# Contents

## Notations

1. $\text{ran}(A) := \{Ax : \forall x \ \in \mathbb{R}^n\}, A \in \mathbb{R}^{m\times n}$, The range of a matrix.
2. $(A)_{i,j}$: The element in i th row and jth column of the matrix $A$.
3. $(A)_{i:i',j:j'}$: The submatrix whose top left corner is the $(i, j)$ element in matrix $A$, and whose' right bottom corner is the $(i', j')$ element in the matrix $A$. The notation is similar to matlab's rules for indexing.
4. $\forall\ 0 \le j \le k$: under certain context it indicates the range for an index: $j = 0, 1, \cdots, k - 1, k$
5. Boldface $\mathbf{0}$ denotes the zero vector or matrix, depending on the context it can be either a zero row/column vector, or a zero matrix.
6. The $\hat{}$ decorator is reserved for denoting the unit vector of some vector. For example $\hat{x} := x/\|x\|$.

## Introduction

# 1 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... (citations here)

## 1.1 Projectors

The properties of projector have importance for subspace projection method. In this section, we go through 2 types of projectors, the orthgonal 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:

$$P^2 = P \tag{1.1.1}$$

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

*Proof.*

$$\begin{aligned} \forall x \in \mathbb{C}^n : P(I-P)x = \mathbf{0} &\implies \operatorname{ran}(I-P) \subseteq \operatorname{null}(P) \qquad (1.1.2)\\ \forall x \in \operatorname{null}(P) : Px = \mathbf{0} &\implies (I-P)x = x \implies x \in \operatorname{ran}(I-P) \qquad (1.1.3)\\ &\implies \operatorname{ran}(I-P) = \operatorname{null}(P) \qquad (1.1.4) \end{aligned}$$

□

### 1.1.1 Orthogonal Projector

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

$$\operatorname{null}(P) \perp \operatorname{ran}(P) \tag{1.1.5}$$

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 $M = \operatorname{ran}(P)$. Consider the following lemma:

**Lemma 1.1.1.**

$$\operatorname{null}(P^H) = \operatorname{ran}(P)^\perp \tag{1.1.6}$$

$$\operatorname{null}(P) = \operatorname{ran}(P^H)^\perp \tag{1.1.7}$$

Using (1.1.4) and consider the proof:

*Proof.*

$$\langle P^H x, y\rangle = \langle x, Py\rangle \tag{1.1.8}$$
$$\forall x \in \text{null}(P^H), y \in \mathbb{C}^n \tag{1.1.9}$$
$$\implies \langle P^H x, y\rangle = 0 = \langle x, Py\rangle \tag{1.1.10}$$
$$\implies \text{null}(P^H) \perp \text{ran}(P) \tag{1.1.11}$$
$$\forall y \in \text{null}(P), x \in \mathbb{C}^n : \tag{1.1.12}$$
$$\langle x, Py\rangle = 0 = \langle P^H x, y\rangle \tag{1.1.13}$$
$$\implies \text{ran}(P^H) \perp \text{null}(P) \tag{1.1.14}$$

□

**Proposition 1.1.** A projector is orthogonal iff it's Hermitian.

*Proof.* $\Longleftarrow$ Assuming the matrix is Hermitian and it's a projector, then we wish to prove that it's an orthogonal projector. Let's recall:

$$\text{null}(P^H) = \text{ran}(P)^\perp \tag{1.1.15}$$
$$\text{null}(P) = \text{ran}(P^H)^\perp \tag{1.1.16}$$

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:

$$\text{null}(P) \perp \text{ran}(P) \tag{1.1.17}$$
$$\text{null}(P^H) \perp \text{ran}(P^H) \tag{1.1.18}$$

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. □

### 1.1.2 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.**

$$Px \in M \quad (I - P)x \perp L \quad \text{where: } M \neq L \tag{1.1.19}$$

An orthgonal 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.

### 1.1.3 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 $M$.

### 1.1.4 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: $A(AA^T)^{-1}A^T$. Let's consider the claim:

$$\|P_M x\|^2 \leq \|x\|^2 \tag{1.1.20}$$

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

$$x = Px + (I - P)x \tag{1.1.21}$$
$$\|x\|^2 = \|Px\|^2 + \|(I - P)x\|^2 \tag{1.1.22}$$
$$\|x\|^2 \geq \|Px\|^2 \tag{1.1.23}$$

□

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

$$\min_{x \in M} \|y - x\|_2^2 = \|y - Py\|_2^2 \tag{1.1.24}$$

Proof:

$$\|y - x\|_2^2 = \|y - Py + Py - x\|_2^2 \tag{1.1.25}$$
$$\|y - x\|_2^2 = \|y - Py\|_2^2 + \|Py - x\|_2^2 \tag{1.1.26}$$
$$\implies \|y - Py\|_2^2 \leq \|y - x\|_2^2 \tag{1.1.27}$$

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.

## 1.2 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:

$$\tilde{x} \in x_0 + \mathcal{K} \text{ s.t: } b - A\tilde{x} \perp \mathcal{L} \tag{1.2.1}$$

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.

$$\text{Let } V \in \mathbb{C}^{n\times m} \text{ be a aasis for: } \mathcal{K} \tag{1.2.2}$$

$$\text{Let } W \in \mathbb{C}^{n\times m} \text{ be a basis for: } \mathcal{L} \tag{1.2.3}$$

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

$$\tilde{x} = x^{(0)} + Vy \tag{1.2.4}$$

$$b - A\tilde{x} \perp \operatorname{ran}(W) \tag{1.2.5}$$

$$W^T(b - A\tilde{x} - AVy) = \mathbf{0} \tag{1.2.6}$$

$$W^T r^{(0)} - W^T AVy = \mathbf{0} \tag{1.2.7}$$

$$W^T AVy = W^T r^{(0)} \tag{1.2.8}$$

### 1.2.1 Prototype Algorithm

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

$$\begin{aligned}
&\text{While not converging :}\\
&\quad \text{Increase Span for: } \mathcal{K}, \mathcal{L}\\
&\quad \text{Choose: } V, W \text{ for } \mathcal{K}, \mathcal{L}\\
&\quad r := b - Ax\\
&\quad y := (W^T AV)^{-1} W^T r\\
&\quad x := x + Vy
\end{aligned} \tag{1.2.9}$$

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

### 1.2.2 Energy Norm Minimization using Gradient

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

$$\min_{x\in x_0+\mathcal{K}} \|b - Ax\|_B^2 \tag{1.2.10}$$

$$= \min_{w\in\mathbb{R}^k} \|b - A(x_0 + Kw)\|_B^2 \tag{1.2.11}$$

$$= \min_{w\in\mathbb{R}^k} \|r_0 - AKw\|_B^2 \tag{1.2.12}$$

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

$$\nabla_w\left[\|r_0 - AKx\|_B^2\right] = \mathbf{0} \tag{1.2.13}$$
$$(AK)^T B(r_0 - AKx) = \mathbf{0} \tag{1.2.14}$$
$$(AK)^T Br_0 - (AK)^T BAKx = \mathbf{0} \tag{1.2.15}$$
$$(AK)^T Br_0 = (AK)^T BAKx \tag{1.2.16}$$

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

**Proposition 1.2** (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 $b - Ax_k$.

$$\min_{x_k \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 \iff K^T Be_k - K^T BKw = \mathbf{0} \tag{1.2.17}$$

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

*Proof.*

$$\min_{x \in x_0 + \mathcal{K}} \|A^{-1}b - x\|_B^2 = \min_{x \in \mathbb{R}} \|A^{-1}b - x_0 - Kw\|_b^2 \tag{1.2.18}$$
$$= \min_{x \in \mathbb{R}^k} \|e_k - Kw\|_b^2 \tag{1.2.19}$$

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

$$\mathbf{0} = \nabla_w[\|e_k - Kw\|_B^2] \tag{1.2.20}$$
$$= \nabla_w[e_k - Kw]^T B(e_k - Kw) \tag{1.2.21}$$
$$= 2K^T B(e_k - Kw) \tag{1.2.22}$$
$$\implies K^T Be_k - K^T BKw = \mathbf{0} \tag{1.2.23}$$

□

This conditions is implicity 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:

$$K^T AA^{-1}r_k - K^T AKw = \mathbf{0} \tag{1.2.24}$$
$$K^T r_k - K^T AKw = \mathbf{0} \tag{1.2.25}$$

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

## 1.3 Krylov Subspace

A Krylov Subspace is a sequence basis paramaterized by $A$, an linear operator, $v$ an initial vector, and $k$, which basis in the sequece of basis that we are looking at.

**Definition 4** (Krylov subspace).

$$\mathcal{K}_k(A|b) = \text{span}(b, Ab, A^2b, \cdots A^{k-1}b) \tag{1.3.1}$$

Please immediately observe that from the definition we have:

$$\forall v : \mathcal{K}_1(A|v) \subseteq \mathcal{K}_2(A|v) \subseteq \mathcal{K}_3(A|v) \cdots \tag{1.3.2}$$

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

$$\forall x \in \mathcal{K}_k(A|v) \; \exists \; w : p_k(A|w)v = x \tag{1.3.3}$$

We use $p_k(A|w)$ to denotes 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.

$$p_k(A|w)v = \sum_{j=0}^{k-1} w_j A^j v \tag{1.3.4}$$

### 1.3.1 The Grade of a Krylov Subspace

The most important porperty of the subspace is the idea of grade denoted as 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 1.3.1** (Grade Lemma 1).

$$\exists 1 \le k \le m + 1 : \mathcal{K}_k(A|v) = \mathcal{K}_{k+1}(A|v) \tag{1.3.5}$$

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

**Lemma 1.3.2** (Grade Lemma 2).

$$\exists ! k \text{ s.t: } \mathcal{K}_k(A|v) = \mathcal{K}_{k+1}(A|v) \implies \mathcal{K}_k(A|v) \text{ is Lin Ind} \wedge \mathcal{K}_{k+1}(A|v) \text{ is Lin Dep.} \tag{1.3.6}$$

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

**Lemma 1.3.3** (Grade Lemma 3)**.**

$$\mathcal{K}_k(A|v) \text{ Lin Dep} \implies \mathcal{K}_{k+1}(A|v) = \mathcal{K}_k(A|v) \tag{1.3.7}$$

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 minumum number when the krylov subspace becomes linear dependent, then all successive krylov subspace spand 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 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:

$$\forall\, k : \mathcal{K}_k \subseteq \mathcal{K}_{k+1} \implies \dim(\mathcal{K}_k) \leq \dim(\mathcal{K}_{k+1}) \tag{1.3.8}$$

$$\mathcal{K}_{k+1} \setminus \mathcal{K}_k = \operatorname{span}(A^k v) \tag{1.3.9}$$

$$\implies \dim(\mathcal{K}_{k+1}) - \dim(\mathcal{K}_k) \leq 1 \tag{1.3.10}$$

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 occurance will exist), implying the the fact that the new added vector from $k$ to $k+1$ is in the span of the previous subspace. □

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

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

$$\begin{aligned}
&\mathcal{K}_{k+1}(A|v) \text{ is Lin dependent} \\
&\implies \exists w^{(k)} : A^k v = p_k(A|w^{(k)})v \\
&x \in \mathcal{K}_{k+1}(A|v) \iff \exists w^{(k+1)} : p_{k+1}(A|w^{(k+1)})v = x \\
&x = w^{(k+1)} A^k v + \sum_{j=0}^{k-1} w_j^{(k+1)} A^j v \\
&x = w_k^{(k+1)} p_k(A|w^{(k)})v + \sum_{j=0}^{k-1} w_j^{(k+1)} A^j v \\
&x \in \mathcal{K}_k(A|v)
\end{aligned}$$

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 $\mathcal{K}_k(A|v)$. □

*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. □

### 1.3.2 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:

$$\exists w \neq \mathbf{0} : p_k(A|w)v = \mathbf{0} \tag{1.3.11}$$

$$\implies w_0 v + \sum_{j=1}^{k-1} w_j A^j v = \mathbf{0} \tag{1.3.12}$$

$$\mathbf{0} = \sum_{j=1}^{k-1} w_j A^j v \tag{1.3.13}$$

$$\mathbf{0} = A \sum_{j=0}^{k-2} w_{j+1} A^j v \tag{1.3.14}$$

$$\implies \sum_{j=0}^{k-2} w_{j+1} A^j v = \mathbf{0} \tag{1.3.15}$$

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, $w_0 \neq 0$. □

## 1.4 Useful Theorems and Mathematical Entities

### 1.4.1 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:

$$\forall x : \mathbf{0} = \sum_{j=0}^{k-1} w_j A^j x \tag{1.4.1}$$

$$w_0 := 0 \tag{1.4.2}$$

$$\forall x : \mathbf{0} = \sum_{j=1}^{k-1} w_j A^j x \tag{1.4.3}$$

$$\forall x : \mathbf{0} = A \sum_{j=0}^{k-2} w_j A^j x \tag{1.4.4}$$

$$\implies \forall x : \mathbf{0} = \sum_{j=0}^{k-2} w_j A^j x \tag{1.4.5}$$

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.

### 1.4.2 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:

$$\theta_j^{(k+1)} \le \theta_j^{(k)} \le \theta_{j+1}^{(k+1)} \quad \forall 1 \le j \le k-1 \tag{1.4.6}$$

### 1.4.3 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.

### 1.4.4 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 infinitty norm over a closed interval:

$$T_k(x) = \arg\min_{p(x)} \{\|p(x)\|_\infty : \forall x \in [-1, 1]\} \tag{1.4.7}$$

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

## 1.5 Deriving Conjugate Gradient from First Principles

### 1.5.1 CG Objective and Framework

We introduce the algorithm as an attempt to minimize the energy norm of the error for a linear equation $Ax = b$, here 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.

$$\min_{w \in \mathbb{R}^k} \|A^{-1}b - (x_0 + P_k w)\|_A^2 \iff P_k^T r_0 = P_k^T A P_k w \tag{1.5.1}$$

Refer back to (1.4) for how to deal with the above minimization objective. Using the matrix form for the Petrov Galerkin Conditions where $W, V$ are both $P_k$, we reformulate the Norm Minimizations conditions under the framework of Petrov Galarkin conditions:

$$\text{choose: } x \in x_0 + \text{ran}(P_k) \text{ s.t: } b - Ax \perp \text{ran}(P_k) \tag{1.5.2}$$

Take note that the link between a norm minimzation and an equivalent subspace Orthogonality conditions don't garantee 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 matrx and hence we let $P_k$ to be a *matrix whose columns are A-Orthogonal vectors.*

$$P_k^T A P_k = D_k \text{ where: } (D_k)_{i,i} = \langle p_{i-1}, Ap_{i-1} \rangle \tag{1.5.3}$$

$$P_k r_0 = P_k^T A P_k w = D_k w \tag{1.5.4}$$

$$w = D_k^{-1} P_k^T r_0 \tag{1.5.5}$$

The idea here is: Accumulating vectors $p_j$ into the matrix $P_k$ and then iterative improve the solution $x_k$, by reducing the error denote as $e_k$ which is 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 $r_k = b - Ax_k$ for the algorithm:

$$\begin{cases} x_k = x_0 + P_k D_k^{-1} P_k^T r_0 \\ r_k = r_0 - AP_k D_k^{-1} P_k^T r_0 \\ P_k^T A P_k = D_k \end{cases} \tag{1.5.6}$$

Let this algorithm be the prototype.

### 1.5.2 Using the Projector

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

*Proof.*

$$
\begin{aligned}
AP_kD_k^{-1}P_k^T(AP_kD_k^{-1}P_k^T) &= AP_kD_k^{-1}P_k^TAP_kD_k^{-1}P_k^T && (1.5.7)\\
&= AP_kD_k^{-1}D_kD_k^{-1}P_k^T && (1.5.8)\\
&= AP_kD_k^{-1}P_k^T && (1.5.9)
\end{aligned}
$$

$$
\begin{aligned}
P_kD_k^{-1}P_k^TA(P_kD_k^{-1}P_k^TA) &= P_kD_k^{-1}D_kD_k^{-1}P_k^TA && (1.5.10)\\
&= P_kD_k^{-1}P_k^TA && (1.5.11)
\end{aligned}
$$

□

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

$$
\begin{aligned}
AP_kD_k^{-1}P_k^T &= A\overline{P}_k && (1.5.12)\\
P_kD_k^{-1}P_k^TA &= \overline{P}_kA && (1.5.13)
\end{aligned}
$$

One immediate consequence is:

$$
\begin{aligned}
&\text{ran}(I - A\overline{P}_k) \perp \text{ran}(P_k) && (1.5.14)\\
&\text{ran}(I - \overline{P}_kA) \perp \text{ran}(AP_k) && (1.5.15)
\end{aligned}
$$

*Proof.*

$$
\begin{aligned}
P_k^T(I - A\overline{P}_k) &= P_k^T - P_k^TA\overline{P}_k && (1.5.16)\\
&= P_k^T - D_kD_k^{-1}P_k^T && (1.5.17)\\
&= \mathbf{0} && (1.5.18)\\
(AP_k)^T(I - \overline{P}_kA) &= P_k^TA - P_k^TA\overline{P}_kA && (1.5.19)\\
&= P_k^TA - P_k^TAP_kD_k^{-1}P_k^TA && (1.5.20)\\
&= P_k^TA - P_k^TA && (1.5.21)\\
&= \mathbf{0} && (1.5.22)
\end{aligned}
$$

□

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

**Proposition 1.3** (Residuals are Orthogonal to $P_k$)**.**

$$
\begin{aligned}
&r_k = r_0 - A\overline{P}_kr_0 = (I - A\overline{P}_k)r_0 && (1.5.23)\\
\implies &r_k \perp \text{ran}(P_k) && (1.5.24)
\end{aligned}
$$

**Proposition 1.4** (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:

$$p_k = (I - \overline{P}_k A)u_k \tag{1.5.25}$$

$$\operatorname{span}(p_k) \perp \operatorname{ran}(AP_k) \tag{1.5.26}$$

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

### 1.5.3 Assisted Conjugate Gradient

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:

$$\begin{cases} x_k = x_0 + \overline{P}_k r_0 \\ r_k = (I - A\overline{P}_k)r_0 \\ P_k^T A P_k = D_k \\ \overline{P}_k = P_k D_k^{-1} P_k^T \\ p_k = (I - \overline{P}_k A)u_k \quad \{u_i\}_{i=0}^{n-1} \text{ is a Basis} \end{cases} \tag{1.5.27}$$

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

### 1.5.4 Properties of Assisted Conjugate Gradient

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

**Proposition 1.5.**

$$p_{k+j}^T r_k = p_{k+j}^T r_0 \quad \forall\, 0 \le j \le n-k \tag{1.5.28}$$

$$\begin{aligned} p_{k+j}^T r_k &= p_k^T (I - A\overline{P}_k) r_0 && (1.5.29) \\ &= (p_{k+j}^T - p_{k+j}^T A\overline{P}_k) r_0 && (1.5.30) \\ &= p_{k+j}^T r_0 && (1.5.31) \end{aligned}$$

This made the recurrence between successive residual from the ACG possible.

Next, we wish to use this property to find out a recurrences for the residuals of ACG, and here is how we do it:

$$
\begin{aligned}
r_k - r_{k-1} &= r_0 - A\overline{P}_k r_0 - (r_0 - A\overline{P}_{k-1} r_0) && (1.5.32)\\
&= A\overline{P}_k r_0 - A\overline{P}_{k-1} r_0 && (1.5.33)\\
&= -Ap_{k-1}\frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (1.5.34)\\
\implies x_k - x_{k-1} &= p_{k-1}\frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (1.5.35)\\
\text{def: } a_{k-1} &:= \frac{\langle p_{k-1}, r_0\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} = \frac{\langle p_{k-1}, r_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} && (1.5.36)
\end{aligned}
$$

We define the value of $a_{k-1}$, and in above, we have 2 equivalent representation. Please take note that, Proposition still remains true for the ACG algorithm we just developed here.

### 1.5.5 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 orthgonal. We wish to proceed to prove that part.

**Lemma 1.5.1.**

$$
\langle p_{k+j}, Ap_k\rangle = \langle r_k, Ap_{k+j}\rangle = \langle p_{k+j}, Ar_k\rangle \quad \forall\, 0 \le j \le n-k \tag{1.5.37}
$$

*Proof.*

$$
\begin{aligned}
p_{k+j}Ap_k &= p_{k+j}^T Ar_k - p_{k+j}^T A\overline{P}_k Ar_k \quad \forall\, 0 \le j \le n-k && (1.5.38)\\
&= p_{k+j}^T Ar_k && (1.5.39)\\
\langle p_{k+j}, Ap_k\rangle &= \langle r_k, Ap_{k+j}\rangle = \langle p_{k+j}, Ar_k\rangle && (1.5.40)
\end{aligned}
$$

□

**Lemma 1.5.2.**

$$
\langle r_k, p_k\rangle = \langle r_k, r_k\rangle \tag{1.5.41}
$$

*Proof.*

$$
\begin{aligned}
\langle r_k, p_k\rangle &= \langle r_k, p_k\rangle && (1.5.42)\\
&= \langle r_k, r_k\rangle - \langle r_k, \overline{P}_k Ar_k\rangle && (1.5.43)\\
&= \langle r_k, r_k\rangle && (1.5.44)
\end{aligned}
$$

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

**Proposition 1.6** (Residual Assisted CG Generates Orthogonal Residuals)**.**

$$\langle r_k, r_j \rangle = 0 \quad \forall\, 0 \le j \le k-1 \tag{1.5.45}$$

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

*Proof.*

$$\begin{aligned} r_{k+1} &= r_k - a_k A p_k && (1.5.46)\\ \implies \langle r_{k+1}, r_k \rangle &= \langle r_k, r_k \rangle - a_k \langle r_k, A p_k \rangle && (1.5.47)\\ &= \langle r_k, r_k \rangle - \frac{\langle r_k, r_k \rangle}{\langle p_k, A p_k \rangle} \langle r_k, A p_k \rangle && (1.5.48)\\ &= 0 && (1.5.49) \end{aligned}$$

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

$$\begin{aligned} p_j &= (I - \overline{P}_j A) r_j \quad \forall\, 0 \le j \le k-1 && (1.5.50)\\ r_j &= p_j + \overline{P}_j A r_j && (1.5.51)\\ r_k &= (I - A\overline{P}_k) P_0 && (1.5.52)\\ r_k \perp \operatorname{ran}(P_k) \implies \langle r_k, r_j \rangle &= \langle r_k, p_j + \overline{P}_j A r_j \rangle = 0 && (1.5.53) \end{aligned}$$

Here we again make use of the projector $I - A\overline{P}_k$. The base case of the argument is simple, because $p_0 = r_0$, and by the property of the projector, $\langle r_1, r_0 \rangle = 0$. The theorem is now proven. □

**Proposition 1.7** (RACG Recurrences)**.**

$$p_k = r_k + b_{k-1} p_{k-1} \quad b_{k-1} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \tag{1.5.54}$$

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

*Proof.*

$$\begin{aligned} p_k &= (I - \overline{P}_k A) r_k && (1.5.55)\\ r_k - \overline{P}_k A r_k &= r_k - P_k D_k^{-1} P_k^T A r_k && (1.5.56)\\ &= r_k - P_k D_k^{-1} (A P_k)^T r_k && (1.5.57) \end{aligned}$$

Observe that the term $(AP_k)^T$ can be expanded and we can make use of the Symmetric Property of the operator $A_k$.

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, A r_k \rangle \\ \langle p_1, A r_k \rangle \\ \vdots \\ \langle p_{k-1}, A r_k \rangle \end{bmatrix} \tag{1.5.58}$$

Next, we can make use of Lemma 2 to get rid of $Ar_k$. Please consider:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} \tag{1.5.59}$$

The second line is using the property that the matrix $A$ is symmetric, the third line is using the recurrence of the residual of ACG, and the last line is true for all $0 \leq j \leq k-2$ by the orhogonality of the residual proved in Claim 1. Therefore we have:

$$(AP_k)^T r_k = \begin{bmatrix} \langle p_0, Ar_k \rangle \\ \langle p_1, Ar_k \rangle \\ \vdots \\ \langle p_{k-1}, Ar_k \rangle \end{bmatrix} = a_{k-1}^{-1} \langle r_k, (r_{k-1} - r_k) \rangle \xi_k \tag{1.5.60}$$

Take note that the vector $\xi_k$ is the k th standard basis vector in $\mathbb{R}^k$. And using this we can simplify the expression for $p_k$ into:

$$p_k = r_k - P_k D_k^{-1} (AP_k)^T r_k \tag{1.5.61}$$

$$= r_k - P_k D_k^{-1} a_{k-1}^{-1} (\langle r_k, (r_{k-1} - r_k) \rangle) \xi_k \tag{1.5.62}$$

$$= r_k - \frac{a_{k-1}^{-1} \langle -r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{1.5.63}$$

$$= r_k + \frac{a_{k-1}^{-1} \langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{1.5.64}$$

$$= r_k + \left( \frac{\langle r_{k-1}, r_{k-1} \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \right)^{-1} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} p_k \tag{1.5.65}$$

$$= r_k + \frac{\langle r_k, r_k \rangle}{\langle r_{k-1}, r_{k-1} \rangle} p_k \tag{1.5.66}$$

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 $p_k$. □

Up until this point we have proven the usual version of conjugate gradient, 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 equalto the set of residuals vector generated from the algorithm itself; after some proof, 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 6** (RACG).

$$p^{(0)} = b - Ax^{(0)} \tag{1.5.67}$$

$$\text{For } i = 0, 1, \cdots \tag{1.5.68}$$

$$\begin{aligned} a_i &= \frac{\|r^{(i)}\|^2}{\|p^{(i)}\|_A^2} \\ x^{(i+1)} &= x^{(i)} + a_i p^{(i)} \\ r^{(i+1)} &= r^{(i)} - a_i A p^{(i)} \\ b_i &= \frac{\|r^{(j+1)}\|_2^2}{\|r^{(i)}\|_2^2} \\ p^{(i+1)} &= r^{(i+1)} + b_i p^{(i)} \end{aligned} \tag{1.5.69}$$

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.

### 1.5.6 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 excerp. The full algorithm can be seem by the short recurrence for the residual and the conjugation vector. This part is trivial. Next, we want to show the relations to the Krylov Subspace, which only occurs for the Residual Assisted Conjugate Gradient algorithm.

**Proposition 1.8.**

$$p_k \in \mathcal{K}_{k+1}(A|r_0) \tag{1.5.70}$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \tag{1.5.71}$$

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

$$r_{k+1} = r_k - a_k A p_k \tag{1.5.72}$$

$$\in r_k + A\mathcal{K}_{k+1}(A|r_0) \tag{1.5.73}$$

$$\in r_k + \mathcal{K}_{k+2}(A|r_0) \tag{1.5.74}$$

$$r_k \in \mathcal{K}_{k+1}(A|r_0) \subseteq \mathcal{K}_{k+2}(A|r_0) \tag{1.5.75}$$

$$\implies r_{k+1} \in \mathcal{K}_{k+2}(A|r_0) \tag{1.5.76}$$

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

$$p_{k+1} = r_{k+1} + b_k p_k \tag{1.5.77}$$

$$\in r_{k+1} + \mathcal{K}_{k+1}(A|r_0) \tag{1.5.78}$$

$$\in \mathcal{K}_{k+2}(A|r_0) \tag{1.5.79}$$

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)**.**

$$\text{choose: } x_k \in x_0 + \mathcal{K}_k(A|r_0) \text{ s.t: } r_k \perp \mathcal{K}_k(A|r_0) \tag{1.5.80}$$

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. □

### 1.5.7 A Remark on Its Importance

STILL WORKING ON THIS PART.

## 1.6 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 amphasize that Lanczos Iterations is just Arnoldi but with the matrix $A$ being a symmetric matrix. Finally we make the link between Lanczos Iterations and Krylov Subspace, which wil inevitably linked back to RACG and plays an important role for the analysis of RACG.

### 1.6.1 The Arnoldi Iterations

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

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

$$\tilde{H}_k = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{1,2} & h_{2,2} & \cdots & h_{2,k} \\ & \ddots & & \vdots \\ & & h_{k,k-1} & h_{k,k} \\ & & & h_{k+1,k} \end{bmatrix} \tag{1.6.1}$$

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

$$(\tilde{H}_k)_{2,1}q_2 = (I - q_1q_1^H)Aq_1 \tag{1.6.2}$$
$$(\tilde{H}_k)_{1,1}q_1 = q_1q_1^H Aq_1 \tag{1.6.3}$$
$$Q_2 := \begin{bmatrix} q_1 & q_2 \end{bmatrix} \tag{1.6.4}$$
$$(\tilde{H}_k)_{3,2}q_3 = (I - Q_2Q_2^H)Aq_2 \tag{1.6.5}$$
$$(\tilde{H}_k)_{1:2,2} = Q_2Q_2^H Aq_2 \tag{1.6.6}$$
$$Q_3 := \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \tag{1.6.7}$$
$$\vdots \tag{1.6.8}$$
$$Q_j := \begin{bmatrix} q_1 & q_2 & \cdots & q_j \end{bmatrix} \tag{1.6.9}$$
$$(\tilde{H}_k)_{j+1,j}q_{j+1} = (I - Q_jQ_j^H)Aq_j \tag{1.6.10}$$
$$(\tilde{H}_k)_{1:j,j} = Q_jQ_j^H Aq_j \tag{1.6.11}$$
$$\vdots \tag{1.6.12}$$
$$Q_k := \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{1.6.13}$$
$$(\tilde{H}_k)_{k+1,k}q_{k+1} = (I - Q_kQ_k^H)Aq_k \tag{1.6.14}$$
$$(\tilde{H}_k)_{1:k,k} = Q_kQ_k^H Aq_k \tag{1.6.15}$$

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

$$AQ_k = Q_{k+1}\tilde{H}_k \tag{1.6.16}$$
$$Q_k^H AQ_k = H_k \tag{1.6.17}$$

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 $Q_k^H AQ_k$ is also symmetric, which will make $H_k$ to be symmetric as well, which implies that $H_k$ will be a Symmetric Tridiagonal Matrix. And under that assumption, we can develop the Lanczos Algorithm. Instead of orthogoalizing against all previous vectors, we have the option to simply orthogonalize against the previous $q_k, q_{k-1}$ vector. And we can reuse the sub-diagonal elements for $q_{k-1}$; giving us the Lanczos Algorithm.

### 1.6.2 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 1.9.**

$$\text{ran}(Q_k) = \mathcal{K}_k(A|q_1) \tag{1.6.18}$$

*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:

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

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

### 1.6.3 The Lanczos Iterations

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

$$\begin{aligned}
& \text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 && (1.6.19) \\
& \text{set: } \beta_0 = 0 && (1.6.20) \\
& \text{For } j = 1, 2, \cdots && (1.6.21) \\
& \quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1}q_{j-1} \\
& \quad \alpha_j := \langle q_j, \tilde{q}_{j+1} \rangle \\
& \quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j && (1.6.22) \\
& \quad \beta_j = \|\tilde{q}_{j+1}\| \\
& \quad q_{j+1} := \tilde{q}_{j+1}/\beta_j
\end{aligned}$$

Here, let it be the case that $H_k$ is a Symmetric Tridiagonal Matrix with $\alpha_i$ on the diagonal, $\beta_i$ 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 $\beta_{j-1}$, 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 $1 \times 1$ matrix can be phrased in the following way:

$$\begin{aligned}
& \text{Given arbitrary: } q_1 \text{ s.t: } \|q_1\| = 1 \\
& \alpha_1 := \langle q_1, Aq_1 \rangle \\
& \beta_0 := 0 \\
& \text{Memorize} : Aq_1 \\
& \text{For } j = 1, 2, \cdots \\
& \quad \tilde{q}_{j+1} := Aq_j - \beta_{j-1}q_{j-1} && (1.6.23) \\
& \quad \tilde{q}_{j+1} \leftarrow \tilde{q}_{j+1} - \alpha_j q_j \\
& \quad \beta_j = \|\tilde{q}_{j+1}\| \\
& \quad q_{j+1} := \tilde{q}_{j+1}/\beta_j \\
& \quad \alpha_{j+1} := \langle q_{j+1}, Aq_{j+1} \rangle \\
& \quad \text{Memorize: } Aq_{j+1}
\end{aligned}$$

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

$$Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{1.6.24}$$

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \tag{1.6.25}$$

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)**.**

$$AQ_k = Q_k T_k + \beta_k q_{k+1}\xi_k^T = Q_{k+1}\tilde{T}_k \tag{1.6.26}$$

$$\implies Aq_k = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \quad \forall\, 2 \le j \le k \tag{1.6.27}$$

$$\implies Aq_1 = \alpha_1 q_1 + \beta_1 q_2 \tag{1.6.28}$$

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 1.10.** The Lanczos Iteration produces a Symmetric Tridiagonal Matrix that has no zero element on its super and sub-diagonal, and if $\beta_k$ is zero, then the algorithm must terminates, and $k$ would equal to grade$(A|q_1)$, the grade of the Krylov Subspace.

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

# 2 Analysis of Conjugate Gradient and Lanczos Iterations

## 2.1 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 2.1** (CG Relative Energy Error)**.**

$$
\begin{aligned}
x_k &\in x_0 + \mathcal{K}_k(A|r_0) && (2.1.1)\\
x_k &= \mathcal{K}(r_0)w + x_0 && (2.1.2)\\
\frac{\|e_k\|_A^2}{\|e_0\|_A^2} &= \min_{w\in\mathbb{R}^k} \|(I + Ap_k(A|w))A^{1/2}e_0\|_2^2 && (2.1.3)\\
&\le \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_{k+1}(x)| && (2.1.4)
\end{aligned}
$$

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

*Proof.*

$$
\begin{aligned}
\|e_k\|_A^2 &= \min_{x_k\in x_0+\mathcal{K}_k(A|r_0)} \|x^+ - x_k\|_A^2 && (2.1.5)\\
x_k \in x_0 + \mathcal{K}_k(A|r_0) \iff & \; e_k = e_0 + p_k(A|w)r_0 && (2.1.6)\\
\implies &= \min_{w\in\mathbb{R}^k} \|e_0 + p_k(A|w)r_0\|_A^2 && (2.1.7)\\
&= \min_{w\in\mathbb{R}^k} \|e_0 + Ap_k(A|w)e_0\|_A^2 && (2.1.8)\\
&= \min_{w\in\mathbb{R}^k} \|A^{1/2}(I + Ap_k(A|w))e_0\|_2^2 && (2.1.9)\\
&\le \min_{w\in\mathbb{R}^k} \|I + Ap_k(A|w)\|_2^2\|e_0\|_A^2 \quad \text{tight} && (2.1.10)\\
&= \min_{w\in\mathbb{R}^k} \left(\max_{i=1,\ldots,n} |1+\lambda_i p_k(\lambda_i|w)|^2\right) \|e_0\|_A^2 && (2.1.11)\\
&\le \min_{w\in\mathbb{R}^k} \left(\max_{x\in[\lambda_{\min},\lambda_{\max}]} |1+\lambda_i p_k(\lambda_i|w)|^2\right) \|e_0\|_A^2 \quad \text{still tight} && (2.1.12)\\
&= \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_{k+1}(x)|^2\|e\|_A^2 && (2.1.13)\\
\implies \frac{\|e_k\|_A}{\|e_0\|_A} &\le \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_{\min},\lambda_{\max}]} |p_{k+1}(x)| && (2.1.14)
\end{aligned}
$$

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 $\Lambda$ are the eigenvalues of $A$. Allow me to explain:

$$
\begin{aligned}
\|I + Ap_k(A|w)\|_2^2 &= \|Q(I + \Lambda p_k(\Lambda|w))Q^T\|_2^2 && (2.1.15)\\
&= \|I + \Lambda p_k(\Lambda|w)\|_2^2 && (2.1.16)\\
&= \max_{i=1,\cdots,n} |1+\lambda_i p_k(\lambda_i|w)|^2 && (2.1.17)
\end{aligned}
$$

Where, the 2-norm of a diagonal matrix $\Lambda$ is just its biggest diagonal element. And then we relax the conditions for $\lambda_i$ 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 thta we use an monic $p_{k+1}(x)$ at the and to simplify things. □

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

## 2.2 Termination Conditions of RACG

Under exact arithememtic, 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 Subpsace, 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 2.2.** The 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 grade$(A|r_0)$ is $k+1$, then $\mathcal{K}_k(A|r_0) = \mathcal{K}_{k=1}(A|v)$, 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 $r_j \perp r_j \;\forall\; 0 \le j \le k-1$. 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 Susbapce 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 heppens, it will result in $b_{k-1}$ being zero because of CG, which will give $p_k = \mathbf{0}$. Which will terminaete 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 Susbspace discussion, when the grade is reached, there exists non trivial polynomial expression where:

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

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 monoic 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 terminates 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 number of iterations required.

## 2.3 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 is idea is to use a Chebyshev Polynomial to establish a bound and it's applicable when the linear operator has extremely high dimension and we limit the number of iterations to $k$ where $k$ is much smaller than $n$, the size of the matrix. We will follow Greenbaum's Analysis but with some more details.

### 2.3.1 Uniformly Distributed Eigenvalues

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

$$\frac{\|e_k\|_A}{\|e_k\|_A} \le 2\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k \tag{2.3.1}$$

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

Before the proof, I need to point out the analysis draws inspiration from the interpolating mononic polynomial for the spectrum of the matrix $A$, and we make use of the Inf Norm minimization property of the Chebyshev Polynomial. Here, we order all the eigenvalues of matrix $A$ so that $\lambda_1, \lambda_n$ 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:

$$T_k(x) = \min_{\substack{p(x)\in\mathcal{P}_{k+1} \\ \text{s.t: } p(0)=1}} \max_{x\in[-1,1]} |p(x)| \tag{2.3.2}$$

$$p_k(x) := \frac{T_k(\varphi(x))}{T_k(\varphi(0))} \quad \text{where: } \varphi(x) := \frac{2x-\lambda_n-\lambda_1}{\lambda_n-\lambda_1} \tag{2.3.3}$$

$$\text{then: } p_k(x) = \min_{\substack{p(x)\in\mathcal{P}_{k+1} \\ \text{s.t: } p(0)=1}} \max_{x\in[\lambda_1,\lambda_n]} |p(x)| \tag{2.3.4}$$

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 Tridiagoal Symmetric Matrix from lanczos. Next, we use the property that the range of the Chebyshev is bounded withint the interval $[-1, 1]$ to obtain inequality:

$$\forall x \in [\lambda_1, \lambda_n] : \left|\frac{T_k(\varphi(x))}{T_k(\varphi(0))}\right| \le \left|\frac{1}{T_k(\varphi(0))}\right| \tag{2.3.5}$$

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 $[-1, 1]$:

$$T_k(x) = \cosh(k \ \text{arccosh}(z)) \quad \forall z \ge 1 \tag{2.3.6}$$

$$\implies T_k(\cosh(\zeta)) = \cosh(k\zeta) \quad z := \cosh(\zeta) \tag{2.3.7}$$

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 $\zeta$.

$$\varphi(0) = \cosh(\zeta) = \cosh(\ln(y)) \quad \ln(y) := \zeta \tag{2.3.8}$$

$$\text{recall: } \cosh(x) = (\exp(-x) + \exp(x))/2 \tag{2.3.9}$$

$$\implies \cosh(\ln(y)) = (y + y^{-1})/2 \tag{2.3.10}$$

$$\varphi(0) = (y + y^{-1})/2 \tag{2.3.11}$$

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

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

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

$$-\frac{\kappa + 1}{\kappa - 1} = \frac{1}{2}(y + y^{-1}) \tag{2.3.12}$$

$$y = \frac{\sqrt{\kappa} \pm 1}{\sqrt{\kappa} \mp 1} \tag{2.3.13}$$

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 susbtitute and get the value of $T_k(\varphi(0))$ interms of $y$ and then $\kappa$:

$$\varphi(0) = \frac{1}{2}(y + y^{-1}) \tag{2.3.14}$$

$$\implies T_k(\varphi(0)) = T_k(\cosh(\ln(y))) \tag{2.3.15}$$

$$= \cosh(k\ln(y)) \tag{2.3.16}$$

$$= (y^k + y^{-k})/2 \tag{2.3.17}$$

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

$$\frac{1}{T_k(\varphi(0))} = 2(y^k + y^{-k})^{-1} \tag{2.3.18}$$

$$= 2\left(\left(\frac{\sqrt{\kappa} \pm 1}{\sqrt{\kappa} \mp 1}\right)^k + \left(\frac{\sqrt{\kappa} \mp 1}{\sqrt{\kappa} \pm 1}\right)^{-k}\right)^{-1} \tag{2.3.19}$$

$$= 2\left(\underbrace{\left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^k}_{>1} + \underbrace{\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{-k}}_{<1}\right)^{-1} \tag{2.3.20}$$

$$\leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{2.3.21}$$

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

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq \min_{p_{k+1}:p_{k+1}(0)=1} \max_{x\in[\lambda_1,\lambda_n]} |p_{k+1}(x)| \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{2.3.22}$$

□

### 2.3.2 One Outlier Eigenvalue

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

**Proposition 2.3** (Big Outlier CG Convergence Rate)**.** If, there exists a $\lambda_n$ that is much later than all previous $n-1$ eigenvalues for the matrix $A$, then a tigher convergence bound that being only paramaterized by the range of clustered eigenvalues can be obtained and it is:

$$\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A} \leq 2\left(\frac{\sqrt{\kappa_{n-1}} - 1}{\sqrt{\kappa_{n-1}} + 1}\right)^{k-1} \qquad \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{2.3.23}$$

Reader please observe that, the outlier eigenvalue $\kappa_n$ plays a smaller role in determining the convergence rate of the algorithm compare to the previous bound.

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

$$\hat{T}^{(k)}_{[a,b]}(x) := T_k\left(\frac{2x - b - a}{b - a}\right) \tag{2.3.24}$$

Next, we consider the following polynomial:

$$p_k(z) := \frac{\hat{T}^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(z)}{T^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(0)} \frac{\lambda_n - z}{\lambda_n} \tag{2.3.25}$$

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 $\lambda_n$ and it's less than one at the left boundary $\lambda_1$. Next, observe the following facts about the above polynomials:

$$\frac{\lambda_n - z}{\lambda_n} \in [0, 1] \quad \forall z \in [\lambda_1, \lambda_n] \tag{2.3.26}$$

$$\frac{\lambda_n - z}{\lambda_n} < 1 \quad \forall z \in [\lambda_1, \lambda_{n-1}] \tag{2.3.27}$$

$$|p_k(x)| \leq \left| \frac{\hat{T}^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(x)}{\hat{T}^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(0)} \frac{\lambda_n - z}{\lambda_n} \right| \leq \frac{1}{\left|\hat{T}^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(0)\right|} \tag{2.3.28}$$

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

$$T^{(k-1)}_{[\lambda_1, \lambda_{n-1}]}(0) = \left| T_{k-1}\left(\frac{-\lambda_{n-1} - \lambda_1}{\lambda_{n-1} - \lambda_1}\right) \right| \tag{2.3.29}$$

$$= \frac{1}{2}(y^{k-1} + y^{-(k-1)}) \tag{2.3.30}$$

$$\text{where: } y = \frac{\sqrt{\kappa_{n-1}} + 1}{\sqrt{\kappa_{n-1}} - 1}, \kappa_{n-1} = \frac{\lambda_{n-1}}{\lambda_1} \tag{2.3.31}$$

Substituting the value for $y$ we obtain the bound:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa_{n-1}} - 1}{\sqrt{\kappa_{n-1}} + 1}\right)^{k-1} \tag{2.3.32}$$

□

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 $\lambda_1$ is much smaller than all other eigenvalues and the other eigenvalues are clutered closed together in an interval uniformaly in some interval.

**Proposition 2.4** (Small Outlier CG Convergence Rate)**.** The convergence rate is:

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\lambda_n - \lambda_1}{\lambda_1}\right)\left(\frac{\sqrt{\kappa_0} - 1}{\sqrt{\kappa_0} + 1}\right)^{k-1} \tag{2.3.33}$$

Where $\kappa_0$ is $\lambda_n/\lambda_2$.

*Proof.*

$$w(z) := \frac{\lambda_1 - z}{\lambda_1} \tag{2.3.34}$$

$$p_k(z) := w(z)\left(\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}\right) \tag{2.3.35}$$

$$\implies \max_{x\in[\lambda_2,\lambda_n]} |w(x)| = \frac{\lambda_n - \lambda_1}{\lambda_1} \tag{2.3.36}$$

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:

$$|p_k(z)| = \left|w(z)\frac{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(z)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| \tag{2.3.37}$$

$$\le \left|\frac{w(z)}{\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)}\right| \tag{2.3.38}$$

$$\le \left|\left(\frac{\lambda_n - \lambda_1}{\lambda_1}\right)\hat{T}^{(k-1)}_{[\lambda_2,\lambda_n]}(0)\right| \tag{2.3.39}$$

$$\implies \le \left(\frac{\lambda_n - \lambda_1}{\lambda_1}\right) 2\left(\frac{\sqrt{\kappa_0}+1}{\sqrt{\kappa_0}-1}\right)^{k-1} \tag{2.3.40}$$

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 $\lambda_1$, making it clearn that if $\lambda_1$ is close to zero the convergence rate is going to be terrible. □

## 2.4 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 whie CG are only for positive definite matrices. To see how everything ties together, we have to go both directions way to show the connections between these 2 iterative algorithms, which are what this section and the subsequent section about. Here, we refers Lanczos vectors as the sequence of $q_j$ generated by the Iterative Lanczos Algorithm.

For this subsection, our objective is to establish the equivalence between the parameters from the lanczos algorithm: $\alpha_k, \beta_k, q_k$ 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 2.5.** The residual and the lanczos vectors have the following relations:

$$q_1 = \hat{r}_0 \tag{2.4.1}$$
$$q_2 = -\hat{r}_1 \tag{2.4.2}$$
$$\vdots \tag{2.4.3}$$
$$q_j = (-1)^{j+1}\hat{r}_{j+1} \tag{2.4.4}$$

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

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le k-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le k-2 \\ \alpha_1 = a_0^{-1} \\ \beta_1 = \frac{\sqrt{b_0}}{\alpha_0} \end{cases} \tag{2.4.5}$$

Where $\alpha_j$ for $1 \le j \le n-1$ are the diagonal of the tridiagonal matrix $T_k$ generated by Lanczos, and $\beta_j$ for $2 \le j \le k-2$ 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.

### 2.4.1 The Base Case

Right from the start of the CG iteration we have:

$$r_0 = p_0 \tag{2.4.6}$$
$$r_1 = r_0 - a_0 A r_0 \tag{2.4.7}$$
$$Ar_0 = a_0^{-1}(r_0 - r_1) \tag{2.4.8}$$
$$Ar_0 = \frac{\|r_0\|_A^2}{\|r_0\|^2}(r_0 - r_1) \tag{2.4.9}$$

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

$$A\|r_0\|q_1 = \frac{\|r_0\|_A^2}{\|r_0\|^2}\left(\|r_0\|q_1 + \|r_1\|q_2\right) \tag{2.4.10}$$
$$= \frac{\|r_0\|_A^2}{\|r_0\|^2}\|r_0\|q_1 + \frac{\|r_1\|}{\|r_0\|}q_2 \tag{2.4.11}$$

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

### 2.4.2 The Inductive Case

**Lemma 2.4.1.** Inductively we wish to show the relation that:

$$\begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le n-2 \end{cases} \tag{2.4.12}$$

*Proof.* We start by consiering:

$$\begin{aligned} r_j &= r_{j-1} - a_{j-1}Ap_{j-1} && (2.4.13)\\ &= r_{j-1} - a_{j-1}A(r_{j-1} + b_{j-2}p_{j-1}) && (2.4.14)\\ &= r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-1} && (2.4.15) \end{aligned}$$

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

$$\begin{aligned} r_{j-1} &= r_{j-1} - a_{j-2}Ap_{j-1} && (2.4.16)\\ r_{j-1} - r_{j-1} &= a_{j-2}Ap_{j-1} && (2.4.17)\\ Ap_{j-1} &= a_{j-2}^{-1}\left(r_{j-2} - r_{j-1}\right) && (2.4.18) \end{aligned}$$

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:

$$\begin{aligned} r_j &= r_{j-1} - a_{j-1}Ar_{j-1} - a_{j-1}b_{j-2}Ap_{j-2} && (2.4.19)\\ &= r_{j-1} - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}\left(r_{j-2} - r_{j-1}\right) && (2.4.20)\\ &= \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - a_{j-1}Ar_{j-1} - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} && (2.4.21)\\ a_{j-1}Ar_{j-1} &= \left(1 + \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-1}\right) - \frac{a_{j-1}b_{j-2}}{a_{j-2}}r_{j-2} && (2.4.22)\\ Ar_{j-1} &= \left(\frac{1}{a_{j-1}} + \frac{b_{j-2}}{a_{j-2}}\right)r_{j-1} + \frac{r_j}{a_{j-1}} - \frac{b_{j-2}}{a_{j-2}}r_{j-2} && (2.4.23) \end{aligned}$$

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

$$Ar_j = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)r_j + \frac{r_{j+1}}{a_j} - \frac{b_{j-1}}{a_{j-1}}r_{j-1} \tag{2.4.24}$$

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

$$q_{j+1} := \frac{r_j}{\|r_j\|} \tag{2.4.25}$$

$$q_j := -\frac{r_{j-1}}{\|r_{j-1}\|} \quad \text{Note: This is Negative} \tag{2.4.26}$$

$$q_{j+2} := \frac{r_{j+1}}{\|r_{j+1}\|} \tag{2.4.27}$$

$$\implies A\|r_j\|q_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)\|r_j\|q_{j+1} + \frac{\|r_{j+1}\|q_{j+2}}{a_j} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}}q_j \tag{2.4.28}$$

$$Aq_{j+1} = \left(\frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}}\right)q_{j+1} + \frac{\|r_{j+1}\|}{a_j\|r_j\|}q_{j+2} + \frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|}q_j \tag{2.4.29}$$

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

$$\frac{\|r_{j+1}\|}{a_j\|r_j\|} = \frac{1}{a_j}\sqrt{b_j} \tag{2.4.30}$$

$$\frac{b_{j-1}\|r_{j-1}\|}{a_{j-1}\|r_j\|} = \frac{b_{j-1}}{a_{j-1}}\frac{1}{\sqrt{b_{j-1}}} = \frac{\sqrt{b_{j-1}}}{a_{j-1}} \tag{2.4.31}$$

$$\implies \begin{cases} \alpha_{j+1} = \frac{1}{a_j} + \frac{b_{j-1}}{a_{j-1}} & \forall 1 \le j \le n-1 \\ \beta_j = \frac{\sqrt{b_{j-1}}}{a_{j-1}} & \forall 2 \le j \le n-2 \end{cases} \tag{2.4.32}$$

Take notes that the form is now matched, but the expression for $\alpha_{j+1}$ 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. $\square$

### 2.4.3 Fixing the Sign

We can't take the triumph yet; we need to take a more careful look into the sign between $q_j$ the Lanczos Vector and its equivalcne residual: $r_{j-1}$ in CG. Here, I want to point out the fact that, there are potentially 2 susbstitution possible for the above derivation for the inductive case and regardless of which one we use, it would still preserve the correctness for the proof. By which I mean the following substitutions would have both made it work:

$$\begin{cases} q_{j+1} := \pm\frac{r_j}{\|r_j\|} \\ q_j := \mp\frac{r_{j-1}}{\|r_{j-1}\|} \\ q_{j+2} := \pm\frac{r_{j+1}}{\|r_{j+1}\|} \end{cases} \tag{2.4.33}$$

Under the context, the operations $\pm, \mp$ are correlated, choose a sign for one, the other must be of opposite sign. In this case both substitutions work the same because multiplying the equation by negative one would give the same equality, and we can always mulitply by and other negative sign to get it back. The key here is that, the sign going from $q_j$ to the next

$q_{j-1}$ will have to alternate. To find out precisely which one it is, we consider the base case for the Lanczos Vectors and Residuals:

$$q_1 = \hat{r}_0 \tag{2.4.34}$$
$$q_2 = -\hat{r}_1 \tag{2.4.35}$$
$$\vdots \tag{2.4.36}$$
$$q_j = (-1)^{j+1}\hat{r}_{j+1} \tag{2.4.37}$$

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

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

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

## 2.5 From Lanczos to Conjugate Gradient

### 2.5.1 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 2.6** (Lanczos Vectors and Residuals)**.** The $Q_k$ is the orthogonal matrix generated by Lanczos Iteration. To match the Krylov Subspace generated by the Lanczos iterations and CG, we initialize $q_1 = \hat{r}_0$, then the following relationship between Lanczos and CG occurs between their parameters:

$$\begin{cases} y_k = T_k^{-1}\beta\xi_1 \\ x_k = x_0 + Q_k y_k \\ r_k = -\beta_k \xi_k^T y_k q_{k+1} \end{cases} \tag{2.5.1}$$

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

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

$$AQ_k = Q_{k+1}\begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} \tag{2.5.2}$$

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:

$$
\begin{aligned}
x_{k+1} &= x_0 + Q_k y_k && (2.5.3)\\
r_{k+1} &= r_0 - AQ_k y_k && (2.5.4)\\
Q_k^H r_{k+1} &= Q_k^H r_0 - Q_k^H A Q_k y_k && (2.5.5)\\
\implies 0 &= \beta\xi_1 - T_k y_k && (2.5.6)\\
y_k &= T_k^{-1}\beta\xi_1 && (2.5.7)
\end{aligned}
$$

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:

$$
\begin{aligned}
r_{k+1} &= r_0 - AQ_k y_k && (2.5.8)\\
&= r_0 - AQ_k T_k^{-1}\beta\xi_1 && (2.5.9)\\
\implies &= \beta q_1 - AQ_k T_k^{-1}\beta\xi_1 && (2.5.10)\\
&= \beta q_1 - Q_{k+1}\begin{bmatrix} T_k \\ \beta_k\xi_k^T \end{bmatrix} T_k^{-1}\beta\xi_1 && (2.5.11)\\
&= \beta q_1 - (Q_k T_k + \beta_k q_{k+1}\xi_k^T)T_k^{-1}\beta\xi_1 && (2.5.12)\\
&= \beta q_1 - (Q_k\beta\xi_1 + \beta_k q_{k+1}\xi_{k+1}T_k^{-1}\beta\xi_1) && (2.5.13)\\
&= -\beta_k q_{k+1}\xi_k^T T_k^{-1}\beta\xi_1 && (2.5.14)
\end{aligned}
$$

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

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

**Proposition 2.7** (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:

$$
P_k = Q_k U_k^{-1} \tag{2.5.15}
$$

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

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

$$
\begin{aligned}
x_k &= x_0 + Q_k y_k && (2.5.16)\\
&= x_0 + Q_k T_k^{-1}\beta\xi_1 && (2.5.17)\\
&= x_0 + Q_k U_k^{-1} L_k^{-1}\beta\xi_1 && (2.5.18)\\
&= x_0 + P_k L_k^{-1}\beta\xi_1 && (2.5.19)
\end{aligned}
$$

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

$$P_k^T A P_k \tag{2.5.20}$$
$$= (Q_k U_k^{-1})^T A Q_k U_k^{-1} \tag{2.5.21}$$
$$= U_k^{-T} Q_k^T A Q_k U_k^{-1} \tag{2.5.22}$$
$$= U_k^{-T} T_k U_k^{-1} \tag{2.5.23}$$
$$= U_k^{-T} L_k \tag{2.5.24}$$

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. □

### 2.5.2 Matching the $a_k, b_k$ in CG

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

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

$$P_k = \begin{bmatrix} p_1 & p_2 & \cdots & p_k \end{bmatrix} \quad Q_k = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \tag{2.5.25}$$

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

$$T_k = L_k U_k = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix} \begin{bmatrix} u_1 & \beta_1 & & \\ & u_2 & \beta_2 & \\ & & \ddots & \beta_{k-1} \\ & & & u_k \end{bmatrix} \tag{2.5.26}$$

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

$$x_k = x_0 + P_k L_k^{-1} \beta \xi_1 \tag{2.5.27}$$
$$x_k - x_{k-1} = P_k L_k^{-1} \beta \xi_1 - P_{k-1} L_{k-1}^{-1} \beta \xi_1 \tag{2.5.28}$$
$$= P_k \beta (L_k^{-1})_{:,1} - P_{k-1} \beta (L_{k-1}^{-1})_{:,1} \tag{2.5.29}$$
$$= \beta (L_k^{-1})_{k,1} P_k \tag{2.5.30}$$
$$\implies x_k = x_{k-1} + \beta (L_k^{-1})_{k,1} p_k \tag{2.5.31}$$

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:

$$\begin{aligned}
P_k &= Q_k U_k^{-1} && (2.5.32)\\
P_k U_k &= Q_k && (2.5.33)\\
\implies \beta_{k-1} p_{k-1} + u_k p_k &= q_k && (2.5.34)\\
u_k p_k &= q_k - \beta_{k-1} p_{k-1} && (2.5.35)\\
p_k &= u_k^{-1}(q_k - \beta_{k-1} p_{k-1}) && (2.5.36)
\end{aligned}$$

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

$$\begin{aligned}
T_k &= L_k U_k && (2.5.37)\\
T_{k+1} &= \begin{bmatrix} T_k & \beta_k \xi_k \\ \beta_k \xi_k^T & \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} L_k & \mathbf{0} \\ l_k \xi_k^{-1} & 1 \end{bmatrix} \begin{bmatrix} U_k & \eta_k \xi_k \\ \mathbf{0} & u_{k+1} \end{bmatrix} && (2.5.38)\\
&= \begin{bmatrix} L_k U_k & \eta_k L_k \xi_k \\ l_k \xi_k^T U_k & \eta_k l_k \xi_k^T \xi_k + u_{k+1} \alpha_k \end{bmatrix} && (2.5.39)\\
&= \begin{bmatrix} T_k & \eta_k (L_k)_{:,k} \\ l_k (U_k)_{k,:} & \eta_k l_k + u_{k+1} \end{bmatrix} && (2.5.40)\\
&= \begin{bmatrix} T_k & \eta_k \\ l_k u_k & \eta_k l_k + u_{k+1} \end{bmatrix} && (2.5.41)
\end{aligned}$$

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

$$\begin{cases} u_{k+1} &= \alpha_{k+1} - \beta_k^2 / u_k \\ l_k &= \beta_k / u_k \end{cases} \tag{2.5.42}$$

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:

$$\begin{aligned}
L_k^{-1} L_k &= I && (2.5.43)\\
\begin{bmatrix} L_k^{-1} & \mathbf{0} \\ s_k^T & d_{k+1} \end{bmatrix} \begin{bmatrix} L_k & \mathbf{0} \\ l_k \xi_k^T & 1 \end{bmatrix} &= I && (2.5.44)\\
\begin{bmatrix} I & \mathbf{0} \\ s^T L_k + d_{k+1} l_k \xi_k^T & d_{k+1} \end{bmatrix} &= I && (2.5.45)
\end{aligned}$$

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

$$s^T L_k + d_{k+1} l_k \xi_k^T = \mathbf{0} \tag{2.5.46}$$

$$L_k^T s_k + d_{k+1} l_k \xi_k = \mathbf{0} \tag{2.5.47}$$

$$s_k + L^{-T} d_{k+1} l_k \xi_k = \mathbf{0} \tag{2.5.48}$$

$$(s_k)_1 + d_{k+1} l_k (L_k^{-1}) \xi_k = 0 \tag{2.5.49}$$

$$(s_k)_1 + d_{k+1} l_k (L_k^{-1})_{k,1} = 0 \tag{2.5.50}$$

$$\implies (s_k)_1 = -l_k (L_k^{-1})_{k,1} \tag{2.5.51}$$

$$(s_k)_1 = (L_{k+1}^{-1})_{k+1,1} \tag{2.5.52}$$

$$\implies (L_{k+1}^{-1})_{k+1,1} = -l_k (L_k^{-1})_{k,1} \tag{2.5.53}$$

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 2.5.1.** 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:

$$AQ_k \approx Q_{k+1} \begin{bmatrix} T_k \\ \beta_k \xi_k^T \end{bmatrix} \tag{2.5.54}$$

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

# 3 Effects of Floating Point Arithematic

In this section, we highlights the practical concerns of the algorithm and showcase it with numerical experiments with analysis, in hope to get deeper insights about the behaviors of

Lanczos and Conjugate Gradient under floating point arithematic.

### Numerical Experiments

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.

# 4 Appendix

# 5 Biliography