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

CENTRUM SCIENTIARUM MATHEMATICARUM

LUND UNIVERSITY

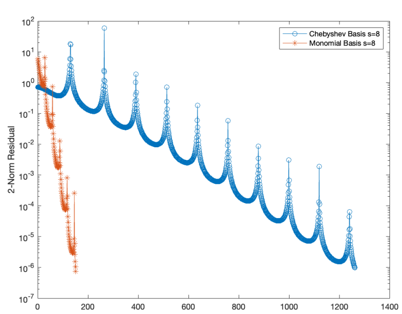
NUMK11

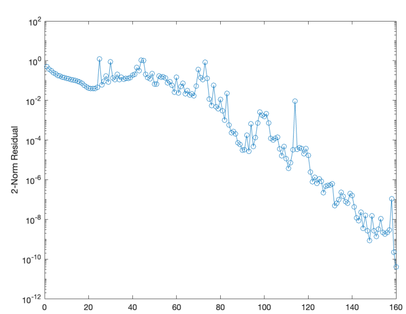
Bachelor's project in Numerical Analysis

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

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:

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

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 bicojugate 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. $x_0 = [0, ..., 0]$, $m = 1$

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 $n \neq m$:

$$\tilde{r}_m^T r_n = 0. \tag{2.1}$$

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:

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

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

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

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]:

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

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:

$$p_1 \in K_2(A, p_0) + K_1(A, r_0). \tag{2.3}$$

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:

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

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:

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

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).

□

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: $K_1(A, v) \subseteq K_2(A, v) \subseteq ... \subseteq K_{s+1}(A, v)$, 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:

$$p_1 \in K_2(A, p_0) + K_1(A, r_0). \tag{2.6}$$

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

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

so we can write:

$$\begin{aligned} p_{s-1} &= r_{s-1} + \beta_{s-1} p_{s-2} \\ &= r_{s-2} - \alpha_{s-2} A p_{s-2} + \beta_{s-1} p_{s-2} \\ &= ... = d_0 r_0 + d_1 A r_0 + d_2 A A r_0 + ... d_{s-2} A^{s-2} r_0 + c_0 p_0 + c_1 A p_0 + ... + c_{s-1} A^{s-1} p_0 \\ &\in K_s(A, p_0) + K_{s-1}(A, r_0). \end{aligned}$$

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

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

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

satisfies

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

Using the hypothesis step, it follows that:

$$\begin{aligned} p_s &= r_s + \beta_s p_{s-1} \\ &= r_{s-1} - \alpha_{s-1} A p_{s-1} + \beta_s p_{s-1} \\ &= d_0 r_0 + d_1 A r_0 + d_2 A A r_0 + ... + d_{s-1} A^{s-1} r_0 + c_0 p_0 + c_1 A p_0 + ... + c_s A^s p_0 \quad \in K_{s+1}(A, p_0) + K_s(A, r_0). \end{aligned}$$

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

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

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

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 $i \in \{0, ..., s-1\}$:

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

here I is the identity matrix and γ_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]:

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

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 $C_{k,s+1} \in \mathbb{C}^{s+1 \times s}$, $C_{k,s} \in \mathbb{C}^{s \times s-1}$:

$$C_{k,s+1} = \begin{pmatrix} \mathrm{a}_0 & \beta_0 & 0 & 0 & 0 & ... & 0 \\ \gamma_0 & \mathrm{a}_1 & \beta_1 & 0 & 0 & ... & 0 \\ 0 & \gamma_1 & \mathrm{a}_2 & \beta_2 & 0 & ... & 0 \\ 0 & 0 & \gamma_2 & \mathrm{a}_3 & \beta_3 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & \beta_{s-1} \\ ... & ... & ... & ... & ... & ... & \mathrm{a}_s \\ 0 & 0 & 0 & 0 & 0 & ... & \gamma_s \end{pmatrix},$$

$$
C_{k,s}=\begin{pmatrix}
\mathrm{a}_0 & \beta_0 & 0 & 0 & 0 & ... & 0 \\
\gamma_0 & \mathrm{a}_1 & \beta_1 & 0 & 0 & ... & 0 \\
0 & \gamma_1 & \mathrm{a}_2 & \beta_2 & 0 & ... & 0 \\
0 & 0 & \gamma_2 & \mathrm{a}_3 & \beta_3 & ... & 0 \\
... & ... & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & \beta_{s-2} \\
... & ... & ... & ... & ... & ... & \mathrm{a}_{s-1} \\
0 & 0 & 0 & 0 & 0 & ... & \gamma_{s-1}
\end{pmatrix}.
$$

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

$$
\mathrm{B}_k=\begin{pmatrix}
[\mathrm{C}_{k,s+1}\ 0_{s+1,1}] & \\
 & [\mathrm{C}_{k,s}\ 0_{s,1}]
\end{pmatrix}.
$$

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:

$$
B_k=\begin{pmatrix}
\mathrm{a}_0 & \beta_0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
\gamma_0 & \mathrm{a}_1 & \beta_1 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
0 & \gamma_1 & \mathrm{a}_2 & \beta_2 & 0 & ... & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
0 & 0 & \gamma_2 & \mathrm{a}_3 & \beta_3 & ... & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & \beta_{s-1} & 0 & 0 & ... & ... & ... & ... & ... \\
... & ... & ... & ... & ... & ... & \mathrm{a}_s & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & ... & \gamma_s & 0 & 0 & 0 & 0 & ... & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & \mathrm{a}_0 & \beta_0 & 0 & ... & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & \gamma_0 & \mathrm{a}_1 & \beta_1 & ... & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & \gamma_1 & \mathrm{a}_2 & ... & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & \gamma_2 & ... & 0 & 0 \\
... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
0 & 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & 0 & ... & \gamma_{s-1} & 0
\end{pmatrix}.
$$

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]:

$$
\begin{aligned}
AV^p_{k,j} &= V^p_{k,j+1}C_{k,j+1}, & AV^r_{k,j-1} &= V^r_{k,j}C_{k,j}, \\
A^TV^{\tilde{p}}_{k,j} &= V^{\tilde{p}}_{k,j+1}C_{k,j+1}, & A^TV^{\tilde{r}}_{k,j-1} &= V^{\tilde{r}}_{k,j}C_{k,j}.
\end{aligned}
$$

Here $V^p_{k,j}$, $V^r_{k,j}$,$V^{\tilde{p}}_{k,j}$,$V^{\tilde{r}}_{k,j}$ are defined as:

$$\begin{aligned} V^p_{k,j} &= [v^p_{k,0}, v^p_{k,1}, ..., v^p_{k,j}], \\ V^r_{k,j} &= [v^r_{k,0}, v^r_{k,1}, ..., v^r_{k,j}], \\ V^{\tilde{p}}_{k,j} &= [v^{\tilde{p}}_{k,0}, v^{\tilde{p}}_{k,1}, ..., v^{\tilde{p}}_{k,j}], \\ V^{\tilde{r}}_{k,j} &= [v^{\tilde{r}}_{k,0}, v^{\tilde{r}}_{k,1}, ..., v^{\tilde{r}}_{k,j}]. \end{aligned}$$

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

$$V^p_{k,3} = [v^p_{k,0}, v^p_{k,1}, v^p_{k,2}],$$

here $v^p_{k,0}$, $v^p_{k,1}$, $v^p_{k,2}$ are:

$$\begin{aligned} v^p_{k,0} &= p_{sk}, \\ v^p_{k,1} &= \frac{1}{\gamma_0}(A - \mathrm{a}_0 I)v^p_{k,0}, \\ v^p_{k,2} &= \frac{1}{\gamma_1}(A - \mathrm{a}_1 I)v^p_{k,1} - \tfrac{\beta_0}{\gamma_1}v^p_{k,0}. \end{aligned}$$

We have that:

$$AV^p_{k,2} = A[v^p_{k,0}, v^p_{k,1}]$$

and:

$$V^p_{k,3}C_{k,3} = [v^p_{k,0}, v^p_{k,1}, v^p_{k,2}] \begin{bmatrix} \mathrm{a}_0 & \beta_0 \\ \gamma_0 & a_1 \\ 0 & \gamma_1 \end{bmatrix} = [v^p_{k,0}\mathrm{a}_0 + v^p_{k,1}\gamma_0, v^p_{k,0}\beta_0 + v^p_{k,1}\mathrm{a}_1 + v^p_{k,2}\gamma_1].$$

It follows from (2.9) that:

$$v^p_{k,0}a_0 + v^p_{k,1}\gamma_0 = p_{sk}a_0 + (\frac{1}{\gamma_0}(A - \mathrm{a}_0\mathrm{I})\mathrm{p}_{sk})\gamma_0 = Ap_{sk},$$

$$v^p_{k,0}\beta_0 + v^p_{k,1}\mathrm{a}_1 + v^p_{k,2}\gamma_1 = p_{sk}\beta_0 + (\frac{1}{\gamma_0}(A - \mathrm{a}_0 I)p_{sk})\mathrm{a}_1 + (\tfrac{1}{\gamma_1}(A - \mathrm{a}_1 I)v^p_{k,1} - \tfrac{\beta_0}{\gamma_1}p_{sk})\gamma_1,$$

after computation, we arrive at:

$$= A(\frac{1}{\gamma_0}(A - \mathrm{a}_0 I))p_{sk}.$$

It follows that:

$$AV^p_{k,2} = V^p_{k,3}C_{k,3}.$$

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 $\bar{V}_k = [\bar{V}_k^p, \bar{V}_k^r]$, $\bar{\tilde{V}}_k = [\bar{V}_k^{\tilde{p}}, \bar{V}_k^{\tilde{r}}]$, 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:

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

In a similar way we have: $A^T\bar{\tilde{V}}_k = \tilde{V}_k B_k$. 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:

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

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:

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

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

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

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

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

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:

$$< \tilde{r}_{sk+j}, r_{sk+j} >, < \tilde{p}_{sk+j-1}, Ap_{sk+j-1} >,$$

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]:

$$\begin{aligned} < \tilde{r}_m, r_m > &= \tilde{r}_m^T r_m \\ &= (\tilde{V}_k \tilde{r}'_{k,j})^T (V_k r'_{k,j}) \\ &= \tilde{r}'^T_{k,j} G_k r'_{k,j}, \end{aligned} \tag{2.18}$$

$$\begin{aligned} < \tilde{p}_{m-1}, Ap_{m-1} > &= \tilde{p}_{m-1}^T A p_{m-1} \\ &= (\tilde{V}_k \tilde{p}'_{k,j-1})^T (A V_k p'_{k,j-1}) \\ &= \tilde{p}'^T_{k,j-1} \tilde{V}_k^T A V_k p'_{k,j-1}, \\ &\text{it follows from (2.12) that} \\ &= \tilde{p}'^T_{k,j-1} \tilde{V}_k^T V_k B_k p'_{k,j-1}, \\ &\text{because } G_k = \tilde{V}_k^T V_k, \text{ we have} \\ &= \tilde{p}'^T_{k,j-1} G_k B_k p'_{k,j-1}. \end{aligned} \tag{2.19}$$

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. $x_0 = [0, ..., 0]$,

2. $r_0 = p_0 = \tilde{r_0} = \tilde{p_0} = b - Ax_0$, $k = 0$
3. *while* k *until convergence*
4. $V_k^p = [v_{k,0}^p, ..., v_{k,s}^p]$
5. $V_k^r = [v_{k,0}^r, ..., v_{k,s-1}^r]$
6. $V_k^{\tilde{p}} = [v_{k,0}^{\tilde{p}}, ..., v_{k,s}^{\tilde{p}}]$
7. $V_k^{\tilde{r}} = [v_{k,0}^{\tilde{r}}, ..., v_{k,s-1}^{\tilde{r}}]$
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* $j = 1, ..., s$
17. $\delta_{m-1} = \tilde{r}'^T_{k,j-1} G_k r'_{k,j-1}$
18. $\alpha_{m-1} = \delta_{m-1} / \tilde{p}'^T_{k,j-1} G_k B_k p'_{k,j-1}$
19. $e_{k,j} = e_{k,j-1} + \alpha_{m-1} p'^T_{k,j-1}$
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. $\delta_m = \tilde{r}'^T_{k,j} G_k r'_{k,j}$
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. $x_{sk+s} = V_k e^T_{k,s} + x_{sk}$
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:

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

The three-term vectors (2.9), become:

$$v^p_{k,i+1} = Av^p_{k,i}, \quad v^r_{k,i+1} = Av^r_{k,i},$$
$$v^{\tilde{p}}_{k,i+1} = Av^{\tilde{p}}_{k,i}, \quad v^{\tilde{r}}_{k,i+1} = Av^{\tilde{r}}_{k,i},$$

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

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

$$v^p_{k,1} = Av^p_{k,0} = Ap_{sk},$$

for i=1:

$$v^p_{k,2} = Av^p_{k,1} = A(Ap_{sk}),$$

and so on, until we reach the following matrix:

$$V^p_k = [v^p_{k,0}, ..., v^p_{k,s}] = [p_{sk}, Ap_{sk}, ..., A^s p_{sk}].$$

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

$$span(V^p_k) = K_{j+1}(A, p_{sk}).$$

In a similar way we obtain: V^r_k, $V^{\tilde{p}}_k$, $V^{\tilde{r}}_k$.

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: $\tau_n(z) = cosh(ncosh^{-1}(z))$ [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]:

$$cosh(z) = cosh(x + iy) = cosh(x)cos(y) + isinh(x)sin(y) = u + iv,$$

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]:

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

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]:

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

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]:

$$\begin{aligned}
&\mathrm{a}_j = d, &&\beta_j = -c\frac{\sigma_j}{2\sigma_{j+1}}, \\
&\gamma_0 = -c\frac{\sigma_1}{\sigma_0}, &&\gamma_j = -c\frac{\sigma_{j+1}}{2\sigma_j},\ j > 0.
\end{aligned} \tag{2.23}$$

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

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

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:

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

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:

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

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

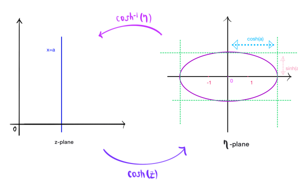


Figure 2.1: - From a line to an ellipse.

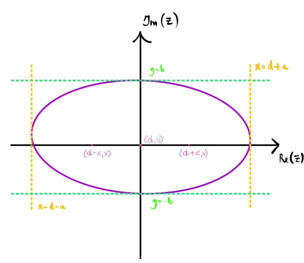


Figure 2.2: - Ellipse.

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.

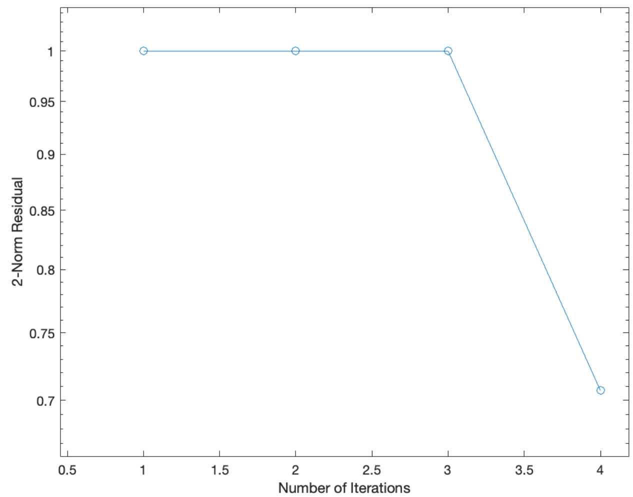


Figure 3.1: plot of the 2-norm residual using BiCG method, first example.

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 $\tilde{r}_1$.

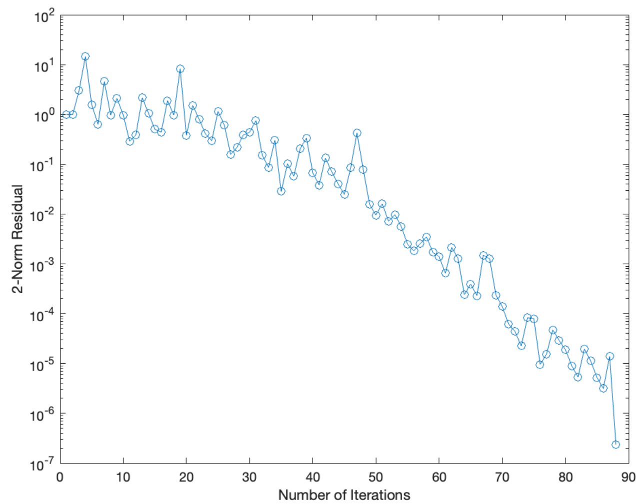


Figure 3.2: plot of the 2-norm residual using BiCG method, first example.

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: a$_j$=d, $\beta_j = \frac{c^2}{4g}$, $\gamma_0 = 2g$ and $\gamma_{j+1} = g$, for $j = 0, ..., s$.

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: $\frac{\|r_{sk+s}\|_2}{\|b_1\|_2} = 7.961976029306281e{-}07$, 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 $s = \{4, 8, 16\}$.

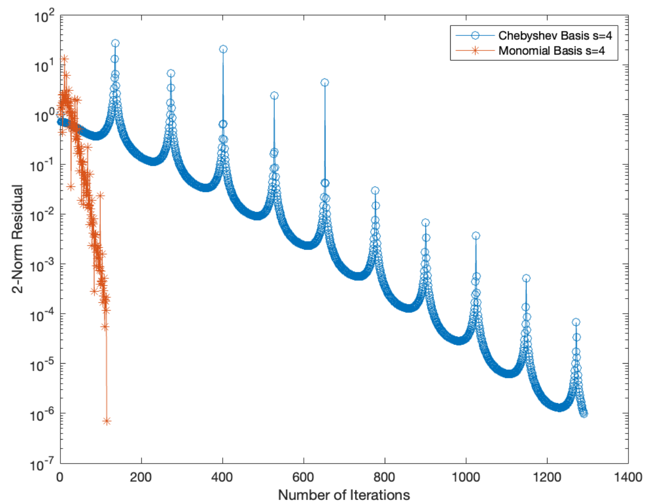


Figure 3.3: plot of the 2-norm residual using the s-step BiCG method, s=4

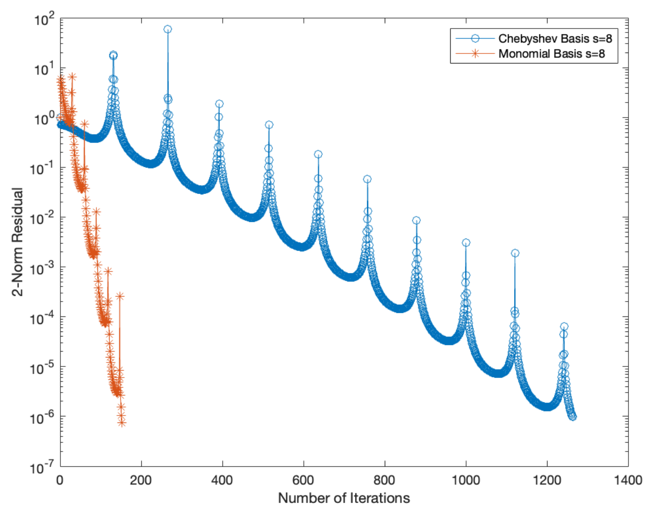


Figure 3.4: plot of the 2-norm residual using the s-step BiCG method, s=8

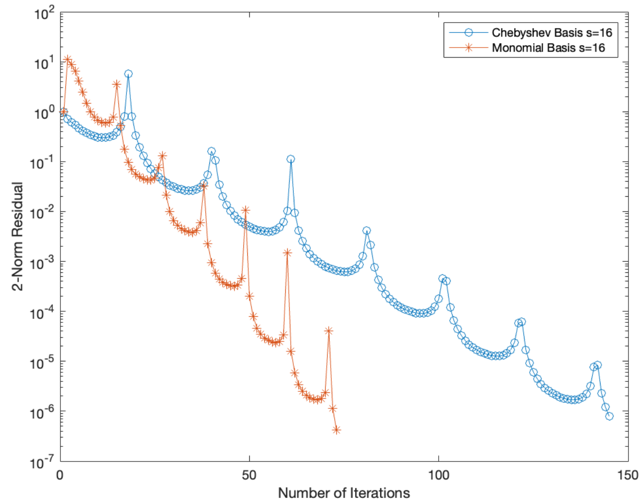


Figure 3.5: plot of the 2-norm residual using the s-step BiCG method, s=16

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:

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

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 $\frac{\|r_m\|_2}{\|b_1\|_2} = 4.0757134858291 66e-11$. 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.

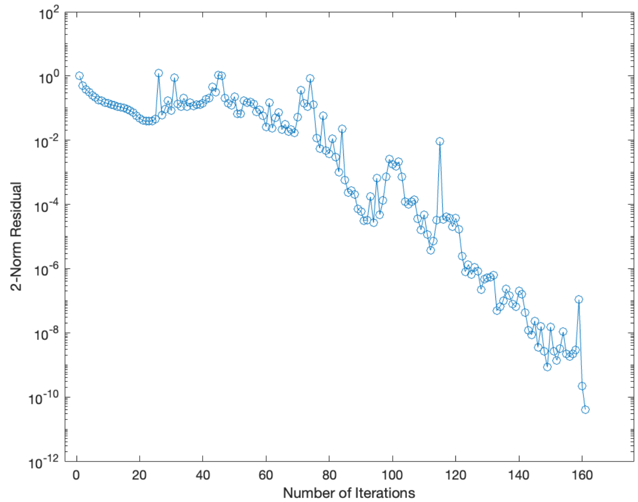


Figure 3.6: plot of the 2-Norm residual using the BiCG method

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.

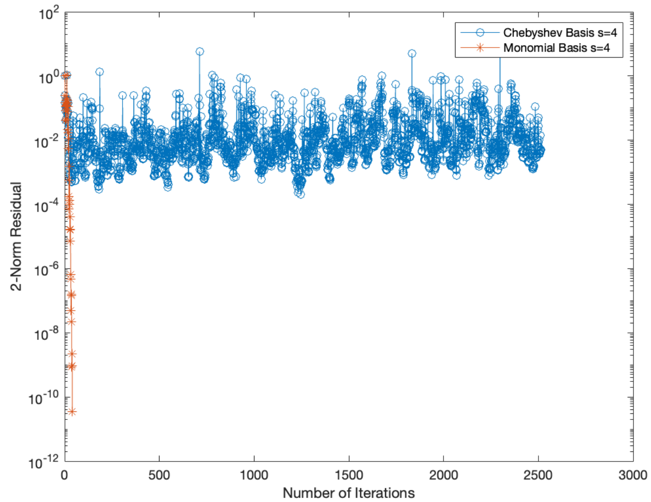


Figure 3.7: plot of the 2-norm residual using the s-step BiCG, s=4, Monomial and Chebyshev bases

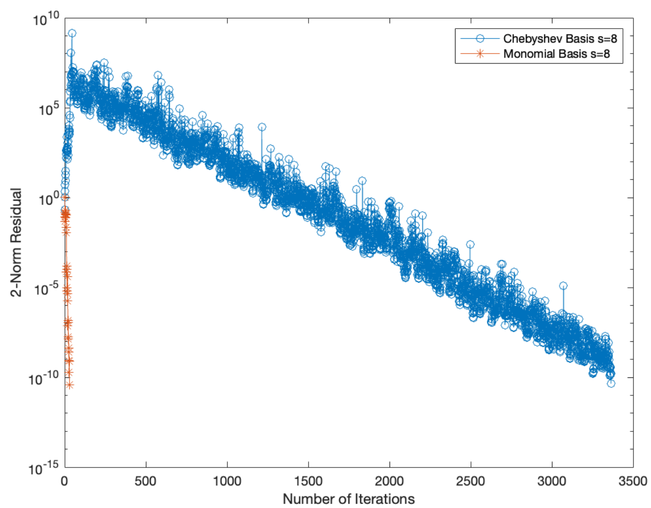


Figure 3.8: plot of the 2-norm residual using the s-step BiCG method, s=8, Monomial and Chebyshev bases

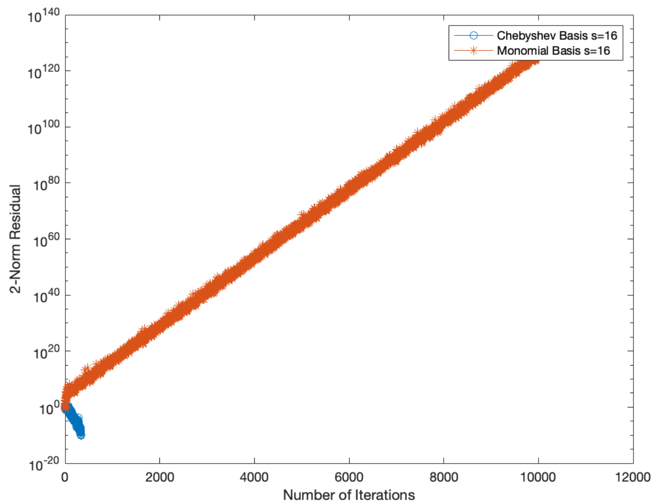


Figure 3.9: plot of the 2-norm residual using the s-step BiCG method, s=16, Monomial and Chebyshev bases

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:

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

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

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

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

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

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

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

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

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

$$\begin{aligned} & r'_{k,j+1} = r'_{k,j} - \alpha_{sk+j} B_k p'_{k,j}, \\ & \text{so we have:} \\ & B_k p'_{k,j} = \frac{1}{\alpha_{sk+j}}(r'_{k,j} - r'_{k,j+1}), \end{aligned} \tag{4.3}$$

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 $j = 0, ..., s-1$. We can write (4.2) as follows:

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

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 $j = 0, ..., s-1$, we get:

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

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:

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

Suppose that:

$$R'_{k,j} = [r'_{k,0}, r'_{k,1}, ..., r'_{k,j}],$$
$$P'_{k,j} = [p'_{k,0}, p'_{k,1}, ..., p'_{k,j}],$$

so we can write (4.5) as:

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

with :

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

Left multiplying (4.7) by A, we get:

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

Define the following matrices:

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

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

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

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

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

$$V_k B_k P'_{k,j} U_{k,j} = V_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{j+1}. \tag{4.10}$$

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:

$$A\bar{V}_k P'_{k,j} U_{k,j} = \bar{V}_k R'_{k,j} L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{j+1}. \tag{4.11}$$

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

$$A\bar{V}_k R'_{k,j} = \bar{V}_k R'_{k,j} T_{k,j} - \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} r'_{k-1,s-1} e'^T_1 - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{j+1}, \tag{4.12}$$

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

$$\begin{aligned}
\beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e'^T_1 &= \beta_{sk} V_{k-1} B_{k-1} p'_{k-1,s-1} e'^T_1 \\
&\text{Using (4.3)} \\
&= \beta_{sk} V_{k-1} \frac{1}{\alpha_{s(k-1)+s-1}} (r'_{k-1,s-1} - r'_{k-1,s}) e'^T_1 \\
&= \frac{\beta_{sk}}{\alpha_{sk-1}} V_{k-1} (r'_{k-1,s-1}) e'^T_1 + \frac{\beta_{sk}}{\alpha_{sk-1}} V_{k-1} r'_{k-1,s} e'^T_1 \\
&\text{by (4.6)} \\
&= \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_{k-1} (r'_{k-1,s-1}) e'^T_1 + \frac{\beta_{sk}}{\alpha_{sk-1}} \bar{V}_k r'_{k,0} e'^T_1
\end{aligned}$$

and it should also be noticed that:

$$\bar{V}_k r'_{k,0} = \bar{V}_k R'_{k,j} e'_1,$$

here $e'_1 = [1, 0, ..., 0]$. And if we define:

$$T_{k,j} = L_{k,j} \Lambda_{k,j}^{-1} U_{k,j} + e'_1 \frac{\beta_{sk}}{\alpha_{sk-1}} e'^T_1,$$

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

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

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

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

Using (4.12), we have:

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

□

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

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

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]:

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

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

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:

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

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

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

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

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

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

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 $B_k p'_{k,j}$:

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

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

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

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:

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

with

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

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

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

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:

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

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:

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

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:

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

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

Using (4.15) we have:

$$|\phi^r_{k-1}| \leq (2s+1)|V_{k-1}||r'_{k-1,s}|,$$
$$|\phi^p_{k-1}| \leq (2s+1)|V_{k-1}||p'_{k-1,s}|.$$

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:

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

Consider (4.7) and define:

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

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

$$\begin{aligned}
\bar{V}_k R'_{k,j} = \bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk}\bar{V}_{k-1} p'_{k-1,s-1} e'^T_1 + \\
\epsilon\bar{V}_k \Delta_{P'_{k,j}} + \epsilon(V_{k-1}\delta_{p'_{k-1,s}} + \phi^r_{k-1} - \phi^p_{k-1}) e'^T_1.
\end{aligned}$$

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

$$\begin{aligned}
A\bar{V}_k R'_{k,j} = A\bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e'^T_1 + \\
\epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1}\delta_{p'_{k-1,s}} + \phi^r_{k-1} - \phi^p_{k-1}) e'^T_1.
\end{aligned} \tag{4.26}$$

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

$$B_k P'_{k,j} = R'_{k,j} L_{k,j} \Lambda^{-1}_{k,j} - \frac{1}{\alpha_m} r'_{k,j+1} e'^T_{j+1} + \epsilon\Delta_{R'_{k,j}}.$$

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

$$V_k B_k P'_{k,j} = V_k R'_{k,j} L_{k,j} \Lambda^{-1}_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e'^T_{j+1} + \epsilon V_k \Delta_{R'_{k,j}}, \tag{4.27}$$

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:

$$v^p_{k,i+1} = \frac{1}{\gamma_i}(A - \mathrm{a}_i I)v^p_{k,i} - \tfrac{\beta_{i-1}}{\gamma_i}v^p_{k,i-1} + \epsilon\delta_{v^p_{k,i+1}},$$

we can rewrite it in this way:

$$Av^p_{k,i} = \gamma_i v^p_{k,i+1} + \mathrm{a}_i v^p_{k,i} + \beta_{i-1}v^p_{k,i-1} - \epsilon\gamma_i\delta_{v^p_{k,i+1}}.$$

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

$$|\delta_{v^p_{k,i+1}}| \leq \frac{1}{|\gamma_i|}((N+2)|A||v^p_{k,i}| + 3|\mathrm{a}_i||v^p_{k,i}| + 2|\beta_{i-1}||v^p_{k,i-1}|).$$

Here N is "the maximum number of nonzeros" in each row of A [2]. In a similar way we can compute $v^r_{k,i+1}$ 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:

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

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:

$$(A\bar{V}_k - \epsilon\Delta_{V_k})P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j}\Lambda^{-1}_{k,j} - \frac{1}{\alpha_m}V_k r'_{k,j+1}e'^T_{j+1} + \epsilon V_k \Delta_{R'_{k,j}},$$

it follows that:

$$A\bar{V}_k P'_{k,j} - \epsilon\Delta_{V_k}P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j}\Lambda^{-1}_{k,j} - \frac{1}{\alpha_m}V_k r'_{k,j+1}e'^T_{j+1} + \epsilon V_k \Delta_{R'_{k,j}},$$

which can be written as:

$$A\bar{V}_k P'_{k,j} = \bar{V}_k R'_{k,j} L_{k,j}\Lambda^{-1}_{k,j} - \frac{1}{\alpha_m}V_k r'_{k,j+1}e'^T_{j+1} + \epsilon V_k \Delta_{R'_{k,j}} + \epsilon\Delta_{V_k}P'_{k,j},$$

and multiplying from the right by $\mathrm{U}_{k,j}$:

$$A\bar{V}_k P'_{k,j}U_{k,j} = \bar{V}_k R'_{k,j} L_{k,j}\Lambda^{-1}_{k,j}U_{k,j} - \frac{1}{\alpha_m}V_k r'_{k,j+1}e'^T_{j+1} + \epsilon(V_k \Delta_{R'_{k,j}} + \Delta_{V_k}P'_{k,j})U_{k,j}. \tag{4.29}$$

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

$$A\bar{V}_k R'_{k,j} + A\bar{V}_k P'_{k,j} U_{k,j} = A\bar{V}_k P'_{k,j} U_{k,j} - \beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1'^T + \\ \epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1}\delta_{p'_{k-1,s}} + \phi^r_{k-1} - \phi^p_{k-1}) e_1'^T + \\ \bar{V}_k R'_{k,j} L_{k,j} \Lambda^{-1}_{k,j} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon (V_k \Delta_{R'_{k,j}} + \Delta_{V_k} P'_{k,j}) U_{k,j}.$$

After computation we have:

$$A\bar{V}_k R'_{k,j} = -\beta_{sk} A\bar{V}_{k-1} p'_{k-1,s-1} e_1'^T + \\ \epsilon A\bar{V}_k \Delta_{P'_{k,j}} + \epsilon A(V_{k-1}\delta_{p'_{k-1,s}} + \phi^r_{k-1} - \phi^p_{k-1}) e_1'^T + \\ \bar{V}_k R'_{k,j} L_{k,j} \Lambda^{-1}_{k,j} U_{k,j} - \frac{1}{\alpha_m} V_k r'_{k,j+1} e_{j+1}'^T + \epsilon (V_k \Delta_{R'_{k,j}} + \Delta_{V_k} P'_{k,j}) U_{k,j}. \tag{4.30}$$

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

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

it follows that:

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

using (4.23) we will have:

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

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:

$$\Delta_{k,j} = (A\bar{V}_k \Delta_{P'_{k,j}} + AV_{k-1} \delta_{p'_{k-1,s}} e_1'^T) \\ + (V_k \Delta_{R'_{k,j}} U_{k,j} - \beta_{sk} V_{k-1} \delta_{r'_{k-1,s-1}} e_1'^T) + (\Delta_{V_k} P'_{k,j} U_{k,j} - \\ + \beta_{sk} \Delta_{V_{k-1}} p'_{k-1,s-1} e_1'^T) + (A(\phi^r_{k-1} - \phi^p_{k-1}) - \frac{\beta_{sk}}{\alpha_{sk-1}} \phi^r_{k-1}) e_1'^T. \tag{4.31}$$

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

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

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

$|\delta_{sk+j}| \leq |A||\bar{V}_k||\delta_{p'_{k,j}}| + |V_k||\delta_{r'_{k,j}}| + |\beta_{sk+j}||V_k||\delta_{r'_{k,j-1}}| + |\Delta_{V_k}||r'_{k,j}|$

here using (4.28) we have:

$$\leq |A||\bar{V}_k|(|r'_{k,j}| + 2|\beta_{sk+j}||p'_{k,j-1}|) + |V_k|((2s+1)|B_k||p'_{k,j}| + \frac{|r'_{k,j}|}{|\alpha_{sk+j}|} + 2|B_k p'_{k,j}|)+$$

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

By the following inequalities:

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

we arrive at:

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

So we have that:

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

For $j = 0$ we have:

$$\delta_{sk} = AV_{k-1}\delta_{p'_{k-1,s}} + V_k\delta_{r'_{k,0}} - \beta_{sk}V_{k-1}\delta_{r'_{k-1,s-1}} + \Delta_{V_k}p'_{k,0} - \beta_{sk}\Delta_{V_{k-1}}p'_{k-1,s-1}+$$
$$(A(\phi^r_{k-1} - \phi^p_{k-1}) - \frac{\beta_{sk}}{\alpha_{sk-1}}\phi^r_{k-1}).$$

Using the norm:

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

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

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

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

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 sprectrum 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 deltx=1/5;
2 A=deltx*[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));
```

```
n=961;
tol=10^-10;
rt1=b1-A*x1;
[x1,k,r1,tr1]=classbicg(A,b1,x1,tol,rt1,n)
```

The command windows for s-step BiCG codes are:

```
%1) example
deltx=1/5;
A=deltx*[1 0 0 0 ; -1 1 0 0 ; 0 -1 1 0 ; 0 0 -1 1];
b1=[1 0 1 0]';
x1=[0 0 0 0]';
tol=10^-6;
I=eye(4);
a=7.9888;
d=7.98;
b=0.010;
[x1,x11,k,kk,r1,r11,tr1,tr11]=step4bicg(A,b1,x1,I,a,b,d,tol)
%[x1,x11,k,kk,r1,r11,tr1,tr11]=step8bicg(A,b1,x1,I,a,b,d,tol)
%[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='cdde1.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

```
function[x1,k,r1,tr1]=classbicg(A,b1,x1,tol,rt1,n)
normb=norm(b1); %normalizing b1
r1=b1-A*x1; %initializing the residual vector
normr=norm(r1); %normalizing the residual vector
tr1=b1-A*x1; %not normalized true residual
p1=r1; %start search direction
pt1=r1; %start search direction tilde
k=1;
rho1=rt1'*r1;
while (norm(r1)/normb>tol)
    sigma=pt1'*A*p1;
    alpha1=rho1/sigma;
    r2=r1-alpha1*A*p1; %residual vector
    x2=x1+alpha1*p1;
    rt2=rt1-alpha1*A'*pt1; %residual tilde vector
```

```
    rho2=rt2'*r2; %rho
    beta2=rho2/rho1;
    p2=r2+beta2*p1; %search direction
    pt2=rt2+beta2*pt1; %search direction tilde vector
    valuex1(:,k)=x1;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    %updating
    p1=p2;
    pt1=pt2;
    r1=r2;
    rt1=rt2;
    x1=x2;
    rho1=rho2;
    tr1=b1-A*x1;
    k=k+1;
    valuex1(:,k)=x1;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    if k==n
        break
    end
   end
semilogy(u1, '-o')
xlabel('Number of Iterations')
ylabel('2-Norm Residual')
```

s-step BiCG code, s=4

```
function[x1,x11,k,kk,r1,r11,tr1,tr11]=step4bicg(A,b1,x1,I,a,b,d,tol)
%residual vectors
r1=b1-A*x1;
x11=x1;
r11=b1-A*x11;
%not normalized true residual vector
tr1=b1-A*x1;
x11=x1;
tr11=b1-A*x11;
%search directions
p1=r1;
p11=r11;
ppt1=p11;
rrt1=r11;
pt1=p1;
rt1=r1;
%coefficient vectors for when we use the Chebyshev basis
p_k0=[1 zeros(1,8)]';
r_k0=[zeros(1,5) 1 zeros(1,3)]';
e_k0=[zeros(1,9)];
pt_k0=[1 zeros(1,8)]';
rt_k0=[zeros(1,5) 1 zeros(1,3)]';
%coefficient vectors for when we make use of the Krylov basis
pp_k0=[1 zeros(1,8)]';
rr_k0=[zeros(1,5) 1 zeros(1,3)]';
ee_k0=[zeros(1,9)];
ppt_k0=[1 zeros(1,8)]';
rrt_k0=[zeros(1,5) 1 zeros(1,3)]';
```

```
%the 5 maximum eigenvalues of the matrix A
eigs(A);
%the minimum eigenvalue of the matrix A
eigs(A,1,'smallestab');
%
c=sqrt(a^2-b^2); %value of the ellipse
%values for the Chebyshev case
aj=d;
g=max(a,b);
beta0=c^2/4*g;
psi_0=2*g;
psi_1=g;
%values for the monomial case
dd=0;
ajj=dd;
bbeta0=0;
ppsi_0=1;
%vectors for the Chebyshev basis
vp_k0=p1;
vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;

vr_k0=r1;
vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;

vtp_k0=pt1;
vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;

vtr_k0=rt1;
vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
%vectors for the monomial basis
vpp_k0=p11;
vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;

vrr_k0=r11;
vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;

vttp_k0=ppt1;
vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
```

```
vttr_k0=rrt1;
vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
%Chebishev basis
V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4];
V_r=[vr_k0,vr_k1,vr_k2,vr_k3];
Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, vtp_k4];
Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3];
V=[V_p, V_r];
Vt=[Vt_p, Vt_r];
G=Vt'*V;
%Monomial basis
VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4];
VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3];
VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, vttp_k4];
VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3];
VV=[VV_p, VV_r];
VVt=[VVt_p, VVt_r];
GG=VVt'*VV;
%matrix B_{k} for when we use Chebyshev basis
B=[aj beta0 0 0 0 0 0 0 0 ;...
    psi_0 aj beta0 0 0 0 0 0 0 ;...
    0 psi_1 aj beta0 0 0 0 0 0 ;...
    0 0 psi_1 aj 0 0 0 0 0 ;...
    0 0 0 psi_1 0 0 0 0 0 ; ...
    0 0 0 0 0 aj beta0 0 0 ;...
    0 0 0 0 0 psi_0 aj beta0 0 ;...
    0 0 0 0 0 0 psi_1 aj 0;...
    0 0 0 0 0 0 0 psi_1 0];
%matrix B_{k} for when we use monomial basis
BB=[ajj bbeta0 0 0 0 0 0 0 0 ;...
    ppsi_0 ajj bbeta0 0 0 0 0 0 0 ;...
    0 ppsi_0 ajj bbeta0 0 0 0 0 0 ;...
    0 0 ppsi_0 ajj 0 0 0 0 0 ;...
    0 0 0 ppsi_0 0 0 0 0 0 ; ...
    0 0 0 0 0 ajj bbeta0 0 0 ;...
    0 0 0 0 0 ppsi_0 ajj bbeta0 0 ;...
    0 0 0 0 0 0 ppsi_0 ajj 0;...
    0 0 0 0 0 0 0 ppsi_0 0];
%loops for when we use the monomial basis
s=4;
kk=1; %starting value
n=2513; %maximum number of iterations for s=4
normb=norm(b1);
while (norm(r11)/normb>tol)
    pp_k0=[1 zeros(1,8)]'; %coefficient vectors
    rr_k0=[zeros(1,5) 1 zeros(1,3)]';
    ee_k0=[zeros(1,9)];
    ppt_k0=[1 zeros(1,8)]';
    rrt_k0=[zeros(1,5) 1 zeros(1,3)]';
    % inside loop
    delt1=rrt_k0'*GG*rr_k0;
    for j=0 : s-1
        alpha1=delt1/(ppt_k0'*GG*BB*pp_k0);
        ee_k1=ee_k0+alpha1*pp_k0';
        rr_k1=rr_k0-BB*(alpha1*pp_k0);
```

```
        rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
        delt3=rrt_k1'*GG*rr_k1;
        betaa=delt3/delt1;
        pp_k1=rr_k1+betaa*pp_k0;
        ppt_k1=rrt_k1+betaa*ppt_k0;
        rr_k0=rr_k1;
        rrt_k0=rrt_k1;
        pp_k0=pp_k1;
        ppt_k0=ppt_k1;
        delt1=delt3;
        ee_k0=ee_k1;
    end
    xmm=VV*ee_k0'+x11;
    rmm=VV*rr_k0;
    pmm=VV*pp_k0;
    rrtm=VVt*rrt_k0;
    pptm=VVt*ppt_k0;
    valuer2(:,kk)=r11;
    u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
    x11=xmm;
    r11=rmm;
    p11=pmm;
    ppt1=pptm;
    rrt1=rrtm;
    tr11=b1-A*x11;
    vpp_k0=p11;
    vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
    vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
    vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
    vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
    vrr_k0=r11;
    vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
    vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
    vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
    vttp_k0=ppt1;
    vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
    vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
    vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
    vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
    vttr_k0=rrt1;
    vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
    vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
    vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
    %Krylov matrices
    VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4];
    VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3];
    VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, vttp_k4];
    VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3];
    VV=[VV_p, VV_r];
    VVt=[VVt_p, VVt_r];
    GG=VVt'*VV;
    kk=kk+1;
    valuer2(:,kk)=r11;
    u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
    if kk==n
          break;
    end
```

```
end
%loops using Chebyshev basis
s=4
k=1;
n=2513;
normb=norm(b1);
while (norm(r1)/normb>tol)
    p_k0=[1 zeros(1,8)]'; %coefficient vectors
    r_k0=[zeros(1,5) 1 zeros(1,3)]';
    e_k0=[zeros(1,9)];
    pt_k0=[1 zeros(1,8)]';
    rt_k0=[zeros(1,5) 1 zeros(1,3)]';
    delt=rt_k0'*G*r_k0;
    for j=0 : s-1
        alpha=delt/(pt_k0'*G*B*p_k0);
        e_k1=e_k0+alpha*p_k0';
        r_k1=r_k0-B*(alpha*p_k0);
        rt_k1=rt_k0-B*(alpha*pt_k0);
        delt2=rt_k1'*G*r_k1;
        beta=delt2/delt;
        p_k1=r_k1+beta*p_k0;
        pt_k1=rt_k1+beta*pt_k0;
        %updating
        r_k0=r_k1;
        rt_k0=rt_k1;
        p_k0=p_k1;
        pt_k0=pt_k1;
        delt=delt2;
        e_k0=e_k1;
    end
    xm=V*e_k0'+x1;
    rm=V*r_k0;
    pm=V*p_k0;
    rtm=Vt*rt_k0;
    ptm=Vt*pt_k0;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    x1=xm;
    r1=rm;
    p1=pm;
    pt1=ptm;
    rt1=rtm;
    tr1=b1-A*x1;
    vp_k0=p1;
    vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
    vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
    vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
    vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
    vr_k0=r1;
    vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
    vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
    vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
    vtp_k0=pt1;
    vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
    vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
    vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
```

```
    vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
    vtr_k0=rt1;
    vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
    vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
    vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
    V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4];
    V_r=[vr_k0,vr_k1,vr_k2,vr_k3];
    Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, vtp_k4];
    Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3];
    V=[V_p, V_r];
    Vt=[Vt_p, Vt_r];
    G=Vt'*V;
    k=k+1;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    if k==n
        break
    end
end
%plot for both bases
semilogy(u1,'-o')
xlabel('Number of Iterations')
ylabel('2-Norm Residual')
hold on
semilogy(u2, '-*')
legend ('Chebyshev Basis s=4 ', 'Monomial Basis s=4')
hold off
```

s-step BiCG code, s=8

```
function[x1,x11,k,kk,r1,r11,tr1,tr11]=step8bicg(A,b1,x1,I,a,b,d,tol)
%residual vectors
r1=b1-A*x1;
x11=x1;
r11=b1-A*x11;
tr1=b1-A*x1; %not normalized true residual vector
x11=x1;
tr11=b1-A*x11;
%search directions
p1=r1;
p11=r11;
ppt1=p11;
rrt1=r11;
pt1=p1;
rt1=r1;
%coefficient vectors for when we make use of the Chebyshev basis
p_k0=[1 zeros(1,16)]';
r_k0=[zeros(1,9) 1 zeros(1,7)]';
e_k0=[zeros(1,17)];
pt_k0=[1 zeros(1,16)]';
rt_k0=[zeros(1,9) 1 zeros(1,7)]';
%coefficients vectors for when we use monomial basis
pp_k0=[1 zeros(1,16)]';
rr_k0=[zeros(1,9) 1 zeros(1,7)]';
ee_k0=[zeros(1,17)];
ppt_k0=[1 zeros(1,16)]';
rrt_k0=[zeros(1,9) 1 zeros(1,7)]';
```

```
%the 5 maximum eigenvalues of the matrix A
eigs(A);
%the minimum eigenvalue of the matrix A
eigs(A,1,'smallestab');
c=sqrt(a^2-b^2); %value of the ellipse
%values for when we use the Chebyshev basis
aj=d;
g=max(a,b);
beta0=c^2/4*g;
psi_0=2*g;
psi_1=g;
%values for when we use the monomial Basis
dd=0;
ajj=dd;
bbeta0=0;
ppsi_0=1;
%Vectors for the Chebyshev basis
vp_k0=p1;
vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;

vr_k0=r1;
vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;

vtp_k0=pt1;
vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;

vtr_k0=rt1;
vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
%vectors for the monomial basis
vpp_k0=p11;
vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
```

```
vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;

vrr_k0=r11;
vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;

vttp_k0=ppt1;
vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;

vttr_k0=rrt1;
vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
%Chebyshev basis s=8
V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4,vp_k5,vp_k6,vp_k7,vp_k8];
V_r=[vr_k0,vr_k1,vr_k2,vr_k3,vr_k4,vr_k5,vr_k6,vr_k7];
Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, ...
    vtp_k4,vtp_k5,vtp_k6,vtp_k7,vtp_k8];
Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3,vtr_k4,vtr_k5,vtr_k6,vtr_k7];
V=[V_p, V_r];
Vt=[Vt_p, Vt_r];
G=Vt'*V;
%Krylov basis s=8
VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4,vpp_k5,vpp_k6,vpp_k7,vpp_k8];
VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3,vrr_k4,vrr_k5,vrr_k6,vrr_k7];
VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, vttp_k4,vttp_k5,vttp_k6,...
    vttp_k7,vttp_k8];
VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3,vttr_k4,vttr_k5,vttr_k6,vttr_k7];
VV=[VV_p, VV_r];
VVt=[VVt_p, VVt_r];
GG=VVt'*VV;
%matrix B_{k} for when we use the Chebyshev basis
B=eye(17);
for i=1:17
    B(i,i)=aj;
```

```
end
for i=1:16
    B(i,i+1)=beta0;
    B(i+1,i)=psi_1;
end
B(2,1)=psi_0;
B(11,10)=psi_0;
B(17,17)=0;
B(9,9)=0;
B(16,17)=0;
B(8,9)=0;
B(10,9)=0;
B(9,10)=0;
%matrix B_{k} for when we use the monomial basis
BB=eye(17);
for i=1:17
    BB(i,i)=ajj;
end
for i=1:16
    BB(i,i+1)=bbeta0;
    BB(i+1,i)=ppsi_0;
end

BB(17,17)=0;
BB(9,9)=0;
BB(16,17)=0;
BB(8,9)=0;
BB(10,9)=0;
s=8;
kk=1; %starting value
n=5025; %maximum number for s=8
normb=norm(b1)
while (norm(r11)/normb>tol)
    pp_k0=[1 zeros(1,16)]';
    rr_k0=[zeros(1,9) 1 zeros(1,7)]';
    ee_k0=[zeros(1,17)];
    ppt_k0=[1 zeros(1,16)]';
    rrt_k0=[zeros(1,9) 1 zeros(1,7)]';
    delt1=rrt_k0'*GG*rr_k0;
    for j=0 : s-1
        alpha1=delt1/(ppt_k0'*GG*BB*pp_k0);
        ee_k1=ee_k0+alpha1*pp_k0';
        rr_k1=rr_k0-BB*(alpha1*pp_k0);
        rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
        delt3=rrt_k1'*GG*rr_k1;
        betaa=delt3/delt1;
        pp_k1=rr_k1+betaa*pp_k0;
        ppt_k1=rrt_k1+betaa*ppt_k0;
        rr_k0=rr_k1;
        rrt_k0=rrt_k1;
        pp_k0=pp_k1;
        ppt_k0=ppt_k1;
        delt1=delt3;
        ee_k0=ee_k1;
    end
    xmm=VV*ee_k0'+x11;
    rmm=VV*rr_k0;
```

```
pmm=VV*pp_k0;
rrtm=VVt*rrt_k0;
pptm=VVt*ppt_k0;
valuer2(:,kk)=r11;
u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
x11=xmm;
r11=rmm;
p11=pmm;
ppt1=pptm;
rrt1=rrtm;
tr11=b1-A*x11; %true residual
vpp_k0=p11;
vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;

vrr_k0=r11;
vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;

vttp_k0=ppt1;
vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;

vttr_k0=rrt1;
vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
vttr_k8=(1/ppsi_0)*(A'-dd*I)*vttr_k7-(bbeta0/ppsi_0)*vttr_k6;

VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4,vpp_k5,vpp_k6,vpp_k7,vpp_k8];
VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3,vrr_k4,vrr_k5,vrr_k6,vrr_k7];
VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, ...
    vttp_k4,vttp_k5,vttp_k6,vttp_k7,vttp_k8];
VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3,vttr_k4,vttr_k5,vttr_k6,vttr_k7];
VV=[VV_p, VV_r];
VVt=[VVt_p, VVt_r];
```

```
    GG=VVt'*VV;
    kk=kk+1;
    valuer2(:,kk)=r11;
    u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
    if kk==n
          break;
    end

end
s=8
k=1;
n=5025
normb=norm(b1);
while (norm(r1)/normb>tol)
     p_k0=[1 zeros(1,16)]';
     r_k0=[zeros(1,9) 1 zeros(1,7)]';
     e_k0=[zeros(1,17)];
     pt_k0=[1 zeros(1,16)]';
     rt_k0=[zeros(1,9) 1 zeros(1,7)]';
     delt=rt_k0'*G*r_k0;
    for j=0 : s-1
        alpha=delt/(pt_k0'*G*B*p_k0);
        e_k1=e_k0+alpha*p_k0';
        r_k1=r_k0-B*(alpha*p_k0);
        rt_k1=rt_k0-B*(alpha*pt_k0);
        delt2=rt_k1'*G*r_k1;
        beta=delt2/delt;
        p_k1=r_k1+beta*p_k0;
        pt_k1=rt_k1+beta*pt_k0;
        %updating
        r_k0=r_k1;
        rt_k0=rt_k1;
        p_k0=p_k1;
        pt_k0=pt_k1;
        delt=delt2;
        e_k0=e_k1;
    end
    xm=V*e_k0'+x1;
    rm=V*r_k0;
    pm=V*p_k0;
    rtm=Vt*rt_k0;
    ptm=Vt*pt_k0;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    x1=xm;
    r1=rm;
    p1=pm;
    pt1=ptm;
    rt1=rtm;
    tr1=b1-A*x1;
    vp_k0=p1;
    vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
    vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
    vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
    vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
    vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
    vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
```

```
    vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
    vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;

    vr_k0=r1;
    vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
    vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
    vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
    vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
    vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
    vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
    vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;

    vtp_k0=pt1;
    vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
    vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
    vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
    vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
    vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
    vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
    vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
    vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;

    vtr_k0=rt1;
    vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
    vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
    vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
    vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
    vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
    vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
    vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;

    V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4,vp_k5,vp_k6,vp_k7,vp_k8];
    V_r=[vr_k0,vr_k1,vr_k2,vr_k3,vr_k4,vr_k5,vr_k6,vr_k7];
    Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, ...
        vtp_k4,vtp_k5,vtp_k6,vtp_k7,vtp_k8];
    Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3,vtr_k4,vtr_k5,vtr_k6,vtr_k7];
    V=[V_p, V_r];
    Vt=[Vt_p, Vt_r];
    G=Vt'*V;
    k=k+1;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    if k==n
        break
    end

end
%plot for both bases
semilogy(u1,'-o')
xlabel('Number of Iterations')
ylabel('2-Norm Residual')
hold on
semilogy(u2, '-*')
legend ('Chebyshev Basis s=8 ', 'Monomial Basis s=8')
hold off
```

s-step BiCG code, s=16

```
function[x1,x11,k,kk,r1,r11,tr1,tr11]=step16bicg(A,b1,x1,I,a,b,d,tol)
%residual vectors
r1=b1-A*x1;
x11=x1;
r11=b1-A*x11;
%not normalized true residual vector
tr1=b1-A*x1;
x11=x1;
tr11=b1-A*x11;
%search directions
p1=r1;
p11=r11;
ppt1=p11;
rrt1=r11;
pt1=p1;
rt1=r1;
%coefficient vectors for when we use Chebyshev basis
p_k0=[1 zeros(1,32)]';
r_k0=[zeros(1,17) 1 zeros(1,15)]';
e_k0=zeros(1,33);
pt_k0=[1 zeros(1,32)]';
rt_k0=[zeros(1,17) 1 zeros(1,15)]';
%coefficients vectors for when we make use of monomial basis
pp_k0=[1 zeros(1,32)]';
rr_k0=[zeros(1,17) 1 zeros(1,15)]';
ee_k0=zeros(1,33);
ppt_k0=[1 zeros(1,32)]';
rrt_k0=[zeros(1,17) 1 zeros(1,15)]';
%the 5 maximum eigenvalues of the matrix A
eigs(A);
%the minimum eigenvalue of the matrix A
eigs(A,1,'smallestab');
c=sqrt(a^2-b^2); %value of the ellipse
%values for when we use the Chebyshev basis
aj=d;
g=max(a,b);
beta0=c^2/4*g;
psi_0=2*g;
psi_1=g;
%values for when we use the monomial Basis
dd=0;
ajj=dd;
bbeta0=0;
ppsi_0=1;
%vectors for the Chebyshev basis
vp_k0=p1;
vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
vp_k9=(1/psi_1)*(A-d*I)*vp_k8-(beta0/psi_1)*vp_k7;
vp_k10=(1/psi_1)*(A-d*I)*vp_k9-(beta0/psi_1)*vp_k8;
vp_k11=(1/psi_1)*(A-d*I)*vp_k10-(beta0/psi_1)*vp_k9;
```

```
vp_k12=(1/psi_1)*(A-d*I)*vp_k11-(beta0/psi_1)*vp_k10;
vp_k13=(1/psi_1)*(A-d*I)*vp_k12-(beta0/psi_1)*vp_k11;
vp_k14=(1/psi_1)*(A-d*I)*vp_k13-(beta0/psi_1)*vp_k12;
vp_k15=(1/psi_1)*(A-d*I)*vp_k14-(beta0/psi_1)*vp_k13;
vp_k16=(1/psi_1)*(A-d*I)*vp_k15-(beta0/psi_1)*vp_k14;
%
vr_k0=r1;
vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
vr_k8=(1/psi_1)*(A-d*I)*vr_k7-(beta0/psi_1)*vr_k6;
vr_k9=(1/psi_1)*(A-d*I)*vr_k8-(beta0/psi_1)*vr_k7;
vr_k10=(1/psi_1)*(A-d*I)*vr_k9-(beta0/psi_1)*vr_k8;
vr_k11=(1/psi_1)*(A-d*I)*vr_k10-(beta0/psi_1)*vr_k9;
vr_k12=(1/psi_1)*(A-d*I)*vr_k11-(beta0/psi_1)*vr_k10;
vr_k13=(1/psi_1)*(A-d*I)*vr_k12-(beta0/psi_1)*vr_k11;
vr_k14=(1/psi_1)*(A-d*I)*vr_k13-(beta0/psi_1)*vr_k12;
vr_k15=(1/psi_1)*(A-d*I)*vr_k14-(beta0/psi_1)*vr_k13;
%
vtp_k0=pt1;
vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
vtp_k9=(1/psi_1)*(A'-d*I)*vtp_k8-(beta0/psi_1)*vtp_k7;
vtp_k10=(1/psi_1)*(A'-d*I)*vtp_k9-(beta0/psi_1)*vtp_k8;
vtp_k11=(1/psi_1)*(A'-d*I)*vtp_k10-(beta0/psi_1)*vtp_k9;
vtp_k12=(1/psi_1)*(A'-d*I)*vtp_k11-(beta0/psi_1)*vtp_k10;
vtp_k13=(1/psi_1)*(A'-d*I)*vtp_k12-(beta0/psi_1)*vtp_k11;
vtp_k14=(1/psi_1)*(A'-d*I)*vtp_k13-(beta0/psi_1)*vtp_k12;
vtp_k15=(1/psi_1)*(A'-d*I)*vtp_k14-(beta0/psi_1)*vtp_k13;
vtp_k16=(1/psi_1)*(A'-d*I)*vtp_k15-(beta0/psi_1)*vtp_k14;
%
vtr_k0=rt1;
vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
vtr_k8=(1/psi_1)*(A'-d*I)*vtr_k7-(beta0/psi_1)*vtr_k6;
vtr_k9=(1/psi_1)*(A'-d*I)*vtr_k8-(beta0/psi_1)*vtr_k7;
vtr_k10=(1/psi_1)*(A'-d*I)*vtr_k9-(beta0/psi_1)*vtr_k8;
vtr_k11=(1/psi_1)*(A'-d*I)*vtr_k10-(beta0/psi_1)*vtr_k9;
vtr_k12=(1/psi_1)*(A'-d*I)*vtr_k11-(beta0/psi_1)*vtr_k10;
vtr_k13=(1/psi_1)*(A'-d*I)*vtr_k12-(beta0/psi_1)*vtr_k11;
vtr_k14=(1/psi_1)*(A'-d*I)*vtr_k13-(beta0/psi_1)*vtr_k12;
vtr_k15=(1/psi_1)*(A'-d*I)*vtr_k14-(beta0/psi_1)*vtr_k13;
```

```
%vectors for the monomial basis
vpp_k0=p11;
vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
vpp_k9=(1/ppsi_0)*(A-dd*I)*vpp_k8-(bbeta0/ppsi_0)*vpp_k7;
vpp_k10=(1/ppsi_0)*(A-dd*I)*vpp_k9-(bbeta0/ppsi_0)*vpp_k8;
vpp_k11=(1/ppsi_0)*(A-dd*I)*vpp_k10-(bbeta0/ppsi_0)*vpp_k9;
vpp_k12=(1/ppsi_0)*(A-dd*I)*vpp_k11-(bbeta0/ppsi_0)*vpp_k10;
vpp_k13=(1/ppsi_0)*(A-dd*I)*vpp_k12-(bbeta0/ppsi_0)*vpp_k11;
vpp_k14=(1/ppsi_0)*(A-dd*I)*vpp_k13-(bbeta0/ppsi_0)*vpp_k12;
vpp_k15=(1/ppsi_0)*(A-dd*I)*vpp_k14-(bbeta0/ppsi_0)*vpp_k13;
vpp_k16=(1/ppsi_0)*(A-dd*I)*vpp_k15-(bbeta0/ppsi_0)*vpp_k14;
%
vrr_k0=r11;
vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
vrr_k8=(1/ppsi_0)*(A-dd*I)*vrr_k7-(bbeta0/ppsi_0)*vrr_k6;
vrr_k9=(1/ppsi_0)*(A-dd*I)*vrr_k8-(bbeta0/ppsi_0)*vrr_k7;
vrr_k10=(1/ppsi_0)*(A-dd*I)*vrr_k9-(bbeta0/ppsi_0)*vrr_k8;
vrr_k11=(1/ppsi_0)*(A-dd*I)*vrr_k10-(bbeta0/ppsi_0)*vrr_k9;
vrr_k12=(1/ppsi_0)*(A-dd*I)*vrr_k11-(bbeta0/ppsi_0)*vrr_k10;
vrr_k13=(1/ppsi_0)*(A-dd*I)*vrr_k12-(bbeta0/ppsi_0)*vrr_k11;
vrr_k14=(1/ppsi_0)*(A-dd*I)*vrr_k13-(bbeta0/ppsi_0)*vrr_k12;
vrr_k15=(1/ppsi_0)*(A-dd*I)*vrr_k14-(bbeta0/ppsi_0)*vrr_k13;
%
vttp_k0=ppt1;
vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;
vttp_k9=(1/ppsi_0)*(A'-dd*I)*vttp_k8-(bbeta0/ppsi_0)*vttp_k7;
vttp_k10=(1/ppsi_0)*(A'-dd*I)*vttp_k9-(bbeta0/ppsi_0)*vttp_k8;
vttp_k11=(1/ppsi_0)*(A'-dd*I)*vttp_k10-(bbeta0/ppsi_0)*vttp_k9;
vttp_k12=(1/ppsi_0)*(A'-dd*I)*vttp_k11-(bbeta0/ppsi_0)*vttp_k10;
vttp_k13=(1/ppsi_0)*(A'-dd*I)*vttp_k12-(bbeta0/ppsi_0)*vttp_k11;
vttp_k14=(1/ppsi_0)*(A'-dd*I)*vttp_k13-(bbeta0/ppsi_0)*vttp_k12;
vttp_k15=(1/ppsi_0)*(A'-dd*I)*vttp_k14-(bbeta0/ppsi_0)*vttp_k13;
vttp_k16=(1/ppsi_0)*(A'-dd*I)*vttp_k15-(bbeta0/ppsi_0)*vttp_k14;
%
vttr_k0=rrt1;
vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
```

```
vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
vttr_k8=(1/ppsi_0)*(A'-dd*I)*vttr_k7-(bbeta0/ppsi_0)*vttr_k6;
vttr_k9=(1/ppsi_0)*(A'-dd*I)*vttr_k8-(bbeta0/ppsi_0)*vttr_k7;
vttr_k10=(1/ppsi_0)*(A'-dd*I)*vttr_k9-(bbeta0/ppsi_0)*vttr_k8;
vttr_k11=(1/ppsi_0)*(A'-dd*I)*vttr_k10-(bbeta0/ppsi_0)*vttr_k9;
vttr_k12=(1/ppsi_0)*(A'-dd*I)*vttr_k11-(bbeta0/ppsi_0)*vttr_k10;
vttr_k13=(1/ppsi_0)*(A'-dd*I)*vttr_k12-(bbeta0/ppsi_0)*vttr_k11;
vttr_k14=(1/ppsi_0)*(A'-dd*I)*vttr_k13-(bbeta0/ppsi_0)*vttr_k12;
vttr_k15=(1/ppsi_0)*(A'-dd*I)*vttr_k14-(bbeta0/ppsi_0)*vttr_k13;
%Chebyshev basis
V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4,vp_k5,vp_k6,vp_k7,vp_k8,...
    vp_k9,vp_k10,vp_k11,vp_k12,vp_k13,vp_k14,vp_k15,vp_k16];
V_r=[vr_k0,vr_k1,vr_k2,vr_k3,vr_k4,vr_k5,vr_k6,vr_k7,vr_k8,...
    vr_k9,vr_k10,vr_k11,vr_k12,vr_k13,vr_k14,vr_k15];
Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, vtp_k4,vtp_k5,vtp_k6,...
    vtp_k7,vtp_k8,vtp_k9,vtp_k10,vtp_k11,vtp_k12, vtp_k13,...
    vtp_k14,vtp_k15,vtp_k16];
Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3,vtr_k4,vtr_k5,vtr_k6,...
    vtr_k7,vtr_k8,vtr_k9,vtr_k10,vtr_k11,vtr_k12,vtr_k13,vtr_k14,vtr_k15];
V=[V_p, V_r];
Vt=[Vt_p, Vt_r];
G=Vt'*V;
%Monomial basis
VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4,vpp_k5,vpp_k6,...
    vpp_k7,vpp_k8,vpp_k9,vpp_k10,vpp_k11,vpp_k12,vpp_k13,...
    vpp_k14,vpp_k15,vpp_k16];
VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3,vrr_k4,vrr_k5,vrr_k6,...
    vrr_k7,vrr_k8,vrr_k9,vrr_k10,vrr_k11,vrr_k12,vrr_k13,...
    vrr_k14,vrr_k15];
VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, vttp_k4,vttp_k5,vttp_k6,...
    vttp_k7,vttp_k8,vttp_k9,vttp_k10,vttp_k11,vttp_k12, vttp_k13,...
    vttp_k14,vttp_k15,vttp_k16];
VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3,vttr_k4,vttr_k5,vttr_k6,...
    vttr_k7,vttr_k8,vttr_k9,vttr_k10,vttr_k11,vttr_k12,vttr_k13,...
    vttr_k14,vttr_k15];
VV=[VV_p, VV_r];
VVt=[VVt_p, VVt_r];
GG=VVt'*VV;
%matrix B_{k} for when we use the Chebyshev basis
B=eye(33);
for i=1:33
    B(i,i)=aj;
end
for i=1:32
    B(i,i+1)=beta0;
    B(i+1,i)=psi_1;
end
B(2,1)=psi_0;
B(19,18)=psi_0;
B(33,33)=0;
B(17,17)=0;
B(32,33)=0;
B(16,17)=0;
```

```
B(18,17)=0;
B(17,18)=0;
%matrix B_{k} for when we use the monomial basis
BB=eye(33);
for i=1:33
    BB(i,i)=ajj;
end
for i=1:32
    BB(i,i+1)=bbeta0;
    BB(i+1,i)=ppsi_0;
end

BB(33,33)=0;
BB(17,17)=0;
BB(32,33)=0;
BB(16,17)=0;
BB(18,17)=0;
%starting the loops
s=16
kk=1
n=10049 %maximum number for s=16
normb=norm(b1)
while (norm(r11)/normb>tol)
    pp_k0=[1 zeros(1,32)]';
    rr_k0=[zeros(1,17) 1 zeros(1,15)]';
    ee_k0=zeros(1,33);
    ppt_k0=[1 zeros(1,32)]';
    rrt_k0=[zeros(1,17) 1 zeros(1,15)]';
    delt1=rrt_k0'*GG*rr_k0;
    for j=0 : s-1
        alpha1=delt1/(ppt_k0'*GG*BB*pp_k0);
        ee_k1=ee_k0+alpha1*pp_k0';
        rr_k1=rr_k0-BB*(alpha1*pp_k0);
        rrt_k1=rrt_k0-BB*(alpha1*ppt_k0);
        delt3=rrt_k1'*GG*rr_k1;
        betaa=delt3/delt1;
        pp_k1=rr_k1+betaa*pp_k0;
        ppt_k1=rrt_k1+betaa*ppt_k0;
        rr_k0=rr_k1;
        rrt_k0=rrt_k1;
        pp_k0=pp_k1;
        ppt_k0=ppt_k1;
        delt1=delt3;
        ee_k0=ee_k1;
    end
    xmm=VV*ee_k0'+x11;
    rmm=VV*rr_k0;
    pmm=VV*pp_k0;
    rrtm=VVt*rrt_k0;
    pptm=VVt*ppt_k0;
    valuer2(:,kk)=r11;
    u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
    x11=xmm;
    r11=rmm;
    p11=pmm;
    ppt1=pptm;
    rrt1=rrtm;
```

```
tr11=b1-A*x11;
%vectors for the monomial basis
vpp_k0=p11;
vpp_k1=(1/ppsi_0)*(A-dd*I)*vpp_k0;
vpp_k2=(1/ppsi_0)*(A-dd*I)*vpp_k1-(bbeta0/ppsi_0)*vpp_k0;
vpp_k3=(1/ppsi_0)*(A-dd*I)*vpp_k2-(bbeta0/ppsi_0)*vpp_k1;
vpp_k4=(1/ppsi_0)*(A-dd*I)*vpp_k3-(bbeta0/ppsi_0)*vpp_k2;
vpp_k5=(1/ppsi_0)*(A-dd*I)*vpp_k4-(bbeta0/ppsi_0)*vpp_k3;
vpp_k6=(1/ppsi_0)*(A-dd*I)*vpp_k5-(bbeta0/ppsi_0)*vpp_k4;
vpp_k7=(1/ppsi_0)*(A-dd*I)*vpp_k6-(bbeta0/ppsi_0)*vpp_k5;
vpp_k8=(1/ppsi_0)*(A-dd*I)*vpp_k7-(bbeta0/ppsi_0)*vpp_k6;
vpp_k9=(1/ppsi_0)*(A-dd*I)*vpp_k8-(bbeta0/ppsi_0)*vpp_k7;
vpp_k10=(1/ppsi_0)*(A-dd*I)*vpp_k9-(bbeta0/ppsi_0)*vpp_k8;
vpp_k11=(1/ppsi_0)*(A-dd*I)*vpp_k10-(bbeta0/ppsi_0)*vpp_k9;
vpp_k12=(1/ppsi_0)*(A-dd*I)*vpp_k11-(bbeta0/ppsi_0)*vpp_k10;
vpp_k13=(1/ppsi_0)*(A-dd*I)*vpp_k12-(bbeta0/ppsi_0)*vpp_k11;
vpp_k14=(1/ppsi_0)*(A-dd*I)*vpp_k13-(bbeta0/ppsi_0)*vpp_k12;
vpp_k15=(1/ppsi_0)*(A-dd*I)*vpp_k14-(bbeta0/ppsi_0)*vpp_k13;
vpp_k16=(1/ppsi_0)*(A-dd*I)*vpp_k15-(bbeta0/ppsi_0)*vpp_k14;

vrr_k0=r11;
vrr_k1=(1/ppsi_0)*(A-dd*I)*vrr_k0;
vrr_k2=(1/ppsi_0)*(A-dd*I)*vrr_k1-(bbeta0/ppsi_0)*vrr_k0;
vrr_k3=(1/ppsi_0)*(A-dd*I)*vrr_k2-(bbeta0/ppsi_0)*vrr_k1;
vrr_k4=(1/ppsi_0)*(A-dd*I)*vrr_k3-(bbeta0/ppsi_0)*vrr_k2;
vrr_k5=(1/ppsi_0)*(A-dd*I)*vrr_k4-(bbeta0/ppsi_0)*vrr_k3;
vrr_k6=(1/ppsi_0)*(A-dd*I)*vrr_k5-(bbeta0/ppsi_0)*vrr_k4;
vrr_k7=(1/ppsi_0)*(A-dd*I)*vrr_k6-(bbeta0/ppsi_0)*vrr_k5;
vrr_k8=(1/ppsi_0)*(A-dd*I)*vrr_k7-(bbeta0/ppsi_0)*vrr_k6;
vrr_k9=(1/ppsi_0)*(A-dd*I)*vrr_k8-(bbeta0/ppsi_0)*vrr_k7;
vrr_k10=(1/ppsi_0)*(A-dd*I)*vrr_k9-(bbeta0/ppsi_0)*vrr_k8;
vrr_k11=(1/ppsi_0)*(A-dd*I)*vrr_k10-(bbeta0/ppsi_0)*vrr_k9;
vrr_k12=(1/ppsi_0)*(A-dd*I)*vrr_k11-(bbeta0/ppsi_0)*vrr_k10;
vrr_k13=(1/ppsi_0)*(A-dd*I)*vrr_k12-(bbeta0/ppsi_0)*vrr_k11;
vrr_k14=(1/ppsi_0)*(A-dd*I)*vrr_k13-(bbeta0/ppsi_0)*vrr_k12;
vrr_k15=(1/ppsi_0)*(A-dd*I)*vrr_k14-(bbeta0/ppsi_0)*vrr_k13;

vttp_k0=ppt1;
vttp_k1=(1/ppsi_0)*(A'-dd*I)*vttp_k0;
vttp_k2=(1/ppsi_0)*(A'-dd*I)*vttp_k1-(bbeta0/ppsi_0)*vttp_k0;
vttp_k3=(1/ppsi_0)*(A'-dd*I)*vttp_k2-(bbeta0/ppsi_0)*vttp_k1;
vttp_k4=(1/ppsi_0)*(A'-dd*I)*vttp_k3-(bbeta0/ppsi_0)*vttp_k2;
vttp_k5=(1/ppsi_0)*(A'-dd*I)*vttp_k4-(bbeta0/ppsi_0)*vttp_k3;
vttp_k6=(1/ppsi_0)*(A'-dd*I)*vttp_k5-(bbeta0/ppsi_0)*vttp_k4;
vttp_k7=(1/ppsi_0)*(A'-dd*I)*vttp_k6-(bbeta0/ppsi_0)*vttp_k5;
vttp_k8=(1/ppsi_0)*(A'-dd*I)*vttp_k7-(bbeta0/ppsi_0)*vttp_k6;
vttp_k9=(1/ppsi_0)*(A'-dd*I)*vttp_k8-(bbeta0/ppsi_0)*vttp_k7;
vttp_k10=(1/ppsi_0)*(A'-dd*I)*vttp_k9-(bbeta0/ppsi_0)*vttp_k8;
vttp_k11=(1/ppsi_0)*(A'-dd*I)*vttp_k10-(bbeta0/ppsi_0)*vttp_k9;
vttp_k12=(1/ppsi_0)*(A'-dd*I)*vttp_k11-(bbeta0/ppsi_0)*vttp_k10;
vttp_k13=(1/ppsi_0)*(A'-dd*I)*vttp_k12-(bbeta0/ppsi_0)*vttp_k11;
vttp_k14=(1/ppsi_0)*(A'-dd*I)*vttp_k13-(bbeta0/ppsi_0)*vttp_k12;
vttp_k15=(1/ppsi_0)*(A'-dd*I)*vttp_k14-(bbeta0/ppsi_0)*vttp_k13;
vttp_k16=(1/ppsi_0)*(A'-dd*I)*vttp_k15-(bbeta0/ppsi_0)*vttp_k14;

vttr_k0=rrt1;
vttr_k1=(1/ppsi_0)*(A'-dd*I)*vttr_k0;
```

```
    vttr_k2=(1/ppsi_0)*(A'-dd*I)*vttr_k1-(bbeta0/ppsi_0)*vttr_k0;
    vttr_k3=(1/ppsi_0)*(A'-dd*I)*vttr_k2-(bbeta0/ppsi_0)*vttr_k1;
    vttr_k4=(1/ppsi_0)*(A'-dd*I)*vttr_k3-(bbeta0/ppsi_0)*vttr_k2;
    vttr_k5=(1/ppsi_0)*(A'-dd*I)*vttr_k4-(bbeta0/ppsi_0)*vttr_k3;
    vttr_k6=(1/ppsi_0)*(A'-dd*I)*vttr_k5-(bbeta0/ppsi_0)*vttr_k4;
    vttr_k7=(1/ppsi_0)*(A'-dd*I)*vttr_k6-(bbeta0/ppsi_0)*vttr_k5;
    vttr_k8=(1/ppsi_0)*(A'-dd*I)*vttr_k7-(bbeta0/ppsi_0)*vttr_k6;
    vttr_k9=(1/ppsi_0)*(A'-dd*I)*vttr_k8-(bbeta0/ppsi_0)*vttr_k7;
    vttr_k10=(1/ppsi_0)*(A'-dd*I)*vttr_k9-(bbeta0/ppsi_0)*vttr_k8;
    vttr_k11=(1/ppsi_0)*(A'-dd*I)*vttr_k10-(bbeta0/ppsi_0)*vttr_k9;
    vttr_k12=(1/ppsi_0)*(A'-dd*I)*vttr_k11-(bbeta0/ppsi_0)*vttr_k10;
    vttr_k13=(1/ppsi_0)*(A'-dd*I)*vttr_k12-(bbeta0/ppsi_0)*vttr_k11;
    vttr_k14=(1/ppsi_0)*(A'-dd*I)*vttr_k13-(bbeta0/ppsi_0)*vttr_k12;
    vttr_k15=(1/ppsi_0)*(A'-dd*I)*vttr_k14-(bbeta0/ppsi_0)*vttr_k13;

    VV_p=[vpp_k0,vpp_k1,vpp_k2,vpp_k3,vpp_k4,vpp_k5,vpp_k6,...
    vpp_k7,vpp_k8,vpp_k9,vpp_k10,vpp_k11,vpp_k12,vpp_k13,...
    vpp_k14,vpp_k15,vpp_k16];
    VV_r=[vrr_k0,vrr_k1,vrr_k2,vrr_k3,vrr_k4,vrr_k5,vrr_k6,...
    vrr_k7,vrr_k8,vrr_k9,vrr_k10,vrr_k11,vrr_k12,vrr_k13,...
    vrr_k14,vrr_k15];
    VVt_p=[vttp_k0,vttp_k1,vttp_k2,vttp_k3, ...
        vttp_k4,vttp_k5,vttp_k6,...
    vttp_k7,vttp_k8,vttp_k9,vttp_k10,vttp_k11,vttp_k12, vttp_k13,...
    vttp_k14,vttp_k15,vttp_k16];
    VVt_r=[vttr_k0,vttr_k1,vttr_k2,vttr_k3,vttr_k4,vttr_k5,vttr_k6,...
    vttr_k7,vttr_k8,vttr_k9,vttr_k10,vttr_k11,vttr_k12,vttr_k13,...
    vttr_k14,vttr_k15];
    VV=[VV_p, VV_r];
    VVt=[VVt_p, VVt_r];
    GG=VVt'*VV;
    kk=kk+1;
    valuer2(:,kk)=r11;
    u2(:,kk)=norm(valuer2(:,kk)/norm(b1));
    if kk==n
          break;
    end

end
s=16
k=1
n=10049
normb=norm(b1);
while (norm(r1)/normb>tol)
    p_k0=[1 zeros(1,32)]';
      r_k0=[zeros(1,17) 1 zeros(1,15)]';
      e_k0=zeros(1,33);
     pt_k0=[1 zeros(1,32)]';
     rt_k0=[zeros(1,17) 1 zeros(1,15)]';
    delt=rt_k0'*G*r_k0;
    for j=0 : s-1
        alpha=delt/(pt_k0'*G*B*p_k0);
        e_k1=e_k0+alpha*p_k0';
        r_k1=r_k0-B*(alpha*p_k0);
        rt_k1=rt_k0-B*(alpha*pt_k0);
        delt2=rt_k1'*G*r_k1;
        beta=delt2/delt;
```

```
        p_k1=r_k1+beta*p_k0;
        pt_k1=rt_k1+beta*pt_k0;
        %updating
        r_k0=r_k1;
        rt_k0=rt_k1;
        p_k0=p_k1;
        pt_k0=pt_k1;
        delt=delt2;
        e_k0=e_k1;
    end
    xm=V*e_k0'+x1;
    rm=V*r_k0;
    pm=V*p_k0;
    rtm=Vt*rt_k0;
    ptm=Vt*pt_k0;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    x1=xm;
    r1=rm;
    p1=pm;
    pt1=ptm;
    rt1=rtm;
    tr1=b1-A*x1;
    vp_k0=p1;
    vp_k1=(1/psi_0)*(A-d*I)*vp_k0;
    vp_k2=(1/psi_1)*(A-d*I)*vp_k1-(beta0/psi_1)*vp_k0;
    vp_k3=(1/psi_1)*(A-d*I)*vp_k2-(beta0/psi_1)*vp_k1;
    vp_k4=(1/psi_1)*(A-d*I)*vp_k3-(beta0/psi_1)*vp_k2;
    vp_k5=(1/psi_1)*(A-d*I)*vp_k4-(beta0/psi_1)*vp_k3;
    vp_k6=(1/psi_1)*(A-d*I)*vp_k5-(beta0/psi_1)*vp_k4;
    vp_k7=(1/psi_1)*(A-d*I)*vp_k6-(beta0/psi_1)*vp_k5;
    vp_k8=(1/psi_1)*(A-d*I)*vp_k7-(beta0/psi_1)*vp_k6;
    vp_k9=(1/psi_1)*(A-d*I)*vp_k8-(beta0/psi_1)*vp_k7;
    vp_k10=(1/psi_1)*(A-d*I)*vp_k9-(beta0/psi_1)*vp_k8;
    vp_k11=(1/psi_1)*(A-d*I)*vp_k10-(beta0/psi_1)*vp_k9;
    vp_k12=(1/psi_1)*(A-d*I)*vp_k11-(beta0/psi_1)*vp_k10;
    vp_k13=(1/psi_1)*(A-d*I)*vp_k12-(beta0/psi_1)*vp_k11;
    vp_k14=(1/psi_1)*(A-d*I)*vp_k13-(beta0/psi_1)*vp_k12;
    vp_k15=(1/psi_1)*(A-d*I)*vp_k14-(beta0/psi_1)*vp_k13;
    vp_k16=(1/psi_1)*(A-d*I)*vp_k15-(beta0/psi_1)*vp_k14;

    vr_k0=r1;
    vr_k1=(1/psi_0)*(A-d*I)*vr_k0;
    vr_k2=(1/psi_1)*(A-d*I)*vr_k1-(beta0/psi_1)*vr_k0;
    vr_k3=(1/psi_1)*(A-d*I)*vr_k2-(beta0/psi_1)*vr_k1;
    vr_k4=(1/psi_1)*(A-d*I)*vr_k3-(beta0/psi_1)*vr_k2;
    vr_k5=(1/psi_1)*(A-d*I)*vr_k4-(beta0/psi_1)*vr_k3;
    vr_k6=(1/psi_1)*(A-d*I)*vr_k5-(beta0/psi_1)*vr_k4;
    vr_k7=(1/psi_1)*(A-d*I)*vr_k6-(beta0/psi_1)*vr_k5;
    vr_k8=(1/psi_1)*(A-d*I)*vr_k7-(beta0/psi_1)*vr_k6;
    vr_k9=(1/psi_1)*(A-d*I)*vr_k8-(beta0/psi_1)*vr_k7;
    vr_k10=(1/psi_1)*(A-d*I)*vr_k9-(beta0/psi_1)*vr_k8;
    vr_k11=(1/psi_1)*(A-d*I)*vr_k10-(beta0/psi_1)*vr_k9;
    vr_k12=(1/psi_1)*(A-d*I)*vr_k11-(beta0/psi_1)*vr_k10;
    vr_k13=(1/psi_1)*(A-d*I)*vr_k12-(beta0/psi_1)*vr_k11;
    vr_k14=(1/psi_1)*(A-d*I)*vr_k13-(beta0/psi_1)*vr_k12;
    vr_k15=(1/psi_1)*(A-d*I)*vr_k14-(beta0/psi_1)*vr_k13;
```

```
    vtp_k0=pt1;
    vtp_k1=(1/psi_0)*(A'-d*I)*vtp_k0;
    vtp_k2=(1/psi_1)*(A'-d*I)*vtp_k1-(beta0/psi_1)*vtp_k0;
    vtp_k3=(1/psi_1)*(A'-d*I)*vtp_k2-(beta0/psi_1)*vtp_k1;
    vtp_k4=(1/psi_1)*(A'-d*I)*vtp_k3-(beta0/psi_1)*vtp_k2;
    vtp_k5=(1/psi_1)*(A'-d*I)*vtp_k4-(beta0/psi_1)*vtp_k3;
    vtp_k6=(1/psi_1)*(A'-d*I)*vtp_k5-(beta0/psi_1)*vtp_k4;
    vtp_k7=(1/psi_1)*(A'-d*I)*vtp_k6-(beta0/psi_1)*vtp_k5;
    vtp_k8=(1/psi_1)*(A'-d*I)*vtp_k7-(beta0/psi_1)*vtp_k6;
    vtp_k9=(1/psi_1)*(A'-d*I)*vtp_k8-(beta0/psi_1)*vtp_k7;
    vtp_k10=(1/psi_1)*(A'-d*I)*vtp_k9-(beta0/psi_1)*vtp_k8;
    vtp_k11=(1/psi_1)*(A'-d*I)*vtp_k10-(beta0/psi_1)*vtp_k9;
    vtp_k12=(1/psi_1)*(A'-d*I)*vtp_k11-(beta0/psi_1)*vtp_k10;
    vtp_k13=(1/psi_1)*(A'-d*I)*vtp_k12-(beta0/psi_1)*vtp_k11;
    vtp_k14=(1/psi_1)*(A'-d*I)*vtp_k13-(beta0/psi_1)*vtp_k12;
    vtp_k15=(1/psi_1)*(A'-d*I)*vtp_k14-(beta0/psi_1)*vtp_k13;
    vtp_k16=(1/psi_1)*(A'-d*I)*vtp_k15-(beta0/psi_1)*vtp_k14;

    vtr_k0=rt1;
    vtr_k1=(1/psi_0)*(A'-d*I)*vtr_k0;
    vtr_k2=(1/psi_1)*(A'-d*I)*vtr_k1-(beta0/psi_1)*vtr_k0;
    vtr_k3=(1/psi_1)*(A'-d*I)*vtr_k2-(beta0/psi_1)*vtr_k1;
    vtr_k4=(1/psi_1)*(A'-d*I)*vtr_k3-(beta0/psi_1)*vtr_k2;
    vtr_k5=(1/psi_1)*(A'-d*I)*vtr_k4-(beta0/psi_1)*vtr_k3;
    vtr_k6=(1/psi_1)*(A'-d*I)*vtr_k5-(beta0/psi_1)*vtr_k4;
    vtr_k7=(1/psi_1)*(A'-d*I)*vtr_k6-(beta0/psi_1)*vtr_k5;
    vtr_k8=(1/psi_1)*(A'-d*I)*vtr_k7-(beta0/psi_1)*vtr_k6;
    vtr_k9=(1/psi_1)*(A'-d*I)*vtr_k8-(beta0/psi_1)*vtr_k7;
    vtr_k10=(1/psi_1)*(A'-d*I)*vtr_k9-(beta0/psi_1)*vtr_k8;
    vtr_k11=(1/psi_1)*(A'-d*I)*vtr_k10-(beta0/psi_1)*vtr_k9;
    vtr_k12=(1/psi_1)*(A'-d*I)*vtr_k11-(beta0/psi_1)*vtr_k10;
    vtr_k13=(1/psi_1)*(A'-d*I)*vtr_k12-(beta0/psi_1)*vtr_k11;
    vtr_k14=(1/psi_1)*(A'-d*I)*vtr_k13-(beta0/psi_1)*vtr_k12;
    vtr_k15=(1/psi_1)*(A'-d*I)*vtr_k14-(beta0/psi_1)*vtr_k13;
    V_p=[vp_k0,vp_k1,vp_k2,vp_k3,vp_k4,vp_k5,vp_k6,vp_k7,vp_k8,...
        vp_k9,vp_k10,vp_k11,vp_k12,vp_k13,vp_k14,vp_k15,vp_k16];
    V_r=[vr_k0,vr_k1,vr_k2,vr_k3,vr_k4,vr_k5,vr_k6,vr_k7,vr_k8,...
        vr_k9,vr_k10,vr_k11,vr_k12,vr_k13,vr_k14,vr_k15];
    Vt_p=[vtp_k0,vtp_k1,vtp_k2,vtp_k3, vtp_k4,vtp_k5,vtp_k6,...
        vtp_k7,vtp_k8,vtp_k9,vtp_k10,vtp_k11,vtp_k12, vtp_k13,...
        vtp_k14,vtp_k15,vtp_k16];
    Vt_r=[vtr_k0,vtr_k1,vtr_k2,vtr_k3,vtr_k4,vtr_k5,vtr_k6,...
        vtr_k7,vtr_k8,vtr_k9,vtr_k10,vtr_k11,vtr_k12,vtr_k13,vtr_k14,vtr_k15];
    V=[V_p, V_r];
    Vt=[Vt_p, Vt_r];
    G=Vt'*V;
    k=k+1;
    valuer1(:,k)=r1;
    u1(:,k)=norm(valuer1(:,k)/norm(b1));
    if k==n
        break
    end

end
%plot for both bases
semilogy(u1,'-o')
```

```
xlabel('Number of Iterations')
ylabel('2-Norm Residual')
hold on
semilogy(u2, '-*')
legend ('Chebyshev Basis s=16 ', 'Monomial Basis s=16')
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 Enigeering and Computer Sciences University of California at Berkeley, 2014.

[5] E. Carson, N. Knight, and J. Demmel. "Avoiding communication in nonsymmetric 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. Siauw, 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/mmiomatlab.html`.

[16] studenti.it website. `https://www.studenti.it/matematica/Proprieta-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/