Emory University

MATH 515 Numerical Analysis I

Learning Notes

Jiuru Lyu

December 14, 2024

Contents

List of Algorithms

1 Linear Algebra Review

Notation 1.1. Vector: $x \in \mathbb{C}^n : x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$, $x_i \in \mathbb{C}$

Notation 1.2. Matrix: $A \in \mathbb{C}^{m\times n} : A = \begin{bmatrix} a_{11} & \cdots & A_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$, $a_{ij} \in \mathbb{C}$

1.1 The Basics

Definition 1.1.1 (Matrix-Vector Product/Mat-Vec).

$$b = Ax$$

Linear Combination Perspective Vector addition and scalar multiplication.

b is a *linear combination* of the columns of A.

Suppose $A = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix}, a_j \in \mathbb{C}^m$, then

$$\begin{aligned} b = Ax &= \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= x_1 a_1 + \cdots + x_n a_n \\ &= \sum_{j=1}^{n} x_j a_j. \end{aligned}$$

Entry-wise Perspective

$$b_i = \sum_{j=1}^{n} a_{ij} x_j$$

This perspective is useful in MATLAB: `b(i) = A(i,:) * x(j).`

Definition 1.1.2 (Matrix-Matrix Multiplication/Mat-Mat).

$$B = AX,$$

where $B \in \mathbb{C}^{m\times \ell}$, $A \in \mathbb{C}^{m\times n}$, and $X \in \mathbb{C}^{n\times \ell}$.

Standard Perspective mat-vec

$$b_j = Ax_j,$$

where b_j is the j-th column of B and x_j is the j-th column of X.

Outer Product mat-mat

$$B = AX = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix} \begin{bmatrix} - & \overline{x}_1^\top & - \\ & \vdots & \\ - & \overline{x}_n^\top & - \end{bmatrix}$$
$$= a_1 \overline{x}_1^\top + \cdots + a_n \overline{x}_n^\top,$$

where $\overline{x}_i^\top$ is the i-th row of X.

1.2 Fundamental Subspaces of Matrices

Definition 1.2.1 (Transpose). The *transpose* of matrix A, denoted $A^\top$, swaps the rows and columns. $A \in \mathbb{C}^{m\times n}$ and $A^\top \in \mathbb{C}^{n\times m}$.

Definition 1.2.2 (Conjugate Transpose). The *conjugate transpose* of a matrix A, denoted A^* or A^H, swaps the rows and columns, and then computes the complex conjugate ($\overline{a+b\mathrm{i}} = a - b\mathrm{i}$) of each entry.

Theorem 1.2.3 Properties of Transpose

$$(AB)^\top = B^\top A^\top \quad \text{and} \quad (AB)^* = B^* A^*.$$

Remark 1.1 (Vector Space) *The detailed definition of vector spaces are omitted here, but the key idea is that* math works *on vector spaces. To put it simple, vector addition and scalar multiplication are defined on vector spaces. For example, $\mathbb{C}^n$ or $\mathbb{R}^n$ are typical examples of vector spaces.*

Theorem 1.2.4 Closure of Subspaces
If U is a vector subspace, $U \subseteq \mathbb{C}^n$, then

$$x, y \in U \implies \alpha x + \beta y \in U, \quad \text{where } \alpha, \beta \in \mathbb{C}.$$

Example 1.2.5 Non-Example of Subspace
Let $U = \left\{ \begin{bmatrix} \alpha \\ 1 \end{bmatrix} : \alpha \in \mathbb{C} \right\}$. Note that $\begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 7-8\mathrm{i} \\ 1 \end{bmatrix} \in U$, but

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 7-8\mathrm{i} \\ 1 \end{bmatrix} = \begin{bmatrix} 10-8\mathrm{i} \\ 2 \end{bmatrix} \notin U.$$

So, U is not a subspace of $\mathbb{C}^2$.

Definition 1.2.6 (Span). The *span* of vectors is all possible linear combinations.

$$\text{span}\{a_1,\dots,a_n\} = \{x_1a_1+\cdots+x_na_n \mid x_i\in\mathbb{C}\} = \{Ax \mid x\in\mathbb{C}^n\}.$$

Remark 1.2 *A span of vectors always forms a subspace.*

Definition 1.2.7 (Linear Independence/L.I.**).** $\{a_1,\dots,a_n\}$ is L.I. if $x_1a_1+\cdots+x_na_n = 0$ only when $x_i = 0$.

Definition 1.2.8 (Basis). A *basis* is a set of L.I. vectors that span a vector subspace.

Example 1.2.9

Consider $U = \left\{ \begin{bmatrix}\alpha\\ \beta\\ 0\end{bmatrix} : \alpha,\beta\in\mathbb{R} \right\}$. Evidently, $\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix} \right\}$ is a basis, but $\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix} \right\}$ is *NOT* a basis because it is not L.I.. Also, $\left\{ \begin{bmatrix}1\\0\\0\end{bmatrix} \right\}$ is *NOT* a basis because it does not span U.

Remark 1.3 *There is* NO *unique basis for* U*. For example,* $\left\{ \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\\0\end{bmatrix} \right\}$ *is also a basis of* U*.*

Definition 1.2.10 (Dimension). The *dimension* of a subspace is the number of vector in any basis.

Remark 1.4 *For example, in Example 1.2.9,* $\dim(U) = 2$.

Definition 1.2.11 (Four Fundamental Subspaces of $A\in\mathbb{C}^{m\times n}$).

- $\text{range}(A) = \text{col}(A) = \{Ax \mid x\in\mathbb{C}^n\} = \text{span}\{\text{columns of } A\} \subseteq \mathbb{C}^m$. This is a subspace of the output space.
- $\text{null}(A) = \ker(A) = \{x\in\mathbb{C}^n \mid Ax = 0\} \subseteq \mathbb{C}^n$. This is a subspace of the input space.
- $\text{range}(A^*) \subseteq \mathbb{C}^n$
- $\text{null}(A^*) \subseteq \mathbb{C}^m$

Theorem 1.2.12 Fundamental Theorem of Linear Algebra

$$\text{range}(A)\oplus\text{null}(A^*) = \mathbb{C}^m \quad \text{and} \quad \text{range}(A^*)\oplus\text{null}(A) = \mathbb{C}^n.$$

Remark 1.5 (The Notation $\oplus$) *The notation $A \oplus B = C$ means that $A \perp B$ and $A \cup B = C$.*

Example 1.2.13

Consider

$$A = \begin{bmatrix} 1 & 2 & 0 & -3 & 4 \\ 0 & 0 & 1 & 5 & -6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{C}^{3\times 5}.$$

Then, $\text{range}(A) = \{Ax \mid x \in \mathbb{C}^5\} = \text{span}\{\text{L.I. columns of } A\} = \text{span}\left\{\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}\right\}$.

Also, $\text{null}(A) = \{x \in \mathbb{C}^5 \mid Ax = 0\} = \text{span}\{\text{basic solutions}\} = \text{span}\left\{\begin{bmatrix}-2\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}3\\0\\-5\\1\\0\end{bmatrix}, \begin{bmatrix}-4\\0\\6\\0\\1\end{bmatrix}\right\}$.

$\text{range}(A^*) = \text{span}\left\{\begin{bmatrix}1\\2\\0\\-3\\4\end{bmatrix}, \begin{bmatrix}0\\0\\1\\5\\-6\end{bmatrix}\right\}$ and $\text{null}(A^*) = \text{span}\left\{\begin{bmatrix}0\\0\\1\end{bmatrix}\right\}$.

Definition 1.2.14 (Rank and Nullity).

$$\begin{aligned}\text{rank}(A) &= \dim(\text{range}(A)) \\ &= \#\text{ of L.I. columns/rows} \\ &= \#\text{ of leading } 1\text{'s/pivots} \\ &= \#\text{ of non-zero singular values} \\ &= \text{the minimal } \#\text{ of rank-}1\text{ matrices that sum to } A \\ &\qquad A = \sum_{i=1}^{r} \underbrace{u_i v_i^*}_{\text{rank-1 matrix}} \quad : \text{ if } r \text{ is minimal, then } \text{rank}(A) = r \\ \text{nullity}(A) &= \dim(\text{null}(A))\end{aligned}$$

Theorem 1.2.15 Rank-Nullity Theorem

If $A \in \mathbb{C}^{m\times n}$, then

$$\text{rank}(A) + \text{nullity}(A) = n.$$

Definition 1.2.16 (Full Column/Row Rank). A matrix A is *full column rank* if $\text{rank}(A) = n$. A is *full row rank* if $\text{rank}(A) = m$. It is *full rank* if $\text{rank}(A) = \min\{m, n\}$.

Remark 1.6 *If A has full column rank,* $\text{null}(A) = \{0\}$*, the* trivial null space*, and the only solution to* $Ax = 0$ *is* $x = 0$.

If A has full row rank, then $\text{range}(A) = \mathbb{C}^m$*, and* $Ax = b$ *is always solvable.*

1.3 Inverse and Invertible Matrices

Definition 1.3.1 (Inverse). A is *nonsingular* or *invertible* if it is square ($A \in \mathbb{C}^{m\times m}$) and it has full rank. We denote the inverse as A^{-1}.

Theorem 1.3.2

The following are equivalent (T.F.A.E.): If $A \in \mathbb{C}^{m\times m}$, then

- A is invertible: $AA^{-1} = A^{-1}A = I$.
- $\text{rank}(A) = m$.
- $\text{range}(A) = \mathbb{C}^m$.
- $\text{null}(A) = \{0\}$.
- 0 is not an eigenvalue of A.
- 0 is not a singular value of A.
- $\det(A) \neq 0$.

Proposition 1.3 : If A, B are invertible and of the same size, $(AB)^{-1} = B^{-1}A^{-1}$.

Inverse in MATLAB

```
% Do not use the following
inv(A);
% To solve Ax=b, use the "\" operator
x = A \ b;
```

1.4 Orthogonal Vectors and Matrices

Definition 1.4.1 (Adjoint/A^*). If $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$, then its adjoint, denoted as A^*, is defined as swapping the rows with columns and then taking the conjugate of each element:

$$A^* = \begin{bmatrix} \overline{a_{11}} & \cdots & \overline{a_{m1}} \\ \vdots & \ddots & \vdots \\ \overline{a_{1m}} & \cdots & \overline{a_{mn}} \end{bmatrix}.$$

Definition 1.4.2 (Inner Product). The operation $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{C}$ is an *inner product* if it satisfies

- Conjugate symmetry: $\langle x, y\rangle = \overline{\langle y, x\rangle}$
- Homogeneity: $\alpha \in \mathbb{C}: \quad \langle \alpha x, y\rangle = \overline{\alpha}\langle x, y\rangle, \qquad \langle x, \alpha y\rangle = \alpha\langle x, y\rangle.$
- Additivity: $\langle x + z, y\rangle = \langle x, y\rangle + \langle z, y\rangle$
- Positive definite: $\langle x, x\rangle \geq 0 \quad$ and $\quad \langle x, x\rangle = 0 \iff x = 0.$

Proposition 1.3 : $\langle Ax, y\rangle = \langle x, A^* y\rangle$.

Definition 1.4.4 (Orthogonal Vector). $x, y \in \mathbb{C}^n$ are *orthogonal* if $x^* y = 0$.

Theorem 1.4.5
Non-zero orthogonal vectors are L.I..

Proof 1. Suppose $\{x_1, \ldots, x_q\}$ are non-zero orthogonal. Then, $x_i^* x_j = 0$ if $i \neq j$. [WTS: x_k is not a linear combination of the remaining vectors for any k.]

WLOG, assume x_q is a linear combination of $x_1, \ldots, x_{q-1}$. Then, $\exists c_1, \ldots, c_{q-1} \in \mathbb{C}\ s.t.$

$$x_q = c_1 x_1 + \cdots + c_{q-1} x_{q-1}.$$

Then,

$$\begin{aligned} x_q^* x_q &= x_q^*(c_1 x_1 + \cdots + c_{q-1} x_{q-1}) \\ &= c_1 x_q^* x_1 + \cdots + c_{q-1} x_q^* x_{q-1} \\ x_q^* x_1 &= 0. \quad \text{[due to orthogonal]} \end{aligned}$$

As $x_q^* x_q = 0 \iff x_q = 0$, ⁎ this contradicts with our assumption that none of $x_1, \ldots, x_q$ is zero, So, there must be no linear dependence. Q.E.D. ■

Definition 1.4.6 (Unitary Matrices). $Q \in \mathbb{C}^{m\times m}$ is *unitary* if $Q^{-1} = Q^*$.

Remark 1.7 *If Q is real-valued, $Q \in \mathbb{R}^{m\times m}$, then Q is orthogonal and $Q^{-1} = Q^\top$.*

Remark 1.8 (Why the Name?) *Note that*

$$Q^* Q = Q^{-1} Q = I = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

Also,

$$Q^* Q = \begin{bmatrix} - & q_1^* & - \\ & \vdots & \\ - & q_m^* & - \end{bmatrix} \begin{bmatrix} | & & | \\ q_1 & \cdots & q_m \\ | & & | \end{bmatrix} = \begin{bmatrix} q_i^* q_j \end{bmatrix}$$

So,

$$q_i^* q_j = \begin{cases} 1 & i = j \text{ (on diagonal, with unit length)} \\ 0 & i \neq j \text{ (off diagonal, orthogonal)} \end{cases}$$

1.5 Vector and Matrix Norms

Definition 1.5.1 (Vector Norm). $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ is a *vector norm* if $\forall x, y \in \mathbb{C}^n$, $\alpha \in \mathbb{C}$, the following satisfies:

- positive definite: $\|x\| \geq 0$ and $\|x\| = 0 \iff x = 0$.
- positive homogeneity: $\|\alpha x\| = |\alpha|\|x\|$, where $|\alpha| = |a + b\text{i}| = \sqrt{a^2 + b^2}$.
- triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$.

Proposition 1.2 Inner Product Induce Norm: If $\langle \cdot, \cdot \rangle$ is an inner product, then $\|x\| \coloneqq \sqrt{\langle x, x \rangle}$ is a norm.

Example 1.5.3 Examples of Vector Norms

- 1-norm:
$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$
- 2-norm:
$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$
- p-norm:
$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad 1 \leq p < \infty.$$
- ∞-norm:
$$\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

Remark 1.9 *When thinking of properties of norms, consider the following ball:*

$$B_r = \{x \in \mathbb{C}^n \mid \|x\| > r\}.$$

Theorem 1.5.4 Unitary Invariance of 2-Norms
Suppose $Q \in \mathbb{C}^{m \times m}$ si unitary. Then,

$$\|Qx\|_2 = \|x\|_2.$$

Proof 1. Suppose $Q \in \mathbb{C}^{m \times m}$ is unitary. Then, $Q^*Q = I = QQ^*$. Hence,

$$\|Qx\|_2^2 = (Qx)^*(Qx) = x^* \underbrace{Q^*Q}_{I} x = x^* I x = x^* x = \|x\|_2^2.$$

Q.E.D. ■

Remark 1.10 (Geometric Interpretation) *Unitary matrices preserve length.*

Proposition 1.5 Some Famous Inequalities:

- Hölder: $\forall 1 \leq p, q \leq \infty$, if $\frac{1}{p} + \frac{1}{q} = 1$, then

$$|x^* y| \leq \|x\|_p \|y\|_q.$$

- Cauchy-Schwarz (consequence of Hölder):

$$|x^* y| \leq \|x\|_2 \|y\|_2.$$

Definition 1.5.6 (Matrix Norms Induced by Vector Norms). Given two vector norms

$$\|\cdot\|_{(n)} : \mathbb{C}^n \to \mathbb{R} \text{ acts on vectors from } \mathbb{C}^n \qquad [\textit{input space}]$$
$$\|\cdot\|_{(m)} : \mathbb{C}^m \to \mathbb{R} \text{ acts on vectors from } \mathbb{C}^m \qquad [\textit{output space}]$$

Suppose $A \in \mathbb{C}^{m \times n}$, then

$$\|A\|_{(m,n)} = \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|Ax\|_{(m)}}{\|x\|_{(n)}} = \sup_{\substack{x \in \mathbb{C}^n \\ \|x\|_{(n)} = 1}} \|Ax\|_{(m)}.$$

Specially, if (n) and (m) are the same, say $(n) = (m) = (p)$, then we write $\|A\|_{(p)}$.

Theorem 1.5.7 Some Matrix Norms

- $\|A\|_1 =$ max. column sum $= \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| = \max_{j=1,\ldots,n} \|a_j\|_1$.
- $\|A\|_\infty =$ max. row sum $= \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}|$.
- $\|A\|_2 =$ largest singular value $= \sqrt{\text{largest eigenvalue of } A^* A}$.

Remark 1.11 (General Proof Structure)

- *Show RHS is an upper bound of the induced matrix norm definition.*
- *Find one vector that achieves this upper bound.*

Proof 2. (of $\|A\|_2$) Recall: $\|A\|_2 = \sup_{\|x\|_2 = 1} \|Ax\|_2$.

Step 1 Find an upper bound of $\|Ax\|_2$ given $\|x\|_2 = 1$.

Note that A^*A is symmetric, so it is unitarily diagonalizable. Hence, $A^*A = V\Lambda V^*$ from the Spectrum Theorem. Also, A^*A is positive semidefinite, so $\lambda_i(A^*A) \geq 0$. Hence, we know

$$\|Ax\|_2^2 = x^* \underbrace{A^*A}x = x^*V\Lambda V^*x = (x^*V)\Lambda(V^*x).$$

Define $V^*x =: y$. Then, $\|y\|_2 = \|V^*x\|_2 = \|x\|_2 = 1$ as V is unitary and 2-norm is unitary invariant. So,

$$\|Ax\|_2^2 = y^*\Lambda y (= y^*\sqrt{\Lambda}^*\sqrt{\Lambda}y) \implies \|A\|_2 = \sup_{\|y\|_2=1} \left\|\sqrt{\Lambda}y\right\|_2$$

Further, as Λ is diagonal,

$$\begin{aligned}\|Ax\|_2^2 = y^*\Lambda y* &= \sum_{i=1}^n \lambda_i |y_i|^2 \\ &\leq \lambda_{\max}\|y\|_2^2 = \lambda_{\max}.\end{aligned}$$

Step 2 Find one vector to achieve the equality.

Let $x_{\max}$ be unit eigenvector of A^*A, corresponding to $\lambda_{\max}$.Then,

$$\begin{aligned}\|Ax_{\max}\|_2^2 &= x_{\max}^* \underbrace{A^*A}x_{\max} \\ &= x_{\max}^*\lambda_{\max}x_{\max} \\ &= \lambda_{\max}(x_{\max}^*x_{\max}) \\ &= \lambda_{\max}\underbrace{\|x_{\max}\|_2^2}_{=1} \\ &= \lambda_{\max}\end{aligned}$$

Q.E.D. ■

Proposition 1.8 Bounding Induced Matrix Norms: Let $A \in \mathbb{C}^{\ell\times m}$ and $B \in \mathbb{C}^{m\times n}$, then

$$\|AB\|_{(\ell,n)} \leq \|A\|_{(\ell,m)}\|B\|_{(m,n)}.$$

Remark 1.12 *Hint to the Proof By definition,*

$$\|A\|_{(\ell,m)} = \sup_{x\neq 0}\frac{\|Ax\|_{(\ell)}}{\|x\|_{(m)}} \geq \frac{\|Ax\|_{(\ell)}}{\|x\|_{(m)}}.$$

So,

$$\|Ax\|_{(\ell)} \leq \|A\|_{(\ell)}\|x\|_{(m)}.$$

Definition 1.5.9 (General Matrix Norms). Frobenius:

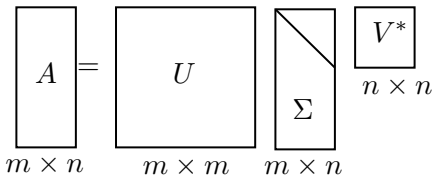
$$\|A\|_F = \sqrt{\sum_{j=1}^n\sum_{i=1}^m |a_{ij}|^2} = \sqrt{\mathrm{tr}(A^*A)} = \sqrt{\sum_{j=1}^n \|a_j\|_2^2}.$$

Proposition 1.10 Properties of Frobenius Norm:

- $\|AB\|_F \leq \|A\|_F \|B\|_F$
- $\|QA\|_F = \|A\|_F$, if Q is an unitary matrix.

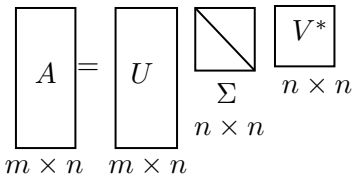
1.6 Singular Value Decomposition (SVD)

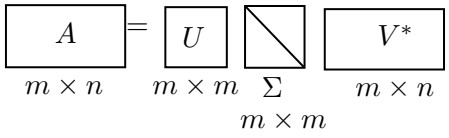
Definition 1.6.1 (Full SVD).



$U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrix, and $\Sigma \in \mathbb{R}^{m\times n}$ with $\mathrm{diag}(\Sigma) = \underbrace{(\sigma_1, \sigma_2, \ldots, \sigma_n)}_{\text{singular values}}$, where $\sigma_i \in \mathbb{R}$ is non-negative and ordered with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

Definition 1.6.2 (Reduced SVD).





Proposition 1.3 : Singular values are square roots of non-zero eigenvalues of A^*A and AA^*.

Proof 1. Suppose $A = U\Sigma V^*$ in the induced form. Then,

$$\begin{aligned} A^*A = (U\Sigma V^*)^*(U\Sigma V^*) &= V\Sigma^* \underbrace{U^*U}_{I} \Sigma V^* \\ &= V\Sigma^*\Sigma V^* \\ &= V\Sigma^2 V^*. \end{aligned}$$

Note that Σ^2 contains squared singular values. As A^*A is PSD, all eigenvalues are non-negative. So, we can take the square root to recover singular values. Q.E.D. ■

Theorem 1.6.4 Existence of SVD
Every matrix $A \in \mathbb{C}^{m\times n}$ has an SVD.

Proof 2. In this proof, we will consider $U^*AV = \Sigma$ (derived from $A = U\Sigma V^*$). Let $\sigma_1 = \|A\|_2$.

- From a *compactness* argument, $\exists\, v_1,\ \|v_1\|_2 = 1\ s.t.\ Av_1 = \sigma_1 u_1$, where $\|u\|_2 = 1$.

  Proof.

  Theorem (Weirestrass) *Continuous function over compact set attains maximum/minimum over that set.*

  Define function $f(x) = \|Ax\|_2$, continuous. Then,

  $$\|A\|_2 = \sup_{\|x\|_2=1} f(x).$$

  Note that $\|x\|_2 = 1$ is a close and bounded domain (compact domain), so we attain a maximum. Suppose v_1 is the vector that attains the maximum, then,

  $$\|Av_1\|_2 = \sigma_1.$$

  Hence, consider the unit vector

  $$u_1 = \frac{Av_1}{\|Av_1\|_2} = \frac{Av_1}{\sigma_1}.$$

  That is, $Av_1 = \sigma_1 u_1$ as desired. $\square$

- Build orthonormal bases: $\{v_1, v_2, \ldots, v_n\} \subset \mathbb{C}^n$ and $\{u_1, u_2, \ldots, u_m\} \subset \mathbb{C}^m$, with $Av_1 = \sigma_1 u_1$. Then, define matrices

  $$V1 = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix} \quad \text{and} \quad U_1 = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_m \\ | & | & & | \end{bmatrix}.$$

  Now, consider

  $$U_1^* A V_1 = \begin{bmatrix} - & u_1^* & - \\ & \vdots & \\ - & u_m^* & - \end{bmatrix} A \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix} = \begin{bmatrix} u_1^* A v_1 & \cdots & u_1^* A v_n \\ \vdots & \ddots & \vdots \\ u_m^* A v_1 & \cdots & u_m^* A v_n \end{bmatrix}$$

  $$= \begin{bmatrix} u_1^* \sigma_1 u_1 & -w^* - \\ u_2^* \sigma_1 u_1 (=0) & \\ \vdots & B \\ u_m^* \sigma_1 u_1 (=0) & \end{bmatrix} \qquad \text{[orthonormal]}$$

  $$= \begin{bmatrix} \sigma_1 & -w^* - \\ 0 & B \end{bmatrix}.$$

- Show $w = 0$.

  – Since U and V are unitary matrices, $\|U_1^* A V_1\|_2 = \|A\|_2 = \sigma_1$.

– Let $S = U_1^* A V_1$, then

$$\|S\|_2 \geq \frac{\|Sx\|_2}{\|x\|_2}, \quad x \neq 0.$$

Pick $x = \begin{bmatrix} \sigma_1 \\ w \end{bmatrix}$. Then,

$$\|S\|_2 \geq \frac{\left\| \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix} \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|_2}{\left\| \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|_2} = \frac{\left\| \begin{bmatrix} \sigma_1^2 + w^* w \\ Bw \end{bmatrix} \right\|_2}{\left\| \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|_2}.$$

Suppose $Bw = 0$. Then, $\left\| \begin{bmatrix} \sigma_1^2 + w^* w \\ 0 \end{bmatrix} \right\|_2 = \sigma_1^2 + w^* w$. If $Bw \neq 0$, the norm will get larger. So, $\sigma_1^2 + w^* w$ is a lower bound of the norm. Then,

$$\|S\|_2 \geq \frac{\sigma_1^2 + w^* w}{\sqrt{\sigma_1^2 + w^* w}} = \sqrt{\sigma_1^2 + w^* w}.$$

So, $\|S\|_2 = \sigma_1 \geq \sqrt{\sigma_1^2 + w^* w}$.
As $w^* w \geq 0$, it must be that $w^* w = 0$ since $w^* w = 0 \iff w = 0$. Then,

$$U_1^* A V_1 = \begin{bmatrix} \sigma_1 & 0 \\ 0 & B \end{bmatrix}.$$

- By induction, if $B = U_2 \Sigma_2 V_2^*$, then

$$A = \underbrace{U_1 \begin{bmatrix} 1 & \\ & U_2 \end{bmatrix}} \begin{bmatrix} \sigma_1 & \\ & \Sigma_2 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & \\ & V_2^* \end{bmatrix} V_1^*}.$$

Q.E.D. ■

Proposition 1.5 Use SVD to Solve Linear System: $Ax = b \implies \Sigma \underbrace{x'}_{V^* x} = \underbrace{b'}_{U^* b}$.

Proposition 1.6 Rank Revealing: $\operatorname{rank}(A) = \#$ of non-zero singular values.

$$A = U \Sigma V^* = \sum_{i=1}^{r} \sigma_i \underbrace{u_i v_i^*}_{\text{rank-1}}, \quad \text{if } \operatorname{rank}(A) = r.$$

Proposition 1.7 Connection to the Subspaces:

$$\operatorname{range}(A) = \operatorname{range}(U) = \operatorname{span}\{u_1, \ldots, u_r\}$$
$$\operatorname{null}(A) = \operatorname{null}(V) = \operatorname{span}\{v_{r+1}, \ldots, v_n\}.$$

Proposition 1.8 Connection to Matrix Norms:

$$\|A\|_2 = \sigma_1$$
$$\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}.$$

Proposition 1.9 Mapping Between Spaces: Note that $Av_i = \sigma_i u_i$ and $A^* u_i = \sigma_i v_i$. Then,

$$\begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \sigma_i \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Application of SVD: Low-Rank Approximation

Definition 1.6.10 (Truncated SVD).

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^* \xleftarrow[\text{approximation}]{\text{low-rank}} A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*, \quad k \leq r.$$

Theorem 1.6.11 Eckart-Yang
For $k \leq r$, A_k is the best rank-k approximation to A:

$$\|A - A_k\|_2 = \inf_{\text{rank}(b) \leq k} \|A - b\|_2 = \sigma_{k+1}.$$

Proof 3.

1. $\|A - A_k\|_2 = \sigma_{k+1}$

   As singular values are ordered,

$$\left\| \sum_{i=1}^{r} \sigma_i u_i v_i^* - \sum_{i=1}^{k} \sigma_i u_i v_i^2 \right\|_2 = \left\| \sum_{i=k+1}^{r} \sigma_i u_i v_i^* \right\|_2 = \sigma_{k+1}.$$

2. Show $\|A - B\|_2 \geq \sigma_{k+1} \quad \forall\, B\ s.t.\ \text{rank}(B) \leq k$.

   As we know nothing on B, we want to eliminate the dependency on B.

$$\begin{aligned} \|A - B\|_2 &= \sup_{\|x\|_2 = 1} \|(A - B)x\|_2 \\ &\geq \|(A - B)z\|_2. \end{aligned}$$

   Choose $z \neq 0\ s.t.\ z \in \text{null}(B)$ but $z \notin \text{null}(A)$ with $\|z\|_2 = 1$. *Note that such z will always exists*

because by rank-nullity theorem, B has a lower rank and A is of a higher rank. So,

$$\|A-B\|_2 \geq \left\|Az - \underbrace{Bz}_{=0}\right\|_2 = \|Az\|_2.$$

By SVD, $A = U\Sigma V^*$. As $z \notin \text{null}(A)$, then $z = V(:,1:k+1) * c$, $\|c\|_2 = 1$. *Note that* $V(:,1:k+1)*c$ *is the linear combination of the first* $k+1$ *columns of* V. *Also, note tat* $\text{null}(A) = \text{null}(V) = \text{span}\{v_{k+1},\ldots,v_n\}$. *Then,* $\text{null}(A)^\perp = \text{span}\{v_1,\ldots,v_{k+1}\}$, *denotes everything not in* $\text{null}(A)$, *and* $\dim(\text{null}(A)) = k+1$. *Further, note that* $\text{null}(B) = \text{span}\{x_1,\ldots,x_{n-k}\}$ *and* $\dim(\text{null}(B)) = n-k$. *Because* $n-k+k+1 = n+1 > n$, $\exists\, z \in \text{null}(B) \cap \text{null}(A)^\perp$. Hence, we have

$$\begin{aligned}
\|Az\|_2^2 &= \|U\Sigma V^*(V(:,1:k+1)*c)\|_2^2 \\
&= \left\|\Sigma\begin{bmatrix} c \\ 0 \end{bmatrix}\right\|_2^2 && [U \text{ is unitary}] \\
&= \sum_{i=1}^{k+1} \sigma_i^2 |c_i|^2 \\
&= \sigma_{k+1}^2 \sum_{i=1}^{k+1} \left(\frac{\sigma_i}{\sigma_{k+1}}\right)^2 |c_i|^2 && [\sigma_i \text{ has smaller indices, so is a larger singular value}] \\
&\geq \sigma_{k+1}^2 \underbrace{\sum_{i=1}^{k+1} |c_i|^2}_{\|c_2\|=1} && \left[\text{As } \sigma_i > \sigma_{k+1}, \text{ we have } \frac{\sigma_i}{\sigma_{k+1}} > 1\right] \\
&= \sigma_{k+1}^2.
\end{aligned}$$

So, $\|Az\|_2 \geq \sigma_{k+1}$. Hence,

$$\|A-B\|_2 \geq \|Az\|_2 \geq \sigma_{k+1}.$$

Q.E.D. ■

2 Conditioning and Stability

2.1 Conditioning & Condition Numbers

Abstract-View: Problem: $f : X \to Y$

1. Well-conditioned: small changes in input $\implies$ small changes in output
2. Ill-conditioned: small changes in input $\implies$ BIG changes in output

Definition 2.1.1 (Condition Number).

1. δx: perturbation of input. $\delta f = f(x+\delta x) - f(x)$: perturbation of output
2. *Absolute Condition Number* ($\widehat{\kappa}$): $\widehat{\kappa} = \widehat{\kappa}(x)$ of problem f at the input x is defined as
$$\widehat{\kappa}(x) \coloneqq \sup_{\delta x} \frac{\|\delta f\|}{\|\delta x\|}.$$
3. Tylor Approximation:
$$f(x+\delta x) \approx f(x) + \overbrace{\underbrace{J(x)}_{\text{Jacobian: } J_{ij}=\frac{\partial f_i}{\partial x_j}} \delta x}^{\delta f}$$
$$\begin{aligned} \|\delta f\| &\approx \|J(x)\delta x\| \\ &\leq \|J(x)\| \cdot \|\delta x\| \\ \widehat{\kappa}(x) = \sup_{\delta x} \frac{\|\delta f\|}{\|\delta x\|} &= \frac{\|J(x)\| \cdot \|\delta x\|}{\|\delta x\|} = \|J(x)\|. \end{aligned}$$
4. *Relative Condition Number*:
$$\kappa(x) \coloneqq \sup_{\delta x} \frac{\left(\|\delta f\| / \|f(x)\|\right)}{\left(\|\delta x\| / \|x\|\right)} = \frac{\|J(x)\|}{\left(\|f(x)\| / \|x\|\right)}.$$

Example 2.1.2 Conditional Number of Functions

1. $f(x) = \frac{1}{2}x$. $J(x) = \frac{1}{2}$.
$$\kappa(x) = \frac{\|J(x)\|}{\|f(x)\| / \|x\|} = \frac{1/2}{1/2} = 1.$$
2. $f(x) = \sqrt{x},\ x > 0$. $J(x) = \frac{1}{2\sqrt{x}}$.
$$\kappa(x) = \frac{\|J(x)\|}{\|f(x)\| / \|x\|} = \frac{1/(2\sqrt{x})}{\sqrt{x}/x} = \frac{1}{2}.$$

Definition 2.1.3 (Conditional Number of Matrices). Suppose $f(x) = Ax$, where $A \in \mathbb{C}^{n\times n}$ is invertible. Then, $J(x) = A$, and

$$\kappa(A) = \frac{\|J(x)\|}{\|f(x)\|/\|x\|} = \|A\| \cdot \frac{\|x\|}{\|Ax\|} \leq \|A\| \cdot \left\|A^{-1}\right\| = \kappa.$$

Proof 1. $\|x\| = \left\|A^{-1}Ax\right\| \leq \left\|A^{-1}\right\| \cdot \|Ax\|$. So,

$$\frac{\|x\|}{\|Ax\|} \leq \left\|A^{-1}\right\| \implies \|A\| \cdot \frac{\|x\|}{\|Ax\|} \leq \|A\| \cdot \left\|A^{-1}\right\|.$$

Q.E.D. ■

1. If $A \in \mathbb{C}^{m\times n}$, then $\kappa = \|A\| \cdot \left\|A^{\dagger}\right\|$, where $A^{\dagger} = V\Sigma^{-1}U^{*}$ from SVD.
2. If $\|\cdot\| = \|\cdot\|_2$, then $\kappa_2 = \dfrac{\sigma_1}{\sigma_r} = \dfrac{\text{largest singular value}}{\text{smallest singular value}}$.

Remark 2.1 *Conditioning is something inherited to problems. We have no control over them. What we can control is the algorithm we use.*

Definition 2.1.4 (Well-Conditioned & Ill-Conditioned).

1. *Well-Conditioned*: κ is small, $\kappa \approx 1$.
2. *Ill-Conditioned*: κ is large: $\kappa \approx \dfrac{1}{\text{numerical accuracy}}$.

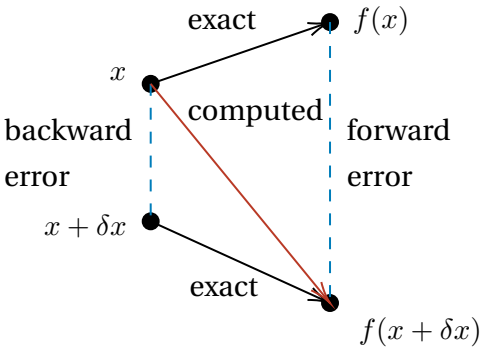
2.2 Backward Stability

Definition 2.2.1 (Stability). How an algorithm performs under perturbations.

Definition 2.2.2 (Backward Stability). Let $\text{alg}(x)$ be the algorithm we use to compute $f(x)$. We say $\text{alg}(x)$ is *backward stable for* $f(x)$ if $\forall\, x$, $\exists$ small δx $s.t.$

$$f(x) \approx \text{alg}(x) = f(x + \delta x)$$

Remark 2.2 *This definition indicates that we can* approximate *f(x) by* exactly solving a nearby problem *using the algorithm.*



2.3 Floating Point (FP) Numbers

Remark 2.3 (Main Takeaway) *Computers only approximate numbers.*

Definition 2.3.1 (Mathematically Representation of FP Numbers).

- $\mathbb{F} \subset \mathbb{R}$, the set of FP numbers.
- β, *base* (typically $\beta = 2, 10, 16$). $\beta \geq 2$, integer.
- t, *precision*, integer.
- $x \in \mathbb{F}$ if $x = 0$ or is written as

$$x = \pm\left(\frac{m}{\beta^t}\right)\beta^e,$$

where m is *mantissa*, significand, integer. For uniqueness,

$$\beta^{t-1} \leq m \leq \beta^t - 1 \implies \frac{m}{\beta^t} < 1.$$

Also, e is the *exponent*, integer.

Example 2.3.2 $x = 23.5$

- Write x out in base β:

  $\beta = 10$: $x = 2 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1}$

  $\beta = 2$: $x = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$

- Write in scientific notation:

  $\beta = 10$: $x = (0.235) \times 10^2$

  $\beta = 2$: $x = (0.101111)_2 \times 10^5$

- Write using the formula:

  $\beta = 10$, $t = 2$, then $10 \leq m \leq 99$.

$$\frac{m}{\beta^t} = 0.235 \implies m = 0.2235 \times 10^2 = 23.5$$

  However, m must be an integer, so we have to round: $m = 24$. So,

$$x = +\left(\frac{24}{10^2}\right) \times 10^2, \quad \text{\# of significant digit: } 2$$

  OTOH, $\beta = 10$, $t = 3$, then $100 \leq m \leq 999$. Then,

$$\frac{m}{\beta^t} = 0.235 \implies m = 0.235 \times 10^3 = 235.$$

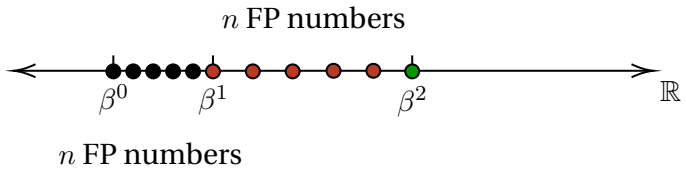
Then,

$$x = +\left(\frac{235}{10^3}\right) \times 10^3, \quad \text{\# of significant digit: } 3$$

Definition 2.3.3 (Normalized Version).

$$x = \pm\left(d_0 + \frac{d_1}{\beta} + \cdots + \frac{d_{t-1}}{\beta^{t-1}}\right)\beta^e,$$

where $0 \le d_0 < \beta$, integer.



Definition 2.3.4 (IEEE Standard). IEEE standard stores FP numbers in three parts:

	$s(x)$	$e(x)$	$f(x)$	total
double precision (DP)	1	11	52	64
single precision (SP)	1	8	23	32
half precision (HP)	1	5	10	16

Example 2.3.5

$s(x) = 0;\ e(x) = 10\cdots0111;\ f(x) = 111010\cdots0$ in double prevision under IEEE.

The fraction bit is

$$m = \frac{2^{51} + 2^{50} + 2^{49} + 2^{47}}{2^{52}} = \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{0}{2^4} + \frac{1}{2^5} + \cdots + \frac{0}{2^{52}} \approx 0.90625.$$

The exponent bit:

$$e = 2^{10} + \cdots + 2^2 + 2^1 + 2^0 = 1031.$$

The sign bit:

$$(-1)^{\text{sign bit}} = (-1)^0 = 1.$$

So,

$$x = +(1 + 0.90625) \cdot 2^8,$$

where 8 is the normalized exponent bit, $e(x) - 1023$.

6. Limitations of IEEE Standard

- Exponent Limitations:
  - too large $\Longrightarrow$ overflow $\Longrightarrow$ `Inf` $\longrightarrow$ fatal error, *but usually avoidable with rescaling*

– too small $\implies$ underflow $\implies$ `0`

Example 2.3.7

Let $c = \sqrt{a^2 + b^2}$. Suppose $a = 10^{170}$ and $b = 1$. Then,

$$c = \sqrt{\texttt{Inf} + 1} = \texttt{Inf} \implies \text{overflow}$$

However, if we do rescaling:

$$c = s\sqrt{\left(\frac{a}{s}\right)^2 + \left(\frac{b}{s}\right)^2} = 10^{170}\sqrt{1+0} = 10^{170}, \implies \text{underflow}$$

where $s = \max\{a, b\}$.

- Fraction Limitations: *rounding*

2.4 Fundamental Theorem of FP Arithmetic and Error

Definition 2.4.1 (Machine Epsilon/$\varepsilon_{\texttt{mach}}$).

$$\varepsilon_{\texttt{mach}} = \frac{1}{2}\beta^{1-t}$$

is

- the resolution of $\mathbb{F}$;
- half the distance from 1 to the next largest FP number;
- maximum relative error due to rounding.

Theorem 2.4.2 Error Storing Numbers as FP

$\forall\, x \in \mathbb{R}$, $\exists\, \varepsilon$ with $|\varepsilon| \leq \varepsilon_{\texttt{mach}}$ $s.t.$

$$\text{fl}(x) = x(1+\varepsilon).$$

Remark 2.4 *Rewrite, and we can get*

$$\text{fl}(x) = x + x\varepsilon \implies \frac{\text{fl}(x) - x}{x} = \varepsilon, \quad \textit{relative error}$$

Example 2.4.3

Suppose $x = \pi \approx 3.14159\ldots$, and we have a computer base-10 with 3 significant digit. Then, $\text{fl}(x) = 3.14$ and $\beta = 10$. Note that $\varepsilon_{\texttt{mach}} = \frac{1}{2}10^{-3}$. Then, $\frac{\text{fl}(x) - x}{x} \approx -5.07 \times 10^{-4} = \varepsilon$. $|\varepsilon| < \varepsilon_{\texttt{mach}}$.

Notation 2.4. Let $\star$ represent one of the four operations: $+$, $-$, $\times$, and $\div$.

- Exact arithmetic: $x, y \in \mathbb{R}$, $x \star y$.
- FP arithmetic: $x, y \in \mathbb{F}$, $x \circledast y = \text{fl}(x \star y)$

Theorem 2.4.5 Fundamental Theorem of FP Arithmetic
$\forall\, x, y \in \mathbb{F}$, $\exists\, \varepsilon$ with $|\varepsilon| \leq \varepsilon_{\texttt{mach}}$ $s.t.$

$$x \circledast y = (x \star y)(1+\varepsilon).$$

Remark 2.5

- *relative error* $= \dfrac{x \circledast y - x \star y}{x \star y} = \varepsilon$.
- *Occasionally, we have to redefine* ε_{mach} *with* $2\varepsilon_{mach}$.
- *Complex arithmetic, similar analysis with larger* ε_{mach}*:*

$$(a+b\text{i})(c+d\text{i}) = (ac-bd)+(ad+bc)\text{i} \implies \textit{more operations involved}$$

Example 2.4.6 Is FP Arithmetic Stable?

Set-Up: $f(x_1, x_2) = x_1 + x_2$ and $\text{alg}(x_1, x_2) = \text{fl}(x_1) \oplus \text{fl}(x_2)$.

Analysis: $\text{fl}(x_1) = x_1(1+\varepsilon_1)$, with $|\varepsilon_1| \leq \varepsilon_{\texttt{mach}}$. Also, $\text{fl}(x_2) = x_2(1+\varepsilon_2)$, with $|\varepsilon_2| \leq \varepsilon_{\texttt{mach}}$. Hence,

$$\text{fl}(x_1) \oplus \text{fl}(x_2) = (\text{fl}(x_1) + \text{fl}(x_2))(1+\varepsilon_3) \quad \text{with } |\varepsilon_3| \leq \varepsilon_{\texttt{mach}}.$$

Combining everything, we have

$$\begin{aligned}
\text{fl}(x_1) \oplus \text{fl}(x_2) &= \big[x_1(1+\varepsilon_1) + x_2(1+\varepsilon_2)\big](1+\varepsilon_3) \\
&= x_1(1+\varepsilon_1)(1+\varepsilon_3) + x_2(x+\varepsilon_2)(1+\varepsilon_3) \\
&= x_1(1+\varepsilon_1+\varepsilon_3+\varepsilon_1\varepsilon_3) + x_2(1+\varepsilon_2+\varepsilon_3+\varepsilon_2\varepsilon_3) \\
&= x_1(1+\varepsilon_4) + x_2(1+\varepsilon_5)
\end{aligned}$$

Note that

$$\begin{aligned}
|\varepsilon_4| &= |\varepsilon_1 + \varepsilon_3 + \varepsilon_1\varepsilon_3| \\
&\leq |\varepsilon_1| + |\varepsilon_2| + |\varepsilon_1\varepsilon_3| \\
&\leq 2\varepsilon_{\texttt{mach}} + \mathcal{O}\big(\varepsilon_{\texttt{mach}}^2\big).
\end{aligned}$$

So, $alg(x_1, x_2) = f(x_1, x_2) + \underbrace{\varepsilon_4 x_1 + \varepsilon_5 x_5}_{\text{forward error}}$

New Question: Is it backward stable? i.e., can we find nearby $\widetilde{x}_1$ and $\widetilde{x}_2$ $s.t.$ $\text{alg}(x_1, x_2) = f(\widetilde{x}_1, \widetilde{x}_2)$?

Define $\widetilde{x}_1 = x_1(1+\varepsilon_4)$ and $\widetilde{x}_2 = x_2(1+\varepsilon_5)$. Then,

$$\frac{|\widetilde{x}_1 - x_1|}{|x_1|} = \mathcal{O}(\varepsilon_{\texttt{mach}}) \longrightarrow \text{small}$$

So, the algorithm is equal to performing exact arithmetic over nearby numbers.

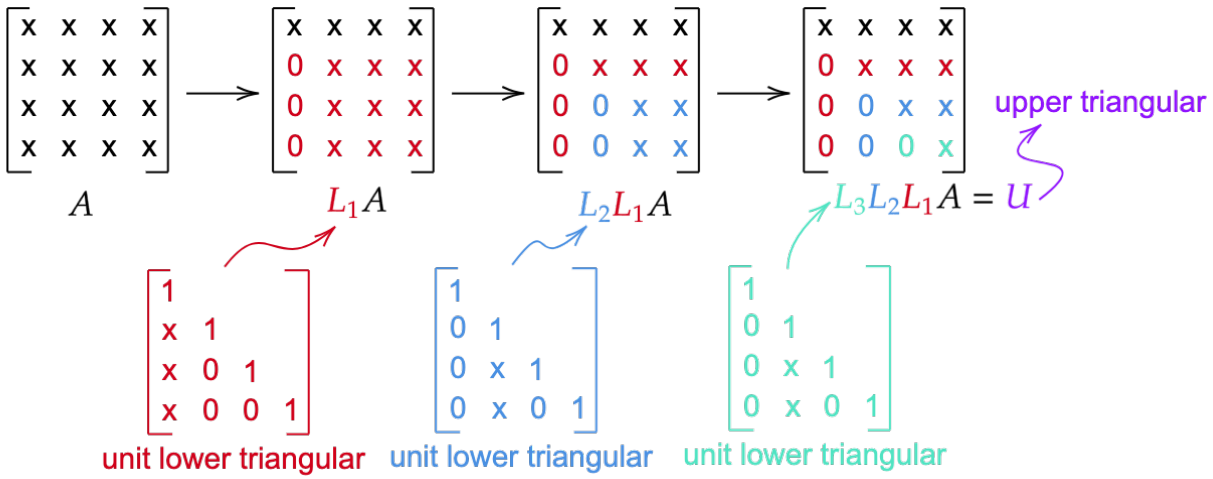
3 Linear Systems of Equations

3.1 Gaussian Elimination & LU Factorization

1. Setting

We will have $A \in \mathbb{C}^{m\times m}$ throughout this section.

2. Gaussian elimination in picture



3. General Formulas and Two Strokes of Luck

- Look at k^{th} column of A:

$$a_k = \begin{bmatrix} a_{1,k} \\ \vdots \\ a_{k-1,k} \\ a_{k,k} \\ a_{k+1,k} \\ \vdots \\ a_{m,k} \end{bmatrix} \xrightarrow{L_k} L_k a_k = \begin{bmatrix} a_{1,k} \\ \vdots \\ a_{k-1,k} \\ a_{k,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

In words, subtract $l_{j,k}$ (row k) from (row j):

$$l_{j,k} = \frac{a_{j,k}}{a_{k,k}} \quad (k < j \le m)$$

$$L_k = \begin{bmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{k+1,k} & \ddots & \\ & & \vdots & & \ddots \\ 0 & & -l_{m,k} & 0 & 1 \end{bmatrix}$$

- **Lucky Break** $\#1$: L_k is easy to invert: $\det(L_k) = 1$.

$$L_k^{-1} = \begin{bmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{k+1,k} & \ddots & & \\ & & \vdots & & \ddots & \\ 0 & & l_{m,k} & 0 & & 1 \end{bmatrix}$$

  Another representation of L_k and L_k^{-1}. Let

$$l_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ l_{k+1,k} \\ \vdots \\ l_{m,k} \end{bmatrix}.$$

  Then,

$$L_k = I - l_k e_k^* \quad \text{and} \quad L_k^{-1} = I + l_k e_k^*.$$

- **Lucky Break** $\#2$: $L_k^{-1} L_{k+1}^{-1}$ is still unit lower triangular.

  Proof 1. Use the previously defined notation:

$$\begin{aligned} L_k^{-1} L_{k+1}^{-1} &= (I + l_k e_k^*)(I + l_{k+1} e_{k+1}^*) \\ &= I + l_k e_k^* + l_{k+1} e_{k+1}^* + l_k e_k^* l_{k+1} e_{k+1}^* \\ &= I + l_k e_k^* + l_{k+1} e_{k+1}^* + \underbrace{(e_k^* l_{k+1})}_{=0} l_k e_{k+1}^* \\ &= I + l_k e_k^* + l_{k+1} e_{k+1}^* \\ &= \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & l_{k+1,k} & 1 & & & \\ & & \vdots & l_{k+2,k+1} & \ddots & & \\ & & \vdots & \vdots & & \ddots & \\ 0 & & l_{m,k} & l_{m,k+1} & 0 & \cdots & 1 \end{bmatrix} \end{aligned}$$

  Q.E.D. ■

- Punchline:

$$\begin{aligned} L_{m-1}\cdots L_2 L_1 A &= U \\ A &= \underbrace{L_1^{-1}L_2^{-1}\cdots L_{m-1}^{-1}}_{=L} U \\ A &= LU, \end{aligned}$$

where $L = \begin{bmatrix} 1 & & & & & & 0 \\ l_{2,1} & 1 & & & & & \\ \vdots & l_{3,2} & \ddots & & & & \\ \vdots & \vdots & \ddots & 1 & & & \\ \vdots & \vdots & & l_{k+1,k} & \ddots & & \\ \vdots & \vdots & & \vdots & & \ddots & \\ l_{m,1} & l_{m,2} & \cdots & l_{m,k} & \cdots & \cdots & 1 \end{bmatrix}$, and $l_{j,k} = \dfrac{a_{j,k}}{a_{k,k}}$.

Assumption: $a_{k,k} \neq 0$

Algorithm 1: LU Factorization

Input: $U = A$; $L = I$;

```
1 begin
2     for k = 1 → m − 1 do
          // loop over columns
3         for j = k + 1 → m do
              // loop over rows below diagonal
4             L(j,k) = U(j,k) / U(k,k);          /* building multiplier */
5             U(j, k : m) = U(j, k : m) − L(j,k) ∗ U(k, k : m);
```

4. How expensive is the algorithm? Operation Count (`flops`)

At k-th step, for row j,

$$U(j,k:m) = U(j,k:m) - L(j,k) * U(k,k:m)$$

- number of $*$: $m-k+1$
- number of $-$: $m-k+1$

$$L(j,k) = \frac{U(j,k)}{U(k,k)}$$

- number of division: 1

At k-th step, for all rows, $j = k+1$ to m:

$$(m-k)(1+(m-k+1)+(m-k+1)) \quad \texttt{flops}$$

For $k = 1$ to $m-1$:

$$\begin{aligned}
&\sum_{k=1}^{m-1}(m-k)(1+(m-k+1)+(m-k+1)) \quad \texttt{flops} \\
&= \sum_{k=1}^{m-1} 2k^2 - 8mk - 3k + 3m + 2m^2 \\
&= \sum_{k=1}^{m-1} 2(\underbrace{m-k}_{m-1 \text{ to } 1})^2 + 3(m-k) = \sum_{k=1}^{m-1} 2k^2 + 3k.
\end{aligned}$$

As $m \to \infty$,

$$\int_1^m 2x^2 + 3x \, \mathrm{d}x = \frac{2}{3}m^3 + \text{smaller things} \implies \text{Work for GE: } \sim \frac{2}{3}m^3 \ \texttt{flops}.$$

Algorithm 2: Solve Linear System with GE

Input: $A = LU$

1 **begin**
2 $y \coloneqq Ux$;
3 Solve $Ly = b$ // L: lower triangular; forward substitution $\sim m^2$
4 Solve $Ux = y$ // U: upper triangular; backward substitution $\sim m^2$

Output: x *s.t.* $Ax = b$

Example 3.1.5

$$L = \begin{bmatrix} 1 & & \\ 2 & 1 & \\ 3 & 4 & 1 \end{bmatrix}; \quad U = \begin{bmatrix} 2 & 1 & 0 \\ & 10 & 3 \\ & & 7 \end{bmatrix}; \quad b = \begin{bmatrix} -4 \\ -5 \\ 7 \end{bmatrix}.$$

- Solve $Ly = b$:

$$\begin{bmatrix} 1 & & \\ 2 & 1 & \\ 3 & 4 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_4 \end{bmatrix} = \begin{bmatrix} -4 \\ -5 \\ 7 \end{bmatrix} \implies \begin{cases} y_1 & = -4 \\ 2y_1 + y_2 & = -5 \\ 3y_1 + 4y_2 + y_4 & = 7 \end{cases} \implies \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 3 \\ 7 \end{bmatrix}.$$

- Solve $Ux = y$:

$$\begin{bmatrix} 2 & 1 & 0 \\ & 10 & 3 \\ & & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 3 \\ 7 \end{bmatrix} \implies \begin{cases} 2x_1 + 2x_2 & = -4 \\ 10x_2 + 3x_3 & = 3 \\ 7x_3 & = 7 \end{cases} \implies \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix}.$$

Example 3.1.6 Instability of GE

- Complete Failure: Suppose

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \implies \kappa_2(A) = \frac{3+\sqrt{5}}{2} \implies A \text{ is well-conditioned}$$

but we still cannot apply GE on A. So, conditioning and stability are two different things.

- Slightly perturbed system: Suppose

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1-10^{20} \end{bmatrix} = LU$$

However, on a computer with $\varepsilon_{\texttt{mach}} = 10^{-6}$, we have

$$\widetilde{L} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad \text{and} \quad \widetilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix}.$$

Note that $\widetilde{L}$ is close to L, and $\widetilde{U}$ is close to U. So, GE (LU factorization) is *forward stable*. However,

$$\widetilde{L}\widetilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix} \not\approx A.$$

As $\widetilde{L}\widetilde{U}$ is not close to input matrix A, GE is *not backward stable*.

Further, if we solve $Ax = b$, where $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Then,

$$LUx = b \implies x = \begin{bmatrix} -1 \\ 1 \end{bmatrix}; \quad \widetilde{L}\widetilde{U}x = b \implies x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The computed result is still not closed to exact arithmetic.

Theorem 3.1.7 Summary on (In)Stability of GE
GE computes LU stably (i.e., $\widetilde{L}$ and $\widetilde{U}$ are close to exact L and U), but it does not solve $Ax = b$ stably. Hence, LU factorization is stable but not backward stable.

3.2 Pivoting

Definition 3.2.1 (Pivot/Pivotting). *Pivot* is the number/entry we divide by to construct multiplier:

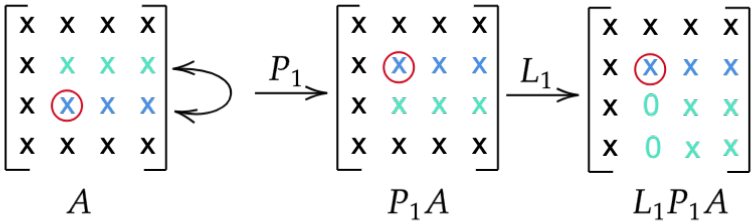
$$l_{j,k} = \frac{x_{j,k}}{x_{k,k}}.$$

$$\begin{bmatrix} \star & \star & \star & \star \\ 0 & x_{k,k} & \star & \star \\ 0 & \star & \star & \star \\ 0 & \star & \star & \star \end{bmatrix}$$

Remark 3.1 *We don't have to always use diagonal as the pivot. We can* permute.

2. Partial Pivotting

Overview: swap rows and create zeros



$$L_{m-1}P_{m-1}\cdots L_2P_2L_1P_1A = U$$

3. Lucky Break #3

$$\underbrace{L_{m-1}P_{m-1}\cdots L_2P_2L_1P_1}\,A = U$$
$$(L'_{m-1}\cdots L'_2L'_1)(P_{m-1}\cdots P_2P_1)A = U,$$

where

$$L'_{m-1} = L_{m-1}, \quad L'_{m-2} = P_{m-1}L_{m-2}P_{m-1}^{-1}, \quad \cdots$$

Proof 1. Inthis proof, we aim to show that $L'_{m-2} = P_{m-1}L_{m-2}P_{m-1}^{-1}$:

$$\begin{aligned} L_{m-1}P_{m-1}L_{m-2}P_{m-2}\cdots L_1P_1 &= L_{m-1}P_{m-2}L_{m-2}IP_{m-2}\cdots L_1P_1 \\ &= L_{m-1}P_{m-1}L_{m-2}\big(P_{m-1}^{-1}P_{m-1}\big)P_{m-2}\cdots L_1P_1 \\ &= L_{m-1}\big(P_{m-1}L_{m-2}P_{m-1}^{-2}\big)P_{m-1}P_{m-2}\cdots L_1P_1 \\ &= L_{m-1}L'_{m-2}P_{m-1}P_{m-2}\cdots L_1P_1 \end{aligned}$$

Q.E.D. ■

Claim 3.4 $L'_{m-1},\ L'_{m-2},\ L'_{m-3}, \ldots, L'_1$ are still lower triangular matrices.

Theorem 3.2.5 GEPP

$$\begin{aligned} L_{m-1}P_{m-1}\cdots L_1P_1A &= U \\ \left(L'_{m-1}L'_{m-2}\cdots L'_1\right)(P_{m-1}\cdots P_1)A &= U \\ PA &= LU \end{aligned}$$

Remark 3.2 (How to choose a privot?) *Choose entry on or below diagonal in a column that has the largest magnitude:*

$$l_{j,k} = \frac{x_{j,k}}{x_{k,k}}.$$

Note that if $x_{k,k}$ is large, we will have underflow, so $l_{j,k} = 0$. If $x_{k,k}$ is small, we will have overflow, so $l_{j,k} =$ `Inf`, which is fatal. However, as we pick $x_{k,k}$, out pivot, as the largest magnitude entry in each column, we know L has lower triangular entires with magnitude ≤ 1.

Algorithm 3: An Unrealistic GEPP Algorithm

```
1 begin
2     Permute rows of A with P;
      // we don't know the true value of P!
3     Use GE on PA = LU;
```

Algorithm 4: GEPP in Practice

Input: $U = A; \quad L = I; \quad P = I$

```
1 begin
2     for k = 1 : m − 1 do
3         Select i ≥ k to maximize |U(i, k)|;
          // on the k-th column; on or below diagonal entry
4         Swap rows:
5             U(k, k : m) ⟷ U(i, k : m)
6             P(k, :) ⟷ P(i, :)
7             L(k, 1 : k − 1) ⟶ L(i, 1 : k − 1)
8         Do j-loop in Algorithm (1)
```

Output: $PA = LU$

Remark 3.3

- *Cost of GEPP is the same as GE in* `flops`

- *Representing matrix P: we don't need to store the matrix. We only need the incides.*
- *Solving $Ax = b$ with $PA = LU$:*
  - $PAx = Pb$
  - *Solve $LUx = Pb$.*
- *Complete Pivoting: Search for the largest entry in magnitude in the entire sub-matrix.*

$$PAQ = LU,$$

where Q is responsible for columns swaps.

3.3 Choleksy Factorization

Remark 3.4 *It is the "LU factorization for Hermitian matrices."*

Definition 3.3.1 (Hermitian). $A \in \mathbb{C}^{m\times m}$ is *Hermitian positive definite* (symmetric positive definite, SPD), if

- $A = A^*$, and
- $x^*Ax > 0 \quad \forall\, x \neq 0 \quad \longrightarrow \quad A$ has positive eigenvalues.

2. GE for SPD

Suppose A is SPD:

$$A = \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix} \qquad \begin{bmatrix} w\in\mathbb{C}^{m-1} \\ K\in\mathbb{C}^{(m-1)\times(m-1)} \quad \text{is SPD} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ w & I \end{bmatrix} \begin{bmatrix} 1 & w^* \\ 0 & K - ww^* \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \\ w & I \end{bmatrix}}_{R_1^*} \begin{bmatrix} 1 & 0 \\ 0 & \boxed{K - ww^*} \end{bmatrix} \underbrace{\begin{bmatrix} 1 & w^* \\ 0 & I \end{bmatrix}}_{R_1} \qquad \text{[by symmetry]}$$

Note that $K - ww^*$ is also SPD, so we can form a recursive algorithm.

Claim 3.3 K is SPD.

Proof 1. Note that

$$\begin{bmatrix} 0 & y \end{bmatrix} \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix} \begin{bmatrix} 0 \\ y \end{bmatrix} = y^*Ky > 0$$

since A is SPD. Then, K must also be SPD. Q.E.D. ■

Theorem 3.3.4
Every SPD has a unique Cholesky factorization.

Theorem 3.3.5 Cholesky Factorization

Suppose A is SPD, then

$$
\begin{aligned}
A &= \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} && [a_{11} > 0] \\
&= \begin{bmatrix} \alpha & 0 \\ w/\alpha & \end{bmatrix} \begin{bmatrix} 1 & \\ & K - \dfrac{ww^*}{\alpha} \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & I \end{bmatrix} && [\alpha = \sqrt{a_{11}}] \\
&= R_1^* A_1 R_1 \\
&= R_1^* R_2^* A_2 R_2 R_1 && [\text{recursively doing the facotrization}] \\
&= (R_1^* R_2^* \cdots R_m^*)(R_m \cdots R_2 R_1) && [R_i : \text{ upper triangular matrix}] \\
&= R^* R && [R = R_m \cdots R_2 R_1, \text{ with } r_{i,i} > 0]
\end{aligned}
$$

Algorithm 5: Cholesky Facotrization

Input: SPD matrix A $\quad R = \texttt{triu}(A)$;

`// triu(`A`) returns a triangular matrix that retains the upper part of the matrix A`

begin
  for $k = 1 : m$ **do**
    for $j = k + 1 : m$ **do**
      $R(j, j : m) = R(j, j : m) - R(k, j : m) \cdot \dfrac{\overline{R}(k, j)}{R(k, k)}$;
    $R(k, k : m) = R(k, k : m)/\sqrt{R(k, k)}$;

Remark 3.5 (Operation Count and Comparison with LU Facotrization)

$$\textit{Operation Count: } \sim \frac{1}{3} m^3.$$

Operation count for LU is $\sim \dfrac{2}{3} m^3$. *As we have symmetry here, we get a cheaper algorithm.*

3.4 Other Special Matrices/Factorization

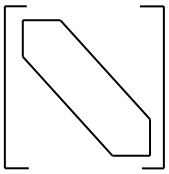
- $A = LDM^*$
  - L and M are unit upper/lower triangular matrices;
  - D is a diagonal matrix.

Proof 1.

$$LU = LD \underbrace{\left(D^{-1}U\right)}_{M^*}$$

Q.E.D. ■

- If A is Hermitian, $A = LDL^*$.
- Banded matrices:



  - b_L and b_U denote the lower and upper bandwidth.
  - $a_{ij} = 0$ for $i > j + b_L$ and $i < j - b_U$.

  Claim 3.1 A is banded and $A = LU$. Then, L and U are banded as well.

- Sparse Matrices: A has lots of 0 entries.

  If A is sparse, $A = LU$, then L and U may not necessarily be sparse.

4 Stability of Solving Linear Systems

4.1 (In)Stability of GE & GEPP

Remark 4.1 *For an overview of Big-Oh notations and how to interprete it in the context of stability, refer to Section 4.5.*

Theorem 4.1.1 Stability of GE
Suppose $A = LU$, where $A =\in \mathbb{C}^{m\times m}$, without pivoting.
If A has LU factorization, then for sufficiently small $\varepsilon_{\texttt{mach}}$, the factorization can be done successfully in FP arithmetic, and $\widetilde{L}$ and $\widetilde{U}$ satisfy

$$\widetilde{L}\widetilde{U} = A + \delta A \quad (\delta A = \widetilde{L}\widetilde{U} - A = \widetilde{L}\widetilde{U} - LU),$$

then

$$\frac{\|\delta A\|}{\|L\| \cdot \|U\|} = \mathcal{O}(\varepsilon_{\texttt{mach}}),$$

which measures how close the product is to our original A.

- If $\|L\| \cdot \|U\| = \mathcal{O}(\|A\|)$, then LU factorization is backward stable.
- If not, we could have instability.

Example 4.1.2 Example 3.1.6 – Revisit

$$A = \begin{bmatrix} 10^{-2} & 1 \\ 1 & 1 \end{bmatrix} \implies L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}.$$

Then,

$$\|L\| \cdot \|U\| \gg \mathcal{O}(\|A\|).$$

So, we have instability when solve $Ax = b$ with LU factorization.

3. Growth Factors for GEPP

- $\|L\| = \mathcal{O}(1)$ (recall the construction of L, each entry is less than 1 magnitude). So, all we worry about this
$$\frac{\|\delta A\|}{\|U\|} = \mathcal{O}(\varepsilon_{\texttt{mach}}).$$
Or, equivalently, if $\|U\| = \mathcal{O}(\|A\|)$, then GEPP is backward stable.
- **Definition 4.1.4 (Growth Factor).** Define the growth factor of $PA = LU$ as
$$\rho = \frac{\max\limits_{i,j} |U_{i,j}|}{\max\limits_{i,j} |A_{i,j}|} \implies \|U\| = \mathcal{O}(\rho\|A\|).$$

In practice, we want ρ to be small, i.e., $\rho \approx 1$.

Theorem 4.1.5 Stability of GEPP
Given $PA = LU$, then our computed solution

$$\widetilde{L}\widetilde{U} = \widetilde{P}A + \delta A,$$

where

$$\frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\rho\varepsilon_{\texttt{mach}}).$$

- If $|l_{i,j}| < 1, i > j, \implies$ no ties $\implies$ unique pivot per column $\implies \widetilde{P} = P$.
- If $\rho = \mathcal{O}(1)$ uniformly for all matrices of a given dimension m, then GEPP is stable.

Example 4.1.6 Worst Case Instability

$$A = \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & \vdots \\ \vdots & -1 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ -1 & -1 & \cdots & -1 & 1 \end{bmatrix}; \quad U = \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & \ddots & & 4 \\ & & & \ddots & \vdots \\ & & & & 2^{m-1} \end{bmatrix}; \quad L = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ -1 & -1 & \cdots & -1 & 1 \end{bmatrix}$$

We do $PA = LU$, GEPP:

- What is the growth factor?

$$\rho = \frac{2^{m-1}}{1} = 2^{m-1} (= \mathcal{O}(2^m) \sim \mathcal{O}(1)), \text{ constant } w.r.t.\ \|A\|.$$

  - Growth factor $\mathcal{O}(2^m)$ corresponds to a loss on the order of m bits of precision.

$$\frac{\|\delta A\|}{\|A\|} \leq c\rho\varepsilon_{\texttt{mach}}.$$

    $\implies \delta A$ can be perturb inputs by a magnitude around 2^m.

    This can be catastrophic: double precision$= 64$ bits.

    We are in trouble when we get about 10×10 *matrix??!! No!*

  - This is an awkward part of the theorem. The theory was for fixed m. It never required uniformty in m. We still have a constant bound in ρ. *For example, the following is problematic:*

$$\rho = 2^{\|A\|}, \quad \rho = 2\|A\| \sim \mathcal{O}(\|A\|).$$

So, as long as ρ does not depend on $\|A\|$, we are good.

– GEPP is *backward stable* even if we are in the worst case.

4.2 Stability of Backward Substitution

Theorem 4.2.1 Stability of Backward Substitution
Given $Ux = b$ with U upper triangular. Backward substitution is backward stable.

$$(U + \delta U)\widetilde{x} = b,$$

with

$$\frac{\|\delta U\|}{\|U\|} = \mathcal{O}(\varepsilon_{\texttt{mach}}).$$

Specifically,

$$\frac{|\delta U_{i,j}|}{|U_{i,j}|} \leq m\varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2).$$

Proof 1.

- When $m = 1$. Solve $U_{1,1}x_1 = b_1$:

  $$\widetilde{x}_1 = b_1 \oslash U_{1,1}$$

  By the Fundamental Theorem of FP Arithmetic:

  $$\widetilde{x}_1 = \left(\frac{b_1}{U_{1,1}}\right)(1 + \varepsilon_1), \quad |\varepsilon_1| \leq \varepsilon_{\texttt{mach}}$$

  Write division with only perturbation on $U_{1,1}$:

  $$\widetilde{x}_1 = \frac{b_1}{U_{1,1}(1 + \varepsilon_1')}, \quad \varepsilon_1' = \frac{-\varepsilon_1}{1 + \varepsilon_1} = -\varepsilon_1(1 - \varepsilon_1 + \varepsilon_1^2 - \varepsilon_1^3 + \cdots) \quad \text{[Geometric Series]}$$

  Then, $|\varepsilon_1'| \leq \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2)$.

  $$\implies \frac{|\delta U_{1,1}|}{|U_{1,1}|} \leq \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2).$$

- When $m = 2$. Solve $\begin{bmatrix} U_{1,1} & U_{1,2} \\ & U_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$.

  $$\widetilde{x}_2 = b_2 \oslash U_{2,2} = \frac{b_2}{U_{2,2}(1 + \varepsilon_1')}, \quad |\varepsilon_1'| \leq \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2).$$

Also,

$$\begin{aligned}
\widetilde{x}_1 &= \big[b1 \ominus (U_{1,2} \otimes \widetilde{x}_2)\big] \oslash U_{1,1} \\
&= \big[b_1 \ominus (U_{1,2}\widetilde{x}_2(1+\varepsilon_2))\big] \oslash U_{1,1} \\
&= [(b_1 - U_{1,2}\widetilde{x}_2(1+\varepsilon_2))(1+\varepsilon_3)] \oslash U_{1,1} \\
&= \left[\frac{\big(b_1 - U_{1,2}\widetilde{x}_2(1+\varepsilon_2)\big)(1+\varepsilon_3)}{U_{1,1}}\right](1+\varepsilon_4) \\
&= \frac{b_1 - U_{1,2}\widetilde{x}_2(1+\varepsilon_2)}{U_{1,1}\underbrace{(1+\varepsilon_3')(1+\varepsilon_4')}_{(1+2\varepsilon_5)}} \\
&= \frac{b_1 - U_{1,2}\widetilde{x}_2(1+\varepsilon_2)}{U_{1,1}(1+2\varepsilon_5)}, \qquad [|\varepsilon_2|, |\varepsilon_5| \le \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2).]
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{|\delta U_{1,2}|}{|U_{1,2}|} &= |\varepsilon_2| \le \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2) \\
\frac{|\delta U_{1,1}|}{|U_{1,1}|} &= 2|\varepsilon_5| \le \underbrace{\boxed{2}}_{m}\, \varepsilon_{\texttt{mach}} + \mathcal{O}(\varepsilon_{\texttt{mach}}^2)
\end{aligned}$$

- When $m = 3$ onwards, error is accumulated when we do substitution more and more times.

Q.E.D. ■

4.3 Perturbation Theory of Linear Systems

Set Up:

Problem

$$Ax = b, \quad \text{where } A \text{ is invertible, and } x \text{ is the exact solution} \tag{P}$$

Perturbed Problem

$$(A + \delta A)\widehat{x} = b + \delta b, \quad \text{where } (A + \delta A) \text{ assumed invertible, and } \widehat{x} \text{ computed solution} \tag{PP}$$

Error in Solution

$$\delta x = \widehat{x} - x. \tag{E}$$

Goal: How big δx is relative to x (find an upper bound)

From (E), we have

$$\widehat{x} = x + \delta x.$$

Plug into (PP), we have

$$
\begin{aligned}
(A+\delta A)(x+\delta x) &= b+\delta b \\
\underbrace{\cancel{Ax}}_{=b} + A\delta x + \delta A x + \delta A \delta x &= \cancel{b} + \delta b \\
(A+\delta A)\delta x &= \delta b - \delta A x \qquad \text{[\textbf{Assumption:} } A+\delta A \text{ invertible]} \\
\delta x &= (A+\delta A)^{-1}(\delta b - \delta A x) \\
\|\delta x\| &\le \left\|(A+\delta A)^{-1}\right\| \cdot \|\delta b - \delta A x\| \\
\frac{\|\delta x\|}{\|x\|} &\le \frac{\left\|(A+\delta A)^{-1}\right\| \cdot \|\delta b - \delta A x\|}{\|x\|} \qquad \text{(Goal)}
\end{aligned}
$$

Lemma 4.1 : If $\|X\| \le 1$, then

- $I - X$ is invertible.
- $(I-X)^{-1} = \sum_{i=0}^{\infty} X^i$
- $\left\|(I-X)^{-1}\right\| \le \dfrac{1}{1-\|X\|}$

Proof 1.

- By contradiction, $(I-X)z = 0,\ z \neq 0 \implies \text{null}(I-X) = \varnothing$
- Show $\left(\sum_{i=0}^{\infty} X^i\right)(I-X) = I - X^{N+1}$.

$$\left\|X^k\right\| \le \|X\|^k \implies \text{as } k \to \infty,\ \left\|X^k\right\| \to 0.$$

- Consider

$$
\begin{aligned}
\left\|(I-X)^{-1}\right\| &= \left\|\sum_{i=0}^{\infty} X^i\right\| \qquad \left[\sum_{i=0}^{\infty} X^i \text{ converges, so we can use triangle inequality safely.}\right] \\
&\le \sum_{i=0}^{\infty} \|X\|^i
\end{aligned}
$$

Note that $\sum_{i=0}^{\infty} \|X\|^i$ is a geometric series, then

$$\sum_{i=0}^{\infty} \|X\|^i = \frac{1}{1-\|X\|}.$$

So, we have

$$\left\|(I-X)^{-1}\right\| \le \frac{1}{1-\|X\|}.$$

Q.E.D. ■

Use Lemma 3.1 to simply (Goal)

$$\left\|(A+\delta A)^{-1}\right\| = \left\|\left(A\left(I+A^{-1}\delta A\right)\right)^{-1}\right\| \leq \left\|A^{-1}\right\| \cdot \left\|\left(I+A^{-1}\delta A\right)^{-1}\right\|$$

Assumption: $\left\|A^{-1}\delta A\right\| \leq \left\|A^{-1}\right\| \cdot \|\delta A\| < 1$ (*In order to use Lemma 3.1*)

Now, we can apply Lemma 3.1:

$$\begin{aligned}\left\|(A+\delta A)^{-1}\right\| &\leq \left\|A^{-1}\right\| \cdot \left\|\left(I+A^{-1}\delta A\right)^{-1}\right\| \\ &\leq \left\|A^{-1}\right\| \cdot \frac{1}{1-\|A^{-1}\delta A\|} \\ &\leq \left\|A^{-1}\right\| \cdot \frac{1}{1-\|A^{-1}\| \cdot \|\delta A\|}\end{aligned}$$

Now, let's go back to (Goal):

$$\begin{aligned}\frac{\|\delta x\|}{\|x\|} &\leq \frac{\left\|(A+\delta A)^{-1}\right\| \cdot \|\delta b - \delta A x\|}{\|x\|} \\ &\leq \left(\frac{\left\|A^{-1}\right\|}{1-\|A^{-1}\| \cdot \|\delta A\|}\right) \cdot \left(\frac{\|\delta b - \delta A x\|}{\|x\|}\right) && \text{[Lemma 3.1]} \\ &\leq \left(\frac{\left\|A^{-1}\right\|}{1-\|A^{-1}\| \cdot \|\delta A\|}\right) \cdot \left(\frac{\|\delta b\| + \|\delta A\| \cdot \|x\|}{\|x\|}\right) && \text{[Triangle Inequality \& Multiplicity]} \\ &= \left(\frac{\left\|A^{-1}\right\|}{1-\|A^{-1}\| \cdot \|\delta A\| \left(\frac{\|A\|}{\|A\|}\right)}\right) \cdot \left(\frac{\|\delta b\| + \|\delta A\| \cdot \|x\|}{\|x\|}\right) \cdot \left(\frac{\|A\|}{\|A\|}\right) && \text{[multiply by magic 1]} \\ &= \left(\frac{\overbrace{\|A\| \cdot \left\|A^{-1}\right\|}}{1-\underbrace{\|A\| \cdot \left\|A^{-1}\right\|}\cdot\frac{\|\delta A\|}{\|A\|}}\right) \cdot \left(\frac{\|\delta b\|}{\|A\| \cdot \|x\|} + \frac{\|\delta A\|}{\|A\|}\right) \\ &= \left(\frac{\kappa(A)}{1-\kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}}\right) \cdot \left(\frac{\|\delta b\|}{\|A\| \cdot \|x\|} + \frac{\|\delta A\|}{\|A\|}\right) && \text{[Definition of Condition \#]}\end{aligned}$$

Recall $Ax = b$, we have

$$\|b\| \leq \|A\| \cdot \|x\| \implies \frac{1}{\|b\|} \geq \frac{1}{\|A\| \cdot \|x\|}.$$

So,

$$\frac{\|\delta x\|}{\|x\|} \leq \left(\frac{\kappa(A)}{1-\kappa(A)\frac{\|\delta A\|}{\|A\|}}\right) \cdot \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|}\right),$$

where

- $\frac{\|\delta x\|}{\|x\|}$: the relative error of solution (typically unknown).

- $\dfrac{\kappa(A)}{1-\kappa(A)\dfrac{\|\delta A\|}{\|A\|}}$: how hard the problem is to solve.

  If $\|\delta A\|$ is small, this term $\approx \kappa(A)$.
- $\dfrac{\|\delta b\|}{\|b\|}+\dfrac{\|\delta A\|}{\|A\|}$: relative perturbation of the problem.

Example 4.3.2 Is this bound pessimistic or tight?

Consider

$$A=\begin{bmatrix}1 & 0\\ 0 & 10^{-6}\end{bmatrix},\quad b=\begin{bmatrix}1\\ 10^{-6}\end{bmatrix},\quad x=\begin{bmatrix}1\\ 1\end{bmatrix}.$$

Then, $\kappa_2(A)=10^6$. Consider

- $\delta b=\begin{bmatrix}10^{-6}\\ 0\end{bmatrix}$, we have $\widehat{x}=\begin{bmatrix}1+10^{-6}\\ 1\end{bmatrix}$ when solving $Ax=b+\delta b$. Then,

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty}\le \kappa_\infty(A)\cdot\left(\frac{\|\delta b\|_\infty}{\|b\|_\infty}\right) \qquad [\text{no perturbation on } A,\ \|\delta A\|=0]$$
$$\frac{10^{-6}}{1}\le \|A\|_\infty\left\|A^{-1}\right\|_\infty\left(\frac{10^{-6}}{1}\right)\implies 10^{-6}\le 1.$$

- $\delta b=\begin{bmatrix}0\\ 10^{-6}\end{bmatrix}$, so $\widehat{x}=\begin{bmatrix}1\\ 2\end{bmatrix}$. Then,

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty}\le \kappa_\infty(A)\cdot\left(\frac{\|\delta b\|_\infty}{\|b\|_\infty}\right)\implies 1\le 1.$$

In this case, we attain the upper bound.

So, theo bound is tight.

4.4 More Practical Perturbation Theory

Remark 4.2 (Problem in our previous perturbation theory) *We don't know perturbations δb and δA.*

Definition 4.4.1 (Residual). We define residual as

$$r:=A\widehat{x}-b.$$

Then, we have the following:

$$\widehat{x}=A^{-1}r+A^{-1}b$$
$$\delta x=\widehat{x}-x=\widehat{x}-A^{-1}b=A^{-1}r \qquad [Ax=b\implies x=A^{-1}b]$$
$$\|\delta x\|\le\left\|A^{-1}\right\|\cdot\|r\|.$$

Theorem 4.4.2 More Practical Perturbation Theory
$\exists\, \delta A \; s.t. \; (A+\delta A)\widehat{x} = b$ with

$$\|\delta A\| \leq \frac{\|r\|}{\|\widehat{x}\|}.$$

- If A is well-conditioned and residual norm is small, then $\widehat{x}$ is a good approximation of x.

Theorem 4.4.3 Point-wise Analysis
If $|\delta A_{ij}| \leq \varepsilon |A_{ij}|$, $|\delta b_i| \leq \varepsilon |b_i|$, and $\varepsilon\kappa(A) < 1$, then

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty} \leq \frac{2\varepsilon}{1-\varepsilon\kappa(A)} \left\| \left|A^{-1}\right| \cdot |A| \right\|_\infty,$$

where $\left\| \left|A^{-1}\right| \cdot |A| \right\|_\infty = \max\limits_{i,j} \left|A^{-1}_{ij}\right| |A_{ij}|$ is the component-wise relative condition number.

Example 4.4.4 Condition Number and Component-Wise Relative Condition Number
Suppose

$$A = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} \alpha \\ 1 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \text{with } \alpha > 1.$$

Then,

$$A^{-1} = \begin{bmatrix} 1/\alpha & 0 \\ 0 & 1 \end{bmatrix}.$$

So,

$$\kappa_\infty(A) = \left\|A^{-1}\right\|_\infty \|A\|_\infty = \alpha \quad \longrightarrow \quad \text{could be large}$$

$$\kappa_{\text{CR}}(A) = \left\| \left|A^{-1}\right| \cdot |A| \right\| = 1 \quad \longrightarrow \quad \text{much tighter}$$

4.5 Big-Oh Notation

Definition 4.5.1 (Big-Oh Notation). If

$$\varphi(t) = \mathcal{O}(\psi(t)),$$

then $\exists\, c > 0$ such that $\forall t$ sufficiently close to an understood limit (e.g. $t \to 0$ or $t \to \infty$),

$$|\varphi(t)| \leq c\psi(t).$$

Example 4.5.2

$$\frac{\|x - \widetilde{x}\|}{\|x\|} = \mathcal{O}(\varepsilon_{\texttt{mach}}) \implies \frac{\|x - \widetilde{x}\|}{\|x\|} \leq c\varepsilon_{\texttt{mach}},$$

where c cannot depend on $\varepsilon_{\texttt{mach}}$. So, we have

$$\|x - \widetilde{x}\| \leq c\varepsilon_{\texttt{mach}}\|x\|.$$

If $\|x\|$ is large, we have more wiggle room.

Example 4.5.3 GE vs. GEPP

Consider

$$A = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 \\ 1/\varepsilon & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - 1/\varepsilon \end{bmatrix}.$$

When $\varepsilon \to 0$, then $\|A\|_1 \to 2$, $\|L\|_1 \to \infty$, and $\|U\|_1 \to \infty$.

By Theorem 3.3.1 and Theorem 3.3.5, we have

$$\|\delta A\| \leq c\varepsilon_{\texttt{mach}} \underbrace{\|L\|}_{\to\infty} \underbrace{\|U\|}_{\to\infty}.$$

Hence, GE is not stable because it allows large $\|\delta A\|$.

To have backward stable, we require $\|L\| \cdot \|U\| = \mathcal{O}(\|A\|)$.

5 Least Squares

5.1 Least Square Problems

1. Least Square as an Optimization Problem

Set-Up: $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ (usually $m > n$). We call A is *overdetermined*.

Goal: find x that minimizes the 2-norm of the residual

$$r = b - Ax \quad \text{or} \quad r = Ax - b.$$

As an Optimization Problem:

$$\min_x \|Ax - b\|_2^2 = \min_x \sum_{i=1}^{m} |A(i,:)x - b_i|^2 \qquad \text{(Problem)}$$

Example 5.1.2 Polynomial Data Fitting/Interpolation

- Given:
  - Data: $(x_i, y_i), \quad i = 1, \ldots, m.$
  - Polynomial: we choose degree $d = m - 1$

$$p_d(x) = c_0 + c_1 x + \cdots + c_d x^d.$$

- Goal: determine coefficients c_i $s.t.$

$$p_d(x_i) = y_i, \quad i = 1, \ldots, m.$$

- The system has m equations and has $d + 1$ unknowns. So,

$$d + 1 = m \implies d = m - 1.$$

  Thus, we can form the system

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^{m-1} \end{bmatrix}}_{A\in\mathbb{C}^{m\times m},\ \text{Vandermonde matrix}} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

- In terms of least square: $\|Ac - y\|_2$. But we can choose c to make $\|Ac - y\|_2 = 0$.

Example 5.1.3 Best-Fit Polynomial

- Given:
  - Data: $(x_i, y_i), \quad i = 1, \ldots, m$
  - Polynomial: we choose degree d beforehand, $d \ll m - 1$.

$$p_d(x) = c_0 + c_1 x + \cdots + c_d x^d.$$

- Goal: determine coefficients c_i $s.t.$

$$p_d(x_i) \approx y_i, \quad i = 1, \ldots, m.$$

- Form a linear system:

$$d + 1 = m \implies d = m - 1.$$

Thus, we can form the system

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^d \end{bmatrix}}_{A \in \mathbb{C}^{m \times (d+1)}, \text{ Vandermonde-like matrix}} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

This system is exactly solvable when $y \in \text{range}(A)$. However, this is very unlikely to happen in practice.

- Now, we are really in the case of least squares:

$$\min_c \|Ac - y\|_2^2 = \sum_{i=1}^{m} (p_d(x_i) - y_i)^2.$$

- Solving this least square in `MATLAB`:

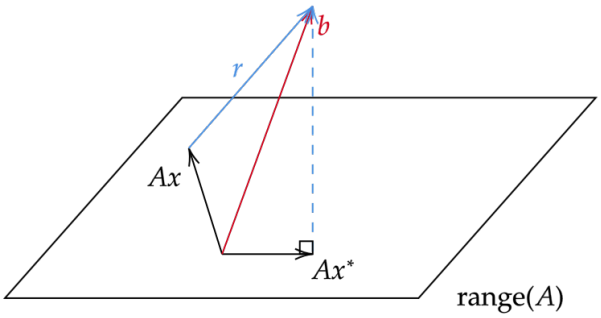
```
c = A \ y;
```

4. How to solve a Least Square Problem: A Geometric and Linear Algebra Story

1. For any x, $Ax \in \text{range}(A)$.

2. $b \in \mathbb{C}^m$ may not be in $\text{range}(A)$.
3. Residual: $r = b - Ax$.
4. When is $\|r\|_2$ as small as possible?

   When Ax is "closest" to b: $Ax \perp r$.



Hence, our goal is to find x such that $Ax \perp (b - Ax)$. That is,

$$(Ax)^*(Ax - b) = 0$$
$$x^*\Big[A^*(Ax - b)\Big] = 0$$

Apparently, $x = 0$ is a trivial solution. It is uninformative, so we ignore it. Suppose $x \neq 0$, we have

$$A^*(Ax - b) = 0$$
$$A^*Ax = A^*b \qquad \text{(Normal Equations)}$$

- A^*A is symmetric positive semi-definite (SPSD)
- (*Assumption*) If A has full column rank, A^*A is SPD $\implies$ unique solution.

5. Ways to Solve Normal Equations

1. GEPP of A^*A: $\quad P(A^*A) = LU$
2. Cholesky Factorization: A^*A is SPD $\implies A^*A = R^*R$.

Problems:

- Conditioning: $\kappa_2(A^*A) = \kappa_2(A)^2$
- A^*A could be dense even if A is sparse.
- More rounding errors

3. SVD of A:

   Suppose $A = U\Sigma V^*$. Then,

$$\begin{aligned} A^*A &= V\Sigma^2 V^* \\ A^*b &= V\Sigma U^* b \\ \implies x &= V\Sigma^{-1}U^*b. \end{aligned}$$

   SVD is a great and stable option, but expensive.

$$\text{Operation counts: } \sim 2mn^2 + 11n^3$$

4. QR Factorization: workhorse

$$A = QR,$$

   where Q has orthogonal columns, and R is upper triangular invertible matrix.

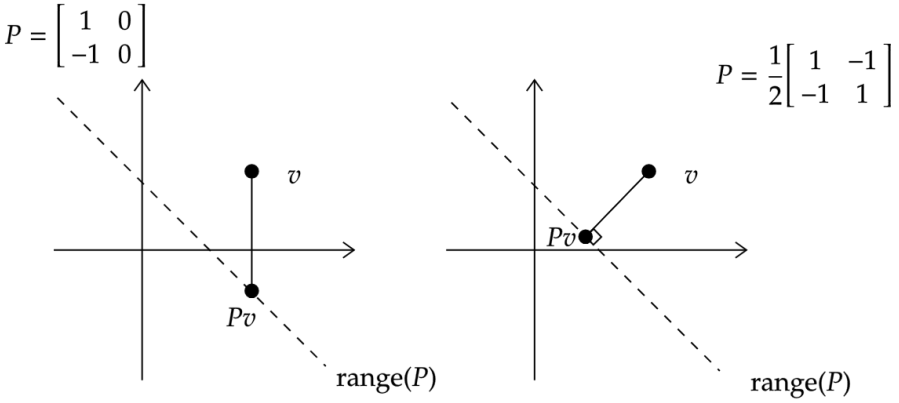
$$\begin{aligned} A^*A &= R^*Q^*QR = R^*R \\ A^*b &= R^*Q^*b \\ \implies \cancel{R^*}Rx &= \cancel{R^*}Q^*b \\ x &= R^{-1}Q^*b \end{aligned}$$

$$\text{Operation counts: } \sim 2mn^2 - \frac{2}{3}n^3$$

5.2 QR Factorization: Gram-Schmidt Orthogonalization

Definition 5.2.1 (Orthogonal Projector).

- P is a *projector* if it is square and $P^2 = P$ (idempotent).
- $I - P$ is a *complementary projector*: $Pv \in \text{range}(P)$ and $(I-P)v \in \text{null}(P)$.
- P is an *orthogonal projector* if $P^* = P$. A non-orthogonal projector is called an *oblique projector*.



Remark 5.1 *P is not necessarily an orthogonal matrix.*

Goal: Find $\{q_1, \ldots, q_n\}$,orthogonal, that span the same space as $\{a_1, \ldots, a_n\}$, L.I..

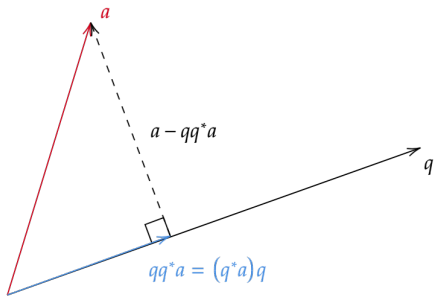
Claim 5.2 q is unit vector $\implies$ qq^* is an orthogonal projector.

Proof 1.

- $(qq^*)^2 = (q\underbrace{q^*}q)(q^*) = qq^*$
- $(qq^*)^* = (q^*)^*q = qq^*$.

Q.E.D. ■

Figure 1: Decomposing a with respect to q



Claim 5.3 Q has orthogonal columns $\implies$ QQ^* is an orthogonal projector.

Theorem 5.2.4 Gram-Schmidt Procedure

Start with

$$\begin{cases} v_1 = a_1 \\ q_1 = \dfrac{v_1}{\|v_1\|_2} \end{cases} \implies \begin{cases} v_2 = a_2 - q_1 q_1^* a_2 \\ q_2 = \dfrac{v_2}{\|v_2\|_2} \end{cases} \text{ (Link to 5.2)} \implies \begin{cases} v_3 = a_3 - q_1 q_1^* a_3 - q_2 q_2^* a_3 \\ q_3 = \dfrac{v_3}{\|v_3\|_2} \end{cases} \cdots$$

In general, in the j-th step:

$$\begin{cases} v_j = a_j - \displaystyle\sum_{k=1}^{j-1} q_k q_k^* a_j \\ q_j = \dfrac{v_j}{\|v_j\|_2} \end{cases}$$

5. From Gram-Schmidt to QR

$$A = QR$$

$$\begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \|v_1\|_2 & q_1^* a_2 & \cdots & \cdots \\ 0 & \|v_2\|_2 & q_2^* a_3 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \|v_n\|_2 \end{bmatrix}$$

In general,

$$r_{i,j} = \begin{cases} q_i^* a_j & i < j \\ \|v_i\|_2, & i = j \\ 0, & i > j. \end{cases}$$

$$\text{Computation cost: } \sim 2mn^2,$$

where m comes from $q_i^* a_j$ and n^2 comes from the number of columns sum and outer loop.

Algorithm 6: Gram-Schmidt (*Unstable*)

```
begin
    for j = 1 : n do
        v_j = a_j;
        // Inner loop:  re-orthogonalize
        for i = 1 : j − 1 do
            r_{i,j} = q_i^* a_j;
            v_j = v_j − r_{i,j} q_i;
        r_{j,j} = ‖v_j‖_2;
        q_j = v_j / r_{j,j};
```

Algorithm 7: Modified Gram-Schmidt (*Stable*)

```
begin
    for i = 1 : n do
        v_j = a_j;
    for i = 1 : n do
        r_{i,i} = ‖v_i‖_2;
        q_i = v_i / r_{i,i};
        // Orthogonalize on the way
        for j = i + 1 : n do
            r_{i,j} = q_i^* v_j;
            v_j = v_j − r_{i,j} q_i;
```

5.3 QR Factorization: Householder Triangularization

Remark 5.2 *This is what* MATLAB *is doing.*

Example 5.3.1 Householder Triangularization

Consider $A \in \mathbb{C}^{5\times 3}$. The idea is to find 3 Q_i's suc that

$$A = \underbrace{Q_1^* Q_2^* Q_3^*}_{Q} \begin{bmatrix} R \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} \xrightarrow{Q_2} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} \xrightarrow{Q_3} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

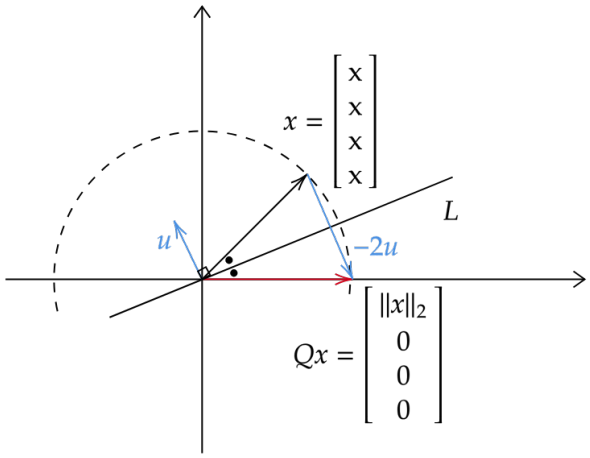
$$A \qquad Q_1A \qquad Q_2Q_1A \qquad Q_3Q_2Q_1A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

2. How to choose Q?

- We want:

$$Q \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} * \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

- Main idea: use reflections:



  - How to describe line L in higher dimension?

    If we are given a normal vector u, then $L = \{x \mid u^*x = 0\}$.

  - From the diagram, we get

$$\begin{aligned} Qx &= x - 2(\text{projection of } x \text{ onto } \{u\}) \\ &= x - 2(u^*x)u \\ &= \underbrace{(I - 2uu^*)}_{Q} x \end{aligned}$$

This is the *Householder reflection*. If u is not normal,

$$Q = I - 2\frac{uu^*}{u^*u}.$$

- How to find u?
  - We know: x and $Qx = \begin{bmatrix} \|x\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$
  - A normal vector to line L will be
    $$u = \|x\|_2 e_1 - x.$$
  - In code:
    $$u = \text{sign}(x_1)\|x\|_2 e_1 - x,$$
    where
    $$\text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0. \end{cases}$$
  - We choose the sign for stability and avoid catastrophic cancellation.

3. Back to Triangularization

$$Q_1 = I - 2\frac{u_1u_1^*}{u_1^*u_2}; \qquad Q_2 = \begin{bmatrix} 1 & \\ & I - 2\frac{u_2u_2^*}{u_2^*u_2} \end{bmatrix}; \qquad Q_3 = \begin{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \\ & I - 2\frac{u_3u_3^*}{u_3^*u_3} \end{bmatrix}$$

Algorithm 8: Householder Triangularization

Input: Matrix $A \in \mathbb{C}^{m\times n}$

```
begin
    for k = 1 : n do
        x = A(k : m, k);
        u_k = sign(x_1)‖x‖_2 e_1 − x;
        u_k = u_k / ‖u_k‖_2;
        A(k : m, k : n) = A(k : m, k : n) − 2u_k ( u_k^* A(k : m, k : n) )  // matrix-vector product is
            more efficient than matrix-matrix product.  So, we do the inner multiplication
            first and not the outer product.
```

Output: Upper triangular matrix $R \in \mathbb{C}^{m\times n}$

4. What about Q?

We only need to store u_k's in practice. If we want to solve $\min \|Ax - b\|_2$ using $A = QR$, we aim to solve $Rx = Q^*b$. To find Qx, we can do something similar to Algorithm 9. So, we can store Q implicitly.

Algorithm 9: Compute Q^*b from Householder Triangularization

```
begin
    for k = 1 : n do
        b(k : m) = b(k : m) − 2u_k(u_k^* b(k : m))
```

Output: Q^*b

$$\text{Computation cost to form } R\text{: } \sim 2mn^2 + \frac{2}{3}n^3.$$

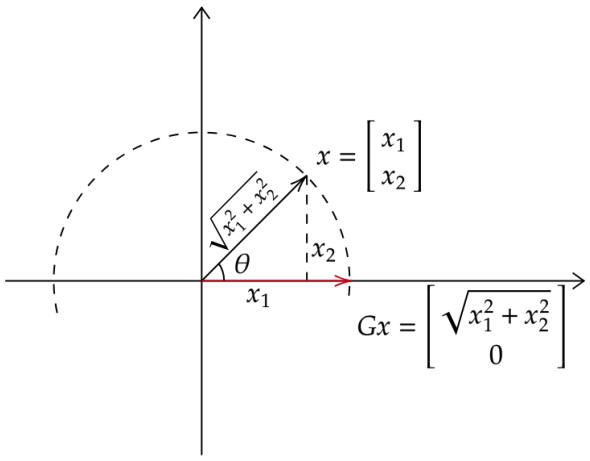
5.4 QR Factorization: Givens Rotations

Example 5.4.1 Givens Rotations Step #1

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \xrightarrow{G_{12}} \begin{bmatrix} x & x & x \\ 0 & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \xrightarrow{G_{13}} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ x & x & x \end{bmatrix} \xrightarrow{G_{14}} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix}$$

$$A \qquad G_{12}A \qquad G_{13}G_{12}A \qquad \underbrace{G_{14}G_{13}G_{12}}_{G} A = GA$$

2. How do we find G?



Consider the 2×2 rotation matrix: if we want to rotate a vector clockwise by θ:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} C & -S \\ S & C \end{bmatrix},$$

where

$$C = \cos(\theta) = \frac{x_1}{\sqrt{x_1^2 + x_2^2}} \quad \text{and} \quad S = \sin(\theta) = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}.$$

Further, $G = \begin{bmatrix} C & S \\ -S & C \end{bmatrix}$.

Proof 1.

$$\begin{aligned} Gx &= \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} Cx_1 + Sx_2 \\ -Sx_1 + Cx_2 \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{bmatrix} \end{aligned}$$

Q.E.D. ■

3.

In general,

$$G_{12} = \begin{bmatrix} C & S & & & \\ -S & C & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}, \quad G_{13} = \begin{bmatrix} C & S & & & & \\ & 1 & & & & \\ -S & & C & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}.$$

$$G_{ij} = \begin{bmatrix} I & & & & \\ & C & & S & \\ & & I & & \\ & -S & & C & \\ & & & & I \end{bmatrix},$$

where the first C appears on the i-th row and i-th column, S appears on the i-th row and j-th column, $-S$ is on the j-th row and i-column, and the second C is on the j-th row and j-th column.

4. Comparison of Householder and Givens

- Householder is more stable and cheaper for dense matrices.
- Givens is cheaper is each step and has its benefits.

5.5 Rank Deficient Least Square

Set-Up: $A \in \mathbb{C}^{m\times n}$ with $m > n$. $\quad \operatorname{rank}(A) = r < n$.

1. Approaches to Solve the System

- QR with column pivotting:

$$AP = QR = Q \begin{bmatrix} \overset{r}{\overset{\searrow}{}} & \overset{n-r}{\square} \\ 0 & 0 \end{bmatrix} \begin{matrix} r \\ n-r \end{matrix}$$

We can further write

$$AP = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

where R_{11} is a non-singular matrix with $\operatorname{rank}(R_{11}) = r$.

– Find column of A with largest 2-norm and swap with first column. Perform QR step:

$$Q_1^* A P_1 = \begin{bmatrix} \alpha_1 & \mathrm{x} & \cdots & \mathrm{x} \\ 0 & & & \\ \vdots & & \boxed{A(2:m, 2:m)} & \\ 0 & & & \end{bmatrix}$$

– Repeat on submatrix:

$$Q_{k-1}^* \cdots Q_2^* Q_1^* A P_1 P_2 \cdots P_{k-1}$$

– Stop when largest column of A has zero norm

– To solve least square problems:

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| \underbrace{AP}\, P^* x - b \right\|_2^2 \\ &= \|QRP^* x - b\|_2^2 \\ &= \left\| R \underbrace{P^* x} - Q^* b \right\|_2^2 \\ &= \left\| \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} - \begin{bmatrix} c \\ d \end{bmatrix} \right\|_2^2 \\ &= \|R_{11} y + R_{12} z - c\|_2^2 + \|d\|_2^2, \end{aligned}$$

where z is arbitrary, and $\|d\|_2^2$ does not depend on x. Hence, we will have ∞-many solutions. Often, $z = 0$ gives the minimum norm solution.

2. SVD Approach (Pseudoinverse Approach)

$$
\begin{aligned}
\|Ax-b\|_2^2 &= \|U\Sigma v^* x - b\|_2^2 \\
&= \left\| \Sigma \underbrace{V^* x}_{y} - U^* b \right\|_2^2 \\
&= \left\| \Sigma y - \underbrace{U^* b}_{z} \right\|_2^2 \\
&= \| \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} y_r \\ \\ y' \end{bmatrix} - \begin{bmatrix} z_r \\ \\ z' \end{bmatrix} \|_2^2 \qquad\qquad y_r, z_r \in \mathbb{C}^r,\ y', z' \in \mathbb{C}^{n-r} \\
&= \| \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}}_{\text{Invertible } \Sigma_r} y_r - \underbrace{z_r}_{U(:,1:r)b} \|_2^2 + \| \underbrace{z'}_{U(:,r+1:n)b} \|_2^2
\end{aligned}
$$

Note that z' does not depend on x or y. Further, $y_r = \Sigma_r^{-1} z_r \implies y'$'s entries are free (and we usually set them to 0)

$$
\begin{aligned}
V(:,1:r)^* x &= \Sigma_r^{-1} U(:,1:r)^* b \\
x &= \underbrace{V(:,1:r)\Sigma_r^{-1} U(:,1:r)^* b}_{\text{pseudoinverse of truncated SVD to rank } r}
\end{aligned}
$$

More generally, $x = A^\dagger b = A^+ b$, where $A^\dagger$ or A^+ are pseudoinverse of A, defined by

$$A^\dagger = V \Sigma^\dagger U^*,$$

where

$$\Sigma^\dagger = \begin{bmatrix} 1/\sigma_1 & & & & & \\ & \ddots & & & & \\ & & 1/\sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

5.6 Perturbation Theory of Least Squares

1. Stability

$$\text{Normal Equation + Cholesky} < \text{QR (Householder)} < \text{SVD}$$

Theorem 5.6.2

Suppose $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ and $\operatorname{rank}(A) = n$. Denote

$$\text{True solution:} \quad x_{\text{LS}} = \arg\min_x \|Ax - b\|_2^2, \qquad A^*Ax_{\text{LS}} = A^*b$$

$$\text{Computed solution:} \quad \widetilde{x}_{\text{LS}} = \arg\min_x \|(A+\delta A)x - (b+\delta b)\|_2^2$$

$$\text{Residual:} \quad r_{\text{LS}} = Ax_{\text{LS}} - b$$

Assume

$$\boxed{\varepsilon = \max\left\{\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}\right\} < \frac{1}{\kappa_2(A)}}$$

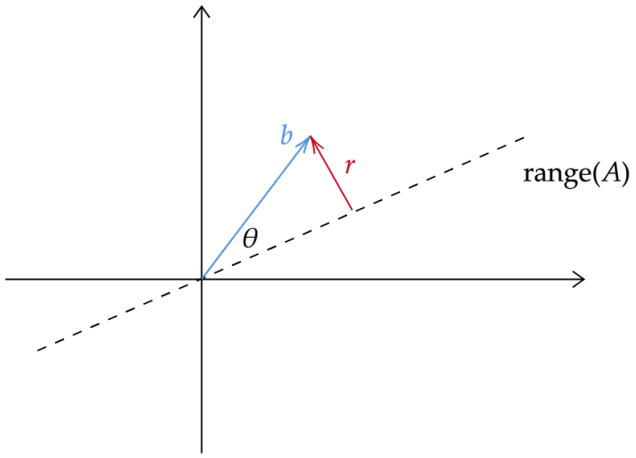
Then,

$$\frac{\|\widetilde{x}_{\text{LS}} - x_{\text{LS}}\|_2}{\|x_{\text{LS}}\|_2} \leq \varepsilon \underbrace{\boxed{\left(\frac{2\kappa_2(A)}{\cos(\theta)} + \tan(\theta)\kappa_2^2(A)\right)}}_{=\kappa_{\text{LS}},\ \text{condition \# of least square}} + \mathcal{O}(\varepsilon^2)$$

$$= \varepsilon\kappa_{\text{LS}} + \mathcal{O}(\varepsilon^2),$$

where $\sin(\theta) = \dfrac{\|r\|_2}{\|b\|_2}$.

Remark 5.3 (Geometric Interpretation of θ) *θ is the angle between b and $\operatorname{range}(A)$.*



Proof 1. (Intuition)
Case I θ is small, $\theta \approx 0$.

b is almost in $\text{range}(A)$
$\implies$ r is small
$\implies$ almost solve a linear system, so the error should mostly depend on $\kappa_2(A)$ and ε

$$\kappa_{\text{LS}} = \frac{2\kappa_2(A)}{\cos(A) \approx 1} + (\tan(\theta) \approx 0)\kappa_2^2(A)$$
$$\approx 2\kappa_2(A)$$

Case II θ is not small nor close to $\frac{\pi}{2}$.

r is moderate in size
$\implies$ condition number could be a bigger problem

$$\kappa_{\text{LS}} = \frac{2\kappa_2(A)}{\cos(A)} + \tan(\theta) \underbrace{\kappa_2^2(A)}_{\text{non-negligible}}.$$

Case III $\theta \approx \frac{\pi}{2}$.

b is almost perpendicular to $\text{range}(A)$
$\implies$ true solution $x_{\text{LS}} \approx 0$
$\implies$ κ_{LS} explodes: $\kappa_{\text{LS}} = \infty$.

$$\kappa_{\text{LS}} = \frac{2\kappa_2(A)}{\cos(A) \approx 0} + \tan(\theta)\kappa_2^2(A)$$

Q.E.D. ■

Remark 5.4 *In the κ_{LS} term,*

- $\kappa_2(A)$ *indicates "can we solve a linear system."*
- $\cos(\theta)$ *indicates "are we completely unable to solve / do we get completely different solutions."*
- $\tan(\theta)\kappa_2^2(A)$ *indicates "do we need to project."*

6 Eigenvalues and Eigenvectors

6.1 Eigendecomposition

Definition 6.1.1 (Eigenvalues & Eigenvectors). $A \in \mathbb{C}^{m\times m}$, a nonzero vector $x \in \mathbb{C}^m$ is an *eigenvector* of A and $\lambda \in \mathbb{C}$ is its corresponding *eigenvalue* if $Ax = \lambda x$.

Definition 6.1.2 (Spectrum of A). The *spectrum of A* is the set of all eigenvalues of A, denoted $\Lambda(A)$.

Definition 6.1.3 (Eigenspace). *Eigenspace* of A, E_λ, is a subspace of $\mathbb{C}^m$, where the action of A mimics scalar multiplication:

$$E_\lambda = \{x \in \mathbb{C}^m \mid Ax = \lambda x\}.$$

Theorem 6.1.4 Eigendecomposition

$$A = X\Lambda X^{-1} \quad \text{or} \quad AX = X\Lambda,$$

where

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix} \quad \text{and} \quad \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix},$$

where x_i's are eigenvectors and λ_i's are eigenvalues.

Remark 6.1 *Not all matrices have an eigendecomposition.*

Example 6.1.5 Matrix without an Eigendecomposition

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

6. Computing Eigendecomposition by Hand

$$\begin{aligned} & Ax = \lambda x \\ \iff & \underbrace{(A - \lambda I)x = 0}_{\text{null space problem}} \quad \text{or} \quad (\lambda I - A)x = 0 \\ \text{Step \#2} \iff & \text{Find a basis for } \operatorname{null}(A - \lambda I). \\ \iff & \text{Eigenvectors are non-zero, so } \operatorname{null}(A - \lambda I) \text{ must be non-trivial} \\ \iff & A - \lambda I \text{ must be singular} \\ \text{Step \#1} \iff & \text{Choose } \lambda \; s.t. \; \det(A - \lambda I) = 0 \end{aligned}$$

Definition 6.1.7 (Characteristic Polynomial/"Eigenpolynomial"). The *characteristic polynomial* of $A \in \mathbb{C}^{m\times m}$ is denoted as

$$p_A(z) = \det(zI - A).$$

Theorem 6.1.8
λ is an eigenvalue of $A \iff p_A(\lambda) = 0$.

Example 6.1.9 Example 6.1.5 Revisit
$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \implies p_A(z) = \det\left(\begin{bmatrix} z-1 & 1 \\ 0 & z-1 \end{bmatrix}\right) = (z-1)^2.$
Therefore, the only eigenvalue is $\lambda = 1$.

6.2 Algebraic and Geometric Multipllicity

Theorem 6.2.1 Fundamental Theorem of Algebra
If $p_A(z)$ is a degree-m polynomial, then

$$p_A(z) = (z-\lambda_1)(z-\lambda_2)\cdots(z-\lambda_m).$$

Definition 6.2.2 (Algebraic Multiplicity of an Eigenvalue). The *algebraic multiplicity of an eigenvalue* is the multiplicity of roots of p_A.

Example 6.2.3 Example 6.1.9 Revisit
$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \implies p_A(z) = (z-1)^2$. Hence, $\lambda = 1$ with algebraic multiplicity of 2.

Remark 6.2 *For simple eigenvalue,* $\text{alg. mult.} = 1$.

Definition 6.2.4 (Geometric Multiplicity of λ).

$$\text{geo. mult.} = \dim(E_\lambda) = \dim(\text{null}(\lambda I - A).$$

Example 6.2.5 Examples of geo. mult.

- $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. For $\lambda = 1$,

$$E_1 = \text{null}(I - A) = \text{null}\left(\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}\right) = \text{span}\left\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\}$$

So, $\text{geo. mult.}(\lambda) = \dim(E_1) = 1$.

- $A = \begin{bmatrix} 1 & & & & & & \\ & 2 & 1 & & & & \\ & & 2 & 1 & & & \\ & & & 3 & & & \\ & & & & 4 & & \\ & & & & & 4 & \\ & & & & & & 4 \end{bmatrix}$. Then, $p_A(z) = (z-1)(z-2)^2(z-3)(z-4)^3$. So, we have

λ_i	alg. mult.	geo. mult.
1	1	1
2	2	1
3	1	1
4	3	3

Definition 6.2.6 (Defective Eigenvalues and Matrices). λ is *defective* if its $\text{alg. mult.} > \text{geo. mult.}$. A matrix with at least one defective eigenvalue is called *defective*.

Example 6.2.7 Example of Defective Matrix

- $A = \begin{bmatrix} 2 & & \\ & 2 & \\ & & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{bmatrix}$.
  A and B have the same characteristic polynomial, but B has only one L.I. eigenvector.
- $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ is defective. $p_A(z) = (z-1)^2 \implies \text{alg. mult.} = 2$. By Example 6.2.5, we have $\text{geo. mult.}(\lambda) = 1$. Therefore, $\lambda = 1$ is defective. This implies that we only have one "eigendirection" when multiplying by A:

$$A^3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = A^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = A \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

Definition 6.2.8 (Similarity Transformations/Similar). If X is non-singular, the map $A \mapsto X^{-1}AX$ is called a *similarity transformation*. Two matrices A, B are *similar* if $\exists$ non-signular X $s.t.$ $B = X^{-1}AX$.

Theorem 6.2.9 Properties of Similarity Transformations
A and $X^{-1}AX$ have the same

- characteristic polynomial
- eigenvalues
- alg. mult. and geo. mult.
- *(and more...)*

Proof 1. We will show the characteristic polynomials are the same.

$$\begin{aligned} p_{X^{-1}AX}(z) &= \det\big(zI - X^{-1}AX\big) \\ &= \det\big(zX^{-1}X - X^{-1}AX\big) \\ &= \det\big(X^{-1}(zI - A)X\big) \\ &= \cancel{\det\big(X^{-1}\big)}\det(zI - A)\cancel{\det(X)} \qquad \left[\det\big(X^{-1}\big) = \frac{1}{\det(X)}\right] \\ &= \det(zI - A) \\ &= p_A(z). \end{aligned}$$

Then, we have eigenvalues and alg. mult. are the same. For geo. mult., if E_λ is an eigenspace of A, then

$$X^{-1}E_\lambda = \big\{X^{-1}y \mid y \in E_\lambda\big\}$$

is the eigenspace of $X^{-1}AX$. Suppose $y \in E_\lambda$. Then, $Ay = \lambda y$. Let $z = X^{-1}y$. Then,

$$\begin{aligned} \boxed{X^{-1}AX}z &= X^{-1}A\underbrace{X(\cancel{X^{-1}}}y) \\ &= X^{-1}\underbrace{Ay} \\ &= X^{-1}(\lambda y) \\ &= \lambda(\underbrace{X^{-1}y}) \\ &= \boxed{\lambda}z. \end{aligned}$$

So, $z \in X^{-1}E_\lambda$, and thus $X^{-1}E_\lambda$ is an eigenspace of $X^{-1}AX$. Then,

$$\dim(X^{-1}E_\lambda) = \dim(E_\lambda) \implies \text{same } \text{geo. mult.}$$

Q.E.D. ■

Theorem 6.2.10
In general, $\text{alg. mult.} \geq \text{geo. mult.}$

Proof 2. Let n be geo. mult. of λ for A. Let $\{v_1, \dots, v_n\}$ be orthonormal basis of E_λ,

$$\widehat{V} = \begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix} \in \mathbb{C}^{m \times n}.$$

We can extend $\widehat{V}$ to an unitary matrix

$$V = \begin{bmatrix} \widehat{V} & V_\perp \end{bmatrix} \in \mathbb{C}^{m \times m},$$

where $V_\perp$ is computed by Gram-Schmidt. Then,

$$\begin{aligned} B &= \underbrace{V^* A V}_{\text{similarity transformation}} \\ &= \begin{bmatrix} \widehat{V}^* \\ V_\perp^* \end{bmatrix} A \begin{bmatrix} \widehat{V} & V_\perp \end{bmatrix} \\ &= \begin{bmatrix} \widehat{V}^* \\ V_\perp^* \end{bmatrix} \begin{bmatrix} \lambda \widehat{V} & A V_\perp \end{bmatrix} \\ &= \begin{bmatrix} \lambda I_n & C \\ 0 & D \end{bmatrix} \end{aligned}$$

If we can derive alg. mult. from B, we can compare it with n as similar matrices share them.

The characteristic polynomial of B is

$$\begin{aligned} p_B(z) &= \det(zI - B) \\ &= \det(zI - \lambda I) \det(zaI - D) \\ &= (z - \lambda)^n \det(zI - D) \end{aligned}$$

Then, the alg. mult. of λ is at least n. Q.E.D. ■

Definition 6.2.11 (Diagonalibility). A is nondefective $\iff$ A is similar to a diagonal matrix. In such case, we call A *diagonalizable* and $A = X^{-1} \Lambda X$.

Theorem 6.2.12

$$\det(A) = \prod_{j=1}^{m} \lambda_j \quad \text{and} \quad \operatorname{tr}(A) = \sum_{j=1}^{m} \lambda_j$$

Definition 6.2.13 (Unitarily Diagonalizable). A is unitarily diagonalizable if $\exists$ unitary Q $s.t.$

$$A = Q \Lambda Q^*.$$

Theorem 6.2.14
A Hermitian matrix is unitarily diagonalizable.

Theorem 6.2.15
A matrix is unitarily diagonalizable $\iff$ it is normal ($A^*A = AA^*$)

Example 6.2.16 Eigenvalues can be Complex even when A is Real-Valued

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -\mathrm{i} & \mathrm{i} \end{bmatrix} \begin{bmatrix} e^{\mathrm{i}\theta} & \\ & e^{-\mathrm{i}\theta} \end{bmatrix} \begin{bmatrix} X^{-1} \end{bmatrix},$$

where $e^{\mathrm{i}\theta} = \cos\theta + \mathrm{i}\sin\theta$.

6.3 Jordan Canonical Form

Theorem 6.3.1
For any matrix $A \in \mathbb{C}^{m\times m}$, $\exists$ non-singular X $s.t.$

$$X^{-1}AX = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_k \end{bmatrix},$$

where $J_i = \begin{bmatrix} \lambda_1 & 1 & & & \\ & \lambda_i & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{m_i\times m_i}$ is called a *Jordan block*, and $m_1+m_2+\cdots+m_k = m$.

Example 6.3.2 Each Jordan Block Corresponds to a Single Eigenvector

$A = \begin{bmatrix} 1 & 1 & & & & & \\ & 1 & & & & & \\ & & 2 & & & & \\ & & & 3 & 1 & & \\ & & & & 3 & 1 & \\ & & & & & 3 & \end{bmatrix}$. Then, $J_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \implies x_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

Meanwhile, we have

$$J_2 = \begin{bmatrix} 2 \end{bmatrix} \implies x_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad \text{and} \quad J_3 = \begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix} \implies x_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Remark 6.3 *It is not good numerically to compute the Jordan Form.*

Theorem 6.3.3 Shur Triangularization (*Numerically Friendly Factorization*)
For any $A \in \mathbb{C}^{m\times m}$, $A = QTQ^*$, unitarily similar to an upper triangular matrix. That is, Q is unitary and T is upper triangular.

Proof 1.

- $m = 1$: trivial to show.
- $m \geq 2$: let x be eigenvector of A with $\|x\| = 1$. Write $U = \begin{bmatrix} x & U_\perp \end{bmatrix}$ unitary. Compute

$$U^*AU = \begin{bmatrix} \lambda & B \\ 0 & C \end{bmatrix}.$$

By induction, assume $C \in \mathbb{C}^{(m-1)\times(m-1)}$ can be written as VTV^*, then let

$$Q = U \begin{bmatrix} 1 & 0 \\ 0 & V \end{bmatrix}.$$

We have

$$Q^*AQ = \begin{bmatrix} \lambda & BV \\ 0 & T \end{bmatrix}, \quad \text{upper triangular.}$$

Q.E.D. ■

6.4 General Eigenvalue Algorithms

1. Find roots of characteristic polynomial

(−) : ill-conditioned to find polynomial roots (even if eigenvalue problem is well-conditioned).

(−) : expensive (computing determinant).

2. Companion matrix

$$p(z) = z^m + a_{m-1}z^{m-1} + \cdots + a_1 z + a_0 \qquad \text{(characteristic polynomial)}$$

- Build the companion matrix:

$$A = \begin{bmatrix} -z & & & & -a_0 \\ 1 & -z & & & -a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & 1 & -z & -a_{m-1} \end{bmatrix} = B - zI.$$

$$p_A(z) = (-1)^m p(z) \implies \text{roots of } p_A(z) = \text{roots of } p(z).$$

Roots of $p(z)$ are eigenvalues of

$$B = \begin{bmatrix} 0 & & & & -a_0 \\ 1 & 0 & & & -a_1 \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & \vdots \\ & & & 1 & -a_{m-1} \end{bmatrix}.$$

- However, it is impossible to solve.

> **Theorem 6.4.3 Galoi's Impossibility Theorem**
> No formula to determine roots of polynomial from its coefficients (such as quadratic formula) for polynomials of degree 5 or more.

- An eigenvalue solver must be *iterative*. We don't have deterministic method such as LU factorization. We have to approximate the eigenvalues in some way.

6.4.1 Power Iteration

Definition 6.4.4 (Rayleigh Quotient). If x is eigenvector of A, then *Rayleigh quotient* is

$$\rho(x, A) = \frac{x^*Ax}{x^*x} = \frac{x^*(\lambda x)}{x^*x} = \frac{\lambda x^*x}{x^*x} = \lambda.$$

Remark 6.4 *In Algorithm* (10)*, as we have normalized v_{k+1}, we don't need to divide by x^*x.*

- Does this converge? And to what?

  Assume A has dominant eigenvalue: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_m| \geq 0$.

Algorithm 10: Power Iteration

Input: $A \in \mathbb{C}^{m\times m}$, $x_0 \in \mathbb{C}^m$

```
begin
    while not converged do
        v_{k+1} = A x_k // dominant cost O(n^2)
        x_{k+1} = v_{k+1} / ||v_{k+1}||_2 // normalization
        λ_{k+1} = x*_{k+1} A x_{k+1}
```

Output: (x, λ), dominant eigenpair

- When doe we stop iterating? How do we measure convergence?

  User defined "small:"

  1. $\|x_{k+1} - x_k\|_2$ is small
  2. $|\lambda_{k+1} - \lambda_k|$ is small

- Drawbacks:
  - $|\lambda_1| > |\lambda_2|$ is not always true, and how do we know if it is true?
  - Slow
  - Starting guess is important: if x_0 doesn't contain any part in the eigenspace E_{λ_1}, then we don't converge to λ_1.
  - Only get one eigenpair.

6.4.2 Shifted Power Method (Inverse Iteration)

- Consider $A - \sigma I$, where $\sigma \in \mathbb{C}$. The eigenvalues of $A - \sigma I$ are $\lambda - \sigma$, where $\lambda' s$ are eigenvalues of A. Also, eigenvalues of $(A - \sigma I)^{-1}$ is $\dfrac{1}{\lambda - \sigma}$.
  - Magnify eigenvalues of A near σ
  - Shifting σ, we make different eigenvalues of A dominant.

Theorem 6.4.5 Convergence of Inverse Interation
Suppose we try to capture (x_J, λ_J), the eigenpair closest to σ. If $|\lambda_J - \sigma| < |\lambda_k - \sigma| \leq |\lambda_j - \sigma|$ for all $J \neq j$, then inverse iteration converges to λ_J with convergence rate $\dfrac{|\lambda_J - \sigma|}{|\lambda_k - \sigma|}$, and

$$\|x_k - x_J\|_2 \leq c\left|\frac{\lambda_J - \sigma}{\lambda_k - \sigma}\right|^k$$

Algorithm 11: Inverse Iteration

Input: $\sigma \in \mathbb{C}$, $x_0 \in \mathbb{C}^m$, $A \in \mathbb{C}^{m\times m}$

begin
  while *not converged* **do**
    Solve $(A-\sigma I)v = x_k$. Denote the output as v_{k+1};
    $x_{k+1} = \dfrac{v_{k+1}}{\|v_{k+1}\|_2}$;
    $\lambda_{k+1} = x_{k+1}^* A x_{k+1}$;

Output: (x, λ), eigenpair closes to σ

Proof 1. Suppose $A = X\Lambda X^{-1}$. Let $v_0 = xy = y_1x_1 + y_2x_2 + \cdots + y_mx_m$. Let $|\lambda_k - \sigma|$ be smallest. That is, $|\lambda_k - \sigma| \le |\lambda_j - \sigma|$ for $j \neq k$. Assume $y_k \neq 0$:

$$v_i = (A-\sigma I)^{-i}v = X(\Lambda-\sigma I)^{-i}X^{-1}v_0 = X\begin{bmatrix}(\lambda_1-\sigma)^{-i}y_1\\(\lambda_2-\sigma)^{-i}y_2\\\vdots\\(\lambda_m-\sigma)^{-i}y_m\end{bmatrix} = \underbrace{y_k(\lambda_k-\sigma)^{-i}}_{\substack{\text{largest in magnitude}\\ \to 1}} X \underbrace{\begin{bmatrix}\frac{y_1}{y_k}\left(\frac{\lambda_k-\sigma}{\lambda_1-\sigma}\right)^i\\\vdots\\\frac{y_m}{y_k}\left(\frac{\lambda_k-\sigma}{\lambda_m-\sigma}\right)^i\end{bmatrix}}_{\substack{\le 1 \text{ and } =1 \text{ for } k\text{-th entry}\\ \to e_k \text{ as } k\to\infty}}$$

$$\xrightarrow{i\to\infty} = y_k \cdot 1 X e_k = y_k \underbrace{x_k}_{k\text{-th eigenvector}}$$

So, the inverse iteration converge to eigenvector corresponding to λ_k at the convergence rate

$$\left|\frac{\lambda_k-\sigma}{\lambda_j-\sigma}\right| \begin{cases}\text{close to } 1\text{, slow convergence}\\ \text{close to } 0\text{, fast convergence}\end{cases}$$

Q.E.D. ■

6.4.3 Variation of Inverse Iteration: Rayleigh Quotient Iteration (RQI)

Algorithm 12: Rayleigh Quotient Iteration (RQI)

Input: A, real and symmetric; x_0

begin
  $\rho_0 = \rho(x_0, A) = \dfrac{x_0^* A x_0}{x_0^* x_0}$;
  while *not converged* **do**
    $y_k = (A-\rho_{k-1}I)^{-1}x_{k-1}$ // Potential stopping criterion: $|Ax_k - \rho_k x_k| <$ tolerance
    $x_k = \dfrac{y_k}{\|y_k\|_2}$;
    $\rho_k = \rho(x_k, A)$;

Theorem 6.4.6 Convergence of RQI
RQI converges to eigenpair for all but a set of measure zero starting vectors x_0. When it converges, the convergence is ultimately cubic. That is, if λ_J is the eigenvalue of A and x_0 is sufficiently close to q_J (true eigenvector), then

$$\|x_{k+1} - (\pm q_J)\| = \mathcal{O}\Big(\|x_k - (\pm q_J)\|^3\Big)$$

and

$$|\lambda_{k+1} - \lambda_J| = \mathcal{O}\Big(|\lambda_k - \lambda_J|^3\Big)$$

as $k \to \infty$.

Theorem 6.4.7 Another Perspective of Cubic Convergence
RQI is locally cubically convergence: # of correct digits triples once error is small enough and eigenvalue is simple

Proof 2. Assume $A = \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)$ is diagonal. Then, e_i's are the eigenvectors. WLOG, assume x_k is converging to e_1. Then,

$$x_k = e_1 + d_k, \quad \text{where } \|d_k\|_2 =: \varepsilon \ll 1$$

To prove cubic convergence, we need to show

$$x_{k+1} = e_1 + d_{k+1}, \quad \text{where } \|d_{k+1}\|_2 = \mathcal{O}(\varepsilon^3)$$

- As we normalize x_k is algorithm,

$$\begin{aligned} 1 &= x_k^* x_k \\ &= (e_1 + d_k)^*(e_1 + d_k) \\ &= e_1^* e_1 + 2e_1^* d_k + d_k^* d_k \\ &= 1 + 2d_{k_1} + \varepsilon^2, \quad \text{where } d_{k_1} \text{ is the first entry of } d_k. \end{aligned}$$

  So,

$$d_{k_1} = -\frac{\varepsilon^2}{2}.$$

- Consider the Rayleigh quotient:

$$\begin{aligned} \rho_k = x_k^* A x_k &= (e_1 + d_k)^* \Lambda (e_1 + d_k) \\ &= \underbrace{e_1^* \Lambda e_1}_{\lambda_1} + \underbrace{2e_1^* \Lambda d_k + d_k^* \Lambda d_k}_{-\eta := -\lambda_1 \varepsilon^2 + d_k^* \Lambda d_k} \end{aligned}$$

- By triangular inequality and other properties, we have

$$|\eta| \le |\lambda_1|\varepsilon^2 + \|\Lambda\|_2 \underbrace{\|d_k\|_2^2}_{\varepsilon^2} \le 2\|\Lambda\|_2\varepsilon^2.$$

Back to the algorithm: in the next iteration: $\begin{bmatrix} \lambda_1-\rho_k & & & \\ & \lambda_2-\rho_k & & \\ & & \ddots & \\ & & & \lambda_m\rho_k \end{bmatrix}^{-1}$, we have

$$\begin{aligned}
y_{k+1} &= (\Lambda-\rho_k I)^{-1}x_k \\
&= \begin{bmatrix} \frac{x_{k_1}}{\lambda_1-\rho_k} & \frac{x_{k_2}}{\lambda_2-\rho_k} & \cdots & \frac{x_{k_m}}{\lambda_m-\rho_k} \end{bmatrix}^* && x_k = e_1 + d_k \\
&= \begin{bmatrix} \frac{1+d_{k_1}}{\lambda_1-\rho_k} & \frac{d_{k_2}}{\lambda_2-\rho_k} & \cdots & \frac{d_{k_m}}{\lambda_m-\rho_k} \end{bmatrix}^* && d_{k_1} = -\frac{\varepsilon^2}{2} \\
&= \begin{bmatrix} \frac{1-\frac{\varepsilon^2}{2}}{\lambda_1-\rho_k} & \frac{d_{k_2}}{\lambda_2-\rho_k} & \cdots & \frac{d_{k_m}}{\lambda_m-\rho_k} \end{bmatrix}^* && \rho_k - \lambda_1 - \eta \\
&= \begin{bmatrix} \frac{1-\frac{\varepsilon^2}{2}}{\eta} & \frac{d_{k_2}}{\lambda_2-\lambda_1+\eta} & \cdots & \frac{d_{k_m}}{\lambda_m-\lambda_1+\eta} \end{bmatrix}^* \\
&= \frac{1-\frac{\varepsilon^2}{2}}{\eta}\begin{bmatrix} 1 & \frac{d_{k_2}}{\left(1-\frac{\varepsilon^2}{2}\right)(\lambda_2-\lambda_1+\eta)} & \cdots & \frac{d_{k_m}}{\left(1-\frac{\varepsilon^2}{2}\right)(\lambda_m-\lambda_1+\eta)} \end{bmatrix}^* \\
&= \frac{1-\frac{\varepsilon^2}{2}}{\eta}\left(e_1 + \widehat{d}_{k+1}\right)
\end{aligned}$$

[WTS: $\left\|\widehat{d}_{k+1}\right\|_2 = \mathcal{O}(\varepsilon^3)$] Define $\mathrm{gap}(\cdot,\cdot)$ as follows: Suppose A has eigenvalues $\lambda_1 \ge \cdots \lambda_m$, then

$$\mathrm{gap}(i, A) = \min_{j\ne i} |\lambda_j - \lambda_i|.$$

So, we have

$$\begin{aligned}
\left\|\widehat{d}_{k+1}\right\|_2 &\le \frac{\overbrace{\|d_k\|_2}^{\text{1 more entry}} \quad \overbrace{|\eta|}^{\text{shared}}}{\underbrace{\left(1-\frac{\varepsilon^2}{2}\right)}_{\text{shared}}\underbrace{(\mathrm{gap}(1,\Lambda)-|\eta|)}_{\text{small denominator}}} \quad \text{a little more than the numerators:} \\
&\le \frac{2\|\Lambda\|_2\varepsilon^3}{\left(1-\frac{\varepsilon^2}{2}\right)(\mathrm{gap}(1,\Lambda)-|\eta|)} && \|d_k\|_2 = \varepsilon, \quad |\eta| \le 2\|\Lambda\|_2\varepsilon^2.
\end{aligned}$$

Note, we have

$$|\lambda_j - \lambda_i + \eta| \leq |\lambda_j - \lambda_i| + |\eta|,$$

and by definition of $\text{gap}(1, \Lambda)$, we know that if the eigenvalues are close to each other, the fraction is larger. When ε is small:

$$\left\|\widehat{d}_{k+1}\right\|_2 \leq \frac{\overbrace{2\|\Lambda\|_2}^{\text{constant}}\,\varepsilon^3}{\underbrace{\left(1 - \frac{\varepsilon^2}{2}\right)}_{\approx 1}\underbrace{(\text{gap}(1,\Lambda) - |\eta|)}_{\approx \text{gap}(1,\Lambda)}} \sim \mathcal{O}(\varepsilon^3)$$

Finally,

$$x_{k+1} = e_1 + d_{k+1} = \frac{e_1 + \widehat{d}_{k+1}}{\left\|e_1 + \widehat{d}_{k+1}\right\|_2} \quad \text{normalized version}$$

One can form a similar argument to show $\|d_{k+1}\|_2 = \mathcal{O}(\varepsilon^3)$.

If A is real and symmetric, A is unitarily diagonalizable: $A = Q\Lambda Q^*$. One can show:

$$\rho(x_k, A) = \rho(\widehat{x}_k, \Lambda), \quad \text{where } \widehat{x}_k = Q^* x_k.$$

So, if RQI converges cubically for diagonal matrices, it also converges cubically for a general real and symmetric matrix. Q.E.D. ■

6.4.4 Orthogonal Iteration/Simultaneous Iteration/Subspace Iteration

Algorithm 13: Orthogonal Iteration

Input: A, $Z_0 \in \mathbb{C}^{m\times p}$ with unitary columns

```
1 begin
2     while not converged do
3         Y_k = A Z_{k-1};
4         [Z_k, R_k] = QR(Y_k) // reduced QR factorization of Y_k
```

- If $p = 1$: power iteration; If $p > 1$: find dominant p eigenvectors all at once.
- Why the algorithm work?

  Key assumption: $\underbrace{|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_p|}_{p \text{ dominant eigenvalues}} > |\lambda_{p+1}| \geq \cdots \geq |\lambda_m| > 0$

  Proof 3. Note that

$$\begin{aligned} \text{range}(Z_k) = \text{range}(Y_k) &= \text{range}(AZ_{k-1}) && \text{by QR factorization} \\ &= \text{range}(A^k Z_0) && AZ_{k-1} = A^k Z_0 \\ &= \text{range}(X\Lambda^k X^{-1} Z_0) && \text{Diagonalization} \end{aligned}$$

Then, we have

$$X\Lambda^k Z^{-1} Z_0 = \underbrace{\lambda_p^k X}_{\text{factoring}} \underbrace{\begin{bmatrix} \boxed{\begin{matrix} \left(\frac{\lambda_1}{\lambda_p}\right)^k & & \\ & \ddots & \\ & & 1 \end{matrix}} & \\ & \boxed{\begin{matrix} \left(\frac{\lambda_{p+1}}{\lambda_p}\right)^k & & \\ & \ddots & \\ & & \left(\frac{\lambda_m}{\lambda_p}\right)^l \end{matrix}} \end{bmatrix}}_{L_p^k} X^{-1} Z_0,$$

where the blue boxed parts are fractions with absolute value ≥ 1, which stick around as $k \to \infty$. On the other hand, green boxed parts are fractions with absolute value < 1, which $\to 0$ as $k \to \infty$. So,

$$L_p^k \to \begin{bmatrix} X & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix}.$$

$$\implies X \underbrace{\Lambda^k X^{-1} Z_0}_{W^k} = \lambda_p^k \left(X(:, 1:p) W^k(1:p, :) + \underbrace{X(:, p+1:m) W^k(p+1:m, :)}_{\to 0 \text{ as } k \to \infty} \right)$$

The idea behind this step is partition:

$$\begin{bmatrix} X_p & X_{m-p} \end{bmatrix} \begin{bmatrix} W_p \\ W_{m-p} \end{bmatrix}, \quad \text{where } W_{m-p} \to 0 \text{ as } k \to \infty.$$

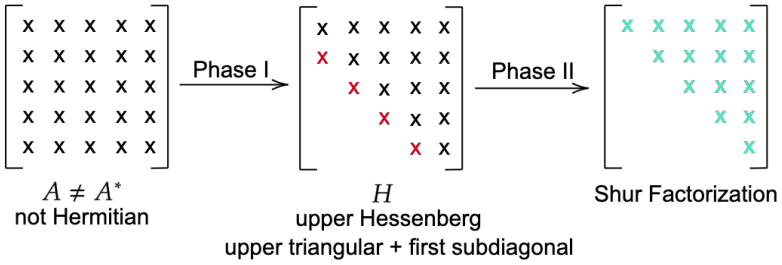
In the long run ($k \to \infty$):

$$\begin{aligned} \text{range}(X\Lambda^k X^{-1} Z_0) &= \text{range}\left(X(:, 1:p) W^k(1:p, :) \right) \quad \text{Assumption: } W^k(1:p, :) \text{ is full column rank} \\ &= \text{range}\left(X(:, 1:p) \right). \end{aligned}$$

Q.E.D. ■

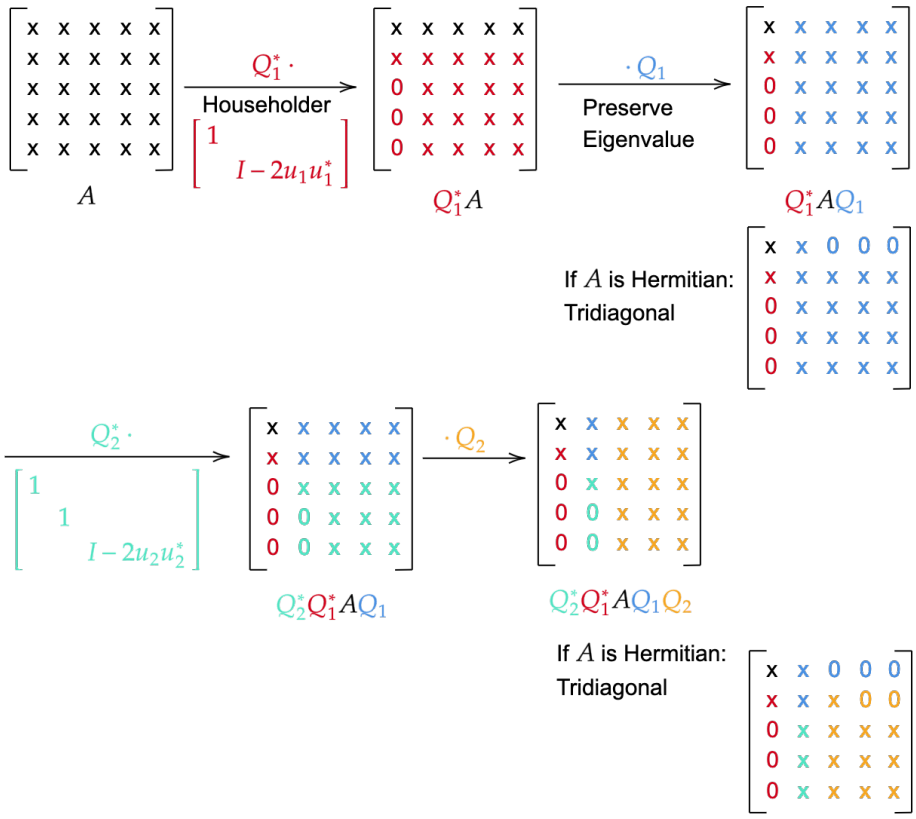
6.4.5 Two Phases Algorithm to Produce Shur Factorization

- Recall: Shur Factorization and Diagonalization: $A = QTQ^*$, where Q is unitary and T is upper triangular. We have some key observations:
  - Eigenvalues of A and T are the same (due to properties of similarity transformation).
  - The eigenvalues of T are its diagonal entries.
- Overview of two Phases Algorithms:



Remark 6.5 *If A is Hermitian, phase I produces a tridiagonal matrix (symmetric), and phase II will be more efficient.*

- Overview of Phase I:



Algorithm 14: Phase I to Produce Upper Hessenberg Matrix

Input: $A \in \mathbb{C}^{m\times m}$

1 **begin**
2 **for** $k = 1 : m-2$ **do**
3 $x = A(k+1:m, k)$;
4 $v_k = \text{sign}(x_1)\|x\|_2 e_1 + x$ `// from Householder`
5 $v_k = \dfrac{v_k}{\|v_k\|_2}$;
6 $A(k+1:m, k:m) = A(k+1:m, k:m) - 2v_k(v_k^* A(k+1:m, k:m))$ `//` $Q_k^* \cdot$
7 $A(1:m, k+1:m) = A(1:m, k+1:m) - 2(A(1:m, k+1:m)v_k)v_k^*$ `//` $\cdot A_k$

- Computational cost:

$$\mathcal{O}(m \cdot m^2) = \mathcal{O}(m^3) \sim \frac{10}{3}m^3,$$

where the first m comes from the loop, and the second m^2 comes from matrix-vector multiplication (Lines 6 and 7).

Remark 6.6 *With Hermitian matrix, if we go from tridiagonal to diagonalization, we may "unzero" some terms. So, we need to be careful.*

6.4.6 QR Algorithm

Algorithm 15: QR Algorithm (Real-Valued)

Input: A

1 **begin**
2 $A^{(0)} = A$;
3 **while** *not converged* **do**
 `/* Potential stopping criteria:` $\|A^{(k)} - A^{(k-1)}\| <$ `tol or` $Q^*AQ \to T$ `*/`
4 $A^{(k-1)} = Q^{(k)}R^{(k)}$ `// QR factorization`
5 $A^{(k)} = R^{(k)}Q^{(k)}$;

Output: $A = QTQ^*$, Shur complement

- Why this algorithm works?

Proof 4. Note that

$$A^{(0)} = A = Q^{(1)}R^{(1)}.$$

Then,

$$A^{(1)} = R^{(1)}Q^{(1)} = \left(Q^{(1)}\right)^* \underbrace{Q^{(1)}R^{(1)}}\, Q^{(1)} = \left\{Q^{(1)}\right\}^* A^{(0)}Q^{(1)} \quad \text{by similarity transformation}$$

$$A^{(2)} = \left(Q^{(2)}\right)^* A^{(1)}Q^{(2)} = \left(Q^{(2)}\right)^* \left(Q^{(1)}\right)^* A^{(0)}Q^{(1)}Q^{(2)} \quad \text{by similarity transformation}$$

Q.E.D. ■

- Pro: converges cubically
- Con: "bad idea:" Shur form in one step

Theorem 6.4.8 Relationship Between QR Algorithm and Orthogonal Iteration
Suppose A_k is from the QR Algorithm (Algorithm (15)) and Z_k, Z_0 are from Orthogonal Iteration (Algorithm (13)). Then,

$$A_k = Z_k^* A Z_k, \quad \text{if } Z_0 = I,$$

and $A_k \to$ Shur form (upper triangular) if eigenvalues all have different magnitude.

Proof 5. (by Induction)
Base Case: Suppose $k = 0$. Then, $A_0 = A$ and $Z_0 = I$. So,

$$A_0 = Z_0^* A Z_0 = IAI = A.$$

Inductive Steps: Assume $A_k = Z_k^* A Z_k$ is true. [WTS: $A_{k+1} = Z_{k+1}^* A Z_{k+1}$.]
From orthogonal iteration, we have

$$AZ_k = \underbrace{Z_{k+1}}_{\text{orthogonal}} \overbrace{R_k}^{\text{upper triangular}}.$$

So,

$$Z_k^* \underbrace{AZ_k}_{} = \underbrace{Z_k^* Z_{k+1}}_{=Q} \underbrace{R_{k+1}}_{=R} \quad \text{product of orthogonal matrices are still orthogonal.}$$

By assumption, we have

$$A_k = Z_k^* A Z_k = QR.$$

Then, by uniqueness of QR factorization (though up to some small changes), we have the following from QR algorithm as Step #1:

$$A_k = QR.$$

Now, consider

$$\begin{aligned} Z_{k+1}^* A Z_{k+1} = Z_{k+1}^* A \underbrace{(Z_k Z_k^*)}_{=I} Z_k = Z_{k+1} \underbrace{AZ_k}_{} \overbrace{Q}^{=Z_k^* Z_k} \qquad & [Y_{k+1} = Z_{k+1} R_{k+1}] \\ = \underbrace{Z_{k+1}^* Z_{k+1}}_{=I} R_{k+1} Q \\ = \underbrace{R_{k+1}}_{=R} Q = RQ. \end{aligned}$$

Therefore, $A_{k+1} = RQ$. Q.E.D. ■

Corollary 6.9 : Because orthogonal iteration converges, and QR algorithm is essentially the same as orthogonal iteration. They both converge in the same way.

6.4.7 Practical QR Iteration: Single-Shift QR Iteration

Example 6.4.10 Motivation

Perform unshifted QR (Algorithm (15)) on

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

- $A_0 = A$.
- factorL: $A_0 = Q_1 R_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- multiply: $A_1 = R_1 Q_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = A \implies$ We do not converge.

Algorithm 16: Single-Shift QR Iteration

Input: $A \in \mathbb{R}^{m \times m}$

begin
  $A_0 = Q_0 A Q_0^*$ `// upper Hessenberg reduction`
  while *not converged* **do**
    Choose shift σ_k close to eigenvalue of A `// How to choose` σ_k`? We will discuss this later.`
    Factor $A_k - \sigma_k I = Q_{k+1} R_{k+1}$;
    Multiply $A_{k+1} = R_{k+1} Q_{k+1} + \sigma_k I$;

Claim 6.11 A_k, A_{k+1} are orthogonal similar.

Proof 6.

$$\begin{aligned} A_{k+1} &= R_{k+1} Q_{k+1} + \sigma_k I \\ &= \underbrace{Q_{k+1}^* Q_{k+1}}(R_{k+1} Q_{k+1} + \sigma_k I) \\ &= Q_{k+1}^* \underbrace{Q_{k+1} R_{k+1}} Q_{k+1} + \sigma_k Q_{k+1}^* Q_{k+1} \\ &= Q_{k+1}^* (Q_{k+1} R_{k+1} + \sigma_k I) Q_{k+1} \\ &= Q_{k+1}^* (A_k - \cancel{\sigma_k I} + \cancel{\sigma_k I}) Q_{k+1} \\ &= Q_{k+1}^* A_k Q_{k+1}. \end{aligned}$$

Q.E.D. ■

Theorem 6.4.12 Choose the shift σ_k: Rayleigh Quotient
We will set

$$\sigma_k = \frac{x_k^* A_k x_k}{x_k^* x_k}$$

with specific choice of x_k based on Q_k. Our choice here is to pick the last column of Q_k, i.e.,

$$\sigma_k = Q_k(:,m)^* A_k Q_k(:,m) \implies \sigma_k = A_k(m,m).$$

Algorithm 17: Single-Shift QR Iteration with σ_k Choice Described in Theorem 6.4.12

Input: $A \in \mathbb{R}^{m\times m}$

```
begin
    H_0 = Q_0 A Q_0^* // upper Hessenberg
    while not converged do
        Choose shift σ_k = H_{k-1}(m,m);
        Factor H_k − σ_k I = Q_{k+1} R_{k+1};
        Multiply H_{k+1} = R_{k+1} Q_{k+1} + σ_k I;
```

Theorem 6.4.13 Convergence of Single-Shift QR Algorithm
If we order eigenvalues of A $s.t.$

$$|\lambda_1 - \sigma| \geq |\lambda_2 - \sigma| \geq \cdots \geq |\lambda_m - \sigma|, \quad \text{for fixed } \sigma,$$

then the p-th sub-diagonal entry of H_k converges to 0 with rate

$$\left|\frac{\lambda_{p+1} - \sigma}{\lambda_p - \sigma}\right|^k$$

Remark 6.7 *Each time, we cut the sub-diagonal entry by this rate. Hence, we want* $\left|\frac{\lambda_{p+1} - \sigma}{\lambda_p - \sigma}\right|^k$ *to be small (that is, smaller than* 1 *and close to* 0*).*

Definition 6.4.14 (Unreduced/Irreducible Matrix). A matrix is *unreduced* (or *irreducible*) if and only if its off-diagonal entries are nonzero.

Theorem 6.4.15
Let σ be an eigenvalue of H, upper Hensenberg (unreduced/irreducible). In each iteration,

$$H - \sigma I = QR$$
$$\widetilde{H} = RQ + \sigma I$$

Then, $\widetilde{H}(m,m-1) = 0$ and $\widetilde{H}(m,m) = \sigma$.

Proof 7.

- H is unreduced $\implies$ first $m-1$ columns are L.I..
- $H - \sigma I = QR \implies R(i,i) \neq 0$ for $i = 1, \ldots, m-1$.
- If $H - \sigma I$ is singular, then $R(1,1) \cdots R(m,m) = 0$.
- By ②, it must be $R(m,m) = 0 \implies$ last row of $\widetilde{H} = \sigma e_m^*$.

$$RQ = \begin{bmatrix} * & * & * & * \\ & * & * & * \\ & & * & * \\ & & & 0 \end{bmatrix} Q = \begin{bmatrix} & & * & \\ & & & \\ 0 & \cdots & & 0 \end{bmatrix} \implies RQ + \sigma I = \begin{bmatrix} & & * & \\ & & & \\ 0 & \cdots & & \sigma \end{bmatrix}$$

Q.E.D. ■

Theorem 6.4.16 Implicit Q Theorem
If H is upper Hessenberg, unreduced, and $H = Q^* A Q$, then columns 2 to m of Q are determined uniquely (up to sings), by the first column of Q.

Proof 8. ("Chase the Bulge") Suppose

$$H = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}$$

- $Q_1^* = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \implies Q_1^* H = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}$.

$$\implies H_1 = Q_1^* H Q_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ \diamondsuit & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}$$

This is a similarity transformation.

This $\diamondsuit$ is the "bulge." Because we alter the first two columns using Householder reflection, we introduce unexpected nonzero entry. Hence, we will restore the upper Hessenberg form by operating the second and the third rows.

- Restore upper Hessenberg:

$$Q_2^* = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \Longrightarrow Q_2^* H = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}$$

$$H_1 = Q_2^* H Q_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ & \diamondsuit & * & * & * \\ & & & * & * \end{bmatrix}$$

- Continuing this process, we have

$$Q_3^* = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix} \quad \text{and} \quad Q_4^* = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix}.$$

So, we have

$$H_4 = \underbrace{Q_4^* Q_3^* Q_2^* Q_1^*}_{Q^*} H \underbrace{Q_1 Q_2 Q_3 Q_4}_{Q} = Q^* H Q.$$

Then, the general Q will be given by

$$Q = \begin{bmatrix} c_1 & * & * & * & * \\ s_1 & * & * & * & * \\ 0 & s_2 & * & * & * \\ 0 & & s_3 & * & * \\ 0 & & & s_4 & * \end{bmatrix},$$

where the first column is the normalized first column of H (which is also the first vector of the QR factorization). Meanwhile, s_2, s_3, and s_4 are defined based on c_1, s_1. Q.E.D. ■

Remark 6.8 *We can also have Doubly-Shifted QR Iteration.*

6.5 Symmetric Eigenvalue Problem

Assumption: $A \in \mathbb{R}^{m \times m}$, symmetric.

Remark 6.9 *Ussing Householder reflection, we can convert and A to a triangonal matrix T. The algorithms will be based on T.*

6.5.1 Divide-and-Conquer Algorithm

$$
\begin{aligned}
T &= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & a_2 & b_2 & & & & & & & \\
 & b_2 & \ddots & \ddots & & & & & & \\
 & & \ddots & a_{k-1} & b_{k-1} & & & & & \\
 & & & b_{k-1} & a_k & b_k & & & & \\
\hline
 & & & & b_k & a_{k+1} & b_{k+1} & & & \\
 & & & & & b_{k+1} & \ddots & \ddots & & \\
 & & & & & & \ddots & \ddots & b_{m-1} & \\
 & & & & & & & b_{m-1} & a_m &
\end{array}\right] \\
&= \left[\begin{array}{ccc|ccc}
\ddots & & & & & \\
 & a_k - b_k & & & & \\
\hline
 & & & a_k - b_k & & \\
 & & & & \ddots &
\end{array}\right] + \left[\begin{array}{cc|cc}
 & & b_k & b_k \\
 & & b_k & b_k \\
\hline
 & & & \\
 & & &
\end{array}\right] \\
&= \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + b_k v v^*,
\end{aligned}
$$

where $v = e_k + e_{k+1} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$.

Assume we have eigendecomposition of T_1 and T_2; that is,

$$T_1 = Q_1 \Lambda_1 Q_1^* \quad \text{and} \quad T_2 = Q_2 \Lambda_2 Q_2^*.$$

Then,

$$
\begin{aligned}
T = \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + b_k v v^* &= \left[\begin{array}{c|c} Q_1 \Lambda_1 Q_1^* & \\ \hline & Q_2 \Lambda_2 Q_2^* \end{array}\right] + b_k v v^* \\
&= \left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] \left(\left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right] + b_k u u^* \right) \left[\begin{array}{c|c} Q_1^* & \\ \hline & Q_2^* \end{array}\right],
\end{aligned}
$$

where $u = \left[\begin{array}{c|c} Q_1^* & \\ \hline & Q_2^* \end{array}\right] v$.

Rewrite

$$\left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right] + b_k uu^* = \underbrace{D}_{\text{diagonal}} + \overbrace{\rho uu^*}^{\text{rank 1 update}}$$

Goal: Find eigenvalues of $D + \rho uu^*$.

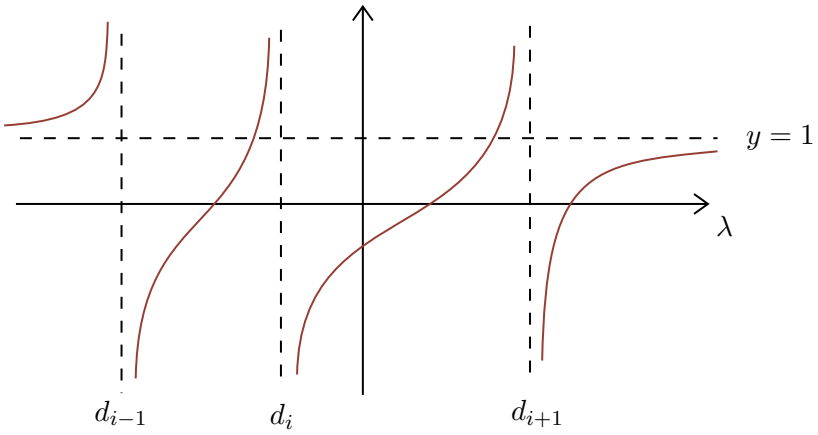
Assume $D - \lambda I$ is nonsingular, then characteristic polynomial of $D + \rho uu^*$ is

$$\begin{aligned} p_{D+\rho uu^*}(\lambda) &= \det(D + \rho uu^* - \lambda I) \\ &= \det\left(\underbrace{(D - \lambda I)}_{\neq 0 \text{ by non-singularity}} \quad \overbrace{\left(I + \rho(D - \lambda I)^{-1} uu^*\right)}^{\text{identity+rank 1}} \right) \end{aligned}$$

From Homework 1, we have

$$\begin{aligned} \det\left(I + \rho(D - \lambda I)^{-1} uu^*\right) &= 1 + \rho u^*(D - \lambda I)^{-1} u \\ &= 1 + \rho \sum_{i=1}^{m} \frac{u_i^2}{d_i - \lambda} \\ &=: f(\lambda) \qquad \text{(Secular Equation)} \end{aligned}$$

Goal (updated): Find the roos of $f(\lambda)$.



To find $f(\lambda)$: use Newton's method on each interval (d_i, d_{i+1}).

Computational cost:

- Find one root: $\mathcal{O}(m)$.
- Find all roots: $\mathcal{O}(m^2)$.

To write this algorithm, we use recursion. Define

$$[\mathtt{Q}, \mathtt{\Lambda}] = \mathtt{dcEig}(\mathtt{T}, \underbrace{Q, \Lambda}_{\text{optional}}), \qquad T = Q\Lambda Q^*$$

- 1×1 Case:

$$T = \begin{bmatrix} a_1 \end{bmatrix}$$

$$[Q,\ \Lambda] = [1,\ a_1] = \texttt{dcEig}(T)$$

- 2×2 Case:

$$T = \begin{bmatrix} a_1 & b_1 \\ b_1 & a_2 \end{bmatrix} = \left[\begin{array}{c|c} \overbrace{a_1 - b_1}^{1\times 1} & 0 \\ \hline 0 & \underbrace{a_2 - b_1}_{1\times 1} \end{array}\right] + b_1 \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix}}_{vv^*} = \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_1 vv^*.$$

So, we can solve two 1×1 cases:

$$[Q_1,\ \Lambda_1] = [1,\ a_1 - b_1] = \texttt{dcEig}(T_1) \quad \text{and} \quad [Q_2,\ \Lambda_2] = [1,\ a_2 - b_1] = \texttt{dcEig}(T_2).$$

- Build $D + \rho uu^*$:

$$D = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix},\ u = \underbrace{\begin{bmatrix} Q_1^* & \\ & Q_2^* \end{bmatrix}}_{=I} v = v$$

$$\implies D + \rho uu^* = \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} + b_1 uu^*.$$

- Find eigenvalues of $D + \rho uu^*$ by finding roots of the secular equation.
- Find eigenvectors of $D + \rho uu^*$: $(D - \lambda I)^{-1}u$ are eigenvectors, where λ is an eigenvalue of $D + \rho uu^*$. *However, this method is not numerically stable.*
- Orthogonalize eigenvectors and form Q'
- $Q = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} Q'$
- Return Q, Λ (eigenvalues of $D + \rho uu^*$).

- 4×4 Case:

$$T = \left[\begin{array}{cc|cc} a_1 & b_2 & & \\ b_1 & a_2 & b_2 & \\ \hline & b_2 & a_3 & b_3 \\ & & b_3 & a_4 \end{array}\right] = \left[\begin{array}{c|c} \overbrace{T_1}^{2\times 2} & \\ \hline & \underbrace{T_2}_{2\times 2} \end{array}\right] + b_2 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}.$$

So, we can solve two 2×2 cases:

$$[Q_1,\ \Lambda_1] = \texttt{dcEig}(T_1) \quad \text{and} \quad [Q_2,\ \Lambda_2] = \texttt{dcEig}(T_2).$$

6.5.2 Bisection Method (Finding a subset of Eigenvalues)

Assumption: T is tridiagonal, symmetric, irreducible (off-diagonal entires are non-zero)

Remark 6.10 *If we have a reducible matrix,*

$$\left[\begin{array}{ccc|cc} * & * & & & \\ & * & * & & \\ & * & * & 0 & \\ \hline & & 0 & * & * \\ & & & * & * \end{array}\right] = \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right],$$

we can perform the algorithm in each submatrix.

Definition 6.5.1 (Principal Minor). Given an $m \times m$ matrix T, the upper left $k \times k$ sub-matrix is called the k*-th principal minor* and is denoted by $T^{(k)}$.

Example 6.5.2 Principal Minor

Consider a 3×3 matrix

$$T = \begin{bmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & & & \\ & & & \ddots & & \\ & & & & \ddots & b_{m-1} \\ & & & & b_{m-1} & a_m \end{bmatrix}.$$

Then the principal minors are

$$T^{(1)} = \begin{bmatrix} a_1 \end{bmatrix}, \quad T^{(2)} = \begin{bmatrix} a_1 & b_1 \\ b_1 & a_2 \end{bmatrix}, \quad T^{(3)} = \begin{bmatrix} a_1 & b_1 & \\ b_1 & a_2 & b_2 \\ & b_2 & a_3 \end{bmatrix}.$$

Proposition 6.3 : Eigenvalues of T and $T^{(k)}$ are distinct. For $T^{(k)}$,

$$\lambda_1^{(k)} < \lambda_2^{(k)} < \cdots < \lambda_k^{(k)}.$$

One can also show, eigenvalue strictly interlace. i.e.,

$$\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k1)}.$$

Definition 6.5.4 (Sturm Sequence (As a Consequence of Proposition)). The *Sturm sequence* is defined as

$$1,\ \det\left(T^{(1)}\right),\ \det\left(T^{(2)}\right), \ldots, \det\left(T^{(m)}\right).$$

Note that

$$\#\text{ of negative eigenvalues} = \#\text{ of sign changes in Sturm sequence.}$$

So, given $T - xI$, we can determine # of eigenvalues in any interval $[a, b)$ by

$$(\#\text{ of negative eigenvalues of } T - bI) - (\#\text{ of negative eigenvalues of } T - aI).$$

6.6 Eigenvalue Perturbation Theory

Question: If we perturb A, how much do eigenpairs change?

Theorem 6.6.1 Gershgorin Circles/Disks
Suppose $A = D + F$, where $A = \operatorname{diag}(d_1, \ldots, d_m)$ and F has 0's on tis diagonal. Then,

$$\Lambda(A) \subseteq \bigcup_{i=1}^{m} D_i,$$

where

$$D_i = \left\{ z \in \mathbb{C} \middle| |z - d_i| \leq \sum_{j=1}^{m} |f_{ij}| \right\} = ().$$

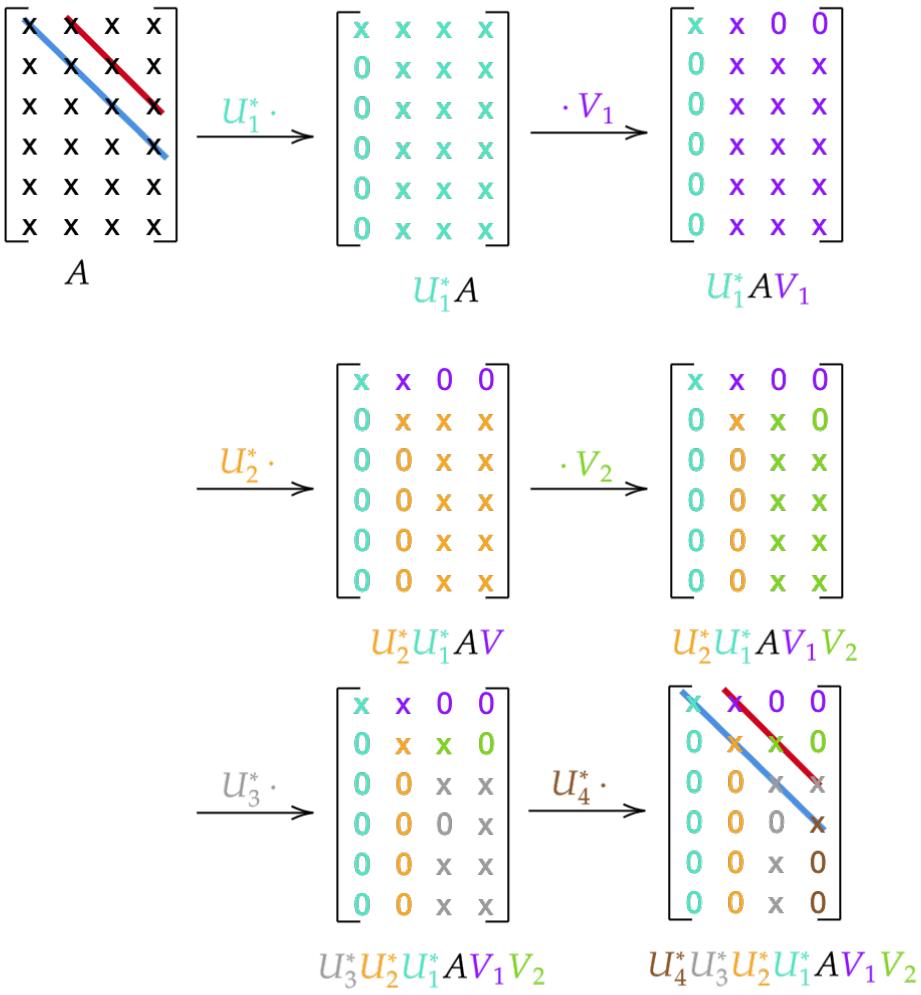
7 Computing SVD

Generally speaking, computing SVD is a **two phases algorithm framework**.

Given $A \in \mathbb{C}^{m\times n}$ with $m > n$.

- Reduction to bidiagonal form B. (Deterministic; $\mathcal{O}(mn^2)$)
- SVD of B. (Iterative; infinite number of steps in theory; in practice, $\mathcal{O}(n \log|\log \varepsilon_{\texttt{mach}}|)$)

7.1 Phase I: Golub-Kahan (GK) Bidiagonalization



The computation cost

$$\sim 4mn^2 - \frac{4}{3}n^3 \quad \texttt{flops}.$$

Algorithm 18: GK Bidiagonalization

Input: $A \in \mathbb{R}^{m\times n}$ with $m \le n$

begin
  $w = \texttt{randn}(n,1)$ `// create first normalized column of` W `randomly`
  $W(:,1) = W/\|w\|_2$;
  `// main loop`
  for $k = 1 : m$ **do**
    `// update` Q
    $Q(:,k) = AW(:,k)$;
    if $k \ge 1$ **then**
      $Q(:,k) = Q(:,k) - B(k-1,k)Q(:,k-1)$;
    $B(k,k) = \|Q(:,k)\|_2$;
    $Q(:,k) = \dfrac{Q(:,k)}{B(k,k)}$;
    `// update` W
    if $k < m$ **then**
      $W(:,k+1) = A^*Q(:,k) - B(k,k)W(:,k)$;
      $B(k,k+1) = \|W(:,k+1)\|_2$;
      $W(:,k+1) = \dfrac{W(:,k+1)}{B(k,k+1)}$;

Output: $Q \in \mathbb{R}^{m\times m}$ unitary; $B \in \mathbb{R}^{m\times n}$ upper bidiagonal; $W \in \mathbb{R}^{n\times m}$ unitary ($W^*W = I_m$)

- Other (potentially faster) Approach: Lawson-Hanson-Chan (LHC):

$$\underset{\substack{m\times n\\ m>n}}{A} \xrightarrow{\text{QR}} \underset{n\times n}{R} \to B$$

  - LHC is less expensive when $m > \dfrac{5}{3}n$.
  - We can do this with GK. We run LHC for a huge matrix for some iterations, and then run GK on the submatrix.

7.2 Phase II: SVD of Bidiagonal Matrix

- Overview:

$$A \xrightarrow{\text{Phase I}} \underset{U_A^*AV_A}{B} \xrightarrow{\text{Phase II}} U_B\Sigma V_B^*$$

$$\implies A = U_ABV_A^* = U_A(U_B\Sigma V_B^*)V_A^* = (U_AU_B)\Sigma(V_AV_B)^*.$$

- A simple idea: leverage eigendecomposition algorithms.
  - Consider $C = \begin{bmatrix} 0 & B^* \\ B & 0 \end{bmatrix} \in \mathbb{C}^{(2n)\times(2n)}$. Define

$$P = \begin{bmatrix} e_1 & e_{n+1} & e_2 & e_{n+2} & \cdots & e_n & e_{2n} \end{bmatrix}. \qquad \text{(Perfect Shuffle)}$$

Then, $T = P^*CP$ is symmetric and tridiagonal with the following properties:

* all zeros on main diagonal of T (traceless).
* off-diagonal alternate entries in B. That is,

$$B = \begin{bmatrix} a_1 & b_1 & & & \\ & a_2 & b_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & b_{n-1} \\ & & & & a_n \end{bmatrix} \implies T = \begin{bmatrix} 0 & a_1 & & & & \\ a_1 & 0 & b_1 & & & \\ & b_1 & 0 & a_2 & & \\ & & a_2 & 0 & \ddots & \\ & & & \ddots & \ddots & a_n \\ & & & & a_n & 0 \end{bmatrix}$$

Then, we can apply algorithms such as divide-and-conquer or bisection algorithm to calculate the eigenvalues of T.

Permutation matrices are orthogonal. So, $T = P^*CP$ is similar to B.

– If x_i is eigenvector of T, $Tx_i = \alpha_i x_i$, then $\alpha_i = \pm\sigma_i(B)$ and

$$p_{x_i} = \frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix},$$

where v_i and u_i are left and right singular vectors of B.

– Warning: Running divide-and-conquer or QR iteration on C is impractical.

* We only need non-negative eigenvalues of T (might do 2 times more work).
* Small numerical problems with small singular values.

- Idea # 2:

– Consider $T = BB^* \in \mathbb{C}^{n\times n}$, symmetric, tridiagonal. Then,

$$T = \begin{bmatrix} a_1^2 + b_1^2 & a_2 b_1 & & & & & \\ * & a_2^2 + b_2^2 & a_3 b_2 & & & & \\ & * & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & * & a_{n-1}^2 + b_{n-1}^2 & a_n b_{n-1} & \\ & & & & * & a_n^2 & \end{bmatrix}.$$

– Singular values of B are square roots of eigenvalues of T.

However, we only get left singular vectors of B. i.e, $B = U\Sigma V^* \implies T = U\Sigma^* U^*$.

- Idea # 3: $T = B^*B$

– T looks similar as in idea # 2, but slightly different.

– Problem: only get right singular vectors of B.

– Even worse: ill-conditioning. It is numerically unstable to build BB^* or B^*B.

Example 7.2.1

Consider $B = \begin{bmatrix} 1 & 1 \\ 1 & \sqrt{\eta} \end{bmatrix}$, where η is small. Then, $\sigma(B) \approx \left\{ \sqrt{2}, \sqrt{\frac{\eta}{2}} \right\}$. On the other hand,

$$B^*B = \begin{bmatrix} 1 & 1 \\ 1 & 1+\sqrt{\eta} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ will give us wrong singular values.}$$

- Idea # 4: Differential Quotient-Difference Algorithm with Shift (DQDS)

Algorithm 19: A Mathematically Equivalent Algorithm: LR Iteration

Input: any symmetric tridiagonal T_0

```
1 begin
2     while not converged do
3         Choose a shift τ_k^2 smaller than the smallest eigenvalue of T_k;
4         Compute Cholesky factorization T_k − τ_k^2 I = B_k^* B_k;
          // We never want to form B_k^* B_k explicitly
5         Update T_{k+1} = B_k B_k^* + τ_k^2 I;
```

Output: a (tri)diagonal matrix T_k

– One can show: T_k and T_{k+1} are similar.

– Two steps of LR with $\tau_k = 0$ is the same as QR iteration.

– B_k is upper bidiagonal:

$$B_k = \begin{bmatrix} a_1 & b_1 & & & \\ & a_2 & b_2 & & \\ & & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} \\ & & & & a_n \end{bmatrix}, \qquad B_{k+1} \begin{bmatrix} \widehat{a}_1 & \widehat{b}_1 & & & \\ & \widehat{a}_2 & \widehat{b}_2 & & \\ & & \ddots & \ddots & \\ & & & \widehat{a}_{n-1} & \widehat{b}_{n-1} \\ & & & & \widehat{a}_n \end{bmatrix}.$$

and

$$T_{k+1} = B_k B_k^* + \tau_k^2 I \qquad \text{(update at } k\text{-th iteration)}$$

$$T_{k+1} = B_{k+1}^* B_{k+1} + \tau_{k+1}^2 I \qquad \text{(factorization at } (k+1)\text{-th iteration)}$$

So,

$$B_{k+1}^* B_{k+1} + \tau_{k+1}^2 I = B_k B_k^* + \tau_k^2 I.$$

On the diagonal, we have

$$\widehat{a}_j^2 + \widehat{b}_{j-1}^2 + \tau_{k+1}^2 = a_j^2 + b_j^2 + \tau_k^2 \qquad (n \text{ equations})$$

On the sub-diagonal, we have

$$\widehat{a}_j^2\widehat{b}_j^2 = a_{k+1}^2 b_j \qquad (n-1 \text{ equations})$$

Goal: Write $\widehat{a}_j, \widehat{b}_j$ in terms of a_j, b_j.

Remark 7.1 *When solving, we have to assume* $\widehat{b}_0 = b_0 = b_n = \widehat{b}_n = 0$. *As we have, in total,* $2n-1$ *unknowns (*n *for* a_j *and* $n-1$ *for* b_j*) and* $2n-1$ *equations. We can solve formulas for* $\widehat{a}_j$ *and* $\widehat{b}_j$.

8 Iterative Methods

8.1 Introduction

Definition 8.1.1 (Direct Methods). *Direct methods* have explicit procedure with known stopping point. (*For example, Gaussian elimination or QR factorization.*) Typically, it requires $\mathcal{O}(m^3)$ `flops`.

(−) too expensive when matrix size is large.

(−) only need $\mathcal{O}(m^2)$ in storage of A, but need many more `flops`.

Definition 8.1.2 (Iterative Methods).

- Rules: we don't form A. We can only apply A (or A^*) to a vector. That is, we know Ax (and A^*y).
- If A is sparse, compute Ax costs $\mathcal{O}(\nu m)$, where $\nu = \#$ of entries per row, and $\nu \ll m$.

Example 8.1.3

Consider

$$A = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \implies \nu = 3 \implies \sim \mathcal{O}(3m).$$

Table 1: Applications of Iterative Methods

	Solve $Ax = b$	Solve $Ax = \lambda x$
Symmetric $A = A^*$	CG (Conjugate Gradient)	Lanczos
Non-symmetric $A \neq A^*$	GMRES, CGN, BCG	Arnoldi

Example 8.1.4 Solving $Ax = b$ Iteratively: Two Schools of Thoughts (*there are others*)

- Split $A = M - K$, where M is non-singular. *For example, Jacobi, Gauss-Seidel, Successive over-relaxation (SoR)*
- Krylov subspace method. *For example, CG, GMRES, ...*

Definition 8.1.5 (Krylov Subspace). The *Krylov subspace*, $\mathcal{K}_n(A, b)$, is defined by

$$\mathcal{K}_n(A, b) = \operatorname{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\}.$$

Example 8.1.6 How do we use $\mathcal{K}_n(A,b)$ to (approximately) solve $Ax=b$?

- We try to find the "best" solution in Krylov subspace

$$\min_{x\in\mathcal{K}_n(A,b)} \|Ax-b\|_2.$$

That is,

$$\begin{aligned} x &= c_0 b + c_1 Ab + \cdots + c_{n-1}A^{n-1}b \\ &= (c_0 I + c_1 A + \cdots + c_{n-1}A^{n-1})b \\ &= p(A)b. \end{aligned}$$

For example, MINRES for symmetric matrix A, and GMRES for non-symmetric matrix A.

- If A is SPD, we could use

$$\min_{x\in\mathcal{K}_n(A,b)} \|Ax-b\|_{A^{-1}},$$

where $\|r\|_{A^{-1}} = \left(r^* A^{-1} r\right)^{1/2}$.

For example, CG.

8.2 Arnoldi Method

Definition 8.2.1 (Krylov Matrix). Recall the *Krylov subspace* is defined by

$$\mathcal{K}_n(A,b) = \operatorname{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\}.$$

Think power method:

$$\begin{aligned} y_1 &= b && = b \\ y_2 &= Ay_1 && = Ab \\ &\vdots \\ y_n &Ay_{n-1} && A^{n-1}b \end{aligned}$$

Then, the *Krylov matrix*, $K \in \mathbb{R}^{m\times n}$, is defined as

$$K = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}.$$

2. Properties of Krylov Matrix and a Theoretic Idea

- $A\mathcal{K}_{n+1}(A,b) = \operatorname{span}\left\{Ab, A^2b, \ldots, A^nb\right\} \implies A\mathcal{K}(A,b) \subseteq \mathcal{K}_{n+1}(A,b)$.

Proof 1.

$$\begin{aligned} AK &= \begin{bmatrix} Ay_1 & Ay_2 & \cdots & Ay_n \end{bmatrix} \\ &= \begin{bmatrix} y_2 & y_3 & \cdots & A^n y_1 \end{bmatrix}. \end{aligned}$$

Q.E.D. ■

- $AK = KC$, where

$$C = \begin{bmatrix} e_2 & e_3 & \cdots & e_n & -c \end{bmatrix} = \begin{bmatrix} 0 & 0 & & & -c_1 \\ 1 & 0 & & & \vdots \\ 0 & 1 & & & \vdots \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & \vdots & & 0 & -c_{m-1} \\ 0 & 0 & \cdots & 1 & -c_m \end{bmatrix}, \quad \text{upper Hessenberg.}$$

  Further,

$$c = -K^{-1}A^n y.$$

  This may be a good idea: $AK = KC \implies K^{-1}AK = C$.

- Danger:
  - K is likely to be ill-conditioned (hard to invert).
  - Solving for C, we apply A n times, which is slow.

3. A Practical Idea

$$\begin{aligned} \mathcal{K}_n(A, b) &= \operatorname{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\} \\ &= \operatorname{span}\{\underbrace{q_1, q_2, q_3, \ldots, q_n}_{\text{orthogonal basis}}.\} \end{aligned}$$

We find $q_1, \ldots, q_n$ by a Gram-schmidt-ish procedure. Note that $\operatorname{col}(K) = \mathcal{K}_n(A, b)$. Suppose $K = QR$, we have

$$\begin{aligned} K^{-1}AK = (QR)^{-1}A(QR) &= C \\ \implies R^*Q^*AQR &= C \\ Q^*AQ = RCR^{-1} &= H \quad \text{another upper Hessenberg} \end{aligned}$$

4. Arnoldi Algorithm

$$Q^*AQ = H \implies AQ = QH.$$

- Orthonormal basis for $\{b\}$:

$$q_1 = \frac{b}{\|b\|_2}.$$

- Orthonormal basis for $\{b, Ab\}$:

$$\begin{aligned}
q_1 &= \frac{b}{\|b\|_2} && \text{from Gram-Schmidt} \\
Aq_1 &= h_{11}q_1 + h_{21}q_2 && \text{from Krylovian} \\
q_1^* Aq_1 &= h_{11}\cancel{q_1^* q_1}^{1} + h_{21}\cancel{q_1^* q_2}^{0} \\
h_{11} &= q_1^*(Aq_1) && \text{Rayleigh Quotient} \\
\implies v &:= h_{21}q_2 = Aq_1 - h_{11}q_1 \implies v \text{ is parallel to } q_2 \\
h_{21} &= \|v\|_2, \quad q_2 = \frac{v}{h_{21}} = \frac{v}{\|v\|_2}.
\end{aligned}$$

So,

$$Aq_1 = \begin{bmatrix} q_1 & q_2 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix}$$

- Continuing this process, we should get

$$AQ_n = Q_n H_n + \underbrace{w_n e_n^*}_{\substack{\text{left-over} \\ \text{“}Av_n - A^n v\text{”}}} = Q_{n+1} \underbrace{\overline{H}_{(n+1,n)}}_{\text{upper Hessenberg}}$$

Algorithm 20: Arnoldi's Method

Input: unit vector q, linear operator $A : \mathbb{R}^m \to \mathbb{R}^m$

```
begin
    for j = 1, 2, ..., n do
        w_j = A q_j // next column in K_n(A, q_1)
        for i = 1, ..., j do
            // orthogonalization: Gram-Schmidt
            h_ij = q_i^* w_j;
            w_j = w_j − h_ij q_i;
        h_(j+1),j = ||w_j||_2;
        q_(j+1) = w_j / h_(j+1),j;
```

Output: Q_{n+1}, $\overline{H}_n$

- How to use Arnold to Solve $Ax = b$?
  - Let x_0 be an initial guess for solution, and let $r_0 = b - Ax_0$.
  - Build $\mathcal{K}_n(A, r_0) = \text{span}\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{n-1} r_0\right\}$.

– Goal: Find solution $x_n \in x_0 + \mathcal{K}_n(A, r_0)$ $s.t.$

$$b - Ax_n \perp \mathcal{K}_n(A, r_0).$$

Remark 8.1 $x_0 + \mathcal{K}_n(A, r_0)$ *means:* $x_n = x_0 + Q_n C_n$*, where* Q_n *is a* $m \times n$ *matrix whose columns form orthonormal basis of* $\mathcal{K}_n(A, r_0)$*, from Arnoldi's Method.*

– Snapshot of solution:

$$AQ_n = Q_{n+1}\overline{H}_n = Q_n H_n + w_n \cdot e_n^*,$$

where w_n is multiple of q_{n+1}.

By orthogonality of columns of Q_n, we have

$$Q_n^* A Q_n = H_n.$$

– End product: good approximate solution is

$$x_n = x_0 + \underbrace{Q_n\big(H_n^{-1}\beta e_1\big)}_{\in \mathcal{K}_n(A, r_0)}, \quad \beta = \|r_0\|_2$$

– Benefits of using Arnoldi's method:

(+) H_n is $n \times n$, small and upper Henssenberg $\implies$ easy to invert.

(+) βe_1 is a basis vector $\implies$ easy to work with

(+) Punchline: with Krylov methods, we do the work in small spaces $(n \times n)$.

- Arnoldi & Eigenvalues:

– Main idea: estimate eigenvalues of A using H_n at each iteration. Usually, we find extreme eigenvalues first. *We might not get all eigenvalues.*

– Some algebraic intuition:

Recall: $x \in \mathcal{K}_n(A, b) \equiv \operatorname{span}\{b, Ab, \ldots, A^{n-1}b\}$. Then, we have

$$\begin{aligned} x &= c_0 b + c_1 Ab + \cdots + c_{n-1}A^{n-1}b \\ &= \big(c_0 I + c_1 A + \cdots + c_{n-1}A^{n-1}\big)b \\ &= q(A)b, \end{aligned}$$

where $q(\cdot)$ is a polynomial, where $q(z) = c_0 + c_1 z + \cdots + c_{n-1}z^{n-1}$.

– Arnoldi Approximation Problem:

$$\min \|p_n(A)\|_2, \quad s.t.\ p_n \text{ is a degree } n \text{ polynomial with } c_n = 1 \text{ (monic)}$$

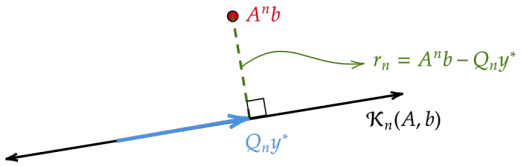
(Arnoldi Approxmation)

That is, $p_n(z) = c_0 + c_1 z + \cdots + c_{n-1} z^{n-1} + 1 \cdot z^n$. By Arnoldi's method, we can have

$$p_n(A)b = A^n b - Q_n y, \quad y \in \mathbb{C}^n,$$

where Q_n is from Arnoldi, with columns being orthonormal basis of $\mathcal{K}_n(A, b)$.
Equivalently, (Arnoldi Approxmation) can be written as a least square problem:

$$\min_{y \in \mathbb{C}^n} \|A^n b - Q_n y\|_2 \qquad \text{(Arnoldi Approximation II)}$$



Theorem 8.2.5
If Arnoldi Iteration does not break down (i.e., $\dim(\mathcal{K}_n(A, b)) = n$), then the characteristic polynomial of H_n minimizes (Arnoldi Approxmation) Problem.

8.3 Generalized Minimal Residual Method (GMRES)

1. Main Idea

Approximate solution to $Ax = b$ via $x_n \in \mathcal{K}_n(A, b)$. We will do this in a least square way:

$$\min_{x \in x_0 \in \mathcal{K}_n(A, r_0)} \|b - Ax\|_2, \qquad \text{(P)}$$

where $r_0 \coloneqq b - Ax_0$.

2. Equivalent Problem Statement: Change of Variables

Define $x = x_0 + z$, (P) becomes an easier constrained problem:

$$\min_{z \in \mathcal{K}_n(A, r_0)} \|b - A(x_0 + z)\|_2 = \|r_0 - Az\|_2 \qquad \text{(P)}$$

3. Rewrite (P) using Arnoldi

If $z \in \mathcal{K}_n(A, r_0)$, using Arnoldi, $\exists\, y \in \mathbb{C}^n \; s.t. \; z = Q_n y$, where Q_n is from Arnoldi. Then, (P) becomes an unconstraineed least square

$$\min_{y \in \mathbb{C}^n} \|r_0 - A \underbrace{Q_n y}\|_2 \qquad \text{(P)}$$

From Arnoldi: $AQ_n = Q_{n+1}\overline{H}_n$, so

$$\min_{y\in\mathbb{C}^n} \left\| r_0 - Q_{n+1}\overline{H}_b y \right\|_2 \tag{P}$$

Recall that we can view Arnoldi as Gram Schmidt on $\mathcal{K}_n(A, r_0) = \text{span}\left\{r_0, Ar_0, \ldots, A^{n-1}r_0\right\}$. Then,

$$Q_{n+1}(:,1) = \frac{r_0}{\|r_0\|_2}.$$

Then, (P) is further reduced to

$$\min_{y\in\mathbb{C}^n} \| \underbrace{Q_{n+1}}_{\text{orthonormal}} (\beta e_1 - \overline{H}_n y)\|_2, \quad \text{where } \beta = \|r_0\|_2. \tag{P}$$

By 2-norm unitary invariance, (P) is equivalent to

$$\min_{y\in\mathbb{C}^n} \|\beta e_1 - \underbrace{\overline{H}_n}_{(n+1)\times n} y\|_2 \tag{GMRES}$$

We can easily solve for y.

Algorithm 21: GMRES

Input: $A \in \mathbb{C}^{m\times m}$, $x_0 \in \mathbb{C}^m$, $b \in \mathbb{C}^n$

```
begin
    r_0 = b − Ax_0;
    [Q_{n+1}, H̄_n] = arnoldi(A, r_0, n);
    Solve y* ∈ arg min_y ‖βe_1 − H̄_n y‖_2;
    Update x* = x_0 + Q_n y*;
```

4. Convergence of GMRES

- Main question: How many steps n do we need to reach desired accuracy of $\frac{\|r_n\|_2}{\|b\|_2}$?
- Observation:
  - $\|r_{n+1}\|_2 \le \|r_n\|_2$, where $r_n = b - Ax_n$ and x_n is returned from `GMRES(n)`
    Intuition: $\mathcal{K}_n(A, r_0) \subseteq \mathcal{K}_{n+1}(A, r_0)$.
  - How many steps until $\|r_n\|_2 = 0$? (in exact arithmetic): m steps (i.e., we exactly solve $Ax = b$).
- Assume A is diagonalizable: $A = X\Lambda X^{-1}$. Then,

$$\|r_n\|_2 \le \underbrace{\kappa_n(X)}_{\substack{\text{how orthogonal}\\ \text{the eigenvectors are}}} \|p(\Lambda)\|_2 \underbrace{\|r_0\|_2}_{\text{initial guess}},$$

  where p is a degree n polynomial with $p(0) = 1$ (i.e., $c_0 = 1$).

Remark 8.2 (Intuition for p_n)

$$\min_{\substack{p:\text{degree } n \\ p(0)=1}} \|p(A)r_0\|_2 = \|(I - Aq_{n-1}(A))b\|_2 = \|b - Aq_{n-1}(A)b\|_2$$

Proof 1. If $x \in \mathcal{K}_n(A,b)$, then

$$\|b - Ax\|_2 = \|p(A)r_0\|_2 = \left\|Xp(\Lambda)X^{-1}r_0\right\|_2 \leq \underbrace{\|X\|_2\left\|X^{-1}\right\|_2}_{\kappa_2(X)} \|p(\Lambda)\|_2\|r_0\|_2.$$

Q.E.D. ■

8.4 Lanczos Method

1. Overview: Arnoldi for Symmetric Matices

$$AQ_n = Q_{n+1}\overline{T}_n,$$

where T_n is tridiagonal.

$$A\begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | & | \\ q_1 & q_2 & \cdots & q_n & q_{n+1} \\ | & | & & | & | \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \beta_2 & \\ & \beta_2 & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \\ & & & * \end{bmatrix}; \qquad \text{Note: } \overline{T}(:,j) = \begin{bmatrix} \vdots \\ \beta_{j-1} \\ \alpha_j \\ \beta_j \\ \vdots \end{bmatrix}$$

$$Aq_j = Q_{n+1}\overline{T}(:,j)$$

$$Aq_j = \beta_{j-1}\underline{q_{j-1}} + \alpha_j\underline{q_j} + \boxed{\beta_j q_{j+1}}$$

where β_{j-1} and α_j are computable, and q_{j-1} and q_j are known

$$\beta_j q_{j+1} = Aq_j - \beta_{j-1}q_{j-1} - \alpha_j q_j \qquad \text{(3 Term Recurrsion)}$$

Algorithm 22: Lanczos Method

```
1 begin
2     q_1 = b/‖b‖_2, β_0 = 0, q_0 = 0;
3     for n = 1, 2, ... do
4         v = A q_n;
5         α_n = q_n^* v;
6         v = v − β_{n−1} q_{n−1} − α_n q_n // orthogonalization
7         β_{n+1} = ‖v‖_2;
8         q_{n+1} = v/β_{n+1};
```

8.5 Gradient Descent (GD)

Remark 8.3 *In this section and the next, assume A to be real and SPD.*

1. Problem Set-up

$$\min_x f(x) \equiv \frac{1}{2}x^\top Ax - b^\top x.$$

Gradient:

$$\nabla f(x) = Ax - b.$$

2. In each iteration

$$x_n = x_{n-1} - \alpha_n \nabla f(x_{n-1}) \tag{GD}$$

Claim 8.3 Connection with Solving Linear System

GD solves this problem by solving the system $Ax = b$.

Proof 1. By first order condition: $\nabla f(x) = 0 \implies Ax - b = 0$. Q.E.D. ■

4. Rewrite (GD) **using residual:** $r = b - Ax$

Define $\nabla f(x) = Ax - b = -r$, negative residual. Then, GD iteration gives

$$x_n = x_{n-1} + \alpha_n r_{n-1} \tag{GD}$$

5. Can we update r_n iteratively?

That is, can we rewrite $r_n = r_{n-1} +$ update. Note that

$$\begin{aligned} r_n &\equiv b - Ax_n \\ &= b - A(x_{n-1} + \alpha_n r_{n-1}) \\ &= \underbrace{b - Ax_{n-1}} - \alpha_n A r_{n-1} \\ &= r_{n-1} - \alpha_n A r_{n-1} \end{aligned}$$

6. Hoe to choose α_n? Pick the *optimal* one

Define $\varphi(\alpha) \equiv f(x_{n-1} + \alpha r_{n-1})$.

Goal: Find α $s.t.$ $\varphi'(\alpha) = 0$.

$$\begin{aligned} \varphi(\alpha) &= \frac{1}{2}(x_{n-1} + \alpha r_{n-1})^\top A(x_{n-1} + \alpha r_{n-1}) - b^\top(x_{n-1} + \alpha r_{n-1}) \\ &= f(x_{n-1}) + \alpha x_{n-1}^\top A r_{n-1} + \frac{1}{2}\alpha^2 r_{n-1}^\top A r_{n-1} - \alpha b^\top r_{n-1}. \\ \varphi'(\alpha) * &= x_{n-1}^\top A r_{n-1} + \alpha r_{n-1}^\top A r_{n-1} - b^\top r_{n-1} \overset{\text{set}}{=\!=} 0 \\ \alpha_n^* &= \frac{b^\top r_{n-1} - x_{n-1}^\top A r_{n-1}}{r_{n-1}^\top A r_{n-1}} \end{aligned}$$

Algorithm 23: Gradient Descent

Input: A, b, x_0

begin
- $r_0 = b - Ax_0$;
- **for** $n = 1, 2, \ldots$ **do**
  - $\alpha_n = \dfrac{b^\top r_{n-1} - x_{n-1}^\top \boxed{Ar_{n-1}}}{r_{n-1}^\top \boxed{Ar_{n-1}}}$;
  - $x_n = x_{n-1} + \alpha_n r_{n-1}$;
  - $r_n = r_{n-1} - \alpha_n \boxed{Ar_{n-1}}$;
  - // Computing Ar_{n-1} is expensive, but we can store its values and avoid repeated computation.

8.6 Conjugate Gradient (CG)

Algorithm 24: Conjugate Gradient

Input: A, b, x_0

begin
- $r_0 = b - Ax_0$;
- $p_0 = r_0$;
- **for** $n = 1, 2, \ldots$ **do**
  - $\alpha_n = \dfrac{\boxed{r_{n-1}^\top r_{n-1}}}{p_{n-1}^\top \boxed{Ap_{n-1}}}$;
  - $x_n = x_{n-1} + \alpha_n p_{n-1}$;
  - $r_n = r_{n-1} - \alpha_n \boxed{Ap_{n-1}}$;
  - $\beta_n = \dfrac{r_n^\top r_n}{\boxed{r_{n-1}^\top r_{n-1}}}$;
  - $p_n = r_n + \beta_n p_{n-1}$;
  - // Improve computational cost: store those boxed values to avoid repeated computation

Definition 8.6.1 (A-norm). Let A be real-valued SPD matrix. Then, the A-*norm* of x is given by

$$\|x\|_A = \sqrt{x^\top Ax}.$$

2. What is p_n's? Conjugate Gradient

- Conjugate: p_n's are A-*conjugate*: orthogonal w.r.t. inner product $\langle \cdot, \cdot \rangle_A$.

$$\langle p_k, p_j \rangle_A = p_k^\top Ap_j = 0 \quad \text{if } k \neq j$$

- Gradient: search direction to update x_n.

3. Properties of CG

- $x_n \in \mathcal{K}_n(A,b)$.
- Residual are orthogonal:

$$r_k^\top r_j = 0 \quad \text{for } k \neq j \implies r_n \perp \mathcal{K}_{n-1}(A,b),$$

  allows CG to converge fast if at most m iterations (*under some assumptions*).

- Search directions are A-conjugate:

$$p_k^\top A p_j = 0 \quad \text{for } k \neq j \implies \text{efficient search through space.}$$

4. How do we ensure orthogonal residual?

Goal: $r_n^\top r_{n-1} = 0$

$$\begin{aligned} r_n^\top r_{n-1} &= (r_{n-1} - \alpha A p_{n-1})^\top r_{n-1} && \text{Line \#7 from Algorithm (24)} \\ &= r_{n-1}^\top r_{n-1} - \alpha p_{n-1}^\top A r_{n-1} \overset{\text{set}}{=} 0 && A = A^\top \text{ since } A \text{ is symmetric} \end{aligned}$$

Solve for α, we get

$$\alpha_n = \frac{r_{n-1}^\top r_{n-1}}{p_{n-1}^\top A r_{n-1}}.$$

To get α_n matching Algorithm (24), plug $p_{n-1} = r_{n-1} + \beta_{n-1} p_{n-2}$ into α_n.

5. How to ensure A-conjugate search direction?

Goal: $p_{n-1}^\top A p_n = 0$.

$$\begin{aligned} p_{n-1}^\top A p_n &= p_{n-1}^\top A (r_n + \beta p_{n-1}) && \text{Line \#9 of Algorithm (24)} \\ &= p_{n-1}^\top A r_n + \beta p_{n-1}^\top A p_{n-1} \overset{\text{set}}{=} 0 \end{aligned}$$

Solve for β:

$$\beta_n = \frac{p_{n-1}^\top A r_n}{p_{n-1}^\top A p_{n-1}}.$$

To get β_n exactly matching Algorithm (24), plut $r_n = r_{n-1} - \alpha_n A p_{n-1}$ into β_n.

Theorem 8.6.6
If A is SPD, and CG has not already converged (i.e., $r_n \neq 0$), then $x_n \in \mathcal{K}_n(A,b)$ is unique that minimizes $\|x_* - x_n\|_A$.

Remark 8.4 *Note that x_n is the best approximation to the solution of $Ax = b$ that lives in $\mathcal{K}_n(A,b)$.*

Proof 1. Pick arbitrary $x = x_n - \Delta x \in \mathcal{K}_n$.
Goal: $\Delta x = 0$.

Consider error: $e = x - x_* = e_n + \Delta x$. Then,

$$\begin{aligned}\|x\|_A^2 &= (e_n + \Delta x)^\top A(e_n + \Delta x) \\ &= \|e_n\|_A^2 + 2r_n^\top(\Delta x) + \|\Delta x\|_A^2 \qquad \text{expand and simplify}\end{aligned}$$

Recall that residuals are constructed such that $r_n \perp \mathcal{K}_n(A,b) \implies r_n \perp \Delta x$

$\implies \|e\|_A^2 = \|e_n\|_A^2 + \|\Delta x\|_A^2$ is mallest when $\Delta x = 0$. Q.E.D. ■

7. Intuition: Building A-conjugate basis

Given $\{q_1, \dots, q_n\}$ orthonormal. How to build $\{p_1, \dots, p_n\}$, A-conjugate, where A is SPD?

Since A is SPD, by Cholesky Facotrization, we have $A = LL^\top$. Define $p_i = L^{-\top} q_i$. Then,

$$p_i^\top A p_j = q_i^\top L^{-1}\Big(LL^\top\Big)L^{-\top} q_j = q_i^\top q_j = 0 \quad \text{for } i \neq j.$$

Suppose $\mathcal{K}_n(A,b) = \operatorname{span}\{p_1, \dots, p_n\}$. Then, $\mathcal{K}_{n+1}(A,b) = \mathcal{K}_n(A,b) \cup \operatorname{span}\{A^n b\}$. So, we have

$$\begin{aligned}p_j^\top(A^n b) &= p_j^\top A \underbrace{\big(A^{n-1}b\big)}_{\in \mathcal{K}_n(A,b)} \\ &= p_j^\top A(c_1 p_1 + \cdots + c_n p_n) \qquad \text{almost everything cancels by orthogonality} \\ &= c_j p_j^\top A p_j\end{aligned}$$

However, if we directly consider $q_j^\top(A^n b) = q_j^\top A(c_1 q_1 + \cdots + c_n q_n) =$??? This is why we relay on A-conjugate bases in CG.

Theorem 8.6.8 Connect Residual with Error

$$\|r_n\|_{A^{-1}} = \|e_n\|_A$$

Remark 8.5 *Intuitively, this connection makes sense: e_n lives in the input space so we can compute its A-norm, whereas r_n lives in the output space so we need to compute its A^{-1}-norm.*

Proof 2.

$$\begin{aligned}\|r_n\|_{A^{-1}}^2 = r_n^\top A^{-1} r_n &= (b - Ax_n)^\top A^{-1}(b - Ax_n) \\ &= (Ax_* - Ax_n)^\top A^{-1}(Ax_* - Ax_n) && Ax_* = b \\ &= (A(x_* - x_n))^\top A^{-1}(A(x_* - x_n)) && e_n = x_* - x_n \\ &= e_n^\top \underbrace{A^\top A^{-1}}_{AA^{-1}=I} A e_n && A \text{ is SPD, } A^\top = A \\ &= e_n^\top A e_n \\ &= \|e_n\|_A^2\end{aligned}$$

Q.E.D. ■

Theorem 8.6.9 Convergence of CG

$$\frac{\|r_n\|_{A^{-1}}}{\|r_0\|_{A^{-1}}} = \frac{\|e_n\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n,$$

where $e_n = x_* - x_n$ if x_* denotes the true solution, and $\kappa = \kappa_2(A)$.

Remark 8.6

$$(\textit{identity matrix, CG solves instantly}) \quad 0 \leq \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \leq 1 \quad (\textit{large } \kappa\textit{, slow convergence})$$

10. Polynomial Approximation of Error for CG & Proof of Theorem 8.6.9

Let $f(z) = \|b - Az\|^2_{A^{-1}}$. Consider the following optimization problem:

$$f(x_n) = \min_{z \in \mathcal{K}_n(A,b)} f(z), \qquad \text{(CG Problem)}$$

where $z \in \mathcal{K}_n(A,b) \implies z = p_{n-1}(A)b = \underbrace{p_{n-1}(A)A}_{q_n(A)} x_* = q_n(A)x_*$.

Now, we can write (CG Problem) as

$$\begin{aligned}
f(x_n) &= \min_{z \in \mathcal{K}_n(A,b)} f(z) \\
&= \min_{z \in \mathcal{K}_n(A,b)} \underbrace{(x_* - z)^\top}_{[(I - q_n(A))x_*]^\top} A(x_* - z) && \|e_n\|_A = \|r_n\|_{A^{-1}} \\
f(x_n) &= \min_{\substack{p_n \in P_n \\ p_n(0)=1}} x_*^\top p_n(A) A p_n(A) x_* \\
&= \min_{\substack{p_n \in P_n \\ p_n(0)=1}} \|p_n(A)e_0\|_A, && \text{(Polynomial Approx. of Error for CG)}
\end{aligned}$$

with $x_0 = 0$ and thus $e_0 = x_* - x_0 = x_*$.

Since A is SPD, we can orthogonally diagonalize it: $A = Q\Lambda Q^\top$. Denote $\|r_n\|^2_{A^{-1}} = \|e_n\|^2_A = f(x_n)$.

$$\begin{aligned}
f(x_n) &= \min_{p_n \in P_n, p_n(0)=1} y^\top \underbrace{p_n(\Lambda)\Lambda p_n(\Lambda)}_{\text{diagonal}} y && y = Q^\top x_* \\
&= \min_{p_n \in P_n p_n(0)=1} \sum_{i=1}^m y_i^2 \lambda_i p_n(\lambda_i)^2 && \lambda_i\text{:eigenvalue},\ m\text{:\# of eigenvalues} \\
&\leq \min_{p_n \in P_n p_n(0)=1} \left(\max_{\lambda_i \in \Lambda(A)} p_n(\lambda_i)^2\right) \boxed{\sum_{i=1}^m y_i^2 \lambda_i} && \text{If } x_0{=}0\text{, boxed part} = f(x_0) = \|r_0\|^2_{A^{-1}} \text{ or } \|e_0\|^2_A \\
\frac{f(x_n)}{f(x_0)} &\leq \min_{p_n \in P_n p_n(0)=1} \left(\max_{\lambda_i \in \Lambda(A)} p_n(\lambda_i)^2\right) && \text{take square root} \\
\frac{\|e_n\|_A}{\|e_0\|_A} &\leq \min_{p_n \in P_m, p_n(0)=1} \left(\max_{\lambda_i \in \Lambda(A)} |p_n(\lambda_i)|\right) \leq \max_{\lambda_i \in \Lambda(A)} |\widehat{p}_n(\lambda_i)| && \text{We don't solve the problem exactly, We just want to find an upper bound}
\end{aligned}$$

It turns out that Chebyshev polynomial is a good choice.

Definition 8.6.11 (Chebyshev Polynomial). For $x \in [-1,1]$, the Chebyshev polynomial is given by

$$T_n(x) = \cos\left(n\cos^{-1}(x)\right), \quad \text{with } |T_n(x)| \leq 1.$$

After shifting the polynomial to fit in $\Lambda(A)$, we have that $\xi \in [\lambda_{\min}, \lambda_{\max}]$, and

$$\widehat{q}_n(\xi) = T_n\left(\frac{\lambda_{\max} + \lambda_{\min} - 2\xi}{\lambda_{\max} - \lambda_{\min}}\right).$$

8.7 Polynomial Approximation Perspective

1. Arnoldi: Gram-Schmidt-ish method to find the next q_{n+1}

$$\min_{p_n \in P_n, \text{monic}} \|p_n(A)b\|_2$$

Note that $p_n(A)b = A^n b - Q_n y$. The closest vector in $\mathcal{K}_n(A,b)$, $Q_n y$, to next vector $A^n b$. Form normal questions:

$$A_n^* A^n b = \cancel{Q_n^*}\cancel{Q_n} y \implies y = Q_n^* A^n b.$$

Closest vector: $Q_n y = Q_n Q_n^* A^* b$ can be viewed as an orthogonal projection. Next q_{n+1}:

$$q_{n+1} = (I_m - Q_n Q_n^*) A^n b$$

A Further Explanation

- Arnoldi's method finds $q_1, \ldots, q_n$, an orthonormal basis of the Krylov subspace $\mathcal{K}_n(A,b) = \text{span}\left\{b, Ab, A^2 b, \ldots, A^{n-1}b\right\}$.
- **Arnoldi Relation**:

  $$AQ_n = Q_{n+1} H_n,$$

  where $Q_n = \begin{bmatrix} q_1 & \ldots & q_n \end{bmatrix}$ and H_n is an upper Hessenberg matrix.

  By construction of Q_n, we know that $Q_n y \in \mathcal{K}_n(A,b)$.

  For the next iteration, we add $A^n b$ to the Krylov subspace, and seek q_{n+1} $s.t.$

  $$q_{n+1} \perp \mathcal{K}_n(A,b) \cup \{A^n b\}.$$

  That is, we minimize the distance between $A^n b$ and the Krylov subspace.
- **Optimization Problem**:

  As $Q_n y \in \mathcal{K}_n(A,b)$, rewrite the problem as minimizing the residual $r_n = A^n b - Q_n y$:

  $$\min_{y \in \mathbb{C}^n} \|r_n\|_2 = \min_{y \in \mathbb{C}^n} \|A^n b - Q_n y\|_2. \qquad \text{(Arnoldi Approx.)}$$

- **Polynomial Representation**: Since $Q_n y \in \mathcal{K}_n(A,b) = \text{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\}$, rewrite $Q_n y$ as a linear combination of $b, Ab, \ldots, A^{n-1}b$:

$$Q_n y = y_1 b + y_2 Ab + \cdots + y_n A^{n-1} b.$$

Substituting into (Arnoldi Approx.), we have:

$$\begin{aligned}
\min_{y\in\mathbb{C}^n} \|r_n\| &= \min_{y\in\mathbb{C}^n} \|A^n b - Q_n y\|_2 \\
&= \min_{y\in\mathbb{C}^n} \left\|A^n b - (y_1 b + y_2 Ab + \cdots + y_n A^{n-1} b)\right\|_2 \\
&= \min_{y\in\mathbb{C}^n} \left\|(-y_1 b - y_2 Ab - \cdots - y_n A^{n-1} b + A^n b)\right\|_2 \\
&= \min_{y\in\mathbb{C}^n} \|(\underbrace{-y_1 I - y_2 A - \cdots - y_n A^{n-1} + A^n}_{p_n(A)})b\|_2 \\
&= \min_{\substack{p_n\in P_n \\ p_n \text{ monic}}} \|p_n(A)b\|_2.
\end{aligned}$$

2. GMRES: Solve $Ax = b$

$$\min_{p_n\in P_n, p_n(0)=1} \|p_n(A)b\|_2$$

$$\begin{aligned}
p_n(A)b = (I - Aq_{n-1}(A))b &= b - Aq_{n-1}(A)b && q_{n-1}(A)b \in \mathcal{K}_n(A,b) \\
&= b - AQ_n y && Q_n y \in \mathcal{K}_n(A,b)
\end{aligned}$$

Find coefficients y *s.t.* $A \underbrace{Q_n y}_{\text{solution}}$ is as close as possible to b.

3. CG: Solve $Ax = b$

$$\min_{p_n\in P_n, p_n(0)=1} \|p_n(A)e_0\|_A \quad \text{where } e_0 = x_* - x.$$