# Recycling Bi-Lanczos Algorithms: BiCG, CGS, and BiCGSTAB

Kapil Ahuja

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mathematics

Dr. Eric de Sturler, Chair
Dr. Christopher A. Beattie
Dr. Jeffrey T. Borggaard
Dr. Serkan Gugercin

August 13, 2009
Blacksburg, Virginia

Keywords: Krylov subspace recycling, bi-Lanczos method, Petrov-Galerkin formulation.

**Recycling Bi-Lanczos Algorithms: BiCG, CGS, and BiCGSTAB**

Kapil Ahuja

Abstract

Engineering problems frequently require solving a sequence of dual linear systems. This paper introduces recycling BiCG, that recycles the Krylov subspace from one pair of linear systems to the next pair. Augmented bi-Lanczos algorithm and modified two-term recurrence are developed for using the recycle space. Recycle space is built from the approximate invariant subspace corresponding to eigenvalues close to the origin. Recycling approach is extended to the CGS and the BiCGSTAB algorithms. Experiments on a convection-diffusion problem give promising results.

Acknowledgments

I would like to thank Dr. de Sturler for guiding me through this research for the past two years. He gently corrected me at every stage: coming up with the relevant ideas, implementing them efficiently, presenting them with less ambiguity, and writing concisely. I am also grateful to him for supporting me as a research assistant.

I would also like to thank Eun Chang who is my fellow graduate student here at VT. The initial work of this thesis was done as a class project with her. Thanks to Hans-Werner Van Wyk for friendly advice on many issues.

I am thankful to all my committee members, Dr. Beattie, Dr. Borggaard, and Dr. Gugercin, for always being available for help. Especially to Dr. Gugercin for helping me with the model reduction basics. Finally, I am greatly indebt to my family for their constant support. Words can't express my thankfulness to them.

This material is based upon work supported by the National Science Foundation under Grant No. NSF-EAR 0530643.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# Contents

**1 Introduction** **1**

**2 Recycling BiCG: Using a Recycle Space** **8**

2.1 Augmented Bi-Lanczos Algorithm . . . . . 9

2.2 Updating the Solution Using the Recycle Space . . . . . 12

2.3 Iteration Vectors for the Elegant Expression: $x$, $p$, and $r$ . . . . . 15

2.4 Scalars for the Elegant Expression: $\alpha$ and $\beta$ . . . . . 19

**3 Recycling BiCG: Computing a Recycle Space** **22**

3.1 Using Harmonic Ritz Pairs . . . . . 23

3.2 The First Linear System and the First Cycle . . . . . 27

3.3 Building Bi-orthogonality between the $C$'s . . . . . 29

3.4 Simplification of Matrix Blocks . . . . . 30

3.5 Computing Tridiagonal from $\alpha$'s and $\beta$'s . . . . . 35

**4 Recycling CGS** **39**

4.1 Polynomials in RBiCG . . . . . 40

4.2 Polynomials in RCGS . . . . . 41

4.3 Iteration Scalars and Vectors . . . . . 43

**5 Recycling BiCGSTAB** **45**

**6 Implementation and Results** **49**

6.1 Implementation . . . . . 49

6.2 Results . . . . . 51

**7 Conclusion** **56**

**Bibliography** **57**

# List of Figures

3.1 Non-zero pattern of $\Gamma_j$ and $\tilde{\Gamma}_j$. . . . . 25

6.1 Coefficients for the PDE. . . . . 52

6.2 Recycling in BiCG. . . . . 53

6.3 Recycling in CGS and BiCGSTAB. . . . . 54

6.4 Using RBiCG in IRKA. . . . . 55

# Chapter 1

# Introduction

We focus on solving a sequence of dual linear systems defined as follows:

$$A^{(j)}x^{(j)} = b^{(j)}, \quad A^{*(j)}\tilde{x}^{(j)} = \tilde{b}^{(j)}, \tag{1.1}$$

where the matrix $A^{(j)} \in \mathbb{C}^{n\times n}$ and the right hand sides $b^{(j)}, \tilde{b}^{(j)} \in \mathbb{C}^n$ vary with $j$. These systems are large and sparse. Krylov subspace methods for solving such systems outperform direct methods, both in terms of the amount of storage and the computational cost [8]. In many practical applications, the change from one pair of systems to the next is small. An important example arises while using the Iterative Rational Krylov Algorithm (IRKA) [9] for model reduction. While solving a pair of systems we store the Krylov subspace generated, and use a selected part of this space for solving the next pair. This process is called "Krylov subspace recycling", and leads to faster convergence for the new pair of systems. The convergence of Krylov subspace methods for solving a linear system, to a great extent, depends on the spectrum of the matrix. Moreover, the deflation or removal of the eigenvalues close to the origin improves the convergence rate [12]. Eigenvalues can be deflated by including

the corresponding eigenvectors in the Krylov subspace. Therefore, we use the approximate invariant subspace corresponding to the small eigenvalues (in absolute value) as our recycle Krylov subspace.

For solving a single linear system, this idea has been used in the GCROT [4] and the GMRES-DR [12] algorithms. For solving a sequence of linear systems, this idea was first proposed in [13] where it is applied to the GCROT and the GCRO-DR algorithms. The idea is further adapted in the RMINRES [19] algorithm. GCROT as in [13], GCRO-DR, and RMINRES all focus on solving a sequence of single systems and not on a sequence that contains a pair of systems. For a comprehensive discussion of recycling algorithms see [13].

Since the most relevant Krylov subspace method for solving systems of the type (1.1) is the BiConjugate Gradient (BiCG) algorithm [7], we focus on Krylov subspace recycling for BiCG in this thesis. We use RBiCG to indicate the recycling BiCG method. To simplify notation, we drop the superscript $j$ in (1.1). At any particular point in the sequence of systems, we refer to $Ax = b$ as the primary system and $A^*\tilde{x} = \tilde{b}$ as the dual system. Throughout the thesis, $||\cdot||$ refers to the two-norm, $(\cdot, \cdot)$ defines the standard inner product, and $e_i$ is the $i$-th canonical vector. Next, we briefly describe Krylov subspace methods, in particular, BiCG.

For the primary system, let $x_0$ be the initial solution vector with $r_0 = b - Ax_0$ as the corresponding residual. Krylov subspace methods find approximate solutions by projections onto the Krylov subspace associated with $A$ and $r_0$ [18]. The $i$-th solution iterate is given by

$$x_i = x_0 + \varrho_i, \tag{1.2}$$

where $\varrho_i \in K^i(A, r_0) \equiv span\{r_0,\ Ar_0,\ A^2r_0,\ \cdots,\ A^{i-1}r_0\}$ would be defined by some projection. Before computing the solution over the Krylov subspace, we need to find a suitable basis for this subspace. Using $r_0,\ Ar_0, \cdots, A^{i-1}r_0$ as the basis for $K^i(A, r_0)$ is not good

numerically. Successive basis vectors point more and more in the direction of the dominant eigenvector, leading to dependence between them in finite precision arithmetic [18]. The solution is to find an orthogonal basis for the Krylov subspace. For Hermitian matrices, the Lanczos method is used to construct an orthogonal basis using a three-term recurrence [8]. For non-Hermitian matrices, the orthogonal basis is typically generated using the Arnoldi method which is expensive because of the lack of a short-term recurrence. Alternatively, one can generate bi-orthogonal basis, which avoids the cost of the full recurrence by using a pair of three-term recurrences. This method is called the bi-Lanczos method [11]. Next, we derive the bi-Lanczos method. First we initialize the Lanczos vectors as follows:

$$v_1 = r_0/||r_0||, \quad \tilde{v}_1 = \tilde{r}_0/||\tilde{r}_0||. \tag{1.3}$$

The bi-Lanczos method computes the subsequent Lanczos vectors as follows:

$$\begin{aligned} &v_{i+1} \in K^{i+1}(A, r_0) \quad \text{s.t.} \quad v_{i+1} \perp \tilde{v}_1, \ldots, \tilde{v}_i \quad \text{and} \\ &\tilde{v}_{i+1} \in K^{i+1}(A^*, \tilde{r}_0) \quad \text{s.t.} \quad \tilde{v}_{i+1} \perp v_1, \ldots, v_i. \end{aligned} \tag{1.4}$$

We define $V_i = [v_1\ v_2\ \ldots\ v_i]$ and $\tilde{V}_i = [\tilde{v}_1\ \tilde{v}_2\ \ldots\ \tilde{v}_i]$, and use the above orthogonality condition to compute the $(i+1)$-th Lanczos vector for the primary system by $\hat{v}_{i+1} = Av_i - V_i\tau \perp \tilde{V}_i$. This implies

$$\tilde{V}_i^* A v_i - \tilde{V}_i^* V_i \tau = 0. \tag{1.5}$$

Using (1.4), we get the following equation:

$$\mathcal{D}_i \tau = \tilde{V}_i^* A v_i, \tag{1.6}$$

where $\mathcal{D}_i = \tilde{V}_i^* V_i$ is a diagonal matrix. The above construction relies on $(v_i, \tilde{v}_i) \neq 0$. If either $v_i = 0$ or $\tilde{v}_i = 0$, then the bi-Lanczos algorithm has found an invariant subspace, and this is referred to as "regular termination" [8]. If $(v_i, \tilde{v}_i) = 0$ but neither $v_i = 0$ nor $\tilde{v}_i = 0$, then the algorithm breaks down and this is referred to as "serious breakdown" [8]. There exist look-ahead Lanczos methods that avoid this breakdown (for details see [10]). Since the diagonal matrix $\mathcal{D}_i$ is assumed to be nonsingular, the above linear system has a unique solution. Therefore, substituting this unique $\tau$ in $\hat{v}_{i+1}$, gives us our $(i+1)$-th Lanczos vector. The Lanczos vectors obtained are uniquely defined up to a scalar, and there is a degree of freedom in choosing the scaling for them [8, 10, 14]. One choice requires $||v_i|| = 1$ and $||\tilde{v}_i|| = 1$. The other choice is to make $||v_i|| = 1$ and $(v_i, \tilde{v}_i) = 1$. Using the first scaling we get the following:

$$\begin{aligned} v_{i+1} &= \hat{v}_{i+1}/||\hat{v}_{i+1}||, \\ \tilde{v}_{i+1} &= (A^*\tilde{v}_i - \tilde{V}_i\tilde{\tau})/||(A^*\tilde{v}_i - \tilde{V}_i\tilde{\tau})||, \end{aligned} \tag{1.7}$$

where $\tilde{\tau}$ is computed by following the analogous steps as for computing $\tau$. The bi-orthogonality condition (1.4) converts the pair of full recurrences above to a pair of 3-term recurrences (see [14]) such that computation of the $(i+1)$-th Lanczos vector requires only the $i$-th and the $(i-1)$-th Lanczos vector. These pair of 3-term recurrences in the matrix form are called bi-Lanczos relations, and are given as follows:

$$\begin{aligned} AV_i &= V_{i+1}\underline{T}_i = V_i T_i + t_{i+1,i} v_{i+1} e_i^T, \\ A^*\tilde{V}_i &= \tilde{V}_{i+1}\underline{\tilde{T}}_i = \tilde{V}_i \tilde{T}_i + \tilde{t}_{i+1,i} \tilde{v}_{i+1} e_i^T, \end{aligned} \tag{1.8}$$

where $t_{i+1,i}$ is the last element of the last row of the tridiagonal $\underline{T}_i$, and similarly, $\tilde{t}_{i+1,i}$ is the last element of the last row of the tridiagonal $\underline{\tilde{T}}_i$. To summarize, in exact arithmetic,

the bi-Lanczos method computes columns of $V_i$ and $\tilde{V}_i$ such that

$$V_i \perp_b \tilde{V}_i, \tag{1.9}$$

where $\perp_b$ denotes bi-orthogonality and implies that $\tilde{V}_i^* V_i$ is a diagonal matrix. Now that we have a good basis for our two Krylov subspaces $K^i(A, r_0)$ and $K^i(A^*, \tilde{r}_0)$, the next step is to find approximate solutions by projections. Two standard ways of finding an approximate solution over the Krylov subspace include the Ritz-Galerkin approach and the minimum norm residual approach [18]. The CG algorithm is an example of Ritz-Galerkin approach where the residual is orthogonal to the the current Krylov subspace. The GMRES algorithm is an example of minimum norm residual approach and corresponds to solving a least squares problem. Since we don't have an orthogonal basis, we utilize the bi-orthogonality condition to define a cheap projection. This leads to a non-optimal solution, and is an example of the Petrov-Galerkin approach. From (1.2) we know $\varrho_i \in K^i(A, r_0)$. With columns of $V_i$ denoting the basis of $K^i(A, r_0)$, we get $\varrho_i = V_i y_i$. The Petrov-Galerkin condition implies

$$r_i = b - A(x_0 + \varrho_i) = r_0 - AV_i y_i \perp \tilde{V}_i. \tag{1.10}$$

Similarly, for the dual system we get

$$\tilde{r}_i = \tilde{r}_0 - A^* \tilde{V}_i \tilde{y}_i \perp V_i. \tag{1.11}$$

Next, we use the initial Lanczos vector (1.3), the bi-orthogonality condition (1.4), the bi-Lanczos relation (1.8), and the Petrov-Galerkin condition (1.10) to derive the expression for

$y_i$.

$$\begin{aligned}
&\tilde{V}_i^* r_0 - \tilde{V}_i^* A V_i y_i = 0 && \Rightarrow \\
&\tilde{V}_i^* V_i e_1 ||r_0|| - \tilde{V}_i^* V_i T_i y_i - \tilde{V}_i^* v_{i+1} e_i^T t_{i+1,i} = 0 && \Rightarrow \\
&\mathcal{D}_i e_1 ||r_0|| - \mathcal{D}_i T_i y_i = 0 && \Rightarrow \\
&y_i = ||r_0|| T_i^{-1} e_1.
\end{aligned} \tag{1.12}$$

The steps are similar for computing $\tilde{y}_i$. If the $T_i$'s become singular at any iterative step, then we get a breakdown of the second kind [8]. The previous breakdown was in the underlying bi-Lanczos algorithm. Extensive experiments show that BiCG works well, and these breakdowns do not happen often in practise. Next, we do a LDU decomposition of the tridiagonal matrix $T_i$, to obtain a two-term recurrence that has numerical advantages [10]. Note that we can do $T_i = L_i D_i R_i$ because all leading principal submatrices of $T_i$ are nonsingular (since $T_i$ has $T_{i-1}$ as a leading principal submatrix, and at all steps, the tridiagonal is assumed to be nonsingular) [6]. If

$$\begin{aligned}
T_i &= L_i D_i R_i, \\
G_i &= V_i R_i^{-1}, \text{ and} \\
\varphi_i &= ||r_0|| D_i^{-1} L_i^{-1} e_1,
\end{aligned} \tag{1.13}$$

then the two-term recurrence for the primary system solution update is given by [5]

$$x_i = x_{i-1} + \varphi_{i,i} g_i, \tag{1.14}$$

where $g_i$ is the last column of $G_i$ and $\varphi_{i,i}$ is the last entry of the vector $\varphi_i$. It is similar for the dual system. The above two-term recurrence can be made more elegant. The solution $(x)$ is now updated using standard iteration vectors $p$ and $r$, and scalars $\alpha$ and $\beta$. We skip the derivation of this elegant expression, and state the final result in Algorithm 1. The

**Algorithm 1.** *BiCG*
1. Choose initial guesses $x_0$ and $\tilde{x}_0$. Compute $r_0 = b - Ax_0$ and $\tilde{r}_0 = \tilde{b} - A^*\tilde{x}_0$.
2. **if** $(r_0, \tilde{r}_0) = 0$ **then** initialize $\tilde{r}_0$ to a random vector.
3. Set $p_0 = 0$, $\tilde{p}_0 = 0$, and $\beta_0 = 0$. Let $tol$ be the convergence tolerance, and $itn$ be the maximum iterative steps the algorithm can take.
4. **for** $i = 1 \ldots itn$ **do**
◇ $p_i = r_{i-1} + \beta_{i-1} p_{i-1}$.
◇ $\tilde{p}_i = \tilde{r}_{i-1} + \bar{\beta}_{i-1} \tilde{p}_{i-1}$.
◇ $q_i = Ap_i$.
◇ $\tilde{q}_i = A^*\tilde{p}_i$.
◇ $\alpha_i = (\tilde{r}_{i-1}, r_{i-1})/(\tilde{p}_i, q_i)$.
◇ $x_i = x_{i-1} + \alpha_i p_i$.
◇ $\tilde{x}_i = \tilde{x}_{i-1} + \bar{\alpha}_i \tilde{p}_i$.
◇ $r_i = r_{i-1} - \alpha_i q_i$.
◇ $\tilde{r}_i = \tilde{r}_{i-1} - \bar{\alpha}_i \tilde{q}_i$.
◇ **if** $||r_i|| \leq tol$ and $||\tilde{r}_i|| \leq tol$ **then break**.
◇ $\beta_i = (\tilde{r}_i, r_i)/(\tilde{r}_{i-1}, r_{i-1})$.
5. **end for**.

analogous derivation is done for RBiCG in Chapter 2. Algorithm 1 solves both the systems (primary and dual) in the pair of systems. The traditional BiCG algorithm available in the literature solves only the primary system. This is because most of the time one is required to solve the primary system only.

The remainder of this thesis consists of six chapters. In Chapter 2, we derive the RBiCG iteration that uses an existing recycle space. Computing a recycle space cheaply is covered in Chapter 3. In Chapters 4 and 5, we extend the recycling approach to Conjugate Gradient Squared (CGS) [16] and its smoothly converging variant BiCGSTAB [17] respectively. For these two algorithms, we only focus on using the recycle space (generated by BiCG). Computing the recycle space within CGS and BiCGSTAB is future work. We give implementation details and results in Chapter 6. Chapter 7 provides conclusions and discusses future work.

# Chapter 2

# Recycling BiCG: Using a Recycle Space

In this chapter we modify the BiCG algorithm to use a given recycle space. First, we briefly describe the recycling idea used in the GCRO-DR algorithm. This is because we closely follow this idea for our RBiCG. After solving the $j$-th primary system in (1.1), GCRO-DR retains $k$ approximate eigenvectors ($w_i$'s) of A. It then computes matrices $U,\ C \in \mathbb{C}^{n\times k}$ from $W = [w_1, \ldots, w_k]$ such that $AU = C$ and $C^*C = I$ (where $A$ is the matrix for the $(j+1)$-th system). It then adapts the Arnoldi process to compute the orthogonal basis for the Krylov subspace such that each new Krylov vector $v$ is orthogonal to $range(C)$. This produces the Arnoldi relation

$$(I - CC^*)AV_i = V_{i+1}\underline{H}_i,$$

where $\underline{H}_i$ is $(i+1)\times i$ upper Hessenburg matrix. GCRO-DR finds solution over the recycle space $range(U)$ and the new Krylov space generated $range(V_i)$.

For our BiCG too, we let the columns of matrix $U$ define the basis of the primary system

recycle space, and define $C = AU$. $A$ is the matrix from the $(j+1)$-th linear system and $U$ available from the approximate right eigenvectors of the $j$-th system. Similarly, we let columns of matrix $\tilde{U}$ define the basis of the dual system recycle space, and define $\tilde{C} = A^*\tilde{U}$. $\tilde{U}$ is computed from the approximate left eigenvectors of the $j$-th system. As in GCRO-DR, the number of vectors selected for recycling is denoted by $k$ such that $U$, $\tilde{U}$, $C$, and $\tilde{C}$ are $n \times k$ matrices. We define columns of $C$ and $\tilde{C}$ as follows:

$$C = [c_1\ c_2\ \ldots\ c_k], \quad \tilde{C} = [\tilde{c}_1\ \tilde{c}_2\ \ldots\ \tilde{c}_k].$$

$C$ here is no longer an orthogonal matrix. Rather we build bi-orthogonality between $C$ and $\tilde{C}$. An alternative choice is discussed later in the chapter.

The rest of this chapter is divided into four sections. In Section 2.1, we derive augmented bi-Lanczos algorithm that computes basis for our two Krylov subspaces that satisfy an augmented bi-orthogonality condition. The two-term recurrence for the solution update in RBiCG is derived in Section 2.2. In Sections 2.3 and 2.4, this two-term recurrence is formulated into an elegant expression.

## 2.1 Augmented Bi-Lanczos Algorithm

The standard bi-Lanczos algorithm computes columns of $V_i$ and $\tilde{V}_i$ such that, in exact arithmetic, $V_i \perp_b \tilde{V}_i$; see (1.9). Since we recycle spaces $U$ and $\tilde{U}$, the bi-Lanczos algorithm can be modified to compute the columns of $V_i$ and $\tilde{V}_i$ such that either

$$[U\ V_i] \perp_b \left[\tilde{U}\ \tilde{V}_i\right] \tag{2.1}$$

or

$$[C\ V_i] \perp_b \left[\tilde{C}\ \tilde{V}_i\right]. \tag{2.2}$$

We decided to build bi-orthogonality given by (2.2) because this leads to simpler algebra. It also has the advantage that the RBiCG algorithm has a form similar to the standard BiCG algorithm. Next, we derive the recurrences that build the above bi-orthogonality (2.2). We assume

$$C \perp_b \tilde{C}.$$

We show how to enforce this bi-orthogonality in Chapter 3 (where we build the recycle space). As in the standard BiCG algorithm, we assume $v_1$ and $\tilde{v}_1$ are available from the initial residuals $r_0$ and $\tilde{r}_0$. We make this statement more precise later in the chapter. The $(i+1)$-th Lanczos vector for the primary system is computed by $\hat{v}_{i+1} = Av_i - V_i\tau - C\rho \perp \left[\tilde{C}\ \tilde{V}_i\right]$. This implies

$$\begin{bmatrix} \tilde{C}^* \\ \tilde{V}_i^* \end{bmatrix} (Av_i - V_i\tau - C\rho) = 0. \tag{2.3}$$

Using (2.2), we get the following equations:

$$\begin{aligned} \mathcal{D}_c\rho &= \tilde{C}^* A v_i, \\ \mathcal{D}_i\tau &= \tilde{V}_i^* A v_i, \end{aligned} \tag{2.4}$$

where $\mathcal{D}_c = \tilde{C}^*C$ and $\mathcal{D}_i = \tilde{V}_i^*V_i$. As in the standard BiCG algorithm we assume $\mathcal{D}_i$ is nonsingular ("serious breakdown" does not occur). We ensure $\mathcal{D}_c$ is nonsingular in Chapter 3. Substituting $\tau$ and $\rho$ from (2.4) into (2.3) gives the $(i+1)$-Lanczos vector. The type of scaling used for the Lanczos vectors, analogous to (1.7), is discussed later in this chapter. As in the standard bi-Lanczos algorithm, the bi-orthogonality condition (2.2) converts the full

recurrence of $\hat{v}_{i+1}$ to a $(3+k)$-term recurrences where $k$ is the number of columns of $C$. This implies the computation of the $(i+1)$-th Lanczos vector require the $i$-th and $(i-1)$-th Lanczos vectors and $k$ columns of $C$. Similarly, we get a $(3+k)$ term recurrence for computing the Lanczos vectors for the dual system. These pair of $(3+k)$-term recurrences in the matrix form are termed as augmented bi-Lanczos relations. The augmented bi-Lanczos relation for the primary system is given as follows:

$$(I - C\hat{C}^*)AV_i = V_{i+1}\underline{T}_i, \tag{2.5}$$

where $\hat{C} = \left[\frac{\tilde{c}_1}{c_1^*\tilde{c}_1} \ \frac{\tilde{c}_2}{c_2^*\tilde{c}_2} \ \cdots \ \frac{\tilde{c}_k}{c_k^*\tilde{c}_k}\right]$ and $\underline{T}_i$ is the $(i+1)\times i$ tridiagonal matrix. Similarly, we get the augmented bi-Lanczos relation for the dual system as follows:

$$(I - \tilde{C}\check{C}^*)A^*\tilde{V}_i = \tilde{V}_{i+1}\underline{\tilde{T}}_i, \tag{2.6}$$

where $\check{C} = \left[\frac{c_1}{\tilde{c}_1^*c_1} \ \frac{c_2}{\tilde{c}_2^*c_2} \ \cdots \ \frac{c_k}{\tilde{c}_k^*c_k}\right]$. One drawback of the current form of the augmented bi-Lanczos relations, (2.5) and (2.6), is that the two matrices, $(I-C\hat{C}^*)A$ and $(I-\tilde{C}\check{C}^*)A^*$, are not conjugate transpose of each other as in the standard BiCG algorithm. This complicates the algebra while deriving the RBiCG solution update recurrence. This problem is solved if we use (2.2) and write the the two relations in the following form:

$$A_1V_i = V_{i+1}\underline{T}_i, \quad \text{where} \quad A_1 = (I - C\mathcal{D}_c^{-1}\tilde{C}^*)A(I - C\mathcal{D}_c^{-1}\tilde{C}^*), \tag{2.7}$$

$$A_1^*\tilde{V}_i = \tilde{V}_{i+1}\underline{\tilde{T}}_i, \quad \text{where} \quad A_1^* = (I - \tilde{C}\mathcal{D}_c^{-*}C^*)A^*(I - \tilde{C}\mathcal{D}_c^{-*}C^*), \tag{2.8}$$

with $\tilde{C}^*C = \mathcal{D}_c$. $\mathcal{D}_c$ is a diagonal matrix and so $\mathcal{D}_c^* = \bar{\mathcal{D}}_c$. The next step is to obtain the solution update recurrence that utilizes the recycle space. This is described in the next section.

## 2.2 Updating the Solution Using the Recycle Space

In the standard BiCG algorithm, the $i$-th solution iterate is given by (1.2). That is,

$$x_i = x_0 + V_i y_i, \quad \tilde{x}_i = \tilde{x}_0 + \tilde{V}_i \tilde{y}_i.$$

The $i$-th solution iterates in the RBiCG algorithm, as in the GCRO-DR and the RMINRES algorithms, are given as follows:

$$x_i = x_0 + U z_i + V_i y_i, \quad \tilde{x}_i = \tilde{x}_0 + \tilde{U} \tilde{z}_i + \tilde{V}_i \tilde{y}_i. \tag{2.9}$$

In the standard BiCG algorithm, $y_i$ and $\tilde{y}_i$ are computed by (1.10) and (1.11). That is, by enforcing the following orthogonality conditions (Petrov-Galerkin approximation):

$$r_i = r_0 - A V_i y_i \perp \tilde{V}_i, \quad \tilde{r}_i = \tilde{r}_0 - A^* \tilde{V}_i \tilde{y}_i \perp V_i.$$

With recycling, the Petrov-Galerkin approximation gives

$$r_i = r_0 - AUz_i - AV_i y_i \perp \left[\tilde{C}\ \tilde{V}_i\right], \quad \tilde{r}_i = \tilde{r}_0 - A^* \tilde{U} \tilde{z}_i - A^* \tilde{V}_i \tilde{y}_i \perp [C\ V_i]. \tag{2.10}$$

For the primary system, the above orthogonality condition leads to the following:

$$\begin{bmatrix} \tilde{C}^* \\ \tilde{V}_i^* \end{bmatrix} \left[ r_0 \ - \ A[U\ V_i] \begin{bmatrix} z_i \\ y_i \end{bmatrix} \right] = 0. \tag{2.11}$$

The steps are similar for the dual system. The actual computation is somewhat more efficient than the above equation. Lets first look at $r_0$. Defining $\zeta = ||(I - C\hat{C}^*)r_0||$ we get

$$\begin{aligned} r_0 &= C\hat{C}^* r_0 + r_0 - C\hat{C}^* r_0, \\ &= C\hat{C}^* r_0 + \zeta v_1, \qquad \text{taking } v_1 = \tfrac{(I-C\hat{C}^*)r_0}{\zeta}, \\ &= C\hat{C}^* r_0 + \zeta V_{i+1} e_1, \\ &= [C\ V_{i+1}] \begin{bmatrix} \hat{C}^* r_0 \\ \zeta e_1 \end{bmatrix}. \end{aligned} \tag{2.12}$$

It is here that we define our first Lanczos vector $v_1$. Note that $v_1$ is obtained from the initial residual $r_0$. Now let's look at the second term of (2.11). For the following derivation, augmented bi-Lanczos relation (2.5) is used.

$$\begin{aligned} A\,[U \quad V_i] \begin{bmatrix} z_i \\ y_i \end{bmatrix} &= [AU \quad AV_i] \begin{bmatrix} z_i \\ y_i \end{bmatrix}, \\ &= \begin{bmatrix} C & C\hat{C}^* AV_i + V_{i+1}\underline{T}_i \end{bmatrix} \begin{bmatrix} z_i \\ y_i \end{bmatrix}, \\ &= [C \quad V_{i+1}] \begin{bmatrix} I & \hat{C}^* AV_i \\ 0 & \underline{T}_i \end{bmatrix} \begin{bmatrix} z_i \\ y_i \end{bmatrix}. \end{aligned} \tag{2.13}$$

Substituting (2.12) and (2.13) in (2.11) gives

$$\begin{bmatrix} \tilde{C}^* \\ \tilde{V}_i^* \end{bmatrix} [C\ V_{i+1}] \left[ \begin{bmatrix} \hat{C}^* r_0 \\ \zeta e_1 \end{bmatrix} - \begin{bmatrix} I & \hat{C}^* AV_i \\ 0 & \underline{T}_i \end{bmatrix} \begin{bmatrix} z_i \\ y_i \end{bmatrix} \right] = 0. \tag{2.14}$$

Using the bi-orthogonality condition (2.2) in the above equation we get[1]

$$\begin{bmatrix} \hat{C}^* r_0 \\ \zeta e_1 \end{bmatrix} - \begin{bmatrix} I & \hat{C}^* A V_i \\ 0 & T_i \end{bmatrix} \begin{bmatrix} z_i \\ y_i \end{bmatrix} = 0. \tag{2.15}$$

Therefore, the equations for finding $y_i$ and $z_i$ are given by

$$\begin{aligned} y_i &= \zeta T_i^{-1} e_1, \\ z_i &= \hat{C}^* r_0 - \hat{C}^* A V_i y_i. \end{aligned}$$

As in the standard BiCG algorithm, the algorithm here fails if a singular tridiagonal is encountered [8] (breakdown of the second kind discussed in Chapter 1). For clarity of exposition we assume this does not occur. Substituting the above expressions in (2.9) leads to the following solution update for the primary system:

$$x_i = \chi_0 + (I - U\hat{C}^* A) V_i y_i, \tag{2.16}$$

where $\chi_0 = x_0 + U\hat{C}^* r_0$ and $y_i = \zeta T_i^{-1} e_1$. Next, as in the standard BiCG algorithm, we compute LDU decomposition of the tridiagonal matrix $T_i$ to get a two-term recurrence. If

$$\begin{aligned} T_i &= L_i D_i R_i, \\ G_i &= (I - U\hat{C}^* A) V_i R_i^{-1}, \text{ and} \\ \varphi_i &= \zeta D_i^{-1} L_i^{-1} e_1, \end{aligned} \tag{2.17}$$

then the two-term recurrence for the solution update of the primary system is given by

$$x_i = x_{i-1} + \varphi_{i,i} g_i, \tag{2.18}$$

[1]Note that the dimension of $e_1$ in (2.15) is one less than that of $e_1$ in (2.14), although both denote the first canonical vector.

where $g_i$ is the last column of $G_i$ and $\varphi_{i,i}$ is the last entry of the vector $\varphi_i$. It is similar for the dual system. We now do a slight change of notation to make the future derivations simpler. Let $x_{-1}$ be the initial guess and $r_{-1} = b - Ax_{-1}$ the corresponding initial residual. We define

$$x_0 = x_{-1} + U\hat{C}^* r_{-1} \text{ and } r_0 = (I - C\hat{C}^*)r_{-1}. \tag{2.19}$$

Similarly, for the dual system, let $\tilde{x}_{-1}$ be the initial guess and $\tilde{r}_{-1} = \tilde{b} - A^*\tilde{x}_{-1}$ the corresponding initial residual. We define

$$\tilde{x}_0 = \tilde{x}_{-1} + \tilde{U}\check{C}^*\tilde{r}_{-1} \text{ and } \tilde{r}_0 = (I - \tilde{C}\check{C}^*)\tilde{r}_{-1}. \tag{2.20}$$

$x_0$ in (2.18) is defined by (2.19). We follow this convention for $x_0$, $\tilde{x}_0$, $r_0$, and $\tilde{r}_0$ for the rest of the thesis. Note that in (2.17) we never compute any explicit matrix inverse. The matrices under consideration are either diagonal, or lower triangular, or upper triangular. Moreover, we can obtain the required information even without generating the tridiagonal matrix explicitly. In the next section we do simplifications to this solution update recurrence leading to a form similar to the standard BiCG algorithm. That is, we derive the analogous version of the standard iteration vectors: $x$ (solution), $p$, and $r$ (residual). The corresponding scalars, $\alpha$ and $\beta$, are derived in Section 2.4.

## 2.3 Iteration Vectors for the Elegant Expression: $x$, $p$, and $r$

We first develop the recurrences for $x$, followed by recurrences for $p$, and finally for $r$. The main idea is to simplify the notation, and use the orthogonality condition of the bi-Lanczos

method. For the $x$ recurrences, we write (2.18) in the following form:

$$x_i = x_{i-1} + \alpha_i Z p_i, \tag{2.21}$$

where $\alpha_i = \frac{\varphi_{i,i}}{||r_{i-1}||}$, $Zp_i = ||r_{i-1}||g_i$, and $Z = (I - U\hat{C}^* A)$. For the dual system, solution update is given by

$$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{Z} \tilde{p}_i, \tag{2.22}$$

where $\tilde{Z} = (I - \tilde{U}\check{C}^* A^*)$. If the recycle space does not exist, then the above solution update recurrences are the same as in the standard BiCG (see Algorithm 1).

Next, we find the relationship between the residual and the Lanczos vectors. This relationship is described by Theorem 2.1 and is used in computing the $p$ vector. Theorem 2.1 also indicates how to compute the Lanczos vectors that are used in building of the recycle space (Chapter 3).

**Theorem 2.1.** *Prove that $r_i = \nu_i v_{i+1}$ and $\tilde{r}_i = \tilde{\nu}_i \tilde{v}_{i+1}$, where $\nu_i$ and $\tilde{\nu}_i$ are scalars.*

*Proof.* Using augmented bi-Lanczos relation (2.5), the first Lanczos vector (2.12), the solution update expression (2.16), and the initial residual (2.19), we get

$$\begin{aligned}
r_i &= b - Ax_i, \\
&= r_0 - A(I - U\hat{C}^* A)V_i y_i, \\
&= r_0 - V_{i+1}\underline{T}_i y_i, \\
&= r_0 - V_{i+1}\underline{T}_i T_i^{-1} \zeta e_1, \\
&= r_0 - \zeta v_1 - (t_{i+1}^T T_i^{-1} \zeta e_1) v_{i+1}, \text{ where } t_{i+1} \text{ is the last column of } \underline{T}_i^T, \\
&= \nu_i v_{i+1},
\end{aligned}$$

where $\nu_i$ is a scalar. We can prove $\tilde{r}_i = \tilde{\nu}_i \tilde{v}_{i+1}$ in a similar fashion. □

As discussed in Chapter 1, there is a degree of freedom in choosing the scaling of the Lanczos vectors $v_i$ and $\tilde{v}_i$ [8, 10, 14]. The following scaling ensures that the computation of the recycle space stays cheap:

$$||v_i|| = 1, \qquad (v_i, \tilde{v}_i) = 1. \tag{2.23}$$

This is discussed more in Chapter 3. Hence, the Lanczos vectors are computed as follows:

$$v_i = \frac{r_{i-1}}{||r_{i-1}||}, \qquad \tilde{v}_i = \frac{\tilde{r}_{i-1}}{(v_i, \tilde{r}_{i-1})}. \tag{2.24}$$

Now we find the recurrence for the $p$ vector. From the LDU decomposition in (2.17), we know $G_i = ZV_iR_i^{-1}$, where, as before, $Z = (I - U\hat{C}^*A)$. Taking $ZV_i = F_i$ we get the following:

$$F_i = G_iR_i \Rightarrow$$

$$\begin{bmatrix} f_1 & f_2 & \cdots & f_i \end{bmatrix} = \begin{bmatrix} g_1 & g_2 & \cdots & g_i \end{bmatrix} \begin{bmatrix} 1 & \epsilon_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \epsilon_{i-1} \\ & & & 1 \end{bmatrix}.$$

$R_i$ above is bidiagonal because it is the upper triangular matrix from the LDU decomposition of a tridiagonal matrix. By using the $x$ recurrences, (2.21) and (2.22), and Theorem 2.1 we get

$$\begin{aligned} g_i &= f_i - \epsilon_{i-1}g_{i-1} && \Rightarrow \\ g_i &= Zv_i - \epsilon_{i-1}g_{i-1} && \Rightarrow \\ \tfrac{Zp_i}{||r_{i-1}||} &= \tfrac{Zr_{i-1}}{||r_{i-1}||} - \epsilon_{i-1}\tfrac{Zp_{i-1}}{||r_{i-2}||} && \Rightarrow \\ p_i &= r_{i-1} - \tfrac{\epsilon_{i-1}||r_{i-1}||}{||r_{i-2}||}p_{i-1}. \end{aligned}$$

In the above derivation, we assume $Z$ is nonsingular. Taking $\beta_{i-1} = -\frac{\epsilon_{i-1}||r_{i-1}||}{||r_{i-2}||}$, the $p$

recurrences are given by the following equations:

$$p_i = r_{i-1} + \beta_{i-1} p_{i-1}, \tag{2.25}$$

$$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_{i-1} \tilde{p}_{i-1}. \tag{2.26}$$

The above $p$ recurrences are the same as in BiCG (see Algorithm 1). These recurrences need the residual recurrences. The residual recurrences are obtained from the recurrences for the solution update i.e. (2.21) and (2.22), and are given as follows:

$$r_i = r_{i-1} - \alpha_i (I - C\hat{C}^*) A p_i, \tag{2.27}$$

$$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i (I - \tilde{C}\check{C}^*) A^* \tilde{p}_i. \tag{2.28}$$

If these recurrences are used for further derivations, then the expressions for the iteration scalars $\alpha$ and $\beta$ are not as simplified as in the standard BiCG algorithm. However, if the augmented bi-Lanczos relations of the form in (2.7) and (2.8) are used, then we get simpler algebraic relations. Using these different form of augmented bi-Lanczos relations, we get the residual recurrences as follows:

$$r_i = r_{i-1} - \alpha_i A_1 p_i, \tag{2.29}$$

$$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i A_1^* \tilde{p}_i. \tag{2.30}$$

If we replace $A_1$ by $A$ above, then these recurrences are the same as in the standard BiCG algorithm (see Algorithm 1). For theoretical purpose, these new recurrences are ideal. While for implementation, we still use the old residual recurrences i.e. (2.27) and (2.28). This is because, working with $A_1$ and $A_1^*$ is more expensive than working with $(I - C\hat{C}^*)A$ and $(I - \tilde{C}\check{C}^*)A^*$. With iterations vectors at hand, we next find expressions for $\alpha$'s and $\beta$'s.

## 2.4 Scalars for the Elegant Expression: $\alpha$ and $\beta$

First we prove a result that is central to CG and its derived algorithms. In CG, this result states that the $p$ vectors are A-orthogonal. For BiCG, the equivalent result is that $p$ and $\tilde{p}$ are A-orthogonal. Theorem 2.3 below, gives the analogous result for RBiCG. Theorem 2.3 uses a result that is derived in Theorem 2.2 below.

**Theorem 2.2.** *If $T_i = L_i D_i R_i$ and $\tilde{T}_i = \tilde{L}_i \tilde{D}_i \tilde{R}_i$, then show that $R_i^* = \tilde{L}_i$ and $L_i^* = \tilde{R}_i$.*

*Proof.* We know from (2.8) that the augmented bi-Lanczos relation for the dual system is given by

$$\begin{aligned} A_1^* \tilde{V}_i &= \tilde{V}_{i+1} \underline{\tilde{T}}_i \Rightarrow \\ V_i^* A_1^* \tilde{V}_i &= \tilde{T}_i, \text{ since } V_i^* \tilde{V}_i = I \text{ from the choice of scaling (2.23).} \end{aligned}$$

Similarly, for the primary system, we get the following:

$$\begin{aligned} A_1 V_i &= V_{i+1} \underline{T}_i \Rightarrow \\ V_i^* A_1^* \tilde{V}_i &= T_i^*, \text{ again using (2.23).} \end{aligned}$$

Equating the two $V_i^* A_1^* \tilde{V}_i$, we get that $T_i^* = \tilde{T}_i$. LDU factorization of the two tridiagonals implies $R_i^* D_i^* L_i^* = \tilde{L}_i \tilde{D}_i \tilde{R}_i$. As the LDU factorization is unique, we get that $R_i^* = \tilde{L}_i$ and $L_i^* = \tilde{R}_i$. □

Next, we define $P_i = [p_1\ p_2\ \ldots\ p_i]$, and find a relation between the matrices $P_i$ and $V_i$. From

the LDU decomposition in (2.17) and the $x$ recurrence (2.21) we know

$$\begin{aligned} & G_i R_i = Z V_i, \text{ with } Z = (I - U\hat{C}^* A) \quad \Rightarrow \\ & Z[\tfrac{p_1}{||r_0||} \ \tfrac{p_2}{||r_1||} \ \cdots \ \tfrac{p_i}{||r_{i-1}||}] R_i = Z V_i \qquad \Rightarrow \\ & P_i = V_i R_i^{-1} Y_i, \end{aligned} \tag{2.31}$$

where $Y_i = diag(||r_0||, ||r_1||, \ldots, ||r_{i-1}||)$. As earlier, we assume $Z$ is nonsingular. Similarly, for the dual system, this relation is defined by the following equation:

$$\tilde{P}_i = \tilde{V}_i \tilde{R}_i^{-1} \tilde{Y}_i, \tag{2.32}$$

where $\tilde{Y}_i = diag((v_1, ||\tilde{r}_0||), (v_2, ||\tilde{r}_1||), \ldots, (v_i, ||\tilde{r}_{i-1}||))$. Now the main A-conjugation result:

**Theorem 2.3.** *Show that* $\tilde{p}_i^* A_1 p_j = 0$ *for* $i \neq j$.

*Proof.* We equivalently prove that $\tilde{P}_i^* A_1 P_i$ is a diagonal matrix. The proof below uses Theorem 2.2, (2.31) and (2.32).

$$\begin{aligned} \tilde{P}_i^* A_1 P_i &= \tilde{Y}_i^* \tilde{R}_i^{-*} \tilde{V}_i^* A_1 V_i R_i^{-1} Y_i, \\ &= \tilde{Y}_i^* \tilde{R}_i^{-*} T_i R_i^{-1} Y_i, \\ &= \tilde{Y}_i^* \tilde{R}_i^{-*} L_i D_i R_i R_i^{-1} Y_i, \\ &= \tilde{Y}_i^* \tilde{R}_i^{-*} \tilde{R}_i^* D_i Y_i, \\ &= \tilde{Y}_i^* D_i Y_i. \end{aligned}$$

$\tilde{Y}_i^* D_i Y_i$ is a product of diagonal matrices, and hence, is a diagonal matrix. □

With the entire framework ready, we now find expressions for $\alpha$'s and $\beta$'s. First we show how to compute $\alpha$. For this we use the bi-orthogonality condition (2.2), Theorems 2.1 and 2.3, the recurrences for the $p$ vectors (2.25)-(2.26), and the recurrences for the residuals

(2.29)-(2.30). That is,

$$\begin{aligned}
&\tilde{v}_i^* v_{i+1} = 0 && \Rightarrow \\
&\tilde{r}_{i-1}^* r_i = 0 && \Rightarrow \\
&\tilde{r}_{i-1}^* r_{i-1} - \alpha_i \tilde{r}_{i-1}^* A_1 p_i = 0 && \Rightarrow \\
&\tilde{r}_{i-1}^* r_{i-1} - \alpha_i (\tilde{p}_i - \tilde{\beta}_{i-1}\tilde{p}_{i-1})^* A_1 p_i = 0 && \Rightarrow \\
&\tilde{r}_{i-1}^* r_{i-1} - \alpha_i \tilde{p}_i^* A_1 p_i = 0.
\end{aligned}$$

Thus, the expressions for the $\alpha$'s are given as follows:

$$\alpha_i = \frac{\tilde{r}_{i-1}^* r_{i-1}}{\tilde{p}_i^* A_1 p_i}, \text{ and } \tilde{\alpha}_i = \bar{\alpha}_i. \tag{2.33}$$

As discussed before for the residual recurrence, for theoretical purpose, (2.33) is ideal. For implementation, following version of (2.33) is better suited:

$$\alpha_i = \frac{\tilde{r}_{i-1}^* r_{i-1}}{\tilde{p}_i^* (I - C\hat{C}^*) A p_i}. \tag{2.34}$$

Similarly, $\beta$'s are given by the following equation:

$$\beta_i = \frac{\tilde{r}_i^* r_i}{\tilde{r}_{i-1}^* r_{i-1}}, \text{ and } \tilde{\beta}_i = \bar{\beta}_i. \tag{2.35}$$

# Chapter 3

# Recycling BiCG: Computing a Recycle Space

In this chapter we are concerned with how to build $U$ and $\tilde{U}$. As mentioned in Chapter 2, the columns of $U$ define the basis of the primary system recycle space, and the columns of $\tilde{U}$ define the basis of the dual system recycle space. In GCRO-DR [13] and RMINRES [19], approximate invariant subspaces have been successfully used to build the recycle space. It has been demonstrated there (in [13] and [19]) and in [12] that using the approximate invariant subspace corresponding to small eigenvalues (in absolute value) is effective. For RBiCG we follow the same strategy. For a detailed discussion on the choices for computing a recycle space see [13].

Harmonic Ritz vectors provide an approximate invariant subspace cheaply, and are defined using the theorem below (see Theorem 5.1 from [15]).

**Theorem 3.1.** *Let $S$ be a subspace of $\mathbb{C}^n$. A value $\lambda \in \mathbb{C}$ is a harmonic Ritz value of $A$*

*with respect to the subspace* $\mathcal{W} = AS$ *if and only if*

$$(Au - \lambda u) \perp AS \text{ for some } u \in S,\ u \neq 0.$$

In Theorem 3.1, $u$ is called the harmonic Ritz vector associated with the harmonic Ritz value $\lambda$. Typically, we compute the harmonic Ritz values (and vectors) with respect to the current Krylov subspace [13, 19]. Working directly with the above orthogonality condition is hard. We first convert this orthogonality condition into a generalized eigenvalue problem in Section 3.1. This eigenvalue problem is of much smaller dimension than the dimension of matrix $A$. The first system in our sequence of linear systems requires special attention since there is no recycle space available at the start. This is covered in Section 3.2. After obtaining the $U$ and $\tilde{U}$, we build $C$ and $\tilde{C}$ that need not satisfy the bi-orthogonality condition (2.2). We explicitly build bi-orthogonality between them in Section 3.3. Although, the generalized eigenvalue problem developed in Section 3.1 is of smaller dimension, it is still expensive to solve. The bi-orthogonality condition (2.2) and the choice of scaling (2.23) can be exploited to reduce the cost further. This is covered in Section 3.4. The generalized eigenvalue problem of Section 3.1 requires the tridiagonals of the bi-Lanczos method explicitly. These tridiagonals, as in the CG algorithm [14], can be built inexpensively using $\alpha$ and $\beta$ derived in the previous chapter. Section 3.5 gives such a derivation.

## 3.1 Using Harmonic Ritz Pairs

Until now in the discussion, we did not need the Lanczos vectors $v_i$ and $\tilde{v}_i$ explicitly. However, these Lanczos vectors are needed to build the recycle space. Instead of using all the Lanczos vectors to build the recycle space, we update the recycle space periodically to keep the

memory requirements modest [19]. The iteration process between two updates of the recycle space is referred to as a "cycle". The length of the cycle "s" refers to the number of iterations between updating the recycle space. Let $V_j$ and $\tilde{V}_j$ contain the Lanczos vectors generated during the $j^{th}$ cycle. That is,

$$V_j = \begin{bmatrix} v_{(j-1)s+1} & \dots & v_{js} \end{bmatrix} \quad \text{and} \quad \tilde{V}_j = \begin{bmatrix} \tilde{v}_{(j-1)s+1} & \dots & \tilde{v}_{js} \end{bmatrix}.$$

Let $\Upsilon_j$ and $\tilde{\Upsilon}_j$ be $V_j$ and $\tilde{V}_j$ respectively with one previous and one subsequent Lanczos vector. That is,

$$\Upsilon_j = \begin{bmatrix} v_{(j-1)s} & \dots & v_{js+1} \end{bmatrix} \quad \text{and} \quad \tilde{\Upsilon}_j = \begin{bmatrix} \tilde{v}_{(j-1)s} & \dots & \tilde{v}_{js+1} \end{bmatrix}.$$

The augmented bi-Lanczos relations for the $j^{th}$ cycle are now given by

$$\begin{aligned} (I - C\hat{C}^*)AV_j &= \Upsilon_j \Gamma_j \quad \text{and} \\ (I - \tilde{C}\check{C}^*)A^*\tilde{V}_j &= \tilde{\Upsilon}_j \, \tilde{\Gamma}_j, \end{aligned} \tag{3.1}$$

where $\Gamma_j$ and $\tilde{\Gamma}_j$ are the tridiagonals from (2.5) and (2.6) with an extra row at the top (corresponding to $v_{(j-1)s}$ and $\tilde{v}_{(j-1)s}$) and at the bottom (corresponding to $v_{js+1}$ and $\tilde{v}_{js+1}$). To be precise, the new tridiagonals have the non-zero pattern shown in Figure (3.1).

We now discuss how to build the recycle space. The discussion in this paragraph concerns only the primary system ($Ax = b$). Similar results hold for the dual system ($A^*\tilde{x} = \tilde{b}$). Let columns of $U_{j-1}$ denote the recycle space generated at the end of the $(j-1)$ cycle for the current linear system, and columns of $U$ be the recycle space available from the previous linear system. We want to obtain $U_j$ from $V_j$, $U_{j-1}$, and $U$. It is important to note that $U_j$ is not used for solving the current linear system (for that $U$ is used). The final $U_j$ here (at the end of solving the current linear system) would be the $U$ matrix for the next linear system.

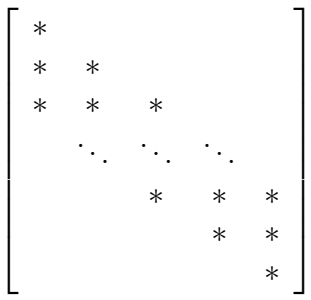

Figure 3.1: Non-zero pattern of $\Gamma_j$ and $\tilde{\Gamma}_j$.

There are many options for selecting $U_j$ [19]. For RBiCG we build $U_j$ from $range([U_{j-1}\ V_j])$ because it keeps the algebra simple. Let

$$\Phi_j = [U_{j-1}\ V_j]\,,\ \Psi_j = [C\ C_{j-1}\ \Upsilon_j]\,, \text{ and } H_j = \begin{bmatrix} 0 & B_j \\ I & 0 \\ 0 & \Gamma_j \end{bmatrix},$$

where $B_j = \hat{C}^* A V_j$. The augmented bi-Lanczos relation for the primary system (3.1) leads to

$$A\Phi_j = \Psi_j H_j. \tag{3.2}$$

Similarly, let

$$\tilde{\Phi}_j = \left[\tilde{U}_{j-1}\ \tilde{V}_j\right],\ \tilde{\Psi}_j = \left[\tilde{C}\ \tilde{C}_{j-1}\ \tilde{\Upsilon}_j\right], \text{ and } \tilde{H}_j = \begin{bmatrix} 0 & \tilde{B}_j \\ I & 0 \\ 0 & \tilde{\Gamma}_j \end{bmatrix},$$

where $\tilde{B}_j = \check{C}^* A^* \tilde{V}_j$. The augmented bi-Lanczos relation for the dual system (3.1) gives

$$A^*\tilde{\Phi}_j = \tilde{\Psi}_j \tilde{H}_j. \tag{3.3}$$

In RMINRES [19] harmonic Ritz pairs of $A$ with respect to the subspace $range(A\Phi_j)$ have been successfully used to build the recycle space. Since here we work in a Petrov-Galerkin framework, using harmonic Ritz pairs with respect to the subspace $range(A^*\tilde{\Phi}_j)$ is more intuitive. This leads to simpler algebra and cheaper computations later. Let $(\lambda, u)$ denote the harmonic Ritz pair of $A$ we are interested in. Then, using Theorem 3.1 and the above discussion, $\lambda$ and $u$ are defined by the condition

$$(Au - \lambda u) \perp range\left(A^*\tilde{\Phi}_j\right), \tag{3.4}$$

where $u \in range(\Phi_j)$ (since we build $U_j$ from $range([U_{j-1}\ V_j])$). Using $u = \Phi_j w$ and (3.2) - (3.3) in (3.4) gives

$$\begin{aligned} \left(A^*\tilde{\Phi}_j\right)^* A\Phi_j w &= \lambda \left(A^*\tilde{\Phi}_j\right)^* \Phi_j w \quad \Rightarrow \\ \left(\tilde{\Psi}_j\tilde{H}_j\right)^* \Psi_j H_j w &= \lambda \left(\tilde{\Psi}_j\tilde{H}_j\right)^* \Phi_j w. \end{aligned}$$

Thus, condition (3.4) is equivalent to the following generalized eigenvalue problem:

$$\tilde{H}_j^*\tilde{\Psi}_j^*\Psi_j H_j w = \lambda \tilde{H}_j^*\tilde{\Psi}_j^*\Phi_j w. \tag{3.5}$$

Let the $k$ chosen right eigenvectors of the above generalized eigenvalue problem form the columns of matrix $W_j$. As discussed at the start of this chapter, we choose eigenvectors corresponding to the eigenvalues close to the origin because deflating these eigenvalues improves the convergence rate [12, 13, 19]. Thus, $U_j = \Phi_j W_j$ (since $u = \Phi_j w$). Let columns of $\tilde{W}_j$ denote the $k$ chosen left eigenvectors of the generalized eigenvalue problem, then $\tilde{U}_j = \tilde{\Phi}_j\tilde{W}_j$. The following derivation confirms this. Let $(\mu, \tilde{u})$ denote the harmonic Ritz pair of $A^*$ we

need. These harmonic Ritz pair are given by

$$(A^*\tilde{u} - \mu\tilde{u}) \perp range\,(A\Phi_j)\,, \tag{3.6}$$

where $\tilde{u} \in range(\tilde{\Phi}_j)$. Using $\tilde{u} = \tilde{\Phi}_j\tilde{w}$ and (3.2) - (3.3) in the above condition leads to the following:

$$\begin{aligned} (A\Phi_j)^* A^*\tilde{\Phi}_j\tilde{w} &= \mu\,(A\Phi_j)^*\,\tilde{\Phi}_j\tilde{w} \quad \Rightarrow \\ (\Psi_j H_j)^*\,\tilde{\Psi}_j\tilde{H}_j\tilde{w} &= \mu\Phi_j^*\left(\tilde{\Psi}_j\tilde{H}_j\right)\tilde{w}. \end{aligned}$$

The generalized eigenvalue problem that is equivalent to condition (3.6) is given by

$$\tilde{w}^*\tilde{H}_j^*\tilde{\Psi}_j^*\Psi_j H_j = \bar{\mu}\tilde{w}^*\tilde{H}_j^*\tilde{\Psi}_j^*\Phi_j. \tag{3.7}$$

Hence, (3.5) and (3.7) are the same except that the former computes the right eigenvectors, while the later computes the left eigenvectors (note that $\lambda = \bar{\mu}$ since the eigenvalues of $A^*$ are the complex conjugates of the eigenvalues of $A$). The first linear system and the first cycle of all linear systems needs special attention since either $U$ (and $\tilde{U}$) or $U_{j-1}$ (and $\tilde{U}_{j-1}$) or both are unavailable. We discuss these cases in the next section.

## 3.2 The First Linear System and the First Cycle

For the first cycle of the first system neither $U$ (and $\tilde{U}$) nor $U_{j-1}$ (and $\tilde{U}_{j-1}$) are available. Thus, instead of solving the generalized eigenvalue problem, Ritz pairs are computed as follows:

$$T_1 w = \lambda w \quad \text{and} \quad \tilde{T}_1\tilde{w} = \mu\tilde{w}.$$

Note that the tridiagonals, $T_1$ and $\tilde{T}_1$ in the above equations, are $s \times s$ instead of usual $(s+1) \times s$. From Theorem 2.2 we know $\tilde{T}_1 = T_1^*$, and hence, we just need to solve one system for both the left and the right eigenvectors. Also, as in (3.5) and (3.7), $\lambda$ and $\mu$ are complex conjugate of each other. Now the recycle space is given by

$$U_1 = V_1 W_1 \quad \text{and} \quad \tilde{U}_1 = \tilde{V}_1 \tilde{W}_1.$$

During the second and subsequent cycles of the first linear system, $U_{j-1}$ and $\tilde{U}_{j-1}$ are available but $U$ and $\tilde{U}$ not. Thus, the following variables are used in the generalized eigenvalue problem (3.5):

$$\Phi_j = \begin{bmatrix} U_{j-1} & V_j \end{bmatrix}, \quad \Psi_j = \begin{bmatrix} C_{j-1} & \Upsilon_j \end{bmatrix}, \text{ and } H_j = \begin{bmatrix} I & 0 \\ 0 & \Gamma_j \end{bmatrix},$$

$$\tilde{\Phi}_j = \begin{bmatrix} \tilde{U}_{j-1} & \tilde{V}_j \end{bmatrix}, \quad \tilde{\Psi}_j = \begin{bmatrix} \tilde{C}_{j-1} & \tilde{\Upsilon}_j \end{bmatrix}, \text{ and } \tilde{H}_j = \begin{bmatrix} I & 0 \\ 0 & \tilde{\Gamma}_j \end{bmatrix}.$$

For the first cycle of each of the subsequent linear systems (i.e. $j = 1$), $U$ and $\tilde{U}$ are available, while $U_{j-1}$ and $\tilde{U}_{j-1}$ are not. Thus, the generalized eigenvalue problem (3.5) uses the following variables:

$$\Phi_1 = \begin{bmatrix} U & V_1 \end{bmatrix}, \quad \Psi_1 = \begin{bmatrix} C & \underline{V}_1 \end{bmatrix}, \text{ and } H_1 = \begin{bmatrix} I & B_1 \\ 0 & \underline{T}_1 \end{bmatrix},$$

$$\tilde{\Phi}_1 = \begin{bmatrix} \tilde{U} & \tilde{V}_1 \end{bmatrix}, \quad \tilde{\Psi}_1 = \begin{bmatrix} \tilde{C} & \underline{\tilde{V}}_1 \end{bmatrix}, \text{ and } \tilde{H}_1 = \begin{bmatrix} I & \tilde{B}_1 \\ 0 & \underline{\tilde{T}}_1 \end{bmatrix},$$

where $\underline{V}_1$ and $\underline{\tilde{V}}_1$ denote $V_1$ and $\tilde{V}_1$ with one subsequent Lanczos vector. $\underline{T}_1$ and $\underline{\tilde{T}}_1$ are the

corresponding tridiagonals (with one extra row at the bottom). The final $C_j$ and $\tilde{C}_j$ of the current linear system are the $C$ and $\tilde{C}$ for the next linear system, and thus, need to satisfy the bi-orthogonality condition (2.2). The $C_j$ and $\tilde{C}_j$ as obtained from the previous derivation need not be bi-orthogonal. We discuss how to satisfy this requirement in the next section.

## 3.3 Building Bi-orthogonality between the $C$'s

As seen in Section (3.1), $U_j$ and $\tilde{U}_j$ are given by the following equations:

$$U_j = \Phi_j W_j \quad \text{and} \quad \tilde{U}_j = \tilde{\Phi}_j \tilde{W}_j.$$

Also as seen before, $C_j$ and $\tilde{C}_j$ are given by

$$C_j = AU_j \quad \text{and} \quad \tilde{C}_j = A^* \tilde{U}_j.$$

To construct bi-orthogonal $C_j$ and $\tilde{C}_j$, we use the singular value decomposition (SVD) of $\tilde{C}_j^* C_j$

$$\tilde{C}_j^* C_j = M_j \Sigma_j N_j^*. \tag{3.8}$$

Next we define

$$U_j = \Phi_j W_j N_j = [U_{j-1}\ V_j] W_j N_j \quad \text{and} \quad \tilde{U}_j = \tilde{\Phi}_j \tilde{W}_j M_j = [\tilde{U}_{j-1}\ \tilde{V}_j] \tilde{W}_j M_j. \tag{3.9}$$

The new $C$'s are given as follows:

$$C_j = AU_j = A[U_{j-1}\ V_j] W_j N_j \quad \text{and} \quad \tilde{C}_j = A^* \tilde{U}_j = A^* [\tilde{U}_{j-1}\ \tilde{V}_j] \tilde{W}_j M_j. \tag{3.10}$$

Note that these new $C$'s are bi-orthogonal (i.e. $\tilde{C}_j^* C_j = \Sigma_j$). We build this bi-orthogonality at the end of each cycle. To solve the generalized eigenvalue problem (3.5), we need matrices $\tilde{H}_j^* \tilde{\Psi}_j^* \Psi_j H_j$ and $\tilde{H}_j^* \tilde{\Psi}_j^* \Phi_j$. Computing these matrices explicitly is expensive. However, with algebraic simplifications, these matrices can be built cheaply. We discuss this aspect in the following section.

## 3.4 Simplification of Matrix Blocks

The main cost of building $\tilde{H}_j^* \tilde{\Psi}_j^* \Psi_j H_j$ and $\tilde{H}_j^* \tilde{\Psi}_j^* \Phi_j$ for (3.5) is in computing matrices $\tilde{\Psi}_j^* \Psi_j$ and $\tilde{\Psi}_j^* \Phi_j$ (cross product of two dense matrices). We can simplify these matrices as follows (using bi-orthogonality condition (2.2) and the choice of scaling (2.23)):

$$\tilde{\Psi}_j^* \Psi_j = \begin{bmatrix} \tilde{C}^* \\ \tilde{C}_{j-1}^* \\ \tilde{\Upsilon}_j^* \end{bmatrix} \begin{bmatrix} C & C_{j-1} & \Upsilon_j \end{bmatrix} = \begin{bmatrix} \mathcal{D}_c & \tilde{C}^* C_{j-1} & 0 \\ \tilde{C}_{j-1}^* C & \Sigma_{j-1} & \tilde{C}_{j-1}^* \Upsilon_j \\ 0 & \tilde{\Upsilon}_j^* C_{j-1} & I \end{bmatrix},$$

$$\tilde{\Psi}_j^* \Phi_j = \begin{bmatrix} \tilde{C}^* \\ \tilde{C}_{j-1}^* \\ \tilde{\Upsilon}_j^* \end{bmatrix} \begin{bmatrix} U_{j-1} & V_j \end{bmatrix} = \begin{bmatrix} \tilde{C}^* U_{j-1} & 0 \\ \tilde{C}_{j-1}^* U_{j-1} & \tilde{C}_{j-1}^* V_j \\ \tilde{\Upsilon}_j^* U_{j-1} & \underline{\overline{I}} \end{bmatrix},$$

where $\mathcal{D}_c = \tilde{C}^* C$ is a diagonal matrix independent of the cycle, and $\underline{\overline{I}}$ is an identity matrix with an extra row of zeros at the top and at the bottom. Most of the blocks above can be simplified further. Before doing more simplifications, it is important to point out that because of the bi-orthogonality condition (2.2) and the SVD relations (3.8)-(3.10), we get the following orthogonality conditions on the side:

$$\tilde{C}_{j-2} \perp \Upsilon_j \text{ and} \tag{3.11}$$

$$C_{j-2} \perp \tilde{\Upsilon}_j. \tag{3.12}$$

We now simplify the blocks of the above two matrices (going from top-to-bottom and left-to-right). For this we use the bi-orthogonality condition (2.2), the augmented bi-Lanczos relations (3.1), the SVD relations (3.8)-(3.10), and the new orthogonality conditions (3.11)-(3.12).

- $$\begin{aligned}\tilde{C}^* C_{j-1} = \tilde{C}^* A U_{j-1} &= \begin{bmatrix} \tilde{C}^* A U_{j-2} & \tilde{C}^* A V_{j-1} \end{bmatrix} W_{j-1} N_{j-1}, \\ &= \begin{bmatrix} \tilde{C}^* C_{j-2} & \tilde{C}^* \begin{bmatrix} C\hat{C}^* A V_{j-1} + \Upsilon_{j-1} & \Gamma_{j-1} \end{bmatrix} \end{bmatrix} W_{j-1} N_{j-1}, \\ &= \begin{bmatrix} \tilde{C}^* C_{j-2} & \mathcal{D}_c B_{j-1} \end{bmatrix} W_{j-1} N_{j-1},\end{aligned}$$

where $\tilde{C}^* C_{j-2}$ is available from the previous cycle, and $B_{j-1}$ is computed during the iterations of the bi-Lanczos method (see Chapter 6).

- $$\begin{aligned}\tilde{C}_{j-1}^* C = \tilde{U}_{j-1}^* A C &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{U}_{j-2}^* \\ \tilde{V}_{j-1}^* \end{bmatrix} AC, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{C}_{j-2}^* C \\ \tilde{V}_{j-1}^* AC \end{bmatrix}, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{C}_{j-2}^* C \\ \left[\tilde{C}\check{C}^* A^* \tilde{V}_{j-1} + \tilde{\Upsilon}_{j-1} \tilde{\Gamma}_{j-1}\right]^* C \end{bmatrix}, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{C}_{j-2}^* C \\ \tilde{B}_{j-1}^* \mathcal{D}_c \end{bmatrix}.\end{aligned}$$

As in the previous block, $\tilde{C}_{j-2}^* C$ is available from the previous cycle, and $\tilde{B}_{j-1}$ is computed during the iterations of the bi-Lanczos method.

- $$\begin{aligned}\tilde{C}_{j-1}^* \Upsilon_j = \tilde{U}_{j-1}^* A \Upsilon_j &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{U}_{j-2}^* \\ \tilde{V}_{j-1}^* \end{bmatrix} A \Upsilon_j, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} 0 \\ \tilde{V}_{j-1}^* A \Upsilon_j \end{bmatrix}, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} 0 \\ \left[\tilde{C}\check{C}^* A^* \tilde{V}_{j-1} + \tilde{\Upsilon}_{j-1}\tilde{\Gamma}_{j-1}\right]^* \Upsilon_j \end{bmatrix}, \\ &= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} 0 \\ \tilde{\Gamma}_{j-1}^* \tilde{\Upsilon}_{j-1}^* \Upsilon_j \end{bmatrix},\end{aligned}$$

where

$$\tilde{\Gamma}_{j-1}^* \tilde{\Upsilon}_{j-1}^* \Upsilon_j = \tilde{\Gamma}_{j-1}^* \begin{bmatrix} \tilde{v}_{(j-2)s}^* \\ \tilde{v}_{(j-2)s+1}^* \\ \vdots \\ \tilde{v}_{(j-1)s}^* \\ \tilde{v}_{(j-1)s+1}^* \end{bmatrix} \begin{bmatrix} v_{(j-1)s} & v_{(j-1)s+1} & \cdots & v_{js} & v_{js+1} \end{bmatrix},$$

$$= \tilde{\Gamma}_{j-1}^* \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \end{bmatrix}.$$

- $\tilde{\Upsilon}_j^* C_{j-1} = \tilde{\Upsilon}_j^* A U_{j-1}$,

$$
\begin{aligned}
&= \tilde{\Upsilon}_j^* A \begin{bmatrix} U_{j-2} & V_{j-1} \end{bmatrix} W_{j-1} N_{j-1}, \\
&= \begin{bmatrix} 0 & \tilde{\Upsilon}_j^* A V_{j-1} \end{bmatrix} W_{j-1} N_{j-1}, \\
&= \begin{bmatrix} 0 & \tilde{\Upsilon}_j^* \left[ C\hat{C}^* A V_{j-1} + \Upsilon_{j-1} \Gamma_{j-1} \right] \end{bmatrix} W_{j-1} N_{j-1}, \\
&= \begin{bmatrix} 0 & \tilde{\Upsilon}_j^* \Upsilon_{j-1} \Gamma_{j-1} \end{bmatrix} W_{j-1} N_{j-1},
\end{aligned}
$$

where

$$
\begin{aligned}
\tilde{\Upsilon}_j^* \Upsilon_{j-1} \Gamma_{j-1} &= \begin{bmatrix} \tilde{v}_{(j-1)s}^* \\ \tilde{v}_{(j-1)s+1}^* \\ \vdots \\ \tilde{v}_{js}^* \\ \tilde{v}_{js+1}^* \end{bmatrix} \begin{bmatrix} v_{(j-2)s} & v_{(j-2)s+1} & \cdots & v_{(j-1)s} & v_{(j-1)s+1} \end{bmatrix} \Gamma_{j-1}, \\
&= \begin{bmatrix} 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \Gamma_{j-1}.
\end{aligned}
$$

- $\tilde{C}^* U_{j-1} = \tilde{C}^* \begin{bmatrix} U_{j-2} & V_{j-1} \end{bmatrix} W_{j-1} N_{j-1} = \begin{bmatrix} \tilde{C}^* U_{j-2} & 0 \end{bmatrix} W_{j-1} N_{j-1}$,

where $\tilde{C}^* U_{j-2}$ is available from the previous cycle.

- $$\tilde{C}_{j-1}^* U_{j-1} = M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{U}_{j-2}^* \\ \tilde{V}_{j-1}^* \end{bmatrix} A \begin{bmatrix} U_{j-2} & V_{j-1} \end{bmatrix} W_{j-1} N_{j-1},$$
$$= M_{j-1}^* \tilde{W}_{j-1}^* \begin{bmatrix} \tilde{C}_{j-2}^* U_{j-2} & \tilde{C}_{j-2}^* V_{j-1} \\ \tilde{V}_{j-1}^* C_{j-2} & \tilde{V}_{j-1}^* A V_{j-1} \end{bmatrix} W_{j-1} N_{j-1},$$

where $\tilde{C}_{j-2}^* U_{j-2}$ and $\tilde{C}_{j-2}^* V_{j-1}$ are available from blocks of $\tilde{\Psi}_{j-1}^* \Phi_{j-1}$. $\tilde{V}_{j-1}^* C_{j-2}$ is a subset of $\tilde{\Upsilon}_{j-1}^* C_{j-2}$, and hence, is available from $\tilde{\Psi}_{j-1}^* \Psi_{j-1}$. $\tilde{V}_{j-1}^* A V_{j-1}$ can be simplified as follows:

$$\left(A^* \tilde{V}_{j-1}\right)^* V_{j-1} = \left[\tilde{C}\check{C}^* A^* \tilde{V}_{j-1} + \tilde{\Upsilon}_{j-1} \tilde{\Gamma}_{j-1}\right]^* V_{j-1} = \tilde{\Gamma}_{j-1}^* \tilde{\Upsilon}_{j-1}^* V_{j-1},$$
$$= \tilde{\Gamma}_{j-1}^* \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} = \tilde{T}_{j-1}.$$

- $\tilde{C}_{j-1}^* V_j$ is a subset of $\tilde{C}_{j-1}^* \Upsilon_j$, and hence, is available from $\tilde{\Psi}_j^* \Psi_j$.

Thus, only $\tilde{\Upsilon}_j^* U_{j-1}$ needs to be computed explicitly. For solving the generalized eigenvalue problem (3.5) we need the tridiagonals $T_i$ and $\tilde{T}_i$ explicitly. Only the tridiagonal for the primary system ($T_i$) needs to be computed since the tridiagonal for the dual system ($\tilde{T}_i$) is its conjugate transpose (from Theorem 2.2). The $\alpha$'s and $\beta$'s, as derived in the previous chapter, help in cheaply computing the tridiagonal. Hence, we discuss this aspect in the next section.

## 3.5 Computing Tridiagonal from $\alpha$'s and $\beta$'s

We derive the expression of the tridiagonal for our RBiCG on the same lines as the derivation of the tridiagonal for CG [14]. Giving this derivation here is advantageous for two reasons. Firstly, here we work with $A_1$ instead of $A$ so the steps are not the same. Secondly, in the literature for BiCG, the derivation of the expression for the tridiagonal is hard to find. We start with the augmented bi-Lanczos relation (2.5) to compute the terms of the tridiagonal. That is, writing $(I - C\hat{C}^*)AV_i = V_{i+1}\underline{T}_i$ in the expanded form gives the following:

$$(I - C\hat{C}^*)A[v_1\ v_2 \cdots v_i] = [v_1\ v_2 \cdots v_{i-1}\ v_i\ v_{i+1}] \begin{bmatrix} \delta_1 & \eta_2 & 0 & \cdots & 0 \\ \xi_1 & \delta_2 & \eta_3 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \xi_{i-2} & \delta_{i-1} & \eta_i \\ 0 & \cdots & 0 & \xi_{i-1} & \delta_i \\ 0 & \cdots & 0 & 0 & \xi_i \end{bmatrix}.$$

Equating the $i^{th}$ column above implies

$$\begin{aligned} (I - C\hat{C}^*)Av_i &= \eta_i v_{i-1} + \delta_i v_i + \xi_i v_{i+1} \Rightarrow \\ \xi_i v_{i+1} &= Av_i - \delta_i v_i - \eta_i v_{i-1} - C\hat{C}^* Av_i. \end{aligned}$$

Therefore the "central" equation here is

$$\xi_i v_{i+1} = Av_i - \delta_i v_i - \eta_i v_{i-1} - c_1\hat{c}_1^* Av_i - c_2\hat{c}_2^* Av_i\ \cdots\ - c_k\hat{c}_k^* Av_i, \tag{3.13}$$

The above equation is called "central" because it is the starting point of deriving expressions for $\delta_i$, $\eta_i$, and $\xi_i$. We now construct the expression for $\delta_i$. This is followed by the derivation of the expression for $\eta_i$ and $\xi_i$. Taking the dot product of the above equation with $\tilde{v}_i$, and

using the bi-orthogonality condition (2.2) along with the relation between the residual and the Lanczos vector (Theorem 2.1), leads to the following:

$$\begin{aligned} 0 &= \tilde{v}_i^* A v_i - \delta_i \tilde{v}_i^* v_i - 0 - 0 - 0 \cdots - 0 \quad \Rightarrow \\ \delta_i &= \frac{\tilde{v}_i^* A_1 v_i}{\tilde{v}_i^* v_i} \quad \Rightarrow \\ \delta_i &= \frac{\tilde{r}_{i-1}^* A_1 r_{i-1}}{\tilde{r}_{i-1}^* r_{i-1}}. \end{aligned}$$

The numerator of the above expression can be simplified further using the $p$ recurrences (2.25)-(2.26) and A-orthogonality of the $p$ vectors (Theorem 2.3) as follows:

$$\begin{aligned} \tilde{r}_{i-1}^* A_1 r_{i-1} &= (\tilde{p}_i - \tilde{\beta}_{i-1} \tilde{p}_{i-1})^* A_1 (p_i - \beta_{i-1} p_{i-1}), \\ &= \tilde{p}_i^* A_1 p_i + \beta_{i-1}^2 \tilde{p}_{i-1}^* A_1 p_{i-1}. \end{aligned}$$

Substituting this numerator back in $\delta_i$ and using expressions for $\beta$ (2.35), gives the following:

$$\delta_i = \frac{\tilde{p}_i^* A_1 p_i}{\tilde{r}_{i-1}^* r_{i-1}} + \beta_{i-1} \frac{\tilde{r}_{i-1}^* r_{i-1}}{\tilde{r}_{i-2}^* r_{i-2}} \cdot \frac{\tilde{p}_{i-1}^* A_1 p_{i-1}}{\tilde{r}_{i-1}^* r_{i-1}}.$$

Therefore the $\delta_i$, using expressions for $\alpha$'s (2.33), is given by

$$\delta_i = \frac{1}{\alpha_i} + \frac{\beta_{i-1}}{\alpha_{i-1}}. \tag{3.14}$$

For $i = 1$, the second term in the expression should be omitted. Next we work on the expression for $\eta_i$. Taking the dot product of (3.13) with $\tilde{v}_{i-1}$, and using the bi-orthogonality condition (2.2) along with the Lanczos vector expressions (2.24), leads to the following:

$$\begin{aligned} 0 &= \tilde{v}_{i-1}^* A v_i - 0 - \eta_i \tilde{v}_{i-1}^* v_{i-1} - 0 - 0 \cdots - 0 \quad \Rightarrow \\ \eta_i &= \frac{\tilde{v}_{i-1}^* A_1 v_i}{\tilde{v}_{i-1}^* v_{i-1}} \quad \Rightarrow \\ \eta_i &= \frac{||r_{i-2}||}{||r_{i-1}||} \cdot \frac{\tilde{r}_{i-2}^* A_1 r_{i-1}}{\tilde{r}_{i-2}^* r_{i-2}}. \end{aligned}$$

As in the case of $\delta_i$, we simplify the numerator of the $\eta_i$ above using the bi-orthogonality condition (2.2), Theorem 2.1, $p$ recurrence (2.25), and the residual recurrence (2.29).

$$\begin{aligned}
\tilde{r}_{i-2}^* A_1 r_{i-1} &= \tilde{r}_{i-2}^* A_1 (p_i - \beta_{i-1} p_{i-1}), \\
&= \tilde{r}_{i-2}^* \tfrac{(r_{i-1}-r_i)}{\alpha_i} - \beta_{i-1} \tilde{r}_{i-2}^* \tfrac{(r_{i-2}-r_{i-1})}{\alpha_{i-1}}, \\
&= 0 - \tfrac{\beta_{i-1}}{\alpha_{i-1}} (\tilde{r}_{i-2}^* r_{i-2}) - 0.
\end{aligned}$$

Substituting this numerator back in $\eta_i$ leads to the following:

$$\eta_i = -\frac{||r_{i-2}||}{||r_{i-1}||} \cdot \frac{\beta_{i-1}}{\alpha_{i-1}}. \tag{3.15}$$

Note that $\eta_1$ above does not exist. We now derive the expression for $\xi_i$. This derivation follows the same steps as the derivation for $\eta_i$ except that here we start by taking a dot product with $\tilde{v}_{i+1}$ instead of $\tilde{v}_{i-1}$. That is,

$$\xi_i = \tfrac{\tilde{v}_{i+1}^* A_1 v_i}{\tilde{v}_{i+1}^* v_{i+1}} = \tfrac{||r_i||}{||r_{i-1}||} \cdot \tfrac{\tilde{r}_i^* A_1 r_{i-1}}{\tilde{r}_i^* r_i}.$$

Consider the numerator of the second fraction above.

$$\begin{aligned}
\tilde{r}_i^* A_1 r_{i-1} &= \tilde{r}_i^* A_1 (p_i - \beta_{i-1} p_{i-1}), \\
&= \tilde{r}_i^* \tfrac{(r_{i-1}-r_i)}{\alpha_i} - \beta_{i-1} \tilde{r}_i^* \tfrac{(r_{i-2}-r_{i-1})}{\alpha_{i-1}}, \\
&= -\tfrac{\tilde{r}_i^* r_i}{\alpha_i}.
\end{aligned}$$

Substituting this simplified numerator back in $\xi_i$ above, gives the following:

$$\xi_i = -\frac{||r_i||}{||r_{i-1}||} \cdot \frac{1}{\alpha_i}. \tag{3.16}$$

The tridiagonal is given as follows:

$$T_i = \begin{pmatrix} \frac{1}{\alpha_1} & -\frac{||r_0||}{||r_1||} \cdot \frac{\beta_1}{\alpha_1} & & & \\ -\frac{||r_1||}{||r_0||} \cdot \frac{1}{\alpha_1} & \frac{1}{\alpha_2} + \frac{\beta_1}{\alpha_1} & -\frac{||r_1||}{||r_2||} \cdot \frac{\beta_2}{\alpha_2} & & \\ & & \cdots & & \\ & & \ddots & & -\frac{||r_{i-2}||}{||r_{i-1}||} \cdot \frac{\beta_{i-1}}{\alpha_{i-1}} \\ & & & -\frac{||r_{i-1}||}{||r_{i-2}||} \cdot \frac{1}{\alpha_{i-1}} & \frac{1}{\alpha_i} + \frac{\beta_{i-1}}{\alpha_{i-1}} \end{pmatrix}.$$

As discussed before, for computing the tridiagonal of the dual system, Theorem 2.2 is used. That is, $\tilde{T}_i = T_i^*$.

# Chapter 4

# Recycling CGS

If the problem at hand requires solving both the primary system and the dual system, then BiCG (or RBICG here) is the best choice. However, if only the primary system is to be solved, then two other bi-Lanczos based algorithms, namely, Conjugate Gradient Squared (CGS) [16] and BiCGSTAB [17], might be better. In this chapter we extend recycling to the CGS algorithm. We apply the recycling approach to the BiCGSTAB algorithm in the next chapter.

The BiCG algorithm has a number of drawbacks. It requires the matrix transpose, which may be expensive or in some cases not available. Another disadvantage is that it requires two matrix-vector products to extend the Krylov subspace for the solution $x_i$ by one vector [2]. This lead to the development of the CGS algorithm [16]. In addition, the CGS algorithm often converges faster. We now derive the Recycling Conjugate Gradient Squared (RCGS) algorithm from our RBiCG in the same way as the CGS algorithm is derived from BiCG in [16].

In the next section we describe the polynomial representation of the RBiCG iteration vectors.

We exploit this to develop the polynomial representation of the RCGS iteration vectors in Section 4.2. In Section 4.3, iteration scalars and vectors of the RCGS algorithm are derived. Here we assume the recycle space to exist. One way to generate a recycle space for RCGS is to solve every $k^{th}$ linear system by RBiCG. Computing the recycle space within RCGS is an area of future work. As previously emphasized, $r_0$ and $\tilde{r}_0$ are given by (2.19) and (2.20).

## 4.1 Polynomials in RBiCG

Here we derive the polynomial representation of the $p$ and $r$ vectors. For this we use (2.25) and (2.29).

**Theorem 4.1.** *For the primary system*

$$r_i = \Theta_i(A_1)r_0, \quad p_i = \Pi_{i-1}(A_1)r_0, \tag{4.1}$$

*where* $\Theta_i(A_1)$ *and* $\Pi_{i-1}(A_1)$ *are* $i$*-th and* $(i-1)$*-th degree polynomials in* $A_1$ *with the following polynomial recurrences:*

$$\Theta_i(A_1) = \Theta_{i-1}(A_1) - \alpha_i A_1 \Pi_{i-1}(A_1), \quad \Pi_{i-1}(A_1) = \Theta_{i-1}(A_1) + \beta_{i-1}\Pi_{i-2}(A_1). \tag{4.2}$$

*Proof.* We prove this by induction. The hypothesis is true for $i = 1$ because

$$p_1 = r_0 = \Pi_0(A_1)r_0 \text{ and}$$
$$r_1 = r_0 - \alpha_1 A_1 p_1 = (I - \alpha_1 A_1)r_0 = \Theta_1(A_1)r_0.$$

Assume the hypothesis is true for $i$. Then

$$\begin{aligned} p_{i+1} &= r_i + \beta_i p_i, \\ &= (\Theta_i(A_1) + \beta_i \Pi_{i-1}(A_1))r_0, \\ &= \Pi_i(A_1) r_0, \end{aligned}$$

where $\Pi_i(A_1) = \Theta_i(A_1) + \beta_i \Pi_{i-1}(A_1)$. Also,

$$\begin{aligned} r_{i+1} &= r_i - \alpha_{i+1} A_1 p_{i+1}, \\ &= (\Theta_i(A_1) - \alpha_{i+1} A_1 \Pi_i(A_1))r_0, \\ &= \Theta_{i+1}(A_1) r_0, \end{aligned}$$

where $\Theta_{i+1}(A_1) = \Theta_i(A_1) - \alpha_{i+1} A_1 \Pi_i(A_1)$. Thus the hypothesis is true for $i+1$ and so the proof is complete. □

Similarly, for the dual system

$$\tilde{r}_i = \bar{\Theta}_i(A_1^*)\tilde{r}_0, \quad \tilde{p}_i = \bar{\Pi}_{i-1}(A_1^*)\tilde{r}_0, \tag{4.3}$$

where $\bar{\Theta}_i(\cdot)$ and $\bar{\Pi}_i(\cdot)$ are obtained from $\Theta_i(\cdot)$ and $\Pi_i(\cdot)$ respectively after complex conjugation of the polynomial coefficients (the scalars $\alpha$ and $\beta$). Next, we use this polynomial representation to develop the polynomial representation in RCGS.

## 4.2 Polynomials in RCGS

From the orthogonality condition (2.10) we know that $r_i \perp \tilde{V}_i$. This along with Theorem 2.1 gives $r_i \perp \tilde{r}_j$ for $j < i$. The polynomial representation of the $r_i$ and the $\tilde{r}_j$ vectors from (4.1)

and (4.3) lead to the following:

$$(\bar{\Theta}_j(A_1^*)\tilde{r}_0, \Theta_i(A_1)r_0) = 0 \text{ for } j < i \Leftrightarrow$$
$$(\tilde{r}_0, \Theta_j(A_1)\Theta_i(A_1)r_0) = 0 \text{ for } j < i.$$

The latter form of the inner product contains no matrix transpose, and can be used to compute the iteration vectors of RBiCG. This idea was first proposed in [16] and is fundamental to the CGS algorithm. Consider the numerator and denominator of $\alpha_i = (\tilde{r}_{i-1}, r_{i-1})/(\tilde{p}_i, A_1 p_i)$ from (2.33).

$$(\tilde{r}_{i-1}, r_{i-1}) = (\bar{\Theta}_{i-1}(A_1^*)\tilde{r}_0, \Theta_{i-1}(A_1)r_0) = (\tilde{r}_0, (\bar{\Theta}_{i-1}(A_1^*))^*\Theta_{i-1}(A_1)r_0) = (\tilde{r}_0, \Theta_{i-1}^2(A_1)r_0),$$
$$(\tilde{p}_i, A_1 p_i) = (\bar{\Pi}_{i-1}(A_1^*)\tilde{r}_0, A_1\Pi_{i-1}(A_1)r_0) = (\tilde{r}_0, A_1\Pi_{i-1}^2(A_1)r_0).$$

In the above expressions we need vectors $\Theta_{i-1}^2(A_1)r_0$ and $\Pi_{i-1}^2(A_1)r_0$. Squaring the polynomial recurrences in (4.2) leads to the following recurrences:

$$\Theta_i^2(A_1) = \Theta_{i-1}^2(A_1) + \alpha_i^2 A_1^2 \Pi_{i-1}^2(A_1) - 2\alpha_i A_1 \Theta_{i-1}(A_1)\Pi_{i-1}(A_1),$$
$$\Pi_{i-1}^2(A_1) = \Theta_{i-1}^2(A_1) + \beta_{i-1}^2 \Pi_{i-2}^2(A_1) + 2\beta_{i-1}\Theta_{i-1}(A_1)\Pi_{i-2}(A_1).$$

In order to compute the above recurrences, we need two more recurrences as follows:

$$\Theta_{i-1}(A_1)\Pi_{i-1}(A_1) = \Theta_{i-1}^2(A_1) + \beta_{i-1}\Theta_{i-1}(A_1)\Pi_{i-2}(A_1),$$
$$\Theta_{i-1}(A_1)\Pi_{i-2}(A_1) = \Theta_{i-2}(A_1)\Pi_{i-2}(A_1) - \alpha_{i-1}A_1\Pi_{i-2}^2(A_1).$$

Based on the above idea, we now find the RCGS iteration scalars and vectors.

## 4.3 Iteration Scalars and Vectors

Analogous to CGS, in RCGS we interpret the vectors $\Theta_i^2(A_1)r_0$ as the residuals for the solution estimates $x_i$. That is,

$$r_i = \Theta_i^2(A_1)r_0 \text{ and}$$
$$p_i = \Pi_{i-1}^2(A_1)r_0.$$

The iterations scalars, $\alpha$ and $\beta$, are now given by

$$\alpha_i = \frac{(\tilde{r}_0, r_{i-1})}{(\tilde{r}_0, A_1 p_i)}, \quad \beta_i = \frac{(\tilde{r}_0, r_i)}{(\tilde{r}_0, r_{i-1})}.$$

Using the vectors $u_i = \Theta_{i-1}(A_1)\Pi_{i-1}(A_1)r_0$ and $q_i = \Theta_i(A_1)\Pi_{i-1}(A_1)r_0$, iteration vectors for the RCGS algorithm are defined as follows:

$$\begin{aligned}
u_i &= r_{i-1} + \beta_{i-1} q_{i-1}, \\
p_i &= u_{i-1} + \beta_{i-1}(q_{i-1} + \beta_{i-1} p_{i-1}), \\
q_i &= u_i - \alpha_i A_1 p_i, \\
r_i &= r_{i-1} - \alpha_i A_1 (u_i + q_i).
\end{aligned}$$

Next, we derive the recurrence for the solution update from the recurrence for the residual. That is,

$$\begin{aligned}
& r_i = r_{i-1} - \alpha_i A_1(u_i + q_i) && \Rightarrow \\
& b - Ax_i = b - Ax_{i-1} - \alpha_i A_1(u_i + q_i) && \Rightarrow \\
& Ax_i = Ax_{i-1} + \alpha_i (I - C\hat{C}^*)A(I - C\hat{C}^*)(u_i + q_i) && \Rightarrow \\
& x_i = x_{i-1} + \alpha_i (I - U\hat{C}^* A)(I - C\hat{C}^*)(u_i + q_i).
\end{aligned}$$

Until now for theoretical derivations we have worked with matrix $A_1 = (I - C\hat{C}^*)A(I - C\hat{C}^*)$. As in the RBiCG algorithm, the computations in RCGS can be performed with $A_0 = (I - C\hat{C}^*)A$ instead of $A_1$. This is because $A_1 p_i = A_0 p_i$, $A_1 u_i = A_0 u_i$, and $A_1 q_i = A_0 q_i$. Therefore the iteration scalars and vectors used in computation are

$$
\begin{aligned}
\alpha_i &= \tfrac{(\tilde{r}_0,\ r_{i-1})}{(\tilde{r}_0,\ A_0 p_i)}, \\
\beta_i &= \tfrac{(\tilde{r}_0,\ r_i)}{(\tilde{r}_0,\ r_{i-1})}, \\
u_i &= r_{i-1} + \beta_{i-1} q_{i-1}, \\
p_i &= u_{i-1} + \beta_{i-1}(q_{i-1} + \beta_{i-1} p_{i-1}), \\
q_i &= u_i - \alpha_i A_0 p_i, \\
r_i &= r_{i-1} - \alpha_i A_0 (u_i + q_i), \\
x_i &= x_{i-1} + \alpha_i (I - U\hat{C}^* A)(u_i + q_i).
\end{aligned}
$$

# Chapter 5

# Recycling BiCGSTAB

In general, the CGS algorithm has irregular convergence behavior, and this sometimes leads to cancellation errors [2]. This drawback lead to the development of BiCGSTAB [17]; a smoothly converging variant of CGS. Next, we extend our recycling CGS to the recycling BiCGSTAB algorithm (RBiCGSTAB).

With recycling, the columns of $V_i$ define the basis for $K^i(A_1, r_0)$ and the columns of $\tilde{V}_i$ define the basis for $K^i(A_1^*, \tilde{r}_0)$. As emphasized earlier, $r_0$ and $\tilde{r}_0$ are given by (2.19) and (2.20). Again, in this section we assume that the recycle space has been computed previously.

From the orthogonality condition (2.10) and the above result, we get that $r_i \perp K^i(A_1^*, \tilde{r}_0)$. Using the polynomial representation of $r_i$ (4.1), we get that $\Theta_i(A_1)r_0 \perp K^i(A_1^*, \tilde{r}_0)$. In BiCG, $K^i(A_1^*, \tilde{r}_0) = span\{\tilde{r}_0,\ \bar{\Theta}_1(A_1^*)\tilde{r}_0,\ \cdots,\ \bar{\Theta}_{i-1}(A_1^*)\tilde{r}_0\}$. This leads to the orthogonality condition of the form

$$(\bar{\Theta}_j(A_1^*)\tilde{r}_0, \Theta_i(A_1)r_0) = 0 \text{ for } j < i. \tag{5.1}$$

As observed in [17], the above orthogonality condition must be satisfied by other basis of

$K^i(A_1^*, \tilde{r}_0)$ too. So, other polynomials can be used as well [20]. If $K^i(A_1^*, \tilde{r}_0) = span\{\tilde{r}_0,\ \bar{\Omega}_1(A_1^*)\tilde{r}_0,\ \cdots,\ \bar{\Omega}_{i-1}(A_1^*)\tilde{r}_0\}$, then

$$(\bar{\Omega}_j(A_1^*)\tilde{r}_0, \Theta_i(A_1)r_0) = 0 \text{ for } j < i. \tag{5.2}$$

As in the BiCGSTAB algorithm we define

$$\Omega_i(A_1) = (I - \omega_1 A_1)(I - \omega_2 A_1)\dots(I - \omega_i A_1), \tag{5.3}$$

where $\omega_i$ is selected by minimizing the residual $r_i$ with respect to $\omega_i$, and $\bar{\Omega}_i(A_1) = (I - \bar{\omega}_1 A_1)(I - \bar{\omega}_2 A_1)\dots(I - \bar{\omega}_i A_1)$. Next, we apply the same technique as in the CGS algorithm to get the iteration parameters. We replace expression (5.2) by

$$(\tilde{r}_0, \Omega_j(A_1)\Theta_i(A_1)r_0) = 0 \text{ for } j < i. \tag{5.4}$$

Using the recurrence for the $p$ vector in RBiCG (2.25) and it's corresponding polynomial representation (4.1), we now derive the recurrence for the $p$ vector in RBiCGSTAB.

$$\begin{aligned}
&p_i = r_{i-1} + \beta_{i-1}p_{i-1} && \Rightarrow \\
&\Omega_{i-1}(A_1)p_i = \Omega_{i-1}(A_1)(r_{i-1} + \beta_{i-1}p_{i-1}) && \Rightarrow \\
&\Omega_{i-1}(A_1)\Pi_{i-1}(A_1)r_0 = \Omega_{i-1}(A_1)\Theta_{i-1}(A_1)r_0 + \beta_{i-1}(1 - \omega_{i-1}A_1)\Omega_{i-2}(A_1)\Pi_{i-2}(A_1)r_0.
\end{aligned}$$

Analogous to BiCGSTAB, in RBiCGSTAB we interpret the vectors $\Omega_i(A_1)\Theta_i(A_1)r_0$ as the residuals for the solution estimates $x_i$. Hence, taking $r_i = \Omega_i(A_1)\Theta_i(A_1)r_0$ and $p_i = \Omega_{i-1}(A_1)\Pi_{i-1}(A_1)r_0$, the recurrence for the $p$ vector is given by

$$p_i = r_{i-1} + \beta_{i-1}p_{i-1} - \beta_{i-1}\omega_{i-1}A_1 p_{i-1}. \tag{5.5}$$

Using recurrence of the residual in RBiCG (2.29) and it's corresponding polynomial representation (4.1), we derive the recurrence for the residual in RBiCGSTAB below.

$$\begin{aligned}
&r_i = r_{i-1} - \alpha_i A_1 p_i && \Rightarrow \\
&\Omega_i(A_1) r_i = \Omega_i(A_1)(r_{i-1} - \alpha_i A_1 p_i) && \Rightarrow \\
&\Omega_i(A_1)\Theta_i(A_1) r_0 = (1 - \omega_i A_1)\Omega_{i-1}(A_1)(\Theta_{i-1}(A_1) r_0 - \alpha_i A_1 \Pi_{i-1}(A_1) r_0).
\end{aligned}$$

Therefore, the residual recurrence is given by

$$r_i = r_{i-1} - \alpha_i A_1 p_i - \omega_i A_1 (r_{i-1} - \alpha_i A_1 p_i). \tag{5.6}$$

The derivation of $\alpha$ and $\beta$ in (5.5) and (5.6) closely follows the derivation of BiCGSTAB [2].

$$\alpha_i = \frac{(\tilde{r}_0, r_{i-1})}{(\tilde{r}_0, A_1 p_i)}, \tag{5.7}$$

$$\beta_i = \frac{(\tilde{r}_0, r_i)}{(\tilde{r}_0, r_{i-1})} . \frac{\alpha_i}{\omega_i}. \tag{5.8}$$

Next, we show how to compute the coefficients $\omega_i$'s. From (5.6), we have $r_i = s - \omega_i t$, where $s = r_{i-1} - \alpha_i A_1 p_i$ and $t = A_1 s$. To minimize $r_i$ with respect to $\omega_i$, we need

$$\begin{aligned}
&(r_i, t) = 0 && \Rightarrow \\
&(s, t) - \omega_i (t, t) = 0 && \Rightarrow \\
&\omega_i = \tfrac{(s,t)}{(t,t)}.
\end{aligned}$$

The last step is to develop the recurrence for the solution update. The residual from (5.6)

is given by

$$\begin{aligned} r_i &= r_{i-1} - \alpha_i A_1 p_i - \omega_i A_1 (r_{i-1} - \alpha_i A_1 p_i) && \Rightarrow \\ b - Ax_i &= b - Ax_{i-1} - \alpha_i A_1 p_i - \omega_i A_1 (r_{i-1} - \alpha_i A_1 p_i) && \Rightarrow \\ Ax_i &= Ax_{i-1} + \alpha_i A_1 p_i + \omega_i A_1 (r_{i-1} - \alpha_i A_1 p_i). \end{aligned}$$

Using $A_1$ from (2.7), and taking $s = r_{i-1} - \alpha_i A_1 p_i$, we get the following:

$$\begin{aligned} Ax_i &= Ax_{i-1} + \alpha_i (I - C\hat{C}^*) A (I - C\hat{C}^*) p_i + \omega_i (I - C\hat{C}^*) A (I - C\hat{C}^*) s, \\ &= Ax_{i-1} + \alpha_i (A - AU\hat{C}^* A)(I - C\hat{C}^*) p_i + \omega_i (A - AU\hat{C}^* A)(I - C\hat{C}^*) s. \end{aligned}$$

Therefore, the solution update is given as follows:

$$x_i = x_{i-1} + \alpha_i (I - U\hat{C}^* A)(I - C\hat{C}^*) p_i + \omega_i (I - U\hat{C}^* A)(I - C\hat{C}^*)(r_{i-1} - \alpha_i A_1 p_i). \quad (5.9)$$

As stated earlier for RBiCG and RCGS, for theoretical derivations it is convenient to work with $A_1 = (I - C\hat{C}^*)A(I - C\hat{C}^*)$ because it emphasizes the symmetry in the bi-Lanczos relations (Section 2.1). For computations, we work with $A_0 = (I - C\hat{C}^*)A$ instead of $A_1$ because it is cheaper. Therefore, the iteration scalars and vectors used in computation are as follows:

$$\begin{aligned} &\alpha_i = \tfrac{(\tilde{r}_0,\ r_{i-1})}{(\tilde{r}_0,\ A_0 p_i)}, \\ &\beta_i = \tfrac{(\tilde{r}_0,\ r_i)}{(\tilde{r}_0,\ r_{i-1})} . \tfrac{\alpha_i}{\omega_i}, \\ &\omega_i = \tfrac{(s,t)}{(t,t)}, \text{ with } s = r_{i-1} - \alpha_i A_0 p_i \text{ and } t = A_0 s, \\ &p_i = r_{i-1} + \beta_{i-1} p_{i-1} - \beta_{i-1}\omega_{i-1} A_0 p_{i-1}, \\ &r_i = r_{i-1} - \alpha_i A_0 p_i - \omega_i A_0 (r_{i-1} - \alpha_i A_0 p_i), \\ &x_i = x_{i-1} + \alpha_i (I - U\hat{C}^* A) p_i + \omega_i (I - U\hat{C}^* A)(r_{i-1} - \alpha_i A_0 p_i). \end{aligned} \quad (5.10)$$

# Chapter 6

# Implementation and Results

This chapter starts with a discussion on code optimizations and other implementation details in Section 6.1. Section 6.2 provides results for numerical experiments on RBiCG, RCGS, and RBiCGSTAB for a convection-diffusion problem and IRKA. Until now we have worked in a complex framework. However, our implementation is only for reals. That is, matrix $A^{(j)} \in \mathbb{R}^{n\times n}$ and the right hand sides $b^{(j)}, \tilde{b}^{(j)} \in \mathbb{R}^n$.

## 6.1 Implementation

First we discuss efficient implementation of some vector updates and matrix compositions. Preconditioning is briefly discussed towards the end of the section. The optimizations are done for all the algorithms described (RBiCG, RCGS, and RBiCGSTAB), however, we demonstrate them here for RBiCG.

For updating the solution and residual as in (2.21) and (2.27), vectors $(I - C\hat{C}^T)Ap_i$ and $(I - U\hat{C}^T A)p_i$ are required. The implementation below requires only matrix-vector products,

and hence, keeps the cost low. The solution vector is not used during the iterative steps of the algorithm. Hence, to reduce the cost further, a cumulative update is done to the solution vector.

**During the iterations:**

$$
\begin{aligned}
&1.\ h_i = Ap_i,\ \kappa_i = \hat{C}^T h_i,\ h_i = h_i - C\kappa_i,\\
&2.\ \alpha_i = \frac{\tilde{r}_{i-1}^T r_{i-1}}{\tilde{p}_i^T h_i},\\
&3.\ x_i = x_{i-1} + \alpha_i p_i,\ \varsigma = \varsigma + \alpha_i \kappa_i,\\
&4.\ r_i = r_{i-1} - \alpha_i h_i.
\end{aligned}
$$

**At the end of all iterations:**

$$x_i = x_i - U\varsigma.$$

Solution and residual updates for the dual system have a similar form. For (3.2) and (3.3), computing matrices $B_j = \hat{C}^T A V_j$ and $\tilde{B}_j = \check{C}^T A^T \tilde{V}_j$ is an expensive operation. $\kappa_i$ at the $i^{th}$ iteration, as computed above, can be utilized to cheaply build $B_j$ as follows ($p_i$ used below is defined by (2.25)):

$$
\begin{aligned}
&\kappa_i = \hat{C}^T A p_i = \hat{C}^T A(r_{i-1} + \beta_{i-1} p_{i-1}) \Rightarrow\\
&\kappa_i = \hat{C}^T A r_{i-1} + \beta_{i-1}\kappa_{i-1}.
\end{aligned}
$$

Hence the $i^{th}$ column of $B_j$ is given by

$$\hat{C}^T A v_i = \frac{\kappa_i - \beta_{i-1}\kappa_{i-1}}{||r_{i-1}||}, \tag{6.1}$$

since $v_i = \frac{r_{i-1}}{||r_{i-1}||}$ from (2.24). Similarly, the $i^{th}$ column of $\tilde{B}_j$ is given by

$$\check{C}^T A^T \tilde{v}_i = \frac{\tilde{\kappa}_i - \beta_{i-1}\tilde{\kappa}_{i-1}}{(v_i, \tilde{r}_{i-1})}, \tag{6.2}$$

where $\tilde{\kappa}_i = \check{C}^T \tilde{h}_i$ with $\tilde{h}_i = A^T \tilde{p}_i$. (6.1) and (6.2) are not exact dual of each other because the scaling for $\tilde{v}_i$ is different from that for $v_i$ in (2.24). For some "hard" problems, the algorithm at hand does not converge or takes too much time. In these cases, preconditioning the linear system helps. If preconditioning is used, then the generated recycle space pertains to the preconditioned linear system. We have implemented split preconditioning inside the recycling algorithms. This type of preconditioning is defined as [1, 14]

$$\mathcal{M}_2^{-1} A \mathcal{M}_1^{-1} \chi = \mathcal{M}_2^{-1} b, \; x = \mathcal{M}_1^{-1} \chi.$$

## 6.2 Results

For testing our algorithms, we use the linear system obtained by finite difference discretization of a partial differential equation. The partial differential equation used is [17]

$$-(\mathcal{A}\vartheta_x)_x - (\mathcal{A}\vartheta_y)_y + \mathcal{B}(x,y)\vartheta_x = \mathcal{F},$$

with $\mathcal{A}$ as shown in Figure 6.1, $\mathcal{B}(x,y) = 2e^{2(x^2+y^2)}$, and $\mathcal{F} = 0$ everywhere except in a small square in the center (see Figure 6.1) where $\mathcal{F} = 100$. The domain is $(0,1) \times (0,1)$ with Dirichlet boundary conditions

$$\begin{aligned} &\vartheta(0,y) = \vartheta(1,y) = \vartheta(x,0) = 1, \\ &\vartheta(x,1) = 0. \end{aligned}$$

The standard second order central difference scheme is used for the discretization, and the mesh width is taken as $1/128$. This leads to a nonsymmetric linear system of $127^2$ unknowns. The linear system is preconditioned by a Crout version of ILUT factorization [14] with a

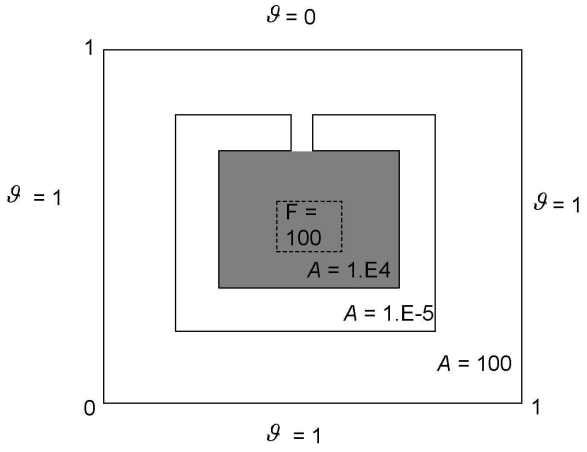


Figure 6.1: Coefficients for the PDE.

drop tolerance of 0.2.

The results are summarized in Figures 6.2 and 6.3. Figure 6.2 shows the benefit of recycling in BiCG. The linear system is solved twice with RBiCG[1]. The subspace is recycled from the first run to the second. This is the "ideal" case for recycling; "We do this to exclude the effects of right-hand sides having slightly different eigenvector decompositions" [13]. Note that the second solve uses 25% fewer iterations than the first solve. For this experiment we take $s = 80$ and $k = 20$. These are chosen based on experience with other recycling algorithms [13, 19]. We do a third run too (that uses the recycle space generated in the second run) to show that the recycle space gets accurate with more runs. This is evident in the figure (the algorithm converges fastest in the third run).

The benefit of recycling in CGS and BiCGSTAB algorithms is shown in Figure 6.3. The

[1]The current implementation scales the Lanczos vectors as follows: $||v_i|| = 1$ and $||\tilde{v}_i|| = 1$ instead of (2.23). This leads to some extra work in computing $\tilde{T}_i$ because now $\tilde{T}_i \neq T_i^T$. Overall, this doesn't affect the results because we are not doing timing comparisons here.

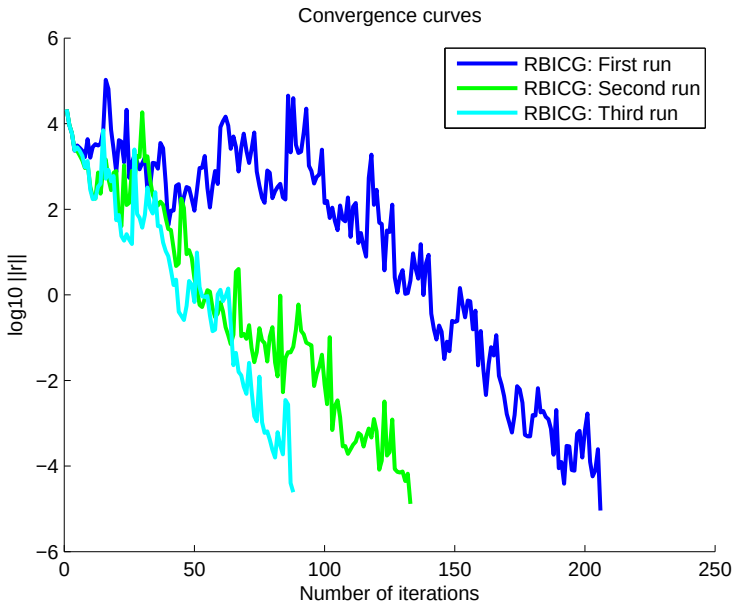


Figure 6.2: Recycling in BiCG.

linear system is first solved with the standard CGS and BiCGSTAB algorithms. After that, the same linear system is solved with RCGS and RBiCGSTAB (using the recycle space developed during the second run of RBiCG in Figures 6.2). As evident in the figure, the recycling algorithms converge much faster. Generating recycle space within RCGS and RBiCGSTAB is an area of future work. This is because real deflation of the primary system residual is done by the left eigenvectors [3], and these left eigenvectors not available during the RCGS and RBiCGSTAB iterations.

Next, we briefly show the application of RBiCG in IRKA [9] that is used for model reduction. IRKA requires linear solves for building matrices

$$\mathcal{V}_i = [A(\sigma_{i_1})^{-1}b,\ \ldots,\ A(\sigma_{i_j})^{-1}b],$$
$$\mathcal{W}_i = [A(\sigma_{i_1})^{-T}\tilde{b},\ \ldots,\ A(\sigma_{i_j})^{-T}\tilde{b}],$$

where $A$ is a matrix that depends on the scalar $\sigma$. This is a good example to show the

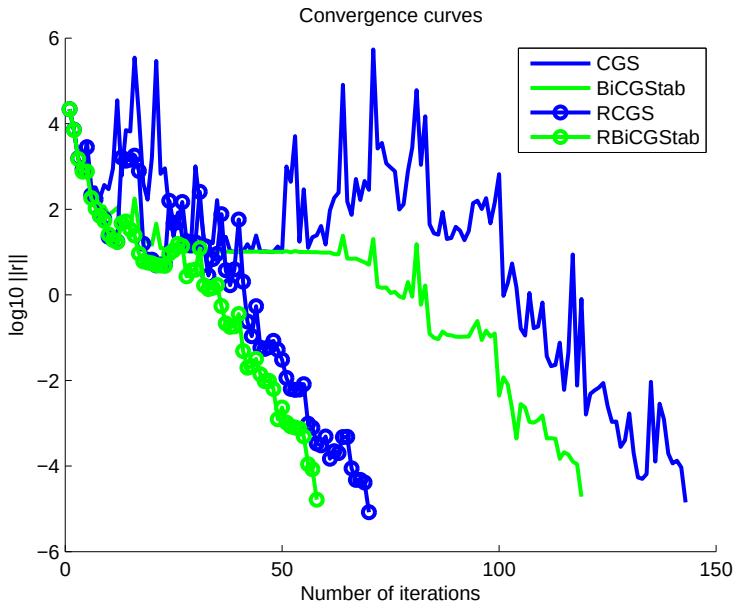


Figure 6.3: Recycling in CGS and BiCGSTAB.

usefulness of RBiCG because of the following reasons. A column of $\mathcal{V}_i$ denotes the primary system and the corresponding column of $\mathcal{W}_i$ denotes the dual system. The change in $\sigma$, both horizontally (along the columns of $\mathcal{V}_i$ and $\mathcal{W}_i$) and vertically (with change in $i$), is not substantial.

We apply the RBiCG algorithm to the above solves as follows. While solving the system $A(\sigma_{i+1_m})^{-1}b$ (also the dual system), with $m = 1, \ldots, j$, we pick the recycle space generated in the previously solved system that has $\sigma$ closest to $\sigma_{i+1_m}$. The pool for picking $\sigma$ currently is $\sigma_{i_1}, \ldots \sigma_{i_j}, \sigma_{i+1_1}, \ldots, \sigma_{i+1_{m-1}}$.

Our $1357 \times 1357$ test dynamical system comes from the rail models (courtesy: Dr. Peter Benner). We take $j = 6$ and initialize the $\sigma$'s as $logspace(-5, 0.7, 6)$. We do the linear solves both with RBiCG and the standard BiCG algorithm. Figure 6.4 shows the convergence curves at $i = 3$. This is for the primary systems (the columns of $\mathcal{V}_i$). Similar graph exists for the dual systems (the columns of $\mathcal{W}_i$). The curves in bold are corresponding to the systems

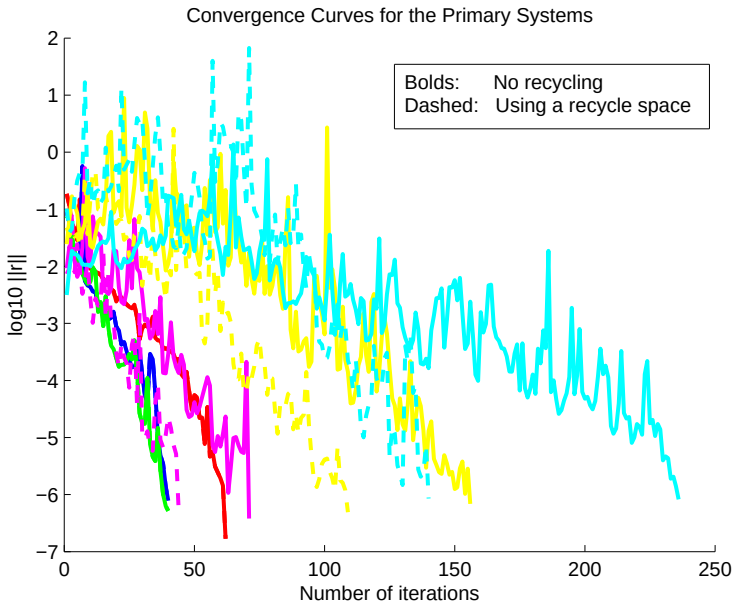


Figure 6.4: Using RBiCG in IRKA.

without recycling, and the dashed curves are corresponding to the systems that use recycling. It is evident that the systems that use a recycle space converge in fewer iterations as compared to the systems that do not use recycling. Note that only three systems use recycling. This is because the other three converge fast, and applying recycling to them is not useful. To summarize, RBiCG works well in IRKA for small models. For larger models, deterioration of the space $range\left(A^* \left[\tilde{U}_{j-1}\ \tilde{V}_j\right]\right)$ makes the algorithm converge poorly. Having a good basis for this space fixes the problem, and we are currently working to obtain such a basis cheaply.

# Chapter 7

# Conclusion

In this thesis, we apply Krylov subspace recycling to the BiCG, CGS, and BiCGSTAB algorithms. For BiCG, we first assume the recycle space exists and modify the algorithm to utilize this space. Second, we show how to build the recycle space cheaply. The resulting algorithm is termed RBiCG. For the recycling version of CGS and BiCGSTAB (RCGS and RBiCGSTAB respectively), we assume that the recycle space is present. Computing the recycle space within RCGS and RBiCGSTAB is an area of future work.

We test our algorithms on a linear system arising from the finite difference discretization of a PDE. To simulate slowly changing linear systems, we solve our linear system twice where-in the subspace is recycled from the first run to the second. This example is an "ideal" case of recycling because it best exemplifies the usefulness of the recycling algorithms. The results indicate that these recycling bi-Lanczos based algorithms can help solve slowly varying sequence of dual linear systems faster. A "real life" example for RBiCG arises while performing model reduction using IRKA [9]. Utilizing RBiCG inside IRKA is currently being researched upon.

# Bibliography


[1] K. Chen. *Matrix Preconditioning Techniques and Applications.* Cambridge University Press, Cambridge, England, UK, 2005.

[2] E. de Sturler. *Iterative Methods on Distributed Memory Computers.* Ph.D. thesis, Delft University of Technology, Delft, The Netherlands, 1994.

[3] E. de Sturler. The convergence of BiCG. Presented at Technische Universität Bergakademie Freiberg / Institut für Angewandte Mathematik II, Freiberg, Germany, 11-15 November 1998.

[4] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3):864–889, 1999.

[5] E. de Sturler. Iterative methods and multigrid. *Course Notes, Numerical Analysis and Software, Department of Mathematics, Virginia Tech*, 2007.

[6] J. W. Demmel. *Applied Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, 1997.

[7] R. Fletcher. Conjugate gradient methods for indefinite systems. *Lecture Notes in Mathematics, Springer Berlin-Heidelberg*, 506:73–89, 1976.

[8] A. Greebaum. *Iterative Methods for Solving Linear Systems.* Society for Industrial and Applied Mathematics, 1997.

[9] S. Gugercin, A. C. Antoulas, and C. A. Beattie. H2 model reduction for large-scale linear dynamical systems. *SIAM Journal on Matrix Analysis and Applications*, 30(2):609–638, 2008.

[10] M. H. Gutknecht. Lanczos-type solvers for nonsymmetric linear systems of equations. *Acta Numerica*, pages 271–397, 1997.

[11] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.*, 49:33–53, 1952.

[12] R. B. Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37, 2002.

[13] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[14] Y. Saad. *Iterative Methods for Sparse Linear Systems, 1st Edition.* PWS (and now ITP), 1996.

[15] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17:401–425, 1996.

[16] P. Sonneveld. CGS, a fast lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36–52, 1989.

[17] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[18] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003.

[19] S. Wang, E. de Sturler, and G. H. Paulino. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. *International Journal for Numerical Methods in Engineering*, 69(12):2422–2468, 2006.

[20] S.-L. Zhang. GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 18(2):537–551, 1997.