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 Name: Department of Applied Mathematics
University Name: University of Washington

Abstract

In this paper we derive and analyzie the Conjugate Gradient algorithm and Lanczos Iterations, look into their relations and analyze them in details. We start with deriving the method of Conjugate Gradient using projectors and its properties using the foundations established. Then we clarify the relations between the Conjugaet Gradient and Lanczos Algorithm and using the relations to reveal inspirations in the area of symmetric indefinite solvers and convergence rate anaysis. We then analyze algorithms behaviors under floating points arithematic, specifically how the floating point arithemtic slows down the convergence of CG and how it's related to the lost of orthogonality of Lanczos vectors, and then we present some theorems regarding which directions the Lanczos vectors are losing their orthogonality together with illustrative numerical experiments.

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 i th 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. $\forall\ 0 \leq j \leq k$: 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 $\sum_{j=0}^{k} w_j A^j$.

2 Introduction

The Conjugate Gradient method is an iterative method used for solving linear systems which date 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 Conjugate Gradient without pre-conditioning by deriving it and analyzing it along with Lanczos Iterations, a tightly related algorithm for symmetric eigenproblems. Finally, we use their connections to analyze its behaviour under floating-point arithmetic. The thesis will require some background in numerical linear algebra for the best understanding.

In the first section, we introduce the projectors and Krylov subspace as important mathematical objects for the subspace projections method, the frameworks for subspace projection methods, and the Lanczos Iteration as a symmetric case of the Arnoldi Iterations. At the end of the section, we proceed and derive the conjugate gradient method using only these ideas and concepts. In the second section, we analyze the behaviors of Conjugate gradient and Lanczos Iterations, including the terminations conditions for both algorithm, their convergence rate, and the equivalence between them. The goal is to show how they can be the same and have similar properties and how the connections between them can spark other methods for solving a symmetric indefinite linear system. And in the final parts of the second section, we derive the convergence bound for the conjugate gradient algorithm.

In the third section, we demonstrate the behaviors of Lanczos Iterations and Conjugate Gradient numerically and then show that the algorithm is backward stable. We show how floating points affect the Lanczos algorithm more rigor and how it may be fixed and mitigated, consequently improving the conjugate gradient algorithm.

Acknowledgement

3 Foundations

In this section, we go over the foundations of Conjugate Gradient and the Lanczos Algorithm. We introduce the important ideas at the beginning and then we proceed to prove the conjugate gradient algorithm via the method conjugate directions, and then we state the Lanczos Iterations as a symmetric case of the Arnoldi Iterations.

3.1 The Basics

In this subsection we go over some basics concepts and mathematical entities that are important to the 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, for every element in the subspace, it's a matrix polynomial, we write it as $p_{k-1}(A|w)$, a $k-1$ degree matrix polynomial where A is the matrix and w is a vector denoting the coefficients.

Definition 2 (The Grade of Krylov Subspace)**.** The grade of the krylov subspace is the the smallest $\mathcal{K}_k(A|v)$ such that the vectors in the span is linear dependent, denoted as grade$(A|v)$.

The word grade of a krylov Subspace is usedn Y.Saad's work (**CITATION NEEDED**) in reference to Krylov Subspace. And once the grade is reached, the Krylov Subspace becomes an invariant subspace to the matrix A. For a proof, refers to Krylov Subspace Grade Invariant Theorem in the appendix. This concept is useful for determining the termination conditions for Lanczos iterations and Conjugate Gradient.

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 k be the grade, then it's the minimum number such that the matrix polynomial of A multiplied by u equals to the zero vector, and it's nontrivial.

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

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

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

When non trivial solution exists for some $w_j \neq 0$, it will be the case that whenever $(V^{-1}u)_i$ is not zero, then the a polynomial will have to interpolate λ_i. Therefore, minimum k is the number of unique λ_i such that $(V^{-1}u)_i \neq 0$ □

3.1.2 Projectors

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

There are 2 types of projectors, oblique and orthogonal projectors. A projector is orthogonal projector when it's Hermitian, it's oblique when it's not Harmitian.

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

$$\text{ran}(P) = \text{null}(I - P) \tag{3.1.4}$$
$$\text{ran}(I - P) = \text{null}(P) \tag{3.1.5}$$

The proof is immediate from the definition. For more coverage of facts, refer to Trefethen's Book on Numerical Linear Algebra(**CITATION HERE**).

3.2 Subspace Projection Methods

Let $\mathcal{K}, \mathcal{L}$ be 2 subspaces. $\mathcal{K}$ can be viewed as a subspace where we choose our solutions for the x in the linear system $Ax = b$, and $\mathcal{L}$ is a subspace where we test our residual $r = b - Ax$. This is a description of this framework:

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

We look for a x in the affine linear subspace $x_0 + \mathcal{K}$ such that it's perpendicular to the subspace $\mathcal{L}$, or, equivalently, minimizing the projection onto the subspace $\mathcal{L}$.

The above conditions can be expressed using matrix.

$$\text{Let } V \in \mathbb{R}^{n\times m} \text{ be a aasis for: } \mathcal{K} \tag{3.2.2}$$
$$\text{Let } W \in \mathbb{R}^{n\times m} \text{ be a basis for: } \mathcal{L} \tag{3.2.3}$$

Substituting the matrices to the above set of conditions:

$$\tilde{x} = x_0 + Vy \tag{3.2.4}$$
$$\text{choose } x \text{ s.t: } b - A\tilde{x} \perp \text{ran}(W) \tag{3.2.5}$$
$$\implies W^T(b - Ax_0 - AVy) = \mathbf{0} \tag{3.2.6}$$
$$W^T r_0 - W^T AVy = \mathbf{0} \tag{3.2.7}$$
$$W^T AVy = W^T r_0 \tag{3.2.8}$$

3.2.1 Prototype Algorithm

And from here, we can define a simple prototype algorithm using this framworks.

$$\begin{aligned}
&\text{While not converging :}\\
&\quad \text{Increase Span for: } \mathcal{K}, \mathcal{L}\\
&\quad \text{Choose: } V, W \text{ for } \mathcal{K}, \mathcal{L}\\
&\quad y := (W^T AV)^{-1} W^T r_0\\
&\quad x := x + Vy\\
&\quad r := r_0 - AVy
\end{aligned} \tag{3.2.9}$$

Each time, we increase the span of the subspace $\mathcal{K}, \mathcal{L}$, which gives us more space to choose the solution x, and more space to reduce the residual vector r. This idea is incredibly flexible, and we will see in later part where it reduces to a more concrete algorithm. Finally, when $\mathcal{K} = \mathcal{L}$, this is referred to as a Petrov Galerkin's Conditions (**CHECK THIS**).

Remark 3.2.1 (Projector in the Prototype Algorithm)**.** One may proceed to find a projector in the above prototype algorithm.

$$y = (W^T Av)^{-1} W^T r_0 \tag{3.2.10}$$
$$AVy = AV(W^T AV)^{-1} W^T r_0 \tag{3.2.11}$$
$$x = x_0 + AVy \tag{3.2.12}$$
$$b - Ax = b - Ax_0 - AVy \tag{3.2.13}$$
$$r = r_0 - AV(W^T AV)^{-1} W^T r_0 \tag{3.2.14}$$

In here, $AV(W^T AV)^{-1} W^T$ is a projector, assuming the invertibility of $W^T AV$.

3.2.2 Energy Norm Minimization using Gradient

Other times, iterative method will choose to build up a subspace for each step with a subspace generator, and build up the solution on this expanding subspace, but with the additional objective of minimizing the residual under some norm. Assuming that the vector $x \in x_0 + \mathcal{K}$, and we want to minimize the residual under a norm induced by positive definite operator B. Let it be the case that the columns of matrix K span subspace $\mathcal{K}$ with $\dim(\mathcal{K}) = k$, then one may consider using gradient as a more direct approach instead of projector.

$$\min_{x \in x_0 + \mathcal{K}} \|b - Ax\|_B^2 \tag{3.2.15}$$
$$= \min_{w \in \mathbb{R}^k} \|b - A(x_0 + Kw)\|_B^2 \tag{3.2.16}$$
$$= \min_{w \in \mathbb{R}^k} \|r_0 - AKw\|_B^2 \tag{3.2.17}$$

We take the derivative of it and set the derivative to zero, skipping the proof that the derivative of $\nabla_x[\frac{1}{2}\|x\|_A^2] = Ax$ and just apply this formula for computing the gradient $\nabla_x[f(Ax)] = A^T \nabla[f(x)]$ where $f(x)$ is a mapping from $\mathbb{R}^n$ to $\mathbb{R}$.

$$\nabla_w \left[\|r_0 - AKx\|_B^2\right] = \mathbf{0} \tag{3.2.18}$$
$$(AK)^T B(r_0 - AKx) = \mathbf{0} \tag{3.2.19}$$
$$(AK)^T Br_0 - (AK)^T BAKx = \mathbf{0} \tag{3.2.20}$$
$$(AK)^T Br_0 = (AK)^T BAKx \tag{3.2.21}$$

The above formulation is powerful. I used gradient instead of projector for the simplicity of the argument. One can derive the same using orhtogonal projector to minimizes the 2 norm, but the math is bit more tedious. However, this minimization objective is minimizing the residual, which is fine for deriving subspace methods such as the GMRes, or the Minres and Orthomin, however, for the sake of the conjugate gradient, we have to consider the altenative. Let's this be a proposition that we proceed to prove.

Proposition 3.3 (Conditions for Minimum Error Under Energy Norm)**.** Here, we let matrix B be positive definite so it can induce a norm, we let K be a matrix whose columns forms a basis for $\mathcal{K}$, we let e_k denotes the error, given by: $A^{-1}b - x_k$, and we let r_k denotes the residual given as $b - Ax_k$.

$$\min_{x_k \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 \iff K^T B e_0 - K^T B K w = \mathbf{0} \tag{3.2.22}$$

Next, we proceed to prove it and explain its interpretations and importance:

Proof.

$$\min_{x \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 = \min_{x \in \mathbb{R}} \|A^{-1}b - x_0 - Kw\|_B^2 \tag{3.2.23}$$

$$= \min_{x \in \mathbb{R}^k} \|e_0 - Kw\|_B^2 \tag{3.2.24}$$

To attain the minimum of the norm, we take the derivative and set it to be zero, giving us:

$$\mathbf{0} = \nabla_w [\|e_0 - Kw\|_B^2] \tag{3.2.25}$$

$$= \nabla_w [e_0 - Kw]^T B (e_k - Kw) \tag{3.2.26}$$

$$= 2K^T B (e_0 - Kw) \tag{3.2.27}$$

$$\implies K^T B e_0 - K^T B K w = \mathbf{0} \tag{3.2.28}$$

□

This conditions is implicity describing the objective of a Preconditioned Conjugate Gradient algorithm, where B is the M^{-1} matrix (**VERIFICATION NEEDED**), however this discussion right now it's a digression. Instead let's set B to A, so that it's equivalent to the Energy Norm minimization of Conjugate Gradient, giving us this conditions:

$$K^T A A^{-1} r_0 - K^T A K w = \mathbf{0} \tag{3.2.29}$$

$$K^T r_0 - K^T A K w = \mathbf{0} \tag{3.2.30}$$

Here, we jsut made the substitution of $e_k = A^{-1} r_k$, and $B = A$. Later, we will see how this condition is linked to the idea of an Oblique Projector, similar to how an Orthogonal Projector is able to minimize the 2-Norm of the residual.

3.3 Deriving Conjugate Gradient from Conjugate Directions

By the time this is being written, it's been 70 years since the first time Conjugate Gradient algorithm is proposed by Hestenes and Stiefel back in 1952(**CITATION NEEDED**). Upon their first discussion of the algorithm, numerious 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 iterations. 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 and in the end we point out

its relations to Krylov Subspace, which ultimately, leads to other important new ideas that are not yet present back in 1952. In most time under classroom settings or textbook, 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 we can move on and talk about other things. In this section of the paper, we take a different approach that similar in spirit to a coursenotes by Shewchuk (**CITATION NEEDED**) where we focuses more on that inspirations behind the algorithm and disregard Lanczos Algorithms and Krylov Subspace until the very end.

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) Further assume another matrix $P_k = [p_0\ p_1\ \cdots p_{k-1}]$ as a matrix whose columns is a basis for x_k.

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

Refer back to Energy Norm Minimization using Gradient (3.2.2) for how to obtain the above minimization objective. Using the matrix from the subspace projection method (3.2) where W, V are both P_k, we reformulate the norm minimizations conditions as:

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

Take note that the link between a norm minimzation and an equivalent subspace orthogonality conditions don't garantee to happen for other subspace projection methods, for example the FOM and Bi-Lanczos Methods are orthogonalizations method that doesn't directly link to a norm minimization objective (**CITATION NEEDED**).

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 to be a diagonal matrix and hence we let P_k to be a *matrix whose columns are A-Orthogonal vectors.*

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

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

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

The idea here is: Accumulating vectors p_j into the matrix P_k and then iteratively improve the solution x_k by reducing the error denote as e_k defined as $A^{-1}b - x_k$. Then, we derive the following expression for x_k and the residual $r_k = b - Ax_k$:

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

Let this algorithm be the prototype.

3.3.2 Using the Projector

Here, we consider the above prototype algorithm. Please observe that $AP_kD_k^{-1}P_k$ is a projector, and so is $P_kD_k^{-1}P_k^TA$.

Proof.

$$AP_kD_k^{-1}P_k^T(AP_kD_k^{-1}P_k^T) = AP_kD_k^{-1}P_k^TAP_kD_k^{-1}P_k^T \tag{3.3.7}$$
$$= AP_kD_k^{-1}D_kD_k^{-1}P_k^T \tag{3.3.8}$$
$$= AP_kD_k^{-1}P_k^T \tag{3.3.9}$$

$$P_kD_k^{-1}P_k^TA(P_kD_k^{-1}P_k^TA) = P_kD_k^{-1}D_kD_k^{-1}P_k^TA \tag{3.3.10}$$
$$= P_kD_k^{-1}P_k^TA \tag{3.3.11}$$

□

Both matrices are indeed projectors. Please take note that they are not Hermitian, which would mean that they are not orthogonal projector, hence, oblique projectors. For notational convenience, we denote $\overline{P}_k = P_kD_k^{-1}P_k^T$; then these 2 projectros are:

$$AP_kD_k^{-1}P_k^T = A\overline{P}_k \tag{3.3.12}$$
$$P_kD_k^{-1}P_k^TA = \overline{P}_kA \tag{3.3.13}$$

One immediate consequence is:

$$\text{ran}(I - A\overline{P}_k) \perp \text{ran}(P_k) \tag{3.3.14}$$
$$\text{ran}(I - \overline{P}_kA) \perp \text{ran}(AP_k) \tag{3.3.15}$$

Proof.

$$P_k^T(I - A\overline{P}_k) = P_k^T - P_k^TA\overline{P}_k \tag{3.3.16}$$
$$= P_k^T - D_kD_k^{-1}P_k^T \tag{3.3.17}$$
$$= \mathbf{0} \tag{3.3.18}$$
$$(AP_k)^T(I - \overline{P}_kA) = P_k^TA - P_k^TA\overline{P}_kA \tag{3.3.19}$$
$$= P_k^TA - P_k^TAP_kD_k^{-1}P_k^TA \tag{3.3.20}$$
$$= P_k^TA - P_k^TA \tag{3.3.21}$$
$$= \mathbf{0} \tag{3.3.22}$$

□

Using the properties of the oblique projector, we can proof 2 facts about this simple norm minimization method we developed:

Proposition 3.4 (Residuals are Orthogonal to P_k)**.**

$$r_k = r_0 - A\overline{P}_kr_0 = (I - A\overline{P}_k)r_0 \tag{3.3.23}$$
$$\implies r_k \perp \text{ran}(P_k) \tag{3.3.24}$$

Proposition 3.5 (Generating A Orthogonal Vectors)**.** Given any set of basis vector, for example $\{u_k\}_{i=0}^{n-1}$, one can generate a set of A-Orthogonal vectors from it. More specifically:

$$p_k = (I - \overline{P}_k A)u_k \tag{3.3.25}$$
$$\text{span}(p_k) \perp \text{ran}(AP_k) \tag{3.3.26}$$

For above propositions, we used the immediate consequence of the range of these oblique 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 computing the solution x_k at each step, and the residual at each step, while also getting the residual vector at each step too. However, it would be great if we can accumulate on the same subspace P_k and look for a chance to reuse the computational results from the previous iterations of the algorithm:

Definition 4 (Method of Conjugate Directions)**.**

$$\begin{cases} x_k = x_0 + \overline{P}_k r_0 \\ r_k = (I - A\overline{P}_k)r_0 \\ P_k^T A P_k = D_k \\ \overline{P}_k = P_k D_k^{-1} P_k^T \\ p_k = (I - \overline{P}_k A)u_k \quad \{u_i\}_{i=0}^{n-1} \text{ is a Basis} \end{cases} \tag{3.3.27}$$

With the assitance of a set of basis vector that span the whole space, this algorithm is possible to achieve the objective. Take note that we can accumulate the solution for x_k accumulatively, instead of computing the whole projector process, we have the choice to update it recursively as the newest p_k vector is introduced at that step. Let's Call this formulation of the algorithm: *Conjugate Direction Method* (CDM).

Remark 3.3.1. This CDM method is nothing new, in the original paper from Hestenes and Stiefel back in 1952(**CITATION NEEDED**), they commented on the method of Conjugate Direction, for each chose of basis $\{u_i\}_{i=1}^{n}$ there resides a unique algorithm. If one were to choose the basis to be the set of standard basis vector, then the resulting algorithm will be the equivalent of a Gaussian Eliminations.

Remark 3.3.2 (Geometric Intutions of CDM)**.** What is happening geometrically is that the A-Orthogonal vectors are orthogonal if desribe it under the alternative eigensubspaces. Intuition one should think of a hyper dimensional ellipsoid that sits along the standard basis vector, and transformation of A is stretching and rotating's axis, resulting in a new ellipsoid in a different orientations; if such a transformation is also applied to the axis of ellipsoid then the transformed axis are conjugate vectors. Tracing along the direction of these vectors will ensure minimum redundancy of search directions.

3.3.4 Properties of CDM

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

Proposition 3.6.

$$p_{k+j}^T r_k = p_{k+j}^T r_0 \quad \forall\, 0 \le j \le n-k \tag{3.3.28}$$
$$p_{k+j}^T r_k = p_k^T (I - A\overline{P}_k) r_0 \tag{3.3.29}$$
$$= (p_{k+j}^T - p_{k+j}^T A\overline{P}_k) r_0 \tag{3.3.30}$$
$$= p_{k+j}^T r_0 \tag{3.3.31}$$

This is true because the vector p_{k+j}, a conjugate vector in the future is orthogonal to all previous conjugte vectors.

As a consequence of the above proposition, we obtained a short recurrence for the residuals and solution x of CDM:

Proposition 3.7 (CDM Recurrence)**.**

$$r_k - r_{k-1} = r_0 - A\overline{P}_k r_0 - (r_0 - A\overline{P}_{k-1} r_0) \tag{3.3.32}$$
$$= A\overline{P}_k r_0 - A\overline{P}_{k-1} r_0 \tag{3.3.33}$$
$$= -A p_{k-1} \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} \tag{3.3.34}$$
$$\implies x_k - x_{k-1} = p_{k-1} \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} \tag{3.3.35}$$
$$\text{def: } a_{k-1} := \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} = \frac{\langle p_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} \tag{3.3.36}$$

We define the value of a_{k-1}, and in above, we have 2 equivalent representation. Please take note that, this proposition remains true for the future CG algorithm that we are going to develop.

3.3.5 Conjugate Gradient

Now, consider the case where, the set of basis vector: $\{u\}_{i=0}^{n-1}$ to be the residual vector generated from the CDM itself. Then there are a series of new added lemmas that are true. However, this is where things started to get exciting, because a short recurrence for p_k during each iteration arised and residuals are all orthgonal. We wish to proceed to prove that part.

Lemma 3.3.1.

$$\langle p_{k+j}, A p_k \rangle = \langle r_k, A p_{k+j} \rangle = \langle p_{k+j}, A r_k \rangle \quad \forall\, 0 \le j \le n-k \tag{3.3.37}$$

Proof.

$$p_{k+j} A p_k = p_{k+j}^T A r_k - p_{k+j}^T A\overline{P}_k A r_k \quad \forall\, 0 \le j \le n-k \tag{3.3.38}$$
$$= p_{k+j}^T A r_k \tag{3.3.39}$$
$$\langle p_{k+j}, A p_k \rangle = \langle r_k, A p_{k+j} \rangle = \langle p_{k+j}, A r_k \rangle \tag{3.3.40}$$

□

Lemma 3.3.2.

$$\langle r_k, p_k \rangle = \langle r_k, r_k \rangle \tag{3.3.41}$$

Proof.

$$\begin{aligned} \langle r_k, p_k \rangle &= \langle r_k, p_k \rangle && (3.3.42)\\ &= \langle r_k, r_k \rangle - \langle r_k, \overline{P}_k A r_k \rangle && (3.3.43)\\ &= \langle r_k, r_k \rangle && (3.3.44) \end{aligned}$$

From the first line to the second line, we make use of the definition proposed. □

Proposition 3.8 (CG Generates Orthogonal Residuals)**.**

$$\langle r_k, r_j \rangle = 0 \quad \forall\, 0 \leq j \leq k-1 \tag{3.3.45}$$

Let this above claim be inductively true then consider the following proof:

Proof.

$$\begin{aligned} r_{k+1} &= r_k - a_k A p_k && (3.3.46)\\ \implies \langle r_{k+1}, r_k \rangle &= \langle r_k, r_k \rangle - a_k \langle r_k, A p_k \rangle && (3.3.47)\\ &= \langle r_k, r_k \rangle - \frac{\langle r_k, r_k \rangle}{\langle p_k, A p_k \rangle} \langle r_k, A p_k \rangle && (3.3.48)\\ &= 0 && (3.3.49) \end{aligned}$$

The first line is from the reccurrence of CDM residuals, and then next we make use of the updated definition for a_k. Next we consider:

$$\begin{aligned} p_j &= (I - \overline{P}_j A) r_j \quad \forall\, 0 \leq j \leq k-1 && (3.3.50)\\ &\implies r_j = p_j + \overline{P}_j A r_j && (3.3.51)\\ r_k &= (I - A\overline{P}_k) r_0 && (3.3.52)\\ r_k \perp \text{ran}(P_k) &\implies \langle r_k, r_j \rangle = \langle r_k, p_j + \overline{P}_j A r_j \rangle = 0 && (3.3.53) \end{aligned}$$

The second line is a result of the first line. Here we again make use of the projector $I - A\overline{P}_k$. 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 3.9 (CG Recurrences)**.**

$$p_k = r_k + b_{k-1} p_{k-1} \quad b_{k-1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \tag{3.3.54}$$

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

$$\begin{aligned} p_k &= (I - \overline{P}_k A) r_k && (3.3.55)\\ r_k - \overline{P}_k A r_k &= r_k - P_k D_k^{-1} P_k^T A r_k && (3.3.56)\\ &= r_k - P_k D_k^{-1} (A P_k)^T r_k && (3.3.57) \end{aligned}$$

Observe:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} \tag{3.3.58}$$

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

$$\langle p_j, Ar_k \rangle \quad \forall\, 0 \leq j \leq k-2 \tag{3.3.59}$$
$$\langle p_j, Ar_k \rangle = \langle r_k, Ap_j \rangle \tag{3.3.60}$$
$$= \langle r_k, a_j^{-1}(r_j - r_{j+1}) \rangle \tag{3.3.61}$$
$$= a_j^{-1} \langle r_k, (r_j - r_{j+1}) \rangle = 0 \tag{3.3.62}$$

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 3.6)), and the last line is true for all $0 \leq j \leq k-2$ by the orhogonality of the residual proved in CG Generates Orthogonal Residuals (Proposition 3.7). Therefore we have:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} = a_{k-1}^{-1} \langle r_k, (r_{k-1} - r_k) \rangle \xi_k \tag{3.3.63}$$

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:

$$p_k = r_k - P_k D_k^{-1} (AP_k)^T r_k \tag{3.3.64}$$
$$= r_k - P_k D_k^{-1} a_{k-1}^{-1} (\langle r_k, (r_{k-1} - r_k) \rangle) \xi_k \tag{3.3.65}$$
$$= r_k - \frac{a_{k-1}^{-1} \langle -r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{3.3.66}$$
$$= r_k + \frac{a_{k-1}^{-1} \langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{3.3.67}$$
$$= r_k + \left(\frac{\langle r_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \right)^{-1} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{3.3.68}$$
$$= r_k + \frac{\langle r_k, r_k \rangle}{\langle r_{k-1}, r_{k-1} \rangle} p_k \tag{3.3.69}$$

We make use of the definition for a_{k-1} for the CDM algorithm. At this point, we have proven the short CG recurrences for p_k. □

Up until this point we have proven the usual of conjugate gradient proposed by Hestenes & Stiefe (**CITATION NEEDED**), we started with the minimizations objective and the

properties of P_k, then we define a recurrences for the residual (Simultaneously the solution x_k), and the A-Orthogonal vectors using a basis as assistance for the generations process. Next, we make the key changes of the assistance basis, making it equal to the set of residuals vector generated from the algorithm itself; after some proofs, we uncovered the exact same parameters found in most of the definitions of the CG algorithm, which we refers to as Residual Assisted Conjugate Gradient. Here we proposed the CG:

Definition 5 (CG)**.**

$$p^{(0)} = b - Ax^{(0)} \tag{3.3.70}$$
$$\text{For } i = 0, 1, \cdots \tag{3.3.71}$$
$$a_i = \frac{\|r^{(i)}\|^2}{\|p^{(i)}\|_A^2}$$
$$x^{(i+1)} = x^{(i)} + a_i p^{(i)}$$
$$r^{(i+1)} = r^{(i)} - a_i A p^{(i)} \tag{3.3.72}$$
$$b_i = \frac{\|r^{(j+1)}\|_2^2}{\|r^{(i)}\|_2^2}$$
$$p^{(i+1)} = r^{(i+1)} + b_i p^{(i)}$$

That is the algorithm, stated with all the iteration number listed as a super script inside of a parenthesis. Which is equivalent to what we have proven for the Residual Assisted Conjugate Gradient.

3.3.6 CG and Krylov Subspace

The conjugate Gradient Algorithm is actually a residual assisted conjugate gradient, a special case of the algorithm we derived at the start of the excerp. The full algorithm can be seem by the short recurrence for the residual and the conjugation vector. This part is trivial. Next, we want to show the relations to the Krylov Subspace, which only occurs for the Residual Assisted Conjugate Gradient algorithm.

Proposition 3.10.

$$p_k \in \mathcal{K}_{k+1}(A|r_0) \tag{3.3.73}$$
$$r_k \in \mathcal{K}_{k+1}(A|r_0) \tag{3.3.74}$$

Proof. The base case is tivial and it's directly true from the definition of Residual Assisted Conjugate Gradient: $r_0 \in \mathcal{K}_1(A|r_0), p_0 = r_0 \in \mathcal{K}_1(A|r_0)$. 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:

$$r_{k+1} = r_k - a_k A p_k \tag{3.3.75}$$
$$\in r_k + A\mathcal{K}_{k+1}(A|r_0) \tag{3.3.76}$$
$$\in r_k + \mathcal{K}_{k+2}(A|r_0) \tag{3.3.77}$$
$$r_k \in \mathcal{K}_{k+1}(A|r_0) \subseteq \mathcal{K}_{k+2}(A|r_0) \tag{3.3.78}$$
$$\implies r_{k+1} \in \mathcal{K}_{k+2}(A|r_0) \tag{3.3.79}$$

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

$$p_{k+1} = r_{k+1} + b_k p_k \tag{3.3.80}$$
$$\in r_{k+1} + \mathcal{K}_{k+1}(A|r_0) \tag{3.3.81}$$
$$\in \mathcal{K}_{k+2}(A|r_0) \tag{3.3.82}$$

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)**.**

$$\text{choose: } x_k \in x_0 + \mathcal{K}_k(A|r_0) \text{ s.t: } r_k \perp \mathcal{K}_k(A|r_0) \tag{3.3.83}$$

Take note that, $\text{ran}(P_k) = \mathcal{K}_k(A|r_0)$ because the index starts with zero. The above formulations gives theoretical importance for the Conjugate Gradient Algorithm. □

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 amphasize 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 wil inevitably 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 orthgonal 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 algorithm will inherit the properties of the Arnoldi Iterations.

Before stating the algorithm, I would like to point out the interpreations 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 orhtogonal projector with the vector q_1, which is $q_1 q_1^H$, 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 $I - q_1 q_1^H$ be $h_{1,2} q_2$, and let the projection onto $q_1 q_1^H$

be $h_{1,1}$. This completes the first column of H_k, we do this recursively. Please allow me to demonstrate:

$$(\tilde{H}_k)_{2,1}q_2 = (I - q_1q_1^H)Aq_1 \tag{3.4.2}$$
$$(\tilde{H}_k)_{1,1}q_1 = q_1q_1^HAq_1 \tag{3.4.3}$$
$$Q_2 := \begin{bmatrix} q_1 & q_2 \end{bmatrix} \tag{3.4.4}$$
$$(\tilde{H}_k)_{3,2}q_3 = (I - Q_2Q_2^H)Aq_2 \tag{3.4.5}$$
$$(\tilde{H}_k)_{1:2,2} = Q_2Q_2^HAq_2 \tag{3.4.6}$$
$$Q_3 := \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \tag{3.4.7}$$
$$\vdots \tag{3.4.8}$$
$$Q_j := \begin{bmatrix} q_1 & q_2 & \cdots & q_j \end{bmatrix} \tag{3.4.9}$$
$$(\tilde{H}_k)_{j+1,j}q_{j+1} = (I - Q_jQ_j^H)Aq_j \tag{3.4.10}$$
$$(\tilde{H}_k)_{1:j,j} = Q_jQ_j^HAq_j \tag{3.4.11}$$
$$\vdots \tag{3.4.12}$$
$$Q_k := \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{3.4.13}$$
$$(\tilde{H}_k)_{k+1,k}q_{k+1} = (I - Q_kQ_k^H)Aq_k \tag{3.4.14}$$
$$(\tilde{H}_k)_{1:k,k} = Q_kQ_k^HAq_k \tag{3.4.15}$$

Reader please observe that Q_k is going to be orthogonal because how at the start, $q_1q_1^H$ and $I - q_1q_1^H$ is giving us an orthogonal subspace. As a consequence, we can express the recurrences of the subspace vector in matrix form:

$$AQ_k = Q_{k+1}\tilde{H}_k \tag{3.4.16}$$
$$Q_k^HAQ_k = H_k \tag{3.4.17}$$

And here, we explicitly define H_k to be the principal submatrix of $\tilde{H}_k$. Reader please immediately observe that, if A is symmetric, then it has to be the case that $Q_k^HAQ_k$ is also symmetric, which will make H_k to be symmetric as well, which implies that H_k will be a Symmetric Tridiagonal Matrix. And under that assumption, we can develop the Lanczos Algorithm. Instead of orthogoalizing against all previous vectors, we have the option to simply orthogonalize against the previous q_k, q_{k-1} vector. And we can reuse the sub-diagonal elements for q_{k-1}; giving us the Lanczos Algorithm.

3.4.2 Arnoldi Produces Orthogonal Basis for Krylov Subspace

One important observations 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 3.11.

$$\text{ran}(Q_k) = \mathcal{K}_k(A|q_1) \tag{3.4.18}$$

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:

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

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 $Aq_k \in \text{ran}(Q_k)$; 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 $\text{ran}(Q_k)$, resulging in $h_{k+1,k} = 0$. □

3.4.3 The Lanczos Iterations

Definition 6 (Lanczos Iterations)**.**

$$\begin{aligned} & \text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 && (3.4.19) \\ & \text{set: } \beta_0 = 0 && (3.4.20) \\ & \text{For } j = 1, 2, \cdots && (3.4.21) \\ & \quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1} q_{j-1} \\ & \quad \alpha_j := \langle q_j, \tilde{q}_{j+1} \rangle \\ & \quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j && (3.4.22) \\ & \quad \beta_j = \|\tilde{q}_{j+1}\| \\ & \quad q_{j+1} := \tilde{q}_{j+1}/\beta_j \end{aligned}$$

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. Note that another equivalent algorithm where I tweaked it to handle the base case of T_k being a 1×1 matrix can be phrased in the following way:

$$\begin{aligned} & \text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \\ & \alpha_1 := \langle q_1, Aq_1 \rangle \\ & \beta_0 := 0 \\ & \text{Memorize : } Aq_1 \\ & \text{For } j = 1, 2, \cdots \\ & \quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1} q_{j-1} && (3.4.23) \\ & \quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\ & \quad \beta_j = \|\tilde{q}_{j+1}\| \\ & \quad q_{j+1} := \tilde{q}_{j+1}/\beta_j \\ & \quad \alpha_{j+1} := \langle q_{j+1}, Aq_{j+1} \rangle \\ & \quad \text{Memorize: } Aq_{j+1} \end{aligned}$$

The algorithm generates the following 2 matrices, Q_k which is orthogonal and it spans $\mathcal{K}_k(A|q_1)$, and a Symmetric Tridiagonal Matrix:

$$Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{3.4.24}$$

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{3.4.25}$$

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 as it simply make use of the previous 2 vectors:

Theorem 2 (Lanczos Recurrenes)**.**

$$AQ_k = Q_k T_k + \beta_k q_{k+1} \xi_k^T = Q_{k+1} \tilde{T}_k \tag{3.4.26}$$

$$\implies Aq_k = \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \quad \forall\, 2 \leq j \leq k \tag{3.4.27}$$

$$\implies Aq_1 = \alpha_1 q_1 + \beta_1 q_2 \tag{3.4.28}$$

Often time, we refers the $k \times k$ symmetric tridiagonal matrix generated from Iterative Lanczos as T_k. Finally; I wish to make the following important remark about the algorithm for later use. Given a matrix A and an initial vector q_1, The lanczos algorithm produces an irreducible Symmetric Tridiagonal Matrix that has unique eigenvalues. The proof for the fact that any Symmetric Tridiaogonal Matrices with Non-zeros on the sub/super diagonal must have unique non-zero eigenvalues is skipped. What we can immediate show here is the fact that Lanczos Algorithm will produce such a matrix.

Proposition 3.12. 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 subpace under A, and in that sense, the algorithm has to terminate due to a divides by zero error. □

4 Analysis of Conjugate Gradient and Lanczos Iterations

In this section we state the terminations conditions for the Lanczos iterations and the CG algorithm we developed using the property of Krylov Subspace. This is just a throughly discussion about these 2 algorithm and applying the foundations and it's not following any references.

4.1 Conjugate Gradient and Matrix Polynomial

One important result of the optimization objective listed CG and Krylov Subspace is the connections to 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 denotes the error vector.

Proof.

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

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

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

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

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

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

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

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

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

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

(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 mulitplied by r_0 and then use $A^{-1}b$ to subtract both side 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 symetric 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 eigendecompoition 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 theminimum and the maximum of the eigenvalues for the matrix A (from (4.1.9) to (4.1.10)). Finally, please notice thta we use an monic $p_{k+1}(x)$ at the and to simplify things. □

The above results will be useful for proving the convergenc 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 $\|Ax\| \leq \|A\|\|x\|$ 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.

4.2 Termination Conditions of CG

Proposition 4.2 (Termination of CG)**.** The CG terminates after k number of iterations where k is the grade of the krylov subspace of A wrt to r_0.

Recall from CG and Krylov Subspace (3.3.6) where we established that $r_k \in \mathcal{K}_{k+1}(A|r_0), r_k \perp \mathcal{K}_k(A|r_0)$. Recall that once the grade of the Krylov subspace is reached, Krylov subspace becomes invariant, and hence, $r_k = \mathbf{0}$, the algorithm terminates.

In addition, grade$(A|r_0)$ depends on the number of unique λ_i where $q_i^T r_0$ is not zero, q_i here is the eigenvector that has eigenvalue r_i. We consider the eigen decomposition of matrix A, then it's direct using When the Grade is Reached (proposition 3.1).

4.3 Convergence Rate of CG under Exact Arithematic

In this section we make heavy use of Greenbaum's Analysis for convergence rate of the algorithm. The core is 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.

4.3.1 Uniformly Distributed Eigenvalues

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

$$\frac{\|e_k\|_A}{\|e_k\|_A} \leq 2\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k \tag{4.3.1}$$

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 are 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 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, while also making it monic:

$$T_k(x) = \arg\min_{p \in \mathcal{P}_k} \max_{x \in [-1,1]} |p(x)| \tag{4.3.2}$$

$$p_k(x) := \frac{T_k(\varphi(x))}{T_k(\varphi(0))} \quad \text{where: } \varphi(x) := \frac{2x - \lambda_1 - \lambda_n}{\lambda_n - \lambda_1} \tag{4.3.3}$$

$$\implies p_k(x) = \arg \min_{\substack{p \in \mathcal{P}_k \\ \text{s.t: } p(0)=1}} \max_{x \in [\lambda_1, \lambda_n]} |p(x)| \tag{4.3.4}$$

At this point, we have defined a new polynomial p_k that minimizes the inf norm over the convex hull of the eigenvalues and it's monic. 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:

$$\forall x \in [\lambda_1, \lambda_n] : \left| \frac{T_k(\varphi(x))}{T_k(\varphi(0))} \right| \leq \left| \frac{1}{T_k(\varphi(0))} \right| \tag{4.3.5}$$

Next, our objective is to find any upper bound for the quantities on the RHS in relations 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 alterantive form of Chebyshev T for values outside of the $[-1, 1]$:

$$T_k(x) = \cosh(k \ \text{arccosh}(z)) \quad \forall z \geq 1 \tag{4.3.6}$$

$$\implies T_k(\cosh(\zeta)) = \cosh(k\zeta) \quad z := \cosh(\zeta) \tag{4.3.7}$$

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.

$$\varphi(0) = \cosh(\zeta) = \cosh(\ln(y)) \quad \ln(y) := \zeta \tag{4.3.8}$$

$$\text{recall: } \cosh(x) = (\exp(-x) + \exp(x))/2 \tag{4.3.9}$$

$$\implies \cosh(\ln(y)) = (y + y^{-1})/2 \tag{4.3.10}$$

$$\varphi(0) = (y + y^{-1})/2 \tag{4.3.11}$$

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:

$$-\frac{\kappa+1}{\kappa-1} = \frac{1}{2}(y + y^{-1}) \tag{4.3.12}$$

$$y = \frac{\sqrt{\kappa} \pm 1}{\sqrt{\kappa} \mp 1} \tag{4.3.13}$$

It's a quadratic and we solved it. The above $\pm, \mp$ are correlated, meaning that they are of opposite sign, which gives us 2 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:

$$\varphi(0) = \frac{1}{2}(y + y^{-1}) \tag{4.3.14}$$

$$\implies T_k(\varphi(0)) = T_k(\cosh(\ln(y))) \tag{4.3.15}$$

$$= \cosh(k\ln(y)) \tag{4.3.16}$$

$$= (y^k + y^{-k})/2 \tag{4.3.17}$$

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

$$\frac{1}{T_k(\varphi(0))} = 2(y^k + y^{-k})^{-1} \tag{4.3.18}$$

$$= 2\left(\left(\frac{\sqrt{\kappa} \pm 1}{\sqrt{\kappa} \mp 1}\right)^k + \left(\frac{\sqrt{\kappa} \mp 1}{\sqrt{\kappa} \pm 1}\right)^{-k}\right)^{-1} \tag{4.3.19}$$

$$= 2\left(\underbrace{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k}_{>1} + \underbrace{\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{-k}}_{<1}\right)^{-1} \tag{4.3.20}$$

$$\leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{4.3.21}$$

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

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_1,\lambda_n]} |p_{k+1}(x)| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{4.3.22}$$

□

4.3.2 One Outlier Eigenvalue

Using the derived theorem, we can extend it to other type of distributions of eigenvalues. Imagine one of the extreme case where some matrices that have one group of eigenvalues that are close together and one signle 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 tigher convergence bound that being only paramaterized by the range of clustered eigenvalues can be obtained and it is:

$$\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A} \leq 2\left(\frac{\sqrt{\kappa_{n-1}}-1}{\sqrt{\kappa_{n-1}}+1}\right)^{k-1} \qquad \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{4.3.23}$$

Reader please observe that, the outlier eigenvalue λ_n plays a smaller role in determining the convergence rate of the algorithm compare 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:

$$\hat{T}^{(k)}_{[a,b]}(x) := T_k\left(\frac{2x-b-a}{b-a}\right) \tag{4.3.24}$$

Next, we consider the following polynomial:

$$p_k(x) := \frac{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(x)}{T^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)}\left(\frac{\lambda_n - x}{x}\right) \tag{4.3.25}$$

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:

$$\frac{\lambda_n - z}{\lambda_n} \in [0,1] \quad \forall z \in [\lambda_1, \lambda_n] \tag{4.3.26}$$

$$|p_k(x)| \leq \left|\frac{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(x)}{\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)}\frac{\lambda_n - z}{\lambda_n}\right| \leq \frac{1}{\left|\hat{T}^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0)\right|} \tag{4.3.27}$$

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

$$T^{(k-1)}_{[\lambda_1,\lambda_{n-1}]}(0) = \left|T_{k-1}\left(\frac{-\lambda_{n-1}-\lambda_1}{\lambda_{n-1}-\lambda_1}\right)\right| \tag{4.3.28}$$

$$= \frac{1}{2}(y^{k-1} + y^{-(k-1)}) \tag{4.3.29}$$

$$\text{where: } y = \frac{\sqrt{\kappa_{n-1}}+1}{\sqrt{\kappa_{n-1}}-1}, \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{4.3.30}$$

Substituting the value for y we obtain the bound:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa_{n-1}}-1}{\sqrt{\kappa_{n-1}}+1}\right)^{k-1} \tag{4.3.31}$$

□

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:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)\left(\frac{\sqrt{\kappa_0}-1}{\sqrt{\kappa_0}+1}\right)^{k-1} \tag{4.3.32}$$

Where κ_0 is λ_n/λ_2.

Proof.

$$w(z) := \frac{\lambda_1 - z}{\lambda_1} \tag{4.3.33}$$

$$p_k(z) := w(z)\left(\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}\right) \tag{4.3.34}$$

$$\implies \max_{x\in[\lambda_2,\lambda_n]} |w(x)| = \frac{\lambda_n-\lambda_1}{\lambda_1} \tag{4.3.35}$$

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:

$$|p_k(x)| = \left|w(x)\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(x)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| \tag{4.3.36}$$

$$\leq \left|\frac{w(x)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| \tag{4.3.37}$$

$$\leq \left|\left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)\right| \tag{4.3.38}$$

$$\implies \leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)2\left(\frac{\sqrt{\kappa_0}+1}{\sqrt{\kappa_0}-1}\right)^{k-1} \tag{4.3.39}$$

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 make the convergence better or worse, but in this case, the weight $w(x)$ introduces a term that grows inversely proportional to λ_1. □

4.4 From Conjugate Gradient to Lanczos

Up until this point of disucssion, we had been brewing the fact that, the Iterative Lanczos Algorithm and the Conjugate gradient algorithm are pretty much the same thing. 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 generates orthogonal vectors, for Lanczos they are the q_i vector and for CG they are the r_i vectors.

These 2 properties in particular, is 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 as CG in the first section, and yet it doesn't make use of any orthogonal projector. Further more, one might also notice that Iterative Lanczos are for General Symmetric Matrices whie CG are only for positive definite matrices. To see how everything ties together, we have to go both directions way to show the connections between these 2 iterative algorithms, which are what this section and the subsequent section about. Here, we refers 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:

$$q_1 = \hat{r}_0 \tag{4.4.1}$$
$$q_2 = -\hat{r}_1 \tag{4.4.2}$$
$$\vdots \tag{4.4.3}$$
$$q_j = (-1)^{j+1}\hat{r}_{j+1} \tag{4.4.4}$$

Here, $\hat{r}_j := r_j/\|r_j\|$ and we can fill in the Lanczos Tridiagonal matrix using the CG parameters to obtain the tridiagonalization of the Positive definite matrix A:

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \leq j \leq k-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \leq j \leq k-2 \\ \alpha_1 = a_0^{-1} & \\ \beta_1 = \frac{\sqrt{b_0}}{\alpha_0} & \end{cases} \tag{4.4.5}$$

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

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 Tridiaogonal matrix and a_k, b_k, finall we resolve the sign problem between the Lanczos vectors and the residual vectors.

4.4.1 The Base Case

Right from the start of the CG iteration we have:

$$r_0 = p_0 \tag{4.4.6}$$
$$r_1 = r_0 - a_0 A r_0 \tag{4.4.7}$$
$$Ar_0 = a_0^{-1}(r_0 - r_1) \tag{4.4.8}$$
$$Ar_0 = \frac{\|r_0\|_A^2}{\|r_0\|^2}(r_0 - r_1) \tag{4.4.9}$$

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

$$A\|r_0\|q_1 = \frac{\|r_0\|_A^2}{\|r_0\|^2}\left(\|r_0\|q_1 + \|r_1\|q_2\right) \tag{4.4.10}$$
$$= \frac{\|r_0\|_A^2}{\|r_0\|^2}\|r_0\|q_1 + \frac{\|r_1\|}{\|r_0\|}q_2 \tag{4.4.11}$$

And from this relation, using the Lanczos recurrence theoremit 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 Lancozs and the CG for the first iterations of the CG algorithm.

4.4.2 The Inductive Case

Lemma 4.4.1. Inductively we wish to show the relation that:

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \leq j \leq n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \leq j \leq n-2 \end{cases} \tag{4.4.12}$$

Proof. We start by consiering:

$$r_j = r_{j-1} - a_{j-1}Ap_{j-1} \tag{4.4.13}$$
$$= r_{j-1} - a_{j-1}A(r_{j-1} + b_{j-2}p_{j-1}) \tag{4.4.14}$$
$$= r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-1} \tag{4.4.15}$$

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

$$r_{j-1} = r_{j-1} - a_{j-2}Ap_{j-1} \tag{4.4.16}$$
$$r_{j-1} - r_{j-1} = a_{j-2}Ap_{j-1} \tag{4.4.17}$$
$$Ap_{j-1} = a_{j-2}^{-1}(r_{j-2} - r_{j-1}) \tag{4.4.18}$$

Here, we can subtitute the results of for the term Ap_{j-1}, and then we can express the

recurrence of residual purely in terms of residual. Consider:

$$r_j = r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-2} \tag{4.4.19}$$

$$= r_{j-1} - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}\left(r_{j-2} - r_{j-1}\right) \tag{4.4.20}$$

$$= \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} \tag{4.4.21}$$

$$a_{j-1}Ar_{j-1} = \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} \tag{4.4.22}$$

$$Ar_{j-1} = \left(\frac{1}{a_{j-1}} + \frac{b_{j-2}}{a_{j-2}}\right)r_{j-1} + \frac{r_j}{a_{j-1}} - \frac{b_{j-2}}{a_{j-2}}r_{j-2} \tag{4.4.23}$$

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:

$$Ar_j = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)r_j + \frac{r_{j+1}}{a_j} - \frac{b_{j-1}}{a_{j-1}}r_{j-1} \tag{4.4.24}$$

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 quiet matching the form of the recurrence of β_k from the lanczos algorithm. To match it, we need the coeffcients of r_{j-1} and r_{j+1} to be in the same form, paramaterized by the same iterations parameter: j. To do that, consider the doing this:

$$q_{j+1} := \frac{r_j}{\|r_j\|} \tag{4.4.25}$$

$$q_j := -\frac{r_{j-1}}{\|r_{j-1}\|} \quad \text{Note: This is Negative} \tag{4.4.26}$$

$$q_{j+2} := \frac{r_{j+1}}{\|r_{j+1}\|} \tag{4.4.27}$$

$$\implies A\|r_j\|q_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)\|r_j\|q_{j+1} + \frac{\|r_{j+1}\|q_{j+2}}{a_j} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}}q_j \tag{4.4.28}$$

$$Aq_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)q_{j+1} + \frac{\|r_{j+1}\|}{a_j\|r_j\|}q_{j+2} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|}q_j \tag{4.4.29}$$

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:

$$\frac{\|r_{j+1}\|}{a_j\|r_j\|} = \frac{1}{a_j}\sqrt{b_j} \tag{4.4.30}$$

$$\frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|} = \frac{b_{j-1}}{a_{j-1}}\frac{1}{\sqrt{b_{j-1}}} = \frac{\sqrt{b_{j-1}}}{a_{j-1}} \tag{4.4.31}$$

$$\implies \begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \leq j \leq n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \leq j \leq n-2 \end{cases} \tag{4.4.32}$$

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. $\square$

4.4.3 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 equivalcne residual: r_{j-1} in CG. Here, I want to point out the fact that, there are potentially 2 substitution 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:

$$\begin{cases} q_{j+1} := \pm \frac{r_j}{\|r_j\|} \\ q_j := \mp \frac{r_{j-1}}{\|r_{j-1}\|} \\ q_{j+2} := \pm \frac{r_{j+1}}{\|r_{j+1}\|} \end{cases} \tag{4.4.33}$$

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 negative one would give the same equality, and we can always mulitply by and other 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:

$$q_1 = \hat{r}_0 \tag{4.4.34}$$
$$q_2 = -\hat{r}_1 \tag{4.4.35}$$
$$\vdots \tag{4.4.36}$$
$$q_j = (-1)^{j+1} \hat{r}_{j+1} \tag{4.4.37}$$

And at this point, we have established the equivalence going from the Conjugate Gradient algorithm to the Lanczos Algorithm. And the moral of the story is, CG is a special case of applying the Lanczos Iterations with $q_1 = r_0$ to a positive definite matrix. However, something is still off and one can ask the following questions to inquiry further, leading us to more discussion between these 2 algorithms.

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

2.) How are the A-Orthogonal vectors p_k from CG relates to Lancozs?

4.5 From Lanczos to Conjugate Gradient

In this section, we show the equivalence of the Conjugate Gradient and Lanczos iterations by deriving the CG using the Lanczos.

The goal is to show that CG is a special case of Lanczos. In the end we discuss the key that it can inspire solvers for symmetric indefinite systems of linear equations.

We start of by considering the LU decomposition of the the Tridiagonal matrix of Lanczos and us 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.

4.5.1 Matching the Residual and Conjugate Vectors

In this section, we go from the Iterative Lanczos algorithm to the Conjugate Gradient algorithm and we seek to establish a link between the solution x_k, p_k from CG with the lanczos vectors and the symmetric tridiagonal matrix generated by Lanczos. This section will also play an important role for the backwards analysis for the floating point behaviors for the CG algorithm in the later parts of the paper. At the end, we highlight some of the hidden insights that this particular derivation of quivalence leads to, and how it inspires algorithms that directly solves 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:

$$\begin{cases} y_k = T_k^{-1}\beta\xi_1 \\ x_k = x_0 + Q_k y_k \\ r_k = -\beta_k \xi_k^T y_k q_{k+1} \end{cases} \tag{4.5.1}$$

The quantities α, β are the diaognal and the sub or super diagonal of the matrix T_k from the Iterative Lanczos Algorithm, and 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. β without the subscript denotes $\|r_0\|$.

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

$$AQ_k = Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} \tag{4.5.2}$$

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 that: $p_k \in \mathcal{K}_{k+1}(A|r_0)$, the matrix P_k, Q_k spans the same subspace, and that means:

$$x_{k+1} = x_0 + Q_k y_k \tag{4.5.3}$$
$$r_{k+1} = r_0 - AQ_k y_k \tag{4.5.4}$$
$$Q_k^H r_{k+1} = Q_k^H r_0 - Q_k^H A Q_k y_k \tag{4.5.5}$$
$$\implies 0 = \beta\xi_1 - T_k y_k \tag{4.5.6}$$
$$y_k = T_k^{-1}\beta\xi_1 \tag{4.5.7}$$

Now to get the residual we simply consider:

$$
\begin{aligned}
r_{k+1} &= r_0 - AQ_k y_k && (4.5.8)\\
&= r_0 - AQ_k T_k^{-1}\beta\xi_1 && (4.5.9)\\
\implies &= \beta q_1 - AQ_k T_k^{-1}\beta\xi_1 && (4.5.10)\\
&= \beta q_1 - Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} T_k^{-1}\beta\xi_1 && (4.5.11)\\
&= \beta q_1 - (Q_k T_k + \beta_k q_{k+1}\xi_k^T) T_k^{-1}\beta\xi_1 && (4.5.12)\\
&= \beta q_1 - (Q_k \beta\xi_1 + \beta_k q_{k+1}\xi_{k+1} T_k^{-1}\beta\xi_1) && (4.5.13)\\
&= -\beta_k q_{k+1}\xi_k^T T_k^{-1}\beta\xi_1 && (4.5.14)
\end{aligned}
$$

On the third line we racall 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 orthgonal 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:

$$P_k = Q_k U_k^{-1} \tag{4.5.15}$$

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:

$$
\begin{aligned}
x_k &= x_0 + Q_k y_k && (4.5.16)\\
&= x_0 + Q_k T_k^{-1}\beta\xi_1 && (4.5.17)\\
&= x_0 + Q_k U_k^{-1} L_k^{-1}\beta\xi_1 && (4.5.18)\\
&= x_0 + P_k L_k^{-1}\beta\xi_1 && (4.5.19)
\end{aligned}
$$

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-orthgonal. To show that we consider:

$$
\begin{aligned}
&P_k^T A P_k && (4.5.20)\\
&= (Q_k U_k^{-1})^T A Q_k U_k^{-1} && (4.5.21)\\
&= U_k^{-T} Q_k^T A Q_k U_k^{-1} && (4.5.22)\\
&= U_k^{-T} T_k U_k^{-1} && (4.5.23)\\
&= U_k^{-T} L_k && (4.5.24)
\end{aligned}
$$

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. □

4.5.2 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 advantange 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.

$$P_k = \begin{bmatrix} p_1 & p_2 & \cdots & p_k \end{bmatrix} \quad Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{4.5.25}$$

Using the invertibility of the matrix A and Cauchy Interlace Theorem, T_k is invertible, we consider the LU decomposition of the symmetric tridiagonal matrix:

$$T_k = L_k U_k = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix} \begin{bmatrix} u_1 & \beta_1 & & \\ & u_2 & \beta_2 & \\ & & \ddots & \beta_{k-1} \\ & & & u_k \end{bmatrix} \tag{4.5.26}$$

Reader should agree with some considerations that the upper diagonal of U_k are indeed the same as the upper diagonal of the SymTridiagonal matrix T_k. And reacall the expression for x_k from the previous section, we have:

$$\begin{aligned} x_k &= x_0 + P_k L_k^{-1} \beta \xi_1 && (4.5.27) \\ x_k - x_{k-1} &= P_k L_k^{-1} \beta \xi_1 - P_{k-1} L_{k-1}^{-1} \beta \xi_1 && (4.5.28) \\ &= P_k \beta (L_k^{-1})_{:,1} - P_{k-1} \beta (L_{k-1}^{-1})_{:,1} && (4.5.29) \\ &= \beta (L_k^{-1})_{k,1} P_k && (4.5.30) \\ \implies x_k &= x_{k-1} + \beta (L_k^{-1})_{k,1} p_k && (4.5.31) \end{aligned}$$

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:

$$\begin{aligned} P_k &= Q_k U_k^{-1} && (4.5.32) \\ P_k U_k &= Q_k && (4.5.33) \\ \implies \beta_{k-1} p_{k-1} + u_k p_k &= q_k && (4.5.34) \\ u_k p_k &= q_k - \beta_{k-1} p_{k-1} && (4.5.35) \\ p_k &= u_k^{-1} (q_k - \beta_{k-1} p_{k-1}) && (4.5.36) \end{aligned}$$

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:

$$T_k = L_k U_k \tag{4.5.37}$$

$$T_{k+1} = \begin{bmatrix} T_k & \beta_k \xi_k \\ \beta_k \xi_k^T & \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} L_k & \mathbf{0} \\ l_k \xi_k^{-1} & 1 \end{bmatrix} \begin{bmatrix} U_k & \eta_k \xi_k \\ \mathbf{0} & u_{k+1} \end{bmatrix} \tag{4.5.38}$$

$$= \begin{bmatrix} L_k U_k & \eta_k L_k \xi_k \\ l_k \xi_k^T U_k & \eta_k l_k \xi_k^T \xi_k + u_{k+1} \alpha_k \end{bmatrix} \tag{4.5.39}$$

$$= \begin{bmatrix} T_k & \eta_k (L_k)_{:,k} \\ l_k (U_k)_{k,:} & \eta_k l_k + u_{k+1} \end{bmatrix} \tag{4.5.40}$$

$$= \begin{bmatrix} T_k & \eta_k \\ l_k u_k & \eta_k l_k + u_{k+1} \end{bmatrix} \tag{4.5.41}$$

Note that I changed the upper diagonal of matrix U at the top to be η_k instead of β_k so 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:

$$\begin{cases} u_{k+1} & = \alpha_{k+1} - \beta_k^2 / u_k \\ l_k & = \beta_k / u_k \end{cases} \tag{4.5.42}$$

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:

$$L_k^{-1} L_k = I \tag{4.5.43}$$

$$\begin{bmatrix} L_k^{-1} & \mathbf{0} \\ s_k^T & d_{k+1} \end{bmatrix} \begin{bmatrix} L_k & \mathbf{0} \\ l_k \xi_k^T & 1 \end{bmatrix} = I \tag{4.5.44}$$

$$\begin{bmatrix} I & \mathbf{0} \\ s_k^T L_k + d_{k+1} l_k \xi_k^T & d_{k+1} \end{bmatrix} = I \tag{4.5.45}$$

It equals to the identity matrix therefore $d_{k+1} = 1$, and it has to be that the 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:

$$s^T L_k + d_{k+1} l_k \xi_k^T = \mathbf{0} \tag{4.5.46}$$
$$L_k^T s_k + d_{k+1} l_k \xi_k = \mathbf{0} \tag{4.5.47}$$
$$s_k + L^{-T} d_{k+1} l_k \xi_k = \mathbf{0} \tag{4.5.48}$$
$$(s_k)_1 + d_{k+1} l_k ((L_k^{-1})\xi_k)_1 = 0 \tag{4.5.49}$$
$$(s_k)_1 + d_{k+1} l_k (L_k^{-1})_{k,1} = 0 \tag{4.5.50}$$
$$\implies (s_k)_1 = -l_k (L_k^{-1})_{k,1} \tag{4.5.51}$$
$$(s_k)_1 = (L_{k+1}^{-1})_{k+1,1} \quad \text{by definition} \tag{4.5.52}$$
$$\implies (L_{k+1}^{-1})_{k+1,1} = -l_k (L_k^{-1})_{k,1} \tag{4.5.53}$$

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 side by inverse of L_k^T. And from the third to the forth 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 at the same time.

Remark 4.5.1. The derivation might seems excessive for the discussion, but it's part of the analysis of the Conjugate Gradient. It derived the conjugate gradient without using the fact that A is symmetric positive definite providing potential to new algorithm that can solve symmetric indefinite system directly. The Lanczos Algorithm for linear system (we refer to the method derived in the above section) is a special case of FOM (**CITATION NEEDED**) when the matrix A is symmetric. The above algorithm is just FOM with a short term recurrences for its parameters, and it's based on the fact that A is symmetric. It's implied from the above derivation that under exact arithemtic, CG can be applied to symmetric indefinite system, if we have the luck where T_j is non-singular for all $j \leq k$. 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 points for the indefinite error norm. However there are 2 problems solving Symmetric Indefinite using the CG is that, the following recurrence doesn't hold anymore and it should be restated as:

$$AQ_k \approx Q_{k+1} \begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} \tag{4.5.54}$$

Another problem is when T_k is singular during one iteration of the Lanczos Algorithm. For example, T_j will become singular when A is a symmetric tridiagonal matrix with zeros as its diagonals and all ones on it's subdiagonals. However, these problems can be solved by considering something other than LU without pivoting. In fact, there are works of Paige, Y. Saad extending the idea and derived algorithms that can solve a symmetric indefinite system without using the norm equation. **CITATION NEEDED**.

5 Effects of Floating Point Arithematic

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

5.1 Partial Orthogonalization and Full Orthogonalization

In this section, we use some of the results and insight obtained from the derivation of the CG algorithm to develop a theoretical fix for the loss of precision under floating point arithmetic for the CG algorithm. The floating errors inside of the CG algorithm is manifested as a loss of orthogonality and loss of conjugacy for the vectors r_k, p_k. To fix this, we simply use the CDM algorithm's projector to re-orthogonalize the conjugate vectors and the residual vectors so the A-Orhtogonality and Orthogonality are both preserved for the r_k, p_k vectors. Such idea is not new and it's stated in the original paper by Hestenes and Stiefel back in 1952; here we present the second more computationally expensive idea in the paper by Hestenes and Stieful, but using the quantities we derived in the first section. Firstly recall the proof for **proposition 3.7**(ADD HYPERREF). Next, we inductively consider the case where the newest residual vector involving round off error $\overline{r}_{k+1}$ are given and it breaks the orhogonality 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 conjuate vectors p_k correctly by faithfully reproducing the term $(AP_k)^T \overline{r}_k$, and then we upadate the residual $\overline{r}_k$ into r_k by orthogonalizing it against all previous residual vectors. Doing ths require the 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 Points

Under 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 (2.2) of the discussion. However in practice, this is not always the case. Similar experiments are

conducted in Greenbaum's works (**CITATION NEEDED**). 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 a 16 digits floats to exagerate the floating point roud off error. The spectrum of the matrix we are using are taken to be the same from Greenbaum's works (**CITATION NEEDED**):

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

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 evenlly distributed around the interval $[\lambda_{\min}, \lambda_{\max}]$. The Chebyshev bound is no longer a tight bound because the distribution of

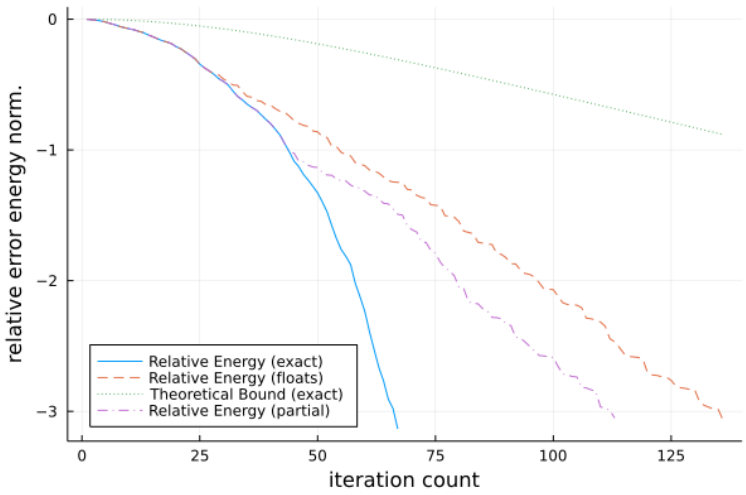

Figure 1: The relative energy norm error for different methods. Blue solid line: The exact conjugate gradient convergence. Purple dotdashed: The conjugate gradient that is partially orthgonalized with previous 32 residual and conjugate vectors. organed dashed: the Original conjugate gradient. Green dot: The tighter theretical upper bound derived by chebyshev (4.3.20). For this plot: $\rho = 0.9$, the matrix is 256×256.

the eigenvalues are not perfectly uniform. The partially orthogonalized methods diverges from the exact error after more steps of iterations compare to the relative error without any orthogonalizations. These are seemed in (fig 1).

Remark 5.2.1. The convergence is disappointing under floating point arithematic and the promised efficiency of the algorithm is not there anymore if the matrix is not neccessarily ill-conditioned. If we were to use a fully re-orthogonalized variant of the CG, the memory needed and the total complexity for a fixed tolerance is going to be on the same scale of solving it by direct methods.

5.3 Paige's Floating Point Analysis

In this section we present the backwards analysis of the algorithm in a more thoroughly manner and exam its consequences. When floating point arithematic is used, the eigenvalues of the Tridiagonal matrices might inttroduces ghost eigenvectors for the Lanczos Iterations, and using the equivalence of the Lanczos iterations and CG, we can capture a posteri bound on how much the error is exactly during the iterations of CG.

5.3.1 Bounding the The Relative Residuals

Recall from proposition Propposition 4.6 that the residual of the CG can be expressed interms of the Lanczos vectors. However, the Lanczos Algorithm 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 Algorithm will be the truncation of the tridiagonal parts of the matrix $\tilde{Q}_k^H A\tilde{Q}_k$. However when we solve for y_k using the expression $y_k = \beta T_k^{-1}\xi_1$, and this is what the CG algorithm does faithfully. The algorithm still thought T_k produced by itself is perfectly tridiagonal which is not true because Q_k is not quiet orthogonal. As a results the algorithm never quiet find the optimal under the conjugate basis. At first glance, tiny round-off errors in the Lanczos vectors is very problematic.

Surprisingly, we can leverage what CG does and assume: $y_k = \beta T^{-1}\xi_1$ is at least, exact. Then it only left us with fewer type of floating errors to keep track of. Next, we proceed to look for the residual of the CG algorithm by assuming that the lanczos recurrences has a backwards errors:

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

Reader please reflect on the fact that, right now the Q_k which is EXACT and T_k is produced by the Lanczos Algorithm, and we are fixing the recurrences with F_k, a matrix full of floating error to correct it so that the equality holds true.

$$r_k = r_0 - AQ_k y_k \tag{5.3.2}$$

$$r_k = r_0 - \left(Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} + F_k\right) y_k \tag{5.3.3}$$

$$r_k = \underbrace{\left(r_0 - Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} y_k\right)}_{=-\beta_k \xi_k^T y_k q_{k+1}} + F_k \beta T_k^{-1}\xi_1 \tag{5.3.4}$$

$$\implies \frac{\|r_{k+1}\|}{\|r_0\|} \leq \beta_k \|\xi_k^T T_k^{-1}\xi_1 q_{k+1}\| + \|F_k T_k^{-1}\xi_1\| \tag{5.3.5}$$

$$\frac{\|r_{k+1}\|}{\|r_0\|} \leq \beta_k |\xi_k^T T_k^{-1}\xi_1| + \|F_k\|\|T_k^{-1}\xi_1\| \tag{5.3.6}$$

Here, because of the assumption of exactness for the Q_k matrices we are able to reuse Propposition 4.6 to regaudge it and obtain a similar expression but this time, taking the floating error into account. In fact, this formula is elegant in the sense that it's determined by the scalar $\xi_k^T T_k^{-1}\xi_1$ and the floating point error F_k produced by the Lanczos iterations.

Remark 5.3.1. Here we present the core idea behind Greenbaum's analysis for the convergence rate of Conjugate Gradient under floating arithematic. The idea is that the bound for the exact arithemtic, CG convergence rate can be applied here for the term: $\beta_k |\xi_k^T T_k^{-1}\xi_1|$.

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 $\overline{r}_k$ 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 excitment, we already have the bound for $\overline{r}_{k+1}$. which eventually, proving that the CG algorithm is backwards stable.

5.3.2 Paige's Theorem and Backwards Stability

Now, we introduce a new theorem poposed by Paige (**CITATION NEEDED**). Which gives a bound to the floating point errors for the Lanczos Algorithm, and consequently, giving us a bound on the relative error for CG under floating points. The theorem goes as follow:

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

$$\lambda_1 - j^{5/2}\epsilon_2\|A\| \leq \theta_i^{(j)} \leq \lambda_n + j^{5/2}\epsilon_2\|A\| \tag{5.3.7}$$

$$\epsilon_2 := \sqrt{2}\max\{6\epsilon_0, \epsilon_1\} \tag{5.3.8}$$

Along with this theorem, the following quantities are also defined and cited in Greenbaum's work (**CITATION NEEDED**) from the work of Paige.

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

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

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

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

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

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

The quantity k is the current iterations number of the Lanczos Iterations, $j \leq k$. m is the maximum number of non-zero element in the matrix A. Using Paige's theroem, we can bound the conditions 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 κ is the bound of the condition number of the T_k matrix, the tridiagonal matrix produced by the finite precision Lanczos.

Proof. Using the lemma 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 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 becaus $Ae_k = r_k$ by definition. Subtituting the quantity $\kappa(T_k)$, the conditions 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 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, considering:

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

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

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

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

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

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

Which completes the proof for the upper bound on the convergence rate for the Conjugate Gradient Method.

Remark 5.3.2. This proof showed that the Conjugate Gradient method for T_{k+1} that is non-singular then it will be backwards stable. As long as the value of β_k is non zero, and the eigenvalues of T_{k+1} is not containing zero, the conjugate gradient method will continue to converge in the future iterations. So in laymen's term, it doesn't matter if the roudoff 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 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 parallelizism, or floating point specific summation instructons, which reduces the relative sizes for the sumees, hence reducing the total roudoff error accumulations.

5.3.3 Ghost Eigenvalues

The name Ghost Eigenvalues refers to the phenomena where the Lanczos Algorithm seems to find eigenvalues that are extremely close to each other, when in fact, the extremely closed eigenvalues are a single eigenvalues. More specifically, the eigenvalues in T_k seems to cluster gruops of eigenvalues that are extemely close to the true eigenvalues of the original matrix A. We know for a fact that the tridiagonal matrix produced via Lanczos can't have any repeated eigenvalues (**NONTRIVIAL FACTS**). This phenomena is in fact, due to the limited floating point accuracy of the Lancozs Orthogonalization process. Here, we conducted the numerical experiments and carefully reproduce the phenomena for a diagonal matrix A with diagonals given by the formula: $\lambda_i = \left(-1 + \frac{2(i-1)}{(n-1)}\right)^3$ where $A \in \mathbb{R}^{n \times n}$. For this experiment, we set $n = 64$ and we use Float64. We run the Lanczos Iterations and mark the smallest and largest 10 eigenvalues during the iterations and plotted out their trajectories during 64 iterations. The results can be seemed at (fig 2).

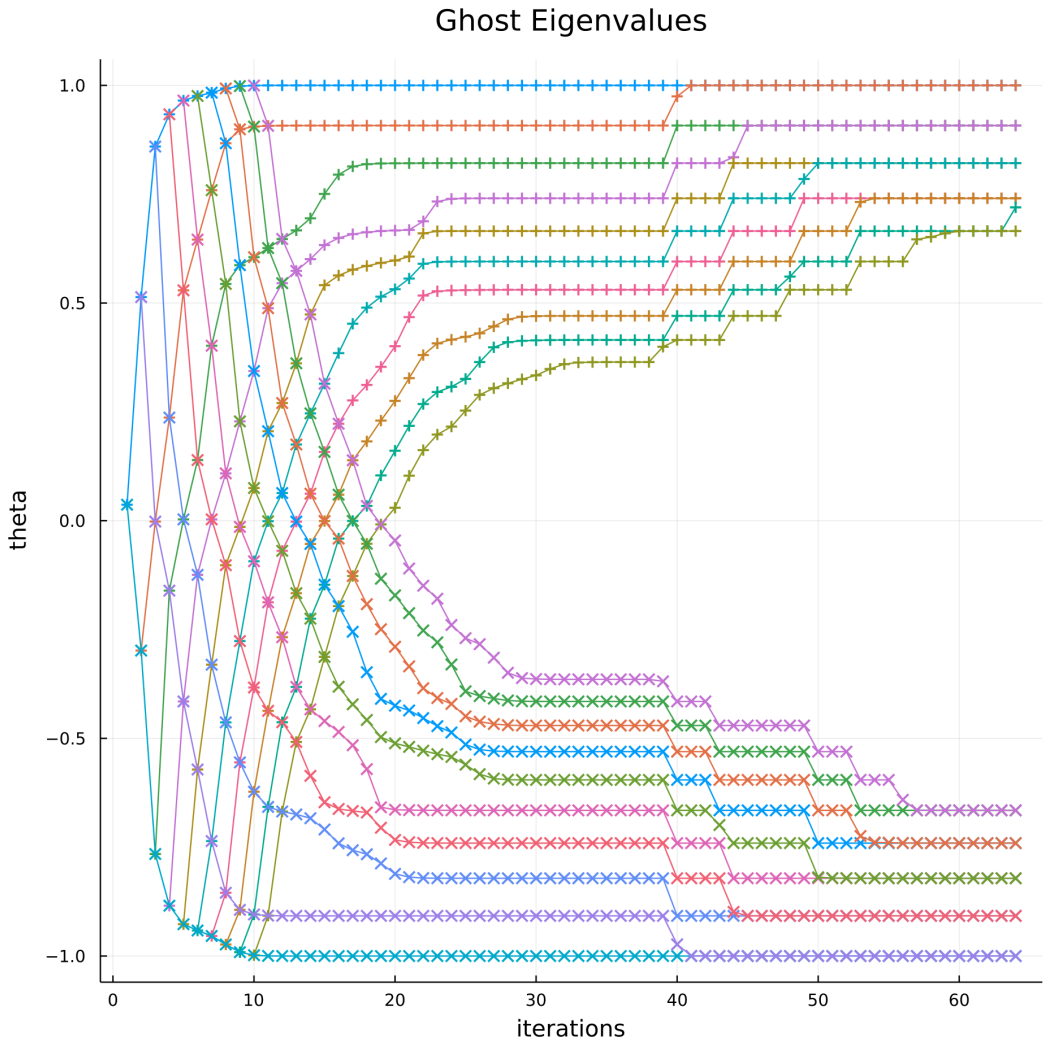

Figure 2: The highest and lowest 10 eigenvalues of the matrix T_k during the Lanczos Iterations are being tracked by their ranking, and plotted with their trajectories.

Recall from the Cauchy Interlace Theorem, the eigenvalues of the Tridiagonal Matrix T_k from iterations to iterations will never crosses each other, which implies that, the $\theta_i^{(k)}$, the i th eigenvalues during the k th iterations will move monotonically moves upwards or downwards during the Lanczos Iterations. The ghoest eigenvalues on the figure heppens when some of the interior eigenvalues suddenly switched to another eigenvalue that is on the exterior of the spectrum. It appears as the matrix T_k has repeated eigenvalues but in fact they are not and they are just very close.

However, judging from the eigenvalues of the matrix T_k alone will not tells us whether 2 very close eigenvalues corresponds to the same eigenvalue of A, even if their trajectories seems to suggest it that way. We can't tell it because if I keep the matrix T_k generated from an lanczos algorithm and all it A, which has eigenvalues that are extremely close to each other, and then ran another Lanczos iteations on it but with the initial vector ξ_1, then I will exactly reproduce the lanczos matrix vector itself. But in this case, the eigenvalues of T_k is exactly the same as A, and 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 is 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 intuitive, we bear in mind the fact that ghost eigenvalues happens together with the lost of orhogonality of the Q_k matrix. If Q_k matrix is perfectly orthogonal, then there is no ghost eigenvalues, regardless of how the trajectories of the eigevalues of T_k looks like. In fact, a corresponding plot of $\tilde{Q}_k^H \tilde{Q}_k$ are plotted in (fig 3) where we plotted the heat map of the matrix $\tilde{Q}_k^H \tilde{Q}_k$ under the $\log 2$ scale of the absolute values of its entries.

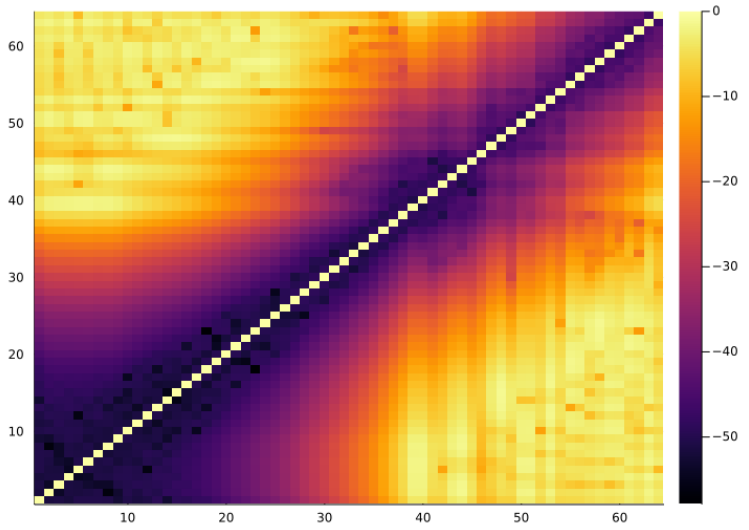

Figure 3: The heatmap of the plot of log2 of the absolute values of the matrix $\tilde{Q}_k^H \tilde{Q}_k$.

In addition to the lost orthogonality of the matrix Q_k, we also visualized the actual tridiagonal matrix reproduced by $\tilde{Q}_k A \tilde{Q}_k$ which is plotted in (fig 4). 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 an extremely bad conditions number. What is important is the blob of non-zero entries on the top left and the bottom right of the plot, which looks similar to how the $\tilde{Q}_k^H \tilde{Q}_k$. And this is a hint that the lost of orthogonality will create extra errors on the off tridiagonal parts of the matrix T_k.

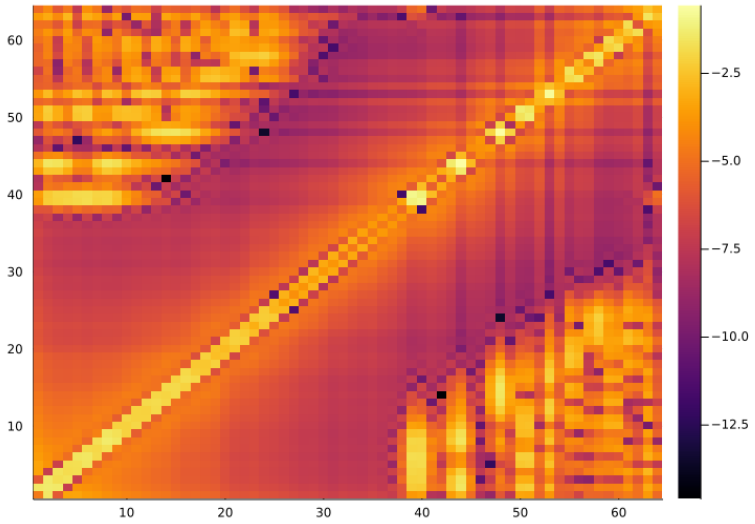

Figure 4: The plot of the T_k matrix as reproduced by $\tilde{Q}_k A \tilde{Q}_k$

Remark 5.3.3. As a final remark for these numerical experiments, I sugguest 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 (Manifested as the usually stable trajectories of eigenvalues on the exerior of the spectrum for the matrix T_k in (fig 2)), and when it happens, the Lanczos vectors produced by the algorithm has lost its orthogonality corresondingly, which then causes the interior eigenvalues of T_k to shift, creating ghost eigenvalues for the matrix T_k.

5.4 The Rizt Vector's View and Another Paige's Theorem

In this section, we prove another analysis of the Lanczos Iterations from Dammel's work(**Citation Needed**) where he introduced proof of another Paige's theorem that shows exactly how to measure the lost of orthogonality of the Lanczos vectors while Lanczos algorithm is running. Here, we wish to prove it in much more details with more thoroughness.

The theorem highlights 2 important facts. The first is the the lost of orthogonality of Lanczos Vector and Ghost eigenvalues appears at the same time and they are sysmatic. The second is that the projection of the Lanczos vector onto the converged ritze vectors can be numerically attained, which tells us how much lost of orthogonality is occuring 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)**.**

5.5 Forward Error Analysis

6 Appendix

6.1 Useful Lemmas

Lemma 6.1.1 (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 incuded 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 $A^{1/2}A^{1/2}$: □

$$\lambda_1\|x\| \leq \|Ax\| \leq \lambda_2\|x\| \tag{6.1.1}$$

$$\sqrt{\lambda_1}\|x\| \leq \|A^{1/2}x\| \leq \sqrt{\lambda_n}\|x\| \tag{6.1.2}$$

$$\implies \sqrt{\lambda_1} \leq \frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n} \tag{6.1.3}$$

Consider another vector y:

$$\sqrt{\lambda_1} \leq \frac{\|Ay\|}{\|A^{1/2}y\|} \leq \sqrt{\lambda_n} \tag{6.1.4}$$

Combining the two we have:

$$\sqrt{\lambda_1}\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n}\sqrt{\lambda_1} \tag{6.1.5}$$

$$\sqrt{\lambda_1}\sqrt{\lambda_n} \geq \sqrt{\lambda_n}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{6.1.6}$$

$$\implies \sqrt{\lambda_1}\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{6.1.7}$$

$$\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\kappa(A)}\frac{\|Ay\|}{\|A^{1/2}y\|} \tag{6.1.8}$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)}\frac{\|A^{1/2}x\|}{\|A^{1/2}y\|} \tag{6.1.9}$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)}\frac{\|x\|_A}{\|y\|_A} \tag{6.1.10}$$

6.2 Theorems, Propositions

Proposition 6.1 (Krylov Subspace Grade Invariant Theorem)**.** Once the subspace becomes linear dependent, the subspace becomes invariant.

$$K_k = \begin{bmatrix} b & AB & \cdots & A^{k-1}b \end{bmatrix} \tag{6.2.1}$$

$$K_k \text{ Lin Dep} \implies A^{k-1}b = AK_{k-1}w \tag{6.2.2}$$

$$\implies AK_k = K_k \underbrace{\begin{bmatrix} e_2 & \cdots & e_k & c_k \end{bmatrix}}_{:=C_k} \tag{6.2.3}$$

$$\implies A^2K_k = AK_kC_k = K_kC_k^2 \tag{6.2.4}$$

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