Thesis Title Add later I forgot what title I submitted for my thesis along with my graduation request

Author Name: Hongda Li

A thesis presented for the degree of: ????

Department of Applied Mathematics
University of Washington

Abstract

We review results from the literature on the conjugate gradient algorithm for solving symmetric positive definite linear systems and the related Lanczos algorithm. We derive the conjugate gradient algorithm from the more general conjugate direction method, using projectors. We establish error bounds using exact arithmetic theory and also discuss what can happen when floating point arithmetic is used. We present numerical experiments to illustrate this behavior.

Contents

1 Notations

1. $\text{ran}(A) := \{Ax : \forall x \ \in \mathbb{R}^n\}, A \in \mathbb{R}^{m\times n}$, The range of a matrix.
2. $(A)_{i,j}$: The element in ith row and jth column of the matrix A.
3. $(A)_{i:i',j:j'}$: The submatrix whose top left corner is the (i, j) element in matrix A, and whose' right bottom corner is the (i', j') element in the matrix A. The notation is similar to MATLAB's rules for indexing.
4. $\forall\ 0 \le j \le k$: under certain context it indicates the range for an index: $j = 0, 1, \cdots, k-1, k$
5. Boldface $\mathbf{0}$ denotes the zero vector or matrix, depending on the context it can be either a zero row/column vector, or a zero matrix.
6. The $\hat{}$ decorator is reserved for denoting the unit vector of some non zero vector. For example $\hat{x} := x/\|x\|, x \neq \mathbf{0}$.
7. $p_k(A|w)$ denotes the matrix polynomial $\sum_{j=0}^{k} w_j A^j$.
8. ξ_i are used for the i standard basis vector, the size of the vector depends on the context, sometimes it's denoted without ambguity for example: $\xi_i^{(k)}$ would denote the i th standard basis vector for $\mathbb{R}^k$.

2 Introduction

The Conjugate Gradient method is an iterative method used for solving symmetric positive definite linear systems. It dates back to the period when computers were programmed using punched cards. It didn't receive much attention at the start but was revised and reappeared as a method for solving large sparse linear systems decades later, becoming the best option for positive definite linear systems that are sparse and large and, by extension, for optimizing strongly convex functions as well. In this thesis, we discuss the Conjugate Gradient method without pre-conditioning by deriving it and analyzing it along with the Lanczos algorithm, a closely related algorithm for solving symmetric eigenproblems. Finally, we use their connections to analyze their behaviors under floating-point arithmetic. The thesis will require some background in numerical linear algebra for the best understanding.

In the first section, we introduce projectors and subspace projection methods. We then specialize to Krylov subspaces and demonstrate that the conjugate gradient method can be thought of as producing an oblique projection onto a Krylov subspace. We also derive the Lanczos algorithm as the symmetric version of the more general Arnoldi algorithm. In the second section, we establish the well-known relationship between these two algorithms and we derive bounds on the convergence rate of the conjugate gradient algorithm. In the third section, we use numerical experiments to better understand the algorithm behaviors under floating-point arithmetic, and we discuss possible ways to mitigate the effects of floating-point arithmetic, as well as what to expect if these effects are not mitigated.

Acknowledgement Thanks to Professor Greenbaum for taking the time to review the first draft of this thesis and for her help with my research throughout the year. Thanks to my old undergraduate classmate and friend who helped with fixing the typos on the later drafts.

3 Foundations

In this section, we go over the foundations of the Conjugate Gradient and the Lanczos algorithms. We introduce the important ideas at the beginning, and then we proceed to derive the conjugate gradient algorithm from the method of conjugate directions. We then derive the Lanczos algorithm as a symmetric case of the Arnoldi iteration.

3.1 The Basics

In this subsection, we go over some basic concepts and mathematical entities that are important to Subspace Projection methods in general.

3.1.1 Krylov Subspace

Definition 1 (Krylov Subspace)**.**

$$\mathcal{K}_k(A|b) = \text{span}(b, Ab, A^2b, \cdots A^{k-1}b)$$

Observe that every element in the subspace is the product of a matrix polynomial with the vector b; we write it as $p_{k-1}(A|w)b$, where p_{k-1} is a $(k-1)^{st}$ degree polynomial and w is a vector denoting the coefficients.

Definition 2 (The Grade of Krylov Subspace)**.** The grade of the Krylov subspace for matrix A and vector b is $k-1$ where k is the smallest k such that the vectors in $\mathcal{K}_k(A|b)$ are linearly dependent. This will be denoted as grade$(A|b)$. Alternatively, it is also the degree of the minimal polynomial $p(A)$ such that $p(A)b = \mathbf{0}$.

The terminology "grade of a Krylov subspace" is used in Y. Saad's work[10]. Once the grade is reached, the Krylov subspace becomes an invariant subspace for the matrix A. For a proof, see Krylov Subspace Grade Invariant Theorem (B.1) in the appendix.

Proposition 3.1 (When the Grade is Reached)**.** Assuming that matrix A is diagonalizable, $A = V\Lambda V^{-1}$, then grade$(A|u)$ is the number of unique λ_i such that $(V^{-1}u)_i$ is non-zero.

Proof. Let $\mathcal{K}_{k+1}(A|u)$ be linearly dependent, then there is a nonzero vector w such that:

$$\mathbf{0} = \sum_{j=0}^{k} w_j A^j u \tag{3.1.1}$$

$$\mathbf{0} = V \sum_{j=0}^{k} w_j \Lambda^j V^{-1} u \tag{3.1.2}$$

$$\forall i \quad 0 = \left(\sum_{j=0}^{k} w_j \lambda_i^j\right) (V^{-1}u)_i \tag{3.1.3}$$

It follows that $\sum_{j=0}^{k} w_j \lambda_i^j = 0$ whenever $(V^{-1}u)_i \neq 0$. If there are more than k indices i, corresponding to distinct eigenvalues λ_i, for which $(V^{-1}u)_i \neq 0$, then the only vector w for which the above equations will be satisfied is $w = \mathbf{0}$. This is because the $k+1$ by $k+1$ matrix whose $(i, j+1)$ entry is λ_i^j is a Vandermonde matrix and hence is nonsingular. However, if there are k such indices i, then there will be a nonzero vector w that satisfies the above k equations in the $k+1$ unknowns, $w_0, \ldots, w_k$. If there are fewer than k nonzero entries of $(V^{-1}u)_i$ corresponding to distinct eigenvalues λ_i, then, by the same arguments, the grade will be less than k. □

3.1.2 Projectors

Definition 3. A matrix P is a projector when $P^2 = P$, we call this property idempotent.

There are two types of projectors, oblique and orthogonal projectors. A projector is an orthogonal projector when it's Hermitian and oblique when it's not Hermitian.

Proposition 3.2 (Projector Complementary)**.** The projector $I - P$ projects onto the null space of P and vice versa.

$$\operatorname{ran}(P) = \operatorname{null}(I - P) \tag{3.1.4}$$

$$\operatorname{ran}(I - P) = \operatorname{null}(P) \tag{3.1.5}$$

The proof is immediate from the definition. For more coverage of facts, refer to Trefethen's Book on Numerical Linear Algebra[14].

3.2 Subspace Projection Methods

Let $\mathcal{K}, \mathcal{L}$ be two subspaces of $\mathbb{R}^n$. We will choose approximate solutions to our linear system $Ax = b$ from $\mathcal{K}$, and we will orthogonalize the residual $b - A\tilde{x}$ against $\mathcal{L}$. This is a description of this framework:

$$\text{choose } \tilde{x} \in x_0 + \mathcal{K} \text{ s.t: } b - A\tilde{x} \perp \mathcal{L}. \tag{3.2.1}$$

Let the columns of $V \in \mathbb{R}^{n \times m}$ be a basis for $\mathcal{K}$ and let the columns of $W \in \mathbb{R}^{n \times m}$ be a basis for $\mathcal{L}$. Then

$$\tilde{x} = x_0 + Vy \tag{3.2.2}$$

$$\text{choose } x \text{ s.t: } b - A\tilde{x} \perp \operatorname{ran}(W) \tag{3.2.3}$$

$$\implies W^T(b - Ax_0 - AVy) = \mathbf{0} \tag{3.2.4}$$

$$W^T r_0 - W^T AVy = \mathbf{0} \tag{3.2.5}$$

$$W^T AVy = W^T r_0 \tag{3.2.6}$$

Thus we can determine the approximate solution $\tilde{x}$ by solving the linear system $W^T AVy = W^T r_0$ for y (assuming that the matrix $W^T AV$ is nonsingular) and then setting $\tilde{x} = x_0 + Vy$. The new residual is $\tilde{r} = b - A\tilde{x} = b - Ax_0 - AVy = r_0 - AV(W^T AV)^{-1} W^T r_0$, and the matrix $AV(W^T AV)^{-1} W^T$ is a projection since

$$[AV(W^T AV)^{-1} W^T][AV(W^T AV)^{-1} W^T] = AV(W^T AV)^{-1} W^T \tag{3.2.7}$$

Alternatively, for some symmetric positive definite matrix B, one might choose $\tilde{x} = x_0 + Vy$ to minimize the B-norm of the residual $\|r_0 - AVy\|_B = \langle r_0 - AVy, B(r_0 - AVy)\rangle^{1/2}$. Setting the gradient of this function to zero leads to the normal equations:

$$V^T A^T BAVy = V^T A^T Br_0 \tag{3.2.8}$$

If A itself is symmetric and positive definite, then we can take $B = A^{-1}$ and minimize the A^{-1}-norm of the residual or, equivalently, the A-norm of the error $\langle A^{-1}b - \tilde{x}, A(A^{-1}b - \tilde{x})\rangle$. The formula for y then becomes

$$V^T AVy = V^T r_0 \tag{3.2.9}$$

This is what the conjugate gradient algorithm does, taking the columns of V to be an orthonormal basis of the Krylov space $\mathcal{K} = \mathcal{K}(A|b)$. Note that this also involves a projection but the two spaces $\mathcal{K}$ and $\mathcal{L}$ and their bases V and W described above, are the same. Now

$$\tilde{r} = b - A\tilde{x} \tag{3.2.10}$$
$$= b - Ax_0 - AVy \tag{3.2.11}$$
$$= r_0 - AV(V^T AV)^{-1}V^T r_0 \tag{3.2.12}$$

and the matrix $AV(V^T AV)^{-1}V^T$ satisfies

$$[AV(V^T AV)^{-1}V^T][AV(V^T AV)^{-1}V^T] = AV(V^T AV)^{-1}V^T \tag{3.2.13}$$

The A-norm of the error often represents energy in a mechanical system and so it is often referred to as the energy norm.

3.3 Deriving Conjugate Gradient from Conjugate Directions

At the time this is being written, it's been 70 years since the Conjugate Gradient algorithm was proposed by Hestenes and Stiefel back in 1952[6]. Upon their first discussion of the algorithm, numerous perspectives were explored. Three of the most important ideas are using Conjugate Directions, minimizing the energy norm of the error of the linear system and coming up with an update of the conjugate vectors using the residual vector at the current iteration. Here, we use the exact same idea, but we diverge from Hestenes and Stiefel's approach in favor of using the oblique projector and the subspace orthogonality conditions to derive it. The ideas are rehashed we point out its relations to Krylov Subspace only at the end. Usually under classroom settings or textbooks, the relations of Conjugate Gradient, Lanczos Iterations and Krylov Subspaces are discussed together to explain some of the more important properties of the algorithm so that we can move on and talk about other things. However, in this section we derive it in a way similar to the approach used in course notes by Shewchuk [12].

3.3.1 CG Objective and Framework

We introduce the algorithm as an attempt to minimize the energy norm of the error for a system of linear equations $Ax = b$, and we make the assumptions:

1) The matrix A is symmetric positive definite.

2) There is a matrix $P_k = [p_0\ p_1\ \cdots p_{k-1}]$ whose columns form a basis for the space over which we are minimizing.

Let's consider the following objective of minimizing the energy norm of the error over a subspace.

$$\min_{w\in\mathbb{R}^k} \|A^{-1}b - (x_0 + P_k w)\|_A^2 \iff P_k^T r_0 = P_k^T A P_k w \tag{3.3.1}$$

Refer back to equation (3.2.9) for how to obtain the above condition. Using the matrix from equation (3.2.2), where W, V are both P_k, we reformulate the norm minimization conditions as:

$$\text{choose: } x \in x_0 + \operatorname{ran}(P_k) \text{ s.t: } b - Ax \perp \operatorname{ran}(P_k) \tag{3.3.2}$$

Take note that the link between a norm minimization and an equivalent subspace orthogonality condition isn't guaranteed to happen for other subspace projection methods. For example, the FOM and Bi-Lanczos Methods are orthogonalization methods that don't directly link to a norm minimization objective [11].

To solve for w, we wish to make $P_k^T A P_k$ to be an easy-to-solve matrix. Let the easy-to-solve matrix be a diagonal matrix and hence we let P_k be a *matrix whose columns are A-Orthogonal vectors*. It's also referred to as *conjugate vectors*.

$$P_k^T A P_k = D_k \text{ where: } (D_k)_{i,i} = \langle p_{i-1}, Ap_{i-1}\rangle \tag{3.3.3}$$

$$P_k^T r_0 = P_k^T A P_k w = D_k w \tag{3.3.4}$$

$$w = D_k^{-1} P_k^T r_0 \tag{3.3.5}$$

Now we have the following expressions for x_k and r_k:

$$\begin{cases} x_k = x_0 + P_k D_k^{-1} P_k^T r_0 \\ r_k = r_0 - AP_k D_k^{-1} P_k^T r_0 \end{cases} \tag{3.3.6}$$

Let this algorithm be the prototype.

3.3.2 Using the Projector

Observe that $AP_k D_k^{-1} P_k$ is a projector, and so is $P_k D_k^{-1} P_k^T A$. We can check by:

$$AP_k D_k^{-1} P_k^T (AP_k D_k^{-1} P_k^T) = AP_k D_k^{-1} \underbrace{(P_k^T A P_k)}_{D_k} D_k^{-1} P_k^T = AP_k D_k^{-1} P_k^T \tag{3.3.7}$$

$$P_k D_k^{-1} \underbrace{P_k^T A (P_k}_{D_k} D_k^{-1} P_k^T A) = P_k D_k^{-1} D_k D_k^{-1} P_k^T A = P_k D_k^{-1} P_k^T A \tag{3.3.8}$$

They are not Hermitian,therefore they are oblique projectors. For convenience, we denote $\overline{P}_k = P_k D_k^{-1} P_k^T$; So we can simply denotes them by $A\overline{P}_k, \overline{P}_k A$. Observe that:

$$\operatorname{ran}(I - A\overline{P}_k) \perp \operatorname{ran}(P_k) \tag{3.3.9}$$

$$\operatorname{ran}(I - \overline{P}_k A) \perp \operatorname{ran}(AP_k) \tag{3.3.10}$$

because:

$$P_k^T(I - A\overline{P}_k) = P_k^T - P_k^T A\overline{P}_k \tag{3.3.11}$$

$$= P_k^T - D_k D_k^{-1} P_k^T = \mathbf{0} \tag{3.3.12}$$

$$(AP_k)^T(I - \overline{P}_k A) = P_k^T A - P_k^T A\overline{P}_k A \tag{3.3.13}$$

$$= P_k^T A - P_k^T A P_k D_k^{-1} P_k^T A \tag{3.3.14}$$

$$= P_k^T A - P_k^T A = \mathbf{0} \tag{3.3.15}$$

Proposition 3.3 (Generating A-Orthogonal Vectors)**.** Given any set of linearly independent vectors, for example $\{u_i\}_{i=0}^{n-1}$, one can generate a set of A-Orthogonal vectors from it. More specifically:

$$p_k = (I - \overline{P}_k A)u_k \implies p_k \perp \operatorname{ran}(AP_k) \tag{3.3.16}$$

Proof. It's direct from the properties of the projectors. □

3.3.3 Method of Conjugate Directions

So far, we have this particular scheme of solving the optimization problem, coupled with the way to compute the solution x_k at each step, and the residual r_k at each step. However, it would be great if we could update x_k, r_k, and p_k using results from previous iterations.

Definition 4 (Conjugate Direction Method)**.**

$$\begin{cases} \overline{P}_k = P_k D_k^{-1} P_k^T \\ x_k = x_0 + \overline{P}_k r_0 \\ r_k = (I - A\overline{P}_k) r_0 \\ P_k^T A P_k = D_k \\ p_k = (I - \overline{P}_k A)u_k \quad \{u_i\}_{i=0}^{n-1} \text{linearly independent vectors} \end{cases} \tag{3.3.17}$$

With the assistance of a set of basis vectors that span the whole space, this algorithm can achieve the objective.

Remark 3.3.1. This CDM method is nothing new, in the original paper from Hestenes and Stiefel back in 1952[6], they commented on the method of Conjugate Direction, for each choice of basis $\{u_i\}_{i=0}^{n-1}$ there is a unique algorithm. If one were to choose the basis to be the set of standard basis vectors, then the resulting algorithm would be the equivalent of Gaussian Elimination.

Remark 3.3.2 (Geometric Intuition of CDM)**.** What is happening geometrically is that the A-Orthogonal vectors are orthogonal if described under the alternative eigenspace. Intuitively, one should think of a high-dimensional sphere that sits along some orthogonal basis, and the transformation of A is stretching and rotating sphere, along with the orthogonal axis, resulting in a new ellipsoid in a different orientation; when the transformation is applied, the orthogonal coordinate inside the sphere got stretched along with it, and now these axes had become A-orthogonal vectors. Tracing along the direction of these vectors will ensure minimum redundancy of search directions.

3.3.4 Properties of CDM

Here we set up several useful lemma and propositions that can derive the short recurrences of A-Orthogonal vectors

Proposition 3.4 (CDM Property 1)**.**

$$p_{k+j}^T r_k = p_{k+j}^T r_0 \quad \forall\, 0 \le j \le n-k \tag{3.3.18}$$

Proof.

$$\begin{aligned} p_{k+j}^T r_k &= p_{k+j}^T (I - A\overline{P}_k) r_0 && (3.3.19)\\ &= (p_{k+j}^T - p_{k+j}^T A\overline{P}_k) r_0 && (3.3.20)\\ &= p_{k+j}^T r_0 && (3.3.21) \end{aligned}$$

□

Proposition 3.5 (CDM Recurrence)**.**

$$\begin{aligned} r_k - r_{k-1} &= r_0 - A\overline{P}_k r_0 - (r_0 - A\overline{P}_{k-1} r_0) && (3.3.22)\\ &= -A\overline{P}_k r_0 + A\overline{P}_{k-1} r_0 && (3.3.23)\\ &= -Ap_{k-1}\frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (3.3.24)\\ \implies x_k - x_{k-1} &= p_{k-1}\frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (3.3.25)\\ \text{def: } a_{k-1} := \frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} &= \frac{\langle p_{k-1}, r_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (3.3.26) \end{aligned}$$

On (3.3.26) we used CDM Property 1. The value of a_{k-1} is defined above,, we have two equivalent representations for a_{k-1}. This recurrence remains true for the future regardless of the set $\{u\}_{i=0}^{n-1}$ that generates these conjugate vectors.

3.3.5 Conjugate Gradient

Now, consider the case where the set of basis vectors: $\{u_i\}_{i=0}^{n-1}$ are the residual vectors generated from the CDM itself. This generates the CG method.

Lemma 3.3.1.

$$\langle p_{k+j}, Ap_k\rangle = \langle r_k, Ap_{k+j}\rangle = \langle p_{k+j}, Ar_k\rangle \quad \forall\, 0 \le j \le n-k \tag{3.3.27}$$

Proof.

$$\begin{aligned} p_{k+j}^T Ap_k &= p_{k+j}^T Ar_k - p_{k+j}^T A\overline{P}_k Ar_k \quad \forall\, 0 \le j \le n-k && (3.3.28)\\ &= p_{k+j}^T Ar_k && (3.3.29)\\ \langle p_{k+j}, Ap_k\rangle &= \langle r_k, Ap_{k+j}\rangle = \langle p_{k+j}, Ar_k\rangle && (3.3.30) \end{aligned}$$

On the first line we invoked the CDM algorithm's definition back in (3.3.27), replacing u_k with r_k, hence $p_k = (I - \overline{P}_k A) r_k$, which is then substituted into line (3.3.28). □

Lemma 3.3.2.

$$\langle r_k, p_k\rangle = \langle r_k, r_k\rangle \tag{3.3.31}$$

Proof.

$$\langle r_k, p_k\rangle = \langle r_k, r_k\rangle - \langle r_k, \overline{P}_k A r_k\rangle = \langle r_k, r_k\rangle \tag{3.3.32}$$

First equality used $p_k = (I - \overline{P}_k A)r_k$, second equality used the fact that r_k is orthogonal to P_k. □

Proposition 3.6 (CG Generates Orthogonal Residuals)**.**

$$\langle r_k, r_j\rangle = 0 \quad \forall\, 0 \le j \le k-1 \tag{3.3.33}$$

Let this above claim be inductively true then consider:

Proof.

$$\begin{aligned}
r_{k+1} &= r_k - a_k A p_k && (3.3.34)\\
\implies \langle r_{k+1}, r_k\rangle &= \langle r_k, r_k\rangle - a_k\langle r_k, Ap_k\rangle && (3.3.35)\\
&= \langle r_k, r_k\rangle - \frac{\langle r_k, r_k\rangle}{\langle p_k, Ap_k\rangle}\langle r_k, Ap_k\rangle && (3.3.36)\\
&= 0 && (3.3.37)
\end{aligned}$$

The first line is from the recurrence of CDM residuals, and then next we make use of a_k from (CDM Recurrence (3.5)) together with Lemma 3.3.1. Next we consider:

$$\begin{aligned}
p_j &= (I - \overline{P}_j A)r_j \quad \forall\, 0 \le j \le k-1 && (3.3.38)\\
\implies r_j &= p_j + \overline{P}_j A r_j && (3.3.39)\\
r_k &= (I - A\overline{P}_k)r_0 && (3.3.40)\\
r_k \perp \operatorname{ran}(P_k) \implies \langle r_k, r_j\rangle &= \langle r_k, p_j + \overline{P}_j A r_j\rangle = 0 && (3.3.41)
\end{aligned}$$

The second line (3.3.39) is a result of the first line (3.3.38) rearranged. Here we again make use of the projector $I - A\overline{P}_k$. The last line (3.3.53) is using the second line 3.3.39. The base case of the argument is simple, because $p_0 = r_0$, and by the property of the projector, $\langle r_1, r_0\rangle = 0$. The theorem is now proven. □

Proposition 3.7 (CG Recurrences)**.**

$$p_k = r_k + b_{k-1}p_{k-1} \quad b_{k-1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \tag{3.3.42}$$

Proof. The proof is direct, starting with the definition of CDM, which is given as:

$$\begin{aligned}
p_k &= (I - \overline{P}_k A)r_k && (3.3.43)\\
r_k - \overline{P}_k A r_k &= r_k - P_k D_k^{-1} P_k^T A r_k && (3.3.44)\\
&= r_k - P_k D_k^{-1}(AP_k)^T r_k && (3.3.45)
\end{aligned}$$

Observe:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} \tag{3.3.46}$$

Next, we can make use of lemma 3.3.1 to get rid of Ar_k:

$$\langle p_j, Ar_k \rangle \quad \forall\, 0 \le j \le k-2 \tag{3.3.47}$$

$$\begin{aligned} \langle p_j, Ar_k \rangle &= \langle r_k, Ap_j \rangle && (3.3.48) \\ &= \langle r_k, a_j^{-1}(r_j - r_{j+1}) \rangle && (3.3.49) \\ &= a_j^{-1} \langle r_k, (r_j - r_{j+1}) \rangle = 0 && (3.3.50) \end{aligned}$$

The second line is also using the property that the matrix A is symmetric, the third line is using the recurrence of the residual established for CDM (CDM Recurrences (Proposition 3.5)), and the last line is true for all $0 \le j \le k-2$ by the orthogonality of the residual proved in CG Generates Orthogonal Residuals (Proposition 3.6). Therefore we have:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} = a_{k-1}^{-1} \langle r_k, (r_{k-1} - r_k) \rangle \xi_k \tag{3.3.51}$$

Take note that the vector ξ_k is the k th standard basis vector in $\mathbb{R}^k$, keep in mind that $r_k \perp r_{k-1}$ as well. Using these facts we can simplify the expression for p_k into:

$$\begin{aligned} p_k &= r_k - P_k D_k^{-1} (AP_k)^T r_k && (3.3.52) \\ &= r_k - P_k D_k^{-1} a_{k-1}^{-1} (\langle r_k, (r_{k-1} - r_k) \rangle) \xi_k && (3.3.53) \\ &= r_k - \frac{a_{k-1}^{-1} \langle -r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k && (3.3.54) \\ &= r_k + \frac{a_{k-1}^{-1} \langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k && (3.3.55) \\ &= r_k + \left(\frac{\langle r_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \right)^{-1} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k && (3.3.56) \\ &= r_k + \frac{\langle r_k, r_k \rangle}{\langle r_{k-1}, r_{k-1} \rangle} p_k && (3.3.57) \end{aligned}$$

We make use of the definition for a_{k-1} for the CDM algorithm (proposition 3.5 together with lemma 3.3.2). At this point, we have proven the short CG recurrences for p_k. □

Up until this point we have developed the standard form of the conjugate gradient algorithm proposed by Hestenes & Stiefel[6]. We started with the minimization objective and the properties of P_k, then we defined a recurrence for the residual (and simultaneously the solution x_k), and the A-Orthogonal vectors using a set of basis vectors to assist in the generation process. Next, we chose the basis vectors to be the set of residual vectors generated from the algorithm itself; after some proofs, we uncovered the exact same parameters found in most of the definitions of the CG algorithm:

Definition 5 (CG)**.**

$$p^{(0)} = b - Ax^{(0)} \tag{3.3.58}$$

$$\text{For } i = 0, 1, \cdots \tag{3.3.59}$$

$$\begin{aligned} a_i &= \frac{\|r^{(i)}\|^2}{\|p^{(i)}\|_A^2} \\ x^{(i+1)} &= x^{(i)} + a_i p^{(i)} \\ r^{(i+1)} &= r^{(i)} - a_i A p^{(i)} \\ b_i &= \frac{\|r^{(i+1)}\|_2^2}{\|r^{(i)}\|_2^2} \\ p^{(i+1)} &= r^{(i+1)} + b_i p^{(i)} \end{aligned} \tag{3.3.60}$$

That is the algorithm, stated with all the iteration numbers listed as superscripts inside parentheses. Which is equivalent to what we have proven for the CG.

3.3.6 CG and Krylov Subspace

The conjugate Gradient Algorithm is actually a CDM. It's a special case of the CDM method where the first direction of descend is the gradient at the initial guess (The residual). Next, we want to show how CG is related to the Krylov Subspace, which only happens with CG and not the CDM.

Proposition 3.8.

$$p_k \in \mathcal{K}_{k+1}(A|r_0) \tag{3.3.61}$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \tag{3.3.62}$$

Proof. The base case is trivial and it's directly true from the definition of CG: $r_0 \in \mathcal{K}_1(A|r_0), p_0 = r_0 \in \mathcal{K}_1(A|r_0)$. Next, we inductively assume that $r_k \in \mathcal{K}_{k+1}(A|r_0), p_k \in \mathcal{K}_{k+1}(A|r_0)$, then we consider:

$$r_{k+1} = r_k - a_k A p_k \tag{3.3.63}$$

$$\in r_k + A\mathcal{K}_{k+1}(A|r_0) \tag{3.3.64}$$

$$\in r_k + \mathcal{K}_{k+2}(A|r_0) \tag{3.3.65}$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \subseteq \mathcal{K}_{k+2}(A|r_0) \tag{3.3.66}$$

$$\implies r_{k+1} \in \mathcal{K}_{k+2}(A|r_0) \tag{3.3.67}$$

At the same time the update of p_k would assert the property that:

$$p_{k+1} = r_{k+1} + b_k p_k \tag{3.3.68}$$

$$\in r_{k+1} + \mathcal{K}_{k+1}(A|r_0) \tag{3.3.69}$$

$$\in \mathcal{K}_{k+2}(A|r_0) \tag{3.3.70}$$

This is true because r_{k+1} is already a member of the expanded subspace $\mathcal{K}_{k+2}(A|r_0)$. And from this formulation of the algorithm, we can update the Petrov Galerkin's Conditions to be:

Theorem 1 (CG and Krylov Subspace)**.**

$$\text{choose: } x_k \in x_0 + \mathcal{K}_k(A|r_0) \text{ s.t: } r_k \perp \mathcal{K}_k(A|r_0) \tag{3.3.71}$$

Take note that, $\text{ran}(P_k) = \mathcal{K}_k(A|r_0)$ because the index starts with zero for the Conjugate Vectors. □

3.4 Arnoldi Iterations and Lanczos

In this section, we introduce another important algorithm: The Lanczos Algorithm. However, to give more context for the discussion, the Arnoldi iteration is considered as well and it's used to emphasize that Lanczos Iterations is just Arnoldi but with the matrix A being a symmetric matrix. Finally we make the link between Lanczos Iterations and Krylov Subspace, which will inevitably be linked back to CG and plays an important role for the analysis of CG.

3.4.1 The Arnoldi Iterations

We first define the Arnoldi Algorithm, and then we proceed to derive it using the idea of an orthogonal projector. Next, we discuss a special case of the Arnoldi Iteration: the Lanczos Algorithm, which is just Arnoldi applied to a symmetric matrix. And such an algorithm will inherit the properties of the Arnoldi Iterations.

Before stating the algorithm, I would like to point out the interpretations of the algorithm and its relations to Krylov Subspace. Consider a matrix of Hessenberg Form:

$$\tilde{H}_k = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{1,2} & h_{2,2} & \cdots & h_{2,k} \\ & \ddots & & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix} \tag{3.4.1}$$

We initialize the orthogonal projector with the vector q_1, which is $q_1q_1^H$, next, we apply the linear operator A on the current range of the projector: Aq_1, then, we orthogonalize it against q. Let the projection of Aq_1 onto $I - q_1q_1^H$ be $h_{1,2}q_2$, and let the projection onto $q_1q_1^H$ be $h_{1,1}$. This completes the first column of H_k, we do this recursively. Please allow me to demonstrate:

$$Q_1 = q_1 \tag{3.4.2}$$

$$(\tilde{H}_k)_{j+1,k}q_{j+1} = (I - Q_jQ_j^H)Aq_j \tag{3.4.3}$$

$$(\tilde{H}_j)_{1:j,j} = Q_jQ_j^H Aq_j \tag{3.4.4}$$

Usually when implementing, the subdiagonal of H_k are chosen to be

$$\tilde{H}_{k+1,k} := \|(I - Q_k Q_k^T) A q_k\| \tag{3.4.5}$$

which makes then strictly positive, and the sign goes into q_k to assert the equalities above. Q_k is going to be orthogonal because we are using orthogonal projectors. As a consequence, we can express the recurrences of the subspace vector in matrix form:

$$AQ_k = Q_{k+1}\tilde{H}_k \tag{3.4.6}$$

$$Q_k^H A Q_k =: H_k \tag{3.4.7}$$

We define H_k to be the principal submatrix of $\tilde{H}_k$. Please observe that, if A is symmetric, then $Q_k^H A Q_k$ is also symmetric, which makes H_k symmetric, implying H_k is a symmetric tridiagonal Matrix, giving us the tridiagonal factorizations. Instead of orthogonalizing against all previous vectors,The Lanczos algorithm simply orthogonalize against the previous q_k, q_{k-1} vectors, reusing the sub-diagonal elements for q_{k-1}; we refers to this as *Lanczos Iterations*.

3.4.2 Arnoldi Produces Orthogonal Basis for Krylov Subspace

One important observation reader should make about the idea of Arnoldi Iteration is that, during each iteration, the matrix Q_k spans the same range as $\mathcal{K}_k(A|q_1)$.

Proposition 3.9.

$$\operatorname{ran}(Q_k) = \mathcal{K}_k(A|q_1) \tag{3.4.8}$$

Proof. The base case is simple: $q_1 \in \mathcal{K}_1(A|q_1)$, inductively assuming the proposition is true, using the polynomial property of Krylov Subspace we consider:

$$\begin{aligned}
& Q_k \in \mathcal{K}_k(A|q_1) \\
\iff\ & w_k^+ : \exists\, p_k(A|w_k^+)q_1 = q_k \\
& \implies Aq_k = Ap_k(A|w_k^+)q_1 \in \mathcal{K}_{k+1}(A|w_k^+) \\
& q_{k+1} \in \mathcal{K}_{k+1}(A|q_1) \\
\implies\ & \operatorname{ran}(Q_{k+1}) = \mathcal{K}_{k+1}(A|q_1)
\end{aligned}$$

The Arnoldi Algorithm terminates if the value $h_{k+1,k}$ is set to be zero. This is the case because the normalization process is dividing by $h_{k+1,k}$ to get q_{k+1}. This only happens when $Aq_k \in \operatorname{ran}(Q_k)$; because $h_{k+1,k}$ is given by the projector of $I - Q_k Q^H$ applied to Aq_k and the null space of this projector is $\operatorname{ran}(Q_k)$, resulting in $h_{k+1,k} = 0$. ☐

Remark 3.4.1 (Arnoldi Produces Minimal Monic in Krylov Subspace)**.** The characteristic polynomial of H_k, minimizes $\|p(A|w)q_1\|_2$ among all monic polynomials with degree $k-1$. For more information, Trefethen has a coverage on the topic in his works [14]. The minimization property in Arnoldi translates to Lanczos Iterations as well.

3.4.3 The Lanczos Iterations

Definition 6 (Lanczos Iterations)**.**

$$\text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \tag{3.4.9}$$
$$\text{set: } \beta_0 = 0 \tag{3.4.10}$$
$$\text{For } j = 1, 2, \cdots \tag{3.4.11}$$
$$\begin{aligned} &\tilde{q}_{j+1} := Aq_j - \beta_{j-1}q_{j-1} \\ &\alpha_j := \langle q_j, \tilde{q}_{j+1}\rangle \\ &\tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\ &\beta_j = \|\tilde{q}_{j+1}\| \\ &q_{j+1} := \tilde{q}_{j+1}/\beta_j \end{aligned} \tag{3.4.12}$$

Here, let it be the case that H_k is a symmetric tridiagonal Matrix with α_i on the diagonal, β_i on the sub and super diagonal; the Lanczos is Arnoldi, but we make use of the symmetric properties to orthogonalize Aq_j against q_{j-1} using β_{j-1}, and in this case, each iteration only consists of one vector inner product.Other variants of the Lanczos Iterations algorithm exists. See appendix item 7 for one such variant.

The algorithm generates the following two matrices, Q_k which is orthogonal and it spans $\mathcal{K}_k(A|q_1)$, and a symmetric tridiagonal Matrix:

$$Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{3.4.13}$$

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{3.4.14}$$

Similar to the recurrence from the Arnoldi Algorithm, the Lanczos also create a recurrence between Aq_k and Q_k and q_{k+1}, but the recurrence is shorter so that it simply makes use of the previous two vectors. In addition, the tridiagonal matrix that is produced has no repeated eigenvalues, which is a useful fact and for a proof, see appendix item B.5.

Theorem 2 (Lanczos Recurrences)**.**

$$AQ_k = Q_k T_k + \beta_k q_{k+1}\xi_k^T = Q_{k+1}\tilde{T}_k \tag{3.4.15}$$
$$\implies Aq_j = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \quad \forall\, 2 \le j \le k \tag{3.4.16}$$
$$\implies Aq_1 = \alpha_1 q_1 + \beta_1 q_2 \tag{3.4.17}$$

Oftentimes, we refer to the $k \times k$ symmetric tridiagonal matrix generated from Iterative Lanczos as T_k.

Proposition 3.10 (Lanczos Termination Conditions)**.** The Lanczos Iteration produces a symmetric tridiagonal Matrix that has no zero element on its super and sub-diagonal, and if β_k is zero, then the algorithm must terminate, and k would equal to grade$(A|q_1)$, the grade of the Krylov Subspace.

Proof. It's true because the β_k in the Lanczos is equivalent to $h_{k+1,k}$. It's been discussed previously that if $h_{k+1,1} = 0$ for the Arnoldi's Iteration, then the Krylov Subspace $\mathcal{K}_k(A|q_1)$ became an invariant subspace under A, and in that sense, the algorithm has to terminate due to a divide by zero error. □

Remark 3.4.2 (Minimal Polynomial from Lanczos Iterations)**.** The characteristic polynomial of T_k has a special minimization property. Here recall remark 3.4.1, we make use of the minimization property of the characteristic polynomial of the Hessenberg matrix from the Arnoldi Iterations. Under Lanczos iterations the matrix H_k becomes the tridiagonal T_k. Since matrix A is symmetric, we consider its eigendecomposition in the form: $A = V\Lambda V^T$, we let $\bar{p}_{k-1}(x)$ denote the characteristic polynomial of matrix T_k, then using the 2-norm minimization properties we have:

$$\min_{p_{k-1}:\text{monic}} \|p(A)q_1\|_2 \tag{3.4.18}$$

$$= \|\overline{p}_{k-1}(A)q_1\|_2 \tag{3.4.19}$$

$$= \|V\bar{p}_{k-1}(\Lambda)V^T q_1\|_2 \tag{3.4.20}$$

$$= \|\bar{p}_{k-1}(\Lambda)V^T q_1\|_2 \tag{3.4.21}$$

$$= \sqrt{\sum_{i=1}^{n} p_{k-1}(\lambda_i)^2 (V^T q_1)_1^2} \tag{3.4.22}$$

The last line is saying the characteristic polynomial for T_k from the Lanczos iterations is minimizing a weighted squared sum at the eigenvalues of the matrix A. we can refine the termination condition from proposition 3.10 for the Lanczos iterations using information of the initial vector q_1. Given the information for the initial vector q_1 and under exact arithmetic, the number of iterations underwent by Lanczos equals the unique number of eigenvalues λ_i where $(V^T q_1)_i \neq 0$, using a very similar argument compare to proposition 3.1.

In fact, the idea can be taken further. The Lanczos vector q_k represents orthogonal polynomial in $\mathcal{K}_k(A|q_1)$ under a weighted subspace, and such a polynomial is a rescaled version of the characterstic polynomial of T_k. (see proposition B.2 in the appendix for a proof)

4 Analysis of Conjugate Gradient and Lanczos Iterations

In this section, we state the termination conditions for the Lanczos iterations and the CG algorithm we developed using the property of Krylov Subspace. This is just a thorough discussion of these two algorithms and applying the foundations and is not following any references.

4.1 Conjugate Gradient and Matrix Polynomial

One important result of the optimization objective listed CG and Krylov Subspace1 is the connection to the matrix polynomial of A and Conjugate Gradient. More specifically we consider the following proposition:

Proposition 4.1 (CG Relative Energy Error)**.**

$$x_k \in \mathcal{K}(A|r_0)w + x_0 \tag{4.1.1}$$

$$\frac{\|e_k\|_A^2}{\|e_0\|_A^2} = \min_{w\in\mathbb{R}^k} \|(1 + Ap_k(A|w))A^{1/2}e_0\|_2^2 \leq \min_{p_k:p_k(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_k(x)| \tag{4.1.2}$$

Here we use the notation $e_k = A^{-1}b - x_k$ to denote the error vector.

Proof.

$$\|e_k\|_A^2 = \min_{x_k\in x_0+\mathcal{K}_k(A|r_0)} \|x^+ - x_k\|_A^2 \tag{4.1.3}$$

$$x_k \in x_0 + \mathcal{K}_k(A|r_0) \implies e_k = e_0 + p_{k-1}(A|w)r_0 \tag{4.1.4}$$

$$\implies = \min_{w\in\mathbb{R}^k} \|e_0 + p_{k-1}(A|w)r_0\|_A^2 \tag{4.1.5}$$

$$= \min_{w\in\mathbb{R}^k} \|e_0 + Ap_{k-1}(A|w)e_0\|_A^2 \tag{4.1.6}$$

$$= \min_{w\in\mathbb{R}^k} \|A^{1/2}(I + Ap_{k-1}(A|w))e_0\|_2^2 \tag{4.1.7}$$

$$\leq \min_{w\in\mathbb{R}^k} \|I + Ap_{k-1}(A|w)\|_2^2\|e_0\|_A^2 \tag{4.1.8}$$

$$= \min_{w\in\mathbb{R}^k} \left(\max_{i=1,\ldots,n} |1 + \lambda_i p_{k-1}(\lambda_i|w)|^2\right) \|e_0\|_A^2 \tag{4.1.9}$$

$$\leq \min_{w\in\mathbb{R}^k} \left(\max_{x\in[\lambda_{\min},\lambda_{\max}]} |1 + \lambda_i p_{k-1}(\lambda_i|w)|^2\right) \|e_0\|_A^2 \tag{4.1.10}$$

$$= \min_{p_k:p_k(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_k(x)|^2\|e\|_A^2 \tag{4.1.11}$$

$$\implies \frac{\|e_k\|_A}{\|e_0\|_A} \leq \min_{p_k:p_k(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_k(x)| \tag{4.1.12}$$

(4.1.3) is the Error Energy norm minimization objective of CG, we proceed with writing up the affine subspace where x_k is from: $x_0 + \mathcal{K}_k(A|r_0)$ at (4.1.4), putting Krylov subspace in terms of a matrix polynomial multiplied by r_0 and then use $A^{-1}b$ to subtract both sides to get the expression for e_k. From the (4.15) line to the (4.16), we use the fact that $r_0 = Ae_0$, allowing us to extract out a factor A.

Next, from (4.1.6) to (5.1.7), we use the fact that every symmetric definite matrix A has the factorization of $A^{1/2}A^{1/2}$ where $A^{1/2}$ is also a symmetric definite matrix. After that we moved the $A^{1/2}$ to e_0 to get $\|e_0\|_A^2$ from (4.1.7) to (4.1.8), the matrix polynomial part is left with the 2-norm. From (4.1.8) to (4.1.9) we use the eigendecomposition of A which is diagonalizable with a unitary transform, giving us the form of $Q\Lambda Q^T = A$ where Q is an Unitary Matrix and diagonals of Λ are the eigenvalues of A. Allow me to explain:

$$\|I + Ap_{k-1}(A|w)\|_2^2 = \|Q(I + \Lambda p_{k-1}(\Lambda|w))Q^T\|_2^2 \tag{4.1.13}$$

$$= \|I + \Lambda p_{k-1}(\Lambda|w)\|_2^2 \tag{4.1.14}$$

$$= \max_{i=1,\cdots,n} |1 + \lambda_i p_{k-1}(\lambda_i|w)|^2 \tag{4.1.15}$$

Where, the 2-norm of a diagonal matrix Λ is just its biggest diagonal element. And then we relax the conditions for λ_i by reducing it to be some element in the interval between the minimum and the maximum of the eigenvalues for the matrix A (from (4.1.9) to (4.1.10)). Finally, please notice that we use a monic $p_{k+1}(x)$ at the end to simplify things. $\square$

The above results will be useful for proving the convergence of CG.

Remark 4.1.1 (The CG Relative Energy Error Norm is Tight)**.** The bound is tight refers to the fact that the matrix is SPD, therefore $\|Ax\| \le \|A\|\|x\|$ is tight, which justifies that (4.1.10) is tight in the sense that for any iteration k, we can choose an initial vector e_0 such that the equality is achieved. (4.1.12) can still be tight if we have the freedom to choose the eigenvalues of the matrix A. However, the bound is rarely tight if the initial error vector e_0 and the matrix A is fixed.

4.1.1 Termination Conditions of CG

Proposition 4.2 (Termination of CG)**.** For all initial guesses, the maximum iterations underwent by the CG algorithm is the number of unique eigenvalues for the matrix A.

This result is direct from (4.1.8), the CG algorithm terminates when a polynomial that interpolates all the unique eigenvalues is found. This bound is true for all initial guesses, and sometimes for some given e_0, the terminations can come with fewer iterations.

4.2 Convergence Rate of CG under Exact Arithmetic

In this section discuss Greenbaum's Analysis for convergence rate of the algorithm[5] in thorough details. The core idea is to use a Chebyshev Polynomial to establish a bound and it's applicable when the linear operator has extremely high dimension and we limit the number of iterations to k where k is much smaller than n, the size of the matrix. We will follow Greenbaum's Analysis but with some more details.

4.2.1 Uniformly Distributed Eigenvalues

Theorem 3 (CG Convergence Rate)**.** The relative error squared measured over the energy norm is bounded by:

$$\frac{\|e_k\|_A}{\|e_k\|_A} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{4.2.1}$$

Where k is the number of iterations, and $e_k = A^{-1}b - x_k$, the upper bound is the most general and it's able to bound the convergence given $\lambda_{\min}, \lambda_{\max}$ of the operator A. The bound is loose if there is some kind of clustering of the eigenvalue of matrix A, and the bound would be tighter given that $k << n$ and the eigenvalues of A are evenly spread out on the spectrum.

Before the proof, I need to point out that the analysis draws inspiration from the interpolating mononic polynomial for the spectrum of the matrix A, and we make use of the Inf Norm minimization property of the Chebyshev Polynomial. Here, we order all the eigenvalues of matrix A so that λ_1, λ_n denotes the maximum and the minimum eigenvalues for A.

Proof. We start by adapting the Chebyshev Polynomial to the convex hull of the spectrum for matrix A.

$$T_k(x) = \arg\min_{p\in\mathcal{P}_k} \max_{x\in[-1,1]} |p(x)| \tag{4.2.2}$$

$$p_k(x) := \frac{T_k(\varphi(x))}{T_k(\varphi(0))} \quad \text{where: } \varphi(x) := \frac{2x-\lambda_1-\lambda_n}{\lambda_n-\lambda_1} \tag{4.2.3}$$

$$\implies p_k(x) = \arg \min_{\substack{p\in\mathcal{P}_k \\ \text{s.t: } p(0)=1}} \max_{x\in[\lambda_1,\lambda_n]} |p(x)| \tag{4.2.4}$$

At this point, we have defined a new polynomial p_k that minimizes the inf norm over the convex hull of the eigenvalues. Note that here we use T_k for the type T Chebyshev Polynomial of degree k and it's not the tridiagonal symmetric matrix from Lanczos iterations. Next, we use the property that the range of the Chebyshev is bounded within the interval $[-1, 1]$ to obtain inequality:

$$\forall x \in [\lambda_1, \lambda_n] : \left|\frac{T_k(\varphi(x))}{T_k(\varphi(0))}\right| \le \left|\frac{1}{T_k(\varphi(0))}\right| \tag{4.2.5}$$

Next, our objective is to find any upper bound for the quantities on the RHS in relation to the Condition number for matrix A and the degree of the Chebyshev polynomial. Firstly observe that $\varphi(0) < -1, \varphi(0) \notin [\lambda_1, \lambda_n]$, because all Eigenvalues are larger than zero, therefore it's out of the range of the Chebyshev polynomial and we need to find the actual value of it by considering alternative form of Chebyshev T for values outside of the $[-1, 1]$:

$$T_k(x) = \cosh(k\ \text{arccosh}(z)) \quad \forall z \ge 1 \tag{4.2.6}$$

$$\implies T_k(\cosh(\zeta)) = \cosh(k\zeta) \quad z := \cosh(\zeta) \tag{4.2.7}$$

We need to match the form of the expression $T_k(\varphi(0))$ with the expression of the form $T_k(\cosh(\zeta))$ given the freedom of varying ζ. To do that we consider a substitution of $\zeta = \ln(y)$, so that we only need to match $\varphi(0)$ with the form $(y+y^{-1})/2$, which is just a quadratic equation.

$$\varphi(0) = \cosh(\zeta) = \cosh(\ln(y)) \quad \ln(y) := \zeta \tag{4.2.8}$$

$$\text{recall: } \cosh(x) = (\exp(-x) + \exp(x))/2 \tag{4.2.9}$$

$$\implies \cosh(\ln(y)) = (y + y^{-1})/2 \tag{4.2.10}$$

$$\varphi(0) = (y + y^{-1})/2 \tag{4.2.11}$$

Recall the definition of $\varphi(x)$ and then simplifies:

$$\begin{aligned}
\varphi(0) &= \frac{-\lambda_n - \lambda_1}{\lambda_n - \lambda_1} \\
&= \frac{-\lambda_n/\lambda_1 - 1}{\lambda_n/\lambda_1 - 1} \\
&= -\frac{\lambda_n/\lambda_1 + 1}{\lambda_n/\lambda_1 - 1} \\
\implies \varphi(0) &= -\frac{\kappa + 1}{\kappa - 1}
\end{aligned}$$

Our objective is now simple. We know what $\varphi(0)$ is, we want it to form match with $\cosh(\ln(y))$, and hence we simply solve for y:

$$-\frac{\kappa+1}{\kappa-1} = \frac{1}{2}(y+y^{-1}) \tag{4.2.12}$$

$$y = \frac{\sqrt{\kappa}\pm 1}{\sqrt{\kappa}\mp 1} \tag{4.2.13}$$

It's a quadratic and we solved it. The above $\pm, \mp$ are correlated, meaning that they are of opposite sign, which gives us two roots for the quadratic expression. Now, given the hyperbolic form for $\varphi(0)$, we can substitute and get the value of $T_k(\varphi(0))$ in terms of y and then κ:

$$\varphi(0) = \frac{1}{2}(y+y^{-1}) \tag{4.2.14}$$

$$\implies T_k(\varphi(0)) = T_k(\cosh(\ln(y))) \tag{4.2.15}$$

$$= \cosh(k\ln(y)) \tag{4.2.16}$$

$$= (y^k + y^{-k})/2 \tag{4.2.17}$$

Then, substituting the value of y, and invert the quantity we have:

$$\frac{1}{T_k(\varphi(0))} = 2(y^k + y^{-k})^{-1} \tag{4.2.18}$$

$$= 2\left(\left(\frac{\sqrt{\kappa}\pm 1}{\sqrt{\kappa}\mp 1}\right)^k + \left(\frac{\sqrt{\kappa}\mp 1}{\sqrt{\kappa}\pm 1}\right)^{-k}\right)^{-1} \tag{4.2.19}$$

$$= 2\left(\underbrace{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k}_{>1} + \underbrace{\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{-k}}_{<1}\right)^{-1} \tag{4.2.20}$$

$$\le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{4.2.21}$$

Which completes the proof. Recall from the previous discussion for the squared of the relative error, we have:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \le \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_1,\lambda_n]} |p_{k+1}(x)| \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{4.2.22}$$

□

4.2.2 One Outlier Eigenvalue

Using the derived theorem, we can extend it to other types of distributions of eigenvalues. Imagine an extreme case where some matrices have one group of eigenvalues that are close together and one single eigenvalue that is far away from the cluster. In that case, we can use Chebyshev differently by focusing its minimizing power across the clustered eigenvalues and use a simple polynomial to interpolate the outlier eigenvalue. Consider the following proposition:

Proposition 4.3 (Big Outlier CG Convergence Rate)**.** If, there exists a λ_n that is much later than all previous $n-1$ eigenvalues for the matrix A, then a tighter convergence bound that being only parameterized by the range of clustered eigenvalues can be obtained and it is:

$$\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A} \le 2\left(\frac{\sqrt{\kappa_{n-1}}-1}{\sqrt{\kappa_{n-1}+1}}\right)^{k-1} \qquad \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{4.2.23}$$

Reader, please observe that the outlier eigenvalue λ_n plays a smaller role in determining the convergence rate of the algorithm compared to the previous bound.

Proof. Here, we wish to show that a more focused use of the Chebyshev will introduce a better convergence rate for the Conjugate Gradient. We define the notation for the adapted k-th degree Chebyshev Polynomial over an closed interval: $[a, b]$ as:

$$\hat{T}^{(k)}_{[a,b]}(x) := T_k\left(\frac{2x-b-a}{b-a}\right) \tag{4.2.24}$$

Next, we consider the following polynomial:

$$p_k(x) := \frac{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(x)}{T^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)}\left(\frac{\lambda_n - x}{x}\right) \tag{4.2.25}$$

Where, we use an $k-1$ degree polynomial for the clustered eigenvalues, and then we multiply that by a linear function $(\lambda_n - z)/\lambda_n$ which is zero at right boundary λ_n and it's less than one at the left boundary λ_1. Next, observe the following facts about the above polynomials:

$$\frac{\lambda_n - z}{\lambda_n} \in [0, 1] \quad \forall z \in [\lambda_1, \lambda_n] \tag{4.2.26}$$

$$|p_k(x)| \le \left|\frac{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(x)}{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)}\frac{\lambda_n - z}{\lambda_n}\right| \le \frac{1}{\left|\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)\right|} \tag{4.2.27}$$

As a result, we can apply the convergence rate we proven for the uniform case, giving us:

$$T^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0) = \left|T_{k-1}\left(\frac{-\lambda_{n-1}-\lambda_1}{\lambda_{n-1}-\lambda_1}\right)\right| \tag{4.2.28}$$

$$= \frac{1}{2}(y^{k-1} + y^{-(k-1)}) \tag{4.2.29}$$

$$\text{where: } y = \frac{\sqrt{\kappa_{n-1}}+1}{\sqrt{\kappa_{n-1}}-1}, \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{4.2.30}$$

Substituting the value for y we obtain the bound:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \le 2\left(\frac{\sqrt{\kappa_{n-1}}-1}{\sqrt{\kappa_{n-1}}+1}\right)^{k-1} \tag{4.2.31}$$

$\square$

Another case that is worth considering is when there is one eigenvalue that is smaller than all the other eigenvalues which are clustered at a way larger value than it, by which I mean the value of λ_1 is much smaller than all other eigenvalues and the other eigenvalues are clustered close together in an interval uniformly.

Proposition 4.4 (Small Outlier CG Convergence Rate)**.** The convergence rate is:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \le 2\left(\frac{\lambda_n - \lambda_1}{\lambda_1}\right)\left(\frac{\sqrt{\kappa_0}-1}{\sqrt{\kappa_0}+1}\right)^{k-1} \tag{4.2.32}$$

Where κ_0 is λ_n/λ_2.

Proof.

$$w(z) := \frac{\lambda_1 - z}{\lambda_1} \tag{4.2.33}$$

$$p_k(z) := w(z)\left(\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}\right) \tag{4.2.34}$$

$$\implies \max_{x\in[\lambda_2,\lambda_n]} |w(x)| = \frac{\lambda_n - \lambda_1}{\lambda_1} \tag{4.2.35}$$

In this case, the maximal value of the linear function w is achieved via $x = \lambda_1$, and the absolute value swapped the sign of the function. Therefore, we have:

$$\begin{aligned} |p_k(x)| &= \left| w(x)\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(x)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| && (4.2.36)\\ &\le \left|\frac{w(x)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| && (4.2.37)\\ &\le \left|\left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)\right| && (4.2.38)\\ \implies &\le \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)2\left(\frac{\sqrt{\kappa_0}-1}{\sqrt{\kappa_0}+1}\right)^{k-1} && (4.2.39)\end{aligned}$$

We applied the Chebyshev Bound theorem proved in the previous part. And $\kappa_0 = \lambda_n/\lambda_2$, and that is the maximal bound for the absolute value of the polynomial.

Take notice that it's not immediately clear which type of outlier eigenvalue makes the convergence better or worse, but in this case, the weight $w(x)$ introduces a term that grows inversely proportional to λ_1. □

4.3 From Conjugate Gradient to Lanczos

We had been brewing the fact that the Iterative Lanczos Algorithm and the Conjugate gradient algorithm are related. From the previous discussion we can observe that:

1.) Both Lanczos and CG terminates when the grade of Krylov subspace is reached. For Lanczos it's $\mathcal{K}_k(A|q_1)$ and for CG it's $\mathcal{K}_k(A|r_0)$.

2.) Both Lanczos and CG generate orthogonal vectors, for Lanczos they are the q_i vector and for CG they are the r_i vectors.

These two properties, in particular, are hinting at an equivalence between the residual vectors r_j from CG and the orthogonal vectors q_j from Lanczos. However, it's also not entirely obvious because CG is derived using CDM in the first section, and yet it doesn't make use of any orthogonal projector. Furthermore, one might also notice that Iterative Lanczos are for General Symmetric Matrices while CG is only for positive definite matrices. To see how everything ties together, we have to go both directions to show the connections between these two iterative algorithms, which are what this section and the subsequent section about. Here, we refer to Lanczos vectors as the sequence of q_j generated by the Iterative Lanczos Algorithm.

For this subsection, our objective is to establish the equivalence between the parameters from the Lanczos algorithm: α_k, β_k, q_k and the a_j, b_j, r_j from the conjugate gradient algorithm. We establish it by going from the conjugate gradient to the Lanczos Algorithm.

Proposition 4.5. The residual and the Lanczos vectors have the following relations:

$$q_1 = \hat{r}_0 \tag{4.3.1}$$
$$q_2 = -\hat{r}_1 \tag{4.3.2}$$
$$\vdots \tag{4.3.3}$$
$$q_j = (-1)^{j+1}\hat{r}_{j+1} \tag{4.3.4}$$

Here, $\hat{r}_j := r_j/\|r_j\|$ and we can fill in the Lanczos Tridiagonal matrix using the CG parameters.

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le k-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le k-2 \\ \alpha_1 = a_0^{-1} \\ \beta_1 = \frac{\sqrt{b_0}}{\alpha_0} \end{cases} \tag{4.3.5}$$

Where α_j for $1 \le j \le n-1$ are the diagonal of the tridiagonal matrix T_k generated by Lanczos, and β_j for $2 \le j \le k-2$ are the lower and upper subdiagonals of the matrix T_k.

Proof. The proof is long and it's presented in appendex item: B.7 □

CG is a special case of applying the Lanczos Iterations with $q_1 = r_0$ to a positive definite matrix. However there are still questions left.

1.) How are the solutions x_k generated by CG related to the Lanczos Iterations?

2.) How are the A-Orthogonal vectors p_k from CG related to Lanczos?

Remark 4.3.1 (A Better Terminations Conditions for CG)**.** The derivation hinted at a better termination condition for the CG algorithm. Because CG is equivalent to the Lanczos Iterations initialized with $q_1 = r_0$, and we can directly apply from proposition 3.4.2 to get the precise number of iterations of CG under exact arithmetic given r_0, improving the bound we got from proposition 4.2.

4.4 From Lanczos to Conjugate Gradient

In this section, we state the Conjugate Gradient parameters using parameters from the Lanczos iterations.

The relations exposes the fact that CG is a special case of Lanczos. In the end we discuss the key that can inspire solvers for symmetric indefinite systems of linear equations.

We start off by considering the LU decomposition of the tridiagonal matrix of Lanczos and use its entries to express the scalars a_k, b_k in CG. In addition, we express the conjugate vectors p_k as a short recurrence of the Lanczos vectors q_k.

4.4.1 Matching the Residual and Conjugate Vectors

In this section, we state the connections between the solution x_k, p_k from CG with the Lanczos vectors and the symmetric tridiagonal matrix generated by Lanczos which plays an role for the analysis for the floating-point behavior of the CG algorithm in the later parts of the paper. In the remark we highlight some of the insights that the connections lead to solvers for symmetric indefinite systems.

Proposition 4.6 (Lanczos Vectors and Residuals)**.** The Q_k is the orthogonal matrix generated by Lanczos Iteration. To match the Krylov Subspace generated by the Lanczos iterations and CG, we initialize $q_1 = \hat{r}_0$, then the following relationship between Lanczos and CG occurs between their parameters:

$$
\begin{cases}
y_k = T_k^{-1}\beta\xi_1 \\
x_k = x_0 + Q_k y_k \\
r_k = -\beta_k \xi_k^T y_k q_{k+1}
\end{cases}
\tag{4.4.1}
$$

The quantities α_i, β_i are the diagonal and the sub or super diagonal of the matrix T_k from the Iterative Lanczos Algorithm but β without the subscript denotes $\|r_0\|$. r_k is the residual from the Conjugate Gradient Algorithm, and Q_k is the orthogonal matrix generated from the Lanczos Algorithm. For notations, we use ξ_i to denote the ith canonical basis vector.

Proof. To start recall that the Lanczos Algorithm Asserts the following recurrences:

$$
AQ_k = Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix}
\tag{4.4.2}
$$

Recall that the Conjugate Gradient algorithm takes the guesses from the affine span of $x_0 + \mathcal{K}_k(A|r_0)$, from section CG and Krylov Subspace 3.3.6 we know that: $p_k \in \mathcal{K}_{k+1}(A|r_0)$, the matrix P_k, Q_k spans the same subspace, and that means:

$$\begin{aligned}
x_{k+1} &= x_0 + Q_k y_k && (4.4.3)\\
r_{k+1} &= r_0 - AQ_k y_k && (4.4.4)\\
Q_k^H r_{k+1} &= Q_k^H r_0 - Q_k^H AQ_k y_k && (4.4.5)\\
\implies 0 &= \beta\xi_1 - T_k y_k && (4.4.6)\\
y_k &= T_k^{-1}\beta\xi_1 && (4.4.7)
\end{aligned}$$

Now to get the residual we simply consider:

$$\begin{aligned}
r_{k+1} &= r_0 - AQ_k y_k && (4.4.8)\\
&= r_0 - AQ_k T_k^{-1}\beta\xi_1 && (4.4.9)\\
\implies &= \beta q_1 - AQ_k T_k^{-1}\beta\xi_1 && (4.4.10)\\
&= \beta q_1 - Q_{k+1}\begin{bmatrix} T_k \\ \beta_k\xi_k^T \end{bmatrix} T_k^{-1}\beta\xi_1 && (4.4.11)\\
&= \beta q_1 - (Q_k T_k + \beta_k q_{k+1}\xi_k^T)T_k^{-1}\beta\xi_1 && (4.4.12)\\
&= \beta q_1 - (Q_k\beta\xi_1 + \beta_k q_{k+1}\xi_{k+1}T_k^{-1}\beta\xi_1) && (4.4.13)\\
&= -\beta_k q_{k+1}\xi_k^T T_k^{-1}\beta\xi_1 && (4.4.14)
\end{aligned}$$

On the third line we recall the fact that $q_1 = \hat{r}_0$ which initialized the Krylov Subspace for the Lanczos Iteration. At the 4th line, we make use of the Lanczos Vector recurrences and we simply substituted it.

By observing the fact that $\xi_k^T T_k^{-1}\xi_1$ the $(k, 1)$ element of the matrix T_k^{-1} which is a scalar, we can conclude that the residual from CG and the Lanczos vector are scalar multiple of each other, therefore, r_k from the CG must be orthogonal as well. □

Proposition 4.7 (Lanczos Vectors and Conjugate Vectors)**.** The P_k matrix as derived in the CG algorithm can be related to the Lanczos iterations by the formula:

$$P_k = Q_k U_k^{-1} \qquad (4.4.15)$$

Where $T_k = L_k U_k$, representing the LU decomposition of the tridiagonal matrix T_k from the Lanczos Iterations. Because of the tridiagonal nature of the matrix T_k, L_k will be a unit bi-diagonal matrix and U_k will be an upper bi-diagonal matrix.

Proof. To prove it, we start by considering the x_k at step k of the iterations:

$$\begin{aligned}
x_k &= x_0 + Q_k y_k && (4.4.16)\\
&= x_0 + Q_k T_k^{-1}\beta\xi_1 && (4.4.17)\\
&= x_0 + Q_k U_k^{-1} L_k^{-1}\beta\xi_1 && (4.4.18)\\
&= x_0 + P_k L_k^{-1}\beta\xi_1 && (4.4.19)
\end{aligned}$$

So far we have written the solution vector x_k. Next, we are going to prove that the matrix

P_k indeed consists of vectors that are A-orthogonal. To show that we consider:

$$
\begin{aligned}
& P_k^T A P_k && (4.4.20)\\
&= (Q_k U_k^{-1})^T A Q_k U_k^{-1} && (4.4.21)\\
&= (U_k^{-1})^T Q_k^T A Q_k U_k^{-1} && (4.4.22)\\
&= (U_k^{-1})^T T_k U_k^{-1} && (4.4.23)\\
&= (U^{-1})_k^T L_k && (4.4.24)
\end{aligned}
$$

Reader please observe that U_k is upper triangular, therefore, it's inverse it's also upper triangular, therefore, U_k^{-T} is lower triangular, and because L_k is also lower triangular, their product is a lower triangular matrix, and therefore, the resulting matrix above is lower triangular, however, given that $P_k^T A P_k$ is symmetric, therefore, $U_k^{-T} L_k$ will have to be symmetric as well, and a matrix that is lower triangular and symmetric has to be diagonal. Therefore, the columns of P_k are conjugate vectors. □

4.4.2 Matching the a_k, b_k in CG

Similar to how we can generate the tridiagonal matrix for the Lanczos iterations with $q_1 = \hat{r}_0$, we can also generate the parameters a_k, b_k in the CG algorithm using parameters from the Lanczos Iterations. To achieve it, one can simply build up the recurrences for the y_k vectors using the elements from the L_k, U_k matrix which comes from LU decomposition of the T_k matrix. This will come at the expense of losing some degree of accuracy because it's equivalent to doing the LU decomposition of T_k without pivoting, but it comes at the advantage computing $\xi_k^T T_k^{-1} \xi_1$ with as little efforts as possible. Let's take a look.

For discussion in this section, we briefly switch the indexing and let it start counting from one instead of zero.

$$
P_k = \begin{bmatrix} p_1 & p_2 & \cdots & p_k \end{bmatrix} \quad Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{4.4.25}
$$

Next, when T_k is invertible, we consider the LU decomposition of the symmetric tridiagonal matrix (See remark for more information about the conditions for invertibility of the matrix):

$$
T_k = L_k U_k = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix} \begin{bmatrix} u_1 & \beta_1 & & \\ & u_2 & \beta_2 & \\ & & \ddots & \beta_{k-1} \\ & & & u_k \end{bmatrix} \tag{4.4.26}
$$

The upper diagonal of U_k is indeed the same as the upper diagonal of the symmetric tridiagonal matrix T_k. And recall the expression for x_k from the previous section, we have:

$$
\begin{aligned}
x_k &= x_0 + P_k L_k^{-1} \beta \xi_1 && (4.4.27)\\
x_k - x_{k-1} &= P_k L_k^{-1} \beta \xi_1 - P_{k-1} L_{k-1}^{-1} \beta \xi_1 && (4.4.28)\\
&= P_k \beta (L_k^{-1})_{:,1} - P_{k-1} \beta (L_{k-1}^{-1})_{:,1} && (4.4.29)\\
&= \beta (L_k^{-1})_{k,1} P_k && (4.4.30)\\
\implies x_k &= x_{k-1} + \beta (L_k^{-1})_{k,1} p_k && (4.4.31)
\end{aligned}
$$

On the third line, we factor out the last column for the matrix P_k. Next, we wish to derive the recurrence between p_{k+1} and p_k. Which is:

$$
\begin{aligned}
P_k &= Q_k U_k^{-1} && (4.4.32)\\
P_k U_k &= Q_k && (4.4.33)\\
\implies \beta_{k-1}p_{k-1} + u_k p_k &= q_k && (4.4.34)\\
u_k p_k &= q_k - \beta_{k-1}p_{k-1} && (4.4.35)\\
p_k &= u_k^{-1}(q_k - \beta_{k-1}p_{k-1}) && (4.4.36)
\end{aligned}
$$

In fact, a short recurrence can be built for u_k^{-1} and $(L_k^{-1})_{k,1}$, we stated it below:

$$
\begin{cases}
u_{k+1} & = \alpha_{k+1} - \beta_k^2/u_k \\
l_k & = \beta_k/u_k \\
(L_{k+1}^{-1})_{k+1,1} & = -l_k(L_k^{-1})_{k,1}
\end{cases}
\tag{4.4.37}
$$

The derivation for above recurrence is more relevant to the LU decomposition of a Tridiagonal matrix, which is a digression and we put the proof in appendix item B.8.

Remark 4.4.1. Using the Lanczos iterations, we derived the conjugate gradient without using the fact that A is symmetric positive definite, which hinted at potential new algorithms that can solve symmetric indefinite systems directly.

The Lanczos Algorithm for linear systems (we refer to the method derived in the above section) is a special case of FOM [11] when the matrix A is symmetric. The above algorithm is just FOM with a short term recurrence for its parameters, and it's based on convenience of solving a symmetric tridiagonal matrix. It's implied from the above derivation that under exact arithmetic, CG can be applied to a symmetric indefinite system, if we have the luck where T_j is non-singular for all $j \le k$. Recall that when we derived the CG algorithm, we convert it into solving the system: $P_k^T r_0 = P_k^T A P_k w$, and when the matrix A is indefinite, we can still solve the system and get the saddle point for the indefinite error norm.

However there are two problems solving Symmetric Indefinite using the CG. The first problem is the bad numerical accuracy of using the recurrence to perform LU decomposition on T_K without pivoting. Another problem is T_k can be singular during some iterations of the Lanczos Algorithm. For example, T_j will become singular if A is a symmetric tridiagonal matrix with zeros as its diagonals and all ones on its subdiagonals. The good part is that T_j never will not become singular for more than 2 consecutive iteration[3]. These problems can be overcome by considering something other than LU without pivoting. In fact, there is work by C. C. Paige and M. A. Saunders which extends the idea and derives algorithms that can solve a symmetric indefinite system without using the norm equation [7].

5 Effects of Floating-Point Arithmetic

In this section, we highlight the practical concerns of the algorithm and showcase it with numerical experiments and some analysis, in hope to get deeper insights about the behaviors of Lanczos and Conjugate Gradient under floating-point arithmetic.

5.1 Partial Orthogonalization and Full Orthogonalization

The floating-point errors inside the CG algorithm accumulates and manifested as a loss of orthogonality and loss of conjugacy for the vectors r_k, p_k. Many stable algorithm such as Modified Gram Schmidt orthgonalize the current vector against all previous vectors to assert orthogonality of the basis vector, CG on the other hand, relies only on the short recurrences of r_k, p_k. One way to mitigate the effect is to use the CDM algorithm's projector to re-orthogonalize the conjugate vectors and then reorthogonalized residual vectors against all previous residuals vectors. Such idea is not new and it's stated in the original paper by Hestenes and Stiefel back in 1952[6]; here we present the second more computationally expensive idea in the paper by Hestenes and Stiefel, but using the quantities we derived in the first section.

Recall the proof for proposition 3.7. Next, we inductively consider the case where the newest residual vector is involving some round-off error $\overline{r}_{k+1}$ and it breaks the orthogonality conditions $\overline{r}_{k+1} \perp r_j \; \forall \; 0 \leq j \leq k$:

$$(AP_k)^T \overline{r}_k = \begin{bmatrix} \langle p_0, A\overline{r}_k \rangle \\ \langle p_1, A\overline{r}_k \rangle \\ \vdots \\ \langle p_{k-1}, A\overline{r}_k \rangle \end{bmatrix} \tag{5.1.1}$$

$$= a_{k-1}^{-1} \langle r_k, (r_{k-1} - r_k) \rangle \xi_k + \sum_{j=0}^{k-1} \langle p_j, A\overline{r}_k \rangle \xi_j \tag{5.1.2}$$

$$p_k := \overline{r}_k + b_k p_k - \frac{\langle \overline{r}_k, r_{k-1} \rangle}{\langle r_{k-1}, r_{k-1} \rangle} p_k - \sum_{j=0}^{k-1} \frac{\langle p_j, A\overline{r}_k \rangle}{\langle p_k, Ap_k \rangle} p_j \tag{5.1.3}$$

$$r_k := \overline{r}_k - \sum_{j=0}^{k-1} \langle \hat{r}_j, \overline{r}_k \rangle \hat{r}_j \tag{5.1.4}$$

Here, we generate the conjugate vectors p_k correctly by faithfully expanding the term $(AP_k)^T \overline{r}_k$, and then we update the residual $\overline{r}_k$ into r_k by orthogonalizing it against all previous residual vectors. Such procedure requires expensive storage of previous vectors p_k. One can use alternative formulas to A-orthogonalize p_k and re-orthogonalize r_k. In addition, we have the options for partially orthogonalizing the r_k, p_k vectors for less memory usage.

5.2 Relative Errors of CG Under Floating-Point Arithmetic

We investigate the relative energy norm of the error for fully re-orthogonalized CG, partially re-orthogonalized CG, and CG without re-orthogonalization numerically.

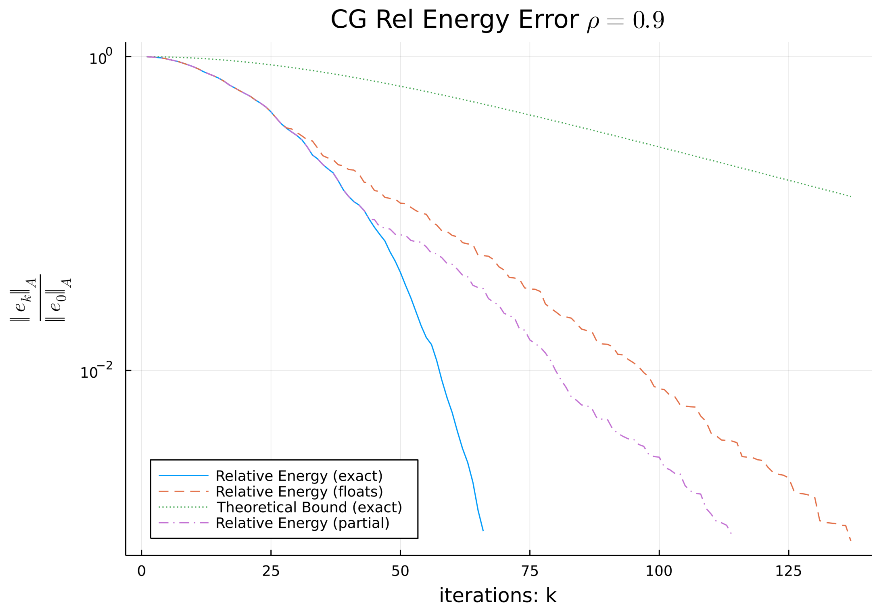
5.2.1 Experiments

Here we use full re-orthogonalization to emulate the effects of exact arithmetic. the step required for convergence is less than or equal to the number of unique eigenvalues for the symmetric definite matrix, this is established in part Termination of CG 4.2 of the discussion.

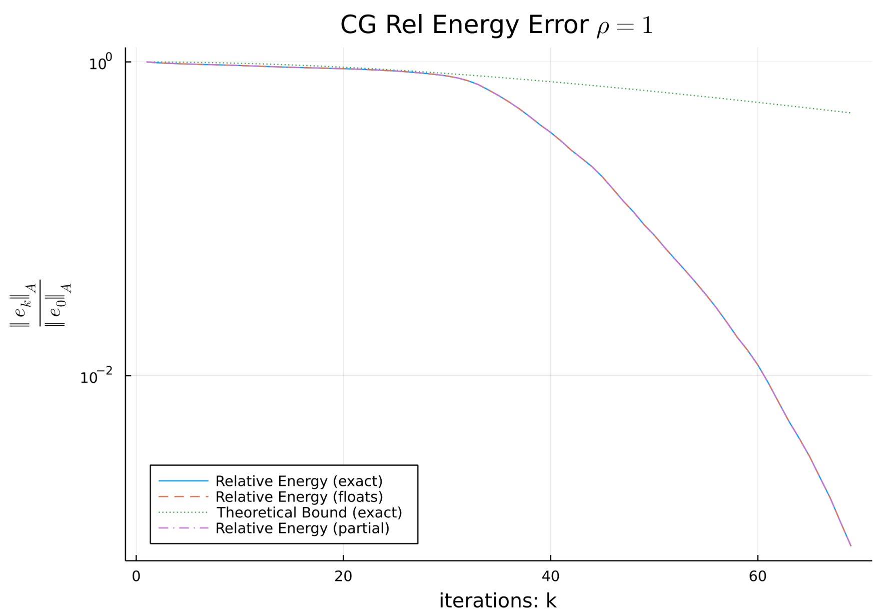
However, in practice, this is not always the case. Similar experiments are conducted in Greenbaum's book chapter[5] 4. In this section, we replicate the same set of experiments using modern Julia to showcase the extra steps required for the CG algorithm to converge. For testing the convergence of the algorithm, we use float16 (16 digits binary digits), $\lambda_{\min} = 1e-4, \lambda_{\max} = 1$ to exaggerate the convergence a bit, and the RHS vector b is all ones and $x_0 = b + \epsilon$ (ϵ is some tiny random vector as noise). The spectrum of the matrix we are using is taken to be the same from Greenbaum's book [5].

$$\lambda_{\min} + \left(\frac{j}{N-1}\right)(\lambda_{\max} - \lambda_{\min})\rho^{N-j+1} \quad \forall\, 1 \le j \le N-1,\ 0 \le \rho \le 1 \tag{5.2.1}$$

If the value of ρ is close to zero, then the eigenvalues are clustered around the origin, if it's close to 1, then the eigenvalues are tend to be more evenly distributed around the interval $[\lambda_{\min}, \lambda_{\max}]$.



(a) $\rho = 0.9$



(b) $\rho = 1$

Figure 5.2.1: The relative energy norm error for different methods. Blue solid line: The exact conjugate gradient convergence. Purple dot dashed: The conjugate gradient that is partially orthogonalized with previous 8 residual and conjugate vectors. orange dashed: the Original conjugate gradient. Green dot: The tighter theoretical upper bound derived by Chebyshev (4.2.20). The matrix is 256×256

The Chebyshev bound is no longer a tight bound because the distribution of the eigenvalues is not perfectly uniform. The partially orthogonalized methods diverge from the exact error after more steps of iterations compared to the relative error without any orthogonalizations. These are seen in (fig 5.2.1a). In contrast, on the right, then the eigenvalues are uniformly distributed, the convergence of all 3 types of methods aligns with the exact convergence in (fig 5.2.1b).

Remark 5.2.1. The convergence is disappointing under floating-point arithmetic and the promised efficiency of the algorithm is not there anymore even if the matrix is not necessarily ill-conditioned. Just from (fig 5.2.1) it seems like outlier eigenvalues provide fast convergence under full orthogonalization, but not for floating point.

However, please observe from (fig 5.2.1b) that, all these 3 methods are identical, and the Chebyshev bound is relatively tight at the first few iterations of the algorithm, which is related to the discussion later and linked to the fact that the characteristic polynomial of T_k tends to approximate spectrum of matrix A where eigenvalues are sparse before converging to area of the spectrum where eigenvalues are denser.

5.3 Paige's Convergence rate of CG under Floating Points

In this section we present the backwards analysis of the CG convergence rate in a more thorough manner and examine its consequences. When floating-point arithmetic is used, the eigenvalues of the tridiagonal matrices might introduce ghost eigenvectors for the Lanczos Iterations, and using the equivalence of the Lanczos iterations and CG, we can capture a posteriori bound on how much the error is exactly during the iterations of CG.

5.3.1 Bounding the Relative Residuals

Recall from proposition Proposition 4.6 that the residual of the CG can be expressed in terms of the Lanczos vectors. However, the Lanczos iterations under floating doesn't produces perfectly orthogonal Lanczos Vectors (The lost of orthogonality is experimented and visualized in the next section), $\tilde{Q}_k$ is not quiet orthogonal, which would also means that $\tilde{Q}_k^H A\tilde{Q}_k \approx T_k$ where T_k is the results from the Lanczos iterations will not equal to $\tilde{Q}_k^T A\tilde{Q}_k$. However the Lanczos iterations will still solve for y_k using the expression $y_k = \beta T_k^{-1}\xi_1$, the algorithm still thought T_k produced by itself is perfectly tridiagonal. As a result, the algorithm never quite finds the optimal under the conjugate basis. At first glance, tiny round-off errors in the Lanczos vectors are very problematic.

Surprisingly, the recurrence formula for Lanczos still holds to some extent, in addition we can leverage the fact that y_k is solved exactly and assume: $y_k = \beta T^{-1}\xi_1$ is at least, exact. Then it left us with fewer types of floating-point errors to keep track of. Next, we proceed to look for the residual of the CG algorithm by stating that the Lanczos recurrences[1]:

$$AQ_k = Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} + F_k \tag{5.3.1}$$

Reader, please reflect on the fact that the Q_k which is not orthogonal, and we are fixing the recurrences with F_k, a matrix representing the floating error to correct it so that the equality

[1]statement 2.11 from C. C. Paige's 1980 paper.[8]. The result is surprising in the sense that Q_k being none orthogonal in finite precision won't affect the recurrence.

holds true. $\|F_k\|$ is small and it's on the magnitude of $\mathcal{O}(\epsilon\|A\|)$.

$$r_k = r_0 - AQ_k y_k \tag{5.3.2}$$
$$r_k = r_0 - \left(Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} + F_k\right) y_k \tag{5.3.3}$$
$$r_k = \underbrace{\left(r_0 - Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} y_k\right)}_{=-\beta_k \xi_k^T y_k q_{k+1}} + F_k \beta T_k^{-1}\xi_1 \tag{5.3.4}$$
$$\implies \frac{\|r_{k+1}\|}{\|r_0\|} \le \beta_k \|\xi_k^T T_k^{-1}\xi_1 q_{k+1}\| + \|F_k T_k^{-1}\xi_1\| \tag{5.3.5}$$
$$\frac{\|r_{k+1}\|}{\|r_0\|} \le \beta_k |\xi_k^T T_k^{-1}\xi_1| + \|F_k\|\|T_k^{-1}\xi_1\| \tag{5.3.6}$$

We make use of Proposition 4.6 and obtain a similar expression because it didn't make use of the fact that Q_k is orthogonal. This time, we take F_k into account. The residual is now bounded by the sum of scalar $\xi_k^T T_k^{-1}\xi_1$ and the floating-point error matrix F_k produced by the Lanczos iterations.

Remark 5.3.1 (When Finite Arithmetic Lanczos is Exact)**.** We can bound the first term that made up the upper bound for the residual of CG using previous convergence results of CG under exact arithmetic; recall CG convergence rate (theorem 3). It can be applied here for the first term in (5.3.6): $\beta_k|\xi_k^T T_k^{-1}\xi_1|$.

This is true because if we were to perform an CG on the T_{k+1} produced by the finite precision algorithm with the initial Lanczos vector q_1 being $\xi_1^{(n)}$, then its residual $\overline{r}_k$ of equivalent CG would be exact and it's given as $-\beta_k T_k^{-1}\xi_k q_{k+1}$, but with $q_{k+1} = \xi_{k+1}^{(n)}$, the $k+1$ th standard basis vector in $\mathbb{R}^n$, and $T_k = (T_{k+1})_{1:k,1:k}$. And to our excitement, we already have the bound for $\overline{r}_{k+1}$ proven in theorem 3.

5.3.2 Paige's Theorem and Floating-point Convergence of CG

We now introduce a new theorem proposed by Paige in chapter 4 of Greenbaum's book[5], originally appear in 3.48 of C.C Paige's Thesis([8]). It gives a bound to the floating-point errors for the CG by bounding the condition number of T_k from Lanczos iterations. It's stated as follows:

Theorem 4 (Paige's Theorem)**.** The eigenvalues $\theta_i^{(j)}, i = 1, \cdots, j$ of the tridiagonal matrix T_j satisfies:

$$\lambda_1 - j^{5/2}\epsilon_2\|A\| \le \theta_i^{(j)} \le \lambda_n + j^{5/2}\epsilon_2\|A\| \tag{5.3.7}$$
$$\epsilon_2 := \sqrt{2}\max\{6\epsilon_0, \epsilon_1\} \tag{5.3.8}$$

Along with this theorem, the following quantities from Paige are also defined and cited in

Greenbaum book chapter 4.

$$\epsilon_0 \equiv 2(n+4)\epsilon \tag{5.3.9}$$

$$\epsilon_1 \equiv 2(7+m\| \, |A| \, \|/\|A\|)\epsilon \tag{5.3.10}$$

$$\epsilon_0 < \frac{1}{12} \quad k(3\epsilon_0 + \epsilon_1) < 1 \tag{5.3.11}$$

$$\|F_k\| \leq \sqrt{k}(\epsilon_1)\|A\| \tag{5.3.12}$$

$$\|q_j^T q_j - 1\| \leq 2\epsilon_0 \tag{5.3.13}$$

$$\beta_j \leq \|A\|(1 + (2n+6)\epsilon + j(3\epsilon_0 + \epsilon_1)) \tag{5.3.14}$$

The quantity k is the current iterations number of the Lanczos Iterations, $j \leq k$. m is the maximum number of non-zero elements in the matrix A. Using Paige's theorem, we can bound the condition number for the matrix T_{k+1} produced by the finite precision Lanczos, which is given by:

$$\tilde{\kappa} = \frac{\lambda_n + (k+1)^{5/2}\epsilon_2\|A\|}{\lambda_1 - (k+1)^{5/2}\epsilon_2\|A\|} \tag{5.3.15}$$

Using (remark 5.3.1), we can make the following proposition

Proposition 5.1.

$$|\beta_k \xi_k^T T_k^{-1} \xi_1| \leq 2\sqrt{\tilde{\kappa}} \left(\frac{\sqrt{\tilde{\kappa}} - 1}{\sqrt{\tilde{\kappa}} + 1} \right)^k \tag{5.3.16}$$

Where $\tilde{\kappa}$ is the bound of the condition number of the T_k matrix (from (5.3.15)), the tridiagonal matrix produced by the finite precision Lanczos.

Proof. Using the lemma A.1.1 in appendix, we can derive the relations between the 2-norm of the relative residuals and the energy norm of the relative error:

$$\frac{\|Ae_k\|}{\|Ae_0\|} \leq \kappa(T_k)\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\sqrt{\kappa(T_k)} \left(\frac{\sqrt{\tilde{\kappa}} - 1}{\sqrt{\tilde{\kappa}} + 1} \right)^k \tag{5.3.17}$$

$$\frac{\|r_k\|}{\|r_0\|} = |\beta_k \xi_k^T T_k^{-1} \xi_1| \quad \text{by (remark 5.3.1)} \tag{5.3.18}$$

$$\implies |\beta_k \xi_k^T T_k^{-1} \xi_1| \leq 2\sqrt{\tilde{\kappa}} \left(\frac{\sqrt{\tilde{\kappa}} - 1}{\sqrt{\tilde{\kappa}} + 1} \right)^k \tag{5.3.19}$$

The third inequality is simply from CG Convergence Rate (theorem 3) when we assume that the eigenvalues are uniformly distributed convex hull of the spectrum of A. The first fraction is actually the relative error of the 2-norm of the residual because $Ae_k = r_k$ by definition. Substituting the quantity $\kappa(T_k)$, the condition number of the matrix T_k, which we figured out using Paige's theorem and denoted it as $\tilde{\kappa}$. □

Finally, if we assume that T_k^{-1} is actually invertible, which requires that the conditions for all the quantities: ϵ_0, ϵ_1 holds true, and $\lambda_1 - (k+1)^{5/2}\epsilon_2\|A\| > 0$. Finally, we make can bound the relative residual of the CG algorithm by considering:

$$\frac{\|r_{k+1}\|}{\|r_0\|} \le \beta_k\|\xi_k^T T_k^{-1}\xi_1 q_{k+1}\| + \|F_k T_k^{-1}\xi_1\| \tag{5.3.20}$$

$$\le \beta_k|\xi_k^T T_k^{-1}\xi_1|\|q_{k+1}\| + \|F_k\|\|T_k^{-1}\xi_1\| \tag{5.3.21}$$

$$\le 2\|q_{k+1}\|\sqrt{\tilde{\kappa}}\left(\frac{\sqrt{\tilde{\kappa}}-1}{\sqrt{\tilde{\kappa}}+1}\right)^k + \sqrt{k}(\epsilon_1)\|A\|\|T_k^{-1}\| \tag{5.3.22}$$

Now, observe that $|q_j^T q_j - 1| \le 2\epsilon_0$ from (5.3.13), which implies that $\|q_{k+2}\|^2 \le (1+3\epsilon_0)$ which is $\|q_{k+1}\| \le \sqrt{1+2\epsilon_0}$. In pursuit of mathematical beauty, we look for alternative expression for the quantity $\|A\|\|T_k^{-1}\|$ giving us:

$$\|A\|\|T_k^{-1}\| = \frac{\lambda_n}{\lambda_1 - k^{5/2}\epsilon_2\|A\|} \le \tilde{\kappa} \tag{5.3.23}$$

$$\implies \frac{\|r_{k+1}\|}{\|r_0\|} \le 2\sqrt{1+2\epsilon_0}\sqrt{\tilde{\kappa}}\left(\frac{\sqrt{\tilde{\kappa}}-1}{\sqrt{\tilde{\kappa}}+1}\right)^k + \sqrt{k}(\epsilon_1)\tilde{\kappa} \tag{5.3.24}$$

Which completes the proof for the upper bound on the convergence rate for the Conjugate Gradient Method under floating-point arithmetic.

Remark 5.3.2. This proof showed that the T_{k+1} generated by floating point CG is non-singular, $\beta_k \neq 0$, the restrictions for the quantities of theorem 4 then the CG method will continue to converge in the future iterations. So in layman's terms, it doesn't matter if the round-off error accumulated, Conjugate Gradient will converge as long as the problem is not too insane, or A being too pathological to deal with.

Finally, I want to point out the fact that Paige's theorem (theorem 4) is derived using forward error analysis on the Lanczos Iterations, which is the absolute worst case. For most cases in modern computing platforms, the summation process of vector dot products has much higher floating-accuracy compare to older computing platforms due to the use of parallelism, or floating-point specific summation instructions, which reduces the relative sizes for the summands, hence reducing the total round-off error accumulations. The bound of convergence rate we derive can be a huge over estimation.

5.4 Ghost Eigenvalues and Losing Orthogonality

The name Ghost Eigenvalues refers to the phenomena where the Lanczos Algorithm seems to produce tridiagonal matrix T_k whose eigenvalues are clustered extremely close to a simple eigenvalues of the matrix A, when in fact, those extremely close eigenvalues are a single eigenvalue of the original matrix A's spectrum.

The name for it comes from Trefethen's Book[14]. This phenomena is more pronounce to eigenvalues in the exterior of A's spectrum. We know for a fact that the tridiagonal matrix produced via Lanczos can't have any repeated eigenvalues (appendix item B.5). What happens in this case is the floating point error propagating through the Lanczos Iterations causing lost of orthogonality of Q_k and eventually produce ghost eigenvalues.

5.4.1 Ghost Eigenvalues Experiments

Here, we conducted numerical experiments and carefully reproduce the phenomena for a diagonal matrix A with diagonals given by the formula:

$$\lambda_i = \left(-1 + \frac{2(i-1)}{(n-1)}\right)^3 \quad 1 \le i \le n \tag{5.4.1}$$

where $A \in \mathbb{R}^{n\times n}$. This matrix is particularly good for reproducing the phenomena. For this experiment, we set $n = 64$ and we use Float64.

We run the Lanczos iterations with q_1 being the vector of all ones, we marked the smallest and largest 10 eigenvalues during the iterations and plotted their trajectories from iteration 20 to 64. The results can be seen in (figure 5.4.1 left). On figure 5.4.1 right, we made the plot for what would happen if the Lanczos iterations are free of numerical round-off error. We didn't use exact arithmetic, instead, we simply re-orthogonalized all the Lanczos vector q_k using all previously obtained Lanczos vectors to emulate the effect, which is just an Arnoldi iteration. Please bear in mind that there are eigenvalues in the middle interior part of the spectrum, they are just not plotted in the figure.

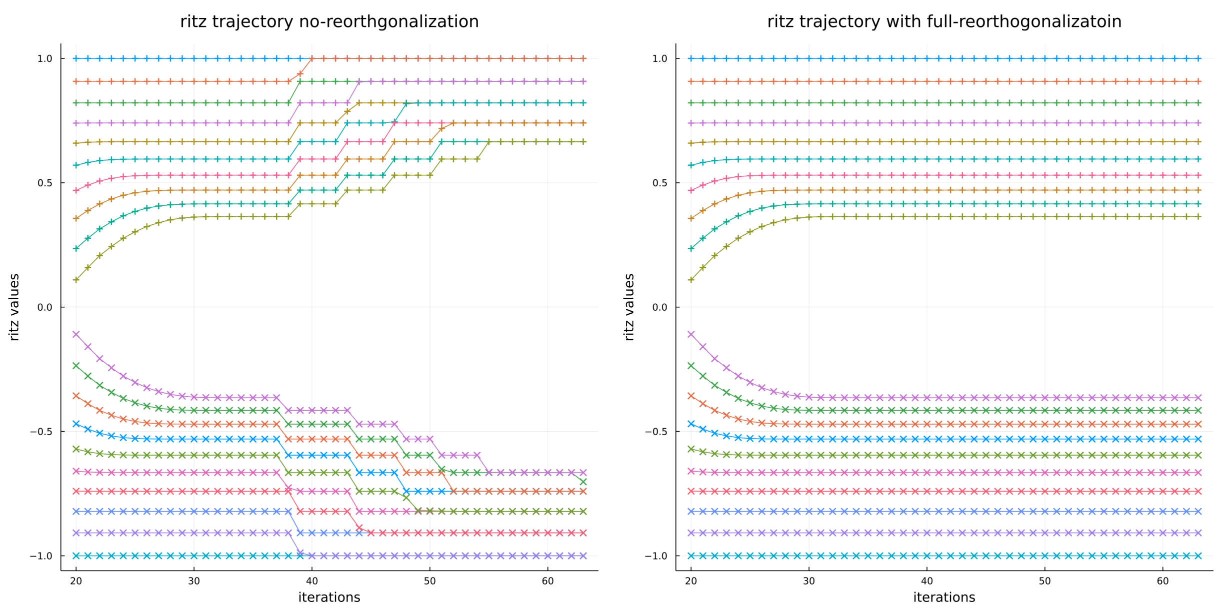


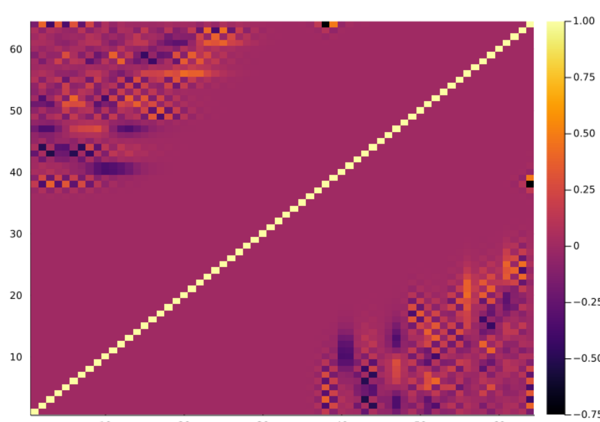
Figure 5.4.1: The highest and lowest 10 eigenvalues of the matrix T_k during the Lanczos Iterations are being tracked by their relative order. During each iteration, the first, the second, the third, ... etc eigenvalues of T_k are linked together by a line in a different color. Left is Lanczos with numerical round-off error, right is Lanczos iterations that fully re-orthogonalize q_k for each iteration which emulates the behaviors under exact arithmetic.

Recall from the Cauchy Interlace Theorem (theorem 6), the eigenvalues of the tridiagonal matrix T_{k+1} has to be in between each eigenvalues of T_k except for the first and the last eigenvalue of T_{k+1}. This implies that, the $\theta_i^{(k)}$, the i th eigenvalues during the k th iteration will move monotonically upwards or downwards during the Lanczos iterations. The ghost eigenvalues on the figure appear when some of the interior eigenvalues suddenly switch to another eigenvalue's trajectory that is on the exterior of the spectrum. It appears as though the matrix T_k has repeated eigenvalues which we know is not true due to (appendix item B.5), they are just very close.

However, judging the eigenvalues of the matrix T_k alone WILL NOT distinguish between two very close eigenvalues correspond to two different eigenvalues of A or it's due to the floating-point round-off error. It also will NOT tell whether the Lanczos vectors are losing orthogonality, even if the eigenvalue trajectories seem to suggest it. The lost of orthogonality must happen for the Lanczos vector while at the same time, we observed extremely close eigenvalues of T_k clustring around eigenvalues of A to confirm the fact that they are indeed ghost eigenvalues.

We can't tell it because if I keep the matrix T_k we used to produce fig 5.4.1 generated from a Lanczos iterations and use it as A with $q_1 - \xi_1$, then it will reproduce exactly the same graph as in left of fig 5.4.1 when we plot out the trajectories of the eigenvalues of T_k, but the Q_k in this case is the $k \times k$ identity matrix and it's exact (Using the exact same idea appeared in remark 5.3.1). Now consider performing another Lanczos iterations $A := T_k$ it but with the initial vector ξ_1, then we will exactly reproduce A itself because it's tridiagonal. But in this case, the eigenvalues of T_k are exact after termination of Lanczos iteration. In this case, all eigenvalues are actually presented in the original matrix A, which is just T_k, itself.

In fact, the ghost eigenvalues here are produced by floating-point errors because firstly we know what the actual eigenvalues of A is, we made A. To make sense of it better intuitively, we observe from the experiments that the loss of orthogonality of Q_k happens together with ghost eigenvalues on the spectrum of T_k. If the Q_k matrix is perfectly orthogonal, then there are no ghost eigenvalues, regardless of what the trajectories of the eigenvalues of T_k look like. In fact, a corresponding plot of $Q_k^H Q_k$ are plotted in (fig 5.4.1 left) for demonstrating the loss of orthogonality for the same diagonal matrix A proposed earlier. We plotted the heat map of the matrix $Q_k^H Q_k$ directly as well.



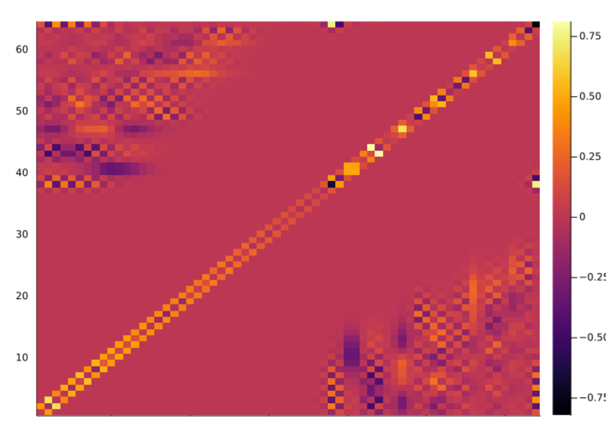


Figure 5.4.2: left: The heatmap of the plot of the absolute values of the matrix $Q_k^T Q_k$. right: The plot of $Q_k^T A Q_k$ from floating-point Lanczos iterations

In addition to the lost orthogonality of the matrix Q_k, we also visualized the actual tridiagonal matrix reproduced by $Q_k A Q_k$ which is plotted in (fig 5.4.1 right). Observe that most of the off tridiagonal entries are non-zero and relatively huge, which doesn't worry us too much because A itself has a large condition number. What is concerning is the blob of non-zero entries on the top left and the bottom right of the plot. And this is a significant loss of orthogonality created by floating point arithmetic.

Remark 5.4.1. As a final remark for these numerical experiments, I suggest an intuitive way of understanding them. Which will be useful when we actually wish to analyze it rigorously. Simply put, the Lanczos Iterations might "forget" about the eigenvalues when it converged (usually manifested as the stable trajectories of eigenvalues on the exterior of the spectrum for the matrix T_k in (fig 5.4.1)), and when it happens, the Lanczos vectors produced by the algorithm has lost its orthogonality correspondingly, which then causes the interior eigenvalues of T_k to shift, creating ghost eigenvalues that doesn't exist in the spectrum of A.

Secondly, there is another phenomenon of Ritz values during the iterations of Lanczos iterations called misconvergence. It describes the process which a Ritz value is stuck between two eigenvalues of A, stagnated for few iterations and then suddenly shifts away, which is extremely similar to the shifting we observed in figure 5.4.1. It happens when 2 eigenvalues of matrix A is extremely close to each other. It should not be confused with ghost eigenvalues because they are two distinct phenomena where misconvergence can happen under exact arithmetic. For more description of such phenomena, refer to the book by M. G. Cox and S. Hammarling [1].

5.4.2 Lanczos Vectors Losing Orthogonality on converged Ritzvectors

To gain a better understanding, let's define the notion of Ritz values and Ritz vectors. For our discussion, the Ritz value $\theta_i^{(k)}$ are the i th eigenvalues of the matrix T_k from the and the Ritz vectors are $Q_k s_i^{(k)}$ where s is the i th eigenvector for the matrix T_k. Recall from remark 3.4.2, the characteristic polynomial of T_k is the monic polynomial that minimizes the 2-norm error for the vector $p_k(A)q_1$; intuitively, the Ritz values and Ritz vectors approximates eigenvalues and eigenvectors of matrix A due to the apprixmating characteristic polynomial. Let's suppose that $s_i^{(k)}$ for T_k is a good approximation for λ_j, let's consider the Lanczos iterations recurrences:

$$AQ_k = Q_kT_k + q_{k+1}\beta_k\xi_k^T \tag{5.4.2}$$

$$AQ_k s_i^{(k)} = \theta_j^{(k)} Q_k s_i^{(k)} + q_{k+1}\beta_k\xi_k^T v \tag{5.4.3}$$

$$AQ_k s_i^{(k)} = \theta_j^{(k)} Q_k s_i^{(k)} + \beta_k q_{k+1}(s_i^{(k)})_k \tag{5.4.4}$$

$$AQ_k s_i^{(k)} - \theta_j^{(k)} Q_k s_i^{(k)} = \beta_k q_{k+1}(s_i^{(k)})_k \tag{5.4.5}$$

Upon brief examinations, convergence of the Ritz value $\theta_i^{(k)}$ depends on β_k and $(s_i^{(k)})_k$, intuitively as the Krylov subspace expands, it contains more and more space for the the eigenvectors of A, and the Ritz vector will have more "room" to get closer to the eigenvector of A, by the approximation property of Lanczos. Assuming good convergence of $s_i^{(k)}$ convergences so that the value of $\beta_k, (s_i^{(k)})_k$ are both small (it's true regardless of orthogonality of Q_k), then we consider the projection of most recent lanczos vector q_k onto the Ritz vector $Q_k s_i^{(k)}$, which is $q_k^T Q_k s_i^{(k)} = (s_i^{(k)})_k$. We expect the projection onto the Ritz vector to be small if the Ritz vector is converging to an eigenvector of A.

However, under floating-point arithmetic, once the Ritz vector $s_i^{(k)}$ is converging to an eigebvalue of A, then the projection of the latest Lanczos vector onto the Ritz vector begins to grow. On the plot showed in figure 5.4.2, we projected the log absolute value of $q_k^T Q_k s_i^{(k)}$

for $i = 1, 2, 3$ for all $k = 20, \cdots, 64$, and we used the same setup from the last section where we demonstrated ghost eigenvalues. For comparison, figure 5.4.2 showed us what happens in exact arithmetic. One very important observation to make from figure 5.4.2 is that the peak of the blue curve, projection onto the largest Ritz value happens around iteration 3.8, and around that exact same iteration in figure 5.4.1 is when the second-largest eigenvalue of T_k decides to shift over to the blue curve. Bear in mind that this is in a log plot, and without the log it looks like a sharp spike.

While floating arithematic may sometimes cause the most recent Lanczos vector to lose orthogoality against converged Ritz vectors, the converse is not true. The phenomena itself doesn't imply the fact that floating point error is present and it's causing lost of orthogonality of Lanczos iterations, much similar to what had been discussed about ghost eigenvalues. To illustrate, an experiment is conducted in this way:

1. T_n is generated from the diagonal matrix A from finite precision Lanczos iterations. The same setup from ghost eigenvalues 5.4.1.
2. Lanczos is then performed on T_k initalized with $q_1 = \xi_i$.
3. The projection the most recent lanczos vector q_k is projected onto $Q_k s_i^{(k)}$ for $1 \le i \le 3$. Notice that Q_k is gonna be the $k \times k$ identity matrix. ''

The results of the projection are showed in fig 5.4.2. Please observe that prjection onto the second Ritzvector $Q_k s_2^{(k)}$ seems to decrease and them jumped up again. This happens around the same iterations when the second largest eigenvalue of T_k shift to the trajectory of the largest eigenvalue in fig 5.4.1 left. However, this time the matrix Q_k is perfect orthogonal. Therefore, small ritz projections of the most recent Lanczos vector doesn't mean that the Ritz value has converged for all future iterations. If q_k is losing orthgonality against some Ritzvectors after it has converged, it doesn't mean that Q_k is losing orthogonality.

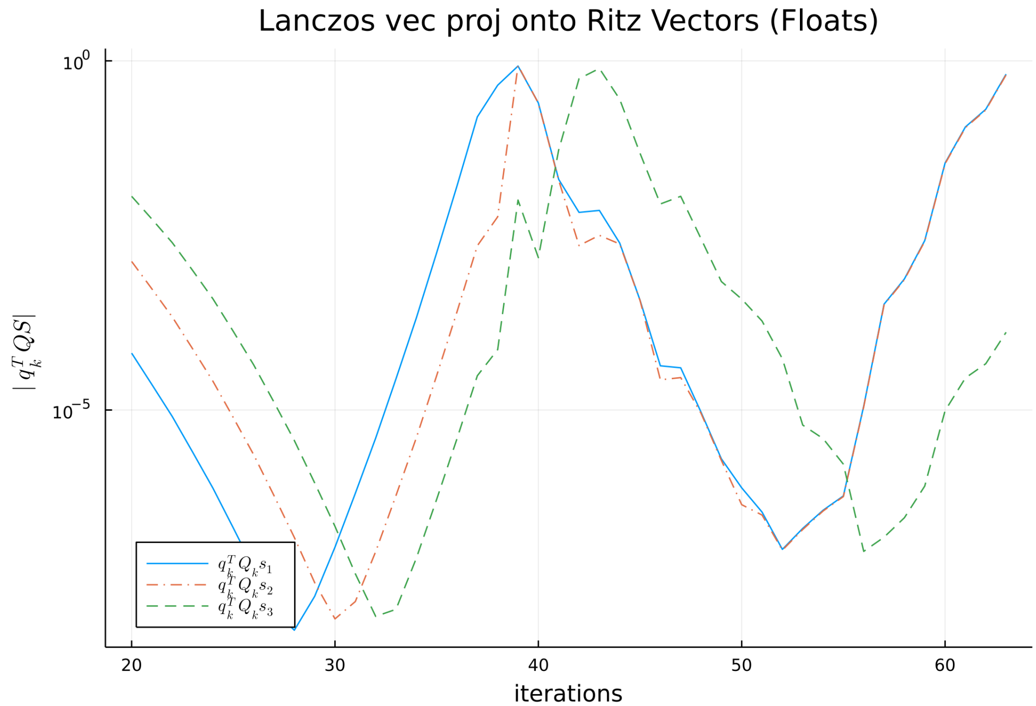


Figure 5.4.3: Projection of floating-point Lanczos vector q_k onto 3 of the largest Ritz vectors: $Q_k s_i^{(k)}$, for $i = k, k-1, k-2$

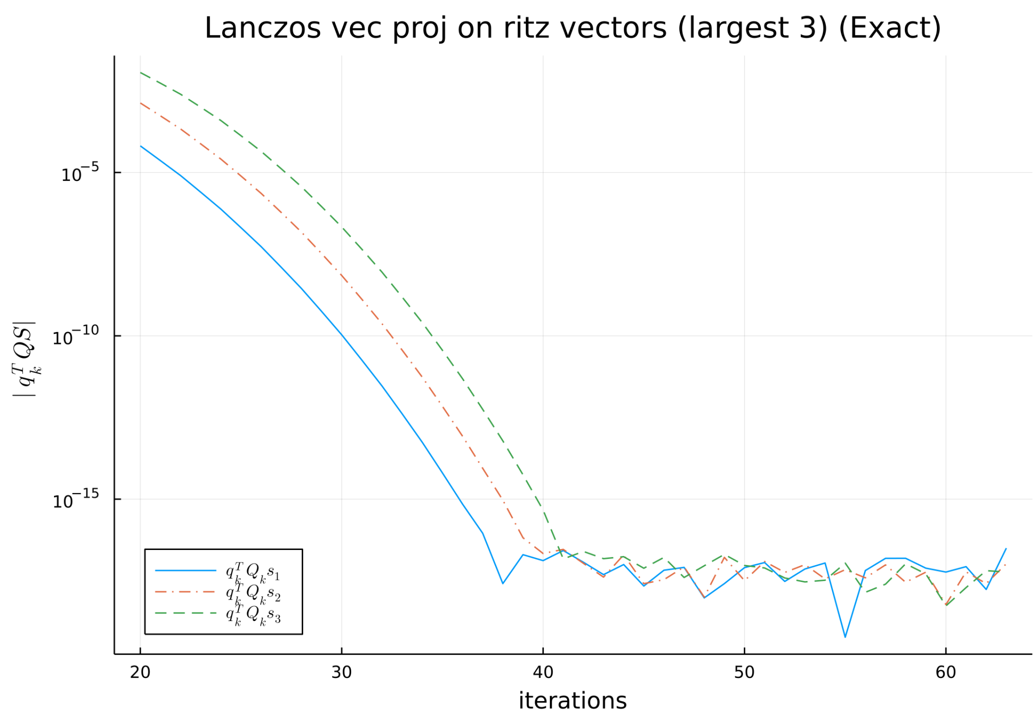


Figure 5.4.4: prHjection of the exact Lanczos vector q_k onto 3 of the largest Ritz vectors: $Q_k s_i^{(k)}$, for $i = k, k-1, k-2$

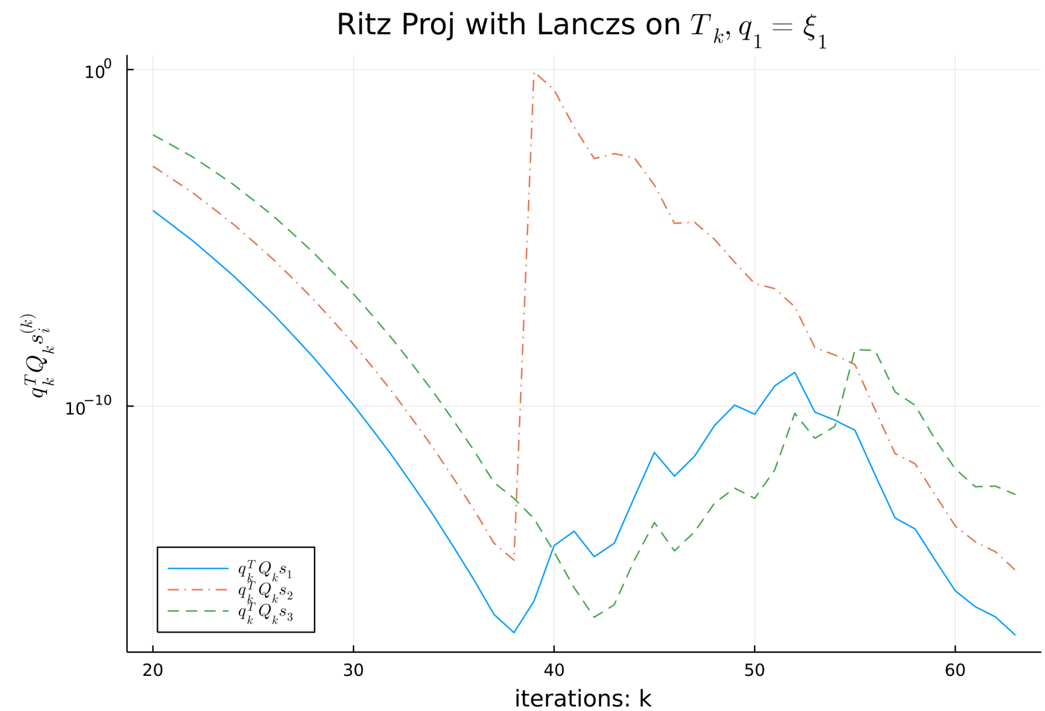


Figure 5.4.5: The projection of most recent Lanczos vector q_k onto the first 3 Ritzvectors: $Q_k s_i^{(k)}$ for $1 \le i \le 3$, it's performed on a Tridiagonal matrix T_k generated by finite precision lanczos with $q_1 = \xi_1$.

Remark 5.4.2 (Convergence Bounds on Ritzvalues)**.** The above presentation for the convergence of a Ritz Value and Ritz vector is an oversimplification. What happened is complicated. The characteristic polynomial of T_k minimizes under some weighted measured, hence the ritz values tends to approximate the eigenvalues of A, but this characteristic alone cannot dictate the way certain Ritzvalues converges.

It's not always the case that $\theta_1^{(j)}$ for example, it's the best approximation for λ_1 of A, and it's especially true when iterations j is relative small compare to n. The theoretical importance is to find an interval of how far are the λ_i from $\theta_{i'}^{(k)}$, where λ_i denotes the actual eigenvalue in A where the Ritz value $\theta_{i'}^{(k)}$ is trying to approximate. The bound for the Ritz interval was refined by Y. Saad back in 1980[9] and first discovered by Shmuel Kaniel back in 1966[13].

5.4.3 Greenbaum's Tiny Interval Experiments

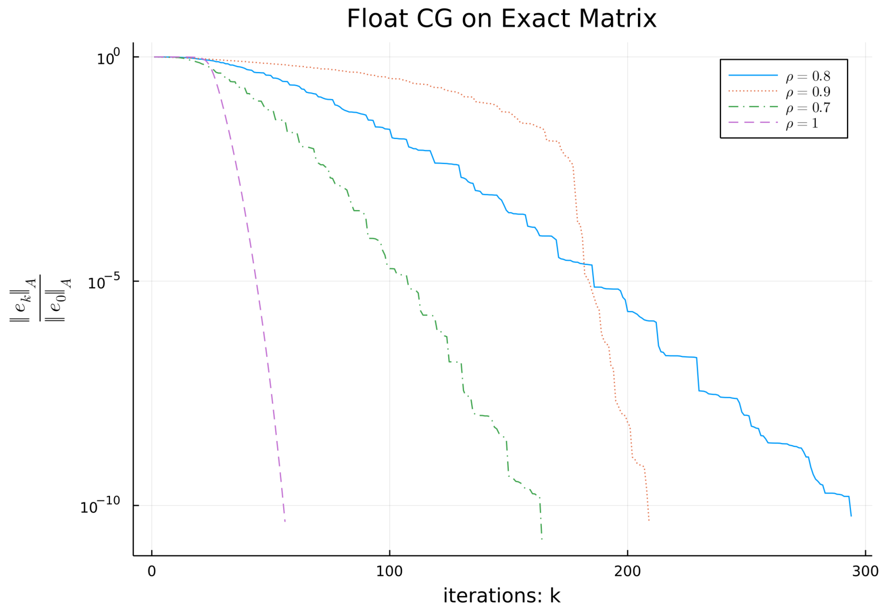
A smarter way of looking at the phenomenon of ghost eigenvalues (figure 5.4.1 left) is to take advantage of the clustering of the ghost eigenvalues and think of them as the eigenvalues of a potentially a larger matrix, denoted as $\tilde{A}$ whose eigenvalues are clustered around the eigenvalues of A within a tiny interval. The idea is if we perform exact Lanczos on A, then we get similar results for applying floating-point Lanczos on $\tilde{A}$. Simply put, due to the effect of round of errors, the floating-point Lanczos iterations can't see the spectrum of A clearly and instead, it sees $\tilde{A}$ whose eigenvalues are smeared out version of A, and there are many of them clustered around. More specifically, assuming A has eigenvalues: $\lambda_1, \cdots, \lambda_n$, the eigenvalues of $\tilde{A}$ lies in:

$$\bigcup_{n=1}^{n}[\lambda_i - \delta, \lambda_i + \delta] \tag{5.4.6}$$

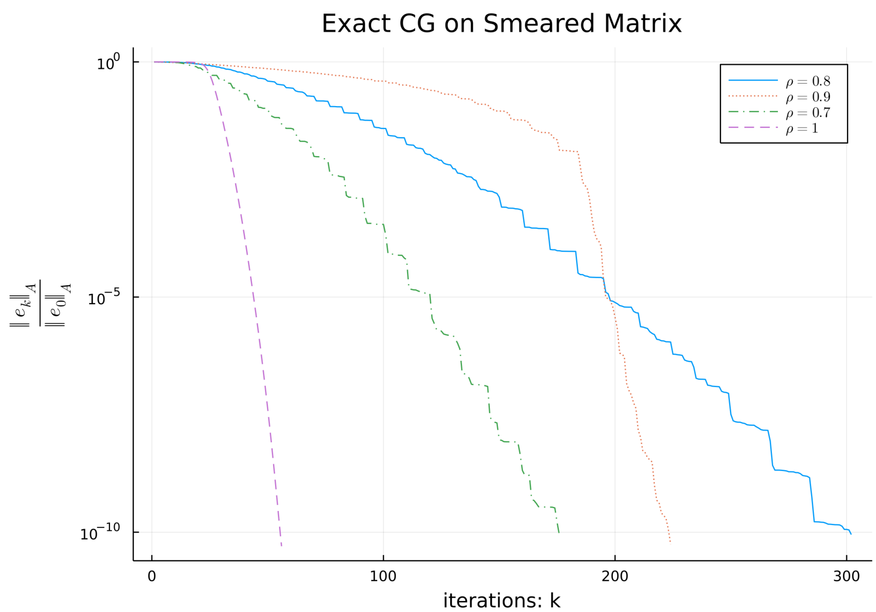
As a result, running an exact Lanczos/CG on $\tilde{A}$ produces similar convergence compared to the floating-point version of the algorithm. Experiments where conducted by A. Greenbaum and Z. Strakos in 1992[4]. Here, we reproduce the experiments on CG and check on the convergence rate of the algorithm.

To reproduce the effects, The same matrix paramaterized by ρ back in matrix 5.2.1. The tiny intervals are set by me via trials and errors. For the experiments, we set $\delta = 2\text{e-}5\|A\|\epsilon$ where ϵ is the machine epsilon for Float64; and 100 equally spaced eigenvalues on the spectrum for the matrix $\tilde{A}$ are clustered inside of the tiny intervals. The right hand side vector $b, \tilde{b}$ are chosen to be a vector of all ones. The matrix A is chosen to be 64×64, and hence $\tilde{A}$ is 6400×6400.

Both exact and float CG are run and terminates onece $\frac{\|e_k\|_A}{\|e_0\|_A}$ is less than 10^{-10}. By exact CG I mean CG with full reorthogonalizations.



(a) Applying CG without re-orthogonalizations on $Ax = b$, the original matrix without the tiny intervals.



(b) Applying CG with full re-orthogonalizations on $\tilde{A}x = \tilde{b}$. $\tilde{A}$ has tiny intervals.

Figure 5.4.6

The results of the experiments are plotted out in fig 5.4.6. For a tolerance of 10^{-10} on the relative energy norm of the error, we reproduced the behaviors of the CG without re-orthogonalizations using the tiny intervals ideas. In fig 5.4.6a is the CG without any re-orthogonalizations applied on $Ax = b$ for different values of ρ, and in fig 5.4.6b, it's the convergence of the CG with full re-orthogonalizations on the system $\tilde{A}x = \tilde{b}$.

Remark 5.4.3 (Optimal in Another Measure, and Open Questions)**.** The idea of tiny intervals allows for a good predictions for the behaviors of CG and Lanczos when finite precision arithemtic is involved. The underlying mechanism was shown by Greenbaum in 1989[2]. The Lanczos iterations generates tridiagonal matrix whose characteristic polynomial is orthogonal under a discrete measure at eigenvalues of A weighted by the vector $(U^H q_1)^2$, (appendix item **MISSING PROOFS**), however under finite arithmetic, they are no longer orthogonal under the original measure but instead, they are orthogonal under a new measured for some tiny intervals around the eigenvalues of A.

It's hypothesized that the tiny intervals are fixed and wrt to the number of iterations, however no current bounds are tight enough to show that's true.

5.5 Another Paige's Theorem

In this section, we prove another analysis of the Lanczos Iterations from Dammel's textbook(**Citation Needed**) where he introduced a proof of another Paige's theorem that shows exactly how to measure the loss of orthogonality of the Lanczos vectors against Ritz vectors. Here, we wish to follow the same proof with more details and thoroughness.

The theorem highlights two important facts. The first is that the loss of orthogonality of Lanczos Vector and Ghost eigenvalues appears at the same time and they are systematic. The second is that the projection of the Lanczos vector onto the converged Ritz vectors can be numerically attained, which tells us how much lost of orthogonality is occurring and which direction we need to re-orthogonalize so that the Lanczos vectors retains orthogonality. Here is the statement of the theorem:

Theorem 5 (Another Paige's Theorem)**.**

$$(y_i^{(k)})^T q_{k+1} = \frac{\mathcal{O}(\epsilon\|A\|)}{\beta_k (v_i)_k} \tag{5.5.1}$$

$$y_i^{(k)} := Q_k v_i \tag{5.5.2}$$

And we assume the following quantities:

$$\begin{aligned}
T_k &:: \text{Tridiagonal at step k of Lanczos} && (5.5.3)\\
Q_k &:: \text{Orthogonal matrix at step k of Lanczos} && (5.5.4)\\
V_k = [v_1\ v_2\ \cdots\ v_k] &:: \text{Eigen Matrix for } T_k && (5.5.5)\\
\theta_i &:: \text{the eigenvalues for } v_i,\ \text{Ritz Value} && (5.5.6)\\
\Lambda_k &:: \text{Eigenvalues Matrix for } T_k && (5.5.7)\\
F &:: \text{The floats error matrix from Lanczos Factorizations} && (5.5.8)\\
\epsilon &:: \text{The machine Epsilon} && (5.5.9)
\end{aligned}$$

5.6 Forward Error Analysis

Appendices

A Useful Lemmas

A.1 Relative Energy Norm and Relative 2-Norm Conversions

Lemma A.1.1 (Relative Energy Norm and Relative 2-Norm Conversions)**.** Let A be a Positive Symmetric Positive Definite Matrix, then it can be said that:

$$\frac{\|Ax\|}{\|Ay\|} \leq \kappa(A)\frac{\|x\|_A}{\|y\|_A}$$

Proof. From the definition of included 2-norm of matrices, assuming that λ_1 is the minimum eigenvalue of the matrix A, and λ_n the maximum, and the fact that matrix A has factorization $A^{1/2}A^{1/2}$:

$$\lambda_1\|x\| \leq \|Ax\| \leq \lambda_2\|x\| \tag{A.1.1}$$

$$\sqrt{\lambda_1}\|x\| \leq \|A^{1/2}x\| \leq \sqrt{\lambda_n}\|x\| \tag{A.1.2}$$

$$\implies \sqrt{\lambda_1} \leq \frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n} \tag{A.1.3}$$

Consider another vector y:

$$\sqrt{\lambda_1} \leq \frac{\|Ay\|}{\|A^{1/2}y\|} \leq \sqrt{\lambda_n} \tag{A.1.4}$$

Combining the two we have:

$$\sqrt{\lambda_1}\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n}\sqrt{\lambda_1} \tag{A.1.5}$$

$$\sqrt{\lambda_1}\sqrt{\lambda_n} \geq \sqrt{\lambda_n}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{A.1.6}$$

$$\implies \sqrt{\lambda_1}\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{A.1.7}$$

$$\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\kappa(A)}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{A.1.8}$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)}\frac{\|A^{1/2}x\|}{\|A^{1/2}y\|} \tag{A.1.9}$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)}\frac{\|x\|_A}{\|y\|_A} \tag{A.1.10}$$

□

B Theorems, Propositions, Proofs

B.1 Krylov Subspace Grade Invariant Theorem

Proposition B.1 (Krylov Subspace Grade Invariant Theorem)**.** Once the subspace becomes linearly dependent, the subspace becomes invariant.

Proof.

$$K_k = \begin{bmatrix} b & AB & \cdots & A^{k-1}b \end{bmatrix} \tag{B.1.1}$$

$$K_k \text{ Lin Dep} \implies A^{k-1}b = K_{k-1}c_k \tag{B.1.2}$$

$$\implies AK_k = K_k \underbrace{\begin{bmatrix} e_2 & \cdots & e_k & c_k \end{bmatrix}}_{:=C_k} \tag{B.1.3}$$

$$\implies A^2K_k = AK_kC_k = K_kC_k^2 \tag{B.1.4}$$

A^2K_k will span the same space as the range of the matrix K_k. □

B.2 Cauchy Interlace Theorem for Tridiagonal Symmetric Matrices

Theorem 6 (Cauchy Interlace Theorem for Tridiagonal Symmetric Matrices)**.** Let T_k be a $k \times k$ symmetric tridiagonal matrix, then its top left upper submatrix: $T_{k-1} = (T_k)_{:k-1,:k-1}$ has eigenvalues interlaced between the eigenvalues of T_k. Denotes all k eigenvalues of T_k as $\theta_i^{(k)}$, and all $k-1$ eigenvalues of T_{k-1} as $\theta_i^{(k-1)}$. Order them so that: $\theta_1^{(k-1)} \le \cdots, \le \theta_i^{(k-1)}$, similarly: $\theta_1^{(k)} \le \cdots \le \theta_i^{(k)}$, then:

$$\theta_k^{(k)} \ge \theta_{k-1}^{(k-1)} \tag{B.2.1}$$

$$\theta_1^{(k)} \le \theta_1^{(k-1)} \tag{B.2.2}$$

$$\theta_{i-1}^{(k-1)} \le \theta_i^{(k)} \le \theta_i^{(k-1)} \tag{B.2.3}$$

Theorem taken from first chapter of Greenbuam's book[5] and it's adapted for symmetric tridiagonal matrix.

Proposition B.2 (Orthogonal Polynomials and Lanczos)**.** The Lanczos algorithm generates orthogonaly polynomial under a discrete weighted measure under the eigenvalues of matrix A, which is also the Lanczos vector q_k represented under the Krylov subspace.

Proof. Let $V_k^H AV_k = T_k$ be the tridiagonalization resulted from Lanczos algorithm and we assume exact arithematic, using the fact that each lanczos vector q_k is an element from the Krylov subspace, we can represent it as a matrix polynomial multiplied by q_k:

$$v_{m+1} = q_m(A)v_1 \in \mathcal{K}(A|v_1) \quad \forall\, m \le k-1 \tag{B.2.4}$$

$$\exists\, q_i \in \mathcal{P}_{i-1} : v_i = q_{i-1}(A)v_1 \tag{B.2.5}$$

Since the exact arithematic generates orthogonal lanczos vectors, let's consider v_i, v_j being represented by polynomial ϕ, φ, then we have:

$$\langle v_i, v_j\rangle = 0 \tag{B.2.6}$$
$$\langle \phi(A)v_1, \varphi(A)v_1\rangle = 0 \tag{B.2.7}$$
$$\langle U\phi(\Lambda)U^H v_1, U\varphi(\Lambda)U^H v_1\rangle = 0 \tag{B.2.8}$$
$$\text{Let: } f_1 = U^H v_1 \text{ Then:} \tag{B.2.9}$$
$$\langle U\phi(\Lambda)f_1, U\varphi(\Lambda)f_1\rangle = 0 \tag{B.2.10}$$
$$\langle \phi(\Lambda)f_1, \varphi(\Lambda)f_1\rangle = 0 \tag{B.2.11}$$
$$\sum_{i=1}^{n}(f_1)_i^2\phi(\lambda_i)\varphi(\lambda_i) = 0 \tag{B.2.12}$$

Therefore the polynomials ϕ, φ are orthogonal under the discrete measure over eigenvalues of A weighted by the vector $(f_1)^2$. □

B.3 Recursion of the Symmetric Tridiagonal Matrix Determinant

Proposition B.3 (Recursion of the Symmetric Tridiagonal Matrix Determinant)**.** Let T_K be a symmetric tridiagonal matrix $\in \mathbb{R}^{k\times k}$ with α on its diagonal and β on its subdiagona. Recursively, we define T_{k-i} to be the top left sub matrix of T_k, with size $(k-1)\times(k-1)$. Using $|\cdot|$ to denote the determinant of a matrix, we have the recurrence relation:

$$|T_k| = \alpha_k|T_{k-1}| - \beta_{k-1}^2|T_{k-2}|$$

Proof. Using the notation of $e_k^{(m)}$ to denote the k^{th} standard basis vector in $\mathbb{R}^m$, consider the Block Matrix:

$$T_k = \begin{bmatrix} T_{k-2} & \beta_{k-2}e_{k-1}^{(k-2)} & \\ \beta_{k-2}e_{k-2}^{(k-2)T} & \alpha_{k-1} & \beta_{k-1}e_{k-1}^{(k-1)} \\ & \beta_{k-1}e_{k-1}^{(k-1)T} & \alpha_k \end{bmatrix} \tag{B.3.1}$$

Now, we use the Laplace Expansion to figure out the determinant we are expanding the Laplace Expansion on the last row of T_k

$$|T_k| = (-1)^{k+(k-1)}\beta_{k-1}\left|\begin{bmatrix} T_{k-2} & \\ \beta_{k-2}e_{k-2}^{(k-2)T} & \beta_{k-1}e_{k-1}^{(k-1)}\end{bmatrix}\right| \; +(-1)^{2k}\alpha_k\left|\underbrace{\begin{bmatrix} T_{k-2} & \beta_{k-2}e_{k-1}^{(k-2)} \\ \beta_{k-2}e_{k-2}^{(k-2)T} & \alpha_{k-1}\end{bmatrix}}_{=T_{k-1}}\right| \tag{B.3.2}$$

$$(-1)^{2k-2}\beta_{k-1}|T_{k-2}| = \left|\begin{bmatrix} T_{k-2} & \\ \beta_{k-1}e_{k-2}^{(k-2)T} & \beta_{k-2}e_{k-1}^{(k-1)}\end{bmatrix}\right| \tag{B.3.3}$$

Substituting the last equation back to the first equation of the first term.

$$|T_k| = (-1)^{2k-2+2k-1}\beta_{k-1}^2|T_{k-2}| + \alpha_k|T_{k-1}| \tag{B.3.4}$$
$$= -\beta_{k-1}^2|T_{k-2}| + \alpha_k|T_{k-1}| \tag{B.3.5}$$

□

B.4 Recurrence of the Characteristic Polynomial of a Symmetric Tridiagonal Matrix

Theorem 7 (Recurrence of the Characteristic Polynomial of a Symmetric Tridiagonal Matrix)**.** The characteristic polynomial of a symmetric tridiagonal matrix satisfies the recurrences:

$$\begin{cases} p_k(x) = -\beta_{k-1}^2 p_{k-2}(x) + (\alpha_k - x)p_{k-1}(x) \\ p_0(x) = 1 \\ p_{-1}(x) = 0 \end{cases}$$

Where, $p_k(x) = |T_k - xI|$, and $p_{k-1}(x) := |(T_k)_{1:k-1,1:k-1}|$, and $p_{k-2} = |(T_k)_{1:k-2,1:k-2}|$.

Proof. Using Proposition B.3, but replacing α_k to be $\alpha_k - x$ due to the shifting introduced by $T_k - xI$, then:

$$|T_k - Ix| = (\alpha_k - x)|T_{k-1} - xI| - \beta_{k-1}^2 |T_{k-2} - Ix| \tag{B.4.1}$$

$$\implies p_k(x) = -\beta_{k-1}^2 p_{k-2}(x) + (\alpha_k - x)p_{k-1}(x) \tag{B.4.2}$$

The recurrence is direct from the recurrences of the determinant of symmetric tridiagonal matrix, and a monic polynomial with degree zero is just 1, therefore the base case matches up as well. □

B.5 Tridiagonal Characteristic Polynomials is Scaled Lanczos Orthgonal Polynomials

Proposition B.4 (Tridiagonal Characteristic Polynomials is Scaled Lanczos Orthgonal Polynomials)**.** The polynomial from Theorem 7 is a scalar multiple of the polynomial that represents the lanczos vector q_k under the Krylov Subspace.

B.6 Irreducible Symmetric Tridiagonal Matrix

Proposition B.5. The tridiagonal matrix T_k generated by the Lanczos algoithm cannot have repeated eigenvalues. It's what referred to as a irreducible symmetric tridiagonal matrix in some literature.

Proof. Let T_k be symmetric tridiaognal $k \times k$ matrix, its all sub/super diagonals are nonzeros. Consider the submatrix $(T_k - \lambda I)_{2:k,1:k-1}$ with the first row and last column removed. Regardless of λ, $(T_k - \lambda I)_{2:k,1:k-1}$ whose diagonals are the sub diagonals of T_k, which is all non-zero. Hence $\det((T_k - \lambda I)_{2:k,1:k-1}) \neq 0 \ \forall \lambda$.

The determinant of $(T_k - \lambda I)_{2:k,1:k-1}$ is always nonzero implies that the full matrix $T_k - \lambda I$ has a rank of at least $k-1$ for all λ; which implies that all roots of $\det(T_k - \lambda I)$ has algebraic multiplicity of strictly 1.

Since the matrix is symmetric, it must be diagonalizable. For contradiction assuming that it has repeated eigenvalues and still diagonalizable, it must have repeated roots, which is a contradiction. Therefore all its eigenvalues are unique. □

B.7 From CG to Lanczos: The Proof

We will break the proof into several parts. Firstly we address the base case, and then we address the inductive case to establish the parameters between the Tridiagonal matrix and a_k, b_k, finally we resolve the sign problem between the Lanczos vectors and the residual vectors.

B.7.1 The Base Case

Right from the start of the CG iteration we have:

$$r_0 = p_0 \tag{B.7.1}$$
$$r_1 = r_0 - a_0 A r_0 \tag{B.7.2}$$
$$Ar_0 = a_0^{-1}(r_0 - r_1) \tag{B.7.3}$$
$$Ar_0 = \frac{\|r_0\|_A^2}{\|r_0\|^2}(r_0 - r_1) \tag{B.7.4}$$

Consider substituting $r_0 = \|r_0\| q_1, r_1 = -\|r_1\| q_2$, then:

$$\begin{aligned} A\|r_0\|q_1 &= \frac{\|r_0\|_A^2}{\|r_0\|^2}\left(\|r_0\|q_1 + \|r_1\|q_2\right) && \text{(B.7.5)}\\ &= \frac{\|r_0\|_A^2}{\|r_0\|^2}\|r_0\|q_1 + \frac{\|r_1\|}{\|r_0\|}q_2 && \text{(B.7.6)}\end{aligned}$$

And from this relation, using the Lanczos recurrence theorem would imply that $\alpha_1 = a_0^{-1}$; $\beta_1 = \frac{\sqrt{b_0}}{\alpha_0}$. So far so good, we have shown that there is an equivalence between the Lanczos and the CG for the first iterations of the CG algorithm.

B.7.2 The Inductive Case

Lemma B.7.1. Inductively we wish to show the relation that:

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le n-2 \end{cases} \tag{B.7.7}$$

Proof. We start by considering:

$$\begin{aligned} r_j &= r_{j-1} - a_{j-1}Ap_{j-1} && \text{(B.7.8)}\\ &= r_{j-1} - a_{j-1}A(r_{j-1} + b_{j-2}p_{j-1}) && \text{(B.7.9)}\\ &= r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-1} && \text{(B.7.10)}\end{aligned}$$

We make use of the recurrence asserted by the CG algorithm, giving us:

$$r_{j-1} = r_{j-1} - a_{j-2}Ap_{j-1} \tag{B.7.11}$$
$$r_{j-1} - r_{j-1} = a_{j-2}Ap_{j-1} \tag{B.7.12}$$
$$Ap_{j-1} = a_{j-2}^{-1}\left(r_{j-2} - r_{j-1}\right) \tag{B.7.13}$$

Here, we can substitute the results for the term Ap_{j-1}, and then we can express the recurrence of residual purely in terms of residual. Consider:

$$r_j = r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-2} \tag{B.7.14}$$

$$= r_{j-1} - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}(r_{j-2} - r_{j-1}) \tag{B.7.15}$$

$$= \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} \tag{B.7.16}$$

$$a_{j-1}Ar_{j-1} = \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} \tag{B.7.17}$$

$$Ar_{j-1} = \left(\frac{1}{a_{j-1}} + \frac{b_{j-2}}{a_{j-2}}\right)r_{j-1} + \frac{r_j}{a_{j-1}} - \frac{b_{j-2}}{a_{j-2}}r_{j-2} \tag{B.7.18}$$

Finally, we increment the index j by one for notational convenience, and therefore we establish the following relations between the residuals of the conjugate gradient algorithm:

$$Ar_j = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)r_j + \frac{r_{j+1}}{a_j} - \frac{b_{j-1}}{a_{j-1}}r_{j-1} \tag{B.7.19}$$

Reader, please observe that this is somewhat similar to the recurrence relations between the Lanczos vectors, however it's failing to match the sign, at the same time, it's not quite matching the form of the recurrence of β_k from the Lanczos algorithm. To match it, we need the coefficients of r_{j-1} and r_{j+1} to be in the same form, parameterized by the same iterations parameter: j. To do that, consider the doing this:

$$q_{j+1} := \frac{r_j}{\|r_j\|} \tag{B.7.20}$$

$$q_j := -\frac{r_{j-1}}{\|r_{j-1}\|} \quad \text{Note: This is Negative} \tag{B.7.21}$$

$$q_{j+2} := \frac{r_{j+1}}{\|r_{j+1}\|} \tag{B.7.22}$$

$$\implies A\|r_j\|q_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)\|r_j\|q_{j+1} + \frac{\|r_{j+1}\|q_{j+2}}{a_j} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}}q_j \tag{B.7.23}$$

$$Aq_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)q_{j+1} + \frac{\|r_{j+1}\|}{a_j\|r_j\|}q_{j+2} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|}q_j \tag{B.7.24}$$

Recall that parameters from Conjugate Gradient, $\sqrt{b}_j = \|r_{j+1}\|/\|r_j\|$, and $a_j = \frac{\|r_j\|^2}{\|p_j\|_A^2}$, and we can use the substitution to match the coefficients for q_{j+2} and q_j, giving us:

$$\frac{\|r_{j+1}\|}{a_j\|r_j\|} = \frac{1}{a_j}\sqrt{b_j} \tag{B.7.25}$$

$$\frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|} = \frac{b_{j-1}}{a_{j-1}}\frac{1}{\sqrt{b_{j-1}}} = \frac{\sqrt{b_{j-1}}}{a_{j-1}} \tag{B.7.26}$$

$$\implies \begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le n-2 \end{cases} \tag{B.7.27}$$

Take notes that the form is now matched, but the expression for α_{j+1} has an extra b_{j-1}/a_{j-1}, to resolve that, we take the audacity to make b_0 so that it's consistent with the base case. $\square$

B.7.3 Fixing the Sign

We can't take the triumph yet; we need to take a more careful look into the sign between q_j the Lanczos Vector and its equivalence residual: r_{j-1} in CG. Here, I want to point out the fact that, there are potentially two substitutions possible for the above derivation for the inductive case and regardless of which one we use, it would still preserve the correctness for the proof. By which I mean the following substitutions would have both made it work:

$$\begin{cases} q_{j+1} := \pm\frac{r_j}{\|r_j\|} \\ q_j := \mp\frac{r_{j-1}}{\|r_{j-1}\|} \\ q_{j+2} := \pm\frac{r_{j+1}}{\|r_{j+1}\|} \end{cases} \tag{B.7.28}$$

Under the context, the operations $\pm, \mp$ are correlated, choose a sign for one, the other must be of opposite sign. In this case both substitutions work the same because multiplying the equation by -1 would give the same equality, and we can always multiply by another negative sign to get it back. The key here is that, the sign going from q_j to the next q_{j-1} will have to alternate. To find out precisely which one it is, we consider the base case for the Lanczos Vectors and Residuals:

$$q_1 = \hat{r}_0 \tag{B.7.29}$$
$$q_2 = -\hat{r}_1 \tag{B.7.30}$$
$$\vdots \tag{B.7.31}$$
$$q_j = (-1)^{j+1}\hat{r}_{j+1} \tag{B.7.32}$$

B.8 Derive CG using Lanczos: Proof

We made use of the fact that the matrix U_k is unit upper bidiagonal. Next, we seek for the recurrences of the parameters u_k, l_k. Let's consider the recurrence using the block structure of the matrices:

$$T_k = L_k U_k \tag{B.8.1}$$
$$T_{k+1} = \begin{bmatrix} T_k & \beta_k\xi_k \\ \beta_k\xi_k^T & \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} L_k & \mathbf{0} \\ l_k\xi_k^{-1} & 1 \end{bmatrix}\begin{bmatrix} U_k & \eta_k\xi_k \\ \mathbf{0} & u_{k+1} \end{bmatrix} \tag{B.8.2}$$
$$= \begin{bmatrix} L_k U_k & \eta_k L_k \xi_k \\ l_k\xi_k^T U_k & \eta_k l_k \xi_k^T \xi_k + u_{k+1}\alpha_k \end{bmatrix} \tag{B.8.3}$$
$$= \begin{bmatrix} T_k & \eta_k (L_k)_{:,k} \\ l_k (U_k)_{k,:} & \eta_k l_k + u_{k+1} \end{bmatrix} \tag{B.8.4}$$
$$= \begin{bmatrix} T_k & \eta_k \\ l_k u_k & \eta_k l_k + u_{k+1} \end{bmatrix} \tag{B.8.5}$$

Note that I changed the upper diagonal of matrix U at the top to be η_k instead of β_k so then we have a chance to convince ourselves that β_k for the upper diagonal of T_k are indeed the same as the η_k for the upper diagonal of matrix U_k. From the results above, $\eta_k = \beta_k$ as expected, and $l_k = \beta_k/u_k$, $u_{k+1} = \alpha_{k+1} - \beta_k l_k$, and hence, to sum up the recurrence relation we have:

$$\begin{cases} u_{k+1} & = \alpha_{k+1} - \beta_k^2/u_k \\ l_k & = \beta_k/u_k \end{cases} \tag{B.8.6}$$

The base case is $u_1 = \alpha_1$. The recurrence of the parameter u_k is immediately useful for figuring out the recurrence for x_k, And to figure out the recurrence relations of $(L_k^{-1})_{k,1}$, we consider the following fact:

$$L_k^{-1}L_k = I \tag{B.8.7}$$

$$\begin{bmatrix} L_k^{-1} & \mathbf{0} \\ s_k^T & d_{k+1} \end{bmatrix} \begin{bmatrix} L_k & \mathbf{0} \\ l_k\xi_k^T & 1 \end{bmatrix} = I \tag{B.8.8}$$

$$\begin{bmatrix} I & \mathbf{0} \\ s_k^T L_k + d_{k+1}l_k\xi_k^T & d_{k+1} \end{bmatrix} = I \tag{B.8.9}$$

It equals the identity matrix therefore $d_{k+1} = 1$, and it has to be that the lower diagonal sub vector in the results has to be zero. For the bi-lower unit diagonal matrix L_k, we cannot predict the structure, most of the time it's likely to be dense and unit lower triangular. We are interested in look for the first element of the vector s_k^T, the equality will assert:

$$s^T L_k + d_{k+1}l_k\xi_k^T = \mathbf{0} \tag{B.8.10}$$

$$L_k^T s_k + d_{k+1}l_k\xi_k = \mathbf{0} \tag{B.8.11}$$

$$s_k + L^{-T}d_{k+1}l_k\xi_k = \mathbf{0} \tag{B.8.12}$$

$$(s_k)_1 + d_{k+1}l_k((L_k^{-1})\xi_k)_1 = 0 \tag{B.8.13}$$

$$(s_k)_1 + d_{k+1}l_k(L_k^{-1})_{k,1} = 0 \tag{B.8.14}$$

$$\implies (s_k)_1 = -l_k(L_k^{-1})_{k,1} \tag{B.8.15}$$

$$(s_k)_1 = (L_{k+1}^{-1})_{k+1,1} \quad \text{by definition} \tag{B.8.16}$$

$$\implies (L_{k+1}^{-1})_{k+1,1} = -l_k(L_k^{-1})_{k,1} \tag{B.8.17}$$

To assert the fact that $L_k^{-1}L_k$ is identity, we carefully consider the last row excluding the bottom right element vector: $s^T L_k + d_{k+1}l_k\xi_k^T$ which will have to be a zero vector. From the first line to the second line, we took the transpose of the vector. From the second to the third line we multiplied both sides by the inverse of L_k^T. And from the third to the fourth line, we took out only the first element in the vector. Observe that the first element of the vector $L_k^{-1}\xi_k$ is just $(L_k^{-1})_{k,1}$.

Therefore, the recurrence for the step size into the direction of the conjugate vector requires us to use the newest element l_k from L_{k+1} and the previous step size in the direction of the conjugate vector p_k. The short recurrence allows us to build up another algorithm that is just as efficient as CG algorithm but running Lanczos Algorithm under the hood.

C Algorithms

Definition 7 (Lanczos Iterations Variants)**.**

$$
\begin{aligned}
&\text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \\
&\alpha_1 := \langle q_1, Aq_1 \rangle \\
&\beta_0 := 0 \\
&\text{Memorize : } Aq_1 \\
&\text{For } j = 1, 2, \cdots \\
&\quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1} q_{j-1} \\
&\quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\
&\quad \beta_j = \|\tilde{q}_{j+1}\| \\
&\quad q_{j+1} := \tilde{q}_{j+1} / \beta_j \\
&\quad \alpha_{j+1} := \langle q_{j+1}, Aq_{j+1} \rangle \\
&\quad \text{Memorize: } Aq_{j+1}
\end{aligned}
\tag{C.0.1}
$$

References

[1] M.G. Cox et al. *Reliable Numerical Computation*. Oxford science publications. Clarendon Press, 1990. ISBN: 9780198535645. URL: https://books.google.com/books?id=CBnvAAAAMAAJ.

[2] A. Greenbaum. "Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences". In: *Linear Algebra and its Applications* 113 (1989), pp. 7–63. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(89)90285-1. URL: https://www.sciencedirect.com/science/article/pii/0024379589902851.

[3] A. Greenbaum, V. Druskin, and L. A. Knizhnerman. "On solving indefinite symmetric linear systems by means of the Lanczos method". In: *Zh. Vychisl. Mat. Mat. Fiz.* 39 (1999), pp. 371–377.

[4] A. Greenbaum and Z. Strakos. "Predicting the Behavior of Finite Precision Lanczos and Conjugate Gradient Computations". In: *SIAM Journal on Matrix Analysis and Applications* 13.1 (1992), pp. 121–137. DOI: 10.1137/0613011. eprint: https://doi.org/10.1137/0613011. URL: https://doi.org/10.1137/0613011.

[5] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, 1997.

[6] Magnus R. Hestenes and Eduard Stiefel. "Methods of Conjugate Gradient for Solving Linear Systems". In: *Journal of Research of the National Bureau of Standards* 49.6 (1952), pp. 409–436.

[7] C. C. Paige and M. A. Saunders. "Solutions of Sparse Indefinite System of Linear Equations". In: *SIAM.J Numerical Analysis* 12.4 (1975), pp. 617–629.

[8] C.C. Paige. "Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem". In: *Linear Algebra and its Applications* 34 (1980), pp. 235–258. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(80)90167-6. URL: https://www.sciencedirect.com/science/article/pii/0024379580901676.

[9] Y. Saad. "On the Rates of Convergence of the Lanczos and the Block-Lanczos Methods". In: *SIAM Journal on Numerical Analysis* 17.5 (1980), pp. 687–706. DOI: 10.1137/0717059. eprint: https://doi.org/10.1137/0717059. URL: https://doi.org/10.1137/0717059.

[10] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[11] Yousef Saad. "Krylov Subspace Methods for Solving Large Unsymmetric Linear Systems". In: *Mathematics of Computations* 37.155 (1981), pp. 105–126.

[12] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Aug. 1994.

[13] Kaniel Shmuel. "estimates for Some Computational Techniques in Linear Algebra". In: *Math. of Comp.* 95 (1966), pp. 369–378.

[14] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. 3rd ed. Philadelphia: Society for Industrial and Applied Mathematics, 1997.