.6)$$

Suppose that:

$$\begin{aligned} R'_{k,j} &= [r'_{k,0}, r'_{k,1}, \dots, r'_{k,j}], \\ P'_{k,j} &= [p'_{k,0}, p'_{k,1}, \dots, p'_{k,j}], \end{aligned}$$

so we can write (4.5) as:

$$\bar{V}_k R'_{k,j} = \bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} \bar{V}_{k-1} p'_{k-1,s-1} e_1'^T \quad (4.7)$$

with :

$$U_{k,j} = \begin{bmatrix} 1 & -\beta_{sk+1} & & & & \\ & 1 & -\beta_{sk+2} & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & -\beta_{sk+j} & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}.$$

Left multiplying (4.7) by A, we get:

$$A \bar{V}_k R'_{k,j} = A \bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} A \bar{V}_{k-1} p'_{k-1,s-1} e_1'^T. \quad (4.8)$$

Define the following matrices:

$$\Lambda_{k,j} = \begin{bmatrix} \alpha_{sk} & & & & \\ & \alpha_{sk+1} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \alpha_{sk+j} \end{bmatrix},$$

$$L_{k,j} = \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \end{bmatrix}.$$

It is possible to write (4.3) in this way:

$$B_k P'_{k,j} = R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} r'_{k,j+1} e_{j+1}'^T, \quad (4.9)$$

here $e_{j+1}'^T = [0, \dots, 0, 1]$. Left-multiplying by V_k and right-multiplying (4.9) by $U_{k,j}$, we have:

$$V_k B_k P'_{k,j} U_{k,j} = V_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}^T. \quad (4.10)$$

Using $A\bar{V}_k = V_k B_k$, $V_k R'_{k,j} = \bar{V}_k R'_{k,j}$, for $j \leq s-1$, we can write (4.10) as follows:

$$A\bar{V}_k P'_{k,j} U_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}^T. \quad (4.11)$$

If we sum (4.8) and (4.11), we get:

$$A\bar{V}_k R'_{k,j} = \bar{V}_k R'_{k,j} T_{k,j} - \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} r'_{k-1,s-1} e_1^T - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}^T, \quad (4.12)$$

for $j=0, \dots, s-1$. This follows from:

$$\beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1^T = \beta_{sk} V_{k-1} B_{k-1} p'_{k-1,s-1} e_1^T$$

Using (4.3)

$$= \beta_{sk} V_{k-1} \frac{1}{\alpha_{s(k-1)+s-1}} (r'_{k-1,s-1} - r'_{k-1,s}) e_1^T$$

$$= \frac{\beta_{sk}}{\alpha_{sk-1}} V_{k-1} (r'_{k-1,s-1}) e_1^T + \frac{\beta_{sk}}{\alpha_{sk-1}} V_{k-1} r'_{k-1,s} e_1^T$$

by (4.6)

$$= \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} (r'_{k-1,s-1}) e_1^T + \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_k r'_{k,0} e_1^T$$

and it should also be noticed that:

$$\bar{V}_k r'_{k,0} = \bar{V}_k R'_{k,j} e_1,$$

here $e_1 = [1, 0, \dots, 0]$. And if we define:

$$T_{k,j} = L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} + e_1 \frac{\beta_{sk}}{\alpha_{sk-1}} e_1^T,$$

we obtain (4.12). If $k=0$, $\frac{\beta_{sk}}{\alpha_{sk-1}} = 0$. Consider now the outside loop and define:

$$\mathcal{V}_k = [\bar{V}_0, \dots, \bar{V}_k],$$

$$\mathcal{R}'_{k,j} = \begin{bmatrix} R'_{0,s-1} & & & \\ & R'_{1,s-1} & & \\ & & \ddots & \\ & & & R'_{k,j} \end{bmatrix},$$

$$\mathcal{T}_{k,j} = \begin{bmatrix} \frac{1}{\alpha_0} & -\frac{\beta_1}{\alpha_0} & & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_1}{\alpha_0} & \ddots & & & \\ & \ddots & \ddots & & & \\ & & & & & \frac{\beta_m}{\alpha_{m-1}} \\ & & & -\frac{1}{\alpha_{m-1}} & \frac{1}{\alpha_m} + \frac{\beta_m}{\alpha_{m-1}} & \\ & & & & & \end{bmatrix}.$$

Using (4.12), we have:

$$AV_k \mathcal{R}'_{k,j} = \mathcal{V}_k \mathcal{R}'_{k,j} \mathcal{T}_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{sk+j+1}. \quad (4.13)$$

□

It is possible to define the residual iterates in this way:

$$\mathcal{R}_m = [r_0, \dots, r_m] = \mathcal{V}_k \mathcal{R}'_{k,j},$$

here $m=sk+j$. And it follows that (4.13) becomes:

$$A\mathcal{R}_m = \mathcal{R}_m \mathcal{T}_m - \frac{1}{\alpha_m} r_{m+1} e'^T_{m+1}.$$

The proof is similar for \tilde{r}_{sk+j} and \tilde{p}_{sk+j} . Tong and Ye obtain the same equation for the BiCG method in their research paper [13].

4.2 Finite precision arithmetic

This section is the same as section (3) in [4] and (5.2.2) and (5.3.3) in [2]. The theorem presented in this section is equal to one given in [13], but this one is for the s-step BiCG method.

In this section we study roundoff errors that are in the s-step BiCG algorithm.

We will use this model of roundoff errors [4], [13]:

$$fl(\alpha x + y) = \alpha x + y + \delta_1, \quad \text{with } |\delta_1| \leq \epsilon 2|\alpha x| + |y| + O(\epsilon^2). \quad (4.14)$$

$$fl(Ax) = Ax + \delta_2, \quad \text{with } |\delta_2| \leq \epsilon N|A||x| + O(\epsilon^2). \quad (4.15)$$

Here $fl(Ax)$ and $fl(\alpha x + y)$ defined calculated values [8], which differ from the ones computed in exact arithmetic. N is “the maximum number of nonzeros per row in A ” [2], while ϵ is “the machine precision unit”, $x, y \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ [13], [2].

Theorem 2. Let ϵ be the machine precision unit and let $r'_{k,j}$, $p'_{k,j}$, $e_{k,j}$, α_m , $p_{k,s}$, $r_{k,s}$, β_m , V_k and B_k be the computed quantities in the finite precision s-step BiCG algorithm. Then:

$$AV_k \mathcal{R}'_{k,j} = \mathcal{V}_k \mathcal{R}'_{k,j} \mathcal{T}_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{sk+j+1} + \epsilon \Delta_{k,j},$$

here \mathcal{V}_k , $\mathcal{R}'_{k,j}$, $\mathcal{T}_{k,j}$ and $\Delta_{k,j}$ are defined as:

$$\mathcal{V}_k = [\bar{V}_0, \dots, \bar{V}_k],$$

$$\mathcal{R}'_{k,j} = \begin{bmatrix} R'_{0,s-1} & & & & \\ & R'_{1,s-1} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & R'_{k,j} \end{bmatrix},$$

$$\mathcal{T}_{k,j} = \begin{bmatrix} \frac{1}{\alpha_0} & -\frac{\beta_1}{\alpha_0} & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_1}{\alpha_0} & \ddots & & \\ & \ddots & \ddots & & \\ & & & \ddots & \\ & & & & \frac{\beta_m}{\alpha_{m-1}} \\ & & & -\frac{1}{\alpha_{m-1}} & \frac{1}{\alpha_m} + \frac{\beta_m}{\alpha_{m-1}} \end{bmatrix},$$

and $\Delta_{k,j} = [\Delta_{0,s-1}, \Delta_{1,s-1}, \dots, \Delta_{k,j}]$.

Proof. In this proof $r'_{k,j}$, $p'_{k,j}$, $\tilde{p}'_{k,j}$, $\tilde{r}'_{k,j}$, α_{sk+j} , $r_{k,s}$, $p_{k,s}$, V_k , B_k are the calculated values in finite precision. We consider the coefficient vectors in the inside loop: $r'_{k,j}$ and $p'_{k,j}$, which are the $(sk+j)$ th iteration. Consider line (20) and (24) of the s-step BiCG algorithm. Line (20) can be written as (4.3). In order to calculate $r'_{k,j}$, from (4.3), in finite arithmetic, we calculate $B_k p'_{k,j}$:

$$\begin{aligned} fl(B_k p'_{k,j}) &= B_k p'_{k,j} + g, \\ |g| &\leq \epsilon(N|B_k||p'_{k,j}|) = \epsilon((2s+1)|B_k||p'_{k,j}|). \end{aligned} \quad (4.16)$$

We used (4.15), here N is “the maximum number of nonzeros” in each row of B_k [2].

$$\begin{aligned} r'_{k,j+1} &= fl(r'_{k,j} - \alpha_m fl(B_k p'_{k,j})) \\ &= r'_{k,j} - \alpha_m fl(B_k p'_{k,j}) + g' \\ &= r'_{k,j} - \alpha_m (B_k p'_{k,j} + g) + g'. \end{aligned} \quad (4.17)$$

Using (4.14) we have:

$$|g'| \leq \epsilon(|r'_{k,j}| + 2|\alpha_m| |fl(B_k p'_{k,j})|).$$

Define:

$$\delta_{r'_{k,j}} = \frac{\alpha_m g + g'}{\epsilon|\alpha_m|},$$

so we can write (4.17) as:

$$r'_{k,j+1} = r'_{k,j} - \alpha_m B_k p'_{k,j} + \epsilon \delta_{r'_{k,j}},$$

Rearranging:

$$\frac{1}{\alpha_m}(r'_{k,j+1} - r'_{k,j}) = -B_k p'_{k,j} + \epsilon \delta_{r'_{k,j}}, \quad (4.18)$$

with

$$|\delta_{r'_{k,j}}| \leq (2s+1)|B_k||p'_{k,j}| + \frac{|r'_{k,j}|}{|\alpha_m|} + 2|B_k p'_{k,j}|. \quad (4.19)$$

The coefficient vector $p'_{k,j}$ in finite arithmetic is:

$$\begin{aligned} p'_{k,j} &= fl(r'_{k,j} + \beta_m p'_{k,j-1}) \\ &= r'_{k,j} + \beta_m p'_{k,j-1} + f. \end{aligned} \quad (4.20)$$

By (4.14), we can write:

$$|f| \leq \epsilon(2|\beta_m||p'_{k,j-1}| + |r'_{k,j}|).$$

If we write $\delta_{p'_{k,j}} = \frac{f}{\epsilon}$, then (4.20) becomes:

$$p'_{k,j} = fl(r'_{k,j} + \beta_m p'_{k,j-1}) = r'_{k,j} + \beta_m p'_{k,j-1} + \epsilon \delta_{p'_{k,j}}, \quad (4.21)$$

with

$$|\delta_{p'_{k,j}}| \leq |r'_{k,j}| + 2|\beta_m||p'_{k,j-1}|.$$

(4.18) can be written as:

$$B_k p'_{k,j} = \frac{1}{\alpha_m}(r'_{k,j} - r'_{k,j+1}) + \epsilon \delta_{r'_{k,j}},$$

we can write (4.21) as:

$$r'_{k,j} = p'_{k,j} - \beta_m p'_{k,j-1} + \epsilon \delta_{p'_{k,j}}. \quad (4.22)$$

If we left-multiply (4.22) by \bar{V}_k , we have:

$$\bar{V}_k r'_{k,j} = \bar{V}_k p'_{k,j} - \beta_m \bar{V}_k p'_{k,j-1} + \epsilon \bar{V}_k \delta_{p'_{k,j}}.$$

It should be noted that $p'_{k,-1}$ is not stated, so the equation is true from $j = 1$.

For $r'_{k,0}$ and $p'_{k,0}$ we have:

$$\begin{aligned} \bar{V}_k r'_{k,0} &= fl(V_{k-1} r'_{k-1,s}) \\ &= V_{k-1} r'_{k-1,s} + \epsilon \phi_{k-1}^r. \end{aligned} \quad (4.23)$$

$$\begin{aligned} \bar{V}_k p'_{k,0} &= fl(V_{k-1} p'_{k-1,s}) \\ &= V_{k-1} p'_{k-1,s} + \epsilon \phi_{k-1}^p. \end{aligned} \quad (4.24)$$

Using (4.15) we have:

$$\begin{aligned} |\phi_{k-1}^r| &\leq (2s+1)|V_{k-1}||r'_{k-1,s}|, \\ |\phi_{k-1}^p| &\leq (2s+1)|V_{k-1}||p'_{k-1,s}|. \end{aligned}$$

Here $(2s+1)$ are “the maximum number of nonzeros” in each row of V_{k-1} [2].

By (4.22) we have:

$$r'_{k-1,s} = p'_{k-1,s} - \beta_{s(k-1)+s}p'_{k-1,s-1} + \epsilon\delta_{p'_{k-1,s}}.$$

We can write (4.23) as follows:

$$\begin{aligned} \bar{V}_k r'_{k,0} &= V_{k-1} r'_{k-1,s} + \epsilon\phi_{k-1}^r, \\ &\text{using (4.22)} \\ &= V_{k-1}(p'_{k-1,s} - \beta_{s(k-1)+s}p'_{k-1,s-1} + \epsilon\delta_{p'_{k-1,s}}) + \epsilon\phi_{k-1}^r \\ &= V_{k-1}p'_{k-1,s} - \beta_{sk}V_{k-1}p'_{k-1,s-1} + \epsilon V_{k-1}\delta_{p'_{k-1,s}} + \epsilon\phi_{k-1}^r, \\ &\text{by (4.24)} \\ &= \bar{V}_k p'_{k,0} - \epsilon\phi_{k-1}^p - \beta_{sk}\bar{V}_k p'_{k-1,s-1} + \epsilon V_{k-1}\delta_{p'_{k-1,s}} + \epsilon\phi_{k-1}^r, \\ &\text{Rearranging} \\ &= \bar{V}_k p'_{k,0} - \beta_{sk}\bar{V}_k p'_{k-1,s-1} + \epsilon(V_{k-1}\delta_{p'_{k-1,s}} + \phi_{k-1}^r - \phi_{k-1}^p). \end{aligned} \tag{4.25}$$

Consider (4.7) and define:

$$\Delta_{R'_{k,j}} = [\delta_{r'_{k,0}}, \dots, \delta_{r'_{k,j}}], \Delta_{P'_{k,j}} = [0_{2s+1}, \delta_{p'_{k,1}}, \dots, \delta_{p'_{k,j}}].$$

Computing (4.7) in finite arithmetic, we will have:

$$\begin{aligned} \bar{V}_k R'_{k,j} &= \bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} \bar{V}_k p'_{k-1,s-1} e_1'^T + \\ &\epsilon \bar{V}_k \Delta_{P'_{k,j}} + \epsilon (V_{k-1} \delta_{p'_{k-1,s}} + \phi_{k-1}^r - \phi_{k-1}^p) e_1'^T. \end{aligned}$$

If we multiply from the left by A , it drives us to:

$$\begin{aligned} A\bar{V}_k R'_{k,j} &= A\bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} A\bar{V}_k p'_{k-1,s-1} e_1'^T + \\ &\epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1} \delta_{p'_{k-1,s}} + \phi_{k-1}^r - \phi_{k-1}^p) e_1'^T. \end{aligned} \tag{4.26}$$

Consider (4.9) and compute it in finite arithmetic, it becomes:

$$B_k P'_{k,j} = R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} r'_{k,j+1} e_{j+1}'^T + \epsilon \Delta_{R'_{k,j}}.$$

If we multiply from the left by V_k , we get:

$$V_k B_k P'_{k,j} = V_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon V_k \Delta_{R'_{k,j}}, \tag{4.27}$$

for $j \leq s - 1$. We have to consider also the roundoff errors of the s-step bases, which means the errors made during the calculation of the bases. In finite precision (2.9) becomes:

$$v_{k,i+1}^p = \frac{1}{\gamma_i}(A - a_i I)v_{k,i}^p - \frac{\beta_{i-1}}{\gamma_i}v_{k,i-1}^p + \epsilon\delta_{v_{k,i+1}^p},$$

we can rewrite it in this way:

$$Av_{k,i}^p = \gamma_i v_{k,i+1}^p + a_i v_{k,i}^p + \beta_{i-1} v_{k,i-1}^p - \epsilon\gamma_i \delta_{v_{k,i+1}^p}.$$

here, using (4.14) and (4.15), we obtain:

$$|\delta_{v_{k,i+1}^p}| \leq \frac{1}{|\gamma_i|}((N+2)|A||v_{k,i}^p| + 3|a_i||v_{k,i}^p| + 2|\beta_{i-1}||v_{k,i-1}^p|).$$

Here N is “the maximum number of nonzeros” in each row of A [2]. In a similar way we can compute $v_{k,i+1}^r$ in finite precision arithmetic.

From chapter 2, we know that:

$$A\bar{V}_k = V_k B_k.$$

Computing it in finite precision, it becomes:

$$\begin{aligned} A\bar{V}_k &= V_k B_k + \epsilon\Delta_{V_k}, \\ \text{here: } |\Delta_{V_k}| &\leq (3+N)|A||\bar{V}_k| + 4|V_k||B_k|, \end{aligned} \quad (4.28)$$

here N is “the maximum number of nonzeros per row over all rows of A” [2], and Δ_{V_k} indicates the roundoff error [2]. The equation in (4.28) can be written as:

$$A\bar{V}_k - \epsilon\Delta_{V_k} = V_k B_k,$$

which means that it is possible to rearrange (4.27) in this way:

$$(A\bar{V}_k - \epsilon\Delta_{V_k})P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon V_k \Delta_{R'_{k,j}},$$

it follows that:

$$A\bar{V}_k P'_{k,j} - \epsilon\Delta_{V_k} P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon V_k \Delta_{R'_{k,j}},$$

which can be written as:

$$A\bar{V}_k P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon V_k \Delta_{R'_{k,j}} + \epsilon\Delta_{V_k} P'_{k,j},$$

and multiplying from the right by $U_{k,j}$:

$$A\bar{V}_k P'_{k,j} U_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon(V_k \Delta_{R'_{k,j}} + \Delta_{V_k} P'_{k,j}) U_{k,j}. \quad (4.29)$$

If we sum (4.26) and (4.29), it follows that:

$$\begin{aligned} A\bar{V}_k R'_{k,j} + A\bar{V}_k P'_{k,j} U_{k,j} &= A\bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1^T + \\ &\quad \epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1} \delta_{p'_{k-1,s}} + \phi_{k-1}^r - \phi_{k-1}^p) e_1^T + \\ &\quad \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}^T + \epsilon(V_k \Delta_{R'_{k,j}} + \Delta_{V_k} P'_{k,j}) U_{k,j}. \end{aligned}$$

After computation we have:

$$\begin{aligned} A\bar{V}_k R'_{k,j} &= -\beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1^T + \\ &\quad \epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1} \delta_{p'_{k-1,s}} + \phi_{k-1}^r - \phi_{k-1}^p) e_1^T + \\ &\quad \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}^T + \epsilon(V_k \Delta_{R'_{k,j}} + \Delta_{V_k} P'_{k,j}) U_{k,j}. \end{aligned} \quad (4.30)$$

Using (4.28) we have:

$$\beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1^T = \beta_{sk} V_{k-1} B_{k-1} p'_{k-1,s-1} e_1^T + \beta_{sk} \Delta_{V_{k-1}} p'_{k-1,s-1} e_1^T.$$

By

$$\begin{aligned} B_{k-1} p'_{k-1,s-1} &= \frac{1}{\alpha_{s(k-1)+s-1}} (r'_{k-1,s-1} - r'_{k-1,s}) + \epsilon \delta_{r'_{k-1,s-1}} \\ &= \frac{1}{\alpha_{sk-1}} (r'_{k-1,s-1} - r'_{k-1,s}) + \epsilon \delta_{r'_{k-1,s-1}}, \end{aligned}$$

it follows that:

$$\beta_{sk} V_{k-1} \left(\frac{1}{\alpha_{sk-1}} (r'_{k-1,s-1} - r'_{k-1,s}) + \epsilon \delta_{r'_{k-1,s-1}} \right) e_1^T + \beta_{sk} \Delta_{V_{k-1}} p'_{k-1,s-1} e_1^T,$$

using (4.23) we will have:

$$= \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} r'_{k-1,s-1} e_1^T - \frac{\beta_{sk}}{\alpha_{sk-1}} (\bar{V}_k r'_{k,0} - \epsilon \phi_{k-1}^T) e_1^T + \epsilon \beta_{sk} V_{k-1} \delta_{r'_{k-1,s-1}} e_1^T + \beta_{sk} \Delta_{V_{k-1}} p'_{k-1,s-1} e_1^T.$$

So we can write (4.30) as:

$$A\bar{V}_k R'_{k,j} = V_k R'_{k,j} T_{k,j} - \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} r'_{k-1,s-1} e_1^T - \frac{1}{\alpha_{sk+j}} V_k r'_{k,j+1} e_{j+1}^T + \epsilon \Delta_{k,j}.$$

With:

$$\begin{aligned} \Delta_{k,j} &= (A\bar{V}_k \Delta_{P'_{k,j}} + A V_{k-1} \delta_{p'_{k-1,s}} e_1^T) \\ &\quad + (V_k \Delta_{R'_{k,j}} U_{k,j} - \beta_{sk} V_{k-1} \delta_{r'_{k-1,s-1}} e_1^T) + (\Delta_{V_k} P'_{k,j} U_{k,j} - \\ &\quad + \beta_{sk} \Delta_{V_{k-1}} p'_{k-1,s-1} e_1^T) + (A(\phi_{k-1}^r - \phi_{k-1}^p) - \frac{\beta_{sk}}{\alpha_{sk-1}} \phi_{k-1}^r) e_1^T. \end{aligned} \quad (4.31)$$

If we define $\Delta_{k,j} = [\delta_{sk}, \dots, \delta_{sk+j}]$, we have that for $j > 0$, the $(sk + j + 1)$ th column of $\Delta_{k,j}$ is

$$\delta_{sk+j} = A\bar{V}_k \delta_{p'_{k,j}} + V_k \delta_{r'_{k,j}} - \beta_{sk+j} V_k \delta_{r'_{k,j-1}} + \Delta_{V_k} r'_{k,j}. \quad (4.32)$$

If we use the norm in (4.32), we will have:

$$\begin{aligned}
|\delta_{sk+j}| &\leq |A||\bar{V}_k||\delta_{p'_{k,j}}| + |V_k||\delta_{r'_{k,j}}| + |\beta_{sk+j}||V_k||\delta_{r'_{k,j-1}}| + |\Delta_{V_k}||r'_{k,j}| \\
&\text{here using (4.28) we have:} \\
&\leq |A||\bar{V}_k|(|r'_{k,j}| + 2|\beta_{sk+j}||p'_{k,j-1}|) + |V_k|((2s+1)|B_k||p'_{k,j}| + \frac{|r'_{k,j}|}{|\alpha_{sk+j}|} + 2|B_k p'_{k,j}|) + \\
&|\beta_{sk+j}||V_k|((2s+1)|B_k||p'_{k,j-1}| + \frac{|r'_{k,j-1}|}{|\alpha_{sk+j-1}|} + 2|B_k p'_{k,j-1}|) + ((3+N)|A||\bar{V}_k| + 4|V_k||B_k|)|r'_{k,j}|.
\end{aligned}$$

By the following inequalities:

$$\begin{aligned}
|\beta_{sk+j}p'_{k,j-1}| &\leq |p'_{k,j}| + |r'_{k,j}| + O(\epsilon), \\
|r'_{k,j-1}| &\leq |r'_{k,j}| + |\alpha_{sk+j-1}||B_k p'_{k,j-1}| + O(\epsilon),
\end{aligned}$$

we arrive at:

$$\begin{aligned}
&\leq |A||\bar{V}_k|(|r'_{k,j}| + 2(|p'_{k,j}| + |r'_{k,j}|)) + |V_k|((2s+1)|B_k||p'_{k,j}| + \frac{|r'_{k,j}|}{|\alpha_{sk+j}|} + 2|B_k p'_{k,j}|) \\
&+ (2s+1)|V_k||B_k|(|p'_{k,j}| + |r'_{k,j}|) + \frac{|\beta_{sk+j}|}{|\alpha_{sk+j-1}|}|V_k|(|r'_{k,j}| + |\alpha_{sk+j-1}||B_k p'_{k,j-1}|) \\
&+ 2|V_k||B_k|(|p'_{k,j}| + |r'_{k,j}|) + ((3+N)|A||\bar{V}_k| + 4|V_k||B_k|)|r'_{k,j}| \\
&\leq |A||\bar{V}_k|(|r'_{k,j}| + 2(|p'_{k,j}| + |r'_{k,j}|)) + (2s+1)|V_k||B_k||p'_{k,j}| + |V_k|\frac{|r'_{k,j}|}{|\alpha_{sk+j}|} + 2|V_k||B_k p'_{k,j}| \\
&+ (2s+1)|V_k||B_k|(|p'_{k,j}| + |r'_{k,j}|) + |V_k|\frac{|\beta_{sk+j}|}{|\alpha_{sk+j-1}|}|r'_{k,j}| + |V_k||B_k|(|p'_{k,j}| + |r'_{k,j}|) \\
&+ 2|V_k||B_k|(|p'_{k,j}| + |r'_{k,j}|) + ((3+N)|A||\bar{V}_k| + 4|V_k||B_k|)|r'_{k,j}| \\
&\leq (2|A||\bar{V}_k| + (4s+7)|V_k||B_k|)|p'_{k,j}| + ((N+6)|A||\bar{V}_k| + (2s+8)|V_k||B_k| + (\frac{1}{|\alpha_{sk+j}|} + \frac{|\beta_{sk+j}|}{|\alpha_{sk+j-1}|})|V_k|)|r'_{k,j}|.
\end{aligned}$$

So we have that:

$$\begin{aligned}
|\delta_{sk+j}| &\leq (2|A||\bar{V}_k| + (4s+7)|V_k||B_k|)|p'_{k,j}| \\
&+ ((N+6)|A||\bar{V}_k| + (2s+8)|V_k||B_k| + (\frac{1}{|\alpha_{sk+j}|} + \frac{|\beta_{sk+j}|}{|\alpha_{sk+j-1}|})|V_k|)|r'_{k,j}|.
\end{aligned}$$

For $j = 0$ we have:

$$\begin{aligned}
\delta_{sk} &= AV_{k-1}\delta_{p'_{k-1,s}} + V_k\delta_{r'_{k,0}} - \beta_{sk}V_{k-1}\delta_{r'_{k-1,s-1}} + \Delta_{V_k}p'_{k,0} - \beta_{sk}\Delta_{V_{k-1}}p'_{k-1,s-1} + \\
&(A(\phi_{k-1}^r - \phi_{k-1}^p) - \frac{\beta_{sk}}{\alpha_{sk-1}}\phi_{k-1}^r).
\end{aligned}$$

Using the norm:

$$\begin{aligned}
|\delta_{sk}| &\leq ((N+2s+7)|A||V_{k-1}| + (2s+8)|V_{k-1}||B_{k-1}|)|r'_{k-1,s}| \\
&+ (\frac{1}{|\alpha_{sk}|} + (2s+2)\frac{|\beta_{sk}|}{|\alpha_{sk-1}|})|V_{k-1}||r'_{k-1,s}| \\
&+ ((2N+4s+16)|A||V_{k-1}| + (6s+22)|V_{k-1}||B_{k-1}|)|p'_{k-1,s}|.
\end{aligned}$$

So from $j = 0$ to $sk + j$, we have:

$$AV_k \mathcal{R}'_{k,j} = V_k \mathcal{R}'_{k,j} \mathcal{T}_{k,j} - \frac{1}{\alpha_{sk+j}} V_k r'_{k,j+1} e'_{sk+j+1}{}^T + \epsilon \Delta_{k,j},$$

here $\Delta_{k,j} = [\Delta_{0,s-1}, \Delta_{1,s-1}, \dots, \Delta_{k,j}]$. □

In the s-step finite precision arithmetic the calculations of the basis and the change of the basis generate errors, as we can see in (4.31), where the third and the fourth element are the errors caused by the calculations of the Krylov basis and by changing the basis [2]. We can see in Chapter 3 that when s is a large value, the number of iterations made by the algorithm, before reaching the convergence, can be a big number. This is caused by roundoff errors in calculations of the basis that can affect the convergence [2], [5].

Chapter 5

Summary

In the thesis we studied how to go from the BiCG algorithm to an s -step BiCG method. We reviewed the Krylov subspaces and the Chebyshev polynomials in order to use the monomial basis and the Chebyshev basis. We compared the two bases through numerical examples, and we saw that for large matrices the monomial basis is good for small values of s , but as s becomes bigger the Chebyshev basis gives better results. Although we have to consider that for using the Chebyshev basis we have to find first the spectrum of the main matrix A and then we have to assign the values of the ellipse, which should be close to the eigenvalues. We compared also the BiCG method with the s -step BiCG method and after studying the roundoff errors of the s -step BiCG method, it seems that the BiCG method gives better results.

Chapter 6

MATLAB codes

The following codes are based on the algorithms written in [13] for the BiCG method, and in [4] and [2] for the s-step BiCG method. The stopping conditions in rows {10, 35}, for the BiCG code and which are also the same for the s-step BiCG codes, are taken from [7]. It should be noticed that the number of iterations of the outside loop starts at $k = 1$, while the number of iterations in the inside loop begins at $j = 0$ and it will end at $s - 1$, performing each time a block of s iterations. The reader, to see the results shown in chapter 3, should write in the command window the following instructions. For the first example in the BiCG method:

```
1 deltax=1/5;
2 A=deltax*[1 0 0 0 ; -1 1 0 0 ; 0 -1 1 0 ; 0 0 -1 1];
3 b1=[1 0 1 0]';
4 x1=[0 0 0 0]';
5 tol=10^-6;
6 rt1=b1-A*x1;
7 n=4;
8 [x1,k,r1,tr1]=classbicg(A,b1,x1,tol,rt1,n)
```

In this case, the code will collapse at $k=4$. For avoiding this problem, we can replace lines {6, 7} as:

```
1 rt1=[1 1 1 1]';
2 n=89;
```

For the second example, using the instructions given in [1] and the function in line (2) given in [15], the reader should digit the following:

```
1 filename='cdde1.mtx';
2 [A, rows, cols, entries]=mmread(filename);
3 x1=zeros(961,1);
4 b1=A*ones(961,1);
5 b1=b1/sqrt(961);
6 I=sparse(eye(961));
```



```

7 n=961;
8 tol=10^-10;
9 r1=b1-A*x1;
10 [x1,k,r1,tr1]=classbicg(A,b1,x1,tol,r1,n)

```

The command windows for s-step BiCG codes are:

```

1 %1) example
2 deltax=1/5;
3 A=deltax*[1 0 0 0 ; -1 1 0 0 ; 0 -1 1 0 ; 0 0 -1 1];
4 b1=[1 0 1 0]';
5 x1=[0 0 0 0]';
6 tol=10^-6;
7 I=eye(4);
8 a=7.9888;
9 d=7.98;
10 b=0.010;
11 [x1,x11,k,kk,r1,r11,tr1,tr11]=step4bicg(A,b1,x1,I,a,b,d,tol)
12 %[x1,x11,k,kk,r1,r11,tr1,tr11]=step8bicg(A,b1,x1,I,a,b,d,tol)
13 %[x1,x11,k,kk,r1,r11,tr1,tr11]=step16bicg(A,b1,x1,I,a,b,d,tol)

```

The function from line (2) is taken from [15].

```

1 %2) example
2 filename='cddel.mtx';
3 [A,rows,cols,entries]=mmread(filename);
4 x1=zeros(961,1);
5 b1=A*ones(961,1);
6 b1=b1/sqrt(961);
7 I=sparse(eye(961));
8 a=8;
9 d=7;
10 b=0.010;
11 tol=10^-10;
12 [x1,x11,k,kk,r1,r11,tr1,tr11]=step4bicg(A,b1,x1,I,a,b,d,tol)
13 %[x1,x11,k,kk,r1,r11,tr1,tr11]=step8bicg(A,b1,x1,I,a,b,d,tol)
14 %[x1,x11,k,kk,r1,r11,tr1,tr11]=step16bicg(A,b1,x1,I,a,b,d,tol)

```

BiCG code

```

1 function[x1,k,r1,tr1]=classbicg(A,b1,x1,tol,r1,n)
2 normb=norm(b1); %normalizing b1
3 r1=b1-A*x1; %initializing the residual vector
4 normr=norm(r1); %normalizing the residual vector
5 tr1=b1-A*x1; %not normalized true residual
6 p1=r1; %start search direction
7 pt1=r1; %start search direction tilde
8 k=1;
9 rhol=rt1'*r1;
10 while (norm(r1)/normb>tol)
11     sigma=pt1'*A*p1;
12     alphas=rhol/sigma;
13     r2=r1-alphas*A*p1; %residual vector
14     x2=x1+alphas*p1;
15     rt2=rt1-alphas*A'*pt1; %residual tilde vector

```

```

16     rho2=rt2'*r2; %rho
17     beta2=rho2/rho1;
18     p2=r2+beta2*p1; %search direction
19     pt2=rt2+beta2*pt1; %search direction tilde vector
20     valuex1(:,k)=x1;
21     valuer1(:,k)=r1;
22     u1(:,k)=norm(valuer1(:,k)/norm(b1));
23     %updating
24     p1=p2;
25     pt1=pt2;
26     r1=r2;
27     rt1=rt2;
28     x1=x2;
29     rho1=rho2;
30     tr1=b1-A*x1;
31     k=k+1;
32     valuex1(:,k)=x1;
33     valuer1(:,k)=r1;
34     u1(:,k)=norm(valuer1(:,k)/norm(b1));
35     if k==n
36         break
37     end
38 end
39 semilogy(u1, '-o')
40 xlabel('Number of Iterations')
41 ylabel('2-Norm Residual')

```

s-step BiCG code, s=4

```

1 function[x1,x11,k,kk,r1,r11,tr1,tr11]=step4bicg(A,b1,x1,I,a,b,d,tol)
2 %residual vectors
3 r1=b1-A*x1;
4 x11=x1;
5 r11=b1-A*x11;
6 %not normalized true residual vector
7 tr1=b1-A*x1;
8 x11=x1;
9 tr11=b1-A*x11;
10 %search directions
11 p1=r1;
12 p11=r11;
13 ppt1=p11;
14 rrt1=r11;
15 pt1=p1;
16 rt1=r1;
17 %coefficient vectors for when we use the Chebyshev basis
18 p_k0=[1 zeros(1,8)]';
19 r_k0=[zeros(1,5) 1 zeros(1,3)]';
20 e_k0=[zeros(1,9)];
21 pt_k0=[1 zeros(1,8)]';
22 rt_k0=[zeros(1,5) 1 zeros(1,3)]';
23 %coefficient vectors for when we make use of the Krylov basis
24 pp_k0=[1 zeros(1,8)]';
25 rr_k0=[zeros(1,5) 1 zeros(1,3)]';
26 ee_k0=[zeros(1,9)];
27 ppt_k0=[1 zeros(1,8)]';
28 rrt_k0=[zeros(1,5) 1 zeros(1,3)]';

```

```

29 %the 5 maximum eigenvalues of the matrix A
30 eigs(A);
31 %the minimum eigenvalue of the matrix A
32 eigs(A,1,'smallestab');
33 %
34 c=sqrt(a^2-b^2); %value of the ellipse
35 %values for the Chebyshev case
36 aj=d;
37 g=max(a,b);
38 beta0=c^2/4*g;
39 psi_0=2*g;
40 psi_1=g;
41 %values for the monomial case
42 dd=0;
43 ajj=dd;
44 bbeta0=0;
45 ppsi_0=1;
46 %vectors for the Chebyshev basis
47 vp_k0=p1;
48 vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
49 vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
50 vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
51 vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
52
53 vr_k0=r1;
54 vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
55 vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
56 vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
57
58 vtp_k0=pt1;
59 vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
60 vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
61 vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
62 vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
63
64 vtr_k0=rt1;
65 vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
66 vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
67 vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
68 %vectors for the monomial basis
69 vpp_k0=p11;
70 vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
71 vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
72 vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
73 vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
74
75 vrr_k0=r11;
76 vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
77 vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
78 vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
79
80 vttp_k0=ppt1;
81 vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
82 vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
83 vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
84 vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
85

```

```

86 vttr.k0=rrt1;
87 vttr.k1=(1/ppsi_0)*(A'-dd*I)*vttr.k0;
88 vttr.k2=(1/ppsi_0)*(A'-dd*I)*vttr.k1-(bbeta0/ppsi_0)*vttr.k0;
89 vttr.k3=(1/ppsi_0)*(A'-dd*I)*vttr.k2-(bbeta0/ppsi_0)*vttr.k1;
90 %Chebishev basis
91 V_p=[vp_k0, vp_k1, vp_k2, vp_k3, vp_k4];
92 V_r=[vr_k0, vr_k1, vr_k2, vr_k3];
93 Vt_p=[vtp_k0, vtp_k1, vtp_k2, vtp_k3, vtp_k4];
94 Vt_r=[vtr_k0, vtr_k1, vtr_k2, vtr_k3];
95 V=[V_p, V_r];
96 Vt=[Vt_p, Vt_r];
97 G=Vt'*V;
98 %Monomial basis
99 VV_p=[vpp_k0, vpp_k1, vpp_k2, vpp_k3, vpp_k4];
100 VV_r=[vrr_k0, vrr_k1, vrr_k2, vrr_k3];
101 VVt_p=[vttp_k0, vttp_k1, vttp_k2, vttp_k3, vttp_k4];
102 VVt_r=[vttr_k0, vttr_k1, vttr_k2, vttr_k3];
103 VV=[VV_p, VV_r];
104 VVt=[VVt_p, VVt_r];
105 GG=VVt'*VV;
106 %matrix B_{k} for when we use Chebyshev basis
107 B=[aj beta0 0 0 0 0 0 0 ;...
108     psi_0 aj beta0 0 0 0 0 0 ;...
109     0 psi_1 aj beta0 0 0 0 0 ;...
110     0 0 psi_1 aj 0 0 0 0 ;...
111     0 0 0 psi_1 0 0 0 0 ; ...
112     0 0 0 0 0 aj beta0 0 0 ;...
113     0 0 0 0 0 psi_0 aj beta0 0 ;...
114     0 0 0 0 0 0 psi_1 aj 0;...
115     0 0 0 0 0 0 0 psi_1 0];
116 %matrix B_{k} for when we use monomial basis
117 BB=[ajj bbeta0 0 0 0 0 0 0 ;...
118     ppsi_0 ajj bbeta0 0 0 0 0 0 ;...
119     0 ppsi_0 ajj bbeta0 0 0 0 0 ;...
120     0 0 ppsi_0 ajj 0 0 0 0 ;...
121     0 0 0 ppsi_0 0 0 0 0 ; ...
122     0 0 0 0 0 ajj bbeta0 0 0 ;...
123     0 0 0 0 0 ppsi_0 ajj bbeta0 0 ;...
124     0 0 0 0 0 0 ppsi_0 ajj 0;...
125     0 0 0 0 0 0 0 ppsi_0 0];
126 %loops for when we use the monomial basis
127 s=4;
128 kk=1; %starting value
129 n=2513; %maximum number of iterations for s=4
130 normb=norm(b1);
131 while (norm(r11)/normb>tol)
132     pp_k0=[1 zeros(1,8)]'; %coefficient vectors
133     rr_k0=[zeros(1,5) 1 zeros(1,3)]';
134     ee_k0=[zeros(1,9)];
135     ppt_k0=[1 zeros(1,8)]';
136     rrt_k0=[zeros(1,5) 1 zeros(1,3)]';
137     % inside loop
138     deltt1=rrt_k0'*GG*rr_k0;
139     for j=0 : s-1
140         alphas1=deltt1/(ppt_k0'*GG*BB*pp_k0);
141         ee_k1=ee_k0+alphas1*pp_k0';
142         rr_k1=rr_k0-BB*(alphas1*pp_k0);

```

```

143     rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
144     delt3=rrt_k1'*GG*rr_k1;
145     betaa=delt3/delt1;
146     pp_k1=rr_k1+betaa*pp_k0;
147     ppt_k1=rrt_k1+betaa*ppt_k0;
148     rr_k0=rr_k1;
149     rrt_k0=rrt_k1;
150     pp_k0=pp_k1;
151     ppt_k0=ppt_k1;
152     delt1=delt3;
153     ee_k0=ee_k1;
154     end
155     xmm=VV*ee_k0'+x11;
156     rmm=VV*rr_k0;
157     pmm=VV*pp_k0;
158     rrtm=VVt*rrt_k0;
159     pptm=VVt*ppt_k0;
160     valuer2(:,kk)=r11;
161     u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
162     x11=xmm;
163     r11=rmm;
164     p11=pmm;
165     ppt1=pptm;
166     rrt1=rrtm;
167     tr11=b1-A*x11;
168     vpp_k0=p11;
169     vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
170     vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
171     vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
172     vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
173     vrr_k0=r11;
174     vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
175     vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
176     vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
177     vttp_k0=ppt1;
178     vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
179     vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
180     vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
181     vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
182     vttr_k0=rrt1;
183     vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
184     vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
185     vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
186     %Krylov matrices
187     VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4];
188     VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3];
189     VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3,vttp_k4];
190     VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3];
191     VV=[VV_p, VV_r];
192     VVt=[VVt_p, VVt_r];
193     GG=VVt'*VV;
194     kk=kk+1;
195     valuer2(:,kk)=r11;
196     u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
197     if kk==n
198         break;
199     end

```

```

200
201 end
202 %loops using Chebyshev basis
203 s=4;
204 k=1;
205 n=2513;
206 normb=norm(b1);
207 while (norm(r1)/normb>tol)
208     p_k0=[1 zeros(1,8)]'; %coefficient vectors
209     r_k0=[zeros(1,5) 1 zeros(1,3)]';
210     e_k0=[zeros(1,9)];
211     pt_k0=[1 zeros(1,8)]';
212     rt_k0=[zeros(1,5) 1 zeros(1,3)]';
213     delt=rt_k0'*G*r_k0;
214     for j=0 : s-1
215         alpha=delt/(pt_k0'*G*B*p_k0);
216         e_k1=e_k0+alpha*p_k0';
217         r_k1=r_k0-B*(alpha*p_k0);
218         rt_k1=rt_k0-B*(alpha*pt_k0);
219         delt2=rt_k1'*G*r_k1;
220         beta=delt2/delt;
221         p_k1=r_k1+beta*p_k0;
222         pt_k1=rt_k1+beta*pt_k0;
223         %updating
224         r_k0=r_k1;
225         rt_k0=rt_k1;
226         p_k0=p_k1;
227         pt_k0=pt_k1;
228         delt=delt2;
229         e_k0=e_k1;
230     end
231     xm=V*e_k0'+x1;
232     rm=V*r_k0;
233     pm=V*p_k0;
234     rtm=Vt*rt_k0;
235     ptm=Vt*pt_k0;
236     valuer1(:,k)=r1;
237     u1(:,k)=norm(valuer1(:,k)/norm(b1));
238     x1=xm;
239     r1=rm;
240     p1=pm;
241     pt1=ptm;
242     rt1=rtm;
243     tr1=b1-A*x1;
244     vp_k0=p1;
245     vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
246     vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
247     vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
248     vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
249     vr_k0=r1;
250     vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
251     vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
252     vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
253     vtp_k0=pt1;
254     vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
255     vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
256     vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;

```

```

257     vtp.k4=(1/psi.l)*(A'-d*I)*vtp.k3-(beta0/psi.l)*vtp.k2;
258     vtr.k0=rtl;
259     vtr.k1=(1/psi.0)*(A'-d*I)*vtr.k0;
260     vtr.k2=(1/psi.l)*(A'-d*I)*vtr.k1-(beta0/psi.l)*vtr.k0;
261     vtr.k3=(1/psi.l)*(A'-d*I)*vtr.k2-(beta0/psi.l)*vtr.k1;
262     V.p=[vp.k0, vp.k1, vp.k2, vp.k3, vp.k4];
263     V.r=[vr.k0, vr.k1, vr.k2, vr.k3];
264     Vt.p=[vtp.k0, vtp.k1, vtp.k2, vtp.k3, vtp.k4];
265     Vt.r=[vtr.k0, vtr.k1, vtr.k2, vtr.k3];
266     V=[V.p, V.r];
267     Vt=[Vt.p, Vt.r];
268     G=Vt'*V;
269     k=k+1;
270     valuer1(:,k)=r1;
271     ul(:,k)=norm(valuer1(:,k)/norm(b1));
272     if k==n
273         break
274     end
275 end
276 %plot for both bases
277 semilogy(u1, '-o')
278 xlabel('Number of Iterations')
279 ylabel('2-Norm Residual')
280 hold on
281 semilogy(u2, '-*')
282 legend('Chebyshev Basis s=4 ', 'Monomial Basis s=4')
283 hold off

```

s-step BiCG code, $s=8$

```

1 function[x1,x11,k,kk,r1,r11,tr1,tr11]=step8bicg(A,b1,x1,I,a,b,d,tol)
2 %residual vectors
3 r1=b1-A*x1;
4 x11=x1;
5 r11=b1-A*x11;
6 tr1=b1-A*x1; %not normalized true residual vector
7 x11=x1;
8 tr11=b1-A*x11;
9 %search directions
10 p1=r1;
11 p11=r11;
12 ppt1=p11;
13 rrt1=r11;
14 pt1=p1;
15 rt1=r1;
16 %coefficient vectors for when we make use of the Chebyshev basis
17 p.k0=[1 zeros(1,16)]';
18 r.k0=[zeros(1,9) 1 zeros(1,7)]';
19 e.k0=[zeros(1,17)];
20 pt.k0=[1 zeros(1,16)]';
21 rt.k0=[zeros(1,9) 1 zeros(1,7)]';
22 %coefficients vectors for when we use monomial basis
23 pp.k0=[1 zeros(1,16)]';
24 rr.k0=[zeros(1,9) 1 zeros(1,7)]';
25 ee.k0=[zeros(1,17)];
26 ppt.k0=[1 zeros(1,16)]';
27 rrt.k0=[zeros(1,9) 1 zeros(1,7)]';

```

```

28 %the 5 maximum eigenvalues of the matrix A
29 eigs(A);
30 %the minimum eigenvalue of the matrix A
31 eigs(A,1,'smallestab');
32 c=sqrt(a^2-b^2); %value of the ellipse
33 %values for when we use the Chebyshev basis
34 aj=d;
35 g=max(a,b);
36 beta0=c^2/4*g;
37 psi_0=2*g;
38 psi_1=g;
39 %values for when we use the monomial Basis
40 dd=0;
41 ajj=dd;
42 bbeta0=0;
43 ppsi_0=1;
44 %Vectors for the Chebyshev basis
45 vp_k0=p1;
46 vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
47 vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
48 vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
49 vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
50 vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
51 vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
52 vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
53 vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
54
55 vr_k0=r1;
56 vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
57 vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
58 vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
59 vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
60 vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
61 vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
62 vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
63
64 vtp_k0=pt1;
65 vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
66 vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
67 vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
68 vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
69 vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
70 vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
71 vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
72 vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
73
74 vtr_k0=rtl;
75 vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
76 vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
77 vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
78 vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
79 vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
80 vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
81 vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
82 %vectors for the monomial basis
83 vpp_k0=p11;
84 vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;

```



```

85 vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
86 vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
87 vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
88 vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
89 vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
90 vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
91 vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
92
93 vrr_k0=r11;
94 vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
95 vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
96 vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
97 vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
98 vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
99 vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
100 vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
101
102 vttp_k0=ppt1;
103 vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
104 vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
105 vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
106 vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
107 vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
108 vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
109 vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
110 vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;
111
112 vttr_k0=r11;
113 vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
114 vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
115 vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
116 vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
117 vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
118 vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
119 vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
120 %Chebyshev basis s=8
121 V_p=[vp_k0, vp_k1, vp_k2, vp_k3, vp_k4, vp_k5, vp_k6, vp_k7, vp_k8];
122 V_r=[vr_k0, vr_k1, vr_k2, vr_k3, vr_k4, vr_k5, vr_k6, vr_k7];
123 Vt_p=[vtp_k0, vtp_k1, vtp_k2, vtp_k3, ...
      vtp_k4, vtp_k5, vtp_k6, vtp_k7, vtp_k8];
124 Vt_r=[vtr_k0, vtr_k1, vtr_k2, vtr_k3, vtr_k4, vtr_k5, vtr_k6, vtr_k7];
125 V=[V_p, V_r];
126 Vt=[Vt_p, Vt_r];
127 G=Vt'*V;
128 %Krylov basis s=8
129 VV_p=[vpp_k0, vpp_k1, vpp_k2, vpp_k3, vpp_k4, vpp_k5, vpp_k6, vpp_k7, vpp_k8];
130 VV_r=[vrr_k0, vrr_k1, vrr_k2, vrr_k3, vrr_k4, vrr_k5, vrr_k6, vrr_k7];
131 VVt_p=[vtpp_k0, vttp_k1, vttp_k2, vttp_k3, vttp_k4, vttp_k5, vttp_k6, ...
      vttp_k7, vttp_k8];
132 VVt_r=[vttr_k0, vttr_k1, vttr_k2, vttr_k3, vttr_k4, vttr_k5, vttr_k6, vttr_k7];
133 VV=[VV_p, VV_r];
134 VVt=[VVt_p, VVt_r];
135 GG=VVt'*VV;
136 %matrix B_{k} for when we use the Chebyshev basis
137 B=eye(17);
138 for i=1:17
139     B(i,i)=aj;
140

```

```

141 end
142 for i=1:16
143     B(i,i+1)=beta0;
144     B(i+1,i)=psi_1;
145 end
146 B(2,1)=psi_0;
147 B(11,10)=psi_0;
148 B(17,17)=0;
149 B(9,9)=0;
150 B(16,17)=0;
151 B(8,9)=0;
152 B(10,9)=0;
153 B(9,10)=0;
154 %matrix B_{k} for when we use the monomial basis
155 BB=eye(17);
156 for i=1:17
157     BB(i,i)=ajj;
158 end
159 for i=1:16
160     BB(i,i+1)=bbeta0;
161     BB(i+1,i)=ppsi_0;
162 end
163
164 BB(17,17)=0;
165 BB(9,9)=0;
166 BB(16,17)=0;
167 BB(8,9)=0;
168 BB(10,9)=0;
169 s=8;
170 kk=1; %starting value
171 n=5025; %maximum number for s=8
172 normb=norm(b1)
173 while (norm(r11)/normb>tol)
174     pp_k0=[1 zeros(1,16)]';
175     rr_k0=[zeros(1,9) 1 zeros(1,7)]';
176     ee_k0=[zeros(1,17)];
177     ppt_k0=[1 zeros(1,16)]';
178     rrt_k0=[zeros(1,9) 1 zeros(1,7)]';
179     deltt1=rrt_k0'*GG*rr_k0;
180     for j=0 : s-1
181         alpha1=deltt1/(ppt_k0'*GG*BB*pp_k0);
182         ee_k1=ee_k0+alpha1*pp_k0';
183         rr_k1=rr_k0-BB*(alpha1*pp_k0);
184         rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
185         deltt3=rrt_k1'*GG*rr_k1;
186         betaa=deltt3/deltt1;
187         pp_k1=rr_k1+betaa*pp_k0;
188         ppt_k1=rrt_k1+betaa*ppt_k0;
189         rr_k0=rr_k1;
190         rrt_k0=rrt_k1;
191         pp_k0=pp_k1;
192         ppt_k0=ppt_k1;
193         deltt1=deltt3;
194         ee_k0=ee_k1;
195     end
196     xmm=VV*ee_k0'+x11;
197     rmm=VV*rr_k0;

```

```

198 pmm=VV*pp_k0;
199 rrtm=VVt*rrt_k0;
200 pptm=VVt*ppt_k0;
201 valuer2(:,kk)=r11;
202 u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
203 x11=xmm;
204 r11=rmm;
205 p11=pmm;
206 ppt1=pptm;
207 rrt1=rrtm;
208 tr11=b1-A*x11; %true residual
209 vpp_k0=p11;
210 vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
211 vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
212 vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
213 vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
214 vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
215 vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
216 vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
217 vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
218
219 vrr_k0=r11;
220 vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
221 vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
222 vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
223 vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
224 vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
225 vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
226 vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
227
228 vtttp_k0=ppt1;
229 vtttp_k1=(1/ppsi_0)*(A'-dd*I)*vtttp_k0;
230 vtttp_k2=(1/ppsi_0)*(A'-dd*I)*vtttp_k1-(bbeta0/ppsi_0)*vtttp_k0;
231 vtttp_k3=(1/ppsi_0)*(A'-dd*I)*vtttp_k2-(bbeta0/ppsi_0)*vtttp_k1;
232 vtttp_k4=(1/ppsi_0)*(A'-dd*I)*vtttp_k3-(bbeta0/ppsi_0)*vtttp_k2;
233 vtttp_k5=(1/ppsi_0)*(A'-dd*I)*vtttp_k4-(bbeta0/ppsi_0)*vtttp_k3;
234 vtttp_k6=(1/ppsi_0)*(A'-dd*I)*vtttp_k5-(bbeta0/ppsi_0)*vtttp_k4;
235 vtttp_k7=(1/ppsi_0)*(A'-dd*I)*vtttp_k6-(bbeta0/ppsi_0)*vtttp_k5;
236 vtttp_k8=(1/ppsi_0)*(A'-dd*I)*vtttp_k7-(bbeta0/ppsi_0)*vtttp_k6;
237
238 vttr_k0=rrt1;
239 vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
240 vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
241 vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
242 vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
243 vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
244 vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
245 vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
246 vttr_k8=(1/ppsi_0)*(A'-dd*I)*vttr_k7-(bbeta0/ppsi_0)*vttr_k6;
247
248 VV_p=[vpp_k0, vpp_k1, vpp_k2, vpp_k3, vpp_k4, vpp_k5, vpp_k6, vpp_k7, vpp_k8];
249 VV_r=[vrr_k0, vrr_k1, vrr_k2, vrr_k3, vrr_k4, vrr_k5, vrr_k6, vrr_k7];
250 VVt_p=[vtttp_k0, vtttp_k1, vtttp_k2, vtttp_k3, ...
        vtttp_k4, vtttp_k5, vtttp_k6, vtttp_k7, vtttp_k8];
251 VVt_r=[vttr_k0, vttr_k1, vttr_k2, vttr_k3, vttr_k4, vttr_k5, vttr_k6, vttr_k7];
252 VV=[VV_p, VV_r];
253 VVt=[VVt_p, VVt_r];

```

```

254 GG=Vt'*VV;
255 kk=kk+1;
256 valuer2(:,kk)=r11;
257 u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
258 if kk==n
259     break;
260 end
261
262 end
263 s=8
264 k=1;
265 n=5025
266 normb=norm(b1);
267 while (norm(r1)/normb>tol)
268     p_k0=[1 zeros(1,16)]';
269     r_k0=[zeros(1,9) 1 zeros(1,7)]';
270     e_k0=[zeros(1,17)];
271     pt_k0=[1 zeros(1,16)]';
272     rt_k0=[zeros(1,9) 1 zeros(1,7)]';
273     delt=rt_k0'*G*r_k0;
274     for j=0 : s-1
275         alpha=delt/(pt_k0'*G*B*p_k0);
276         e_k1=e_k0+alpha*p_k0';
277         r_k1=r_k0-B*(alpha*p_k0);
278         rt_k1=rt_k0-B*(alpha*pt_k0);
279         delt2=rt_k1'*G*r_k1;
280         beta=delt2/delt;
281         p_k1=r_k1+beta*p_k0;
282         pt_k1=rt_k1+beta*pt_k0;
283         %updating
284         r_k0=r_k1;
285         rt_k0=rt_k1;
286         p_k0=p_k1;
287         pt_k0=pt_k1;
288         delt=delt2;
289         e_k0=e_k1;
290     end
291     xm=V*e_k0'+x1;
292     rm=V*r_k0;
293     pm=V*p_k0;
294     rtm=Vt*rt_k0;
295     ptm=Vt*pt_k0;
296     valuer1(:,k)=r1;
297     u1(:,k)=norm(valuer1(:,k)/norm(b1));
298     x1=xm;
299     r1=rm;
300     p1=pm;
301     pt1=ptm;
302     rt1=rtm;
303     tr1=b1-A*x1;
304     vp_k0=p1;
305     vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
306     vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
307     vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
308     vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
309     vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
310     vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;

```

```

311 vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
312 vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
313
314 vr_k0=r1;
315 vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
316 vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
317 vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
318 vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
319 vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
320 vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
321 vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
322
323 vtp_k0=pt1;
324 vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
325 vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
326 vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
327 vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
328 vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
329 vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
330 vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
331 vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
332
333 vtr_k0=rt1;
334 vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
335 vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
336 vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
337 vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
338 vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
339 vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
340 vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
341
342 V_p=[vp_k0, vp_k1, vp_k2, vp_k3, vp_k4, vp_k5, vp_k6, vp_k7, vp_k8];
343 V_r=[vr_k0, vr_k1, vr_k2, vr_k3, vr_k4, vr_k5, vr_k6, vr_k7];
344 Vt_p=[vtp_k0, vtp_k1, vtp_k2, vtp_k3, ...
        vtp_k4, vtp_k5, vtp_k6, vtp_k7, vtp_k8];
345 Vt_r=[vtr_k0, vtr_k1, vtr_k2, vtr_k3, vtr_k4, vtr_k5, vtr_k6, vtr_k7];
346 V=[V_p, V_r];
347 Vt=[Vt_p, Vt_r];
348 G=Vt'*V;
349 k=k+1;
350 valuer1(:,k)=r1;
351 u1(:,k)=norm(valuer1(:,k)/norm(b1));
352 if k==n
353     break
354 end
355
356 end
357 %plot for both bases
358 semilogy(u1, '-o')
359 xlabel('Number of Iterations')
360 ylabel('2-Norm Residual')
361 hold on
362 semilogy(u2, '-*')
363 legend('Chebyshev Basis s=8 ', 'Monomial Basis s=8')
364 hold off

```

s-step BiCG code, s=16

```

1 function[x1,x11,k,kk,r1,r11,tr1,tr11]=step16bicg(A,b1,x1,I,a,b,d,tol)
2 %residual vectors
3 r1=b1-A*x1;
4 x11=x1;
5 r11=b1-A*x11;
6 %not normalized true residual vector
7 tr1=b1-A*x1;
8 x11=x1;
9 tr11=b1-A*x11;
10 %search directions
11 p1=r1;
12 p11=r11;
13 ppt1=p1;
14 rrt1=r11;
15 pt1=p1;
16 rt1=r1;
17 %coefficient vectors for when we use Chebyshev basis
18 p_k0=[1 zeros(1,32)]';
19 r_k0=[zeros(1,17) 1 zeros(1,15)]';
20 e_k0=zeros(1,33);
21 pt_k0=[1 zeros(1,32)]';
22 rt_k0=[zeros(1,17) 1 zeros(1,15)]';
23 %coefficients vectors for when we make use of monomial basis
24 pp_k0=[1 zeros(1,32)]';
25 rr_k0=[zeros(1,17) 1 zeros(1,15)]';
26 ee_k0=zeros(1,33);
27 ppt_k0=[1 zeros(1,32)]';
28 rrt_k0=[zeros(1,17) 1 zeros(1,15)]';
29 %the 5 maximum eigenvalues of the matrix A
30 eigs(A);
31 %the minimum eigenvalue of the matrix A
32 eigs(A,1,'smallestab');
33 c=sqrt(a^2-b^2); %value of the ellipse
34 %values for when we use the Chebyshev basis
35 aj=d;
36 g=max(a,b);
37 beta0=c^2/4*g;
38 psi_0=2*g;
39 psi_1=g;
40 %values for when we use the monomial Basis
41 dd=0;
42 ajj=dd;
43 bbeta0=0;
44 ppsi_0=1;
45 %vectors for the Chebyshev basis
46 vp_k0=p1;
47 vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
48 vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
49 vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
50 vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
51 vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
52 vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
53 vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
54 vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
55 vp_k9=(1/psi_1)*(A-d*I)*vp_k8-(beta0/psi_1)*vp_k7;
56 vp_k10=(1/psi_1)*(A-d*I)*vp_k9-(beta0/psi_1)*vp_k8;
57 vp_k11=(1/psi_1)*(A-d*I)*vp_k10-(beta0/psi_1)*vp_k9;

```

```

58 vp_k12=(1/psi_1)*(A-d*I)*vp_k11-(beta0/psi_1)*vp_k10;
59 vp_k13=(1/psi_1)*(A-d*I)*vp_k12-(beta0/psi_1)*vp_k11;
60 vp_k14=(1/psi_1)*(A-d*I)*vp_k13-(beta0/psi_1)*vp_k12;
61 vp_k15=(1/psi_1)*(A-d*I)*vp_k14-(beta0/psi_1)*vp_k13;
62 vp_k16=(1/psi_1)*(A-d*I)*vp_k15-(beta0/psi_1)*vp_k14;
63 %
64 vr_k0=r1;
65 vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
66 vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
67 vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
68 vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
69 vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
70 vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
71 vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
72 vr_k8=(1/psi_1)*(A-d*I)*vr_k7-(beta0/psi_1)*vr_k6;
73 vr_k9=(1/psi_1)*(A-d*I)*vr_k8-(beta0/psi_1)*vr_k7;
74 vr_k10=(1/psi_1)*(A-d*I)*vr_k9-(beta0/psi_1)*vr_k8;
75 vr_k11=(1/psi_1)*(A-d*I)*vr_k10-(beta0/psi_1)*vr_k9;
76 vr_k12=(1/psi_1)*(A-d*I)*vr_k11-(beta0/psi_1)*vr_k10;
77 vr_k13=(1/psi_1)*(A-d*I)*vr_k12-(beta0/psi_1)*vr_k11;
78 vr_k14=(1/psi_1)*(A-d*I)*vr_k13-(beta0/psi_1)*vr_k12;
79 vr_k15=(1/psi_1)*(A-d*I)*vr_k14-(beta0/psi_1)*vr_k13;
80 %
81 vtp_k0=pt1;
82 vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
83 vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
84 vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
85 vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
86 vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
87 vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
88 vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
89 vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
90 vtp_k9=(1/psi_1)*(A'-d*I)*vtp_k8-(beta0/psi_1)*vtp_k7;
91 vtp_k10=(1/psi_1)*(A'-d*I)*vtp_k9-(beta0/psi_1)*vtp_k8;
92 vtp_k11=(1/psi_1)*(A'-d*I)*vtp_k10-(beta0/psi_1)*vtp_k9;
93 vtp_k12=(1/psi_1)*(A'-d*I)*vtp_k11-(beta0/psi_1)*vtp_k10;
94 vtp_k13=(1/psi_1)*(A'-d*I)*vtp_k12-(beta0/psi_1)*vtp_k11;
95 vtp_k14=(1/psi_1)*(A'-d*I)*vtp_k13-(beta0/psi_1)*vtp_k12;
96 vtp_k15=(1/psi_1)*(A'-d*I)*vtp_k14-(beta0/psi_1)*vtp_k13;
97 vtp_k16=(1/psi_1)*(A'-d*I)*vtp_k15-(beta0/psi_1)*vtp_k14;
98 %
99 vtr_k0=rt1;
100 vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
101 vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
102 vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
103 vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
104 vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
105 vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
106 vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
107 vtr_k8=(1/psi_1)*(A'-d*I)*vtr_k7-(beta0/psi_1)*vtr_k6;
108 vtr_k9=(1/psi_1)*(A'-d*I)*vtr_k8-(beta0/psi_1)*vtr_k7;
109 vtr_k10=(1/psi_1)*(A'-d*I)*vtr_k9-(beta0/psi_1)*vtr_k8;
110 vtr_k11=(1/psi_1)*(A'-d*I)*vtr_k10-(beta0/psi_1)*vtr_k9;
111 vtr_k12=(1/psi_1)*(A'-d*I)*vtr_k11-(beta0/psi_1)*vtr_k10;
112 vtr_k13=(1/psi_1)*(A'-d*I)*vtr_k12-(beta0/psi_1)*vtr_k11;
113 vtr_k14=(1/psi_1)*(A'-d*I)*vtr_k13-(beta0/psi_1)*vtr_k12;
114 vtr_k15=(1/psi_1)*(A'-d*I)*vtr_k14-(beta0/psi_1)*vtr_k13;

```

```

115 %vectors for the monomial basis
116 vpp_k0=p11;
117 vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
118 vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
119 vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
120 vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
121 vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
122 vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
123 vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
124 vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
125 vpp_k9=(1/ppsi_0)*(A-dd*I)*vpp_k8-(bbeta0/ppsi_0)*vpp_k7;
126 vpp_k10=(1/ppsi_0)*(A-dd*I)*vpp_k9-(bbeta0/ppsi_0)*vpp_k8;
127 vpp_k11=(1/ppsi_0)*(A-dd*I)*vpp_k10-(bbeta0/ppsi_0)*vpp_k9;
128 vpp_k12=(1/ppsi_0)*(A-dd*I)*vpp_k11-(bbeta0/ppsi_0)*vpp_k10;
129 vpp_k13=(1/ppsi_0)*(A-dd*I)*vpp_k12-(bbeta0/ppsi_0)*vpp_k11;
130 vpp_k14=(1/ppsi_0)*(A-dd*I)*vpp_k13-(bbeta0/ppsi_0)*vpp_k12;
131 vpp_k15=(1/ppsi_0)*(A-dd*I)*vpp_k14-(bbeta0/ppsi_0)*vpp_k13;
132 vpp_k16=(1/ppsi_0)*(A-dd*I)*vpp_k15-(bbeta0/ppsi_0)*vpp_k14;
133 %
134 vrr_k0=r11;
135 vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
136 vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
137 vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
138 vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
139 vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
140 vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
141 vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
142 vrr_k8=(1/ppsi_0)*(A-dd*I)*vrr_k7-(bbeta0/ppsi_0)*vrr_k6;
143 vrr_k9=(1/ppsi_0)*(A-dd*I)*vrr_k8-(bbeta0/ppsi_0)*vrr_k7;
144 vrr_k10=(1/ppsi_0)*(A-dd*I)*vrr_k9-(bbeta0/ppsi_0)*vrr_k8;
145 vrr_k11=(1/ppsi_0)*(A-dd*I)*vrr_k10-(bbeta0/ppsi_0)*vrr_k9;
146 vrr_k12=(1/ppsi_0)*(A-dd*I)*vrr_k11-(bbeta0/ppsi_0)*vrr_k10;
147 vrr_k13=(1/ppsi_0)*(A-dd*I)*vrr_k12-(bbeta0/ppsi_0)*vrr_k11;
148 vrr_k14=(1/ppsi_0)*(A-dd*I)*vrr_k13-(bbeta0/ppsi_0)*vrr_k12;
149 vrr_k15=(1/ppsi_0)*(A-dd*I)*vrr_k14-(bbeta0/ppsi_0)*vrr_k13;
150 %
151 vttp_k0=pp1;
152 vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
153 vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
154 vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
155 vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
156 vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
157 vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
158 vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
159 vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;
160 vttp_k9=(1/ppsi_0)*(A'-dd*I)*vttp_k8-(bbeta0/ppsi_0)*vttp_k7;
161 vttp_k10=(1/ppsi_0)*(A'-dd*I)*vttp_k9-(bbeta0/ppsi_0)*vttp_k8;
162 vttp_k11=(1/ppsi_0)*(A'-dd*I)*vttp_k10-(bbeta0/ppsi_0)*vttp_k9;
163 vttp_k12=(1/ppsi_0)*(A'-dd*I)*vttp_k11-(bbeta0/ppsi_0)*vttp_k10;
164 vttp_k13=(1/ppsi_0)*(A'-dd*I)*vttp_k12-(bbeta0/ppsi_0)*vttp_k11;
165 vttp_k14=(1/ppsi_0)*(A'-dd*I)*vttp_k13-(bbeta0/ppsi_0)*vttp_k12;
166 vttp_k15=(1/ppsi_0)*(A'-dd*I)*vttp_k14-(bbeta0/ppsi_0)*vttp_k13;
167 vttp_k16=(1/ppsi_0)*(A'-dd*I)*vttp_k15-(bbeta0/ppsi_0)*vttp_k14;
168 %
169 vttr_k0=r1;
170 vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
171 vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;

```



```

172 vttr.k3=(1/ppsi_0)*(A'-dd*I)*vttr.k2-(bbeta0/ppsi_0)*vttr.k1;
173 vttr.k4=(1/ppsi_0)*(A'-dd*I)*vttr.k3-(bbeta0/ppsi_0)*vttr.k2;
174 vttr.k5=(1/ppsi_0)*(A'-dd*I)*vttr.k4-(bbeta0/ppsi_0)*vttr.k3;
175 vttr.k6=(1/ppsi_0)*(A'-dd*I)*vttr.k5-(bbeta0/ppsi_0)*vttr.k4;
176 vttr.k7=(1/ppsi_0)*(A'-dd*I)*vttr.k6-(bbeta0/ppsi_0)*vttr.k5;
177 vttr.k8=(1/ppsi_0)*(A'-dd*I)*vttr.k7-(bbeta0/ppsi_0)*vttr.k6;
178 vttr.k9=(1/ppsi_0)*(A'-dd*I)*vttr.k8-(bbeta0/ppsi_0)*vttr.k7;
179 vttr.k10=(1/ppsi_0)*(A'-dd*I)*vttr.k9-(bbeta0/ppsi_0)*vttr.k8;
180 vttr.k11=(1/ppsi_0)*(A'-dd*I)*vttr.k10-(bbeta0/ppsi_0)*vttr.k9;
181 vttr.k12=(1/ppsi_0)*(A'-dd*I)*vttr.k11-(bbeta0/ppsi_0)*vttr.k10;
182 vttr.k13=(1/ppsi_0)*(A'-dd*I)*vttr.k12-(bbeta0/ppsi_0)*vttr.k11;
183 vttr.k14=(1/ppsi_0)*(A'-dd*I)*vttr.k13-(bbeta0/ppsi_0)*vttr.k12;
184 vttr.k15=(1/ppsi_0)*(A'-dd*I)*vttr.k14-(bbeta0/ppsi_0)*vttr.k13;
185 %Chebyshev basis
186 V_p=[vp.k0, vp.k1, vp.k2, vp.k3, vp.k4, vp.k5, vp.k6, vp.k7, vp.k8, ...
187      vp.k9, vp.k10, vp.k11, vp.k12, vp.k13, vp.k14, vp.k15, vp.k16];
188 V_r=[vr.k0, vr.k1, vr.k2, vr.k3, vr.k4, vr.k5, vr.k6, vr.k7, vr.k8, ...
189      vr.k9, vr.k10, vr.k11, vr.k12, vr.k13, vr.k14, vr.k15];
190 Vt_p=[vtp.k0, vtp.k1, vtp.k2, vtp.k3, vtp.k4, vtp.k5, vtp.k6, ...
191      vtp.k7, vtp.k8, vtp.k9, vtp.k10, vtp.k11, vtp.k12, vtp.k13, ...
192      vtp.k14, vtp.k15, vtp.k16];
193 Vt_r=[vtr.k0, vtr.k1, vtr.k2, vtr.k3, vtr.k4, vtr.k5, vtr.k6, ...
194      vtr.k7, vtr.k8, vtr.k9, vtr.k10, vtr.k11, vtr.k12, vtr.k13, vtr.k14, vtr.k15];
195 V=[V_p, V_r];
196 Vt=[Vt_p, Vt_r];
197 G=Vt'*V;
198 %Monomial basis
199 VV_p=[vpp.k0, vpp.k1, vpp.k2, vpp.k3, vpp.k4, vpp.k5, vpp.k6, ...
200      vpp.k7, vpp.k8, vpp.k9, vpp.k10, vpp.k11, vpp.k12, vpp.k13, ...
201      vpp.k14, vpp.k15, vpp.k16];
202 VV_r=[vrr.k0, vrr.k1, vrr.k2, vrr.k3, vrr.k4, vrr.k5, vrr.k6, ...
203      vrr.k7, vrr.k8, vrr.k9, vrr.k10, vrr.k11, vrr.k12, vrr.k13, ...
204      vrr.k14, vrr.k15];
205 VVt_p=[vttp.k0, vttp.k1, vttp.k2, vttp.k3, vttp.k4, vttp.k5, vttp.k6, ...
206      vttp.k7, vttp.k8, vttp.k9, vttp.k10, vttp.k11, vttp.k12, vttp.k13, ...
207      vttp.k14, vttp.k15, vttp.k16];
208 VVt_r=[vttr.k0, vttr.k1, vttr.k2, vttr.k3, vttr.k4, vttr.k5, vttr.k6, ...
209      vttr.k7, vttr.k8, vttr.k9, vttr.k10, vttr.k11, vttr.k12, vttr.k13, ...
210      vttr.k14, vttr.k15];
211 VV=[VV_p, VV_r];
212 VVt=[VVt_p, VVt_r];
213 GG=VVt'*VV;
214 %matrix B_{k} for when we use the Chebyshev basis
215 B=eye(33);
216 for i=1:33
217     B(i,i)=aj;
218 end
219 for i=1:32
220     B(i,i+1)=beta0;
221     B(i+1,i)=psi-1;
222 end
223 B(2,1)=psi_0;
224 B(19,18)=psi_0;
225 B(33,33)=0;
226 B(17,17)=0;
227 B(32,33)=0;
228 B(16,17)=0;

```

```

229 B(18,17)=0;
230 B(17,18)=0;
231 %matrix B_{k} for when we use the monomial basis
232 BB=eye(33);
233 for i=1:33
234     BB(i,i)=ajj;
235 end
236 for i=1:32
237     BB(i,i+1)=bbeta0;
238     BB(i+1,i)=ppsi_0;
239 end
240
241 BB(33,33)=0;
242 BB(17,17)=0;
243 BB(32,33)=0;
244 BB(16,17)=0;
245 BB(18,17)=0;
246 %starting the loops
247 s=16
248 kk=1
249 n=10049 %maximum number for s=16
250 normb=norm(b1)
251 while (norm(r11)/normb>tol)
252     pp_k0=[1 zeros(1,32)]';
253     rr_k0=[zeros(1,17) 1 zeros(1,15)]';
254     ee_k0=zeros(1,33);
255     ppt_k0=[1 zeros(1,32)]';
256     rrt_k0=[zeros(1,17) 1 zeros(1,15)]';
257     del1=rrt_k0'*GG*rr_k0;
258     for j=0 : s-1
259         alpha1=del1/(ppt_k0'*GG*BB*pp_k0);
260         ee_k1=ee_k0+alpha1*pp_k0';
261         rr_k1=rr_k0-BB*(alpha1*pp_k0);
262         rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
263         del3=rrt_k1'*GG*rr_k1;
264         betaa=del3/del1;
265         pp_k1=rr_k1+betaa*pp_k0;
266         ppt_k1=rrt_k1+betaa*ppt_k0;
267         rr_k0=rr_k1;
268         rrt_k0=rrt_k1;
269         pp_k0=pp_k1;
270         ppt_k0=ppt_k1;
271         del1=del3;
272         ee_k0=ee_k1;
273     end
274     xmm=VV*ee_k0'+x11;
275     rmm=VV*rr_k0;
276     pmm=VV*pp_k0;
277     rrtm=VVt*rrt_k0;
278     pptm=VVt*ppt_k0;
279     valuer2(:,kk)=r11;
280     u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
281     x11=xmm;
282     r11=rmm;
283     p11=pmm;
284     ppt1=pptm;
285     rrt1=rrtm;

```

```

286 tr11=b1-A*x11;
287 %vectors for the monomial basis
288 vpp_k0=p11;
289 vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
290 vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
291 vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
292 vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
293 vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
294 vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
295 vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
296 vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
297 vpp_k9=(1/ppsi_0)*(A-dd*I)*vpp_k8-(bbeta0/ppsi_0)*vpp_k7;
298 vpp_k10=(1/ppsi_0)*(A-dd*I)*vpp_k9-(bbeta0/ppsi_0)*vpp_k8;
299 vpp_k11=(1/ppsi_0)*(A-dd*I)*vpp_k10-(bbeta0/ppsi_0)*vpp_k9;
300 vpp_k12=(1/ppsi_0)*(A-dd*I)*vpp_k11-(bbeta0/ppsi_0)*vpp_k10;
301 vpp_k13=(1/ppsi_0)*(A-dd*I)*vpp_k12-(bbeta0/ppsi_0)*vpp_k11;
302 vpp_k14=(1/ppsi_0)*(A-dd*I)*vpp_k13-(bbeta0/ppsi_0)*vpp_k12;
303 vpp_k15=(1/ppsi_0)*(A-dd*I)*vpp_k14-(bbeta0/ppsi_0)*vpp_k13;
304 vpp_k16=(1/ppsi_0)*(A-dd*I)*vpp_k15-(bbeta0/ppsi_0)*vpp_k14;
305
306 vrr_k0=r11;
307 vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
308 vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
309 vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
310 vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
311 vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
312 vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
313 vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
314 vrr_k8=(1/ppsi_0)*(A-dd*I)*vrr_k7-(bbeta0/ppsi_0)*vrr_k6;
315 vrr_k9=(1/ppsi_0)*(A-dd*I)*vrr_k8-(bbeta0/ppsi_0)*vrr_k7;
316 vrr_k10=(1/ppsi_0)*(A-dd*I)*vrr_k9-(bbeta0/ppsi_0)*vrr_k8;
317 vrr_k11=(1/ppsi_0)*(A-dd*I)*vrr_k10-(bbeta0/ppsi_0)*vrr_k9;
318 vrr_k12=(1/ppsi_0)*(A-dd*I)*vrr_k11-(bbeta0/ppsi_0)*vrr_k10;
319 vrr_k13=(1/ppsi_0)*(A-dd*I)*vrr_k12-(bbeta0/ppsi_0)*vrr_k11;
320 vrr_k14=(1/ppsi_0)*(A-dd*I)*vrr_k13-(bbeta0/ppsi_0)*vrr_k12;
321 vrr_k15=(1/ppsi_0)*(A-dd*I)*vrr_k14-(bbeta0/ppsi_0)*vrr_k13;
322
323 vttp_k0=ppt1;
324 vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
325 vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
326 vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
327 vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
328 vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
329 vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
330 vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
331 vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;
332 vttp_k9=(1/ppsi_0)*(A'-dd*I)*vttp_k8-(bbeta0/ppsi_0)*vttp_k7;
333 vttp_k10=(1/ppsi_0)*(A'-dd*I)*vttp_k9-(bbeta0/ppsi_0)*vttp_k8;
334 vttp_k11=(1/ppsi_0)*(A'-dd*I)*vttp_k10-(bbeta0/ppsi_0)*vttp_k9;
335 vttp_k12=(1/ppsi_0)*(A'-dd*I)*vttp_k11-(bbeta0/ppsi_0)*vttp_k10;
336 vttp_k13=(1/ppsi_0)*(A'-dd*I)*vttp_k12-(bbeta0/ppsi_0)*vttp_k11;
337 vttp_k14=(1/ppsi_0)*(A'-dd*I)*vttp_k13-(bbeta0/ppsi_0)*vttp_k12;
338 vttp_k15=(1/ppsi_0)*(A'-dd*I)*vttp_k14-(bbeta0/ppsi_0)*vttp_k13;
339 vttp_k16=(1/ppsi_0)*(A'-dd*I)*vttp_k15-(bbeta0/ppsi_0)*vttp_k14;
340
341 vttr_k0=rrt1;
342 vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;

```

```

343 vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
344 vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
345 vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
346 vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
347 vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
348 vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
349 vttr_k8=(1/ppsi_0)*(A'-dd*I)*vttr_k7-(bbeta0/ppsi_0)*vttr_k6;
350 vttr_k9=(1/ppsi_0)*(A'-dd*I)*vttr_k8-(bbeta0/ppsi_0)*vttr_k7;
351 vttr_k10=(1/ppsi_0)*(A'-dd*I)*vttr_k9-(bbeta0/ppsi_0)*vttr_k8;
352 vttr_k11=(1/ppsi_0)*(A'-dd*I)*vttr_k10-(bbeta0/ppsi_0)*vttr_k9;
353 vttr_k12=(1/ppsi_0)*(A'-dd*I)*vttr_k11-(bbeta0/ppsi_0)*vttr_k10;
354 vttr_k13=(1/ppsi_0)*(A'-dd*I)*vttr_k12-(bbeta0/ppsi_0)*vttr_k11;
355 vttr_k14=(1/ppsi_0)*(A'-dd*I)*vttr_k13-(bbeta0/ppsi_0)*vttr_k12;
356 vttr_k15=(1/ppsi_0)*(A'-dd*I)*vttr_k14-(bbeta0/ppsi_0)*vttr_k13;
357
358 VV_p=[vpp_k0, vpp_k1, vpp_k2, vpp_k3, vpp_k4, vpp_k5, vpp_k6, ...
359 vpp_k7, vpp_k8, vpp_k9, vpp_k10, vpp_k11, vpp_k12, vpp_k13, ...
360 vpp_k14, vpp_k15, vpp_k16];
361 VV_r=[vrr_k0, vrr_k1, vrr_k2, vrr_k3, vrr_k4, vrr_k5, vrr_k6, ...
362 vrr_k7, vrr_k8, vrr_k9, vrr_k10, vrr_k11, vrr_k12, vrr_k13, ...
363 vrr_k14, vrr_k15];
364 VVt_p=[vttp_k0, vttp_k1, vttp_k2, vttp_k3, ...
        vttp_k4, vttp_k5, vttp_k6, ...
365 vttp_k7, vttp_k8, vttp_k9, vttp_k10, vttp_k11, vttp_k12, vttp_k13, ...
366 vttp_k14, vttp_k15, vttp_k16];
367 VVt_r=[vttr_k0, vttr_k1, vttr_k2, vttr_k3, vttr_k4, vttr_k5, vttr_k6, ...
368 vttr_k7, vttr_k8, vttr_k9, vttr_k10, vttr_k11, vttr_k12, vttr_k13, ...
369 vttr_k14, vttr_k15];
370 VV=[VV_p, VV_r];
371 VVt=[VVt_p, VVt_r];
372 GG=VVt'*VV;
373 kk=kk+1;
374 valuer2(:,kk)=r11;
375 u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
376 if kk==n
377     break;
378 end
379
380 end
381 s=16
382 k=1
383 n=10049
384 normb=norm(b1);
385 while (norm(r1)/normb>tol)
386     p_k0=[1 zeros(1,32)]';
387     r_k0=[zeros(1,17) 1 zeros(1,15)]';
388     e_k0=zeros(1,33);
389     pt_k0=[1 zeros(1,32)]';
390     rt_k0=[zeros(1,17) 1 zeros(1,15)]';
391     delt=rt_k0'*G*r_k0;
392     for j=0 : s-1
393         alpha=delt/(pt_k0'*G*B*p_k0);
394         e_k1=e_k0+alpha*p_k0';
395         r_k1=r_k0-B*(alpha*p_k0);
396         rt_k1=rt_k0-B*(alpha*pt_k0);
397         delt2=rt_k1'*G*r_k1;
398         beta=delt2/delt;

```

```

399     p_k1=r_k1+beta*p_k0;
400     pt_k1=rt_k1+beta*pt_k0;
401     %updating
402     r_k0=r_k1;
403     rt_k0=rt_k1;
404     p_k0=p_k1;
405     pt_k0=pt_k1;
406     delt=delt2;
407     e_k0=e_k1;
408     end
409     xm=V*e_k0'+x1;
410     rm=V*r_k0;
411     pm=V*p_k0;
412     rtm=Vt*rt_k0;
413     ptm=Vt*pt_k0;
414     valuer1(:,k)=r1;
415     u1(:,k)=norm(valuer1(:,k)/norm(b1));
416     x1=xm;
417     r1=rm;
418     p1=pm;
419     pt1=ptm;
420     rt1=rtm;
421     t_r1=b1-A*x1;
422     vp_k0=p1;
423     vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
424     vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
425     vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
426     vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
427     vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
428     vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
429     vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
430     vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
431     vp_k9=(1/psi_1)*(A-d*I)*vp_k8-(beta0/psi_1)*vp_k7;
432     vp_k10=(1/psi_1)*(A-d*I)*vp_k9-(beta0/psi_1)*vp_k8;
433     vp_k11=(1/psi_1)*(A-d*I)*vp_k10-(beta0/psi_1)*vp_k9;
434     vp_k12=(1/psi_1)*(A-d*I)*vp_k11-(beta0/psi_1)*vp_k10;
435     vp_k13=(1/psi_1)*(A-d*I)*vp_k12-(beta0/psi_1)*vp_k11;
436     vp_k14=(1/psi_1)*(A-d*I)*vp_k13-(beta0/psi_1)*vp_k12;
437     vp_k15=(1/psi_1)*(A-d*I)*vp_k14-(beta0/psi_1)*vp_k13;
438     vp_k16=(1/psi_1)*(A-d*I)*vp_k15-(beta0/psi_1)*vp_k14;
439
440     vr_k0=r1;
441     vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
442     vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
443     vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
444     vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
445     vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
446     vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
447     vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
448     vr_k8=(1/psi_1)*(A-d*I)*vr_k7-(beta0/psi_1)*vr_k6;
449     vr_k9=(1/psi_1)*(A-d*I)*vr_k8-(beta0/psi_1)*vr_k7;
450     vr_k10=(1/psi_1)*(A-d*I)*vr_k9-(beta0/psi_1)*vr_k8;
451     vr_k11=(1/psi_1)*(A-d*I)*vr_k10-(beta0/psi_1)*vr_k9;
452     vr_k12=(1/psi_1)*(A-d*I)*vr_k11-(beta0/psi_1)*vr_k10;
453     vr_k13=(1/psi_1)*(A-d*I)*vr_k12-(beta0/psi_1)*vr_k11;
454     vr_k14=(1/psi_1)*(A-d*I)*vr_k13-(beta0/psi_1)*vr_k12;
455     vr_k15=(1/psi_1)*(A-d*I)*vr_k14-(beta0/psi_1)*vr_k13;

```

```

456
457 vtp_k0=pt1;
458 vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
459 vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
460 vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
461 vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
462 vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
463 vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
464 vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
465 vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
466 vtp_k9=(1/psi_1)*(A'-d*I)*vtp_k8-(beta0/psi_1)*vtp_k7;
467 vtp_k10=(1/psi_1)*(A'-d*I)*vtp_k9-(beta0/psi_1)*vtp_k8;
468 vtp_k11=(1/psi_1)*(A'-d*I)*vtp_k10-(beta0/psi_1)*vtp_k9;
469 vtp_k12=(1/psi_1)*(A'-d*I)*vtp_k11-(beta0/psi_1)*vtp_k10;
470 vtp_k13=(1/psi_1)*(A'-d*I)*vtp_k12-(beta0/psi_1)*vtp_k11;
471 vtp_k14=(1/psi_1)*(A'-d*I)*vtp_k13-(beta0/psi_1)*vtp_k12;
472 vtp_k15=(1/psi_1)*(A'-d*I)*vtp_k14-(beta0/psi_1)*vtp_k13;
473 vtp_k16=(1/psi_1)*(A'-d*I)*vtp_k15-(beta0/psi_1)*vtp_k14;
474
475 vtr_k0=rt1;
476 vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
477 vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
478 vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
479 vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
480 vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
481 vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
482 vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
483 vtr_k8=(1/psi_1)*(A'-d*I)*vtr_k7-(beta0/psi_1)*vtr_k6;
484 vtr_k9=(1/psi_1)*(A'-d*I)*vtr_k8-(beta0/psi_1)*vtr_k7;
485 vtr_k10=(1/psi_1)*(A'-d*I)*vtr_k9-(beta0/psi_1)*vtr_k8;
486 vtr_k11=(1/psi_1)*(A'-d*I)*vtr_k10-(beta0/psi_1)*vtr_k9;
487 vtr_k12=(1/psi_1)*(A'-d*I)*vtr_k11-(beta0/psi_1)*vtr_k10;
488 vtr_k13=(1/psi_1)*(A'-d*I)*vtr_k12-(beta0/psi_1)*vtr_k11;
489 vtr_k14=(1/psi_1)*(A'-d*I)*vtr_k13-(beta0/psi_1)*vtr_k12;
490 vtr_k15=(1/psi_1)*(A'-d*I)*vtr_k14-(beta0/psi_1)*vtr_k13;
491 V_p=[vp_k0, vp_k1, vp_k2, vp_k3, vp_k4, vp_k5, vp_k6, vp_k7, vp_k8, ...
492      vp_k9, vp_k10, vp_k11, vp_k12, vp_k13, vp_k14, vp_k15, vp_k16];
493 V_r=[vr_k0, vr_k1, vr_k2, vr_k3, vr_k4, vr_k5, vr_k6, vr_k7, vr_k8, ...
494      vr_k9, vr_k10, vr_k11, vr_k12, vr_k13, vr_k14, vr_k15];
495 Vt_p=[vtp_k0, vtp_k1, vtp_k2, vtp_k3, vtp_k4, vtp_k5, vtp_k6, ...
496      vtp_k7, vtp_k8, vtp_k9, vtp_k10, vtp_k11, vtp_k12, vtp_k13, ...
497      vtp_k14, vtp_k15, vtp_k16];
498 Vt_r=[vtr_k0, vtr_k1, vtr_k2, vtr_k3, vtr_k4, vtr_k5, vtr_k6, ...
499      vtr_k7, vtr_k8, vtr_k9, vtr_k10, vtr_k11, vtr_k12, vtr_k13, vtr_k14, vtr_k15];
500 V=[V_p, V_r];
501 Vt=[Vt_p, Vt_r];
502 G=Vt'*V;
503 k=k+1;
504 valuer1(:,k)=r1;
505 u1(:,k)=norm(valuer1(:,k)/norm(b1));
506 if k==n
507     break
508 end
509
510 end
511 %plot for both bases
512 semilogy(u1, '-o')

```

```
513 xlabel('Number of Iterations')
514 ylabel('2-Norm Residual')
515 hold on
516 semilogy(u2, '-*')
517 legend ('Chebyshev Basis s=16 ', 'Monomial Basis s=16')
518 hold off
```

References

- [1] Z. Bai et al. *A Test Matrix Collection for Non-Hermitian Eigenvalue Problems*. Tech. rep. CS-97-355. Department of Computer Science University of Tennessee, 1997.
- [2] E. Carson. *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*. Tech. rep. UCB/EECS-2015-179. Electrical Engineering and Computer Sciences University of California at Berkeley, 2015.
- [3] E. Carson and J. Demmel. *A Residual Replacement Strategy for Improving the Maximum Attainable Accuracy of s-step Krylov Subspace Methods*. Tech. rep. UCB/EECS-2012-197. Electrical Engineering and Computer Sciences University of California at Berkeley, 2012.
- [4] E. Carson and J. Demmel. *Analysis of the finite precision s-step biconjugate gradient method*. Tech. rep. UCB/EECS-2014-18. Electrical Engineering and Computer Sciences University of California at Berkeley, 2014.
- [5] E. Carson, N. Knight, and J. Demmel. “Avoiding communication in non-symmetric Lanczos - based Krylov subspace methods”. In: *SIAM Journal on Scientific Computing* 35.5 (2013), S42–S61.
- [6] B. A. Cipra. “The Best of the 20th Century: Editors Name Top 10 Algorithms”. In: *SIAM News* 33.4 (2000).
- [7] D. A. Gismalla. “Matlab Software for Iterative Methods and Algorithms to Solve a Linear System”. In: *International Journal of Engineering and Technical Research (IJETR)* 2 (2014).
- [8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press., 1996. ISBN: 9780801854149, 0801854148.
- [9] M. F. Hoemmen. *Communication-avoiding Krylov subspace methods*. Tech. rep. UCB/EECS-2010-37. Electrical Engineering and Computer Sciences University of California at Berkeley, 2010.
- [10] W. D. Joubert and G. F. Carey. “Parallelizable Restarted Iterative Methods for Nonsymmetric Linear Systems. Part I: Theory”. In: *International Journal of Computer Mathematics* 44 (1992), pp. 243–267.
- [11] Q. Kong, T. Siau, and A. M. Bayen. *Python Programming and Numerical Methods: A Guide for Engineers and Scientists*. Academic Press., 2020. ISBN: 9780128195499, 0128195495.

- [12] T. A. Manteuffel. “The Tchebychev Iteration for Nonsymmetric Linear Systems”. In: *Numerische Mathematik* 28 (1977), pp. 307–327.
- [13] C. H. Tong and Q. Ye. “Analysis of the finite precision bi-conjugate gradient algorithm for nonsymmetric linear systems”. In: *Mathematics of Computation* 69.232 (2000), pp. 1559–1575.
- [14] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. Siam., 1997. ISBN: 9780898713619, 0898713617.
- [15] Matrix Market website. URL: <https://math.nist.gov/MatrixMarket/mmio/matlab/mmio matlab.html>.
- [16] studenti.it website. <https://www.studenti.it/matematica/Proprietat-ellisse.jspc>.
- [17] Wikipedia website. URL: https://en.wikipedia.org/wiki/Sparse_matrix.

Bachelor's Theses in Mathematical Sciences 2022:K14
ISSN 1654-6229
LUNFNA-4041-2022
Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden
<http://www.maths.lu.se/>