

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 analyze 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 Conjugate Gradient and Lanczos Algorithm and using the relations to reveal inspirations in the area of symmetric indefinite solvers and convergence rate analysis. We then analyze algorithms behaviors under floating points arithmetic, specifically how the floating point arithmetic slows down the convergence of CG and how it's related to the loss 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	4
2	Introduction	4
3	Foundations	6
3.1	Projectors	6
3.1.1	Orthogonal Projector	6
3.1.2	Oblique Projector	7
3.1.3	Projector Geometric Intuitions	8
3.1.4	Projector as a 2-Norm Minimizer	8
3.2	Subspace Projection Methods	8
3.2.1	Prototype Algorithm	9
3.2.2	Energy Norm Minimization using Gradient	9
3.3	Krylov Subspace	11
3.3.1	The Grade of a Krylov Subspace	11
3.3.2	The Grade and Matrix Polynomial	13
3.4	Useful Theorems and Mathematical Entities	13
3.4.1	Minimal Polynomial of a Matrix	13
3.4.2	Cauchy Interlace Theorem	14
3.4.3	Caley Hamilton's Theorem	14
3.4.4	The Chebyshev Polynomial	14
3.5	Deriving Conjugate Gradient from First Principles	15
3.5.1	CG Objective and Framework	15
3.5.2	Using the Projector	16
3.5.3	Assisted Conjugate Gradient (Conjugate Directions)	17
3.5.4	Properties of Assisted Conjugate Gradient	18
3.5.5	Residual Assisted Conjugate Gradient	19
3.5.6	RACG and Krylov Subspace	22
3.6	Arnoldi Iterations and Lanczos	22
3.6.1	The Arnoldi Iterations	23
3.6.2	Arnoldi Produces Orthogonal Basis for Krylov Subspace	24
3.6.3	The Lanczos Iterations	24
4	Analysis of Conjugate Gradient and Lanczos Iterations	26
4.1	Conjugate Gradient and Matrix Polynomial	26
4.2	Termination Conditions of RACG	28
4.3	Convergence Rate of RACG under Exact Arithmetic	29
4.3.1	Uniformly Distributed Eigenvalues	29
4.3.2	One Outlier Eigenvalue	31
4.4	From Conjugate Gradient to Lanczos	33
4.4.1	The Base Case	34
4.4.2	The Inductive Case	35
4.4.3	Fixing the Sign	36
4.5	From Lanczos to Conjugate Gradient	37

4.5.1	Matching the Residual and Conjugate Vectors	37
4.5.2	Matching the a_k, b_k in CG	39
5	Effects of Floating Point Arithmetic	42
5.1	Partial Orthogonalization and Full Orthogonalization	42
5.2	Relative Errors of CG Under Floating Points	43
5.3	Greenbaum's Backward Analysis and Paige's Floating Point Analysis	44
5.3.1	Bounding the The Relative Residuals	44
5.3.2	Paige's Theorem and Backwards Stability	45
5.3.3	Ghost Eigenvalues	47
5.4	The Ritz Vector's View and Another Paige's Theorem	50
5.5	Forward Error Analysis	50

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 j th 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: $j = 0, 1, \dots, k-1, k$
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 $\hat{\cdot}$ decorator is reserved for denoting the unit vector of some non zero vector. For example $\hat{x} := x/\|x\|, x \neq \mathbf{0}$.

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

This section focuses on important mathematical entities that are important for formulating, analyzing the Conjugate Gradient and the Lanczos Algorithm. Major parts of this section cited from... (CITATION NEEDED)

3.1 Projectors

The properties of projector have importance for subspace projection method. In this section, we go through 2 types of projectors, the orthogonal projector and the oblique projector. The oblique projector is made useful for the derivation of the classic CG algorithm, which is referred to as RACG in the context of this paper. The orthogonal projector is useful for orthogonalizing the Krylov Subspace.

Definition 1. A matrix P is a projector when:

$$P^2 = P \quad (3.1.1)$$

This property is sometimes referred as *idempotent*. As a consequence, $\text{ran}(I - P) = \text{null}(P)$ and here is the proof:

Proof.

$$\forall x \in \mathbb{C}^n : P(I - P)x = \mathbf{0} \implies \text{ran}(I - P) \subseteq \text{null}(P) \quad (3.1.2)$$

$$\forall x \in \text{null}(P) : Px = \mathbf{0} \implies (I - P)x = x \implies x \in \text{ran}(I - P) \quad (3.1.3)$$

$$\implies \text{ran}(I - P) = \text{null}(P) \quad (3.1.4)$$

□

3.1.1 Orthogonal Projector

Definition 2. An orthogonal projector is a projector P which:

$$\text{null}(P) \perp \text{ran}(P) \quad (3.1.5)$$

This property is in fact, very special. A good example of an orthogonal projector would be the Householder Reflector Matrix. Or just any $\hat{u}\hat{u}^H$ where \hat{u} is a unitary vector. For convenience of proving, assume subspace $M = \text{ran}(P)$. Consider the following lemma:

Lemma 3.1.1.

$$\text{null}(P^H) = \text{ran}(P)^\perp \quad (3.1.6)$$

$$\text{null}(P) = \text{ran}(P^H)^\perp \quad (3.1.7)$$

Using (3.1.4) and consider the proof:

Proof.

$$\text{Observe : } \langle P^H x, y \rangle = \langle x, Py \rangle \quad (3.1.8)$$

$$\implies \forall x \in \text{null}(P^H), y \in \mathbb{C}^n : \langle P^H x, y \rangle = 0 = \langle x, Py \rangle \quad (3.1.9)$$

$$\implies \text{null}(P^H) \perp \text{ran}(P) \quad (3.1.10)$$

$$\forall y \in \text{null}(P), x \in \mathbb{C}^n : \langle x, Py \rangle = 0 = \langle P^H x, y \rangle \quad (3.1.11)$$

$$\implies \text{ran}(P^H) \perp \text{null}(P) \quad (3.1.12)$$

□

Proposition 3.1. A projector is orthogonal iff it's Hermitian.

Proof. \Leftarrow Assuming the matrix is Hermitian and it's a projector, then we wish to prove that it's an orthogonal projector. Let's recall the results of previous lemma:

$$\text{null}(P^H) = \text{ran}(P)^\perp \quad (3.1.13)$$

$$\text{null}(P) = \text{ran}(P^H)^\perp \quad (3.1.14)$$

Substituting $P^H = P$, we have $\text{null}(P) = \text{ran}(P)^\perp$, Which is the definition of Orthogonal Projector. Therefore, P is an orthogonal projector by the definition of the projector.

For the \implies direction, we assume that P is an Orthogonal Projector, then we wish to show that it's also Hermitian. Observe that P^H is also a projector because $(P^H)^2 = (P^2)^H$. Then, using the definition of orthogonal projector:

$$\text{null}(P) \perp \text{ran}(P) \quad (3.1.15)$$

$$\text{null}(P^H) \perp \text{ran}(P^H) \quad (3.1.16)$$

Notice that using above statement together with Lemma 1 means $\text{null}(P) = \text{ran}(P)^\perp = \text{null}(P^H)$, and then $\text{ran}(P) = \text{null}(P)^\perp = \text{ran}(P^H)$. Therefore, P^H is an projector such that: $\text{ran}(P) = \text{ran}(P^H) \wedge \text{null}(P) = \text{null}(P^H)$. The range and null space of P^H and P is the same therefore P has to be Hermitian. □

3.1.2 Oblique Projector

An oblique projector a projector but not an orthogonal projector, and vice versa. It's a projector that satisfies the following conditions:

Definition 3.

$$Px \in M \quad (I - P)x \perp L \quad \text{where: } M \neq L \quad (3.1.17)$$

An orthogonal projector is the case when the subspace $M = L$.

A famous example of an orthogonal projector is QQ^H where Q is an Unitary Matrix. This is a Hermitian Matrix and it's idempotent, making it an orthogonal projector.

3.1.3 Projector Geometric Intuitions

A projector describes a given vector using some elements from another basis. The oblique projector creates a light sources in the form of the subspace L and it shoots parallel light rays in the direction orthogonal to L , crossing vectors and projecting their shadow onto subspace M .

3.1.4 Projector as a 2-Norm Minimizer

An orthogonal projector always reduce the 2 norm of a vector. Given any subspace M , we can create a basis of vectors packing into the matrix A , then P_M as a projector onto the basis M as an example can be: $A(AA^T)^{-1}A^T$. What exactly the projector is doesn't matter in the future disussions. Let's consider the claim:

$$\|P_M x\|^2 \leq \|x\|^2 \quad (3.1.18)$$

Proof. For notational convenience, we simply denotes P_M using P .

$$x = Px + (I - P)x \quad (3.1.19)$$

$$\|x\|^2 = \|Px\|^2 + \|(I - P)x\|^2 \quad (3.1.20)$$

$$\|x\|^2 \geq \|Px\|^2 \quad (3.1.21)$$

□

Using this property of the Orthogonal Projector, we consider the following minimizations problem:

$$\min_{x \in M} \|y - x\|_2^2 = \|y - Py\|_2^2 \quad (3.1.22)$$

Proof:

$$\|y - x\|_2^2 = \|y - Py + Py - x\|_2^2 \quad (3.1.23)$$

$$\|y - x\|_2^2 = \|y - Py\|_2^2 + \|Py - x\|_2^2 \quad (3.1.24)$$

$$\implies \|y - Py\|_2^2 \leq \|y - x\|_2^2 \quad (3.1.25)$$

That concludes the proof. Observe that, $y - Py \perp M$ and $Py - x \in M$ because $Py, x \in M$, which allows us to split the norm of $y - x$ into 2 components. In addition using the fact that the projector is orthogonal. That concludes the proof.

3.2 Subspace Projection Methods

Let \mathcal{K}, \mathcal{L} be subspaces where candidates soltuions are chosen and residuals are orthogonalized against. Under the idea case the 2 subspaces spans all dimensions, and it's able to approximate all solutions and forcing the residual vector $(b - Ax)$ to be zero. This is a description of this framework:

$$\tilde{x} \in x_0 + \mathcal{K} \text{ s.t. } b - A\tilde{x} \perp \mathcal{L} \quad (3.2.1)$$

it looks for an x in the affine linear subspace \mathcal{K} such that it's perpendicular to the subspace \mathcal{L} , or, equivalently, minimizing the projection onto the subspace \mathcal{L} . One interpretation of it is an projection of residual onto the basis that is orthogonal to \mathcal{L} .

Sometimes, for convenience and the exposition and exposing hidden connections between ideas, the above conditions can be expressed using matrix.

$$\text{Let } V \in \mathbb{C}^{n \times m} \text{ be a basis for: } \mathcal{K} \quad (3.2.2)$$

$$\text{Let } W \in \mathbb{C}^{n \times m} \text{ be a basis for: } \mathcal{L} \quad (3.2.3)$$

We can then make use of (1.3.1) and express it in the form of:

$$\tilde{x} = x^{(0)} + Vy \quad (3.2.4)$$

$$b - A\tilde{x} \perp \text{ran}(W) \quad (3.2.5)$$

$$W^T(b - A\tilde{x} - AVy) = \mathbf{0} \quad (3.2.6)$$

$$W^T r^{(0)} - W^T AVy = \mathbf{0} \quad (3.2.7)$$

$$W^T AVy = W^T r^{(0)} \quad (3.2.8)$$

3.2.1 Prototype Algorithm

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

While not converging :

Increase Span for: \mathcal{K}, \mathcal{L}

Choose: V, W for \mathcal{K}, \mathcal{L}

$$r := b - Ax \quad (3.2.9)$$

$$y := (W^T AV)^{-1} W^T r$$

$$x := x + Vy$$

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.

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.

$$\min_{x \in x_0 + \mathcal{K}} \|b - Ax\|_B^2 \quad (3.2.10)$$

$$= \min_{w \in \mathbb{R}^k} \|b - A(x_0 + Kw)\|_B^2 \quad (3.2.11)$$

$$= \min_{w \in \mathbb{R}^k} \|r_0 - AKw\|_B^2 \quad (3.2.12)$$

We take the derivative of it and set the derivative to zero. We skip the proof that the derivative of $\nabla_x [\frac{1}{2}\|x\|_A^2] = Ax$, and for a crash course on derivative, the $\nabla_x[f(Ax)] = A^T\nabla[f(x)]$.

$$\nabla_w [\|r_0 - AKx\|_B^2] = \mathbf{0} \quad (3.2.13)$$

$$(AK)^T B(r_0 - AKx) = \mathbf{0} \quad (3.2.14)$$

$$(AK)^T Br_0 - (AK)^T BAKx = \mathbf{0} \quad (3.2.15)$$

$$(AK)^T Br_0 = (AK)^T BAKx \quad (3.2.16)$$

The above formulation is tremendously powerful. I used gradient instead of projector for the simplicity of the argument. One can derive the same using orthogonal projector to minimize 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 alternative. Let's this be a proposition that we proceed to prove.

Proposition 3.2 (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 $b - Ax_k$.

$$\min_{x_k \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 \iff K^T B e_k - K^T B K w = \mathbf{0} \quad (3.2.17)$$

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

Proof.

$$\min_{x \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 = \min_{x \in \mathbb{R}} \|A^{-1}b - x_0 - Kw\|_B^2 \quad (3.2.18)$$

$$= \min_{x \in \mathbb{R}^k} \|e_k - Kw\|_B^2 \quad (3.2.19)$$

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

$$\mathbf{0} = \nabla_w [\|e_k - Kw\|_B^2] \quad (3.2.20)$$

$$= \nabla_w [e_k - Kw]^T B (e_k - Kw) \quad (3.2.21)$$

$$= 2K^T B (e_k - Kw) \quad (3.2.22)$$

$$\implies K^T B e_k - K^T B K w = \mathbf{0} \quad (3.2.23)$$

□

This conditions is implicitly describing the objective of a Preconditioned Conjugate Gradient algorithm, where B is the M^{-1} matrix, 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:

$$K^T A A^{-1} r_k - K^T A K w = \mathbf{0} \quad (3.2.24)$$

$$K^T r_k - K^T A K w = \mathbf{0} \quad (3.2.25)$$

Here, we just 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 Krylov Subspace

A Krylov Subspace is a sequence of bases parameterized by A , the linear operator. v is an initial vector. For each basis in the sequence, it's gotten by applying the linear operator to the previous basis in the sequence. In this part we prove some of the properties of Krylov Subspace. The term "grade" we used is originated from the textbook (**CITATION NEEDED**) from Y.Saad.

Definition 4 (Krylov subspace).

$$\mathcal{K}_k(A|b) = \text{span}(b, Ab, A^2b, \dots, A^{k-1}b) \quad (3.3.1)$$

Please immediately observe that from the definition we have:

$$\forall v : \mathcal{K}_1(A|v) \subseteq \mathcal{K}_2(A|v) \subseteq \mathcal{K}_3(A|v) \dots \quad (3.3.2)$$

Please also observe that, every element inside of krylov subspace generated by matrix A , and an initial vector v can be represented as a polynomial of matrix A multiplied by the vector v and vice versa.

$$\forall x \in \mathcal{K}_k(A|v) \exists w : p_k(A|w)v = x \quad (3.3.3)$$

We use $p_k(A|w)$ to denote a matrix polynomial with coefficients $w \in \mathcal{K}_j$, where w is a vector. No proof this is trivial. Take note that, we can change the field of where the scalar w_i is coming from, but for discussion below, \mathbb{R}, \mathbb{C} doesn't matter and won't change the results so we stick to \mathbb{R} and we let v, A be real vectors and matrices so it's consistent.

$$p_k(A|w)v = \sum_{j=0}^{k-1} w_j A^j v \quad (3.3.4)$$

3.3.1 The Grade of a Krylov Subspace

The most important property of the subspace is the idea of grade denoted as $\text{grade}(A|v)$, indicating when the Krylov Subspace of A wrt to v becomes invariant when the grade of the subspace is reached and it kept its invariance for all subsequent subspaces. To show this idea, we consider the following 3 statements about Krylov Subspace which we will proceed to prove.

Lemma 3.3.1 (Grade Lemma 1).

$$\exists 1 \leq k \leq m+1 : \mathcal{K}_k(A|v) = \mathcal{K}_{k+1}(A|v) \quad (3.3.5)$$

There exists a natural number between 1 and $m+1$ such that, the successive krylov subspace span the same space as the previous one.

Lemma 3.3.2 (Grade Lemma 2).

$$\exists! k \text{ s.t. } \mathcal{K}_k(A|v) = \mathcal{K}_{k+1}(A|v) \implies \mathcal{K}_k(A|v) \text{ is Lin Ind} \wedge \mathcal{K}_{k+1}(A|v) \text{ is Lin Dep.} \quad (3.3.6)$$

There exists a minimum such k that is unique where the immediate next krylov subspace is linear dependent, and $k-1$ would be the grade.

Lemma 3.3.3 (Grade Lemma 3).

$$\mathcal{K}_k(A|v) \text{ Lin Dep} \implies \mathcal{K}_{k+1}(A|v) = \mathcal{K}_k(A|v) \quad (3.3.7)$$

if the k Krylov Subspace is linear dependent, then all successive Krylov Subspace is the same.

Theorem 1 (Unique Existence of Grade for a Krylov Subspace). Let k be the minimum number when the krylov subspace becomes linear dependent, then all successive krylov subspace span the same space. $\mathcal{K}_k(A|v) = \mathcal{K}_{k+j}(A|v) \forall j \geq 0$. The number k is regarded as the grade of krylov subspace wrt to v denoted using $\text{grade}(A|v)$.

Lemma 1, 2 ensures that there exists a term in the sequence of krylov subspace becomes linear dependence, and when that happens all subsequent Krylov Subspace will span the same subspace, this is by Lemma 3. As a results, the grade for the Krylov Subspace exists and it's unique.

Next, let's consider the proof of theorem 1 by proving all 3 of the lemmas.

Krylov Grade Lemma 1. For notational simplicity, \mathcal{K}_k now denotes $\mathcal{K}_k(A|v)$. Let's start the considerations from the definition of the Krylov Subspace:

$$\forall k : \mathcal{K}_k \subseteq \mathcal{K}_{k+1} \implies \dim(\mathcal{K}_k) \leq \dim(\mathcal{K}_{k+1}) \quad (3.3.8)$$

$$\mathcal{K}_{k+1} \setminus \mathcal{K}_k = \text{span}(A^k v) \quad (3.3.9)$$

$$\implies \dim(\mathcal{K}_{k+1}) - \dim(\mathcal{K}_k) \leq 1 \quad (3.3.10)$$

Therefore, the dimension of the successive krylov subspace forms a sequence of positive integer that is monotonically increasing. By the Cayley's Hamilton's theorem (will be stated later), the sequence is bounded by m , since there are $m + 1$ terms, it must be the case that at least 2 of the krylov subspace has the same dimension (And the earliest such occurrence will exist), implying the the fact that the new added vector from k to $k + 1$ is in the span of the previous subspace. \square

Krylov Grade Lemma 3. The direction $\mathcal{K}_k(A|v) \subseteq \mathcal{K}_{k+1}(A|v)$ is trivial.

Assuming that $\mathcal{K}_{k+1}(A|v)$ is linear dependence, we wish to prove that $\mathcal{K}_{k+1}(A|v) \subseteq \mathcal{K}_k(A|v)$ by considering:

$\mathcal{K}_{k+1}(A|v)$ is Lin dependent

$$\implies \exists w^{(k)} : A^k v = p_k(A|w^{(k)})v$$

$$x \in \mathcal{K}_{k+1}(A|v) \iff \exists w^{(k+1)} : p_{k+1}(A|w^{(k+1)})v = x$$

$$x = w^{(k+1)} A^k v + \sum_{j=0}^{k-1} w_j^{(k+1)} A^j v$$

$$x = w_k^{(k+1)} p_k(A|w^{(k)})v + \sum_{j=0}^{k-1} w_j^{(k+1)} A^j v$$

$$x \in \mathcal{K}_k(A|v)$$

For notations, we used $w^{(k)}, w^{(k+1)}$ to represents the vector containing all coefficients for the polynomial and their i element is denoted as $w_i^{(k)}$. From the last line, we proved that for all

x in $\mathcal{K}_{k+1}(A|v)$, it's must also be in $\mathcal{K}(A|v)$. The frist line is using the fact that $\mathcal{K}_{k+1}(A|v)$ is linear dependent, giving us an polynomial for the term $A^k v$. The next line is saying that for any element in $\mathcal{K}_{k+1}(A|v)$ there exists a matrix polynomial representing x . Doingsome algebra, we reduced the polynomial of max degree k into degree $k - 1$, proving that x must also be in $\mathcal{K}_k(A|v)$. \square

Krylov Grade Lemma 2. The proof for Lemma 2 is direct from Lemma 1, 3. Lemma 2 asserts the existence of a unique minimum of k and such k makes $\mathcal{K}_{k-1}(A|v) = \mathcal{K}_k(A|v)$ and $\mathcal{K}_k(A|v)$ is linearly dependence. \square

3.3.2 The Grade and Matrix Polynomial

Theorem 2. Let k be the grade of Krylov Subspace A initialized with v , then exists $p_k(A|w)v = x$ for all x in the subspace $\mathcal{K}_k(A|v)$ with $w \neq \mathbf{0}$, and it must be the case that $w_0 \neq 0$.

Proof. For contradiction suppose otherwise that we can represents $x \in \mathcal{K}_k(A|v)$ and such a polynomial with w_0 exists then:

$$\exists w \neq \mathbf{0} : p_k(A|w)v = \mathbf{0} \quad (3.3.11)$$

$$\implies w_0 v + \sum_{j=1}^{k-1} w_j A^j v = \mathbf{0} \quad (3.3.12)$$

$$\mathbf{0} = \sum_{j=1}^{k-1} w_j A^j v \quad (3.3.13)$$

$$\mathbf{0} = A \sum_{j=0}^{k-2} w_{j+1} A^j v \quad (3.3.14)$$

$$\implies \sum_{j=0}^{k-2} w_{j+1} A^j v = \mathbf{0} \quad (3.3.15)$$

From the second line to the third, I susbstitute $w_0 = 0$ for contradiction. On the last line, it suggested that k is not the smallest, and $k - 1$ might be the grade, contradicting the assumption that k is the grade of the Krylov Subspace. Therefore, $w_0 \neq 0$. \square

3.4 Useful Theorems and Mathematical Entities

3.4.1 Minimal Polynomial of a Matrix

Definition 5. A minimal polynomial $p_k(x)$ is monic such that $p_k(A) = \mathbf{0}$ and k is as small as possible.

One immediate property that might be useful for future constext is the fact that the constant term of the Minimal Polynomial has to have a non-zero coefficient. For contradiction

suppose that is the not the case and $k - 1$ is the lowest degree of a minimal polynomial then:

$$\forall x : \mathbf{0} = \sum_{j=0}^{k-1} w_j A^j x \quad (3.4.1)$$

$$w_0 := 0 \quad (3.4.2)$$

$$\forall x : \mathbf{0} = \sum_{j=1}^{k-1} w_j A^j x \quad (3.4.3)$$

$$\forall x : \mathbf{0} = A \sum_{j=0}^{k-2} w_j A^j x \quad (3.4.4)$$

$$\implies \forall x : \mathbf{0} = \sum_{j=0}^{k-2} w_j A^j x \quad (3.4.5)$$

And we get another polynomial satisfying that conditions that has a degree of $k - 1$, contradicting the condition that k is the minimal such parameter.

3.4.2 Cauchy Interlace Theorem

Theorem 3 (Cauchy Interlace). The cauchy's Iterlace Theorem describes the relations of eigenvalues between the submatrix of a Symmetric Tridiagonal matrix and the bigger matrix. In our caes, let $T_{k-1} \in \mathbb{R}^{(k-1) \times (k-1)}$ be the principal sub-matrix to the matrix T_k , then between every eigenvalue of T_k , there must exists an eigenvalue of T_{k-1} between them. Let $\theta_i^{(k)}$ be the i eigenvalues of T_k , let the eigenvalues be sorted then:

$$\theta_j^{(k+1)} \leq \theta_j^{(k)} \leq \theta_{j+1}^{(k+1)} \quad \forall 1 \leq j \leq k - 1 \quad (3.4.6)$$

3.4.3 Caley Hamilton's Theorem

Theorem 4 (Caley's Hamilton Theorem). A matrix satisfies it's own characteristic equation, let $p(x)$ be the characteristic polynomial for the matrix A , then $p(A) = \mathbf{0}$.

The Calye's Hamilton's Theorem is important in the sense that, a direct consequence is the termination conditions for all krylov Subspace Methods (regardless of initial guess vectors). It's saying that all Krylov Subspace methods will terminates at step $n + 1$ at most, if n is the size of the operator. However, the more important fact is that, when it terminates, the solution x is a weighted sum of the Krylov Subspace vectors and the weights are related to the characterstic polynomial of the matrix.

3.4.4 The Chebyshev Polynomial

The chebyshev polynomial is an orthogonal polynomials under a weighted space; it has amazing properties, for our sake, we focuses the Type T Chebyshev Polynomial which has the property of minimizing the infintty norm over a closed interval:

$$T_k(x) = \arg \min_{p(x)} \{ \|p(x)\|_{\infty} : \forall x \in [-1, 1] \} \quad (3.4.7)$$

This theorem is later used as a pivotal tools for the analysis for floating point error for the conjugate gradient algorithm.

3.5 Deriving Conjugate Gradient from First Principles

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, numerous perspectives were explored. Three of the most important ideas are using Conjugate Directions, minimizing the energy norm of the error of the linear system and coming up with an update of the conjugate vectors using the residual vector at the current 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 Petrov Galerkin 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.

3.5.1 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 semi-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.

$$\min_{w \in \mathbb{R}^k} \|A^{-1}b - (x_0 + P_k w)\|_A^2 \iff P_k^T r_0 = P_k^T A P_k w \quad (3.5.1)$$

Refer back to [Energy Norm Minimization using Gradient](#) for how to obtain the above minimization objective. Using the matrix form for the Petrov Galerkin Conditions where W, V are both P_k , we reformulate the Norm Minimization conditions under the framework of Petrov Galerkin conditions:

$$\text{choose: } x \in x_0 + \text{ran}(P_k) \text{ s.t: } b - Ax \perp \text{ran}(P_k) \quad (3.5.2)$$

Take note that the link between a norm minimization and an equivalent subspace Orthogonality conditions don't guarantee to happen for other subspace projector 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*.

$$P_k^T A P_k = D_k \text{ where: } (D_k)_{i,i} = \langle p_{i-1}, A p_{i-1} \rangle \quad (3.5.3)$$

$$P_k r_0 = P_k^T A P_k w = D_k w \quad (3.5.4)$$

$$w = D_k^{-1} P_k^T r_0 \quad (3.5.5)$$

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 can derive the following expression for the solution at step k: x_k and the residual at step $r_k = b - A x_k$ for the algorithm:

$$\begin{cases} x_k = x_0 + P_k D_k^{-1} P_k^T r_0 \\ r_k = r_0 - A P_k D_k^{-1} P_k^T r_0 \\ P_k^T A P_k = D_k \end{cases} \quad (3.5.6)$$

Let this algorithm be the prototype.

3.5.2 Using the Projector

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

Proof.

$$A P_k D_k^{-1} P_k^T (A P_k D_k^{-1} P_k^T) = A P_k D_k^{-1} P_k^T A P_k D_k^{-1} P_k^T \quad (3.5.7)$$

$$= A P_k D_k^{-1} D_k D_k^{-1} P_k^T \quad (3.5.8)$$

$$= A P_k D_k^{-1} P_k^T \quad (3.5.9)$$

$$P_k D_k^{-1} P_k^T A (P_k D_k^{-1} P_k^T A) = P_k D_k^{-1} D_k D_k^{-1} P_k^T A \quad (3.5.10)$$

$$= P_k D_k^{-1} P_k^T A \quad (3.5.11)$$

□

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 $\bar{P}_k = P_k D_k^{-1} P_k^T$; then these 2 projectors are:

$$A P_k D_k^{-1} P_k^T = A \bar{P}_k \quad (3.5.12)$$

$$P_k D_k^{-1} P_k^T A = \bar{P}_k A \quad (3.5.13)$$

One immediate consequence is:

$$\text{ran}(I - A \bar{P}_k) \perp \text{ran}(P_k) \quad (3.5.14)$$

$$\text{ran}(I - \bar{P}_k A) \perp \text{ran}(A P_k) \quad (3.5.15)$$

Proof.

$$P_k^T(I - A\bar{P}_k) = P_k^T - P_k^T A\bar{P}_k \quad (3.5.16)$$

$$= P_k^T - D_k D_k^{-1} P_k^T \quad (3.5.17)$$

$$= \mathbf{0} \quad (3.5.18)$$

$$(AP_k)^T(I - \bar{P}_k A) = P_k^T A - P_k^T A\bar{P}_k A \quad (3.5.19)$$

$$= P_k^T A - P_k^T A P_k D_k^{-1} P_k^T A \quad (3.5.20)$$

$$= P_k^T A - P_k^T A \quad (3.5.21)$$

$$= \mathbf{0} \quad (3.5.22)$$

□

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

Proposition 3.3 (Residuals are Orthogonal to P_k).

$$r_k = r_0 - A\bar{P}_k r_0 = (I - A\bar{P}_k)r_0 \quad (3.5.23)$$

$$\implies r_k \perp \text{ran}(P_k) \quad (3.5.24)$$

Proposition 3.4 (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:

$$p_k = (I - \bar{P}_k A)u_k \quad (3.5.25)$$

$$\text{span}(p_k) \perp \text{ran}(AP_k) \quad (3.5.26)$$

For above propositions, we used the immediate consequence of the range of these oblique projectors.

3.5.3 Assisted Conjugate Gradient (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 6 (Assited Conjugate Gradient).

$$\begin{cases} x_k = x_0 + \bar{P}_k r_0 \\ r_k = (I - A\bar{P}_k)r_0 \\ P_k^T A P_k = D_k \\ \bar{P}_k = P_k D_k^{-1} P_k^T \\ p_k = (I - \bar{P}_k A)u_k \quad \{u_i\}_{i=0}^{n-1} \text{ is a Basis} \end{cases} \quad (3.5.27)$$

With the assistance 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: *Assisted Conjugate Gradient*.

Remark 3.5.1 (ACG is Conjugate Direction Method). This ACG 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 3.5.2 (Geometric Intutions of Assisted CG). What is happening geometrically is that the A-Orthogonal vectors are orthogonal if describe it under the alternative eigen-subspaces. 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.

3.5.4 Properties of Assisted Conjugate Gradient

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

Proposition 3.5.

$$p_{k+j}^T r_k = p_{k+j}^T r_0 \quad \forall 0 \leq j \leq n - k \quad (3.5.28)$$

$$p_{k+j}^T r_k = p_k^T (I - A\bar{P}_k) r_0 \quad (3.5.29)$$

$$= (p_{k+j}^T - p_{k+j}^T A\bar{P}_k) r_0 \quad (3.5.30)$$

$$= p_{k+j}^T r_0 \quad (3.5.31)$$

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

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

Proposition 3.6 (ACG Recurrence).

$$r_k - r_{k-1} = r_0 - A\bar{P}_k r_0 - (r_0 - A\bar{P}_{k-1} r_0) \quad (3.5.32)$$

$$= A\bar{P}_k r_0 - A\bar{P}_{k-1} r_0 \quad (3.5.33)$$

$$= -Ap_{k-1} \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \quad (3.5.34)$$

$$\implies x_k - x_{k-1} = p_{k-1} \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \quad (3.5.35)$$

$$\text{def: } a_{k-1} := \frac{\langle p_{k-1}, r_0 \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} = \frac{\langle p_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \quad (3.5.36)$$

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 RACG algorithm that we are going to develop.

3.5.5 Residual Assisted 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 ACG 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 orthogonl. We wish to proceed to prove that part.

Lemma 3.5.1.

$$\langle p_{k+j}, Ap_k \rangle = \langle r_k, Ap_{k+j} \rangle = \langle p_{k+j}, Ar_k \rangle \quad \forall 0 \leq j \leq n - k \quad (3.5.37)$$

Proof.

$$p_{k+j}Ap_k = p_{k+j}^T Ar_k - p_{k+j}^T A\bar{P}_k Ar_k \quad \forall 0 \leq j \leq n - k \quad (3.5.38)$$

$$= p_{k+j}^T Ar_k \quad (3.5.39)$$

$$\langle p_{k+j}, Ap_k \rangle = \langle r_k, Ap_{k+j} \rangle = \langle p_{k+j}, Ar_k \rangle \quad (3.5.40)$$

□

Lemma 3.5.2.

$$\langle r_k, p_k \rangle = \langle r_k, r_k \rangle \quad (3.5.41)$$

Proof.

$$\langle r_k, p_k \rangle = \langle r_k, p_k \rangle \quad (3.5.42)$$

$$= \langle r_k, r_k \rangle - \langle r_k, \bar{P}_k Ar_k \rangle \quad (3.5.43)$$

$$= \langle r_k, r_k \rangle \quad (3.5.44)$$

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

Proposition 3.7 (Residual Assisted CG Generates Orthogonal Residuals).

$$\langle r_k, r_j \rangle = 0 \quad \forall 0 \leq j \leq k - 1 \quad (3.5.45)$$

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

Proof.

$$r_{k+1} = r_k - a_k Ap_k \quad (3.5.46)$$

$$\implies \langle r_{k+1}, r_k \rangle = \langle r_k, r_k \rangle - a_k \langle r_k, Ap_k \rangle \quad (3.5.47)$$

$$= \langle r_k, r_k \rangle - \frac{\langle r_k, r_k \rangle}{\langle p_k, Ap_k \rangle} \langle r_k, Ap_k \rangle \quad (3.5.48)$$

$$= 0 \quad (3.5.49)$$

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

$$p_j = (I - \bar{P}_j A) r_j \quad \forall 0 \leq j \leq k-1 \quad (3.5.50)$$

$$\implies r_j = p_j + \bar{P}_j A r_j \quad (3.5.51)$$

$$r_k = (I - A \bar{P}_k) r_0 \quad (3.5.52)$$

$$r_k \perp \text{ran}(P_k) \implies \langle r_k, r_j \rangle = \langle r_k, p_j + \bar{P}_j A r_j \rangle = 0 \quad (3.5.53)$$

The second line is a result of the first line. Here we again make use of the projector $I - A \bar{P}_k$. 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. \square

Proposition 3.8 (RACG Recurrences).

$$p_k = r_k + b_{k-1} p_{k-1} \quad b_{k-1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \quad (3.5.54)$$

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

$$p_k = (I - \bar{P}_k A) r_k \quad (3.5.55)$$

$$r_k - \bar{P}_k A r_k = r_k - P_k D_k^{-1} P_k^T A r_k \quad (3.5.56)$$

$$= r_k - P_k D_k^{-1} (A P_k)^T r_k \quad (3.5.57)$$

Observe:

$$(A P_k)^T r_k = \begin{bmatrix} \langle p_0, A r_k \rangle \\ \langle p_1, A r_k \rangle \\ \vdots \\ \langle p_{k-1}, A r_k \rangle \end{bmatrix} \quad (3.5.58)$$

Next, we can make use of [lemma 3.5.1](#) to get rid of $A r_k$:

$$\langle p_j, A r_k \rangle \quad \forall 0 \leq j \leq k-2 \quad (3.5.59)$$

$$\langle p_j, A r_k \rangle = \langle r_k, A p_j \rangle \quad (3.5.60)$$

$$= \langle r_k, a_j^{-1} (r_j - r_{j+1}) \rangle \quad (3.5.61)$$

$$= a_j^{-1} \langle r_k, (r_j - r_{j+1}) \rangle = 0 \quad (3.5.62)$$

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 ACG ([ACG Recurrences](#)), and the last line is true for all $0 \leq j \leq k-2$ by the orthogonality of the residual proved in [proposition 1.6](#). Therefore we have:

$$(A P_k)^T r_k = \begin{bmatrix} \langle p_0, A r_k \rangle \\ \langle p_1, A r_k \rangle \\ \vdots \\ \langle p_{k-1}, A r_k \rangle \end{bmatrix} = a_{k-1}^{-1} \langle r_k, (r_{k-1} - r_k) \rangle \xi_k \quad (3.5.63)$$

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:

$$p_k = r_k - P_k D_k^{-1} (A P_k)^T r_k \quad (3.5.64)$$

$$= r_k - P_k D_k^{-1} a_{k-1}^{-1} (\langle r_k, (r_{k-1} - r_k) \rangle) \xi_k \quad (3.5.65)$$

$$= r_k - \frac{a_{k-1}^{-1} \langle -r_k, r_k \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} p_k \quad (3.5.66)$$

$$= r_k + \frac{a_{k-1}^{-1} \langle r_k, r_k \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} p_k \quad (3.5.67)$$

$$= r_k + \left(\frac{\langle r_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} \right)^{-1} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, A p_{k-1} \rangle} p_k \quad (3.5.68)$$

$$= r_k + \frac{\langle r_k, r_k \rangle}{\langle r_{k-1}, r_{k-1} \rangle} p_k \quad (3.5.69)$$

We make use of the definition for a_{k-1} for the ACG algorithm. At this point, we have proven the short RACG recurrences for p_k . \square

Up until this point we have proven the usual of conjugate gradient proposed by Hestenes & Stiefel (**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 RACG:

Definition 7 (RACG).

$$p^{(0)} = b - A x^{(0)} \quad (3.5.70)$$

$$\text{For } i = 0, 1, \dots \quad (3.5.71)$$

$$\begin{aligned} a_i &= \frac{\|r^{(i)}\|^2}{\|p^{(i)}\|_A^2} \\ x^{(i+1)} &= x^{(i)} + a_i p^{(i)} \\ r^{(i+1)} &= r^{(i)} - a_i A p^{(i)} \\ b_i &= \frac{\|r^{(j+1)}\|_2^2}{\|r^{(i)}\|_2^2} \\ p^{(i+1)} &= r^{(i+1)} + b_i p^{(i)} \end{aligned} \quad (3.5.72)$$

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.

3.5.6 RACG 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 excerpt. The full algorithm can be seen 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 3.9.

$$p_k \in \mathcal{K}_{k+1}(A|r_0) \quad (3.5.73)$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \quad (3.5.74)$$

Proof. The base case is trivial 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:

$$r_{k+1} = r_k - a_k A p_k \quad (3.5.75)$$

$$\in r_k + A \mathcal{K}_{k+1}(A|r_0) \quad (3.5.76)$$

$$\in r_k + \mathcal{K}_{k+2}(A|r_0) \quad (3.5.77)$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \subseteq \mathcal{K}_{k+2}(A|r_0) \quad (3.5.78)$$

$$\implies r_{k+1} \in \mathcal{K}_{k+2}(A|r_0) \quad (3.5.79)$$

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

$$p_{k+1} = r_{k+1} + b_k p_k \quad (3.5.80)$$

$$\in r_{k+1} + \mathcal{K}_{k+1}(A|r_0) \quad (3.5.81)$$

$$\in \mathcal{K}_{k+2}(A|r_0) \quad (3.5.82)$$

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 5 (CG and Krylov Subspace).

$$\text{choose: } x_k \in x_0 + \mathcal{K}_k(A|r_0) \text{ s.t: } r_k \perp \mathcal{K}_k(A|r_0) \quad (3.5.83)$$

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. \square

3.6 Arnoldi Iterations and Lanczos

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

3.6.1 The Arnoldi Iterations

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

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

$$\tilde{H}_k = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{1,2} & h_{2,2} & \cdots & h_{2,k} \\ & \ddots & & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix} \quad (3.6.1)$$

We initialize the orthogonal 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:

$$(\tilde{H}_k)_{2,1}q_2 = (I - q_1 q_1^H)Aq_1 \quad (3.6.2)$$

$$(\tilde{H}_k)_{1,1}q_1 = q_1 q_1^H Aq_1 \quad (3.6.3)$$

$$Q_2 := [q_1 \quad q_2] \quad (3.6.4)$$

$$(\tilde{H}_k)_{3,2}q_3 = (I - Q_2 Q_2^H)Aq_2 \quad (3.6.5)$$

$$(\tilde{H}_k)_{1:2,2} = Q_2 Q_2^H Aq_2 \quad (3.6.6)$$

$$Q_3 := [q_1 \quad q_2 \quad q_3] \quad (3.6.7)$$

$$\vdots \quad (3.6.8)$$

$$Q_j := [q_1 \quad q_2 \quad \cdots \quad q_j] \quad (3.6.9)$$

$$(\tilde{H}_k)_{j+1,j}q_{j+1} = (I - Q_j Q_j^H)Aq_j \quad (3.6.10)$$

$$(\tilde{H}_k)_{1:j,j} = Q_j Q_j^H Aq_j \quad (3.6.11)$$

$$\vdots \quad (3.6.12)$$

$$Q_k := [q_1 \quad q_2 \quad \cdots \quad q_k] \quad (3.6.13)$$

$$(\tilde{H}_k)_{k+1,k}q_{k+1} = (I - Q_k Q_k^H)Aq_k \quad (3.6.14)$$

$$(\tilde{H}_k)_{1:k,k} = Q_k Q_k^H Aq_k \quad (3.6.15)$$

Reader please observe that Q_k is going to be orthogonal because how at the start, $q_1 q_1^H$ and $I - q_1 q_1^H$ is giving us an orthogonal subspace. As a consequence, we can express the recurrences of the subspace vector in matrix form:

$$AQ_k = Q_{k+1} \tilde{H}_k \quad (3.6.16)$$

$$Q_k^H A Q_k = H_k \quad (3.6.17)$$

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 $Q_k^H A Q_k$ 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 orthogonalizing 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.

3.6.2 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 3.10.

$$\text{ran}(Q_k) = \mathcal{K}_k(A|q_1) \quad (3.6.18)$$

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 $I - Q_k Q_k^H$ applied to Aq_k and the null space of this projector is $\text{ran}(Q_k)$, resulting in $h_{k+1,k} = 0$. \square

3.6.3 The Lanczos Iterations

Definition 8 (Lanczos Iterations).

$$\text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \quad (3.6.19)$$

$$\text{set: } \beta_0 = 0 \quad (3.6.20)$$

$$\text{For } j = 1, 2, \dots \quad (3.6.21)$$

$$\begin{aligned} \tilde{q}_{j+1} &:= Aq_j - \beta_{j-1}q_{j-1} \\ \alpha_j &:= \langle q_j, \tilde{q}_{j+1} \rangle \\ \tilde{q}_{j+1} &\leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\ \beta_j &= \|\tilde{q}_{j+1}\| \\ q_{j+1} &:= \tilde{q}_{j+1}/\beta_j \end{aligned} \quad (3.6.22)$$

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:

$$\begin{aligned}
&\text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \\
&\alpha_1 := \langle q_1, Aq_1 \rangle \\
&\beta_0 := 0 \\
&\text{Memorize : } Aq_1 \\
&\text{For } j = 1, 2, \dots \\
&\quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1}q_{j-1} \\
&\quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\
&\quad \beta_j = \|\tilde{q}_{j+1}\| \\
&\quad q_{j+1} := \tilde{q}_{j+1}/\beta_j \\
&\quad \alpha_{j+1} := \langle q_{j+1}, Aq_{j+1} \rangle \\
&\quad \text{Memorize: } Aq_{j+1}
\end{aligned} \tag{3.6.23}$$

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:

$$Q_k = [q_1 \quad q_2 \quad \dots \quad q_k] \tag{3.6.24}$$

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{3.6.25}$$

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 6 (Lanczos Recurrenes).

$$AQ_k = Q_k T_k + \beta_k q_{k+1} \xi_k^T = Q_{k+1} \tilde{T}_k \tag{3.6.26}$$

$$\implies Aq_k = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \quad \forall 2 \leq j \leq k \tag{3.6.27}$$

$$\implies Aq_1 = \alpha_1 q_1 + \beta_1 q_2 \tag{3.6.28}$$

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 3.11. 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 terminates, and k would equal to $\text{grade}(A|q_1)$, the grade of the Krylov Subspace.

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

4 Analysis of Conjugate Gradient and Lanczos Iterations

In this section we state the terminations conditions for the Lanczos iterations and the RACG algorithm we developed using the property of Krylov Subspace. This is just a thoroughly 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).

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

$$x_k = \mathcal{K}(r_0)w + x_0 \tag{4.1.2}$$

$$\frac{\|e_k\|_A^2}{\|e_0\|_A^2} = \min_{w \in \mathbb{R}^k} \|(I + Ap_k(A|w))A^{1/2}e_0\|_2^2 \tag{4.1.3}$$

$$\leq \min_{p_{k+1}: p_{k+1}(0)=1} \max_{x \in [\lambda_{\min}, \lambda_{\max}]} |p_{k+1}(x)| \tag{4.1.4}$$

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

Proof.

$$\|e_k\|_A^2 = \min_{x_k \in x_0 + \mathcal{K}_k(A|r_0)} \|x^+ - x_k\|_A^2 \quad (4.1.5)$$

$$x_k \in x_0 + \mathcal{K}_k(A|r_0) \iff e_k = e_0 + p_k(A|w)r_0 \quad (4.1.6)$$

$$\implies = \min_{w \in \mathbb{R}^k} \|e_0 + p_k(A|w)r_0\|_A^2 \quad (4.1.7)$$

$$= \min_{w \in \mathbb{R}^k} \|e_0 + Ap_k(A|w)e_0\|_A^2 \quad (4.1.8)$$

$$= \min_{w \in \mathbb{R}^k} \|A^{1/2}(I + Ap_k(A|w))e_0\|_2^2 \quad (4.1.9)$$

$$\leq \min_{w \in \mathbb{R}^k} \|I + Ap_k(A|w)\|_2^2 \|e_0\|_A^2 \quad \text{tight} \quad (4.1.10)$$

$$= \min_{w \in \mathbb{R}^k} \left(\max_{i=1, \dots, n} |1 + \lambda_i p_k(\lambda_i|w)|^2 \right) \|e_0\|_A^2 \quad (4.1.11)$$

$$\leq \min_{w \in \mathbb{R}^k} \left(\max_{x \in [\lambda_{\min}, \lambda_{\max}]} |1 + xp_k(x|w)|^2 \right) \|e_0\|_A^2 \quad \text{still tight} \quad (4.1.12)$$

$$= \min_{p_{k+1}: p_{k+1}(0)=1} \max_{x \in [\lambda_{\min}, \lambda_{\max}]} |p_{k+1}(x)|^2 \|e\|_A^2 \quad (4.1.13)$$

$$\implies \frac{\|e_k\|_A}{\|e_0\|_A} \leq \min_{p_{k+1}: p_{k+1}(0)=1} \max_{x \in [\lambda_{\min}, \lambda_{\max}]} |p_{k+1}(x)| \quad (4.1.14)$$

We proceed with writing up the affine subspace where x_k is from: $x_0 + \mathcal{K}_k(A|r_0)$, putting it in terms of matrix polynomial mulitplied by r_0 and then use $A^{-1}b$ to subtract both side. From the 3rd line to the 4th, we use the fact that $r_0 = Ae_0$, allowing us to extract out a factor A . Next, on the 4th line to the next, we use the fact that every Definite Matrix A has the factorization of $A^{1/2}A^{1/2}$ where $A^{1/2}$ is also a Definite Matrix. After that we moved the $A^{1/2}$ to e_0 to get $\|e_0\|_A^2$ and the matrix polynomial part is left with the 2-norm. Next we use the eigendecompoition of A which diagonalizable and can have unitary eigenvectors, 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:

$$\|I + Ap_k(A|w)\|_2^2 = \|Q(I + \Lambda p_k(\Lambda|w))Q^T\|_2^2 \quad (4.1.15)$$

$$= \|I + \Lambda p_k(\Lambda|w)\|_2^2 \quad (4.1.16)$$

$$= \max_{i=1, \dots, n} |1 + \lambda_i p_k(\lambda_i|w)|^2 \quad (4.1.17)$$

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 . Finally, please notice thtta we use an monic $p_{k+1}(x)$ at the and to simplify things. \square

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

Remark 4.1.1 (The RACG 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. (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.

4.2 Termination Conditions of RACG

Under exact arithmetic, the algorithm terminates at most n iterations where n is the size of the matrix A . This is true due to the [CG and Krylov Subspace](#), the [Grade for a Krylov Subspace](#). However, this bound is true for all definite matrix A , but there are conditions where the termination of the CG algorithm comes early and it depends on the grade of $\mathcal{K}_k(A|r_0)$, which then depends on the eigenvalues of the matrix A and initial guess r_0 .

Proposition 4.2. The $\text{grade}(A|r_0)$ determines the number maximum number of iterations required before the terminations of the CG, and by the time it terminates, the residual will be the zero vector. Further more, the upper bound for the grade of the subspace is the number of unique eigenvalues for the matrix A , There also exists initial guess r_0 where the number of iterations required might be shorter if it's projection onto some of the eigen vectors are zero.

For a justification, we consider the Krylov Subspace accumulated during the CG algorithm. The grade of the Krylov subspace $\mathcal{K}_k(A|r_0)$ determines when the CG algorithm is going to terminate. Suppose that $\text{grade}(A|r_0)$ is $k + 1$, then $\mathcal{K}_k(A|r_0) = \mathcal{K}_{k+1}(A|r_0)$, and \mathcal{K}_k would be linear independent while \mathcal{K}_{k+1} would be dependent. The Conjugate Gradient asserts $r_{k-1} \in \mathcal{K}_k(A|r_0)$ and $r_k \in \mathcal{K}_{k+1}(A|r_0) = \mathcal{K}_k(A|r_0)$. But at the same time the CG algorithm asserts that $r_j \perp r_j \forall 0 \leq j \leq k - 1$. Observe that inductively CG asserts $r_j \in \mathcal{K}_{j+1}(A|r_0)$ and all of them are mutually orthogonal, and there are k of them in total. Using the nesting property of Krylov Subspace we know that $r_k \perp \mathcal{K}_k(A|r_0)$, therefore they must span the whole space. However, $r_k \in \mathcal{K}_k(A|r_0)$ because the subspace becomes invariant after $k - 1$, therefore it has to be the case that $r_k = \mathbf{0}$. When it happens, it will result in b_{k-1} being zero because of CG, which will give $p_k = \mathbf{0}$. Which will terminate the algorithm at step $k + 1$ due to a division of zero inside the expression for a_k .

Next, we wish to say more about the maximum grade of a Krylov Subspace. Recall from the Krylov Subspace discussion, when the grade is reached, there exists non trivial polynomial expression where:

$$\begin{aligned} \mathbf{0} &= r_0 + \sum_{j=1}^{k-1} w_0^{-1} w_j A^j r_0 \\ \mathbf{0} &= Q \left(I + \sum_{j=1}^{k-1} w_0^{-1} w_j A^j \right) Q^T r_0 \end{aligned}$$

We use the eigen factorization for the S.P.D matrix A . One of the immediate consequence of the above equation would imply that, if there exists a monic polynomial interpolating all the eigenvalues of matrix A , then the grade of the Krylov Subspace is reached. As a consequence of that, for any initial vector, then CG must terminate as the same number of unique eigenvalues of matrix A . Finally, take notice that the projections of r_0 only covers a portion of the eigenspace then the CG algorithm will terminate earlier. This is true because $\mathcal{K}_k(A|r_0) = Q \mathcal{K}_k(\Lambda|r_0) Q^T r_0$, and please observe that the maximum dimension equals to the number of non-zero elements in $Q^T r_0$, which further shortens the number of iterations required.

4.3 Convergence Rate of RACG under Exact Arithematic

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

4.3.1 Uniformly Distributed Eigenvalues

Theorem 7 (CG Convergence Rate). The relative error squared measured over the energy norm is bounded by:

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

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 $k \ll n$ 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 monic 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:

$$T_k(x) = \min_{\substack{p(x) \in \mathcal{P}_{k+1} \\ \text{s.t.: } p(0)=1}} \max_{x \in [-1,1]} |p(x)| \quad (4.3.2)$$

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

$$\text{then: } p_k(x) = \min_{\substack{p(x) \in \mathcal{P}_{k+1} \\ \text{s.t.: } p(0)=1}} \max_{x \in [\lambda_1, \lambda_n]} |p(x)| \quad (4.3.4)$$

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, here we use T_k for the type T Chebyshev Polynomial of degree k and it's not the Tridiagonal Symmetric Matrix from Lanczos. Next, we use the property that the range of the Chebyshev is bounded within the interval $[-1, 1]$ to obtain inequality:

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

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 $1 < \varphi(0) \notin [\lambda_1, \lambda_n]$, because all Eigenvalues are larger than zero, therefore it's out of the range of the Cheb and we need to find the actual value of it by considering alterantive form of Chebyshev T for values outside of the $[-1, 1]$:

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

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

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 .

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

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

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

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

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 :

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

$$y = \frac{\sqrt{\kappa} \pm 1}{\sqrt{\kappa} \mp 1} \quad (4.3.13)$$

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 $T_k(\varphi(0))$ interms of y and then κ :

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

$$\implies T_k(\varphi(0)) = T_k(\cosh(\ln(y))) \quad (4.3.15)$$

$$= \cosh(k \ln(y)) \quad (4.3.16)$$

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

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

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

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

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

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

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

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

□

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 single eigenvalue that is far away from the cluster. In that case, we can use Chebyshev differently by focusing its minimizing power across the clustered eigenvalues and use a simple polynomial to interpolate the outlier eigenvalue. Consider the following proposition:

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

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

Reader please observe that, the outlier eigenvalue κ_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:

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

Next, we consider the following polynomial:

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

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:

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

$$\frac{\lambda_n - z}{\lambda_n} < 1 \quad \forall z \in [\lambda_1, \lambda_{n-1}] \quad (4.3.27)$$

$$|p_k(x)| \leq \left| \frac{\hat{T}_{[\lambda_1, \lambda_{n-1}]}^{(k-1)}(x)}{\hat{T}_{[\lambda_1, \lambda_{n-1}]}^{(k-1)}(0)} \frac{\lambda_n - z}{\lambda_n} \right| \leq \frac{1}{\left| \hat{T}_{[\lambda_1, \lambda_{n-1}]}^{(k-1)}(0) \right|} \quad (4.3.28)$$

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

$$T_{[\lambda_1, \lambda_{n-1}]}^{(k-1)}(0) = \left| T_{k-1} \left(\frac{-\lambda_{n-1} - \lambda_1}{\lambda_{n-1} - \lambda_1} \right) \right| \quad (4.3.29)$$

$$= \frac{1}{2} (y^{k-1} + y^{-(k-1)}) \quad (4.3.30)$$

$$\text{where: } y = \frac{\sqrt{\kappa_{n-1}} + 1}{\sqrt{\kappa_{n-1}} - 1}, \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \quad (4.3.31)$$

Substituting the value for y we obtain the bound:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa_{n-1}} - 1}{\sqrt{\kappa_{n-1}} + 1} \right)^{k-1} \quad (4.3.32)$$

□

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 we I mean the value of λ_1 is much smaller than all other eigenvalues and the other eigenvalues are clutered closed together in an interval uniformly in some interval.

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

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2 \left(\frac{\lambda_n - \lambda_1}{\lambda_1} \right) \left(\frac{\sqrt{\kappa_0} - 1}{\sqrt{\kappa_0} + 1} \right)^{k-1} \quad (4.3.33)$$

Where κ_0 is λ_n/λ_2 .

Proof.

$$w(z) := \frac{\lambda_1 - z}{\lambda_1} \quad (4.3.34)$$

$$p_k(z) := w(z) \left(\frac{\hat{T}_{[\lambda_2, \lambda_n]}^{(k-1)}(z)}{\hat{T}_{[\lambda_2, \lambda_n]}^{(k-1)}(0)} \right) \quad (4.3.35)$$

$$\implies \max_{x \in [\lambda_2, \lambda_n]} |w(x)| = \frac{\lambda_n - \lambda_1}{\lambda_1} \quad (4.3.36)$$

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

$$|p_k(z)| = \left| w(z) \frac{\hat{T}_{[\lambda_2, \lambda_n]}^{(k-1)}(z)}{\hat{T}_{[\lambda_2, \lambda_n]}^{(k-1)}(0)} \right| \quad (4.3.37)$$

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

$$\leq \left| \left(\frac{\lambda_n - \lambda_1}{\lambda_1} \right) \hat{T}_{[\lambda_2, \lambda_n]}^{(k-1)}(0) \right| \quad (4.3.39)$$

$$\implies \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_1} \right) 2 \left(\frac{\sqrt{\kappa_0} + 1}{\sqrt{\kappa_0} - 1} \right)^{k-1} \quad (4.3.40)$$

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 , making it clearn that if λ_1 is close to zero the convergence rate is going to be terrible. \square

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 Krylove 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 RACG 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 while 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:

$$q_1 = \hat{r}_0 \quad (4.4.1)$$

$$q_2 = -\hat{r}_1 \quad (4.4.2)$$

$$\vdots \quad (4.4.3)$$

$$q_j = (-1)^{j+1} \hat{r}_{j+1} \quad (4.4.4)$$

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 :

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

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:

$$r_0 = p_0 \quad (4.4.6)$$

$$r_1 = r_0 - a_0 A r_0 \quad (4.4.7)$$

$$A r_0 = a_0^{-1} (r_0 - r_1) \quad (4.4.8)$$

$$A r_0 = \frac{\|r_0\|_A^2}{\|r_0\|^2} (r_0 - r_1) \quad (4.4.9)$$

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

$$A \|r_0\| q_1 = \frac{\|r_0\|_A^2}{\|r_0\|^2} (\|r_0\| q_1 + \|r_1\| q_2) \quad (4.4.10)$$

$$= \frac{\|r_0\|_A^2}{\|r_0\|^2} \|r_0\| q_1 + \frac{\|r_1\|}{\|r_0\|} q_2 \quad (4.4.11)$$

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

4.4.2 The Inductive Case

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

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

Proof. We start by consiering:

$$r_j = r_{j-1} - a_{j-1}Ap_{j-1} \quad (4.4.13)$$

$$= r_{j-1} - a_{j-1}A(r_{j-1} + b_{j-2}p_{j-1}) \quad (4.4.14)$$

$$= r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-1} \quad (4.4.15)$$

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

$$r_{j-1} = r_{j-1} - a_{j-2}Ap_{j-1} \quad (4.4.16)$$

$$r_{j-1} - r_{j-1} = a_{j-2}Ap_{j-1} \quad (4.4.17)$$

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

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

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

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

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

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

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

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:

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

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 coefficients of r_{j-1} and r_{j+1} to be in the same form, parameterized by the same iterations parameter: j . To do that, consider the doing this:

$$q_{j+1} := \frac{r_j}{\|r_j\|} \quad (4.4.25)$$

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

$$q_{j+2} := \frac{r_{j+1}}{\|r_{j+1}\|} \quad (4.4.27)$$

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

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

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:

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

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

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

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 equivalent 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:

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

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 multiply 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:

$$q_1 = \hat{r}_0 \quad (4.4.34)$$

$$q_2 = -\hat{r}_1 \quad (4.4.35)$$

$$\vdots \quad (4.4.36)$$

$$q_j = (-1)^{j+1} \hat{r}_{j+1} \quad (4.4.37)$$

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 Lanczos?

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:

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

The quantities α, β are the diagonal 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 i th canonical basis vector. β without the subscript denotes $\|r_0\|$.

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

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

Recall that the Conjugate Gradient algorithm takes the guesses from the affine span of $x_0 + \mathcal{K}_k(A|r_0)$, and that means:

$$x_{k+1} = x_0 + Q_k y_k \quad (4.5.3)$$

$$r_{k+1} = r_0 - AQ_k y_k \quad (4.5.4)$$

$$Q_k^H r_{k+1} = Q_k^H r_0 - Q_k^H A Q_k y_k \quad (4.5.5)$$

$$\implies 0 = \beta \xi_1 - T_k y_k \quad (4.5.6)$$

$$y_k = T_k^{-1} \beta \xi_1 \quad (4.5.7)$$

Reader please recall from section [RACG and Krylov Subspace](#) that: $p_k \in \mathcal{K}_{k+1}(A|r_0)$, which is implying a link between the p_k A-orthogonal vectors and the Lanczos Orthogonal matrix: Q_k . Now to get the residual we simply consider:

$$r_{k+1} = r_0 - AQ_k y_k \quad (4.5.8)$$

$$= r_0 - AQ_k T_k^{-1} \beta \xi_1 \quad (4.5.9)$$

$$\implies = \beta q_1 - AQ_k T_k^{-1} \beta \xi_1 \quad (4.5.10)$$

$$= \beta q_1 - Q_{k+1} \begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} T_k^{-1} \beta \xi_1 \quad (4.5.11)$$

$$= \beta q_1 - (Q_k T_k + \beta_k q_{k+1} \xi_k^T) T_k^{-1} \beta \xi_1 \quad (4.5.12)$$

$$= \beta q_1 - (Q_k \beta \xi_1 + \beta_k q_{k+1} \xi_{k+1}^T T_k^{-1} \beta \xi_1) \quad (4.5.13)$$

$$= -\beta_k q_{k+1} \xi_k^T T_k^{-1} \beta \xi_1 \quad (4.5.14)$$

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

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

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

$$P_k = Q_k U_k^{-1} \quad (4.5.15)$$

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:

$$x_k = x_0 + Q_k y_k \quad (4.5.16)$$

$$= x_0 + Q_k T_k^{-1} \beta \xi_1 \quad (4.5.17)$$

$$= x_0 + Q_k U_k^{-1} L_k^{-1} \beta \xi_1 \quad (4.5.18)$$

$$= x_0 + P_k L_k^{-1} \beta \xi_1 \quad (4.5.19)$$

So far we have written the solution vector x_k . Next, we are going to prove that the matrix P_k indeed consists of vectors that are A-orthogonal. To show that we consider:

$$P_k^T A P_k \quad (4.5.20)$$

$$= (Q_k U_k^{-1})^T A Q_k U_k^{-1} \quad (4.5.21)$$

$$= U_k^{-T} Q_k^T A Q_k U_k^{-1} \quad (4.5.22)$$

$$= U_k^{-T} T_k U_k^{-1} \quad (4.5.23)$$

$$= U_k^{-T} L_k \quad (4.5.24)$$

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. \square

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 advantage computing $\xi_k^T T_k^{-1} \xi_1$ with as little efforts as possible. Let's take a look.

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

$$P_k = [p_1 \ p_2 \ \cdots \ p_k] \quad Q_k = [q_1 \ q_2 \ \cdots \ q_k] \quad (4.5.25)$$

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:

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

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 recall the expression for x_k from the previous section, we have:

$$x_k = x_0 + P_k L_k^{-1} \beta \xi_1 \quad (4.5.27)$$

$$x_k - x_{k-1} = P_k L_k^{-1} \beta \xi_1 - P_{k-1} L_{k-1}^{-1} \beta \xi_1 \quad (4.5.28)$$

$$= P_k \beta (L_k^{-1})_{:,1} - P_{k-1} \beta (L_{k-1}^{-1})_{:,1} \quad (4.5.29)$$

$$= \beta (L_k^{-1})_{k,1} P_k \quad (4.5.30)$$

$$\implies x_k = x_{k-1} + \beta (L_k^{-1})_{k,1} p_k \quad (4.5.31)$$

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:

$$P_k = Q_k U_k^{-1} \quad (4.5.32)$$

$$P_k U_k = Q_k \quad (4.5.33)$$

$$\implies \beta_{k-1} p_{k-1} + u_k p_k = q_k \quad (4.5.34)$$

$$u_k p_k = q_k - \beta_{k-1} p_{k-1} \quad (4.5.35)$$

$$p_k = u_k^{-1} (q_k - \beta_{k-1} p_{k-1}) \quad (4.5.36)$$

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:

$$T_k = L_k U_k \quad (4.5.37)$$

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

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

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

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

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:

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

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:

$$L_k^{-1}L_k = I \quad (4.5.43)$$

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

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

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:

$$s^T L_k + d_{k+1} l_k \xi_k^T = \mathbf{0} \quad (4.5.46)$$

$$L_k^T s_k + d_{k+1} l_k \xi_k = \mathbf{0} \quad (4.5.47)$$

$$s_k + L^{-T} d_{k+1} l_k \xi_k = \mathbf{0} \quad (4.5.48)$$

$$(s_k)_1 + d_{k+1} l_k ((L_k^{-1}) \xi_k)_1 = 0 \quad (4.5.49)$$

$$(s_k)_1 + d_{k+1} l_k (L_k^{-1})_{k,1} = 0 \quad (4.5.50)$$

$$\implies (s_k)_1 = -l_k (L_k^{-1})_{k,1} \quad (4.5.51)$$

$$(s_k)_1 = (L_{k+1}^{-1})_{k+1,1} \quad \text{by definition} \quad (4.5.52)$$

$$\implies (L_{k+1}^{-1})_{k+1,1} = -l_k (L_k^{-1})_{k,1} \quad (4.5.53)$$

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 RACG 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:

$$A Q_k \approx Q_{k+1} \begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} \quad (4.5.54)$$

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 its 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 Arithmetic

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 arithmetic.

5.1 Partial Orthogonalization and Full Orthogonalization

In this section, we use some of the results and insight obtained from the derivation of the RACG algorithm to develop a theoretical fix for the loss of precision under floating point arithmetic for the RACG 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 ACG algorithm's projector to re-orthogonalize the conjugate vectors and the residual vectors so the A-Orthogonality 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 Stiefel, 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 \bar{r}_{k+1} are given and it breaks

the orhogonality conditions $\bar{r}_{k+1} \perp r_j \forall 0 \leq j \leq k$:

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

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

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

$$r_k := \bar{r}_j - \sum_{j=0}^{k-1} \langle \hat{r}_j, \bar{r}_k \rangle \hat{r}_j \quad (5.1.4)$$

Here, we generate the conjugate vectors p_k correctly by faithfully reproducing the term $(AP_k)^T \bar{r}_k$, and then we update the residual \bar{r}_k into r_k by orthogonalizing it against all previous residual vectors. Doing this 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 exagurate the floating point round off error. The spectrum of the matrix we are using are taken to be the same from Greenbaum's works (**CITATION NEEDED**):

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

If the value of ρ is close to zero, then the eigenvalues are clustered around the origin, if it's close to 1, then the eigenvalues are tend to be more evenly distributed around the interval $[\lambda_{\min}, \lambda_{\max}]$. The Chebyshev bound is no longer a tight bound because the distribution of 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 arithmetic and the promised efficiency of the algorithm is not there anymore if the matrix is ill-conditioned. If we were to use a fully re-orthogonalized variant of the CG, the memory needed and the totally complexity for a fixed tolerance is going to be on the same scale of solving it by direct methods.

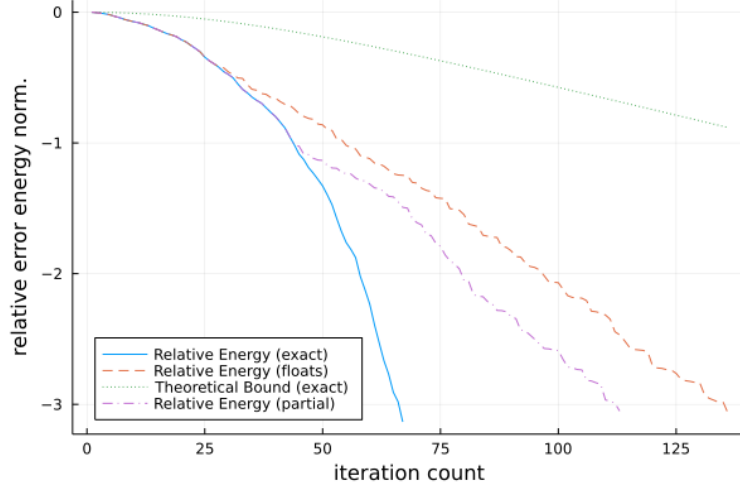


Figure 1: The relative energy norm error for different methods. Blue solid line: The exact conjugate gradient convergence. Purple dottedashed: The conjugate gradient that is partially orthogonalized with previous 32 residual and conjugate vectors. orange dashed: the Original conjugate gradient. Green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

5.3 Greenbaum's Backward Analysis and 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 arithmetic is used, the eigenvalues of the Tridiagonal matrices might introduces ghost eigenvectors for the Lanczos Iterations, and using the equivalence of the Lanczos iterations and CG, we can capture a poster bound on how much the error is exactly during the iterations of CG.

5.3.1 Bounding the The Relative Residuals

Recall from proposition [Proposition 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: $y_k = \beta T_k^{-1} \xi_1$ 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:

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

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.

$$r_k = r_0 - AQ_k y_k \quad (5.3.2)$$

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

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

$$\Rightarrow \frac{\|r_{k+1}\|}{\|r_0\|} \leq \beta_k \|\xi_k^T T_k^{-1} \xi_1 q_{k+1}\| + \|F_k T_k^{-1} \xi_1\| \quad (5.3.5)$$

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

Here, because of the assumption of exactness for the Q_k matrices we are able to reuse [Proposition 4.6](#) to regauge 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 arithmetic. The idea is that the bound for the exact arithmetic, [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 \bar{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 excitement, we already have the bound for \bar{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 proposed 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 8 (Paige's Theorem). The eigenvalues $\theta_i^{(j)}, i = 1, \dots, j$ of the tridiagonal matrix T_j satisfies:

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

$$\epsilon_2 := \sqrt{2} \max\{6\epsilon_0, \epsilon_1\} \quad (5.3.8)$$

Along with this theorem, the following quantities are also defined and cited in Greenbaum's

work (**CITATION NEEDED**) from the work of Paige.

$$\epsilon_0 \equiv 2(n+4)\epsilon \quad (5.3.9)$$

$$\epsilon_1 \equiv 2(7+m\| |A| \|/\|A\|)\epsilon \quad (5.3.10)$$

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

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

$$\|q_j^T q_j - 1\| \leq 2\epsilon_0 \quad (5.3.13)$$

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

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:

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

Using [\(remark 5.3.1\)](#), we can make the following proposition

Proposition 5.1.

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

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:

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

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

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

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. Susbtituting 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 $\tilde{\kappa}$. \square

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:

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

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

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

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 pursue of mathematical beauty, we look for alternative expression for the quantity $\|A\| \|T_k^{-1}\|$ giving us:

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

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

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 Lanczos 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).

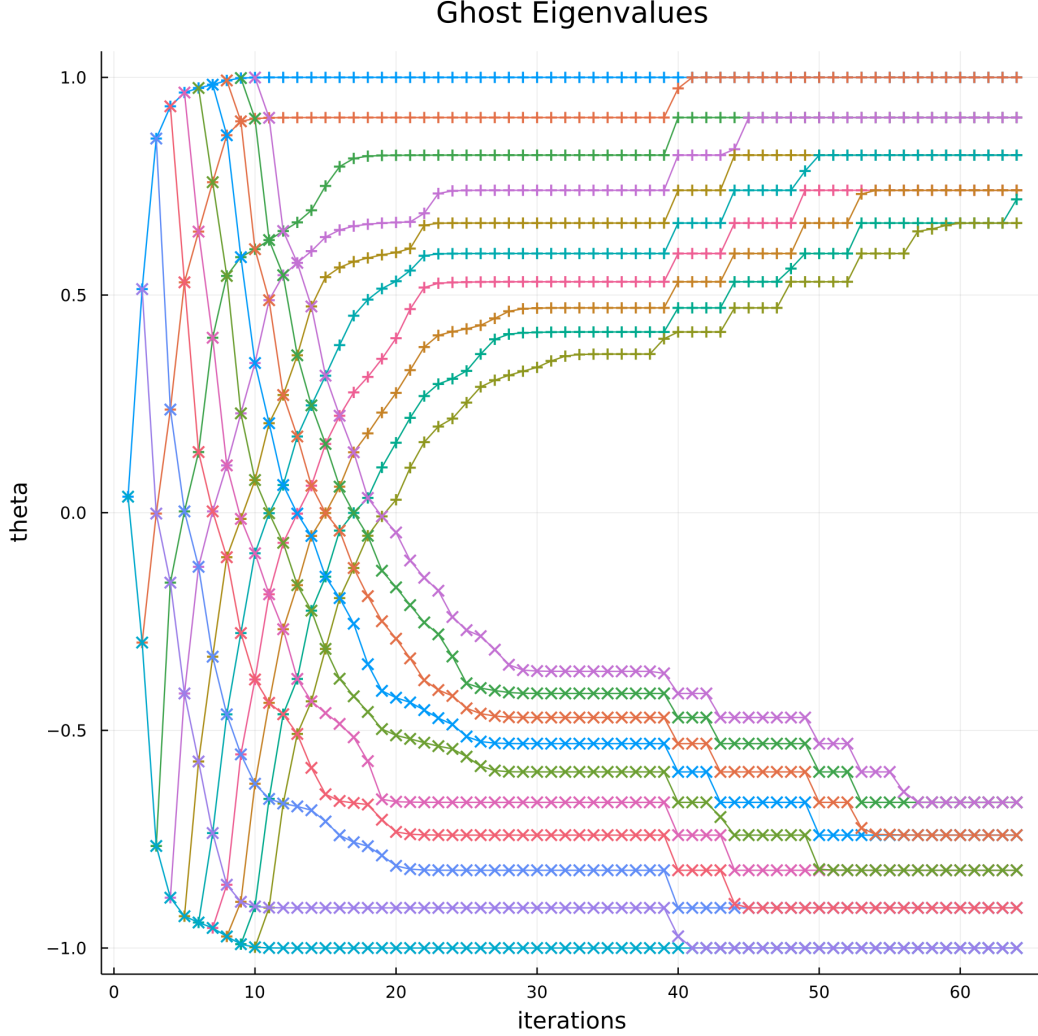


Figure 2: The highest and lowest 10 eigenvalues of the matrix T_k during the Lanczos Iterations are being tracked by their ranking, and plotted with their trajectories.

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 ghost 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 iterations 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.

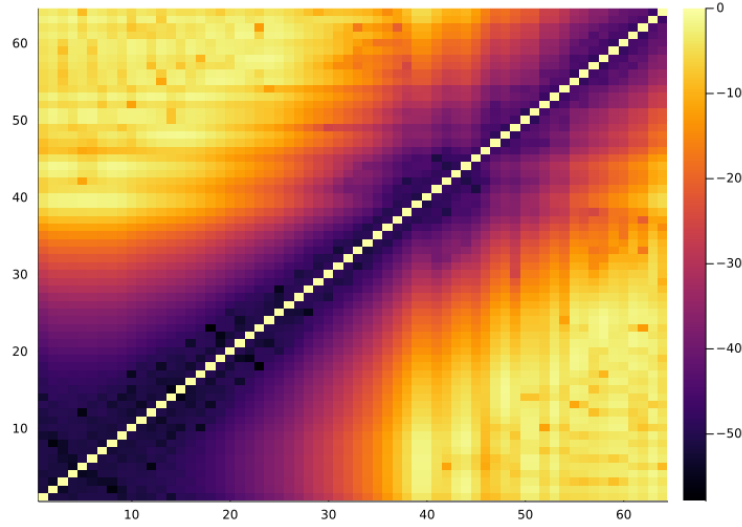


Figure 3: The heatmap of the plot of log2 of the absolute values of the matrix $\tilde{Q}_k^H \tilde{Q}_k$.

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 .

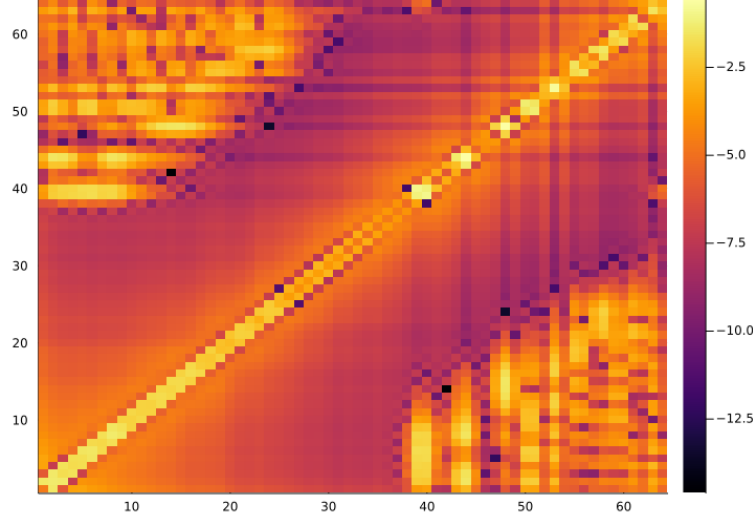


Figure 4: The plot of the T_k matrix as reproduced by $\tilde{Q}_k A \tilde{Q}_k$

Remark 5.3.3. As a final remark for these numerical experiments, I suggest an intuitive way of understanding them. Which will be useful when we actually wish to analyze it rigorously. Simply put, the Lanczos Iterations might “forget” about the eigenvalues when it converged (Manifested as the usually stable trajectories of eigenvalues on the exterior 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 correspondingly, which then causes the interior eigenvalues of T_k to shift, creating ghost eigenvalues for the matrix T_k .

5.4 The Ritz Vector’s View and Another Paige’s Theorem

In this section, we prove another analysis of the Lanczos Iterations from Dammal’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 systematic. The second is that the projection of the Lanczos vector onto the converged ritz vectors can be numerically attained, which tells us how much lost of orthogonality is occurring and which direction we need to re-orthogonalize so that the Lanczos vectors retains orthogonality. Here is the statement of the theorem:

Theorem 9 (Another Paige’s Theorem).

5.5 Forward Error Analysis

Lemma .0.1 (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 $A^{1/2}A^{1/2}$: \square

$$\lambda_1\|x\| \leq \|Ax\| \leq \lambda_n\|x\| \quad (.0.1)$$

$$\sqrt{\lambda_1}\|x\| \leq \|A^{1/2}x\| \leq \sqrt{\lambda_n}\|x\| \quad (.0.2)$$

$$\implies \sqrt{\lambda_1} \leq \frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n} \quad (.0.3)$$

Consider another vector y :

$$\sqrt{\lambda_1} \leq \frac{\|Ay\|}{\|A^{1/2}y\|} \leq \sqrt{\lambda_n} \quad (.0.4)$$

Combining the two we have:

$$\sqrt{\lambda_1} \frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n} \sqrt{\lambda_1} \quad (.0.5)$$

$$\sqrt{\lambda_1} \sqrt{\lambda_n} \geq \sqrt{\lambda_n} \frac{\|Ay\|}{\|A^{1/2}y\|} \quad (.0.6)$$

$$\implies \sqrt{\lambda_1} \frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\lambda_n} \frac{\|Ay\|}{\|A^{1/2}y\|} \quad (.0.7)$$

$$\frac{\|Ax\|}{\|A^{1/2}x\|} \leq \sqrt{\kappa(A)} \frac{\|Ay\|}{\|A^{1/2}y\|} \quad (.0.8)$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)} \frac{\|A^{1/2}x\|}{\|A^{1/2}y\|} \quad (.0.9)$$

$$\frac{\|Ax\|}{\|Ay\|} \leq \sqrt{\kappa(A)} \frac{\|x\|_A}{\|y\|_A} \quad (.0.10)$$