Recycling Krylov Subspaces and Preconditioners

Kapil Ahuja

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

Doctor of Philosophy
in
Mathematics

Eric de Sturler, Chair
Christopher A. Beattie
Jeffrey T. Borggaard
David M. Ceperley
Serkan Gugercin
John F. Rossi

October 10$^{\text{th}}$, 2011
Blacksburg, Virginia

Keywords: Bi-Lanczos method, BiCG, preconditioning, sequence of linear systems, model reduction, Variational Monte Carlo, Krylov subspace recycling, updating preconditioners.

Recycling Krylov Subspaces and Preconditioners

Kapil Ahuja

(ABSTRACT)

Science and engineering problems frequently require solving a sequence of single linear systems or a sequence of dual linear systems. We develop algorithms that recycle Krylov subspaces and preconditioners from one system (or pair of systems) in the sequence to the next, leading to efficient solutions.

Besides the benefit of only having to store few Lanczos vectors, using BiConjugate Gradients (BiCG) to solve dual linear systems may have application-specific advantages. For example, using BiCG to solve the dual linear systems arising in interpolatory model reduction provides a backward error formulation in the model reduction framework. Using BiCG to evaluate bilinear forms – for example, in the variational Monte Carlo (VMC) algorithm for electronic structure calculations – leads to a quadratic error bound. Since one of our focus areas is sequences of dual linear systems, we introduce *recycling BiCG*, a BiCG method that recycles two Krylov subspaces from one pair of dual linear systems to the next pair. The derivation of recycling BiCG also builds the foundation for developing recycling variants of other bi-Lanczos based methods like CGS, BiCGSTAB, BiCGSTAB2, BiCGSTAB(l), QMR, and TFQMR.

We develop a generalized bi-Lanczos algorithm, where the two matrices of the bi-Lanczos procedure are not each other's conjugate transpose but satisfy this relation over the generated Krylov subspaces. This is sufficient for a short term recurrence. Next, we derive an augmented bi-Lanczos algorithm with recycling and show that this algorithm is a special case of generalized bi-Lanczos. The Petrov-Galerkin approximation that includes recycling in the iteration leads to modified two-term recurrences for the solution and residual updates.

We generalize and extend the framework of our recycling BiCG to CGS, BiCGSTAB and BiCGSTAB2. We perform extensive numerical experiments and analyze the generated recycle space. We test all of our recycling algorithms on a discretized partial differential equation (PDE) of convection-diffusion type. This PDE problem provides well-known test cases that are easy to analyze further. We use recycling BiCG in the Iterative Rational Krylov Algorithm (IRKA) for interpolatory model reduction and in the VMC algorithm. For a model reduction problem, we show up to 70% savings in iterations, and we also demonstrate that solving the problem without recycling leads to (about) a 50% increase in runtime. Experiments with recycling BiCG for VMC gives promising results.

We also present an algorithm that recycles preconditioners, leading to a dramatic reduction in the cost of VMC for large(r) systems. The main cost of the VMC method is in constructing a sequence of Slater matrices and computing the ratios of determinants for successive Slater matrices. Recent work has improved the scaling of constructing Slater matrices for insulators, so that the cost of constructing Slater matrices in these systems is now linear in the number of particles. However, the cost of computing determinant ratios remains cubic in the number of particles. With the long term aim of simulating much larger systems, we improve the scaling of computing determinant ratios in the VMC method for simulating insulators by using preconditioned iterative solvers.

The main contribution here is the development of a method to *efficiently* compute for the Slater matrices a sequence of preconditioners that make the iterative solver *converge rapidly*. This involves cheap preconditioner updates, an effective reordering strategy, and a cheap method to monitor instability of ILUTP preconditioners. Using the resulting preconditioned iterative solvers to compute determinant ratios of consecutive Slater matrices reduces the scaling of the VMC algorithm from $O(n^3)$ per sweep to roughly $O(n^2)$, where n is the number of particles, and a sweep is a sequence of n steps, each attempting to move a distinct particle. We demonstrate experimentally that we can achieve the improved scaling without increasing statistical errors.

To my mom, Neelam, and my sister, Puja

Acknowledgments

There are multiple people whom I have to thank for this dissertation. First and foremost is my advisor, Eric de Sturler. His constant support and suggestions have greatly shaped this work. I am grateful to him for helping me become a better researcher, writer, presenter, and teacher. I am indebted to him for guiding me through numerous professional uncertainties. I am thankful to him and the National Science Foundation for supporting me through research grants over these years.

The other members of my committee also deserve a special thanks for their time, important suggestions, and support. I would like to thank Serkan Gugercin for helping me with the model reduction aspect of my research, and always supporting me. A large component of my research involved collaboration with Bryan Clark (Princeton University), David Ceperley (University of Illinois), and Jeongnim Kim (NCSA). I am thankful to all of them, especially to Bryan Clark for the time he spent in answering my QMCPACK questions and to David Ceperley for traveling from Illinois to serve on my committee.

This section would be incomplete without acknowledging the wonderful teachers and mentors I have had in the Math Department. I would like to thank John Rossi, Yuriko Renardy, Lizette Zietsman, Leslie Kay, and Eileen Shugart for teaching, advising, and supporting me on various aspects of academic development. I am thankful to Jeffrey Borggaard, Slimane Adjerid, and Nicole Sutphin for helping and guiding me through the various stages of the graduate program. I am thankful to Christopher Beattie for providing insightful comments on my research, and also for always making me feel welcome to discuss research problems.

I am thankful to Peter Haskell for the teaching assistant support and also for providing travel funding. I am also thankful to Michael Parks (Sandia National Labs) for my summer internship at Sandia. I am thankful to SIAM for supporting me through travel awards that helped me present my work at multiple conferences.

My fellow students and friends have played a big role in making my stay fruitful and pleasant here. Eun Chang, Hans-Werner van Wyk, Branimir Anic, Alexander Bondarenko, Steffen Fischer, Garret Flagg, Sarah Wyatt, and Idir Mechai all deserve a special mention. I am also grateful to Kitty Harmon and Alice de Sturler for making me feel at home in Blacksburg. All the volunteers at the Association for India's Development also deserve a thanks for good company.

Finally, I am indebted to my parents, Neelam Ahuja and Om Prakash Ahuja, and my sister and niece, Puja Kalra and Aashi Kalra for everything. It is difficult to put in words their constant love and support.

This material is based upon work supported by the National Science Foundation under Grant No. NSF-EAR 0530643, NSF-DMS 1025327, and NSF-DMS 0645347. 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

List of Figures

List of Tables

Chapter 1

Introduction

We focus on solving the sequence of single linear systems,

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

and the sequence of dual linear systems,

$$A^{(j)}x^{(j)} = b^{(j)}, \quad A^{(j)*}\tilde{x}^{(j)} = \tilde{b}^{(j)}, \tag{1.2}$$

where $A^{(j)} \in \mathbb{C}^{n\times n}$ and $b^{(j)}, \tilde{b}^{(j)} \in \mathbb{C}^n$ vary with j, the matrices $A^{(j)}$ are large and sparse, and the change from one system (or pair of systems) to the next is small. Such systems arise in numerous application areas, for example, model reduction and variational Monte Carlo (VMC). We describe these applications in detail in Chapter 2.

Preconditioned Krylov subspace methods are one of the best candidates for solving large and sparse linear systems. Since here we have sequences of such systems, we recycle Krylov subspaces and preconditioners from one system (or pair of systems) to the next in the sequence.

Recycling Krylov Subspaces

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 of eigenvalues close to the origin usually improves the convergence rate [63, 78]. If the eigenvector corresponding to an eigenvalue is in the Krylov subspace, then that eigenvalue is deflated. This is one of the ways by which eigenvalues can be deflated. Therefore, while solving a system, we select an approximate invariant subspace of $A^{(j)}$ (corresponding to small eigenvalues in absolute value), and use it to accelerate the solution of the next system. Similarly, while solving a pair of systems, we select approximate invariant subspaces of $A^{(j)}$ and $A^{(j)*}$, and use these to accelerate the solution of the next pair of systems. This process is called *Krylov subspace recycling*, and leads to faster convergence for the next system / pair of systems.

For solving a single linear system, 'recycling' has been used in the GCROT [28] and the GMRES-DR [63] algorithms. For solving a sequence of linear systems, this idea was first proposed in [66] where it is applied to the GCROT and the GCRO-DR algorithms. The idea is further adapted in the RMINRES [85] algorithm. GCROT as in [66], GCRO-DR, and RMINRES all focus on solving a sequence of single systems rather than a sequence of two dual systems, which is one of the focus areas here. For a comprehensive discussion of recycling algorithms see [66].

There are important advantages to solving dual linear systems using the BiCG algorithm [37]. BiCG has a short-term recurrence, so very few Lanczos vectors have to be stored. There are many application areas where using BiCG is particularly beneficial. For example, using BiCG to solve dual linear systems arising in interpolatory model reduction provides a backward error formulation in the model reduction framework [15] (see Section 2.1). This property makes BiCG attractive, even for symmetric positive definite (SPD) systems. Another example is when evaluating bilinear forms of the type $u^*A^{-1}w$ where $u, w \in \mathbb{C}^n$ (as in the VMC algorithms; see Section 2.2). Solving the dual linear systems to compute $u^*A^{-1}w$ provides a quadratic error bound [79].

Since BiCG is advantageous for solving dual linear systems and (1.2) has a sequence of such linear systems, we focus on Krylov subspace recycling for BiCG. We refer to our recycling BiCG method as RBiCG, and derive it in Chapter 3. The BiCG algorithm also forms the basis of other popular bi-Lanczos based algorithms like CGS [76], BiCGSTAB [82], BiCGSTAB2 [50], BiCGSTAB(l) [74], QMR [43], and TFQMR [41]. Hence, in Chapter 4 we extend the recycling framework of RBiCG to CGS , BiCGSTAB, and BiCGSTAB2, all of which focus on efficiently solving sequences of single linear systems.

We test our recycling algorithms on a convection-diffusion type of PDE, Iterative Rational Krylov Algorithm (IRKA) [49] for interpolatory model reduction, the VMC algorithm, and a crack propagation simulation. The motivation for testing on IRKA and VMC has been discussed earlier. We test on the convection-diffusion problem because such PDEs are pervasive and provide well-known test cases that are easy to analyze further. This has another advantage: many physical problems that are formulated as a convection-diffusion type PDE (for example, the Oseen's problem), would lead to a potential model reduction problem. The crack propagation example is a good model problem for testing recycling for sequences of single linear systems [66].

For a model reduction problem we show up to a 70% savings in the iteration count, and demonstrate that BiCG takes (about) 50% more time than RBiCG. Application of RBiCG for VMC and RBiCGSTAB for crack propagation simulation, gives promising results. We also analyze the generated recycle spaces.

Recycling Preconditioners

Quite often for "hard" problems, Krylov subspace algorithms do not converge or take too much time. In these cases, preconditioning the linear system (or pair of systems) helps. Computing a new preconditioner at every step in the sequence is usually very expensive. However, we can exploit the structure of the matrix update from one step in the sequence

to the next, and compute a cheap update to the preconditioner.

In the VMC algorithm, we have a sequence of linear systems of the type (1.1). For every new j, the matrix changes only by one row. Hence, in Chapter 5 we propose a method that recycles preconditioners for VMC. Besides performing cheap preconditioner updates, our method monitors and improves deteriorating preconditioners. We show that our algorithm leads to an *order of magnitude improvement* in the scaling of VMC.

In Chapter 6, we provide concluding remarks, and also discuss future work. Here, we briefly present a multilevel preconditioner to further improve the scaling of the VMC algorithm. The initial results with this preconditioner look promising.

To simplify notation, we drop the superscript j in (1.1) – (1.2). 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 dissertation, $||\cdot||$ refers to the two-norm, $(\cdot,\cdot)$ refers to the standard inner product, and e_i is the i-th canonical basis vector. Unless otherwise stated, we collectively call the primary system recycle space and the dual system recycle space as the recycle space.

Chapter 2

Discussion of Applications

In this chapter, we describe the application areas for our recycling algorithms: model reduction (Section 2.1) and the variational Monte Carlo algorithm (Section 2.2). For each application, we first provide the background information, then analyze the application for the use of iterative solvers, and finally provide motivation for using recycling.

2.1 Model Reduction

Consider a single-input/single-output (SISO) linear time-invariant (LTI) system represented as

$$G : \begin{cases} E\,\dot{x}(t) = Ax(t) + bv(t) \\ y(t) = c^*x(t), \end{cases} \qquad \text{or} \quad G(s) = c^*(sE - A)^{-1}b \tag{2.1}$$

where $E, A \in \mathbb{R}^{n\times n}$ and $b, c \in \mathbb{R}^n$. The time-dependent functions $v(t)$, $y(t)$: $\mathbb{R} \to \mathbb{R}$ are the input and output of $G(s)$, respectively, and $x(t) : \mathbb{R} \to \mathbb{R}^n$ is the associated state. In (2.1), $G(s)$ is the transfer function of the system: Let $V(s)$ and $Y(s)$ denote the Laplace transforms of $v(t)$ and $y(t)$, respectively. Then, the transfer function $G(s)$ satisfies $Y(s) = G(s)V(s)$.

By a common abuse of notation, we denote both the underlying dynamical system and its transfer function with G. The dimension of the underlying state-space, n, is called the dimension of G. Systems of the form (2.1) with extremely large state-space dimension n arise in many applications; see [8] and [58] for a collection of such examples. Simulations in such large scale settings lead to overwhelming demands on computational resources. This is the main motivation for model reduction. The goal is to produce a surrogate model of much smaller dimension which provides a high-fidelity approximation of the input-output behavior of the original model G. Let $r \ll n$ denote the order of the reduced-model. The reduced-model is represented, similar to (2.1), as

$$G_r(s) : \begin{cases} E_r\,\dot{x}_r(t) = A_r x_r(t) + b_r v(t) \\ y_r(t) = c_r^* x_r(t), \end{cases} \qquad \text{or} \quad G_r(s) = c_r^*(sE_r - A_r)^{-1} b_r \tag{2.2}$$

where $E_r, A_r \in \mathbb{R}^{r\times r}$ and $b_r, c_r \in \mathbb{R}^r$. In this setting, the common approach is to construct reduced order models via a Petrov-Galerkin projection. This amounts to choosing two r-dimensional subspaces $\mathcal{V}_r$ and $\mathcal{W}_r$ and matrices $V_r \in \mathbb{R}^{n\times r}$ and $W_r \in \mathbb{R}^{n\times r}$ such that $\mathcal{V}_r = \text{Range}(\text{V}_\text{r})$ and $\mathcal{W}_r = \text{Range}(\text{W}_\text{r})$. Then, we approximate the full-order state $x(t)$ as $x(t) \approx V_r x_r(t)$ and enforce the Petrov-Galerkin condition,

$$W_r^*\left(EV_r\dot{x}_r(t) - AV_r x_r(t) - b\,v(t)\right) = 0, \quad y_r(t) = c^* V_r x_r(t),$$

leading to a reduced-order model as in (2.2) with

$$E_r = W_r^* E V_r, \quad A_r = W_r^* A V_r, \quad b_r = W_r^* b, \quad \text{and} \quad c_r = V_r^* c. \tag{2.3}$$

As (2.3) illustrates, the quality of the reduced model depends solely on the selection of the two subspaces $\mathcal{V}_r$ and $\mathcal{W}_r$. In this dissertation, we will choose V_r and W_r to enforce interpolation. For other selections of $\mathcal{V}_r$ and $\mathcal{W}_r$, we refer the reader to [8].

Interpolatory Model Reduction

For a given full-order model $G(s)$, the goal of interpolatory model reduction is to construct a reduced-order model $G_r(s)$ via rational interpolation. Here, we focus on Hermite interpolation. Given the full-order model (2.1) and a collection of interpolation points (also called shifts) $\sigma_i \in \mathbb{C}$, for $i = 1, \ldots, r$, we must construct a reduced-order system by projection as in (2.3) such that $G_r(s)$ interpolates $G(s)$ and its first derivative at selected interpolation points, i.e.,

$$G(\sigma_i) = G_r(\sigma_i) \qquad \text{and} \qquad G'(\sigma_i) = G_r'(\sigma_i) \quad \text{for} \quad i = 1, \ldots, r.$$

Rational interpolation by projection was first proposed in [30, 88, 89]. How to obtain the required projection was derived in [47] using the rational Krylov method [68]. For the special case of Hermite rational interpolation, the solution of the interpolatory model reduction problem is given in Theorem 2.1. For the more general case, we refer the reader to [47] and the recent survey [9].

Theorem 2.1. *Given $G(s) = c^*(sE - A)^{-1}b$ and r distinct points $\sigma_1, \ldots, \sigma_r \in \mathbb{C}$, let*

$$V_r = [(\sigma_1 E - A)^{-1}b \ldots (\sigma_r E - A)^{-1}b], \qquad W_r^* = \begin{bmatrix} c^*(\sigma_1 E - A)^{-1} \\ \vdots \\ c^*(\sigma_r E - A)^{-1} \end{bmatrix}. \tag{2.4}$$

Using (2.3), define the reduced-order model $G_r(s) = c_r^*(sE_r - A_r)^{-1}b_r$. Then $G(\sigma_i) = G_r(\sigma_i)$ and $G'(\sigma_i) = G_r'(\sigma_i)$, for $i = 1, \ldots, r$.

Theorem 2.1 shows how to solve interpolatory model reduction problem via projection for given shifts. However, it does not provide a strategy for choosing good/ optimal interpolation points. This issue has been recently resolved in [49] for the special case of optimality in the

$\mathcal{H}_2$ norm. The $\mathcal{H}_2$ norm of the dynamical system $G(s)$ is defined as

$$\|G\|_{\mathcal{H}_2} = \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \mid G(\jmath a) \mid^2 da \right)^{1/2}.$$

The $\mathcal{H}_2$ norm of G is indeed the $2-\infty$ induced norm of the underlying convolution operator. Then, for any $v \in L^2(\mathbb{R}^+)$, $\|y - y_r\|_{L^\infty} \leq \|G - G_r\|_{\mathcal{H}_2} \|v\|_{L^2}$. To ensure that the output error $y - y_r$ is small in $L^\infty(\mathbb{R}^+)$ uniformly over all inputs v, say, with $\|v\|_{L^2} \leq 1$, we seek a reduced system G_r that makes $\|G - G_r\|_{\mathcal{H}_2}$ small. This leads to the *optimal $\mathcal{H}_2$ model reduction problem*: Given $G(s)$, and a reduced order $r < n$, find $G_r(s)$ that solves

$$\|G - G_r\|_{\mathcal{H}_2} = \min_{dim(\hat{G}_r)=r} \left\| G - \hat{G}_r \right\|_{\mathcal{H}_2}. \tag{2.5}$$

This problem has been studied extensively [61, 87, 49, 77, 84, 48, 13, 14]. The problem (2.5) is a non-convex optimization problem, which makes finding the global minimum, at best, a hard task. Hence, the common approach is to construct reduced-order models that satisfy, for interpolatory model reduction framework, the following first-order necessary conditions.

Theorem 2.2. *([61, 49]) Given $G(s)$, let $G_r(s) = c_r^*(sE_r - A_r)^{-1}b_r$ be an $\mathcal{H}_2$-optimal reduced order model of order r, and let $\hat{\lambda}_1, \ldots, \hat{\lambda}_r$ denote the poles of $G(s)$. Then*

$$G(-\hat{\lambda}_i) = G_r(-\hat{\lambda}_i) \quad \text{and} \quad G'(-\hat{\lambda}_i) = G_r'(-\hat{\lambda}_i) \quad \text{for} \quad i = 1, \ldots, r. \tag{2.6}$$

So, the $\mathcal{H}_2$ optimal approximant $G_r(s)$ is a Hermite interpolant to $G(s)$ at the mirror image of its poles. These poles, whose mirror images are the optimal interpolation points, are not known *a priori*. Hence, the Iterative Rational Krylov Algorithm (IRKA) [49], starting from an initial selection of interpolation points, iteratively corrects the interpolation points until (2.6) is satisfied. Algorithm 2.1 outlines IRKA; for details, see [49].

Algorithm 2.1. *IRKA ([49])*
1. Make an initial shift selection σ_i for $i = 1, \ldots, r$,
2. $V_r = [(\sigma_1 E - A)^{-1} b,\ \ldots,\ (\sigma_r E - A)^{-1} b]$,
3. $W_r = [(\sigma_1 E - A)^{-*} c,\ \ldots,\ (\sigma_r E - A)^{-*} c]$,
4. while (not converged)
◇ $A_r = W_r^* A V_r$, $E_r = W_r^* A V_r$,
◇ $\sigma_i \leftarrow -\lambda_i(A_r, E_r)$ for $i = 1, \ldots, r$,
◇ $V_r = [(\sigma_1 E - A)^{-1} b,\ \ldots,\ (\sigma_r E - A)^{-1} b]$,
◇ $W_r = [(\sigma_1 E - A)^{-*} c,\ \ldots,\ (\sigma_r E - A)^{-*} c]$,
5. $A_r = W_r^* A V_r$, $E_r = W_r^* A V_r$, $b_r = W_r^* b$, $c_r = V_r^* c_r$.

Benefit of using a Petrov-Galerkin Framework

The main cost in IRKA is solving multiple linear systems to compute V_r and W_r. If the dimension of the state-space, n, is large, these systems are generally solved only approximately by an iterative solver. In this context, it is important to asses the accuracy of the computed reduced order model; that is, given the shifts, how accurately the Hermite interpolation problem is solved. This question was studied extensively in [15]. One of the major results, outlined below for our particular case, provides the main motivation for solving the linear systems associated with corresponding columns of V_r and W_r as pair of dual linear systems using BiCG.

Let $\hat{v}_j$ and $\hat{w}_j$, for $j = 1, \ldots, r$, denote the approximate solutions of $(\sigma_j E - A)v_j = b$ and $(\sigma_j E - A)^* w_j = c$, respectively, with residuals $\eta_j = (\sigma_j E - A)\hat{v}_j - b$ and $\xi_j = (\sigma_j E - A)^* \hat{w}_j - c$. Furthermore, let $\hat{v}_j$, $\hat{w}_j$, η_j, and ξ_j satisfy the Petrov-Galerkin condition that there exist spaces $\mathcal{P}$ and $\mathcal{Q}$ such that $\hat{v}_j \in \mathcal{P}$, $\hat{w}_j \in \mathcal{Q}$, $\eta_j \perp \mathcal{Q}$, and $\xi_j \perp \mathcal{P}$. Define the approximate solution matrices ($\hat{V}_r$ and $\hat{W}_r$), the residual matrices (R_b and R_c), and the rank-2r matrix (F_{2r}) as follows:

$$\begin{aligned} &\hat{V}_r = [\hat{v}_1\, \hat{v}_2\, \ldots\, \hat{v}_r], && \hat{W}_r = [\hat{w}_1\, \hat{w}_2\, \ldots\, \hat{w}_r], \\ &R_b = [\eta_1\, \eta_2\, \ldots\, \eta_r], && R_c = [\xi_1\, \xi_2\, \ldots\, \xi_r], \\ &F_{2r} = R_b(\hat{W}_r^* \hat{V}_r)^{-1} \hat{W}_r^* + \hat{V}_r^* (\hat{W}_r^* \hat{V}_r)^{-1} R_c. \end{aligned} \tag{2.7}$$

Also, define the inexact reduced-order order quantities

$$\hat{A}_r = \hat{W}_r^* A \hat{V}_r, \quad \hat{E}_r = \hat{W}_r^* E \hat{V}_r, \quad \hat{b}_r = \hat{W}_r^* b, \quad \text{and} \quad \hat{c}_r = \hat{V}_r^* c.$$

Then, the computed reduced-order model $\hat{G}_r(s) = \hat{c}_r^*(s\hat{E}_r - \hat{A}_r)^{-1}\hat{b}_r$ *exactly* interpolates the perturbed full-order model $\hat{G}(s) = c^*(sE - (A + F_{2r}))^{-1}b$, i.e.,

$$\hat{G}(\sigma_i) = \hat{G}_r(\sigma_i) \quad \text{and} \quad \hat{G}'(\sigma_i) = \hat{G}_r'(\sigma_i), \quad \text{for} \quad i = 1, \ldots, r.$$

Hence, iteratively solving the linear systems while satisfying the Petrov-Galerkin condition above yields a backward error for the interpolatory model reduction that is bounded by $\|F_{2r}\|$, which is governed by the norms of the residuals. The latter are easily controlled in the iterative solver. For details, we refer the reader to [15].

The easiest way to satisfy the Petrov-Galerkin condition above is by solving the dual pairs of linear systems using BiCG. Hence, BiCG is particularly suitable for solving the linear systems in IRKA. However, as IRKA leads to a sequence of dual linear systems, a recycling BiCG algorithm could reduce the total run time for solving all linear systems. In Chapter 3, we propose recycling BiCG (RBiCG) and using numerical experiments show the benefit of recycling Krylov subspaces. Moreover, if we solve the dual pairs of linear systems arising in IRKA by RBiCG the Petrov-Galerkin condition is still satisfied.

Previous Work in Recycling for Model Reduction

Recycling for interpolatory model reduction in the Galerkin projection setting, i.e., with $W_r = V_r$, has been considered in [17] and [36]. In that case, there are no dual systems to solve, and therefore approaches based on GCR [35] and GMRES [71] are considered, respectively, for a sequence of (single) linear systems, as opposed to our approach based on BiCG for a sequence of dual linear systems. Also in other respects, the approach for improving the linear solver and the model reduction context are quite different from here.

In [17], the focus is on efficiently solving linear systems with a fixed coefficient matrix and multiple right hand sides ($Ax^{(j)} = b^{(j)}$), recycling descent vectors (in GCR). Furthermore, the authors target model reduction with a single interpolation point but interpolating higher derivatives.

2.2 The Variational Monte Carlo Method

Quantum Monte Carlo (QMC) methods [80, 52] produce highly accurate quantitative results for many body systems. They have a wide range of applications, including the study of the electronic structure of helium, molecules, solids, and lattice models.

The variational Monte Carlo method (VMC) method is one of many forms of QMC. Although our focus will be on VMC, our results naturally transfer to diffusion Monte Carlo methods [38] as well. The VMC method for computing the ground-state expectation value of the energy combines variational optimization with the Monte Carlo algorithm to evaluate the energy and possibly other observables [38, 80].

The inner loop of VMC involves sampling over configurations R with the probability density $|\Psi_\alpha(R)|^2$ induced by the many body trial wave function $\Psi_\alpha(R)$ [80]. Here, α denotes the vector of variational parameters over which we minimize, and R is a $3n$-dimensional vector representing the coordinates of all the particles in the system. This sampling is done using a Markov Chain Monte Carlo approach. For each independent configuration R generated from the Markov Chain, observables $\mathcal{O}(R, \Psi_\alpha)$ are computed and averaged. Although there are many possible observables, our discussion will focus on the the local energy, $E_{L,\alpha}$. The average of $E_{L,\alpha}$ is the function being optimized. In the rest of this section, we will further detail the separate pieces of this process with a particular focus on the computational bottlenecks.

Scaling of VMC

The VMC method is computationally expensive, and current algorithms and computers limit typical system sizes for fermionic systems to about a thousand particles [38, 62, 73]. There are two main bottlenecks to simulating larger systems. The first is constructing a sequence of millions of so-called Slater matrices, and the second is computing the ratios of determinants of successive Slater matrices (we will describe Slater matrices and other relevant concepts shortly). Our longer term aim is to develop methods for the efficient simulation of systems with 10^4 to 10^5 particles (on high-end computers).

Let n, the system size, be the number of particles (electrons) in the system, which also equals the number of orbitals. Each orbital is a single particle wave function. For simplicity we ignore spin; incorporation of spin is straightforward. In VMC, we estimate observables by conditionally moving particles one by one and accumulating 'snapshots' of the observables. We define the attempt to move all the particles in the system once as a sweep. The cost of constructing Slater matrices depends on the type of basis functions used for building the orbitals. However, the cost for the generic case is $O(n^2)$ per sweep [86] (constant cost per element in the matrix). Recent physics papers [6, 5, 86] discuss methods to reduce this cost to $O(n)$ by optimizing the orbitals, such that the matrix is optimally sparse. Therefore only a linear number of elements in the matrix must be filled and computing the matrix becomes cheap. This can only be shown to be rigorous for certain physical systems (like insulators). These methods are referred to as 'linear scaling' or $O(n)$ methods, because, in many cases, the cost of computing the Slater matrices dominates. However, these methods are not truly linear, since the cost of computing the determinant ratios is still $O(n^3)$, which will dominate the cost of the VMC method for larger n. In this dissertation, we focus on reducing this cost for insulators (or, more generally, for systems with sparse Slater matrices) to $O(n^2)$.

Computing Determinant Ratio

The VMC method generates a sequence of matrices,

$$A_{k+1} \;=\; A_k + e_{i_k} u_k^T, \tag{2.8}$$

where k indicates the Monte Carlo step or particle move, A_k and A_{k+1} are the Slater matrices before and after the proposed move of particle i_k, e_{i_k} is the corresponding Cartesian basis vector, and u_k gives the change in row i_k resulting from moving particle i_k. The acceptance probability of the (conditional) move depends on the squared absolute value of the determinant ratio of the two matrices,

$$\frac{|\det A_{k+1}|^2}{|\det A_k|^2} = \left|1 + u_k^T A_k^{-1} e_{i_k}\right|^2. \tag{2.9}$$

The *standard algorithm* in VMC [24] uses the explicit inverse of the Slater matrix, A_k^{-1}, to compute this ratio, and it updates this inverse according to the Sherman-Morrison formula [45, p. 50] if the particle move is accepted, resulting in $O(n^3)$ work per sweep. For stability, the inverse is occasionally recomputed from scratch (but such that it does not impact the overall scaling). For systems with fewer than a thousand particles this method is practical. Recently, in [65] a variant of this algorithm was proposed that accumulates the multiplicative updates to the exact inverse,

$$A_{k+1}^{-1} = (I - \frac{1}{1 + u_k^T A_k^{-1} e_{i_k}} A_k^{-1} e_{i_k} u_k^T) A_k^{-1},$$

and applies those recursively to compute the ratio (2.9), an approach well-known in numerical optimization for Broyden-type methods [56, p. 88]. The same idea is also discussed in [10, Appendix A]. This approach requires $O(kn)$ work for the k^{th} Monte Carlo step, and therefore $O(k^2 n)$ work for the first k Monte Carlo steps. Hence, a single sweep (n Monte Carlo steps) takes $O(n^3)$ work. If the inverse is not recomputed once per sweep, the approach will actually scale worse than the standard algorithm. So, this method does not decrease the total number

of required operations (or scaling), but often has superior cache performance resulting in an increase in speed.

In Chapter 5, we propose an algorithm for computing determinant ratios for insulating systems in the VMC method with roughly $O(n^2)$ work per sweep, providing a significant improvement in the order of complexity over the standard algorithm; see the discussion of complexity and Tables 5.4 and 5.5 in Section 5.3.

Rather than keeping and updating the inverse of the Slater matrices, we can compute (2.9) in step k by solving $A_k z_k = e_{i_k}$ and taking the inner product of z_k with u_k. Since, for certain cases (like insulators), the Slater matrices with optimized orbitals are sparse, and the relative sparsity increases with the number of particles and nuclei in a system for a given material, preconditioned iterative methods are advantageous, especially since the accuracy of the linear solve can be modest. As we will show, convergence is rapid with a good preconditioner, and hence we use preconditioned full GMRES [71]. However, other Krylov methods are equally possible and might be more appropriate for harder problems. *The main challenge in our approach is to generate, at low cost, a sequence of preconditioners corresponding to the Slater matrices that result in the efficient iterative solution of each system.* We explain how to resolve this problem in section 5.2.

Another way of computing determinant ratio is to rewrite (2.9) as

$$\frac{|\det \tilde{A}_k|^2}{|\det A_k|^2} = \left|1 + \left(A_k^{-T} u_k\right)^T A_k \left(A_k^{-1} e_{i_k}\right)\right|^2 .$$

Hence, we can solve the following two systems by an iterative solver:

$$A_k x = e_{i_k} \quad \text{and} \quad A_k^T \tilde{x} = u_k. \tag{2.10}$$

In exact arithmetic, we can show that solving these dual linear systems using BiCG provides a quadratic error bound[1].

[1]This is not true in finite precision computations. In [79], the authors propose a solution to this problem.

Theorem 2.3. *([79]) If $A_k x = e_{i_k}$ and $A_k^T \tilde{x} = u_k$ are solved with BiCG such that $\|e_{i_k} - A_k x\| = \mathcal{O}(\epsilon_1)$ and $\|u_k - A_k^T \tilde{x}\| = \mathcal{O}(\epsilon_2)$, then $\|\tilde{x}^T A_k x - u_k^T A_k^{-1} e_{i_k}\| = \mathcal{O}(\epsilon_1 \epsilon_2)$.*

This results makes BiCG competitive to algorithms like GMRES for computing the determinant ratio. However, as we have a sequence of dual linear systems, RBiCG could reduce the total run time for solving all linear systems. Note that the quadratic error bound result holds when RBiCG is used. Therefore in Section 3.5, we apply RBiCG for VMC.

Many Body Wave Function

Although wave functions come in a variety of forms, the most common form is

$$\Psi_\alpha(r_1, r_2, ..., r_n) = \exp\left(\sum_{ij} f(r_i - r_j)\right) \det(A), \quad \text{with} \tag{2.11}$$

$$A = \begin{pmatrix} \phi_1(r_1) & \phi_2(r_1) & \dots & \phi_n(r_1) \\ \phi_1(r_2) & \phi_2(r_2) & \dots & \phi_n(r_2) \\ \vdots & & \ddots & \vdots \\ \phi_1(r_n) & \phi_2(r_n) & \dots & \phi_n(r_n) \end{pmatrix}, \tag{2.12}$$

where f is called the Jastrow factor [24][80, p. 319], r_i represents the coordinates of particle i (part of the vector R), A is called a Slater matrix (with elements $a_{i,j} = \phi_j(r_i)$), and its determinant is called a Slater determinant. Each function ϕ_j is a scalar function of position and is referred to as a single particle orbital or single particle wave function. The calculation optimizes over these single particle orbitals as functions of the parameter vector α. In this dissertation, we assume that the particles are confined to a cube and periodic boundary conditions are used.

The nature of the single particle orbitals depends on the physical system being simulated. (Since the determinants of the Slater matrices are invariant, up to a sign, under unitary

We discuss the solution in Chapter 3, since it easily applies to RBiCG as well.

transformations, we can only talk about properties of the set of orbitals.) Two important general classes of electronic systems are metals and insulators. In a metal, the single particle orbitals extend throughout all of space; there is no unitary transformation such that each orbital will have compact support. Moreover, forcing compact support by maximally localizing the orbitals and truncating them outside a fixed sphere can induce serious qualitative errors, e.g., changing the system from a metal to an insulator. In contrast, for an insulator there exists a choice of single particle orbitals that have local support (or truncating them to introduce such local support produces minimal errors). As the size of the system is increased, the support of the individual orbitals remains fixed, which leads to increasingly sparse matrices.

Previous work [5, 6, 86] has developed methods to find unitary transformations that minimize the spread of the single particle orbitals. These methods often go by the name *linear scaling methods*, although in practice they simply minimize the number of non-zero elements in the Slater matrix. Although this makes computing and updating the Slater matrix cheaper, the determinant computation still scales as $O(n^3)$. Nonetheless, this is an important advance, which means that, for an insulating system, the Slater matrix can be made sparse. In this work, we focus on insulators and we leverage this starting point in our work with the assumption that our single particle orbitals are maximally localized.

The specific details of the single particle wave functions depend sensitively on the exact material being simulated and are unimportant for our study. Instead, we use a set of Gaussians centered on n points tiled in space on a b.c.c. (body centered cubic) lattice as representative insulator single particle orbitals:

$$\phi_j(\mathbf{r}) = e^{-k\|\mathbf{r}-\mathbf{Z}_j\|^2}, \quad \text{for } j = 1 \dots n, \tag{2.13}$$

where the $\mathbf{Z}_j$ are the lattice positions (i.e., physically the position of nucleus j), and k determines the rate of decay of ϕ_j. We will truncate these Gaussian functions so that they vanish outside some fixed radius (see section 5.3). These single particle orbitals have the

same qualitative features as realistic single particle orbitals for insulators, have nice analytical properties that allow for easier analysis, and a solution to this problem will almost certainly translate to generic insulators. These Gaussians have a tuning parameter k that determines their (effective) width and physically the bandwidth of the insulator. For these studies, we focus on $k = 1$. (We remind the reader that we do not take spin degrees of freedom into account; in realistic simulations, one would have separate determinants for spin up and spin down electrons.) Furthermore, we choose the unit of length such that the density of electrons is $3/(4\pi)$. This implies that the b.c.c. lattice spacing is 2.031 and the nearest neighbor distance between lattice/orbital positions is 1.759. In this case, the Slater matrices depend on three choices (parameters), first, the type of lattice defining the orbital/nuclei positions, second, the spacing of the lattice positions, and, third, the decay rate of the Gaussian orbitals defined by k. After fixing the type of lattice (body centered cubic), physically, only the spacing of the orbital centers relative to the decay of the Gaussians is relevant. Hence, for analyzing the dependence of matrix properties on the parameters, we need only vary k.

Markov Chain Monte Carlo

The Monte Carlo method is used to compute high dimensional integrals that arise while studying properties of many body systems (one such property is local energy, which we discuss in the next section). Direct sampling would be very inefficient, because the wave function assumes large values only in a small region of the $3n$ dimensional space. Therefore, a Markov Chain Monte Carlo algorithm (MCMC) using a Metropolis update rule is used. For a comprehensive discussion of the Metropolis algorithm see [24, 38, 80]. The first VMC for bosonic systems was reported in [60].

The MCMC algorithm samples configurations as follows. At each step, the current configuration R (representing the collective coordinates of all particles) is changed by moving one particle a small (random) distance, generating a trial configuration R'. The particles can be moved in order or by random selection. The trial configuration is accepted with a

probability that is equal to the ratio of the probabilities (densities) of the two configurations, assuming uniform sampling of the trial coordinate. Hence we compute $\frac{|\Psi(R')|^2}{|\Psi(R)|^2}$ and compare with a random number drawn from the uniform distribution on $(0, 1)$; the new configuration is accepted if the ratio is larger than the random number. Hence, this is where the determinant ratios arise. The exponentials of Jastrow factors must be computed as well, but since they are cheap for sufficiently large n (in fact, they are identical in form to what is done in well-studied classical simulations [39, 7]), we will ignore them in this dissertation. If the trial configuration is accepted, the new configuration becomes R', otherwise the new configuration is R (again). Since we move a single particle at each step, say particle i, in the trial configuration only r_i is changed to r_i', and the Slater matrices (2.12) for the current configuration R and the trial configuration R' differ only in row i. Therefore, consecutive Slater matrices in our MCMC algorithm differ in one row or are the same (when the trial configuration is rejected). We refer to the attempted move of one particle as a step and to the sequence of attempted moves of all particles as a sweep.[2]

Local Energy

One important property of many body systems to calculate is the expectation value of the energy [38],

$$E_V = \frac{\int \Psi_\alpha^*(R) H \Psi_\alpha(R) dR}{\int \Psi_\alpha^*(R)\Psi_\alpha(R)dR} = \frac{\int |\Psi_\alpha(R)|^2 E_{L,\alpha}(R) dR}{\int \Psi_\alpha^*(R)\Psi_\alpha(R)dR}, \tag{2.14}$$

where $E_{L,\alpha}(R) = (H\Psi_\alpha(R))/\Psi_\alpha(R)$ is referred to as the local energy, and H denotes the Hamiltonian of the system; see, for example, [80, section 4.5] and [52, p. 45]. Notice that the observable $E_{L,\alpha}(R)$ averaged over samples taken from the VMC sampling gives us the expectation value of the energy. The algorithm assumes that $\Psi_\alpha(R)$ and $\nabla\Psi_\alpha(R)$ are continuous in regions of finite potential. The computation of $E_{L,\alpha}(R)$ requires the calculation of the Laplacian $\nabla_i^2\Psi_\alpha(R)/\Psi_\alpha(R)$ and gradient $\nabla_i\Psi_\alpha(R)/\Psi_\alpha(R)$ with respect to each particle

[2]Multiple or all particle moves can also be made but require more sweeps (because the rejection rate is higher) and are no more efficient per sweep.

i. This must be done once per sweep. These quantities are computed by evaluating another determinant ratio. For a given i, we replace row i in the Slater matrix by its Laplacian respectively its gradient, and then evaluate the ratio of the determinant of this matrix with the determinant of the Slater matrix. As this has to be done once per sweep, it also scales as $O(n^3)$, and the methods described in this paper naturally generalize to evaluating these quantities by iterative methods with a reduced scaling.

As a point of note, the only computationally slow aspect of the Diffusion Monte Carlo method (DMC) that differs from VMC involves computing the gradient of the wave function for particle i at each step where particle i is moved. Again, the determinant ratio methods discussed in this paper are also applicable to this situation.

Optimization

The outer loop of the VMC method consists of updating the vector of variational parameters, α, after the MCMC evaluation of the average of $E_{L,\alpha}$, so as to minimize the total average energy. In principle, these variational parameters could vary over all possible functions for all n single particle orbitals. In practice, the orbitals are often optimized over a smaller subclass of possible functions. For example, in our model system, one might imagine optimizing k or the location of the orbitals, Z_j (i.e., off the b.c.c. lattice). In more realistic scenarios, each orbital itself has more structure and might be expanded in terms of basis functions such as plane waves or Gaussian basis functions (2.13). Often, then, the expansion factors will be optimized (in addition to the Jastrow factors). Care must be taken when using more complicated representations. If, for example, each single particle orbital is composed of n plane waves and there are n orbitals to be evaluated for n particles, even constructing the Slater matrix would scale as $O(n^3)$ per sweep. However, by tabulating the orbitals on a grid, and doing a table lookup when needed, the cost of constructing the matrix can be brought down to $O(n^2)$ operations per sweep, since there are $O(n^2)$ matrix elements, or $O(n)$ if the matrix is sparse.

Chapter 3

Recycling BiConjugate Gradient

This chapter consists of five sections. In Section 3.1, we discuss the BiCG algorithm. The RBiCG algorithm is derived in Section 3.2. Sections 3.3, 3.4, and 3.5 describe the use of RBiCG (including results) for a convection-diffusion type of PDE, IRKA for interpolatory model reduction, and VMC respectively. All experiments are done using Matlab based code.

3.1 BiCG

For the primary system ($Ax = b$), let x_0 be the initial guess with residual $r_0 = b - Ax_0$. Krylov subspace methods (in general) find approximate solutions by projection onto the Krylov subspace associated with A and r_0 [83]. The i-th solution iterate is given by

$$x_i = x_0 + \varrho_i, \tag{3.1}$$

where $\varrho_i \in \mathcal{K}^i(A, r_0) \equiv span\{r_0,\ Ar_0,\ A^2r_0,\ \cdots,\ A^{i-1}r_0\}$ is defined by some projection. The BiCG method defines this projection using the Krylov subspace associated with the dual system.

Before defining the projection, we need good bases for the two Krylov subspaces (primary and dual). BiCG uses the bi-Lanczos algorithm [59, 37] to compute these bases. Let columns of $V_i = [v_1\ v_2\ \ldots\ v_i]$ define the basis of the primary system Krylov space, and the columns of $\tilde{V}_i = [\tilde{v}_1\ \tilde{v}_2\ \ldots\ \tilde{v}_i]$ define the basis of the dual system Krylov space. The bi-Lanczos algorithm computes columns of V_i and $\tilde{V}_i$ such that, in exact arithmetic, $V_i \perp_b \tilde{V}_i$, where $\perp_b$ is referred to as bi-orthogonality. The columns of V_i and $\tilde{V}_i$ are called Lanczos vectors. There is a degree of freedom in choosing the scaling of the Lanczos vectors [46, 51, 70]. Using the scaling

$$||v_i|| = 1, \qquad (v_i, \tilde{v}_i) = 1, \tag{3.2}$$

we initialize the Lanczos vectors as follows:

$$v_1 = \tfrac{r_0}{||r_0||}, \quad \tilde{v}_1 = \tfrac{\tilde{r}_0}{(v_1, \tilde{r}_0)}.$$

The $(i+1)$-th Lanczos vectors are given by

$$\begin{aligned} \gamma v_{i+1} &= Av_i - V_i\tau \perp \tilde{V}_i, \\ \tilde{\gamma}\tilde{v}_{i+1} &= A^*\tilde{v}_i - \tilde{V}_i\tilde{\tau} \perp V_i, \end{aligned}$$

where γ and $\tilde{\gamma}$ are unknown scalars, and τ and $\tilde{\tau}$ are unknown vectors. The computation of the $(i+1)$-th Lanczos vectors requires only the i-th and the $(i-1)$-th Lanczos vectors (see [70]). These 3-term recurrences are called the bi-Lanczos relations, and are defined as follows:

$$\begin{aligned} AV_i &= V_{i+1}\underline{T}_i = V_iT_i + t_{i+1,i}v_{i+1}e_i^T, \\ A^*\tilde{V}_i &= \tilde{V}_{i+1}\underline{\tilde{T}}_i = \tilde{V}_i\tilde{T}_i + \tilde{t}_{i+1,i}\tilde{v}_{i+1}e_i^T, \end{aligned} \tag{3.3}$$

where T_i, $\tilde{T}_i$ are $i \times i$ tridiagonal matrices, $t_{i+1,i}$ is the last element of the last row of $\underline{T}_i \in \mathbb{C}^{(i+1)\times i}$, and $\tilde{t}_{i+1,i}$ is the last element of the last row of $\underline{\tilde{T}}_i \in \mathbb{C}^{(i+1)\times i}$.

The next step is to find approximate solutions by projection. To exploit the efficiency of

short-term recurrences in the bi-Lanczos algorithm, BiCG uses the bi-orthogonality condition to define the projection. This leads to a Petrov-Galerkin approach. Since the columns of V_i form a basis for $\mathcal{K}^i(A, r_0)$, we can define ϱ_i in (3.1) as $\varrho_i = V_i y_i$, and the Petrov-Galerkin condition then implies

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

The vector y_i is defined by this orthogonality condition. The solution iterate for the dual system ($A^*\tilde{x} = \tilde{b}$), $\tilde{x}_i$, is similarly defined by $\tilde{x}_i = \tilde{x}_0 + \tilde{V}_i\tilde{y}_i$ and $\tilde{r}_i \perp V_i$. Further simplifications lead to the standard BiCG algorithm (Algorithm 3.1) [37, 83].

Next, we briefly discuss the breakdown conditions in BiCG and their solutions [46, 83]. The breakdown conditions in RBiCG are the same, and hence, similar solutions can be applied. The first breakdown happens when at any i^{th} step, $\tilde{r}_i^* r_i = 0$. This breakdown is in the underlying bi-Lanczos algorithm and is referred to as the *serious breakdown*. There exist look-ahead strategies [42] to avoid this breakdown. The two-term recurrence for the solution update requires a pivotless LDU decomposition of the tridiagonal matrix T_i, which may not always exist. This breakdown is referred to as the breakdown of the *second kind*, and can be avoided by performing decomposition with 2×2 block diagonal elements [11]. Note that extensive experiments show that BiCG and RBiCG work well, and breakdowns do not happen often in practice. Hence, for the purpose of this dissertation, we assume that breakdowns do not occur.

3.2 Recycling Krylov Subspaces in BiCG

We first introduce a generalization of BiCG. Here we show that even for non-Hermitian matrices one can build bi-orthogonal bases (for the associated two Krylov subspaces) using a short-term recurrence. Next, we briefly revisit augmented bi-Lanczos [3]. Expanding the search space to include a recycle space leads to an augmented bi-orthogonality condition.

Algorithm 3.1. *BiCG (adapted from [83])*
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 $\tilde{x}_0$ to a random vector.
3. Set $p_0 = 0$, $\tilde{p}_0 = 0$, and $\beta_0 = 0$. Choose the convergence tolerance (`tol`), and the maximum number of iterations (`itn`).
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**.

The augmented bi-Lanczos algorithm computes bi-orthogonal bases for the two Krylov subspaces such that this augmented bi-orthogonality condition is satisfied. We then show that augmented bi-Lanczos is a special case of generalized bi-Lanczos. We formally derive the two-term recurrence for the solution update, and also review how to compute recycle space cheaply [3]. Finally, we provide a cost analysis of RBiCG.

The Generalized Bi-Lanczos Algorithm

There are number of ways of computing good bases for Krylov subspaces $\mathcal{K}^m(B, v_1)$ and $\mathcal{K}^m(\tilde{B}, \tilde{v}_1)$, where B and $\tilde{B}$ are $n \times n$ general matrices, and v_1 and $\tilde{v}_1$ are any two n dimensional vectors. Let the columns of $V_m = [v_1 \ v_2 \ \ldots \ v_m]$ and $\tilde{V}_m = [\tilde{v}_1 \ \tilde{v}_2 \ \ldots \ \tilde{v}_m]$ define one such pair of good bases for $\mathcal{K}^m(B, v_1)$ and $\mathcal{K}^m(\tilde{B}, \tilde{v}_1)$, respectively. We compute these bases using the following in-principle full recurrences:

$$\begin{aligned} \beta_{i+1,i} v_{i+1} &= Bv_i - \beta_{ii} v_i - \beta_{i-1,i} v_{i-1} - \ldots - \beta_{1i} v_1, \\ \tilde{\beta}_{i+1,i} \tilde{v}_{i+1} &= \tilde{B}\tilde{v}_i - \tilde{\beta}_{ii} \tilde{v}_i - \tilde{\beta}_{i-1,i} \tilde{v}_{i-1} - \ldots - \tilde{\beta}_{1i} \tilde{v}_1, \end{aligned} \tag{3.4}$$

where $i \in \{1, 2, 3, \ldots, m-1\}$ and $\{\beta_{ij}\}$, $\{\tilde{\beta}_{ij}\}$ are scalars to be determined. We assume that for $i < m$, $\mathcal{K}^i(B, v_1)$ is not an invariant subspace of B (similarly, $\mathcal{K}^i(\tilde{B}, \tilde{v}_1)$ is not an invariant subspace of $\tilde{B}$ for $i < m$). We can rewrite the first equation of (3.4) as follows:

$$Bv_i = \beta_{1i}v_1 + \beta_{2i}v_2 + \ldots + \beta_{i-1,i}v_{i-1} + \beta_{ii}v_i + \beta_{i+1,i}v_{i+1}.$$

Combining these equations, for $i \in \{1, 2, 3, \ldots, m-1\}$, into matrix form we get

$$B[v_1\ v_2 \ldots v_{m-1}] = [v_1\ v_2 \ldots v_{m-2}\ v_{m-1}\ v_m] \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \cdots & \beta_{1,m-1} \\ \beta_{21} & \beta_{22} & \beta_{23} & \cdots & \beta_{2,m-1} \\ 0 & \beta_{32} & \beta_{33} & \cdots & \beta_{3,m-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \beta_{m-1,m-2} & \beta_{m-1,m-1} \\ 0 & \ldots & 0 & 0 & \beta_{m,m-1} \end{bmatrix},$$

or

$$BV_{m-1} = V_m \underline{H}_{m-1},$$

where $\underline{H}_{m-1}$ is $m \times (m-1)$ upper Hessenberg matrix. This result holds for the running index i as well. That is, for $i \in \{1, 2, 3, \ldots, m-1\}$ we have the following general relations:

$$\begin{aligned} BV_i &= V_{i+1}\underline{H}_i, \\ \tilde{B}\tilde{V}_i &= \tilde{V}_{i+1}\underline{\tilde{H}}_i. \end{aligned} \tag{3.5}$$

The scalars $\{\beta_{ij}\}$ and $\{\tilde{\beta}_{ij}\}$ are determined by a choice of constraints. One option is to enforce that the columns of V_i (and $\tilde{V}_i$) are orthonormal vectors (as in the Arnoldi algorithm). Another option, as in the bi-Lanczos algorithm, is to enforce[1]

$$V_i \perp_b \tilde{V}_i, \quad ||v_i|| = 1, \quad \text{and} \quad (v_i, \tilde{v}_i) = 1,$$

[1] As mentioned in the previous section, for this dissertation we assume breakdowns do not happen. Hence , $(\tilde{v}_i, v_i) \neq 0$.

or

$$\tilde{V}_i^* V_i = I \quad \text{and} \quad ||v_i|| = 1. \tag{3.6}$$

If $\tilde{B} = B^*$, then (3.5) and (3.6) lead to the bi-Lanczos relations (3.3), which consist of three-term recurrences. Our goal here is to relax the condition $\tilde{B} = B^*$ and still obtain short-term recurrences.

Theorem 3.1. *Let $B, \tilde{B} \in \mathbb{C}^{n\times n}$ and conditions (a) – (c) hold,*

(a) $B - \tilde{B}^* = \tilde{F}_k \tilde{C}_k^* - C_k F_k^*, \quad$ *where* $\quad C_k, \tilde{C}_k, F_k, \tilde{F}_k \in \mathbb{C}^{n\times k}$,

(b) $\forall x : Bx \perp \tilde{C}_k, \quad \forall \tilde{x} : \tilde{B}\tilde{x} \perp C_k$,

(c) $v_1 \perp \tilde{C}_k, \quad \tilde{v}_1 \perp C_k$.

If (3.6) is used to compute the scalars in (3.5), then $\beta_{ij} = 0$ and $\tilde{\beta}_{ij} = 0$ for $j > i+1$, leading to the following three-term recurrences:

$$\beta_{i+1,i} v_{i+1} = Bv_i - \beta_{ii} v_i - \beta_{i-1,i} v_{i-1},$$
$$\tilde{\beta}_{i+1,i} \tilde{v}_{i+1} = \tilde{B}\tilde{v}_i - \tilde{\beta}_{ii} \tilde{v}_i - \tilde{\beta}_{i-1,i} \tilde{v}_{i-1},$$

for $i \in \{1, 2, 3, \ldots, m-1\}$.

Proof. Using (b) and (c) we can show that

$$C_k^* \tilde{V}_i = 0 \quad \text{and} \quad \tilde{C}_k^* V_i = 0. \tag{3.7}$$

We show $C_k^* \tilde{V}_i = 0$ by induction. One can similarly show that $\tilde{C}_k^* V_i = 0$. Clearly $C_k^* \tilde{v}_1 = 0$ using (c). Let $C_k^* \tilde{v}_l = 0$ for $l = \{1, 2, \ldots, i\}$, and consider the case $l = i+1$. From (3.4) we know that

$$\tilde{\beta}_{i+1,i} \tilde{v}_{i+1} = \tilde{B}\tilde{v}_i - \tilde{\beta}_{ii} \tilde{v}_i - \tilde{\beta}_{i-1,i} \tilde{v}_{i-1} - \ldots - \tilde{\beta}_{1i} \tilde{v}_1.$$

$C_k^* \tilde{v}_{i+1} = 0$ since $C_k^* \tilde{B} \tilde{v}_i = 0$ using (b) and $\tilde{\beta}_{li} C_k^* \tilde{v}_l = 0$ for $l \in \{1, 2, \ldots, i\}$ by the induction

hypothesis[2]. This proves (3.7). Multiplying both sides of the first equation in (3.5) by $\tilde{V}_i^*$ and using (3.6) we get

$$\tilde{V}_i^* B V_i = H_i.$$

Substituting (a) in the above equation leads to

$$\begin{aligned} &\tilde{V}_i^* \left(\tilde{B}^* + \tilde{F}_k \tilde{C}_k^* - C_k F_k^* \right) V_i = H_i \qquad \Longleftrightarrow \\ &\tilde{V}_i^* \tilde{B}^* V_i + \tilde{V}_i^* \tilde{F}_k \tilde{C}_k^* V_i - \tilde{V}_i^* C_k F_k^* V_i = H_i. \end{aligned}$$

Using (3.7) we get

$$\begin{aligned} &\tilde{V}_i^* \tilde{B}^* V_i = H_i \quad \Longleftrightarrow \\ &(\tilde{B}\tilde{V}_i)^* V_i = H_i. \end{aligned}$$

Finally, using the second equation of (3.5) and (3.6) in the above equation gives

$$\tilde{H}_i^* = H_i.$$

This implies both H_i and $\tilde{H}_i$ are tridiagonal matrices, and hence $\beta_{ij} = 0$ and $\tilde{\beta}_{ij} = 0$ for $j > i + 1$. □

Augmented Bi-Lanczos and Solution Update

In RBiCG, we use the matrix U to define the primary system recycle space, and compute $C = A^{(j+1)}U$, where U is derived from an approximate right invariant subspace of $A^{(j)}$ and j denotes the index of the linear system in the sequence of linear systems. Similarly, we use the matrix $\tilde{U}$ to define the dual system recycle space, and compute $\tilde{C} = A^{(j+1)*}\tilde{U}$, where $\tilde{U}$ is derived from an approximate left invariant subspace of $A^{(j)}$. U and $\tilde{U}$ are computed such that C and $\tilde{C}$ are bi-orthogonal (see below for the motivation). The number of vectors

[2]We need $\tilde{\beta}_{i+1,i} \neq 0$. This is ensured by our earlier assumption that $\mathcal{K}^i(\tilde{B}, \tilde{v}_1)$ is not an invariant subspace of $\tilde{B}$ for $i < m$.

selected for recycling is denoted by k, and hence, U, $\tilde{U}$, C, and $\tilde{C} \in \mathbb{C}^{n\times k}$. 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

$$[U\ V_i] \perp_b \left[\tilde{U}\ \tilde{V}_i\right] \tag{3.8}$$

or

$$[C\ V_i] \perp_b \left[\tilde{C}\ \tilde{V}_i\right]. \tag{3.9}$$

We chose to implement the bi-orthogonality relation given by (3.9), because this leads to simpler algebra and hence a more efficient algorithm. It also has the advantage that the RBiCG algorithm has a form similar to the standard BiCG algorithm. $C \perp_b \tilde{C}$ is easy to implement when computing the recycle space (discussed later in this section). As in the BiCG algorithm, we let $V_i = [v_1\ v_2\ \ldots\ v_i]$ and $\tilde{V}_i = [\tilde{v}_1\ \tilde{v}_2\ \ldots\ \tilde{v}_i]$. Using the scaling (3.2), we initialize Lanczos vectors as

$$v_1 = \frac{\left(I - C\mathcal{D}_c^{-1}\tilde{C}^*\right) r_0}{\left|\left|\left(I - C\mathcal{D}_c^{-1}\tilde{C}^*\right) r_0\right|\right|}, \quad \tilde{v}_1 = \frac{\left(I - \tilde{C}\mathcal{D}_c^{-1}C^*\right) \tilde{r_0}}{\left(v_1, \left(I - \tilde{C}\mathcal{D}_c^{-1}C^*\right) \tilde{r_0}\right)}. \tag{3.10}$$

Here $\mathcal{D}_c = \tilde{C}^*C$ is a diagonal matrix (implied by $C \perp_b \tilde{C}$). In computing the recycle space, we enforce $\mathcal{D}_c$ to have positive, real coefficients. The $(i+1)$-th Lanczos vectors are given by

$$\begin{aligned} \gamma v_{i+1} &= Av_i - V_i\tau - C\rho \perp \left[\tilde{C}\ \tilde{V}_i\right], \\ \tilde{\gamma}\tilde{v}_{i+1} &= A^*\tilde{v}_i - \tilde{V}_i\tilde{\tau} - \tilde{C}\tilde{\rho} \perp [C\ V_i], \end{aligned} \tag{3.11}$$

where γ, $\tilde{\gamma}$, τ, $\tilde{\tau}$, ρ, and $\tilde{\rho}$ are to be determined. The computation of the $(i+1)$-th Lanczos vector for the primary system now requires the i-th and $(i-1)$-th Lanczos vectors and C (see [3]). This gives a $(3+k)$-term recurrence, where k is the number of columns of C. Similarly, we get a $(3+k)$-term recurrence for computing the Lanczos vectors for the dual system. We refer to this pair of $(3+k)$-term recurrences as the augmented bi-Lanczos

relations, and they are given by

$$\begin{aligned}(I - C\hat{C}^*)AV_i &= V_{i+1}\underline{T}_i,\\ (I - \tilde{C}\check{C}^*)A^*\tilde{V}_i &= \tilde{V}_{i+1}\underline{\tilde{T}}_i,\end{aligned} \tag{3.12}$$

where

$$\begin{aligned}\hat{C} = \left[\frac{\tilde{c}_1}{c_1^*\tilde{c}_1}\ \frac{\tilde{c}_2}{c_2^*\tilde{c}_2}\ \cdots\ \frac{\tilde{c}_k}{c_k^*\tilde{c}_k}\right] &= \tilde{C}\mathcal{D}_c^{-1},\\ \check{C} = \left[\frac{c_1}{\tilde{c}_1^*c_1}\ \frac{c_2}{\tilde{c}_2^*c_2}\ \cdots\ \frac{c_k}{\tilde{c}_k^*c_k}\right] &= C\mathcal{D}_c^{-1}.\end{aligned} \tag{3.13}$$

Lemma 3.1. *Given the above setup,*

(a)

$$\begin{bmatrix}\tilde{C}^*\\ \tilde{V}_i^*\end{bmatrix}[C\ V_{i+1}] = \begin{bmatrix}I_{k\times k} & 0 & 0\\ 0 & I_{i\times i} & 0\end{bmatrix}.$$

(b)

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

(c)

$$r_0 = [C\ V_{i+1}]\begin{bmatrix}\hat{C}^*r_0\\ \zeta e_1\end{bmatrix}, \quad \textit{where} \quad \zeta = ||(I - C\hat{C}^*)r_0||.$$

Proof. Part (a) follows directly from (3.9). Using $AU = C$ and (3.12) proves (b). For (c) rewrite $r_0 = C\hat{C}^*r_0 + \left(I - C\hat{C}^*\right)r_0$. Now use the fact that v_1 is the first column of V_{i+1} (as defined earlier in this section) and (3.10). □

Theorem 3.2. *Let* $v_1 = \eta(I - C\mathcal{D}_c^{-1}\tilde{C}^*)r_0$, $\tilde{v}_1 = \tilde{\eta}(I - \tilde{C}\mathcal{D}_c^{-1}C^*)\tilde{r}_0$, $B = (I - C\mathcal{D}_c^{-1}\tilde{C}^*)A$, *and* $\tilde{B} = (I - \tilde{C}\mathcal{D}_c^{-1}C^*)A^*$ *where* η, $\tilde{\eta}$ *are scalars and* C, $\tilde{C} \in \mathbb{C}^{n\times k}$ *s.t.* $\mathcal{D}_c = \tilde{C}^*C$ *is a diagonal matrix with positive, real coefficients. If (3.6) is used to compute the scalars in*

(3.5), then $\beta_{ij} = 0$ *and* $\tilde{\beta}_{ij} = 0$ *for* $j > i + 1$*, leading to the following short-term recurrences:*

$$\begin{aligned}\beta_{i+1,i} v_{i+1} &= Bv_i - \beta_{ii} v_i - \beta_{i-1,i} v_{i-1},\\ \tilde{\beta}_{i+1,i} \tilde{v}_{i+1} &= \tilde{B}\tilde{v}_i - \tilde{\beta}_{ii} \tilde{v}_i - \tilde{\beta}_{i-1,i} \tilde{v}_{i-1},\end{aligned}$$

for $i \in \{1, 2, 3, \ldots, m-1\}$.

Proof. We show that conditions (a) – (c) of Theorem 3.1 are satisfied. This demonstrates that augmented bi-Lanczos is a special case of generalized bi-Lanczos. We have $B, \tilde{B} \in \mathbb{C}^{n\times n}$ such that

$$\begin{aligned} B - \tilde{B}^* &= A - C\mathcal{D}_c^{-1}\tilde{C}^* A - A + AC\mathcal{D}_c^{-1}\tilde{C}^* \\ &= \left(AC\mathcal{D}_c^{-1}\right)\tilde{C}^* - C\left(A^*\tilde{C}\mathcal{D}_c^{-1}\right)^*. \end{aligned}$$

Defining $F = A^*\tilde{C}\mathcal{D}_c^{-1}$ and $\tilde{F} = AC\mathcal{D}_c^{-1}$ we get

$$B - \tilde{B}^* = \tilde{F}\tilde{C}^* - CF^* \quad \text{where} \quad C, \tilde{C}, F, \tilde{F} \in \mathbb{C}^{n\times k}.$$

Hence (a) is satisfied. For any $\tilde{x}$ consider the following:

$$\begin{aligned} C^*\tilde{B}\tilde{x} &= C^*(I - \tilde{C}\mathcal{D}_c^{-1}C^*)A^*\tilde{x} \\ &= (C^* - \mathcal{D}_c\mathcal{D}_c^{-1}C^*)A^*\tilde{x} = 0. \end{aligned}$$

Similarly, for any x consider the following:

$$\begin{aligned} \tilde{C}^*Bx &= \tilde{C}^*(I - C\mathcal{D}_c^{-1}\tilde{C}^*)Ax \\ &= (\tilde{C}^* - \mathcal{D}_c\mathcal{D}_c^{-1}\tilde{C}^*)Ax = 0. \end{aligned}$$

Hence (b) is satisfied. Similarly, for v_1 and $\tilde{v}_1$ chosen in the theorem $\tilde{C}^*v_1 = 0$ and $C^*\tilde{v}_1 = 0$. Hence, (c) is satisfied. □

Next, we derive the solution update for RBiCG. We focus on the primary system here. The

derivations for the dual system are analogous. We replace the i-th solution iterates of the standard BiCG algorithm by

$$x_i = x_0 + Uz_i + V_i y_i, \quad \tilde{x}_i = \tilde{x}_0 + \tilde{U}\tilde{z}_i + \tilde{V}_i\tilde{y}_i, \tag{3.14}$$

in the RBiCG algorithm. With recycling, the bi-orthogonality condition (3.9) defines the Petrov-Galerkin condition,

$$r_i = r_0 - AUz_i - AV_i y_i \perp \left[\tilde{C}\ \tilde{V}_i\right], \quad \tilde{r}_i = \tilde{r}_0 - A^*\tilde{U}\tilde{z}_i - A^*\tilde{V}_i\tilde{y}_i \perp [C\ V_i]\,. \tag{3.15}$$

The computation of z_i and y_i can be implemented more efficiently than (3.15) suggests. Using Lemma 3.1 and (3.15), we get

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

Here T_i is an $i \times i$ tridiagonal matrix obtained by removing the last row of $\underline{T}_i$. Solving for y_i and z_i from this and substituting in (3.14) leads to the following solution update:

$$x_i = x_0 + U\hat{C}^* r_0 + (I - U\hat{C}^* A)V_i y_i,$$

where y_i is obtained from solving $T_i y_i = \zeta e_1$. All computations here are done with matrix-vector products and $U\hat{C}^* A$ is not computed explicitly.

We introduce a slight change of notation to make future derivations simpler. Let x_{-1}, $\tilde{x}_{-1}$ be the initial guesses and $r_{-1} = b - Ax_{-1}$, $\tilde{r}_{-1} = \tilde{b} - A^*\tilde{x}_{-1}$ the corresponding initial residuals. We define

$$\begin{aligned} x_0 &= x_{-1} + U\hat{C}^* r_{-1}, & r_0 &= (I - C\hat{C}^*)r_{-1}, \\ \tilde{x}_0 &= \tilde{x}_{-1} + \tilde{U}\check{C}^*\tilde{r}_{-1}, & \tilde{r}_0 &= (I - \tilde{C}\check{C}^*)\tilde{r}_{-1}, \end{aligned} \tag{3.16}$$

and follow this convention for x_0, $\tilde{x}_0$, r_0, and $\tilde{r}_0$ for the rest of the chapter. Let

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

As in the standard BiCG algorithm, an LDU decomposition (without pivoting) of T_i might not always exist. We can avoid this breakdown in the same way as done for BiCG (see Section 3.1). The two-term recurrence for the solution update of the primary system is now given by

$$x_i = x_{i-1} + \varphi_{i,i} G_i e_i \quad \text{for} \quad i \geq 1,$$

where $\varphi_{i,i}$ is the last entry of the vector φ_i and x_0 is given by (3.16). Note that we never compute any explicit matrix inverse. The matrices under consideration, D_i, L_i, and R_i, are diagonal, lower triangular, and upper triangular respectively. This two-term recurrence can be simplified such that T_i is not needed explicitly. For further simplifications, we follow steps similar to the ones used in the derivation of BiCG [51]. Algorithm 3.2 outlines RBiCG that includes the recycle space into the search space. The algorithmic improvements to make the code faster are not shown here. For a detailed algebraic derivation and more improvements see [3].

Computing a Recycle Space

We need Lanczos vectors (v_i and $\tilde{v}_i$) and tridiagonal matrices (T_i and $\tilde{T}_i$) explicitly to build the recycle space. As for the initial Lanczos vectors in (3.10), we use the scaling (3.2) to compute the subsequent Lanczos vectors

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

Algorithm 3.2. *RBiCG*
1. Given U and $\tilde{U}$ compute $\check{C}$ and $\hat{C}$ using (3.13). If U and $\tilde{U}$ are not available, then initialize U, $\tilde{U}$, $\check{C}$, and $\hat{C}$ to empty matrices.
2. Choose x_{-1}, $\tilde{x}_{-1}$ and compute x_0, $\tilde{x}_0$, r_0, and $\tilde{r}_0$ using (3.16).
3. **if** $(r_0, \tilde{r}_0) = 0$ **then** initialize $\tilde{x}_{-1}$ to a random vector.
4. Set $p_0 = 0$, $\tilde{p}_0 = 0$, and $\beta_0 = 0$. Choose the convergence tolerance (`tol`), and the maximum number of iterations (`itn`).
5. **for** $i = 1 \ldots$ `itn` **do**
◇ $p_i = r_{i-1} + \beta_{i-1} p_{i-1}$.
◇ $\tilde{p}_i = \tilde{r}_{i-1} + \bar{\beta}_{i-1} \tilde{p}_{i-1}$.
◇ $q_i = (I - U\hat{C}^* A) p_i$.
◇ $\tilde{q}_i = (I - \tilde{U}\check{C}^* A^*) \tilde{p}_i$.
◇ $\alpha_i = (\tilde{r}_{i-1}, r_{i-1})/(\tilde{p}_i, Aq_i)$.
◇ $x_i = x_{i-1} + \alpha_i q_i$.
◇ $\tilde{x}_i = \tilde{x}_{i-1} + \bar{\alpha}_i \tilde{q}_i$.
◇ $r_i = r_{i-1} - \alpha_i A q_i$.
◇ $\tilde{r}_i = \tilde{r}_{i-1} - \bar{\alpha}_i A^* \tilde{q}_i$.
◇ **if** $||r_i|| \leq$ `tol` and $||\tilde{r}_i|| \leq$ `tol` **then break**.
◇ $\beta_i = (\tilde{r}_i, r_i)/(\tilde{r}_{i-1}, r_{i-1})$.
6. **end for**.

T_i can be computed using residual's and scalar's of RBiCG iteration as follows [3, 1]:

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

Using (3.2) and (3.12), one can show $\tilde{T}_i = T_i^*$.

Instead of using all the Lanczos vectors to update the recycle space, we update the recycle space periodically. This strategy keeps the memory requirements modest [85]. 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 updates. From here on, let V_j and

$\tilde{V}_j$ contain the Lanczos vectors generated during the j^{th} cycle,

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

Also, let

$$\Upsilon_j = \begin{bmatrix} v_{(j-1)s} & V_j & v_{js+1} \end{bmatrix}, \quad \tilde{\Upsilon}_j = \begin{bmatrix} \tilde{v}_{(j-1)s} & \tilde{V}_j & \tilde{v}_{js+1} \end{bmatrix},$$

where $v_{(j-1)s}$ and $\tilde{v}_{(j-1)s}$ are the last Lanzos vectors from the previous cycle, and v_{js+1} and $\tilde{v}_{js+1}$ are the first Lanzos vectors from the next cycle. The augmented bi-Lanczos relations for the j^{th} cycle are now given by

$$\begin{aligned} (I - C\hat{C}^*)AV_j &= \Upsilon_j \Gamma_j, \\ (I - \tilde{C}\check{C}^*)A^*\tilde{V}_j &= \tilde{\Upsilon}_j \, \tilde{\Gamma}_j, \end{aligned} \tag{3.17}$$

where Γ_j, $\tilde{\Gamma}_j \in \mathbb{C}^{(s+2)\times s}$ are T_j, $\tilde{T}_j$, respectively, 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 $\tilde{v}_{js+1}$).

The discussion in this paragraph concerns only the primary system. However, an analogous discussion applies to the dual system. Let U define the recycle space available from the previous linear system and U_{j-1} the recycle space generated at the end of cycle $(j-1)$ for the current linear system. We want to obtain an improved 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. At the end of solving the current linear system the final U_j will be U for the next linear system. There are many options for selecting U_j, all of which have similar performance and formulation [85]. For simplicity, we build U_j from range($[U_{j-1}\ V_j]$).

Based on the choices discussed in the previous two paragraphs, we first define few matrices and then develop the generalized eigenvalue problem whose solution gives the invariant

subspace. Let

$$\Phi_j = [U_{j-1}\ V_j]\,, \quad \Psi_j = [C\ C_{j-1}\ \Upsilon_j]\,, \quad H_j = \begin{bmatrix} 0 & B_j \\ I & 0 \\ 0 & \Gamma_j \end{bmatrix},$$
$$\tilde{\Phi}_j = \left[\tilde{U}_{j-1}\ \tilde{V}_j\right], \quad \tilde{\Psi}_j = \left[\tilde{C}\ \tilde{C}_{j-1}\ \tilde{\Upsilon}_j\right], \quad \tilde{H}_j = \begin{bmatrix} 0 & \tilde{B}_j \\ I & 0 \\ 0 & \tilde{\Gamma}_j \end{bmatrix},$$

where $C_{j-1} = AU_{j-1}$, $B_j = \hat{C}^* A V_j$, $\tilde{C}_{j-1} = A^* \tilde{U}_{j-1}$, and $\tilde{B}_j = \check{C}^* A^* \tilde{V}_j$. Then, augmented bi-Lanczos relations (3.17) lead to

$$\begin{aligned} A\Phi_j &= \Psi_j H_j, \\ A^* \tilde{\Phi}_j &= \tilde{\Psi}_j \tilde{H}_j. \end{aligned} \tag{3.18}$$

In GCRO-DR [66] and RMINRES [85], approximate invariant subspace has been successfully used as recycle space. It has also been demonstrated that using the approximate invariant subspace corresponding to small eigenvalues (in absolute value) is effective [63, 66, 85]. For RBiCG we follow the same strategy. We use harmonic Ritz vectors [75], with respect to the current Krylov subspace, to approximate invariant subspace cheaply.

In RMINRES [85], 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, as also suggested in [12]. This leads to simpler algebra and cheaper computations later. Let (λ, u) denote an harmonic Ritz pair of A. Then, we derive λ and $u \in$ range(Φ_j) from the condition

$$(Au - \lambda u) \perp \text{range}\left(A^* \tilde{\Phi}_j\right). \tag{3.19}$$

Using $u = \Phi_j w$ and substituting (3.18) in (3.19) gives the following generalized eigenvalue

problem:

$$\tilde{H}_j^* \tilde{\Psi}_j^* \Psi_j H_j w = \lambda \tilde{H}_j^* \tilde{\Psi}_j^* \Phi_j w. \tag{3.20}$$

Hence, we have $U_j = \Phi_j W_j$, where columns of W_j are the k right eigenvectors corresponding to the eigenvalues closet to the origin. Similarly, $\tilde{U}_j = \tilde{\Phi}_j \tilde{W}_j$, where columns of $\tilde{W}_j$ are the corresponding k left eigenvectors (for an analogous derivation of the dual system recycle space see [3]). The first pair of systems in our sequence of dual linear systems require special attention since there is no recycle space available at the start. We refer the reader to [3] for this small variation.

To implement the bi-orthogonality relation (3.9), we need $C \perp_b \tilde{C}$. We also need $\mathcal{D}_c = \tilde{C}^* C$ nonsingular in (3.13). We enforce these properties at the end of each cycle as follows. After solving the generalized eigenvalue problem (3.20), initially we choose $U_j = \Phi_j W_j, \tilde{U}_j = \tilde{\Phi}_j \tilde{W}_j$ and $C_j = AU_j, \tilde{C}_j = A^* \tilde{U}_j$. We compute the singular value decomposition (SVD)

$$\tilde{C}_j^* C_j = M_j \Sigma_j N_j^*, \tag{3.21}$$

such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq 0$. Given some tolerance $\texttt{tol} > 0$, we pick p such that $\sigma_p \geq \texttt{tol} > \sigma_{p+1}$ (with $p = k$ possible), and redefine $M_j = [m_1, \ldots, m_p], N_j = [n_1, \ldots, n_p]$ where m_i, n_i are left and right singular vectors corresponding to σ_i. Next, we redefine

$$\begin{aligned} U_j &= \Phi_j W_j N_j = [U_{j-1}\ V_j] W_j N_j, \quad \tilde{U}_j = \tilde{\Phi}_j \tilde{W}_j M_j = [\tilde{U}_{j-1}\ \tilde{V}_j] \tilde{W}_j M_j, \\ C_j &= AU_j = A[U_{j-1}\ V_j] W_j N_j, \qquad \tilde{C}_j = A^* \tilde{U}_j = A^* [\tilde{U}_{j-1}\ \tilde{V}_j] \tilde{W}_j M_j. \end{aligned} \tag{3.22}$$

It follows from this construction that $\tilde{C}_j^* C_j$ is diagonal and nonsingular[3].

[3]Theoretically, all singular values can be zero leading to no recycle space.

Cost Analysis

In this section, we describe the extra work done because of recycling in RBiCG. We have split the cost in three parts and only report the terms with $O(n)$ cost. The other terms have cost proportional to combinations of s and k, both of which are much smaller than n. First, the extra cost at every iterative step is $(8kn + 2n)$ flops. This cost is related to extra orthogonalizations because of using the recycle space.

Second, we look at the cost at the end of each cycle. This cost is related to computing the recycle space. Here, computing the matrices $\tilde{\Psi}_j^* \Psi_j$ and $\tilde{\Psi}_j^* \Phi_j$ in (3.20) is one of the most expensive part. These matrices can be constructed efficiently as follows:

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

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

where $\underline{\overline{I}}$ is the $s \times s$ identity matrix with an extra row of zeros at the top and at the bottom. Note that $\mathcal{D}_c$ and Σ_{j-1} are easily available. Remaining blocks of both $\tilde{\Psi}_j^* \Psi_j$ and $\tilde{\Psi}_j^* \Phi_j$ can be simplified further using recurrences. We present the final expressions and cost in Table 3.1. Therefore, the extra cost at the end of each cycle is $(14k^2n + 6kns + 16kn + 4n)$ flops.

Third and last, the extra cost once per linear system is $(10k^2n + 28kn + 14n)$ flops. This cost is related to the initial projection. Note that s and k are much smaller than n. For recycling to be beneficial, the savings in iterations should be high. In the next three sections, we show up to 70% savings in iteration count. We also show that solving a model reduction problem without recycling takes about 50% more time than with recycling.

Initial Block	Simplified Block [3]	Cost (flops)
$\tilde{C}^* C_{j-1}$	$\begin{bmatrix} \tilde{C}^* C_{j-2} & \mathcal{D}_c B_{j-1} \end{bmatrix} W_{j-1} N_{j-1}$	$2k^3 + 2k^2 s + ks$
$\tilde{C}^*_{j-1} C$	$M^*_{j-1} \tilde{W}^*_{j-1} \begin{bmatrix} \tilde{C}^*_{j-2} C \\ \tilde{B}^*_{j-1} \mathcal{D}_c \end{bmatrix}$	$2k^3 + 2k^2 s + ks$
$\tilde{C}^*_{j-1} \Upsilon_j$	$M^*_{j-1} \tilde{W}^*_{j-1} \begin{bmatrix} 0 \\ \tilde{\Gamma}^*_{j-1} \tilde{\Upsilon}^*_{j-1} \Upsilon_j \end{bmatrix}$	$2s^3 + 2k^2 s + 2ks^2 + 4k^2 + 4s^2 + 4ks + 8s$
$\tilde{\Upsilon}^*_j C_{j-1}$	$\begin{bmatrix} 0 & \tilde{\Upsilon}^*_j \Upsilon_{j-1} \Gamma_{j-1} \end{bmatrix} W_{j-1} N_{j-1}$	$2s^3 + 2k^2 s + 2ks^2 + 4k^2 + 4s^2 + 4ks + 8s$
$\tilde{C}^* U_{j-1}$	$\begin{bmatrix} \tilde{C}^* U_{j-2} & 0 \end{bmatrix} W_{j-1} N_{j-1}$	$2k^3 + 2k^2 s$
$\tilde{C}^*_{j-1} U_{j-1}$	$M^*_{j-1} \tilde{W}^*_{j-1} \begin{bmatrix} \tilde{C}^*_{j-2} U_{j-2} & \tilde{C}^*_{j-2} V_{j-1} \\ \tilde{V}^*_{j-1} C_{j-2} & \tilde{T}_{j-1} \end{bmatrix} W_{j-1} N_{j-1}$	$4k^3 + 2s^3 + 6k^2 s + 2ks^2 + 4s^2$
$\tilde{C}^*_{j-1} V_j$	Subset of $\tilde{C}^*_{j-1} \Upsilon_j$	–
$\tilde{\Upsilon}^*_j U_{j-1}$	–	$2kns + 2kn$

Table 3.1: Simplification and cost of blocks of $\tilde{\Psi}^*_j \Psi_j$ and $\tilde{\Psi}^*_j \Phi_j$. For the first block, we describe the advantage of writing in the above form. Computing $\tilde{C}^* C_{j-1}$ by direct multiplication has a $O(n)$ cost, which is expensive. The simplified term in the second column also has $\tilde{C}^* C_{j-2}$, however, this is available from the previous cycle. $\mathcal{D}_c$ is a diagonal matrix independent of the cycle, and B_{j-1} is computed during the iterations. Finally, the matrix-matrix product $\begin{bmatrix} \tilde{C}^* C_{j-2} & \mathcal{D}_c B_{j-1} \end{bmatrix} W_{j-1} N_{j-1}$ does not involve any $O(n)$ operation.

3.3 RBiCG for a Convection-Diffusion Example

We test RBiCG on the linear system obtained by finite difference discretization of the partial differential equation [82]

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

with $\mathcal{A}$ as shown in Figure 3.1 (a), $\mathcal{B}(x,y) = 2e^{2(x^2+y^2)}$, and $\mathcal{F} = 0$ everywhere except in a small square in the center where $\mathcal{F} = 100$ (see Figure 3.1 (a)). The domain is $(0,1) \times (0,1)$ with Dirichlet boundary conditions

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

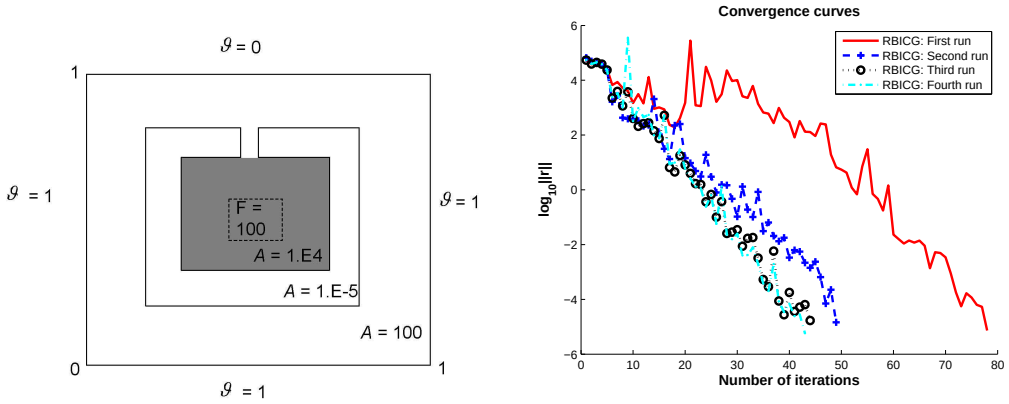
We test our method on this convection-diffusion problem because of the following three reasons. First, this model problem provides well-understood test cases that are easy to analyze further. Second, convection-diffusion problems are ubiquitous. Few examples that lead to a convection-diffusion problem are as follows: a fixed-point linearization of the Navier-Stokes equations (Oseen's problem); heat flow in a medium with transport; chemicals in air flow that diffuse, react, and are transported etc. Third, any such common physical problem would lead to a potential model reduction problem.

We use the second order central difference scheme for discretization with mesh width $h = 1/64$. This leads to a nonsymmetric linear system of 3969 unknowns. The convergence is same for a problem that is four times larger. For the sake of further analysis, we give results for this smaller system size. The primary system right-hand side comes from the PDE. We take vector of all zeros as the dual system right-hand side [83]. Note that the primary system is the only one we are concerned about here.

The dual linear systems are solved four times with RBiCG. The recycle space generated

during the first run is used for solving the same dual systems a second time. The recycle space is further improved during the second run and used in solving the dual systems a third time, and so on. This is the "ideal" case for recycling; it excludes the effects of right-hand sides having slightly different eigenvector decompositions [66]. For this experiment we take $s = 40$ and $k = 10$. These are chosen based on experience with other recycling algorithms [66, 85]. The relative tolerance for RBiCG is taken as 10^{-8}. Initial guess is a vector of all ones. The linear systems are split-preconditioned by a Crout version of the ILUT preconditioner with a drop tolerance of 0.05 [70]. The generated recycle space pertains to the preconditioned linear systems.

Figure 3.1 (b) shows the benefit of using RBiCG for solving the primary system multiple times. For the second run, the savings in iterations is around 35%. The convergence gets better with each run. Next, we present a brief analysis of the generated recycle space. In Table 3.2 we give the cosines of principal angles between the recycle space and the invariant subspace spanned by eight eigenvectors associated with the eigenvalues of smallest magnitude. As for the recycle space, the invariant subspace is computed for the preconditioned operator. For the primary system, we use the invariant subspace spanned by right eigenvectors. For the dual system, we use the invariant subspace spanned by left eigenvectors. We want the angles to tend to zero, and so the cosines should tend to one. The table shows that with more runs, we accurately approximate larger subspace of the invariant subspace. The primary system recycle space captures three, six, and seven right eigenvectors during Runs 1, 2, and 3 respectively. The dual system recycle space captures four, five, and six left eigenvectors during Runs 1, 2, and 3 respectively. Therefore, we see faster convergence for every new run.



(a) Coefficients for the PDE.

(b) Using RBiCG to solve the primary system multiple times. The length of the cycle is 40, the number of vectors selected for recycling are 10, and the preconditioner drop tolerance is 0.05.

Figure 3.1: Using RBiCG for a convection-diffusion type PDE.

Primary System			Dual System		
Start of Run 2	Start of Run 3	Start of Run 4	Start of Run 2	Start of Run 3	Start of Run 4
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.9896	1.0000	1.0000	0.9950	1.0000	1.0000
0.3832	1.0000	1.0000	0.9884	1.0000	1.0000
0.1452	0.9983	1.0000	0.7864	0.9844	1.0000
0.0988	0.9437	0.9970	0.6070	0.9206	0.8141
0.0300	0.1869	0.9567	0.4749	0.4118	0.4721

Table 3.2: Analysis of the recycle space for the convection-diffusion type PDE. The columns list the cosine of principal angles between the recycle space and the invariant subspace spanned by eight eigenvectors associated with the eigenvalues of smallest magnitude. For the primary system, we use the invariant subspace spanned by right eigenvectors. For the dual system, we use the invariant subspace spanned by left eigenvectors. The convergence curves, corresponding to different run's of the primary system, are shown in Figure 3.1 (b).

3.4 RBiCG for Model Reduction

In Section 2.1 we discussed the advantages of using BiCG in solving linear systems arising in IRKA. We also briefly presented motivation of using RBiCG. This discussion further supports usage of RBiCG for IRKA. RBiCG not only naturally results from the need for solving dual systems and obvious recycling framework in IRKA, but also provides a backward error framework. We summarize this observations in the next result, which is a corollary to Theorem 4.1 in [15].

Corollary 3.1. *let $\tilde{G}_r(s)$ be obtained by IRKA where linear systems are solved using RBiCG. Then, $\tilde{G}_r(s)$ satisfies the necessary conditions for $\mathcal{H}_2$ optimality for a near-by full-order model $\tilde{G}(s) = c^T(sE - (A + F_{2r}))^{-1}b$ where F_{2r} is the rank-$2r$ matrix defined in (2.7).*

Proof. Since RBiCG is still performed in a Petrov-Galerkin framework, follow same steps as in the proof of Theorem 4.1 in [15]. □

Next we present different strategies of using RBiCG with-in IRKA. After that we present numerical results, including problem and implementation details; recycle space analysis; and iteration count and time comparisons.

IRKA using RBiCG

IRKA usually converges rather fast [49]. Hence, after one or a few initial steps, the interpolations points from one step of IRKA to the next one do not change substantially. Moreover, for many cases, the change of the (appropriately ordered) $\{\sigma_i\}$ from one column of V_r (and W_r) to the next is also modest. Hence, recycling is typically appropriate for IRKA.

For the special case of $E = I$ in (2.1), alternative solution approaches might be advantageous, as one can solve the linear systems for multiple shifts at once [40, 44, 55, 81]. Combining these strategies with a Petrov-Galerkin framework does not seem complicated. Effective

strategies for Krylov subspace recycling for solving systems of the type, $(\sigma_i I - A)v_i = b$, for multiple shifts at once, as well as for multiple right hand sides, was discussed in [57]. For most model reduction applications, however, $E \neq I$.

There are three basic ways of recycling Krylov subspaces in IRKA. We now describe the first strategy. Let at step m of IRKA (iteration m of the while loop in Algorithm 2.1), we have the shifts $\sigma_i^{(m)}$, for $i = 1, \ldots, r$, with

$$\begin{aligned} V_r^{(m)} &= [(\sigma_1^{(m)}E - A)^{-1}b,\ \ldots,\ (\sigma_r^{(m)}E - A)^{-1}b], \\ W_r^{(m)} &= [(\sigma_1^{(m)}E - A)^{-*}c,\ \ldots,\ (\sigma_r^{(m)}E - A)^{-*}c]. \end{aligned} \tag{3.23}$$

Let at step $m+1$ of IRKA, we have the shifts $\sigma_i^{(m+1)}$, for $i = 1, \ldots, r$, with

$$\begin{aligned} V_r^{(m+1)} &= [(\sigma_1^{(m+1)}E - A)^{-1}b,\ \ldots,\ (\sigma_r^{(m+1)}E - A)^{-1}b], \\ W_r^{(m+1)} &= [(\sigma_1^{(m+1)}E - A)^{-*}c,\ \ldots,\ (\sigma_r^{(m+1)}E - A)^{-*}c]. \end{aligned}$$

One can recycle Krylov subspaces from the i^{th} column of $V_r^{(m)}$ and $W_r^{(m)}$ to the i^{th} column of $V_r^{(m+1)}$ and $W_r^{(m+1)}$. That is, from solving the pair of linear systems

$$(\sigma_i^{(m)}E - A)v_i^{(m)} = b, \quad (\sigma_i^{(m)}E - A)^* w_i^{(m)} = c,$$

to solving the pair of linear systems

$$(\sigma_i^{(m+1)}E - A)v_i^{(m+1)} = b, \quad (\sigma_i^{(m+1)}E - A)^* w_i^{(m+1)} = c,$$

where $i = 1, 2, \ldots, r$. This kind of recycling strategy is always useful since change in a particular shift from one IRKA step to the next is small.

Second, one can recycle selected Krylov subspaces across the columns of the matrices V_r and W_r. Third, the first two recycling strategies can be combined as well. We describe one such combination. Consider solving the system $(\sigma_i^{(m+1)}E - A)v_i^{(m+1)} = b$ and its dual

system. From a set of previously generated recycle spaces (distinguished by their shifts), one can pick the recycle space for which the relative change in σ is the least and less than a relative tolerance. This would ensure that the system that generated the recycle space is close to the current one. A natural pool to pick the σ defining the recycle space would be $\sigma_1^{(m)}, \ldots, \sigma_r^{(m)}, \sigma_1^{(m+1)}, \ldots, \sigma_{i-1}^{(m+1)}$. The second and third recycling strategies are useful when the shifts at an IRKA step are clustered.

For experiments of this paper, r is very small and so the shifts at any particular IRKA step are spread far apart. Hence, we follow the first strategy. That is, for every shift, we recycle Krylov subspaces from one IRKA step to the next. In general, the linear systems corresponding to the relatively large shifts converge very fast, and so recycling Krylov subspaces is not useful for them. Therefore, we carry out recycling only for selected, small shifts.

Analysis and Results

Our test dynamical system is a semi-discretized heat transfer problem for determining the optimal cooling of steel profiles [67, 16, 72]. We call this the rail model [67]. The rail model is available as a multiple-input/multiple-output (MIMO) system. Since we work with a SISO system, we pick b and c of (2.1) as the second column of the input matrix and as the transposed sixth row of the output matrix, respectively. These are our primary and dual linear system right-hand sides. The rail model is available in four sizes: 1357, 5177, 20209, and 79841, corresponding to the different mesh sizes in the discretization.

We say that IRKA has converged to the ideal shifts when the relative change in shifts is less than a certain tolerance. For these experiments, we set 10^{-6} as the relative tolerance. For the rail models, A and E of (2.1) are symmetric negative definite and symmetric positive definite (SPD), respectively. Since our shifts are real and positive at every IRKA step, the matrices for our linear systems, $(\sigma_i^{(m)} E - A)$, stay SPD at all times. Using RBiCG is advantageous here (i.e., even for SPD systems) because of the backward error result discussed in Section 2.1.

We do two sets of experiments on the rail models. The main difference between these two sets of experiments is r, the size to which we reduce the models. Reducing to different r's allows for a broader spectrum of test cases. For RBiCG, computing a recycle space is the most expensive part. It has been shown that a Krylov subspace generated by one system can be useful for multiple consecutive systems [66, 57]. Hence, we also vary the frequency of computing a recycle space between the two sets of experiments.

We implement the first recycling strategy from the previous section (for a few selected shifts). As for the convection-diffusion example, the recycling parameters s and k are chosen based on experience with other recycling algorithms [66, 85]. Based on the definition of s, if a pair of systems converges in iterations less than s, then no recycle space is generated while solving this pair. Hence, for the next pair of systems in the sequence, we use the recycle space from the last pair for which the recycle space was generated.

The relative convergence tolerance for the iterative solvers (BiCG and RBiCG) and the tolerance for constructing nonsingular $\tilde{C}_j^* C_j$ in (3.13) are all taken as 10^{-6}. The linear systems are split-preconditioned by an incomplete LU preconditioner with threshold and pivoting (ILUTP) [70]. We choose different drop tolerances for different sizes to avoid working with a very ill-conditioned matrix. The generated recycle space pertains to the preconditioned linear systems. Initial guess of the preconditioned system is the solution vector from the contiguously previous preconditioned system in the sequence. This helps in convergence because the change from a pair of systems in the sequence to the next is small. For the first IRKA step, when there are no previously solved systems, we take a vector of all zeros as the initial guess. In many cases, the initial guess is picked based on the knowledge of the system so as to avoid orthogonal initial residuals (Algorithm 3.1 Step 2; Algorithm 3.2 Step 3).

For the first set of experiments, we reduce the models to $r = 6$ and use the following initial shifts: $1.00 \times 10^{-5}, 1.38 \times 10^{-4}, 1.91 \times 10^{-3}, 2.63 \times 10^{-2}, 3.63 \times 10^{-1}$, and 5.01. We compute the recycle space at every IRKA step. The results are given in Figures 3.2 – 3.5. These results are for the primary systems (columns of V_r) at a particular IRKA step: that is, for

a specific value of m in (3.23), which is mentioned in the figure captions. Similar graphs exist for other IRKA steps. To avoid repetition, we do not present the graphs for the the dual systems (columns of W_r), which are similar. We implement recycling for the smallest two shifts. Hence, each figure has two solid curves, which correspond to systems solved without recycling, and two dashed–dotted curves, which correspond to systems solved with recycling. As discussed in the previous section, the remaining four (larger) shifts converge fast, so recycling Krylov subspaces is not useful for them. It is evident that systems that use a recycle space converge in much fewer iterations compared to systems that do not use recycling. The savings in iterations are as high as 70% per system.

Next, we analyze the recycle space generated during the first two IRKA steps for the order 5177 rail model corresponding to the smallest shift. In Table 3.3, we give the cosines of principal angles between the recycle space and the invariant subspace spanned by eight eigenvectors associated with the eigenvalues of smallest magnitude. As for the recycle space, the invariant subspace is computed for the preconditioned operator. For the primary system, we use the invariant subspace spanned by right eigenvectors. For the dual system, we use the invariant subspace spanned by left eigenvectors. We want the angles to tend to zero, and so the cosines should tend to one. Consider the results for the primary system. At the first IRKA step and end of the first cycle, we see that the recycle space captures four of the eight eigenvectors. The recycle space gets more accurate at the end of the second cycle and captures seven eigenvectors. During the second IRKA step, we have a new shift, and so the matrix changes. Therefore, at the start of the first cycle, we see a slight deterioration of the recycle space (almost negligible). This recycle space leads to the dashed curve in Figure 3.3. By the end of the first cycle (at this second IRKA step), all eight eigenvectors are captured. The results for the dual system recycle space are similar.

For the second set of experiments, we reduce the models to $r = 3$ and use the initial shifts as follows: $1.00 \times 10^{-5}, 7.08 \times 10^{-3}$, and 5.01. We compute the recycle space at every fifth IRKA step. The results are given in Table 3.4. We implement recycling for the smallest shift only. The linear systems corresponding to the two (larger) shifts converge fast, so recycling

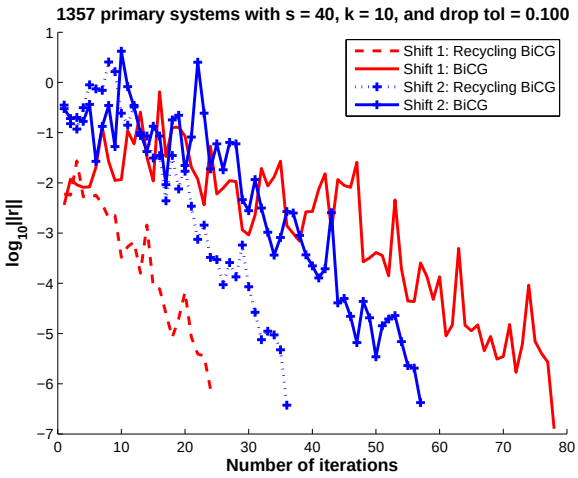


Figure 3.2: Convergence curves of ILU preconditioned RBiCG at the 3rd IRKA step for the 1357×1357 rail model. The length of the cycle is 40, the number of vectors selected for recycling are 10, and the preconditioner drop tolerance is 0.1.

Krylov subspaces is not useful for them. *Total iteration count* refers to the sum of iteration counts for solving linear systems over all shifts and all IRKA steps. *Total time* is the time in seconds required by IRKA to converge to the ideal shifts. This time includes the time for all IRKA computations as well as all linear solves (BiCG or RBiCG, as the case may be). We show that solving the problem without recycling takes about 50% more time than with recycling.

3.5 RBiCG for VMC

In Section 2.2 we presented the result from [79] that shows the advantage of using BiCG in solving dual linear systems arising in the VMC algorithm. That is, in exact arithmetic we can show that solving dual linear systems using BiCG provides a quadratic error bound. Since the VMC algorithm requires solving sequences of such systems, RBiCG can be used to reduce the total runtime further. Note that the same advantage holds when RBiCG is used. The next corollary (to Theorem 2.3) summarizes this result.

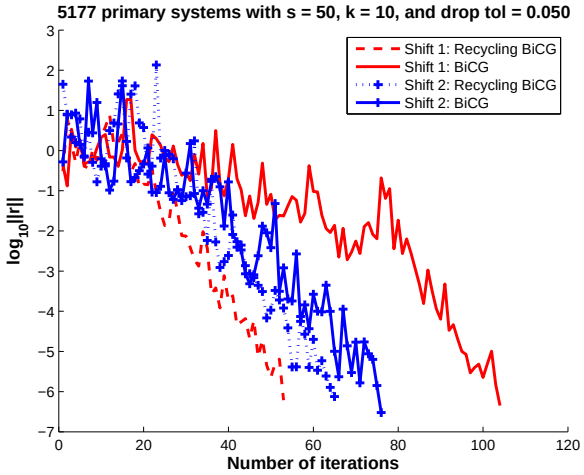


Figure 3.3: Convergence curves of ILU preconditioned RBiCG at the 2nd IRKA step for the 5177×5177 rail model. The length of the cycle is 50, the number of vectors selected for recycling are 10, and the preconditioner drop tolerance is 0.05.

Primary System				Dual System			
IRKA Step 1 $\sigma_1 = 1.000 \times 10^{-5}$		IRKA Step 2 $\sigma_1 = 1.834 \times 10^{-5}$		IRKA Step 1 $\sigma_1 = 1.000 \times 10^{-5}$		IRKA Step 2 $\sigma_1 = 1.834 \times 10^{-5}$	
End of Cycle 1	End of Cycle 2	Start of Cycle 1	End of Cycle 1	End of Cycle 1	End of Cycle 2	Start of Cycle 1	End of Cycle 1
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1.0000	1.0000	1.0000	1.0000	0.9997	1.0000	1.0000	1.0000
0.9987	1.0000	1.0000	1.0000	0.9765	1.0000	1.0000	1.0000
0.9321	1.0000	1.0000	1.0000	0.4936	1.0000	1.0000	1.0000
0.2257	1.0000	0.9998	0.9999	0.0844	0.9995	0.9997	0.9998
0.0260	0.9997	0.9996	0.9997	0.0231	0.9945	0.9945	0.9989
0.0072	0.7813	0.7799	0.9932	0.0068	0.3439	0.3423	0.9876

Table 3.3: Analysis of the recycle space for the 5177×5177 rail model and the sequence of linear systems corresponding to the smallest shift, σ_1. The columns list the cosine of principal angles between the recycle space and the invariant subspace spanned by eight eigenvectors associated with the eigenvalues of smallest magnitude. For the primary system, we use the invariant subspace spanned by right eigenvectors. For the dual system, we use the invariant subspace spanned by left eigenvectors. The recycle space that we use in computing the third column (Primary System; IRKA Step 2; Start of Cycle 1) is the same space that leads to the dashed convergence curve in Figure 3.3.

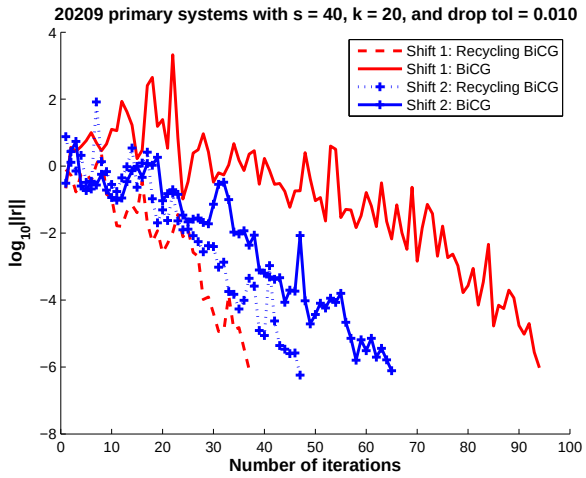


Figure 3.4: Convergence curves of ILU preconditioned RBiCG at the 2nd IRKA step for the 20209×20209 rail model. The length of the cycle is 40, the number of vectors selected for recycling are 20, and the preconditioner drop tolerance is 0.01.

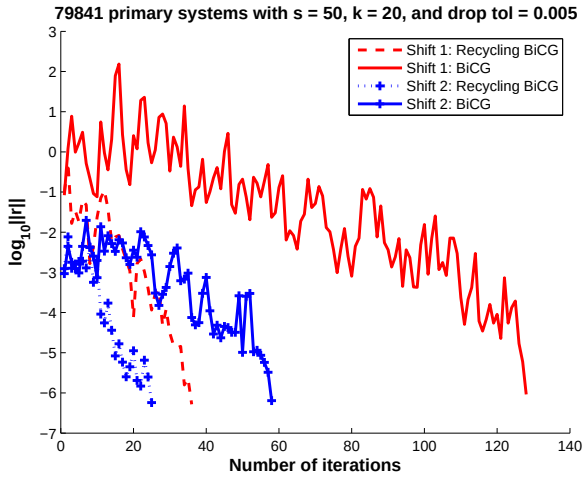


Figure 3.5: Convergence curves of ILU preconditioned RBiCG at the 3rd IRKA step for the 79841×79841 rail model. The length of the cycle is 50, the number of vectors selected for recycling are 20, and the preconditioner drop tolerance is 0.005.

Size	s	k	Drop tol	IRKA steps	Total iteration count			Total time (s)		
					BiCG	RBiCG	Ratio	BiCG	RBiCG	Ratio
20209	40	20	0.01	31	3032	1434	2.11	73.82	54.28	1.36
79841	50	20	0.005	44	6324	2547	2.48	742.83	505.09	1.47

Table 3.4: Results for the second set of experiments. s is the length of cycle, k is the number of vectors selected for recycling, and drop tol is the ILUTP preconditioner drop tolerance. IRKA steps is the total number of IRKA steps needed to converge to the ideal shifts with a relative tolerance of 10^{-6}. The total iteration count is sum of iteration count for solving the three dual linear systems (corresponding to the three shifts) over all IRKA steps. The ratio is computed as the BiCG iteration count divided by RBiCG iteration count. The total time is reported in seconds, and is the time taken for IRKA to converge to the ideal shifts. This time includes the calls to linear solver, BiCG or RBiCG as the case may be. The ratio is computed as the time for IRKA when using BiCG divided by the time for IRKA when using RBiCG.

Corollary 3.2. *If $A_k x = e_{i_k}$ and $A_k^T \tilde{x} = u_k$ are solved with RBiCG such that $\|e_{i_k} - A_k x\| = \mathcal{O}(\epsilon_1)$ and $\|u_k - A_k^T \tilde{x}\| = \mathcal{O}(\epsilon_2)$, then $\|\tilde{x}^T A_k x - u_k^T A_k^{-1} e_{i_k}\| = \mathcal{O}(\epsilon_1 \epsilon_2)$.*

Proof. Follow the same steps as for BiCG in [79]. □

However, in floating point arithmetic this result may not hold because the bi-orthogonality conditions of the bi-Lanczos algorithm and the Petrov-Galerkin approximation are not satisfied in finite precision computations. In [79] the authors propose using the following local bi-orthogonality relations to achieve the same result:

$$\tilde{r}_{j+1}^T p_j = 0 \quad \text{and} \quad \tilde{p}_j^T r_{j+1} = 0, \tag{3.24}$$

where j is the index for the iterative step and $r_j, \tilde{r}_j, p_j, \tilde{p}_j$ are as defined in Algorithm 3.1 (BiCG). If the residuals of the primary and dual systems become small, the bilinear form

Absolute Convergence Tolerance	Relative Convergence Tolerance		Determinant Ratio Error	
	Primary System	Dual System	RBiCG	GMRES
10^{-2}	10^{-2}	10^{-3}	order 10^{-7}	order 10^{-4}
10^{-3}	10^{-3}	10^{-5}	order 10^{-8}	order 10^{-5}

Table 3.5: Determinant ratio error versus tolerance for RBiCG and GMRES. For the same convergence tolerance and roughly same number of iterative steps needed for convergence, RBiCG leads to smaller determinant ratio error. Relative Convergence Tolerance is Absolute Convergence Tolerance normalized by the right hand side vector.

needed in (2.9) can be computed using the BiCG algorithm as follows [79]:

$$u_k^T A_k^{-1} e_{i_k} = \sum_{j=0}^{m} \alpha_j \tilde{r}_j^T r_j, \tag{3.25}$$

where m is the number of iterative steps taken until convergence and α_j is the scalar from Algorithm 3.1 (BiCG). Relations (3.24) and (3.25) hold for the RBiCG algorithm as well. That is, r_j, $\tilde{r}_j$, p_j, $\tilde{p}_j$, and α_j as defined in Algorithm 3.2 (RBiCG).

We now demonstrates that the approach suggested by Corollary 3.2 works (pays off) in practice as well. The test problem is a 1024×1024 system. Single particle orbitals are Gaussians with $k = 1$ (see (2.13)). We first compute determinant ratios using the RBiCG algorithm (using (3.25)) and the GMRES algorithm (solving $A_k z_k = e_{i_k}$ and using it to compute $u_k^T z_k$). We then compute error in the determinant ratio for each case, which is defined as the determinant ratio computed by using dense linear algebra minus the determinant ratio computed by an iterative method (RBiCG or GMRES as the case may be). Table 3.5 shows that computing determinant ratios using RBiCG is competitive with GMRES. For the same convergence tolerance and roughly same number of iterative steps needed for convergence, RBiCG leads to smaller determinant ratio errors. This compensates for the fact that RBiCG requires two matrix-vector products per iterative step as compared with one matrix-vector product per step for GMRES.

Since the linear systems for this problem converge very fast, recycling Krylov subspaces does not provide much benefit. Recycling Krylov subspaces might be useful for certain materials where the linear systems take large number of iterations to converge.

Chapter 4

Recycling in Transpose Free Bi-Lanczos Algorithms

If we need to solve a primary system and a dual system, then BiCG is a good choice. If only a primary system is to be solved, then other popular bi-Lanczos based algorithms like CGS [76], BiCGSTAB [82], BiCGSTAB2 [50], BiCGSTAB(l) [74], QMR [43], and TFQMR [41] might be better. The BiCG algorithm has a couple of drawbacks. It requires the matrix transpose, and it requires *two matrix-vector* products to extend the Krylov subspace by *one vector* [27].

This led to the development of CGS. Besides avoiding the above drawbacks, the CGS algorithm often converges faster. However, CGS may have irregular convergence behavior, and this sometimes leads to cancellation errors. The BiCGSTAB algorithm, which is a smoothly converging variant of CGS, avoids this problem by performing a one-dimensional local minimization of the residual. This minimization can be done in two or more dimensions as well. This led to the development of BiCGSTAB2 and BiCGSTAB(l).

With our focus on sequences of single linear systems of the type (1.1), we discuss recycling CGS (RCGS) and recycling BiCGSTAB (RBiCGSTAB) first. We then present recycling BiCGSTAB2 (RBiCGSTAB2). Finally, we provide experimental results using Matlab.

4.1 RCGS and RBiCGSTAB

In [3], we derived RCGS and RBiCGSTAB based on the operators $A_1 = (I - C\mathcal{D}_c^{-1}\tilde{C}^*)A(I - C\mathcal{D}_c^{-1}\tilde{C}^*)$ and A_1^*. Those same derivations follow through with matrices $B = (I - C\mathcal{D}_c^{-1}\tilde{C}^*)A$ and $\tilde{B} = (I - \tilde{C}\mathcal{D}_c^{-1}C^*)A$. Recall that the initial residuals in RBiCG are given by $r_0 = (I - C\mathcal{D}_c^{-1}\tilde{C}^*)r_{-1}$ and $\tilde{r}_0 = (I - \tilde{C}\mathcal{D}_c^{-1}C^*)\tilde{r}_{-1}$.

Lemma 4.1. *For all $\gamma \in \mathbb{N}$, $B^\gamma r_0 = A_1^\gamma r_0$ and $\tilde{B}^\gamma \tilde{r}_0 = (A_1^*)^\gamma \tilde{r}_0$.*

Proof. It is easily seen that $(I - C\mathcal{D}_c^{-1}\tilde{C}^*)$ and $(I - \tilde{C}\mathcal{D}_c^{-1}C^*)$ are projectors. Hence,

$$\begin{aligned}(I - C\mathcal{D}_c^{-1}\tilde{C}^*)^2 &= (I - C\mathcal{D}_c^{-1}\tilde{C}^*),\\ (I - \tilde{C}\mathcal{D}_c^{-1}C^*)^2 &= (I - \tilde{C}\mathcal{D}_c^{-1}C^*).\end{aligned} \tag{4.1}$$

Using the definitions of A_1, B, and r_0, we get that $Br_0 = A_1 r_{-1}$. Using (4.1) we get that $A_1 r_{-1} = A_1 r_0$. Hence, $Br_0 = A_1 r_0$ and $B^k r_0 = A_1^k r_0$ implies $B^{k+1} r_0 = A_1^{k+1} r_0$ for any $k \in \mathbb{N}$. Thus, $B^\gamma r_0 = A_1^\gamma r_0$ follows by using induction. One can similarly show that $\tilde{B}^\gamma \tilde{r}_0 = (A_1^*)^\gamma \tilde{r}_0$. □

Using the above lemma and following the same steps as in [3], we can derive RCGS and RBiCGSTAB with operators B and $\tilde{B}$ instead of A_1 and A_1^*. These derivations are based on polynomial representation of the RBiCG iteration vectors. This polynomial representation is briefly discussed while deriving RBiCGSTAB2 in the next section. Finally, the RCGS iteration scalars and vectors are given as follows:

$$\begin{aligned}\alpha_i &= \tfrac{(\tilde{r}_0,\ r_{i-1})}{(\tilde{r}_0,\ Bp_i)},\\ \beta_i &= \tfrac{(\tilde{r}_0,\ r_i)}{(\tilde{r}_0,\ r_{i-1})},\\ p_i &= u_{i-1} + \beta_{i-1}(q_{i-1} + \beta_{i-1}p_{i-1}),\\ u_i &= r_{i-1} + \beta_{i-1}q_{i-1},\end{aligned}$$

$$
\begin{aligned}
q_i &= u_i - \alpha_i B p_i, \\
x_i &= x_{i-1} + \alpha_i Z(u_i + q_i), \\
r_i &= r_{i-1} - \alpha_i B(u_i + q_i),
\end{aligned}
$$

with $Z = (I - U\mathcal{D}_c^{-1}\tilde{C}^* A)$. Similarly, for RBiCGSTAB we get the following iteration scalars and vectors:

$$
\begin{aligned}
&\alpha_i = \tfrac{(\tilde{r}_0,\ r_{i-1})}{(\tilde{r}_0,\ Bp_i)}, \\
&\beta_i = \tfrac{(\tilde{r}_0,\ r_i)}{(\tilde{r}_0,\ r_{i-1})} . \tfrac{\alpha_i}{\omega_i}, \\
&\omega_i = \tfrac{(s_i, t_i)}{(t_i, t_i)}, \text{ with } s_i = r_{i-1} - \alpha_i B p_i \text{ and } t_i = B s_i, \\
&p_i = r_{i-1} + \beta_{i-1} p_{i-1} - \beta_{i-1}\omega_{i-1} B p_{i-1}, \\
&x_i = x_{i-1} + \alpha_i Z p_i + \omega_i Z(r_{i-1} - \alpha_i B p_i). \\
&r_i = r_{i-1} - \alpha_i B p_i - \omega_i B(r_{i-1} - \alpha_i B p_i).
\end{aligned}
$$

Note that the above algorithms can be implemented in a more efficient way than presented here (in the same sense as for RBiCG in Chapter 3).

4.2 RBiCGSTAB2

We first describe the polynomial representation of RBiCG iteration vectors r_i, $\tilde{r}_i$, p_i, and $\tilde{p}_i$ using RBiCG scalars α_i, $\tilde{\alpha}_i$, β_i, and $\tilde{\beta}_i$. All scalars and vectors are defined in Algorithm 3.2. We then derive the RBiCGSTAB2 algorithm by closely following the derivation of BiCGSTAB2 [50].

Theorem 4.1. *For the primary system*

$$
r_i = \Theta_i(B) r_0, \quad p_i = \Pi_{i-1}(B) r_0, \tag{4.2}
$$

where $\Theta_i(K)$ and $\Pi_{i-1}(K)$ are i-th and $(i-1)$-th degree polynomials in K with the following

polynomial recurrences:

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

Similarly, for the dual system

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

where $\bar{\Theta}_i(K)$ *and* $\bar{\Pi}_i(K)$ *are* i*-th and* $(i-1)$*-th degree polynomials in* K *with the following polynomial recurrences:*

$$\bar{\Theta}_i(K) = \bar{\Theta}_{i-1}(K) - \bar{\alpha}_i K\bar{\Pi}_{i-1}(K), \quad \bar{\Pi}_{i-1}(K) = \bar{\Theta}_{i-1}(K) + \bar{\beta}_{i-1}\bar{\Pi}_{i-2}(K).$$

Proof. Use Lemma 4.1 and follow the same steps as in the proof of Theorem 4.1 in [3]. □

Using (3.15), (4.2), and (4.3) we obtain

$$(\bar{\Theta}_j(\tilde{B})\tilde{r}_0, \Theta_i(B)r_0) = 0 \text{ for } j < i.$$

This implies $\Theta_i(B)r_0 \perp \mathcal{K}^i(\tilde{B}, \tilde{r}_0)$, where vectors $\tilde{r}_0,\ \tilde{B}\tilde{r}_0,\ \cdots,\ (\tilde{B})^{i-1}\tilde{r}_0$ span the subspace $\mathcal{K}^i(\tilde{B}, \tilde{r}_0)$. As observed in [82], the above orthogonality conditions must be satisfied by any other bases of $\mathcal{K}^i(\tilde{B}, \tilde{r}_0)$ too. So, other polynomials can be used as well [90]. That is,

$$(\bar{\Omega}_j(\tilde{B})\tilde{r}_0, \Theta_i(B)r_0) = 0 \text{ for } j < i.$$

As in the BiCGSTAB algorithm, in RBiCGSTAB we define

$$\Omega_i(K) = (I - \omega_i K)\Omega_{i-1}(K),$$
$$\bar{\Omega}_i(K) = (I - \bar{\omega}_i K)\bar{\Omega}_{i-1}(K),$$

where ω_i is selected by minimizing the residual r_i with respect to ω_i. Similarly, as in the

BiCGSTAB2 algorithm, for RBiCGSTAB2 we define

$$\Omega_{2i+1}(K) = (I - \omega_i K)\Omega_{2i}(K),$$
$$\bar{\Omega}_{2i+1}(K) = (I - \bar{\omega}_i K)\bar{\Omega}_{2i}(K),$$

and

$$\Omega_{2i+2}(K) = (I - \xi_i K)\Omega_{2i}(K) + (\xi_i + \eta_i K)\Omega_{2i+1}(K),$$
$$\bar{\Omega}_{2i+2}(K) = (I - \bar{\xi}_i K)\bar{\Omega}_{2i}(K) + (\bar{\xi}_i + \bar{\eta}_i K)\bar{\Omega}_{2i+1}(K),$$

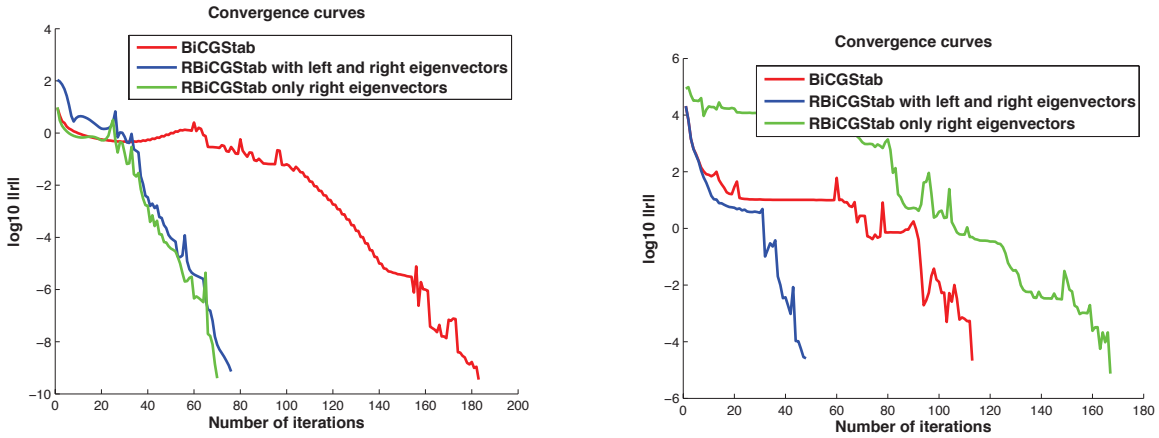
where ω_i is selected by a one-dimensional minimization of the residual, and ξ_i and η_i are obtained from a two-dimensional minimization of the residual. The solution update is now given by

$$x_{2i+1} = x_{2i} + \alpha_{2i} p_{2i} + \omega_i Z s_{2i+1},$$
$$x_{2i+2} = \left[x_{2i} + \alpha_{2i} Z p_{2i} + \alpha_{2i+1} Z t_{2i+1}\right](1 - \xi_i) + \left[x_{2i+1} + \alpha_{2i+1} Z p_{2i+1}\right]\xi_i - \eta_i Z s_{2i+2},$$

where $Z = (I - U\mathcal{D}_c^{-1}\tilde{C}^* A)$; α, ω, ξ, and η are the iteration scalars; and p, s, and t are the iteration vectors. Except the solution update recurrences (as given above), all other scalar and vector recurrences for the RBiCGSTAB2 algorithm can be obtained by performing the following replacements in the respective recurrences of the BiCGSTAB2 algorithm: operator A by $(I - C\mathcal{D}_c^{-1}\tilde{C}^*)A$, primary system initial residual r_{-1} by $(I - C\mathcal{D}_c^{-1}\tilde{C}^*)r_{-1}$, and dual system initial residual $\tilde{r}_{-1}$ by $(I - \tilde{C}\mathcal{D}_c^{-1}C^*)\tilde{r}_{-1}$.

4.3 Analysis and Numerical Experiments

For BiCG, it has been shown that including a left eigenvector is the search space leads to the removal of the corresponding right eigenvector from the right residual (and vice versa) [29]. In our experiments we demonstrate that recycling left eigenvectors may improve the convergence rate in the RBiCGSTAB algorithm. We consider two examples. The first



(a) Example 1: Left eigenvectors not needed.

(b) Example 2: Left eigenvectors needed.

Figure 4.1: Convergence curves for two examples using RBiCGSTAB. The experiment demonstrate that recycling left eigenvectors improves the convergence rate in the RBiCGSTAB algorithm.

example is a 1600×1600 linear system, which we obtain from finite volume vertex centered discretization of the PDE

$$-(pu_x)_x - (qu_y)_y + ru_x + su_y + tu = f,$$

on the unit square with $p = 1, q = 1, r = 10, s = -10, t = 0$, and $f = 0$. We use the following boundary conditions: $\mathrm{u}_{\mathrm{south}} = 1, \mathrm{u}_{\mathrm{west}} = 1, \mathrm{u}_{\mathrm{north}} = 0, \mathrm{u}_{\mathrm{east}} = 0$. We do not use a preconditioner in this example, the initial guess is a vector of all ones, and the relative convergence tolerance is 10^{-10}. As the second example we use the example from Section 3.3. We set the mesh width to $h = 1/128$, leading to a linear system with 16129 unknowns. We use an ILUTP [70] preconditioner with a drop tolerance of 0.2 (split-preconditioned as earlier). The initial guess is 0.5 times a vector of all ones, and the relative convergence tolerance is 10^{-8}.

For each example we do three experiments. First, we solve the system without recycling. Second, we use the right eigenvectors to span the recycle space (implemented by setting $\tilde{U} = U$). Finally, we use both left and right eigenvectors to span the recycle space. For the

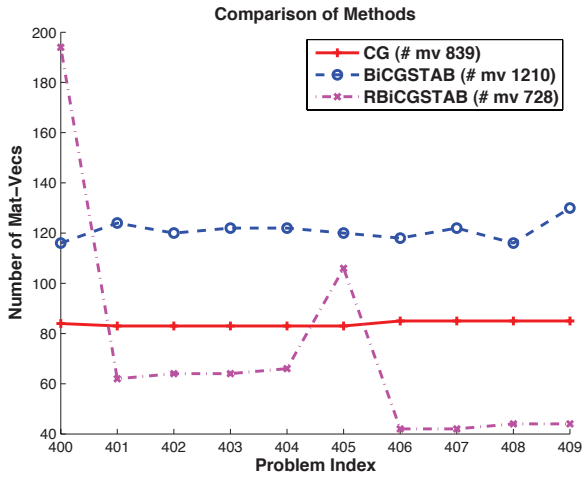


Figure 4.2: Using RBiCGSTAB for a crack propagation example. Mat-Vecs and mv denote matrix-vector product.

first example, we use use five exact eigenvectors (five right; five left and right) computed by using Matlab's `eigs`. For the second example, we use twenty approximate eigenvectors (twenty right; twenty left and right) obtained by solving the problem once with RBiCG.

The results are shown in Figure 4.1 (a) and (b). For the first example, using right eigenvectors or using both left and right eigenvectors works equally well. However, for the second example, we see that not using left eigenvectors leads to convergence that is worse than BiCGSTAB without recycling, and much worse than RBiCGSTAB using both left and right eigenvectors.

Next, we test RBiCGSTAB for an engineering problem. The problem simulates crack propagation in a metal plate using cohesive finite elements. This code was developed by Philippe Geubelle (Aerospace Engineering, University of Illinois at Urbana-Champaign) and Spandan Maiti (now at Mechanical Engineering, Michigan Tech). The problem requires solving a sequence of linear systems where both the matrix and the right hand side change. Each linear system has 3988 unknowns. The matrices are SPD, and hence, we use an IC(0) preconditioner. The systems are preconditioned from the left. We use a relative convergence tolerance of 10^{-8}. The initial guess for each system is the zero vector, since the systems solve for, incremental displacements associated with the loading increments.

A recycle space can be effective for multiple consecutive systems [66, 57, 64]. Since approximate left eigenvectors are not easily available from the RBiCGSTAB iterations[1], we solve every fifth linear system in the sequence using BiCG[2]. We then use the generated recycle space, twenty left and right approximate eigenvectors, to solve the subsequent four linear systems in RBiCGSTAB. The results for solving linear systems from loading step 400 to 409 are given in Figure 4.2. We compare our results with BiCGSTAB as well as CG [53] (since we have SPD systems). RBiCGSTAB requires the least number of total matrix-vector products over the ten systems. In the future, we intend to do rigorous experiments with nonsymmetric matrices, more systems in the sequence, and timing comparisons as well. The graph suggests that the number of mat-vecs for RBiCGSTAB might keep decreasing when solving linear systems from loading step 410 and beyond.

[1]Sometimes left eigenvectors are available from right eigenvectors [2, 64].

[2]Note that here we have SPD systems, and hence, the left and the right eigenvectors are the same. However, this process mimics what one would do in case of linear systems with nonsymmetric matrices

Chapter 5

Recycling Preconditioners for Variational Monte Carlo

In this chapter, we describe our algorithm that reduces the cost of evaluating determinant ratios for physically realistic systems in VMC to about $O(n^2)$ per sweep. In the next section, we study few properties and structures of Slater matrices. We give details about our algorithm in Section 5.2. Finally, in Section 5.3, we provide numerical results that demonstrate close to quadratic scaling for evaluating determinant ratios.

5.1 Analysis of Slater Matrices

Below, we provide a brief overview of properties of Slater matrices arising from the parameters that are used in the simulation (as discussed in Section 2.2). First, in figure 5.1, we show the sparsity pattern of the Slater matrix for a typical configuration with 1024 particles and $k = 1$. As the basis functions have local support (after truncating outside a cut-off radius) and the electrons, on average, are distributed evenly, the sparsity pattern is similar to that of a finite difference matrix for a regular 3D grid and a relatively wide stencil, where the width

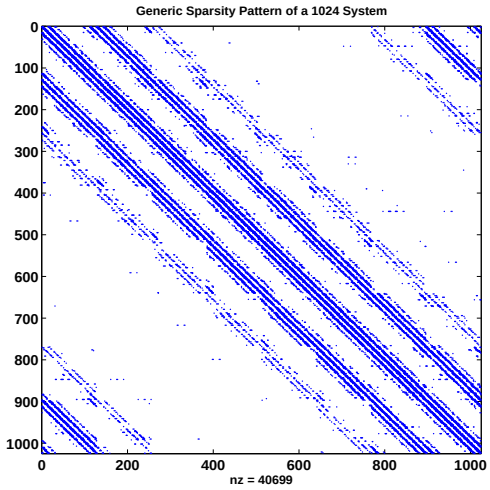


Figure 5.1: Sparsity pattern for a typical configuration of our system for 1024 particles with $k = 1$. For smaller k the pattern remains more or less the same, while the number of nonzeros in the matrix increases. For larger k the pattern also remains more or less the same, while the number of nonzeros in the matrix decreases.

of the stencil is comparable to the cut-off radius of the Gaussian orbitals (which in turn depends on k). Second, we give typical spectra for matrices arising from systems with 1024 particles and $k = 1.5$, 1, and 0.5. We point out that for the physics the ordering of rows and columns is irrelevant, since the square of the determinant is invariant under such changes. However, the eigenvalues can change significantly under reordering of rows and columns, which in turn can have a significant influence on the convergence of iterative methods (see section 5.2 for the reordering algorithm used). In figures 5.2–5.4, for each value of k, we provide the spectrum of a matrix before reordering, the spectrum of that same matrix after reordering, and the spectrum of that matrix after reordering and with preconditioning. The latter spectrum is the most relevant for the iterative solver.

Next, in Table 5.1, we provide typical condition numbers for various values of the parameter k (in the Gaussian orbitals) and three problem sizes, $n = 686$, 1024, and 2000. Note that, although for $k = 0.5$ the condition number appears to increase slightly by preconditioning, the spectrum improves drastically. This also bears out in the iterative solver; preconditioning reduces the number of iterations. From Table 5.1, we see that the condition number of the unpreconditioned Slater matrix increases with decreasing k. This is to be expected; see, e.g.,

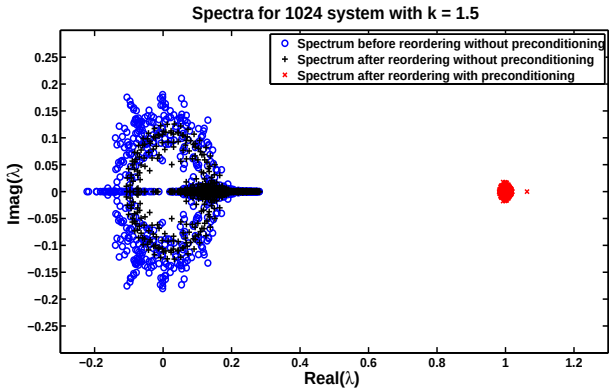


Figure 5.2: Spectra for a typical configuration of our system for 1024 particles with $k = 1.5$, before and after reordering, and with preconditioning for the reordered system.

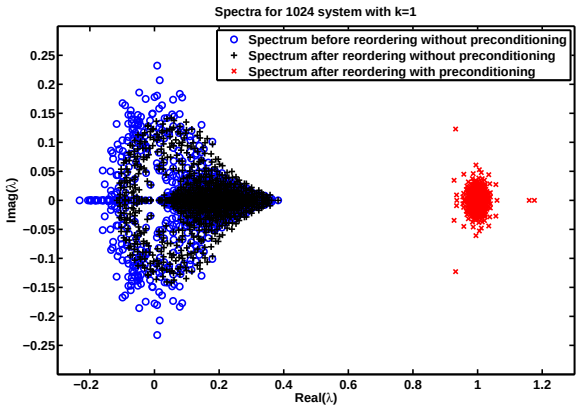


Figure 5.3: Spectra for a typical configuration of our system for 1024 particles with $k = 1$, before and after reordering, and with preconditioning for the reordered system. The value $k = 1$ is used in the experiments reported in section 5.3.

[21].

Although the orbitals for realistic systems (physical materials) may differ significantly from Gaussians, we expect many of the properties of the resulting Slater matrices to be similar. If orbitals decay sufficiently fast, the matrix will have the same banded sparsity pattern (after appropriate reordering) and be diagonally dominant or nearly so. In that case, all or most eigenvalues will be in the right half plane. However, a poor ordering of rows and columns will lead to eigenvalues located around the origin, as is the case here. If the decay is slow the matrix will become more ill-conditioned. Moreover, if the decay is sufficiently slow, there

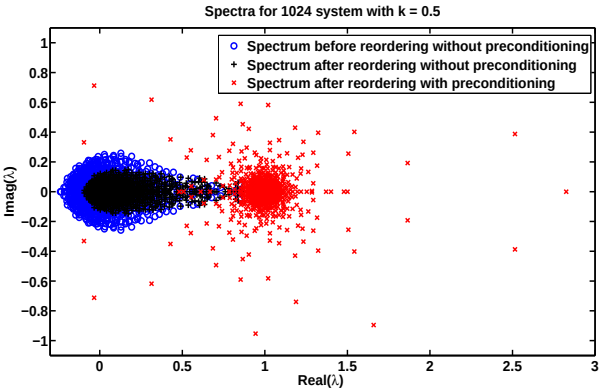


Figure 5.4: Spectra for a typical configuration of our system for 1024 particles with $k = 0.5$, before and after reordering, and with preconditioning for the reordered system. Note that the scale of this picture differs from the previous two. In addition, a lonely eigenvalue of the preconditioned system after reordering at 11.77 has been left out to obtain a better scale.

k	1.5			1.0			0.5		
Size	686	1024	2000	686	1024	2000	686	1024	2000
$\kappa(A)$	73	1.4e2	1.3e3	1.6e2	6.7e2	6.7e2	2.4e3	1.1e4	8.3e3
$\kappa\left(A(LU)^{-1}\right)$	1.2	1.4	4.7	3.1	10	23	8.0e3	1.5e5	1.1e5

Table 5.1: Typical (spectral) condition numbers, κ, for Slater matrices and ILUTP preconditioned Slater matrices for $k = 1.5$, 1, 0.5 and problem sizes $n = 686$, 1024, 2000.

will be no ordering that yields diagonal dominance and the spectrum cannot be guaranteed to be in the right-half plane. In that case, we expect that there will be no ordering that, by itself, will lead to a nice spectrum (for iterative solvers), and preconditioning will be more important. Analyzing these properties and their dependency on the properties of orbitals, decay rate, and lattice type will be future work.

5.2 Algorithmic Improvements

As described in the Section 2.2, the sequence of particle updates, moving particle i_k at step k in the MCMC method, leads to a sequence of matrix updates for the trial configuration,

$$\tilde{A}_k = A_k + e_{i_k} u_k^T = A_k(I + A_k^{-1} e_{i_k} u_k^T), \tag{5.1}$$

where

$$(u_k)_j = \phi_j(r'_{i_k}) - \phi_j(r_{i_k}), \quad \text{for } j = 1, \ldots, n, \tag{5.2}$$

A_k is the Slater matrix at the k^{th} Monte Carlo step, e_{i_k} is the Cartesian basis vector with a 1 at position i_k, the ϕ_j are the single particle orbitals used in (2.11)–(2.12), and r_{i_k} and r'_{i_k} are the old and the new position of the particle i_k, respectively. We do not need to compute $\phi_j(r_{i_k})$ since it equals $(A_k)_{i_k,j}$. The acceptance probability of the trial configuration depends on the squared absolute value of the determinant ratio of the two matrices,

$$\frac{|\det \tilde{A}_k|^2}{|\det A_k|^2} = \left|1 + u_k^T A_k^{-1} e_{i_k}\right|^2, \tag{5.3}$$

which can be computed by solving the linear system $A_k z_k = e_{i_k}$ and taking the inner product $u_k^T z_k$. We compare the value from (5.3) with a random number drawn from the uniform distribution on $(0, 1)$. If the trial configuration giving $\tilde{A}_k$ is accepted, $A_{k+1} = \tilde{A}_k$, if the trial configuration is rejected, $A_{k+1} = A_k$.

The use of maximally localized single particle orbitals leads to a sparse Slater matrix in some cases (insulators). In this case, iterative solvers provide a promising approach to compute these determinant ratios, as long as effective preconditioners can be computed or updated cheaply. Variations of incomplete decompositions (such as incomplete LU) are good candidates for this problem, as they have proven effective for a range of problems and require no special underlying structure (in principle). Unfortunately, the sequence of particle updates leads to matrices that are far from diagonally dominant, often have eigenvalues surrounding the origin, and have unstable incomplete decompositions in the sense defined in [18, 70]. However, the properties of orbitals and localization suggest that with a proper ordering of orbitals and particles the Slater matrix will be nearly diagonally dominant. Our method resolves the preconditioning problem by combining the following three improvements.

First, we have derived a geometric reordering of electrons and orbitals, related to the ordering proposed in [34], that provides nearly diagonally dominant Slater matrices. This ordering

combined with an ILUTP preconditioner [69, 70] leads to very effective preconditioners; see section 5.3. However, reordering the matrix and computing an ILUTP preconditioner every (accepted) VMC step would be too expensive. Therefore, as the second improvement, we exploit the fact that $A_{k+1} = A_k(I + A_k^{-1} e_{i_k} u_k^T)$ (5.1) and use a corresponding update to the right preconditioner, M_k, such that $A_{k+1}M_{k+1} = A_k M_k$. This leads to cheap intermediate updates to our preconditioners. Moreover, if M_k has been computed such that $A_k M_k$ has a favorable spectrum for rapid convergence [83, 46, 70], then subsequent preconditioned matrices, $A_{k+s}M_{k+s}$, have the same favorable spectrum. However, each update increases the cost of applying the preconditioner, and we periodically compute a new ILUTP preconditioner. Third, we assess whether potential instability of the incomplete LU decomposition affects the iteration by applying the stability metrics from [18, 70, 26] in an efficient manner. We only reorder the matrix if instability occurs, or if our iterative solver does not converge to the required tolerance in the maximum number of iterations, or if our iterative solvers takes more than four times the average number of iterations. This monitoring and reordering is necessary as the matrix becomes far from diagonally dominant and the incomplete LU decompositions slowly deteriorate due to the continual updates to the Slater matrix. This approach has proved very effective and limits the number of reorderings to a few per sweep. In spite of the matrix reordering, the explicit reordering of electrons and orbitals, based on the stability metric discussed below or on slow convergence, pivoting in the incomplete LU factorization is necessary. If we do not pivot the factorization occasionally breaks down. Such a breakdown could be avoided by doing the explicit reordering of electrons and orbitals every step, but this would be too expensive. Moreover, not pivoting leads to denser L and U factors and slower convergence in the iterative linear solver; both effects increase the total amount of work.

Reordering for Near Diagonal Dominance and Efficient Reordering Criteria

First, we discuss an efficient way to judge the quality of the preconditioner. In the second part of this section, we discuss a reordering that improves the quality of the preconditioner.

The quality of an ILU preconditioner for the matrix A, $LU \approx A$, can be assessed by its *accuracy* and *stability* [18]. The accuracy of the ILU preconditioner, defined as

$$N_1 = ||A - LU||_F, \tag{5.4}$$

measures how close the product LU is to the matrix A. The stability of the preconditioner, for right preconditioning defined as

$$N_2 = ||I - A(LU)^{-1}||_F, \tag{5.5}$$

measures how close the preconditioned matrix is to the identity. For left preconditioning, $N_2 = ||I - (LU)^{-1}A||_F$. Although for some classes of matrices N_1 is a good measure of preconditioning quality, for general matrices N_2 is a more useful indicator [18, 19]. We will see that this is also the case here. In general, instability arises from a lack of diagonal dominance and manifests itself in very small pivots and/or unstable triangular solves.

In practice, computing N_2 is much too expensive, and we need to consider a more economic indicator. An alternative approach, suggested in [26], is to compute $||(LU)^{-1}e||_\infty$, where e is the vector of all 1's. However, this still requires solving an additional linear system. Instead, we propose to use an *effective* or *local* stability measure,

$$N = \max_i \|v_i - A(LU)^{-1}v_i\|_2, \tag{5.6}$$

where the v_i are the Arnoldi vectors generated in the GMRES algorithm [71] during a linear solve. N measures the instability of the preconditioned matrix over the Krylov space from

which we compute a solution. If N is small, there is no unit vector $z \in K_m(A(LU)^{-1}, r_0)$ for which $\|(I - A(LU)^{-1})z\|_2$ is large (where m is the number of GMRES iterations). Indeed, for unit $z \in K_m(A(LU)^{-1}, r_0)$, $\|(I - A(LU)^{-1})z\|_2 \leq mN$. N can be small or modest when N_2 is large; however, this indicates that the instability does not play a role for vectors in the subspace over which a solution was computed; hence, the name effective or local stability. Note that N can be computed during the GMRES iteration at negligible cost, in contrast to the expensive computation of N_2. If N is large, the preconditioned matrix is ill-conditioned over the Krylov space used to find a solution, and we reorder the matrix as described below. Large N indicates that the solution might be inaccurate and that the preconditioner is deteriorating. This typically would lead to poor convergence either in the present solve or later, and so reordering and updating the preconditioner is better.

To check whether N_1 and N_2 are good indicators of preconditioner quality for our problem, we run the MCMC algorithm for 100 sweeps for a test problem with 2000 particles, and we check N_1 and N_2 each time GMRES does not converge in 15 iterations (which is relatively slow; see section 5.3). It would be useful to check the reverse as well, but computing N_1 and N_2 for every MC step ($200,000$ steps) would be too expensive. If GMRES does not converge in 15 iterations, we reorder the matrix as described below, compute a new ILUTP preconditioner, and solve the linear system again from scratch. This procedure always led to convergence within 15 iterations after the reordering. Although the experiments in section 5.3 are computed using a C/C++ code, for experiments in this subsection we use a Matlab based VMC code, developed for easy experimentation, that uses the `GMRES` and `luinc` routines of Matlab. The code uses left preconditioning, `luinc` with a drop tolerance of $\tau = 0.01$, and it allows pivoting in the entire pivot column (default). Furthermore, we use Gaussian orbitals (2.13) with $k = 1$.

In Figures 5.5 and 5.6 we plot, respectively, N_1 and N_2 at those Monte Carlo steps where the GMRES algorithm does not converge in 15 iterations, implying a deterioration of the preconditioner. We also plot N_1 and N_2 after reordering and recomputing the preconditioner. Figure 5.5 shows that the accuracy is always quite good, and hence accuracy is not a good

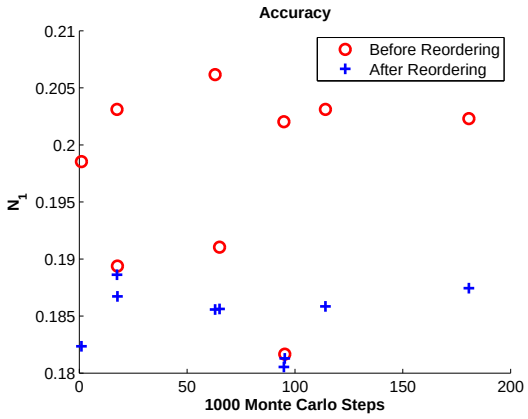


Figure 5.5: Accuracy before and after reordering.

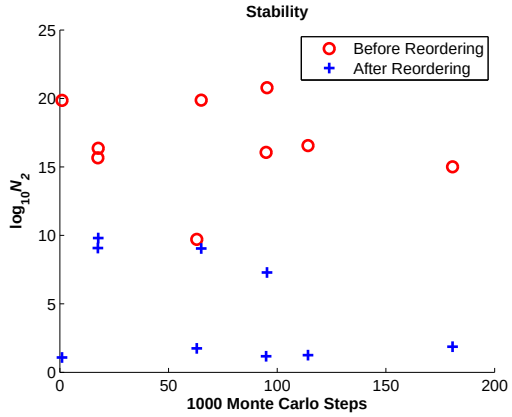


Figure 5.6: Stability before and after reordering.

indicator for reordering the matrix to improve preconditioner quality. This is in line with observations from [18]. Reordering does improve the accuracy further. Figure 5.6 shows that poor convergence goes together with very large values of N_2, and so N_2 is a better indicator for reordering the matrix to improve the preconditioner. Note that reordering improves the stability significantly, and usually reduces it to modest values, but with some exceptions. As N_2 seems a good indicator, but too expensive, we next consider the effective stability N.

Figure 5.7 shows large values of N corresponding to steps where the GMRES algorithm does not converge. The value of N is modest at other Monte Carlo steps (note that it is easy to compute N at every step). This demonstrates that N is an equally good indicator of the

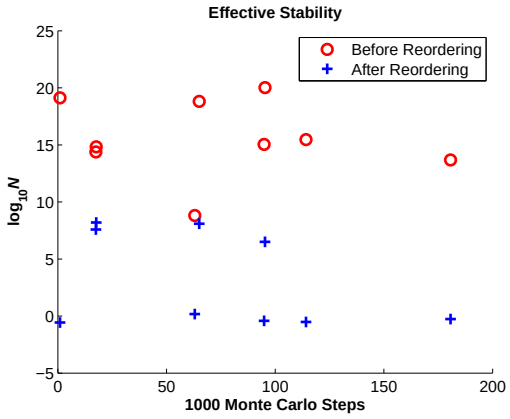


Figure 5.7: Effective stability before and after reordering.

effectiveness of the preconditioner as N_2, and hence we will use N as an indicator for matrix reordering. We remark that more timely reordering (before N gets so large that the matrix is numerically singular) also leads to smaller values for N after reordering the matrix and recomputing the preconditioner. Next, we discuss reordering the matrix when the quality of the preconditioner deteriorates.

Instability in the incomplete factorization, as discussed above (small pivots and/or ill-conditioned triangular solves), is generally associated with a lack of diagonal dominance [26, 18]. Instability can generally be removed or significantly reduced by preprocessing the coefficient matrix. Preprocessing strategies involve permutations and scalings for improving the conditioning, near-diagonal dominance, and the structure of the coefficient matrix; modifications such as perturbing diagonal elements are also possible [26, 18]. Improving near-diagonal dominance of the matrix through a judicious ordering of unknowns and equations has been shown quite effective in addressing instability, leading to a better preconditioner. In [19], the authors show that a simple reordering of the matrix can improve the quality of incomplete factorization preconditioners. More sophisticated reorderings that improve the near-diagonal dominance of the matrix are discussed in [32, 33]. These papers also show that reordering can have a dramatic effect on the convergence of ILU preconditioned iterative methods. Reorderings that exploit the physics of the underlying problem have also

proved quite effective [25].

We remark that the physics underlying our problem and the optimization of orbitals suggest that a proper ordering should lead to a nearly diagonally-dominant matrix. We also observe that, as all orbitals are scaled equally, we do not expect scaling to provide much improvement.

Since the orbitals used in our study are monotonically decreasing with distance, we propose a reordering that is simple, improves the near-diagonal dominance of the Slater matrix, and incorporates the physics of our problem. This reordering strategy performs a geometric reordering of particles and orbitals, and is similar to the reordering of inputs and outputs for a reliable control scheme in [34]. Our algorithm consists of the following steps, ignoring sparsity for simplicity.

1. Label the particles (P_i) and orbitals (O_j) from 1 to n,
   giving the following Slater matrix A:

$$\begin{array}{c} \\ P_1 \\ P_2 \\ \\ P_n \end{array} \begin{array}{c} \begin{array}{cccc} O_1 & O_2 & & O_n \end{array} \\ \begin{pmatrix} \phi_1(r_1) & \phi_2(r_1) & \cdots & \phi_n(r_1) \\ \phi_1(r_2) & \phi_2(r_2) & \cdots & \phi_n(r_2) \\ \vdots & \vdots & & \vdots \\ \phi_1(r_n) & \phi_2(r_n) & \cdots & \phi_n(r_n) \end{pmatrix} \end{array},$$

2. for $i = 1, \ldots, n-1$ do
   Find the closest orbital O_j to P_i for $j \in \{i, i+1, \ldots, n\}$
   if $j \neq i$ then
   renumber O_j as O_i and O_i as O_j (swap columns j and i)
   else
   find the particle P_k closest to orbital O_i for $k \in \{i, i+1, \ldots, n\}$
   if $k \neq i$ then
   renumber P_k as P_i and P_i as P_k (swap rows i and k)

```
        end if
    end if
end for
```

As mentioned above, reordering the matrix and recomputing the ILUTP preconditioner always leads to convergence within 15 GMRES iterations. So, the algorithm is quite effective. We can see from Figure 5.7 that N is always significantly reduced by reordering the matrix and recomputing the preconditioner. Moreover, more timely reordering of the matrix, before N gets so large that the matrix is numerically singular, also leads to smaller values for N after reordering.

The computational cost of a straightforward implementation of this reordering is $O(n^2)$. Since for our current experiments (and problem sizes) the runtime of reordering is negligible and reordering is needed only two or three times per sweep, we have not (yet) focused on an efficient implementation, especially since this likely requires substantial work in the software for matrix storage and manipulation. However, we will do this in the future. We remark that for sparse matrices where the number of nonzeros per row and per column is roughly constant, independent of n, an $O(n)$ implementation is possible. Moreover, this global reordering algorithm ignores the local nature of the particle updates and resulting changes to the matrix. In addition, maintaining further (multilevel) geometric information related to relative positions of particles and orbitals should make reordering more efficient. Hence in long term, we expect to replace this algorithm by one that makes incremental updates to the local ordering. If we start with a good ordering and the matrix is sparse (in the sense that the decay of the orbitals does not depend on the problem size), we expect that such local updates can be done at constant or near constant cost.

Cheap Intermediate Updates to the Preconditioner

Computing a new ILUTP preconditioner for every accepted particle update would be very expensive. However, we can exploit the structure of the matrix update to compute a cheap update to the preconditioner that maintains good convergence properties.

Assume that at some step k we have computed an incomplete LU preconditioner (ILU) with threshold and pivoting for the matrix A_k. Then, we have

$$A_k Q_k \simeq L_k U_k, \tag{5.7}$$

where Q_k is a column permutation matrix, and L_k and U_k are the incomplete lower and upper triangular factors, respectively. We consider right preconditioning, and so, instead of $A_k z_k = e_{i_k}$, we solve the right preconditioned linear system,

$$A_k M_k \tilde{z}_k = e_{i_k} \quad \text{with} \quad z_k = M_k \tilde{z}_k, \tag{5.8}$$

where $M_k = Q_k \left(L_k U_k\right)^{-1}$ is the preconditioner. If the trial move of particle i_k is accepted, we consider for the next step the updated matrix $A_{k+1} = A_k(I + (A_k^{-1} e_{i_k}) u_k^T)$; see (5.1). Now, let M_k be such that $A_k M_k$ has a favorable spectrum for rapid convergence [83, 46, 70]. Then defining the updated preconditioner M_{k+1} such that $A_{k+1} M_{k+1} = A_k M_k$ gives a new preconditioned matrix with the same favorable spectrum. Hence, we define the new preconditioner as

$$M_{k+1} = (I + (A_k^{-1} e_{i_k}) u_k^T)^{-1} M_k = \left(I - \frac{(A_k^{-1} e_{i_k}) u_k^T}{1 + u_k^T A_k^{-1} e_{i_k}} \right) M_k, \tag{5.9}$$

without explicitly computing M_{k+1}. Since $A_k^{-1} e_{i_k} = z_k$ and $1 + u_k^T A_k^{-1} e_{i_k}$ have already been computed to find the determinant ratio (5.3), we get M_{k+1} for free, and the cost of applying M_{k+1} is that of applying M_k plus the cost of a dot product and vector update. Let

$\hat{z}_k = (1 + u_k^T A_k^{-1} e_{i_k})^{-1} z_k$. Then M_{k+1} is defined as

$$M_{k+1} = (I - \hat{z}_k u_k^T) Q_k (L_k U_k)^{-1}, \tag{5.10}$$

where the inverse of $L_k U_k$ is implemented, of course, by a forward solve for L_k and a backward solve for U_k.

Since z_k is approximated by an iterative process, we have $A_{k+1} M_{k+1} \approx A_k M_k$, rather than exact equality. However, we have the following result. Let $\zeta_k = A_k^{-1} e_{i_k}$ and let $r_k = e_{i_k} - A_k z_k = A_k(\zeta_k - z_k)$. Then we have

$$\begin{aligned}
A_{k+1} M_{k+1} &= A_k (I + \zeta_k u_k^T)(I - \frac{z_k u_k^T}{1 + u_k^T z_k}) M_k \\
&= A_k \left(I + \zeta_k u_k^T - \frac{\zeta_k u_k^T}{1 + u_k^T z_k} + \frac{(\zeta_k - z_k) u_k^T}{1 + u_k^T z_k} - \zeta_k u_k^T \frac{u_k^T z_k}{1 + u_k^T z_k} \right) M_k \\
&= A_k \left(I + \frac{(\zeta_k - z_k) u_k^T}{1 + u_k^T z_k} \right) M_k \\
&= A_k M_k + \frac{r_k u_k^T}{1 + u_k^T z_k} M_k.
\end{aligned}$$

So, the relative change in the preconditioned matrix is small unless $|1 + u_k^T z_k|$ is very small or $\|r_k\| \|M_k\|$ is large relative to $\|A_k M_k\|$. However, $|1 + u_k^T z_k|$ governs the acceptance probability of the particle move. So, a very small value would occur in the preconditioner update with very small probability; we do not need to update the preconditioner if a trial move is rejected. This also guarantees that an accepted particle move will never result in a singular matrix, as the move will be accepted with probability 0. If $\|M_k\|$ is large while $\|A_k M_k\|$ is small (say $O(1)$), then A must have small singular values. In that case, an accurate z_k requires a sufficiently small residual. Hence, unless $\|A_k M_k\|$ is large (which monitoring N guards against), a proper choice of stopping criterion for $\|r_k\|$ should keep the relative change in the preconditioned matrix small.

Obviously, we can repeat the update of the preconditioner for multiple updates to the matrix. Defining $z_s = A_s^{-1} e_{i_s}$ (approximately from the iterative solve) and $\hat{z}_s = (1 + u_s^T z_s)^{-1} z_s$, where

u_s is given by (5.2) and $s = k, \ldots, k+m$, we have

$$M_{k+m} \;=\; \left(I - \hat{z}_{k+m-1} u_{k+m-1}^T\right) \cdots \left(I - \hat{z}_{k+1} u_{k+1}^T\right) \left(I - \hat{z}_k u_k^T\right) M_k. \tag{5.11}$$

In this case, the cost of applying the preconditioner slowly increases, and we should compute a new ILUTP preconditioner when the additional cost of multiplying by the multiplicative updates exceeds the (expected) cost of computing a new preconditioner. Of course, we also must compute a new preconditioner if we reorder (large N).

This technique for updating the preconditioner is similar to the idea of updating the (exact) inverse of the Jacobian matrix for Broyden's method [56], which is applied to the exact inverse of the Slater matrix in [65]. See also [20] which uses a similar approach to updating a preconditioner, however, for a general nonlinear iteration. So, the notion to keep the preconditioned matrix fixed, $A_{k+1} M_{k+1} = A_k M_k$, appears to be new.

5.3 Numerical Experiments

In this section, we numerically test our new algorithm. Apart from testing the performance of our algorithm, we must also test its reliability and accuracy. Since we compute the determinant ratios in the MCMC algorithm by iterative methods, we replace the acceptance/rejection test by an approximate test. In addition, the updating of preconditioners and their dependence on occasional reordering may lead to slight inconsistencies in the computed acceptance/rejection probabilities. This may affect the property of detailed balance for equilibrated configurations. Although we assume that using sufficiently small tolerances would make such potential problems negligible, we test and compare an observable (kinetic energy) computed in the new algorithm and in the standard algorithm. We remark that the standard method, with an updated inverse, also has the potential of accumulation of errors, which may affect its accuracy. However, the standard algorithm has performed satisfactorily, and a successful comparison of the results of our algorithm to those of the standard

algorithm should engender confidence.

As described in Section 2.2, we use Gaussian functions for the single-particle orbitals with $k = 1$, and we ignore the Jastrow factor. The lattice (giving the approximate locations of the nuclei in a solid) is selected as a Body Centered Cubic (b.c.c.) lattice, a lattice formed by cubes with nuclei/orbitals on all vertices and one in the middle. To test the scaling of our method, we fix the density of the electrons at $(3/4\pi) = 0.2387$ ptcl/unit and increase the number of electrons n; this corresponds to increasing the number of cubes in the lattice. This causes the number of orbitals to increase linearly with the electron number. As the orbitals are located on a b.c.c. lattice, we choose values of $n = 2K^3$ where K is an integer representing the number of orbitals along one side of the computational domain, and so the b.c.c. lattice is commensurate with the periodic box. Note that this leaves the spacing of the lattice (the distance between orbitals) the same for all n. The length of the side of a cube is 2.031, and the nearest neighbor distance is 1.759. The calculation of properties on larger and larger lattices is a typical procedure in VMC simulations to estimate the errors induced by not simulating an infinite number of particles. It is important to recognize that, although we are simulating an insulator, the electrons are not confined to the neighborhood of a single orbital and move around the entire box (hence the need for occasional reordering of the matrix). Since we use Gaussian orbitals, the Slater matrix has no coefficients that are (exactly) zero. However, most of the coefficients are very small and negligible in computations. Therefore, to make the matrix sparse, we drop entries less in absolute value than 10^{-5} times the maximum absolute value of any coefficient in the matrix. The number of non-zeros per row of the matrix varies between 40 and 50.

We use QMCPACK [54] for our simulations and, in its standard version, for comparison. In order to efficiently implement our iterative algorithms, we rewrote a significant part of QMCPACK to handle sparse matrices [1]. It might be advantageous to work with sparse vectors as well (sometimes referred to as sparse-sparse iterations) as the right hand sides in

[1]Note that the standard algorithm does not (and cannot) exploit sparsity in the matrix, except in computing and updating the Slater matrix itself.

(5.8) are Cartesian basis vectors. On the other hand, the preconditioner may quickly destroy sparsity of the iteration vectors. This remains something to test in the future. We integrated the new components of our VMC algorithm in QMCPACK [54] (written in C/C++); this includes the GMRES algorithm [71], the ILUTP preconditioner [69], our algorithms to update the preconditioner by rank-one updates to the identity (section 5.2), our reordering algorithm (section 5.2), and our test for instability of the preconditioner (section 5.2).

To simulate the system and gather statistics on the quality of the results for the new method and on its performance, we carry out 120 sweeps. We discount the data corresponding to the first 20 sweeps, which are used to ensure that the system is in equilibrium.

In the GMRES algorithm, we set the maximum number of iterations to 40, and the relative convergence tolerance to 10^{-6}. We monitor N, the effective stability (see section 5.2), to decide when to reorder the matrix; we reorder when $N > 100$. This is relatively low, but from experience this leads to faster (average) convergence. Note that, in our experiments, the number of reorderings is never more than three per sweep (see below). Finally, since N does not always predict slow convergence, even though it is quite reliable, we also reorder when the number of GMRES iterations reaches four times the average number of iterations or when the method does not converge in the maximum number of iterations. However, the occurrence of slow convergence while N is small accounts only for a very small fraction of the total number of reorderings, a few percent at most. If $N > 100$ or GMRES does not converge fast enough, we solve the same system again (from scratch) after reordering and computing a new preconditioner (GMRES is not restarted); this always resolves the convergence problem in our tests. In figures 5.8 and 5.9, we provide some information on N for the system with 4394 particles. In figure 5.8, we give $\log_{10}(N)$ for each Monte Carlo step in a representative window of roughly 6000 MC steps (the number of reorderings in this window is slightly higher than average though); in figure 5.9, we provide a histogram for the number of Monte Carlo steps between reorderings (each bin is of size 100). We note that sometimes reorderings follow each other with relatively short intervals. For the 4394 system, we have on average 2.73 reorderings per sweep (see table 5.4); so, the number of

steps between reorderings is, on average, about 1600 steps. However, we have a fair number of much shorter intervals. We conjecture that such shorter intervals occur when multiple particles are not relatively close to any orbital (nucleus), for example, when multiple pairs of electrons 'swap' nuclei. As the matrix changes only by one row per (successful) step and only by a modest amount, it is likely that matrices for which a good preconditioner is difficult to compute are clustered in the MCMC sequence. Alternatively, this phenomenon might suggest that we need a better reordering algorithm.

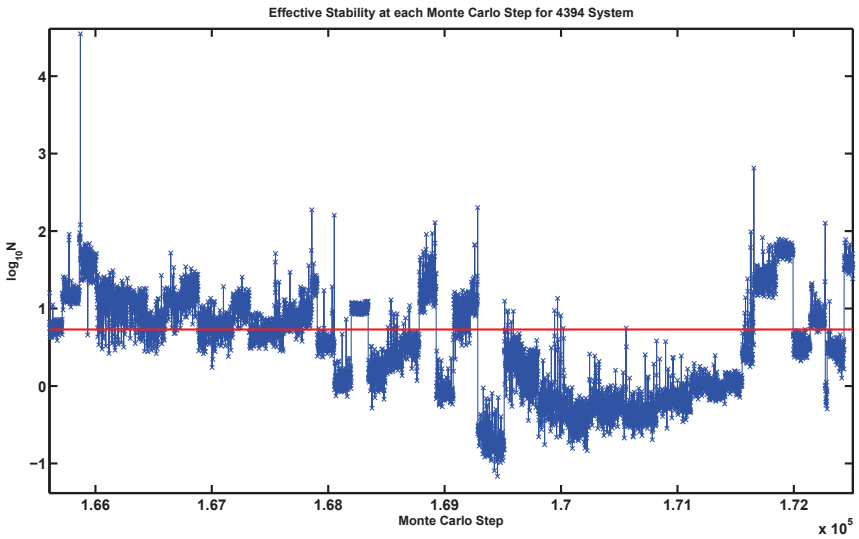


Figure 5.8: Effective stability, N, at each Monte Carlo step for the 4394 system for a representative window of roughly 6000 steps. The horizontal (red) line indicates the ($\log_{10}$ of) the average N (the average N is around 5.36).

For preconditioning, we use the ILUTP preconditioner from SPARSKIT [70]. This preconditioner uses pivoting and a relative threshold to drop (off-diagonal) coefficients based on magnitude to obtain sparse lower triangular and upper triangular factors, L and U, such that $AQ \approx LU$. We set the relative drop tolerance to 0.01 and we allow pivoting in the entire row. We set the permutation tolerance that governs the pivoting to 0.05. Finally, in ILUTP one can set the maximum number of additional nonzeros (fill-in) per row allowed in both the L and U factor. We set this to half the average number of nonzeros per row of the matrix, resulting in at most twice as many nonzeros in the L and U factor together as in the

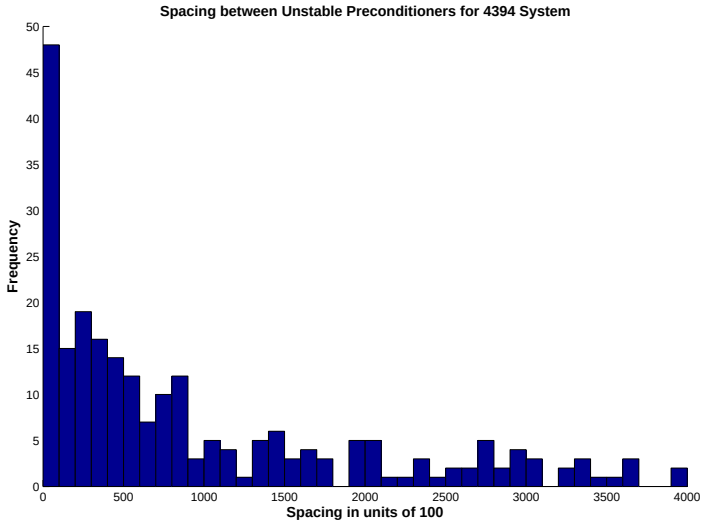


Figure 5.9: Histogram for the number of steps between reorderings (large N or slow convergence) for the 4394 system. The size of each bin is 100.

matrix, A. The average number of nonzeros in the preconditioner remains well below this maximum; see Table 5.4.

The experimental results are given in four tables. In the first two tables, we compare the results of our method to those of the standard method to demonstrate the reliability and accuracy of the new method. In the other two tables, we assess the scaling of our method as the number of particles increases.

First, we assess how close the determinant ratio computed by the new method is to that computed by the standard method. This determines the probability of a *wrong* decision in the acceptance/rejection test. Let q be the exact determinant ratio squared and q_a be the approximate determinant ratio squared computed by an iterative method; then the probability of a wrong decision at one step in the Metropolis algorithm is $f = |min(q, 1) - min(q_a, 1)|$. The average value of f over a random walk (the entire sequence of MCMC steps), $\bar{f}$, gives the expected number of errors in the acceptance/rejection test (Exp. Errors in Table 5.2). We call an approximation

- extremely good if $f < 0.0001$,
- very good if $f < 0.001$, and
- good if $f < 0.01$.

The results of this test for eight successive problem sizes ($2 \cdot 7^3$, $2 \cdot 8^3$, ..., $2 \cdot 14^3$) are given in Table 5.2. The high percentage of extremely good approximations in Table 5.2 shows that approximating determinant ratios does not interfere with the accuracy and reliability of the simulation. In fact, the new algorithm makes a different accept/reject decision from that made by the standard algorithm only once every 10^5 steps. Since the autocorrelation time is smaller than this by over an order of magnitude, the system should quickly forget about this 'incorrect' step. Note that given the tolerance in GMRES the step is not unlikely even in the standard algorithm. We also report the acceptance ratio, which is defined as the ratio of the number of accepts to the total number of Monte Carlo steps. The desired range of the acceptance ratio is between 0.2 and 0.8, and this is satisfied in our simulation. Higher or lower acceptance ratios are likely to create problems, as this typically indicates that successive MC steps remain correlated for a long time [22].

Size	686	1024	1458	2000	2662	3456	4394	5488
Exp. Errors	4.45e-6	4.22e-6	5.03e-6	4.41e-6	4.63e-6	4.50e-6	3.96e-6	4.07e-6
Extr. Good	99.49	99.53	99.57	99.49	99.50	99.47	99.56	99.56
Very Good	99.99	99.98	99.99	99.99	99.99	99.99	99.99	99.99
Good	100	100	100	100	100	100	100	100
Acc. Ratio	0.5879	0.5880	0.5887	0.5880	0.5881	0.5898	0.5878	0.5883

Table 5.2: Accuracy of the acceptance/rejection test, giving the average expected number of errors in the acceptance test (Exp. Errors), the percentage of extremely good approximations (Extr. Good), the percentage of very good approximations (Very Good), the percentage of good approximations (Good), and the acceptance ratio of trial moves (Acc. Ratio).

To further check the accuracy of the results of the simulation, we compute the kinetic energy

of the system (an important observable). The kinetic energy of the system is defined as

$$E = \frac{\hbar^2}{2\rho n} \sum_{ij} \left(6k - 4k^2 ||r_j - Z_i||^2\right) A_{ij} \left(A^{-1}\right)_{ji}, \tag{5.12}$$

where $\hbar$ is the reduced Planck's constant ($h/2\pi$), ρ is the electron mass, n is the system size, $k = 1$ is defined as in (2.13), r_j is the position of particle j, Z_i is the position of orbital i, and A is the Slater matrix as given in (2.12). (We use units where $\hbar = \rho = 1$). It should be noted that we use the exact kinetic energy in this test even though, in practice, our algorithm would also use iterative solvers to efficiently evaluate this observable (which involves the Laplacian of the wave function). This is done so as not to confound two sources of errors (one being a few rare 'incorrect' steps on the Markov chain and the other being errors in computing the kinetic energy). We compute the kinetic energy in two separate experiments. However, in order to emphasize how close the results of the new algorithm are to those of the standard algorithm, *we start both experiments with the same starting configuration and using the same initial seed for the random number generator*. Since the expectation of an incorrect acceptance/rejection is extremely small, both experiments follow the same chain for an extended period of time, and therefore the difference in kinetic energy between the experiments is much smaller than the statistical variation that would be expected if the chains were independent instead of correlated. We compute the kinetic energy by sampling (5.12) at the end of each sweep (doing 120 sweeps and discarding the first 20). The first experiment computes the determinant ratio for acceptance/rejection tests using the standard QMCPACK algorithm, while the second experiment uses the new (sparse) algorithm. The average energy over the whole simulation is listed in Table 5.3. It is evident that the energies from the two algorithms are close for all system sizes. We also compute the standard deviation of the kinetic energy, σ, using DataSpork [23], taking into account the autocorrelation[2].

Next, we analyze the performance of our algorithm and compare the performance experi-

[2]In Markov Chain Monte Carlo, successive states tend to be correlated. The autocorrelation measures how many steps of the algorithm must be taken for states to be uncorrelated [22].

Size	686	1024	1458	2000	2662	3456	4394	5488
Standard QMCPACK Algorithm								
Energy (hartree)	2.0984	2.1074	2.1107	2.1024	2.0964	2.0948	2.1035	2.1016
σ	0.0075	0.0077	0.0040	0.0045	0.0024	0.0028	0.0034	0.0035
Sparse Algorithm								
Energy (hartree)	2.0984	2.1074	2.1107	2.1040	2.1010	2.0999	2.1049	2.1010
σ	0.0075	0.0077	0.0040	0.0034	0.0023	0.0045	0.0033	0.0034

Table 5.3: Kinetic Energy and Standard Deviation.

mentally with that of the standard algorithm.

The computational costs *per sweep*, that is, per n MCMC steps, of the various components of the sparse algorithm are as follows.

- Matrix-vector products in GMRES: $2k_1k_2n^2$, where k_1 is the average number of GMRES iterations per Monte Carlo step (per linear system to solve), and $k_2 = \text{nnz}(A)/n$ (average number of nonzeros (nnz) in A per row).

- Computing ILUTP preconditioners: $\alpha k_2 k_3 s n^2$, where $k_3 \ll 1$ is the number of times the preconditioner is computed per MCMC step, $\alpha = (\text{nnz}(L) + \text{nnz}(U))/\text{nnz}(A)$, and s is the cost per nonzero in the preconditioner of computing the ILUTP preconditioner. Notice that $\alpha \leq 2$ by choice (see above), and effectively is about 1.25 (see Table 5.4). The worst case cost of computing an ILUTP preconditioner with a constant (small) maximum number of nonzeros per row (independent of n) is $O(n^2)$. However, our timings show that, for this problem, the cost is always $O(n)$, which seems to be true in general for the ILUTP preconditioner. The parameter k_3 should be picked to balance the cost of computing the ILUTP preconditioner with the cost of applying the multiplicative updates to the preconditioner. Hence, if computing the preconditioner has linear cost, k_3 should be a constant based on an estimate of the cost of computing the ILUTP preconditioner.

- Applying the preconditioner in GMRES: $2\alpha k_1 k_2 n^2 + 2(k_1/k_3)n^2$, where $(1/2)k_3^{-1}$ is the average number of preconditioner updates in (5.11).

- Matrix reordering: $k_5 k_6 n$, where k_5 is the number of times the reordering is performed per sweep, and k_6 is the average cost of comparisons and swapping rows (or columns) per row. The parameter k_6 can vary from nearly constant to $O(n)$; however, it can be brought down to a constant by a more elaborate implementation. In general, the cost of the reorderings is almost negligible. Moreover, a careful incremental implementation should reduce the overall cost of reordering to $O(n)$ per sweep.

Table 5.4 gives a quick overview of experimental values for the most important parameters. We see that the average number of GMRES iterations, k_1, initially increases slowly but levels off for larger numbers of particles. The numbers of nonzeros in the matrix, k_2, and in the preconditioner, αk_2, are roughly constant with $\alpha \approx 1.25$. Finally, we see that the number of reorderings per sweep, k_5, increases slowly.

Size	686	1024	1458	2000	2662	3456	4394	5488
nr. GMRES iter.	8.91	9.34	9.53	9.59	9.83	10.20	10.22	10.10
nnz(A)/n	42.38	42.38	42.39	42.38	42.37	42.37	42.37	42.37
nnz(L+U)/n	55.04	54.48	54.33	53.61	53.27	53.28	53.28	53.01
nr. reorder/sweep	0.65	1.03	1.19	1.73	2.23	2.63	2.73	3.12

Table 5.4: Analysis of computational cost, providing for each problem size the average number of GMRES iterations per linear system solve (k_1), the average number of nonzeros per row in the matrix A (k_2), the average number of nonzeros (per row) in the L and U factors together (αk_2), and the average number of matrix reorderings per sweep (k_5).

Based on the numbers in Table 5.4 and the complexity analysis above, we expect the experimental complexity of the new algorithm (for the values of n used) to be slightly worse than $O(n^2)$. However, if the average number of GMRES iterations (k_1) remains bounded for increasing n, as suggested by the table, and we change the reordering algorithm to a version that is $O(n)$ per reordering, we have an $O(n^2)$ algorithm.

We compare the average runtimes per sweep of VMC using QMCPACK with the sparse

Size	686	1024	1458	2000	2662	3456	4394	5488
std alg. (s)	2.52	7.18	16.09	36.83	81.27	173.24	340.94	649.94
sparse alg. (s)	5.71	12.18	24.67	47.55	86.11	167.43	312.05	549.17

Table 5.5: Scaling Results: the average runtime of a VMC sweep using QMCPACK with its standard algorithm for determinant ratios and with the new sparse algorithm for determinant ratios.

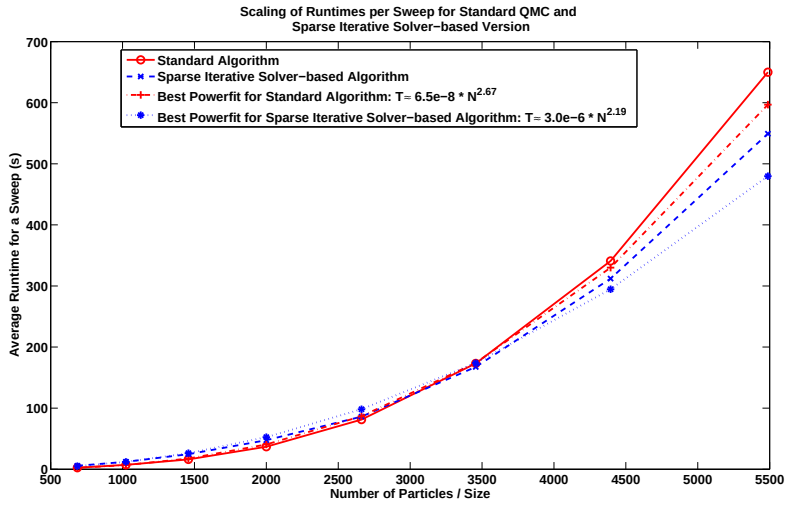


Figure 5.10: The scaling of the runtime for a sweep (including all computations) for QMCPACK with its standard algorithm for determinant ratios and with the new sparse algorithm. The figure also shows the best power-law approximations for both versions.

algorithm with the average runtimes of QMCPACK with the standard algorithm for eight problems sizes in Table 5.5 and Figure 5.10. This comparison includes all parts of the simulation, updating the Slater matrix, computing (some) observables, and the various parts of computing the determinant ratios discussed above. For QMCPACK with the sparse algorithm, the percentage of runtime spent in the linear solver ranges from about 85% for the smallest system to 95% and higher for the larger systems. For QMCPACK with the standard algorithm the percentages are even higher. We see that the break-even point for the new algorithm (for this problem) occurs for about 3000 particles. Fitting both sequences of runtimes to a power law (an^b), we find that QMCPACK scales as $O(n^{2.67})$ with the standard algorithm and as $O(n^{2.19})$ with the sparse algorithm. The exponent for QMCPACK with the sparse algorithm is slightly larger than 2, which is partly explained by the slow increase

in the number of GMRES iterations and in the number of reorderings, although the former appears to level off for larger n. However, we suspect that the exponent is also partly due to cache and other hardware effects. The exponent for QMCPACK with the standard algorithm depends on all components of the algorithm (not just the $O(n^3)$ computation of determinant ratios); it is noteworthy that it is close to 3 already for these system sizes. We discuss how we intend to bring the scaling for the sparse algorithm down further in future work. We remark that based on a straightforward complexity analysis, the scaling for the standard algorithm will approach $O(n^3)$ for large enough n.

Chapter 6

Conclusion and Future Work

The focus of this dissertation has been on efficiently solving sequences of single and dual linear systems by recycling Krylov subspaces and preconditioners. Hence, in this chapter we first provide concluding remarks for these approaches.

Our primary application areas have been model reduction and the Variational Monte Carlo (VMC) method. Our secondary application areas include convection-diffusion type problems and study of crack propagation. While recycling preconditioners for the VMC method we current use an `ILU` type preconditioner, which is not problem specific. Hence, we introduce a multilevel type preconditioner for the VMC method that exploits the structure of the underlying problem. Preliminary results with this preconditioner look promising.

Recycling Krylov Subspaces

For several classes of problems, such as solving the linear systems arising in interpolatory model reduction, or computing bilinear forms arising in Variational Monte Carlo methods, the BiCG algorithm has advantages over methods like GMRES that would solve the primary and the dual system separately. For sequences of dual linear systems arising in such problems, it is advantageous to use Krylov subspace recycling for the BiCG algorithm, and for this

purpose we propose the RBiCG algorithm. The derivation of RBiCG also provides the foundation for recycling variants of other popular bi-Lanczos based methods, like CGS, BiCGSTAB, BiCGSTAB2, BiCGSTAB(l), QMR, and TFQMR. Hence, we propose RCGS, RBiCGSTAB, and RBiCGSTAB2, which focus on efficiently solving sequences of single linear systems.

We have demonstrated the usefulness of RBiCG for interpolatory model reduction using IRKA. In addition, we have analyzed and demonstrated the effectiveness of RBiCG for nonsymmetric linear systems arising from convection-diffusion problems. This suggests that the RBiCG method may be useful in other areas where solving dual systems in a Petrov-Galerkin sense brings special advantages, for example, in computing bilinear forms for the VMC method. We have demonstrated usefulness of RBiCGSTAB on a crack propagation example.

In future work, we plan to extend the use of RBiCG to model reduction for MIMO dynamical systems in a tangential interpolation framework, where the right-hand sides are not constant as in the SISO case. In addition, we will further investigate the use of RBiCG for evaluating bilinear forms arising in VMC algorithms. Our current results for this look promising. We also plan to test RBiCGSTAB and RBiCGSTAB2 on acoustics problems, and extend the recycling framework of RBiCG to RBiCGSTAB(l).

Recycling Preconditioners

We present an efficient algorithm for simulating insulators with the VMC method for large numbers of particles. Our algorithm reduces the scaling of computing the determinant ratios, the dominant computational cost for large n, from $O(n^3)$ to slightly worse than $O(n^2)$, where n is the number of particles. This complements recent improvements in the scaling of constructing Slater matrices. Our main contribution is a method to compute *efficiently* for the Slater matrices a sequence of preconditioners that make the iterative solver *converge rapidly*. This involves cheap preconditioner updates, an effective reordering strategy, and

a cheap method to monitor instability of the ILUTP preconditioners. Furthermore, we demonstrate experimentally that we can achieve the improved scaling without sacrificing accuracy. Our results show that preconditioned iterative solvers can reduce the cost of VMC for large(r) systems.

There are several important improvements to be explored for our proposed algorithm. First, we will implement an $O(n)$ reordering algorithm. This is important for larger system sizes. We will also consider more elaborate matrix reordering strategies like those in [32, 33]. Second, we intend to develop an incremental local reordering scheme that satisfies certain optimality properties. This should allow us to cheaply update the matrix (probably at constant cost) every Monte Carlo step or every few steps and exploit the fact that particle updates are strictly local. Potentially, the field of computational geometry might provide some insights for this effort.

Third, although the ILUTP preconditioner leads to fast convergence, it is not obvious how to update an ILUTP preconditioner in its LU form. Again, the current approach with cheap intermediate, multiplicative updates by rank-one updates to the identity is effective, but it would be better to have preconditioners that can be updated continually with constant cost, local updates, and that adapt in a straightforward manner to a reordering of the matrix (at constant cost). We do not know if such preconditioners exist and whether they would yield fast convergence. In future work, we will explore forms of preconditioning that might satisfy these requirements and an underlying theory of preconditioners for Slater matrices (see the next section). The latter would also include analyzing the matrix properties of Slater matrices.

Fourth, an interesting experiment is to check whether replacing the (sparse) iterative solver by a sparse direct solver might be advantageous for certain problem sizes. We expect that for small problems the standard algorithm is fastest and that for large(r) problems sparse iterative solvers are fastest. However, there might be a range of physically relevant problem sizes for which sparse direct solvers are the best. Fifth, we will extend our algorithm to other

types of orbitals. One specific goal will be to adapt our algorithm to achieve quadratic scaling when single particle orbitals are delocalized. Note that the optimization of orbitals [4, 31, 86] discussed in Section 2.2 leads to rapidly decaying orbitals for many systems. Finally, we will test our algorithm for realistic materials and much larger system sizes.

Multilevel Preconditioner

To further improve the scaling of the VMC algorithm, we present a multilevel preconditioner (instead of the `ILUTP` preconditioner that we currently use). We may interpret Slater matrices as interpolation matrices. If we consider the single particle orbitals $\{\phi_1(x), \ldots, \phi_n(x)\}$ as a set of basis functions, and approximate $f(x)$ using these, then

$$f(x) \simeq \sum_{k=1}^{n} \phi_k(x)\alpha_k, \tag{6.1}$$

where the $\{\alpha_k\}$ are scalars to be determined. Interpolating f at the particle positions $\{r_1, \ldots, r_n\}$ gives

$$\begin{aligned}
\phi_1(r_1)\alpha_1 + \phi_2(r_1)\alpha_2 + \ldots + \phi_n(r_1)\alpha_n &= f(r_1),\\
\phi_1(r_2)\alpha_1 + \phi_2(r_2)\alpha_2 + \ldots + \phi_n(r_2)\alpha_n &= f(r_2),\\
&\vdots\\
\phi_1(r_n)\alpha_1 + \phi_2(r_n)\alpha_2 + \ldots + \phi_n(r_n)\alpha_n &= f(r_n).
\end{aligned}$$

or

$$\tilde{\alpha} = A^{-1}\tilde{f},$$

where $\tilde{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ and $\tilde{f} = (f(r_1), f(r_2), \ldots, f(r_n))^T$. Recall from (2.9) and (5.3) that if at a MCMC step particle i is moved, then we are trying to approximate

$$u^T A^{-1} e_i = u^T z. \tag{6.2}$$

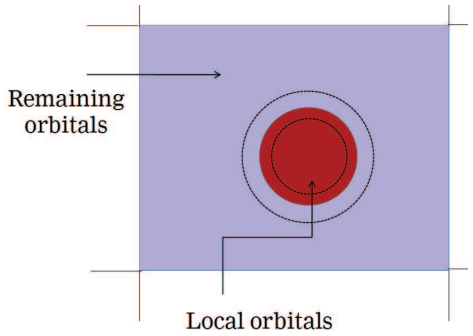


Figure 6.1: Local interpolation and global correction

Thus, computing $z = A^{-1}e_i$ corresponds to interpolating a function f such that $f(r_i) = 1$ and $f(r_j) = 0$ for $j \neq i$. Due to the local support of the basis functions (see Section 2.2), we may compute $A_{11}^{-1}(e_i)_{1\ldots m}$ using a local interpolation. When a particle is selected for a random move at any MCMC step, we first find the closest orbital to this particle. We then pick all the orbitals close to this orbital in a certain radius (local orbitals; see Figure 6.1), and identify the particle closest to each of these selected orbitals. We reorder the selected particles and orbitals such that their interaction represents the A_{11} sub-matrix below[1],

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} e_i \\ 0 \end{pmatrix}.$$

Thus, (6.2) can be rewritten as $u^T z = u_1^T z_1 + u_2^T z_2$. The $u_1^T z_1$ component comes from the local orbitals, and we perform an exact *local interpolation* to compute it's contribution. The $u_2^T z_2$ component is related to the remaining orbitals, and we approximate it's contribution via a *global correction*. Since the number of local orbitals is much smaller than the total number of orbitals, this approach provides the determinant ratio cheaply without substantial loss of accuracy (see the results).

We implement this approach via a two-step fixed point iteration. We first perform the following initializations: $z_1^{(0)} = 0, z_2^{(0)} = 0$, $\rho_1^{(0)} = e_i$, and $\rho_2^{(0)} = 0$, where ρ_i denote the

[1]In practice we do not explicitly swap rows and columns of the Slater matrix. Instead, we keep track of indices of particles and orbitals that represent the A_{11} sub-matrix.

residuals. We then perform the local interpolation, which has a constant cost.

$$\begin{aligned} z_1^{(1)} &= z_1^{(0)} + A_{11}^{-1}\rho_1^{(0)} = A_{11}^{-1}e_i, & z_2^{(1)} &= z_2^{(0)} = 0, \\ \rho_1^{(1)} &= e_i - A_{11}z_1^{(1)} - \underbrace{A_{12}z_2^{(1)}}_{=0} = 0, & \rho_2^{(1)} &= 0 - A_{21}z_1^{(1)} - \underbrace{A_{22}z_2^{(1)}}_{=0} = -A_{21}A_{11}^{-1}e_i. \end{aligned}$$

Finally, we perform an approximate correction for the global problem.

$$\begin{aligned} z_1^{(2)} &= z_1^{(1)}, & z_2^{(2)} &= z_2^{(1)} + \tilde{A}_{22}\rho_2^{(1)}. \\ \rho_1^{(2)} &= \underbrace{e_i - A_{11}z_1^{(2)}}_{=\rho_1^{(1)}} - A_{12}z_2^{(2)}, & \rho_2^{(2)} &= \underbrace{0 - A_{21}z_1^{(2)}}_{=\rho_2^{(1)}} - A_{22}z_2^{(2)}. \end{aligned}$$

Here $\tilde{A}_{22} \simeq A_{22}^{-1}$, and is obtained by some approximation (for example a zero–fill incomplete LU factorization [70]). Combining the above two steps of the fixed point iteration into one, and generalizing for any k^{th} Monte Carlo step we get

$$\underbrace{\begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}}_{\text{M}} \begin{pmatrix} z_1^{(k+1)} \\ z_2^{(k+1)} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & -A_{12} \\ 0 & 0 \end{pmatrix}}_{\text{N}} \begin{pmatrix} z_1^{(k)} \\ z_2^{(k)} \end{pmatrix} + \underbrace{\begin{pmatrix} e_i \\ 0 \end{pmatrix}}_{\text{b}}$$

This gives us the splitting $Mz^{(k+1)} = Nz^{(k)} + b$, where $A = M - N$. The final preconditioned linear system, obtained after applying fixed point iteration twice, is given by

$$\left(2M^{-1} - M^{-1}AM^{-1}\right) Az = \left(2M^{-1} - M^{-1}AM^{-1}\right) b,$$

where $M^{-1} = \begin{pmatrix} A_{11}^{-1} & 0 \\ -\tilde{A}_{22}A_{21}A_{11}^{-1} & \tilde{A}_{22} \end{pmatrix}$.

We give some preliminary results in Table 6.1. We start with an equilibrium configuration, and at every MCMC step we use Matlab's `ILU(0)` to approximate the inverse of A_{22}. From the results we see that the GMRES iteration count is very low, and hence the method is cheap. The approximations are good but not excellent. We expect the approximations to

System Size	A_{11} Size	Rate of Decay of Orbitals	Number of MCMC steps	Average GMRES Iterations	% of Good Steps
1024	27	k = 1	507	3.25	76.23

Table 6.1: Using a multilevel preconditioner for VMC. For the definition of a good step, see Section 5.3.

get better with a larger system size.

Bibliography

[1] A. M. Abdel-Rehim, A. Stathopoulos, and K. Orginos. Extending the eigCG algorithm to non-symmetric Lanczos for linear systems with multiple right-hand sides. Technical Report WM-CS-2009-06, College of William and Mary, 2009.

[2] A. M. Abdel-Rehim, W. Wilcox, and R. B. Morgan. Deflated BiCGStab for linear equations in QCD problems. In *Proceedings of Science, LAT2007*, pages 026/1–026/7, 2007.

[3] K. Ahuja. Recycling bi-Lanczos algorithms: BiCG, CGS, and BiCGSTAB. Master's thesis, Department of Mathematics, Virginia Tech, 2009. Advised by E. de Sturler. Available from `http://scholar.lib.vt.edu/theses/available/etd-08252009-161256/`.

[4] D. Alfé. Order(N) methods in QMC. 2007 Summer School on Computational Materials Science, http://www.mcc.uiuc.edu/summerschool/2007/qmc/.

[5] D. Alfé and M. J. Gillan. An efficient localized basis set for Quantum Monte Carlo calculations on condensed matter. *Physical Review B (Rapids)*, 70:161101 (1–4), 2004.

[6] D. Alfé and M. J. Gillan. Linear-scaling Quantum Monte Carlo with non-orthogonal localized orbitals. *Journal of Physics: Condensed Matter*, 16:L305–L311, 2004.

[7] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[8] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems (Advances in Design and Control)*. SIAM, Philadelphia, PA, USA, 2005.

[9] A. C. Antoulas, C. A. Beattie, and S. Gugercin. Interpolatory model reduction of large-scale dynamical systems. In J. Mohammadpour and K. Grigoriadis, editors, *Efficient Modeling and Control of Large-Scale Systems*. Springer-Verlag, 2010.

[10] Z. Bai, W. Chen, R. Scalettar, and I. Yamazaki. Numerical methods for Quantum Monte Carlo simulations of the Hubbard model. In T. Y. Hou, C. Liu, and J.-G. Liu, editors, *Multi-Scale Phenomena in Complex Fluids, ISBN-9787040173581*. Higher Education Press, China, February 2009. An early version appeared as Technical Report

CSE-2007-36, Department of Computer Science, UC Davis, Dec.4, 2007 and revised on Feb.25, 2008.

[11] R. E. Bank and T. F. Chan. An analysis of the composite step biconjugate gradient method. *Numer. Math.*, 66:295–319, 1993.

[12] C. A. Beattie. Harmonic Ritz and Lehmann bounds. *Electronic Transactions on Numerical Analysis*, 7:18–39, 1998.

[13] C. A. Beattie and S. Gugercin. Krylov-based minimization for optimal $\mathcal{H}_2$ model reduction. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4385–4390, 2007.

[14] C. A. Beattie and S. Gugercin. A trust region method for optimal $\mathcal{H}_2$ model reduction. In *Proceedings of 48th IEEE Conference on Decision & Control and 28th Chinese Control Conference*, pages 5370–5375, 2009.

[15] C. A. Beattie, S. Gugercin, and S. Wyatt. Inexact solves in interpolatory model reduction. *Linear Algebra and its Applications (accepted)*, 2010. Available as arXiv:1007.5213v2.

[16] P. Benner. Solving large-scale control problems. *IEEE Control Systems Magazine*, 24(1):44–59, 2004.

[17] P. Benner and L. Feng. On recycling Krylov subspaces for solving linear systems with successive right-hand sides with applications in model reduction. In P. Benner, M. Hinze, and E. J. W. ter Maten, editors, *Model Reduction for Circuit Simulation*, volume 74 of *Lecture Notes in Electrical Engineering*. Springer-Verlag, 2011 (in press).

[18] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418 – 477, 2002.

[19] M. Benzi, D. B. Szyld, and A. van Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing*, 20(5):1652–1670, 1999.

[20] L. Bergamaschi, R. Bru, A. Martinez, and M. Putti. Quasi-Newton preconditioners for the inexact Newton method. *Electronic Transaction on Numerical Analysis*, 23:76–87, 2006. http://etna.mcs.kent.edu/vol.23.2006/pp76-87.dir/pp76-87.pdf.

[21] J. P. Boyd and K. W. Gildersleeve. Numerical experiments on the condition number of the interpolation matrices for radial basis functions. *Applied Numerical Mathematics*, 61:443–459, 2011.

[22] D. Calvetti and E. Somersalo. *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing*, volume 2 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer, 2007.

[23] D. M. Ceperley. DataSpork analysis toolkit. Materials Computation Center at the University of Illinois at Urbana-Champaign, http://www.mcc.uiuc.edu/dataspork/.

[24] D. M. Ceperley, G. V. Chester, and M. H. Kalos. Monte Carlo simulation of a many-fermion system. *Phys. Rev. B*, 3081(16):3081–3099, 1977.

[25] M. P. Chernesky. On preconditioned Krylov subspace methods for discrete convection-diffusion problems. *Numerical Methods for Partial Differential Equations*, 13(4):321–330, 1997.

[26] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *Journal of Computational and Applied Mathematics*, 86(2):387 – 414, 1997.

[27] E. de Sturler. *Iterative Methods on Distributed Memory Computers*. PhD thesis, Delft University of Technology, 1994.

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

[29] E. de Sturler. BiCG explained. In *Householder Symposium XIV, Proceedings of the Householder International Symposium in Numerical Algebra, Chateau Whistler, Whistler, BC, Canada*, June 13–19, 1999.

[30] C. De Villemagne and R. E. Skelton. Model reductions using a projection formulation. *International Journal of Control*, 46(6):2141–2169, 1987.

[31] N. Drummond and P. L. Rios. Worksheet 1: Using localized orbitals in QMC calculations. 2007 Summer School on Computational Materials Science, http://www.mcc.uiuc.edu/summerschool/2007/qmc/.

[32] I. S. Duff and J. Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 20(4):889–901, 1999.

[33] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM Journal on Matrix Analysis and Applications*, 22(4):973–996, 2001.

[34] J. M. Edmunds. Input and output scaling and reordering for diagonal dominance and block diagonal dominance. *IEE Proceedings - Control Theory and Applications*, 145(6):523–530, 1998.

[35] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for non-symmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[36] L. Feng, P. Benner, and J. Korvink. Parametric model order reduction accelerated by subspace recycling. In *Proceedings of 48th IEEE Conference on Decision & Control and 28th Chinese Control Conference*, pages 4328–4333, 2009.

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

[38] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. Quantum Monte Carlo simulations of solids. *Rev. Mod. Phys.*, 73(1):33–83, 2001.

[39] D. Frenkel and B. Smit. *Understanding Molecular Simulation, Second Edition: From Algorithms to Applications*, volume 1 of *Computational Science Series*. Academic Press, 2002.

[40] R. W. Freund. Solution of shifted linear systems by quasi-minimal residual iterations. In L. Reichel, A. Ruttan, and R. S. Varga, editors, *Numerical Linear Algebra*, pages 101–121. W. de Gruyter, 1993.

[41] R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM Journal on Scientific Computing*, 14(2):470–482, 1993.

[42] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM Journal on Scientific Computing*, 14(1):137–158, 1993.

[43] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[44] A. Frommer. BiCGStab(ℓ) for families of shifted linear systems. *Computing*, 70:87–109, 2003.

[45] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[46] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, 1997.

[47] E. Grimme. *Krylov projection methods for model reduction*. PhD thesis, University of Illinois, 1997.

[48] S. Gugercin. An iterative rational Krylov algorithm (IRKA) for optimal $\mathcal{H}_2$ model reduction. In *Householder Symposium XVI*, Seven Springs Mountain Resort, PA, USA, May 2005.

[49] S. Gugercin, A. C. Antoulas, and C. A. Beattie. $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems. *SIAM Journal on Matrix Analysis and Applications*, 30(2):609–638, 2008.

[50] M. H. Gutknecht. Variants of BICGSTAB for matrices with complex spectrum. *SIAM Journal on Scientific and Statistical Computing*, 14:1020–1033, 1993.

[51] M. H. Gutknecht. Lanczos-type solvers for nonsymmetric linear systems of equations. *Acta Numerica*, 6:271–397, 1997.

[52] B. Hammond, W. A. Lester, and P. J. Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. World Scientific, Singapore, 1994.

[53] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[54] J. Kim et al. QMCPACK. Materials Computation Center at the University of Illinois at Urbana-Champaign, http://cms.mcc.uiuc.edu/qmcpack/.

[55] B. Jegerlehner. Krylov space solvers for shifted linear systems. *Hep-lat/9612014*, 1996.

[56] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. Society for Industrial and Applied Mathematics, 2003.

[57] M. E. Kilmer and E. de Sturler. Recycling subspace information for diffuse optical tomography. *SIAM Journal on Scientific Computing*, 27(6):2140–2166, 2006.

[58] J. Korvink and E. Rudnyi. Oberwolfach benchmark collection. In P. Benner, V. Mehrmann, and D. C. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 311–315. Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[59] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49:33–53, 1952.

[60] W. McMillan. Ground state of liquid helium^4. *Physical Review*, 138(2A):A442 – A451, April 1965.

[61] L. Meier III and D. Luenberger. Approximation of linear constant systems. *IEEE Transactions on Automatic Control*, 12(5):585–588, 1967.

[62] L. Mitas and J. C. Grossman. Quantum Monte Carlo study of Si and C molecular systems. *in Recent Advances in Quantum Monte Carlo Methods, W. A. Lester (Ed.)*, pages 133–162, 1997.

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

[64] R. B. Morgan and D. A. Nicely. Restarting the nonsymmetric Lanczos algorithm. Unpublished manuscript, 2010.

[65] P. K. V. V. Nukala and P. R. C. Kent. A fast and efficient algorithm for Slater determinant updates in Quantum Monte Carlo simulations. *J. Chem. Phys.*, 130(204105), 2009.

[66] 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.

[67] T. Penzl. Algorithms for model reduction of large dynamical systems. *Linear Algebra and its Applications*, 415(2–3):322–343, 2006. Special Issue on Order Reduction of Large-Scale Systems.

[68] A. Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. II. matrix pairs. *Linear algebra and its Applications*, 197–198:283–295, 1994.

[69] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numer. Linear Algebra Appl.*, 1:387–402, 1994.

[70] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 3600 Market Street, Philadelphia, PA 19104-2688, USA, 2nd edition, 2003.

[71] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[72] J. Saak and P. Benner. Efficient numerical solution of the LQR-problem for the heat equation. In *Proceedings in Applied Mathematics and Mechanics*, volume 4, pages 648–649, 2004.

[73] R. T. Scalettar, K. J. Runge, J. Correa, P. Lee, V. Oklobdzija, and J. L. Vujic. Simulations of interacting many body systems using p4. *International Journal of High Speed Computing*, 7(3):327–349, 1995.

[74] G. L. G. Sleijpen and D. R. Fokkema. BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numerical Analysis*, 1:11–32, 1993.

[75] 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(2):401–425, 1996.

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

[77] J. T. Spanos, M. H. Milman, and D. L. Mingori. A new algorithm for L^2 optimal model reduction. *Automatica*, 28(5):897–909, 1992.

[78] A. Stathopoulos and K. Orginos. Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics. *SIAM Journal on Scientific Computing*, 32(1):439–462, 2010.

[79] Z. Strakoš and P. Tichý. On efficient numerical approximation of the bilinear form $c^*A^{-1}b$. *SIAM Journal on Scientific Computing*, 33(2):565–587, 2011.

[80] J. Thijssen. *Computational Physics*. Cambridge University Press, 1999.

[81] J. van den Eshof and G. L. G. Sleijpen. Accurate conjugate gradient methods for families of shifted systems. *Applied Numerical Mathematics*, 49(1):17–37, 2004.

[82] 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.

[83] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[84] P. Van Dooren, K. A. Gallivan, and P.-A. Absil. $\mathcal{H}_2$-optimal model reduction of MIMO systems. *Applied Mathematics Letters*, 21(12):1267–1273, 2008.

[85] 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.

[86] A. J. Williamson, R. Q. Hood, and J. C. Grossman. Linear-scaling Quantum Monte Carlo calculations. *Phys. Rev. Lett.*, 87(24):246406, 2001.

[87] D. A. Wilson. Optimum solution of model-reduction problem. *Proc. of IEE*, 117(6):1161–1165, 1970.

[88] A. Yousuff and R. E. Skelton. Covariance equivalent realizations with application to model reduction of large scale systems. In C. T. Leondes, editor, *Control and Dynamic Systems*, volume 22, pages 273–348. Academic Press, New York, NY, 1985.

[89] A. Yousuff, D. A. Wagie, and R. E. Skelton. Linear system approximation via covariance equivalent realizations. *Journal of Mathematical Analysis and Applications*, 106(1):91–115, 1985.

[90] 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.