

1 System of Linear Equations, $Ax = b$

1.1 p -Norm and Condition Number

Vector p -Norm: $\boxed{\|\vec{x}\|_p = \sqrt[p]{\sum_i |x_i|^p}}$

1-Norm : $\|\vec{x}\|_1 = \sum_i |x_i|$

∞ -Norm : $\|\vec{x}\|_\infty = \max |x_i|$

- $\|x\|_1 \geq \|x\|_2 \geq \|x\|_\infty$
- $\|x\|_1 \leq \sqrt{n} \|x\|_2 \leq \sqrt{n} \|x\|_\infty$

Matrix p -Norm: $\boxed{\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}}$

1-Norm : $\|A\|_1 = \max_j \sum_i |a_{ij}|$

∞ -Norm : $\|A\|_\infty = \max_i \sum_j |a_{ij}|$

- $\|AB\| \leq \|A\| \cdot \|B\|$
 - $\|Ax\| \leq \|A\| \cdot \|x\|$
- For p -norms (not necessarily in general)

Function/Vector Condition Number:

$$\begin{aligned} \text{cond}(f(x)) &= \left| \frac{[f(\hat{x}) - f(x)]/f(x)}{[\hat{x} - x]/x} \right| \\ &= \left| \frac{\Delta y/y}{\Delta x/x} \right| = \left| \frac{y' \cdot \Delta x/y}{\Delta x/x} \right| \\ &= \left| \frac{x f'(x)}{f(x)} \right| \end{aligned}$$

Matrix Condition Number:

$\boxed{\text{cond}_p(A) = \|A\|_p \cdot \|A^{-1}\|_p}$ (∞ if singular)

$$= \frac{\max_{x \neq 0} \|Ax\|_p / \|x\|_p}{\min_{x \neq 0} \|Ax\|_p / \|x\|_p} = \text{cond}_p(\gamma A) \geq 1$$

- Diagonal, D : $\text{cond}(D) = \frac{\max |d_i|}{\min |d_i|}$
- $\|z\| = \|A^{-1}y\| \leq \|A^{-1}\| \cdot \|y\|$
 $\rightarrow \frac{\|z\|}{\|y\|} \leq \max \frac{\|z\|}{\|y\|} \stackrel{?}{=} \|A^{-1}\|$ (optimize)

1.2 Error Bounds and Residuals

Error Bound: $\boxed{\frac{\|\hat{x} - x\|}{\|x\|} \lesssim \text{cond}(A) \epsilon_{\text{mach}}}$ \rightarrow A computed solution is expected to lose about $\log_{10}(\text{cond}(A))$ digits, so the input data must be more accurate to these digits and the working precision must carry more than these digits.

$$A\hat{x} = b + \Delta b = Ax + A\Delta x$$

- $\|b\| \leq \|A\| \cdot \|x\|$
- $\|\Delta x\| \leq \|A^{-1}\| \cdot \|\Delta b\|$

$$\rightarrow \boxed{\frac{\|\Delta x\|}{\|x\|} \leq \text{cond}(A) \frac{\|\Delta b\|}{\|b\|}}$$

$$A\hat{x} + r = b$$

- $\|\Delta x\| = \|A^{-1}(A\hat{x} - b)\| = \|-A^{-1}r\|$
 $\leq \|A^{-1}\| \cdot \|r\|$

$$\rightarrow \boxed{\frac{\|\Delta x\|}{\|\hat{x}\|} \leq \text{cond}(A) \frac{\|r\|}{\|A\| \cdot \|\hat{x}\|}}$$

$$(A + \Delta A)\hat{x} = b$$

- $\|\Delta x\| = \|-A^{-1}(\Delta A)\hat{x}\|$
 $\leq \|A^{-1}\| \cdot \|\Delta A\| \cdot \|\hat{x}\|$

$$\rightarrow \boxed{\frac{\|\Delta x\|}{\|x\|} \leq \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}}$$

$$(A + \Delta A)\hat{x} = b$$

- $\|r\| = \|b - A\hat{x}\| = \|\Delta A \cdot \hat{x}\|$
 $\leq \|\Delta A\| \cdot \|\hat{x}\|$

$$\rightarrow \boxed{\frac{\|r\|}{\|A\| \cdot \|\hat{x}\|} \leq \frac{\|\Delta A\|}{\|A\|}}, \quad \frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \cdot \|r\|}{\|\hat{x}\|} \leq \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}$$

$$\left[A(t)x(t) = b(t) \right] = \left[(A_0 + \Delta A \cdot t)x(t) = b_0 + \Delta b \cdot t \right]$$

- $x'(t) = \frac{b'(t) - A'(t)x(t)}{A(t)} = A^{-1}(t) \left[\Delta b - \Delta A \cdot x(t) \right]$
- $x(t) = x_0 + x'(0)t + \mathcal{O}(t^2)$

$$\rightarrow \boxed{\frac{\|x(t) - x_0\|}{\|x_0\|} \leq \text{cond}(A) \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right) |t| + \mathcal{O}(t^2)}$$

1.3 Gaussian Elimination with LU/PLU/PLDUQ Decomposition

Elementary Elimination Matrices, L_k

$$\begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \frac{-a_{k+1}}{a_k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \frac{-a_n}{a_k} & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- a_k is the “pivot”
- is lower triangular
- $\forall i \neq j \quad (L_k^{-1})_{ij} = -(L_k)_{ij}$

Ex :

$$\begin{pmatrix} 1 & 0 & \dots \\ -a_1/a_2 & 1 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} a_1 \\ 0 \\ \vdots \end{pmatrix}$$

LU/PLU Factorization (w/ partial pivoting)

$$\boxed{A = LU \quad \begin{array}{l} (L \text{ is gen. triang.}) \\ (U \text{ is upp. triang.}) \end{array}} \\ \boxed{L = (\dots L_2 P_2 L_1 P_1)^{-1}}$$

$$\{\dots\}b = (\dots L_2 P_2 L_1 P_1)Ax$$

$$L^{-1}b = (P_1^T L_1^{-1} P_2^T L_2^{-1} \dots)^{-1} Ax \\ = L^{-1}(LU)x = y$$

$$\boxed{b = Ly \quad y = Ux} \\ \text{(forw.-sub.) , (back.-sub.)}$$

- Permutation matrix, P_i , rowswaps s.t. $a_k \neq 0$
- P_i rowswaps s.t. a_k is largest s.t. $a_{k+i}/a_k \leq 1$ for numerical stability/minimize errors
- Pivoting isn't needed if A is diag. dom. ($a_{jj} > \sum_{i \neq j} a_{ij}$)
- A can be singular

$$\boxed{A = PLU \quad \begin{array}{l} (P \text{ is rowswap permu.}) \\ (L \text{ is unit low. triang.}) \\ (U \text{ is upp. triang.}) \end{array}} \\ \boxed{P = (\dots P_2 P_1)^{-1}}$$

$$\{\dots\}b = (\dots P_2 P_1)Ax$$

$$P^T b = (P_1^T P_2^T \dots)^{-1} Ax \\ = P^T (PLU)x = Ly$$

$$\boxed{P^T b = Ly \quad , \quad y = Ux}$$

$$\boxed{P^T A = LDU \quad (D \text{ is diag.})}$$

- LDU is unique up to D
- LDU is unique if L/U are unit low./upp. diag., resp.

$$\boxed{P^T A Q^T = LDU \quad \begin{array}{l} (P \text{ is permu. for rows}) \\ (Q \text{ is permu. for cols.}) \end{array}}$$

- “Complete pivoting” search for largest a_k
- Would be most numerically stable
- Expensive, so not really used

$$\text{Error Bound: } \frac{\|r\|}{\|A\|\|x\|} \leq \frac{\|\Delta A\|}{\|A\|} \leq \rho n^2 \epsilon_{\text{mach}} \sim n \epsilon_{\text{mach}} \\ \text{(Wilkinson) (usually)}$$

(growth factor, ρ , is the largest entry at any point during factorization - usually at U - divided by the largest entry of A)

1.4 Gaussian-Jordan with MD Decomposition

Elementary Elimination Matrices, M_k

$$\begin{pmatrix} 1 & \dots & \frac{-a_1}{a_k} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & \frac{-a_{k+1}}{a_k} & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \frac{-a_n}{a_k} & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- a_k is the “pivot”
- $\forall i \neq j \quad (M_k^{-1})_{ij} = -(M_k)_{ij}$

MD Factorization (w/ partial pivoting)

$$\boxed{A = MD \quad \begin{array}{l} (M \text{ is elem. elim.}) \\ (D \text{ is diag.}) \end{array}}$$

$$M = (\dots M_2 P_2 M_1 P_1)^{-1}$$

$$\{\dots\}b = (\dots M_2 P_2 M_1 P_1)Ax$$

$$M^{-1}b = (P_1^T M_1^{-1} P_2^T M_2^{-1} \dots)^{-1} Ax$$

$$= M^{-1}(MD)x = y$$

$$\boxed{M^{-1}b = y, \quad y = Dx}$$

(division)

- Permutation matrix, P_i , rowswaps s.t. $a_k \neq 0$
- P_i rowswaps cannot ensure numerical stability (≤ 1)
- Division is $\mathcal{O}(n)$, so may be useful for parallel comps.
- Can also find A^{-1}

Finding A^{-1}

$$D^{-1}M^{-1}(A|I) = (I|A^{-1})$$

$$= D^{-1}M^{-1} \left[\begin{array}{ccc|cc} a_{11} & \dots & 1 & 0 \\ \vdots & & a_{nn} & 0 \\ \hline 1 & 0 & a'_{11} & \dots \\ 0 & 1 & \vdots & a'_{nn} \end{array} \right]$$

1.5 Symmetric Matrices

Positive Definite: $x^T Ax \geq 0$

Cholesky Factorization for Sym., Pos. Def.: $A = LL^T = LDL^T$

$$\begin{pmatrix} a_{11} & a_{21} & a_{31} & \dots \\ a_{21} & a_{22} & a_{32} & \dots \\ a_{31} & a_{32} & a_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 & \dots \\ l_{21} & l_{22} & 0 & \dots \\ l_{31} & l_{32} & l_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} & \dots \\ 0 & l_{22} & l_{32} & \dots \\ 0 & 0 & l_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} l_{11}^2 & \dots & \dots & \dots \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & \dots & \dots \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

- Pivoting not needed
- Well defined (always works)
- Only lower triangle needed for storage
- $A = LDL^T$ is sometimes useful, where D is diag.

Symmetric Indefinite Matrices

- Pivoting Needed: $PAP^T = LDL^T$
- Ideally, D is diag., but if not possible, then D is tridiag. (Aasen) or 1x1/2x2 block diag. (Bunch, Parlett, Kaufmann, etc.)

1.6 Banded Matrices

- Similar to normal Gaussian Elim., but less work since more zeroes
- Pivoting means bandwidth will expand no more than double
- Only $\mathcal{O}(\beta n)$ storage needed

1.7 Rank-1 Update with Sherman-Morrison

$$\tilde{A}\tilde{x} = b = (A - uv^T)\tilde{x} \quad \left| \quad \begin{array}{l} \tilde{A}^{-1} = (A - uv^T)^{-1} = A^{-1} + \frac{A^{-1}u}{1 - v^T(A^{-1}u)} v^T A^{-1} \\ \tilde{A}^{-1}b = \quad \quad \quad \tilde{x} = (A^{-1}b) + \frac{A^{-1}u}{1 - v^T(A^{-1}u)} v^T(A^{-1}b) \\ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad x + \frac{y}{1 - v^T y} v^T x \end{array} \right.$$

General Woodbury Formula: $\boxed{(A - UV^T)^{-1} = A^{-1} + (A^{-1}U)(I - V^T A^{-1}U)^{-1} v^T A^{-1}}$

- U and V are general $n \times k$ matrices
- No guarantee of numerical stability, so caution is needed

1.8 Complexity

Explicit Inversion : $\frac{LUA^{-1} = I}{D^{-1}M^{-1}I = A^{-1}} \rightarrow \mathcal{O}(n^3) \quad , \quad A^{-1}b = x \rightarrow \mathcal{O}(n^2)$

Gaussian Elimination : $A = LU \rightarrow \mathcal{O}(n^3/3) \quad , \quad LUx = b \rightarrow \mathcal{O}(n^2)$

Gaussian-Jordan : $A = MD \rightarrow \mathcal{O}(n^3/2) \quad , \quad MDx = b \rightarrow \mathcal{O}(n)$

Symmetric : $\frac{A = LL^T}{PAP^T = LDL^T} \rightarrow \mathcal{O}(n^3/6) \quad , \quad LL^T x = b \rightarrow \mathcal{O}(n^2)$

Banded : $A_\beta = LU \rightarrow \mathcal{O}(\beta^2 n) \quad , \quad LUx = b \rightarrow \mathcal{O}(\beta n)$

Sherman-Woodbury : $\tilde{A} = A - uv^T \rightarrow \mathcal{O}(n^2) \quad , \quad \tilde{x} = \tilde{A}b \rightarrow \mathcal{O}(n^2)$

1.9 Diagonal Scaling

Ill-conditioned

$$\begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \epsilon \end{pmatrix}$$

Well-conditioned

$$\begin{pmatrix} 1 & 0 \\ 0 & 1/\epsilon \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1/\epsilon \end{pmatrix} \begin{pmatrix} 1 \\ \epsilon \end{pmatrix}$$

- No general way to correct poor scaling

1.10 Iterative Refinement

$$\begin{aligned} r_0 &= b - Ax_0 = A\Delta x_0 \\ r_1 &= b - A(x_0 + \Delta x_0) = b - Ax_1 = A\Delta x_1 \\ r_2 &= b - A(x_1 + \Delta x_1) = b - Ax_2 = A\Delta x_2 \end{aligned}$$

$$\boxed{x = x_0 + \lim_{n=0}^{\infty} \Delta x_n} \quad (\text{terminate when } r_n \text{ is small enough})$$

- Double storage needed to hold original matrix
- r_n usually must be computed with higher precision than x_n
- Useful for badly scaled systems, or making unstable systems stable
- If x_n is not accurate, r_n might not need better accuracy

2 Least $\|r\|$ Linear Regression/Fit, $Ax + r = b$

- $A = A_{m \times n}$ $\boxed{(m > n, \text{ underdetermined})}$
- $r(y = Ax)$ is cont. & coer. $\rightarrow \exists \|r(y)\|_{\min}$
- $r(y)$ is strictly convex $\rightarrow y = Ax$ is unique
- $\boxed{\text{rank}(A) = n}$ $\Rightarrow A(x_1 - x_2) = 0$ (unique x)
 $\boxed{(\text{full column rank})} \quad (x_1 - x_2) = 0 \rightarrow x_1 = x_2$

Example - Vandermonde Matrix, A :

$$Ax = \begin{pmatrix} -\vec{f}(t_1) \\ \vdots \\ -\vec{f}(t_m) \end{pmatrix} \begin{pmatrix} | \\ \vec{x} \\ | \end{pmatrix} = \begin{pmatrix} y(t_1) \\ \vdots \\ y(t_m) \end{pmatrix} = \begin{pmatrix} | \\ \vec{y} \\ | \end{pmatrix} = (x^T A^T)^T, \quad y(t) = \sum_{i=1}^n x_i f_i(t) = \vec{x} \cdot \vec{f}$$

Decompose b :

$$\begin{aligned} b &= Ax + r \\ &= y + r \\ &= Pb + P_{\perp} b \end{aligned}$$

Projector of A , P

$$\text{Projector : } P^2 = P \rightarrow PA = A$$

(Idempotent) (Projector of A)

$$\text{Orthogonal Projector : } P^T = P \rightarrow P_{\perp} A = (I - P)A = 0$$

Minimize residual, r :

$$\begin{aligned} \nabla \|r\|_2^2 &= 0 \quad \left(\frac{\partial x^2}{\partial x_i} = 0 \right) \\ &= \nabla [(b - Ax)^T (b - Ax)] \\ &= \nabla (b^T b - 2x^T A^T b + x^T A^T A x) \\ 0 &= 2A^T A x - 2A^T b \\ &\downarrow \\ A^T A x &= A^T b \quad (\text{Solvable with Cholesky}) \end{aligned}$$

$$\begin{aligned} \|r\|_2^2 &= \|Pr + P_{\perp} r\|_2^2 = \|b - Ax\|_2^2 \\ &= \|Pr\|_2^2 + \|P_{\perp} r\|_2^2 \\ &= \cancel{\|Pb - Ax\|_2^2} + \|P_{\perp} b\|_2^2 \\ &\downarrow \\ Ax &= Pb \\ A^T A x &= A^T Pb = (P^T A)^T b \\ A^T A x &= A^T b \quad (\text{System of Normal Equations}) \end{aligned}$$

Cross-Product Matrix of A : $\boxed{A^T A}$

$$\text{Symmetric : } (A^T A)^T = A^T A$$

$$\begin{aligned} \text{Pos. Def. : } \text{rank}(A) &= n \\ &\rightarrow \langle x | A^T A x \rangle = x^T A^T A x \\ &= (Ax)^T (Ax) \\ &= \|Ax\|^2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{Nonsingular : } A^T A x &= 0 \\ &\rightarrow \|Ax\|^2 = 0 = Ax \\ &\rightarrow (x = 0) \end{aligned}$$

System of Normal Equations: $\boxed{A^T A x = A^T b}$

Pseudoinverse, A^+

$$\boxed{x = (A^T A)^{-1} A^T b} \rightarrow \boxed{A^+ \equiv (A^T A)^{-1} A^T}$$

$$\equiv A^+ b \quad A^+ A = I$$

Ortho. Proj., P

$$\boxed{Ax = A(A^T A)^{-1} A^T b} \rightarrow \boxed{P = A(A^T A)^{-1} A^T}$$

$$= Pb \quad = AA^+$$

System of Normal Equations Issues:

- Info can be lost forming $A^T A$, e.g, $A = \begin{pmatrix} 1 & 0 \\ \epsilon & 0 \\ 0 & \epsilon \end{pmatrix} \rightarrow A^T A = \begin{pmatrix} 1+\epsilon^2 & 1 \\ 1 & 1+\epsilon^2 \end{pmatrix} \approx \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ (singular)

- System of Normal Equations: $\boxed{\text{cond}(A^T A) = [\text{cond}(A)]^2}$

2.1 Error Bounds and Residuals

Error Bound: $\boxed{\frac{\|\Delta x\|}{\|x\|} \lesssim \text{cond}(A) \epsilon_{\text{mach}}}$ \rightarrow A computed solution is expected to lose about $\log_{10}(\text{cond}(A))$ digits, so the input data must be more accurate to these digits and the working precision must carry more than these digits.

Norm and Conditioning:

$$\|A\| = \max_{x \neq 0} \left(\frac{\|Ax\|}{\|x\|} = \frac{\|AA^+b\|}{\|A^+b\|} \right)$$

$$\text{cond}(A) = \begin{cases} \|A\|_2 \cdot \|A^+\|_2 & \text{rank}(A) = n \\ \infty & \text{rank}(A) < n \end{cases}$$

$$A^T A(x + \Delta x) = A^T A(b + \Delta b)$$

$$(A + \Delta A)^T (A + \Delta A)(x + \Delta x) = (A + \Delta A)^T b$$

$$\bullet \quad \|\Delta x\| \leq \|A^+\| \cdot \|\Delta b\|$$

$$\bullet \quad \cancel{A^T Ax} + A^T \Delta Ax + (\Delta A)^T Ax + \cancel{(\Delta A)^T \Delta Ax} = \cancel{A^T b} + (\Delta A)^T b \\ + A^T A \Delta x + \cancel{A^T \Delta A \Delta x} + \cancel{(\Delta A)^T A \Delta x} + \cancel{(\Delta A)^T \Delta A \Delta x}$$

$$\rightarrow \boxed{\frac{\|\Delta x\|}{\|\hat{x}\|} \leq \left(\text{cond}(A) \frac{\|b\|}{\|Ax\|} \right) \frac{\|\Delta b\|}{\|b\|} \\ = \left(\text{cond}(A) \frac{1}{\cos \theta} \right) \frac{\|\Delta b\|}{\|b\|}}$$

$$\bullet \quad \|\Delta x\| = \|(A^T A)^{-1} (\Delta A)^T r - A^+ \Delta Ax\| \\ \leq \|(A^T A)^{-1}\| \cdot \|\Delta A\| \cdot \|r\| + \|A^+\| \cdot \|\Delta A\| \cdot \|x\|$$

- Cond. number is a func. of $\text{cond}(A)$ and b
- $Pb \approx 0$ or $\theta \approx 90^\circ$ is highly sensitive

$$\rightarrow \boxed{\frac{\|\Delta x\|}{\|\hat{x}\|} \leq \left([\text{cond}(A)]^2 \frac{\|r\|}{\|Ax\|} + \text{cond}(A) \right) \frac{\|\Delta A\|}{\|A\|} \\ = \left([\text{cond}(A)]^2 \tan \theta + \text{cond}(A) \right) \frac{\|\Delta A\|}{\|A\|}}$$

2.2 Solving $A^T Ax = A^T b$ with an Augmented Matrix

$$\begin{matrix} r + Ax = b \\ A^T r = 0 \end{matrix} \Rightarrow \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} \alpha I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r/\alpha \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

- Solvable with LU Decomp or Symm. Pos. Def. Methods
- α "controls the relative weights of the two subsystems in choosing pivots from either"
- $\alpha = \max a_{ij}/1000$ (rule of thumb)
- MATLAB uses it for large, sparse systems

2.3 QR Decomposition

Motivation: $Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \rightarrow Q^T A x + Q^T r = Q^T b \rightarrow \begin{pmatrix} Rx \\ 0 \end{pmatrix} + \begin{pmatrix} r'_1 \\ r'_2 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \end{pmatrix} \rightarrow \begin{matrix} \|r'\|^2 = \|b'_1 - Rx\|^2 + \|b'_2\|^2 \\ \downarrow \\ Rx = b'_1, r' = \begin{pmatrix} 0 \\ b'_2 \end{pmatrix} \text{ (solve with back-sub)} \end{matrix}$

Orthogonal Matrix, Q

$$Q^T Q = Q Q^T = I$$

QR Factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Reduced QR Factorization

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} Q_{\parallel} & Q_{\perp} \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_{\parallel} R$$

1. Q^T is a $\text{span}(A)$ Plane Rotation through \mathbb{R}^m to $\text{span}([R \ 0]^T)$

2-norm Preserved (Q is a rotation/reflection)

- $\|Qv\|^2 = \langle v | Q^T Q v \rangle = \|v\|^2$
- $\|Q^T v\|^2 = \langle v | Q Q^T v \rangle = \|v\|^2$

- $Q^T = H_n \dots H_1$
- $H_i^T H_i = H_i H_i^T = I$
- $A = [a_1 \dots a_n]$
- $I_n = [e_1 \dots e_n]$

$$H_1 a_1 = \alpha_1 e_1 \quad (\|a_1\| = |\alpha_1|)$$

$$H_i \dots H_1 a_i = \sum_j^i c_j e_j = H_n \dots H_1 a_i$$

$$(\|a_i\|^2 = |\alpha_1|^2 = \sum_j^i c_j^2)$$

$$\langle r | a_i \rangle = 0 \quad (1 \leq i \leq n)$$

$$\langle H_i \dots H_1 r | e_j \rangle = 0 \quad (1 \leq j \leq i)$$

→ $Q^T A$ rotates A until the column vectors are aligned with certain axes described above

2. A is a Lin. Sum of Q_{\parallel} 's Orthogonal Column Vectors Given by R

$$\{Q_{\parallel} = Q_{m \times n} \mid \text{span}(Q_{\parallel}) = \text{span}(A)\}$$

$$\rightarrow Q^+ = (Q^T Q)^{-1} Q^T = Q^T$$

$$\rightarrow P = Q_{\parallel} Q_{\parallel}^T$$

$$\rightarrow Q_{\parallel}^T A x = Q_{\parallel}^T P b = \cancel{Q_{\parallel}^T Q_{\parallel}} Q_{\parallel}^T b$$

$$= Q_{\parallel}^T b \quad (\text{System of Orthogonal Equations?})$$

$$A = Q_{\parallel} R = \begin{pmatrix} | & | & | \\ \vec{q}_1 & \dots & \vec{q}_n \\ | & | & | \end{pmatrix} \begin{pmatrix} r_{11} & \dots & r_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{nn} \end{pmatrix} = \begin{pmatrix} | & | & | \\ \vec{a}_1 & \dots & \vec{a}_n \\ | & | & | \end{pmatrix}$$

- $\vec{a}_j = \sum_i^j r_{ij} \cdot \vec{q}_i$

→ R transforms the Q_{\parallel} column vectors about $\text{span}(A)$, an \mathbb{R}^n plane, until they equal the column vectors of A

2.3.1 Householder Transformation/Elementary Reflector, H

$$\begin{aligned}
 H\vec{a}_1 &= \alpha_1 \vec{e}_1 \quad \begin{array}{l} \|\vec{a}_1\| = |\alpha_1| \\ \text{(rotation)} \end{array} \\
 &= \boxed{\vec{a}_1 - 2\hat{v}(\hat{v} \cdot \vec{a}_1)} \quad \begin{array}{l} [v_\perp \text{ bisects } \theta(a_1, e_1)] \end{array} \rightarrow \boxed{H = I - 2vv^T = I - \frac{2vv^T}{v^T v}} \quad \bullet \ H = H^T = H^{-1} \\
 &\quad \text{(symmetric and orthogonal)}
 \end{aligned}$$

$$\bullet \ \alpha_1 e_1 = a_1 - (2v_1) \frac{v_1 \cdot a_1}{v_1 \cdot v_1} \Rightarrow v_1 = (a_1 - \alpha e_1) \frac{v_1 \cdot v_1}{2v_1 \cdot a_1} \quad \text{(magnitude doesn't matter)}$$

$$\rightarrow \boxed{v_1 = (a_1 - \alpha e_1)}$$

$$\alpha_1 = \pm \|\vec{a}_1\| \rightarrow \boxed{\alpha_i = -\text{sign}(a_i) \|\vec{a}_i\|} \quad \text{(avoid "cancellation" in finite-calc. of } v \text{ above)}$$

$$H_j \dots H_1 a_i = a_i^j \rightarrow \boxed{v_{j+1} = \begin{pmatrix} 0 \\ \vdots \\ (a_i^j)_i \\ \vdots \\ (a_i^j)_m \end{pmatrix} - \alpha_i e_i} \quad \begin{array}{l} \bullet \text{ Store } v_i \text{ and } R \text{ into } A \text{ and an extra } n\text{-vector.} \\ \bullet \text{ } Q \text{ and } H \text{ can be computed if needed.} \\ \bullet \text{ When column } i \text{ is completed, row } i \text{ is too.} \end{array}$$

2.3.2 Givens Rotation, G

$$\begin{aligned}
 \boxed{G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}} &\rightarrow Gx = G \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \pm \begin{pmatrix} \|a\| \\ 0 \end{pmatrix} \quad \begin{array}{l} \bullet \text{ creates 0's one at a time} \\ \bullet \text{ useful for sparse matrices} \\ \bullet \text{ When column } i \text{ is completed, row } i \text{ is too.} \end{array} \\
 &\rightarrow \boxed{c = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}}
 \end{aligned}$$

Avoid squaring any number $\gg 1$ to prevent overflow/underflow

$$\begin{array}{l} \bullet \ t = \frac{a_2}{a_1} < 1 \rightarrow c = \frac{1}{\sqrt{1+t^2}}, \quad s = c \cdot t \\ \bullet \ \tau = \frac{a_1}{a_2} < 1 \rightarrow s = \frac{1}{\sqrt{1+\tau^2}}, \quad c = s \cdot \tau \end{array}$$

2.3.3 Gram-Schmidt Orthogonalization

$$Q_{||}^T = \begin{pmatrix} \hat{q}_1 : q_1 = a_1 \\ \hat{q}_2 : q_2 = a_2 - \hat{q}_1(\hat{q}_1 \cdot a_2) \\ \hat{q}_3 : q_3 = a_3 - \hat{q}_1(\hat{q}_1 \cdot a_3) - \hat{q}_2(\hat{q}_2 \cdot a_3) \\ \vdots \\ \hat{q}_n : q_n = a_n - \sum_j^n \hat{q}_j(\hat{q}_j \cdot a_n) \end{pmatrix}, \quad R = \begin{pmatrix} \|a_1\| & \hat{q}_1 \cdot a_2 & \hat{q}_1 \cdot a_3 & \dots & \hat{q}_1 \cdot a_n \\ 0 & \|a_2\| & \hat{q}_2 \cdot a_3 & \dots & \hat{q}_2 \cdot a_n \\ 0 & 0 & \|a_3\| & \dots & \hat{q}_3 \cdot a_n \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \|a_n\| \end{pmatrix}$$

Classical, Column Oriented: Find \hat{q}_i , then solve for $\widehat{q_{i+1}}$, and continue up to \hat{q}_n .

- For a program, obviously a_k can be replaced by q_k , so less storage is needed.
- Cancellation that causes loss of orthogonality occurs more when ill-conditioned.
- As a result, performing $Q_{\parallel}^T b = b'_1$ isn't always best.
- Can't column-pivot, since that depends on rows being completed first.

Modified, Row Oriented: Let $q_i^{[k]} = a_i - \sum_j^k \hat{q}_j (\hat{q}_j \cdot a_i)$. For all $1 \leq i \leq n$, solve for $q_i^{[k]}$ starting first at $k = 1$, then continue until $k = n$.

- Allows for column pivoting since rows are completed first.
- Cancellation, though still present, is less severe.

Augmented Matrix:

$$(A | b) = (Q_{\parallel} | q_{n+1}) \begin{pmatrix} R & b'_1 \\ 0 & \rho \end{pmatrix}$$

$$\left(\begin{array}{c|c|c|c|c} | & | & | & | & | \\ a_1 & \dots & a_n & b & \\ | & | & | & | & | \end{array} \right) = \left(\begin{array}{c|c|c|c|c} | & | & | & | & | \\ \hat{q}_1 & \dots & \hat{q}_n & q_{n+1} & \\ | & | & | & | & | \end{array} \right) \left(\begin{array}{c|c|c|c|c} r_{11} & \dots & r_{1n} & | & \\ 0 & \ddots & \vdots & b'_1 & \\ \vdots & \ddots & r_{nn} & | & \\ 0 & \dots & 0 & \rho & \end{array} \right)$$

- Use Gram-Schmidt QR on this, then solve $Rx = b'_1$
- This method is preferred numerically to reduce cancelling effects
- Text didn't recommend what q_{+1} or ρ should be.
- ρ or $(q_{n+1})_i$ looks like it should be 0.
- Idk, not much explained.

Reorthogonalizing: Repeating procedure to straighten vectors (usually not needed)

2.3.4 Factorization with Column-Pivoting

- Column with largest norm is pivoted to the current column i to be reduced, and current row i is completed too.
- Choose the next pivoting column based on norms of the smaller columns from remaining uncompleted submatrix.
- Repeat until the end (rank might be n) or if the max norm is smaller than some tolerance (rank might be $k < n$)
- Pivoting avoids working with 0's on the diag.

2.3.5 Rank Deficiency (or Other) Case

If $\text{rank}(A) = k < n$:

$$\begin{pmatrix} Q^T A P \end{pmatrix} \begin{pmatrix} P^T x \end{pmatrix} = Q^T b$$

$$\begin{pmatrix} R & S \\ 0 & 0' \end{pmatrix} \begin{pmatrix} z \\ 0 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \end{pmatrix}$$

- $0'$ is approx. 0 since the remaining norms are too small.
- $R = R_{k \times k}$
- S is the remaining columns after R is completed.
- There are multiple solutions for x .

- For a quick solution, $Rz = b'_1, \quad x = P \begin{pmatrix} z \\ 0 \end{pmatrix}$
- For the minimized-norm solution with the smallest $\|x\|$, S must be annihilated.
- For another method or if underdetermined ($m < n$), something like SVD Decomposition can be used.

2.4 Singular Value Decomposition (SVD)

$$\begin{aligned}
A &= \boxed{U \Sigma V^T} = \left(\begin{array}{c|c|c|c|c} & & & & \\ \hline & & & & \\ \hline u_1 & \dots & u_k & .. & u_m \\ \hline & & & & \\ \hline \end{array} \right) \left(\begin{array}{ccccc} \sigma_1 & 0 & \dots & 0 & \\ \hline 0 & \sigma_2 & \ddots & \vdots & \\ \hline 0 & 0 & \ddots & 0 & \\ \hline \vdots & \vdots & \ddots & \sigma_n & \\ \hline 0 & 0 & \dots & 0 & \\ \hline \vdots & \vdots & & \vdots & \\ \hline 0 & 0 & \dots & 0 & \end{array} \right) \left(\begin{array}{c} \text{---} v_1 \text{---} \\ \vdots \\ \text{---} v_k \text{---} \\ \vdots \\ \text{---} v_n \text{---} \end{array} \right) \\
&= \boxed{U_{\parallel} \Sigma_1 V^T} = (U_{\parallel} \quad U_{\perp}) \begin{pmatrix} \Sigma_1 \\ \mathbf{0} \end{pmatrix} \begin{pmatrix} V_{0\perp}^T \\ V_{0\parallel}^T \end{pmatrix} = \boxed{\sum_i^n \sigma_i \cdot u_i v_i^T}
\end{aligned}$$

- Underdetermined, $m < n$ is possible too.
- Analogous to Gaussian-Jordan Diagonalization method.
- U and V are orthogonal; u_i and v_i are the respective “left” and “right” singular vectors.
- Usually, the singular values are ordered such that $\sigma_1 \geq \sigma_2 \geq \dots$
- $\forall (k < i), \sigma_i = 0 \Rightarrow \text{rank}(A) = k < n$
- $U_{\parallel} = U_{m \times k} : \text{span}(U_{\parallel}) = \text{span}(A), \text{span}(U_{\perp}) = \text{span}(A)^{\perp}$
- $V_{0\perp} = V_{n \times k} : \text{span}(V_{0\parallel}) = \text{null}(A), \text{span}(V_{0\perp}) = \text{null}(A)^{\perp} \quad \text{null}(A) = \{x : Ax = 0\}$

Pseudoinverse:

$$\begin{aligned} A^+ &\equiv V \Sigma^+ U^T \\ \Sigma^+ &\equiv \left[\Sigma^T \quad \text{and} \quad \sigma_i \rightarrow 1/\sigma_i \quad \forall (\sigma_i \neq 0) \right] \end{aligned}$$

- $Ax + r = b \rightarrow$

$$\boxed{x = A^+b = (V\Sigma^+U^T)b}$$

- $$x_{\min} = \sum_{\sigma_i \neq 0} \frac{u_i \cdot b}{\sigma_i} v_i$$
 useful for ill-conditioned or rank deficient since small σ can be dropped.

2.4.1 Other uses

Euclidean 2-norm : $\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_{\max}$

Euclid. Cond. Num. : $\text{cond}_2(A) = \sigma_{\max}/\sigma_{\min}$

Lower Rank Approx. : $A \approx A_k = \sum_i^k \sigma_i (u_i v_i^T)$

- Closest rank= k matrix to A in the Frobenius norm.
- Frobenius Norm = Euclid. Norm for a “vector” in \mathbb{R}^{mn} .

Total Least Squares : $[A \mid y]_{m \times (n+1)} = U \Sigma V_{(n+1) \times (n+1)}^T$

$$\text{rank}([\hat{A} \mid y]) \leq n \rightarrow \sigma_{n+1} = 0 \rightarrow \hat{A} \cdot v_{n+1} = 0$$

$$\left[\widehat{A} \mid y\right] \cdot \begin{bmatrix} x \\ -1 \end{bmatrix} = 0 \rightarrow \begin{bmatrix} x \\ -1 \end{bmatrix} \propto v_{n+1} = \begin{bmatrix} \vec{\nu}_n \\ \nu_{n+1} \end{bmatrix} \rightarrow \boxed{x = \frac{\vec{\nu}_n}{-\nu_{n+1}}}$$

- \hat{A} is an A with uncertainty, like how y normally is.

2.5 Complexity

Normal, Cholesky

- $A^T A = A'$ costs $\frac{mn^2}{2}$
- $A' = LL^T$ costs $\frac{n^3}{6}$
- Rel. Err. $\propto [\text{cond}(A)]^2$
- Bad if $\text{cond}(A) \approx 1/\sqrt{\epsilon_{\text{mach}}}$

Householder

- $Q^T A = R$ costs $mn^2 - \frac{n^3}{3}$
- Rel. Err. $\propto [\text{cond}(A)]^2 \|r\|_2 + \text{cond}(A)$
- Bad if $\text{cond}(A) \approx 1/\epsilon_{\text{mach}}$
- More accurate than Cholesky and broadly applicable
- Usable for rank deficient or nearly rank-deficient

Givens

- The normal implementation needs 50% more work than Householder.
- A more complex implementation makes it comparable to Householder.
- Useful if matrix is sparse or zeros need to be maintained.

SVD

- Most expensive cost at $\propto mn^2 + n^3$, perhaps 4-10 times or more.
- Robust and reliable.

3 Matrix Information

Orthogonal : $QQ^T = Q^TQ = I$

Unitary : $UU^\dagger = U^\dagger U = I$

$$\Leftrightarrow U = e^{iH} = (U_h)e^{iD_h}(U_h)^\dagger$$

Normal : $AA^\dagger = A^\dagger A \Leftrightarrow A = UDU^\dagger$

Symmetric : $S = S^T = QDQ^T$ (D is real)

Hermitian : $H = H^\dagger = UDU^\dagger$ (D is real)

Similar : $A(Ty) = \lambda(Ty)$

$$\rightarrow A \sim B = T^{-1}AT$$

Diagonalize : $T^{-1}AT = D$ $\begin{pmatrix} A \text{ is nondefective} \\ T \text{ is nonsingular} \\ D \text{ is diag.} \end{pmatrix}$

Hessenberg : Triang. but from any diag.

Jordan Form : Nonsing. trans. into a near
diag. w/ entries in diag. above

• $Ax = \lambda x$, $\det(A) \neq 0 \Rightarrow A^{-1}x = (1/\lambda)x$

• Shifting : $(A - \sigma I)x = (\lambda - \sigma)x$

• Simple: Normal [Algebraic] Mult. of 1

• Defective: Geo. Mult. < Alg. Mult.
(eig. vec. #)

Invariant Subspace: $\{\mathcal{S} : (A\mathcal{S} \subseteq \mathcal{S}) \equiv (\forall x \in \mathcal{S} \Rightarrow Ax \in \mathcal{S})\}$

Schur Form

Unitary: $\boxed{T^\dagger AT = R}$ $\begin{matrix} (T \text{ is unitary}) \\ (R \text{ is upp. triang.}) \end{matrix}$

Real : $\boxed{Q^T AQ = R}$ $\begin{matrix} (Q \text{ is ortho.}) \\ (R \text{ is block upp. triang.}) \end{matrix}$
(only for real matrix)

• $A\vec{x}_i = r_{ii}\vec{x}_i$

• $0 = (R - r_{ii}I) \vec{x}_i$

$$= \begin{pmatrix} R_{11} - r_{ii}I & \vec{u} & R_{13} \\ 0 & 0 & \vec{v}^T \\ \mathcal{O} & 0 & R_{33} - r_{ii}I \end{pmatrix} \vec{x}_i$$

$$\Downarrow$$

$$\vec{x}_i = \begin{pmatrix} \vec{y} \\ -1 \\ 0 \end{pmatrix} : (R_{11} - r_{ii}I) \vec{y} = \vec{u}$$

(nonsingular)

• "Schur form of a real matrix will have complex entries if the matrix has any complex eigenvalues"

• $R = \begin{pmatrix} R_{11} & \cdots & R_{1p} \\ \mathcal{O} & \ddots & \vdots \\ \mathcal{O} & \mathcal{O} & R_{pp} \end{pmatrix}$

• $\lambda_i(A) = \lambda(R_{ii})$

• $R_{ii} = \begin{cases} R_{1 \times 1} & (\text{real eigenvalue of } A) \\ R_{2 \times 2} & (\text{complex eigenvalue pairs of } A) \end{cases}$

• All other entries are real

• Reducible: If $PAP^T = R$ (P is permu.)

Block Upper Triangular Transformation

• $X_{\parallel} = (\vec{x}_1 \dots \vec{x}_p)$, $Ax_i \in \text{span}(x_i) = \mathcal{S}$

$$\Rightarrow AX_{\parallel} = X_{\parallel}B$$

• $I_n = X^{-1}X = \begin{pmatrix} Y_{\parallel} \\ Y_{\perp} \end{pmatrix} (X_{\parallel} \ X_{\perp}) = \begin{pmatrix} Y_{\parallel}X_{\parallel} & Y_{\parallel}X_{\perp} \\ Y_{\perp}X_{\parallel} & Y_{\perp}X_{\perp} \end{pmatrix} = \begin{pmatrix} I_p & \mathcal{O} \\ \mathcal{O} & I_{n-p} \end{pmatrix}$

• $X^{-1}AX = \begin{pmatrix} Y_{\parallel}AX_{\parallel} & Y_{\parallel}AX_{\perp} \\ Y_{\perp}AX_{\parallel} & Y_{\perp}AX_{\perp} \end{pmatrix} = \begin{pmatrix} Y_{\parallel}X_{\parallel}B & Y_{\parallel}AX_{\perp} \\ Y_{\perp}X_{\parallel}B & Y_{\perp}AX_{\perp} \end{pmatrix} = \begin{pmatrix} B & Y_{\parallel}AX_{\perp} \\ \mathcal{O} & Y_{\perp}AX_{\perp} \end{pmatrix} = R$

4 Eigenvalue Equation, $Ax = \lambda x$

4.1 Error Bound and Conditioning

$$A + \Delta A = Q(D + \Delta D)Q^{-1}$$

$$(A + \Delta A)(x + \Delta x) = (\lambda + \Delta \lambda)(x + \Delta x)$$

$$\bullet v = (\Delta \lambda I - D)^{-1}(\Delta D)v$$

$$\bullet Ax = \lambda x, \quad y^H A = \lambda y^H$$

$$\bullet \|(\Delta \lambda I - D)^{-1}\|_2^{-1} \leq \|\Delta D\|_2$$

$$\bullet \lambda \text{ is simple} \Rightarrow y^H x \neq 0 \quad (?)$$

$$|\Delta \lambda - \lambda_i| \leq \|Q(\Delta A)Q^{-1}\|_2$$

$$\bullet y^H A x + y^H A \Delta x + y^H (\Delta A)x + y^H (\Delta A)\Delta x \\ \approx y^H \lambda x + y^H \lambda \Delta x + y^H (\Delta \lambda)x + y^H (\Delta \lambda)\Delta x$$

$$\rightarrow \boxed{|\Delta \lambda - \lambda_i| \leq \text{cond}(Q) \|\Delta A\|_2}$$

$$\rightarrow \boxed{|\Delta \lambda| \lesssim \frac{\|y\|_2 \cdot \|x\|_2}{|y^H x|} \|\Delta A\|_2 = \frac{1}{\cos \theta} \|\Delta A\|_2}$$

- $AA^\dagger = A^\dagger A \rightarrow \text{cond}(A) = 1$
- Non-simple (multiple) eigenvalue is complicated:
- allows $y^H x = 0$, depends on eigenvalue spacings, vector angles, etc.
- Balancing (diagonal rescaling) can improve conditioning

4.2 QR Iteration

4.2.1 Intro

Power Iteration:

$$A^k x_0 = A^k \sum_{i=1}^n c_i v_i = \sum_{i=1}^n c_i (A^k v_i) = \sum_{i=1}^n c_i (\lambda_i^k v_i) \\ = \lambda_1^k \left[\sum_{i=1}^j c_i v_i + \sum_{i=j+1}^n c_i \left(\frac{\lambda_i}{\lambda_1} \right)^k v_i \right] \quad \begin{matrix} (\lambda_1 = \dots = \lambda_j) \\ (\lambda_1 \geq \dots \geq \lambda_n) \end{matrix}$$

$$\boxed{(A^k x_0 = x_k) \approx \lambda_1^k \sum_i^j c_i v_i = \lambda_1^k (c_1 v_1) \text{ if } j = 1}$$

Rayleigh Quotient (for shifts):

$$\left[x\lambda + r = Ax \right] \Rightarrow \left[x^H x \lambda = x^H Ax \right] \Rightarrow \boxed{\lambda = \frac{\langle x | Ax \rangle}{\langle x | x \rangle}}$$

•

Inverse [Power] Iteration:

$$\left[(A^{-1})^k x_0 = x_k \right] \Rightarrow \boxed{(Ax_{k+1} = x_k) \rightarrow \left(\lambda_1 \rightarrow \frac{1}{\lambda_1} \right)}$$

•

Inverse Iteration W/ Shifts:

$$\boxed{\left[(A - \sigma I)^k x_0 = x_k \right] \Rightarrow \left[(\lambda_1 - \sigma) \rightarrow \frac{1}{\lambda_1 - \sigma} \right]}$$

•

Deflation (concept):

$$T^{-1} A_k T = \begin{pmatrix} \lambda_1 & b^T \\ 0 & A_{k+1} \end{pmatrix}$$

•

Simultaneous/Subspace [Power] Iteration:

$$X_0 = X_{n \times p} \quad , \quad \text{rank}(X_0) = p$$

$$\rightarrow X_{k+1} = AX_k = A^{k+1} X_0$$

$$\rightarrow \lim_{k \rightarrow \infty} \text{span}(X_k) = \text{span}(A)$$

•

Orthogonal [Subspace] Iteration:

$$X = X_{n \times p} \quad , \quad \text{rank}(X) = p$$

$$\rightarrow X_k = Q_k R_k \quad (Q = Q_{n \times p})$$

$$\rightarrow X_{k+1} = A Q_k \begin{bmatrix} = Q_k B_{p \times p} \end{bmatrix}$$

•

4.2.2 QR Iteration

$$\begin{aligned}
X_{k+1} &\equiv AQ_k \quad (\text{use } X_0 = I \rightarrow X_1 = A) \\
Q_{k+1}R_{k+1} &= Q_k(Q_k^H A Q_k) \\
\rightarrow \boxed{A_k &\equiv Q_k^H X_{k+1} = Q_k^H A Q_k} \\
&= \begin{cases} (Q_k^H Q_{k+1}) R_{k+1} &\equiv \boxed{Q_{k+1}^{(A)} R_{k+1}} \\ Q_k^H (Q_k R_k Q_k^H) Q_k &= R_k Q_k^{(A)} \end{cases} \\
\rightarrow \boxed{A_{k+1} &= R_{k+1} Q_{k+1}^{(A)}}
\end{aligned}$$

Inverse QR Iteration

$$\begin{aligned}
\left(A_k^{-H} &= Q_{k+1}^{(A)} R_{k+1}^{-H} \right), \quad \left(A_{k+1}^{-H} = R_{k+1}^{-H} Q_{k+1}^{(A)} \right) \\
\rightarrow \left(A^{-H} \right)^k &= Q_k R_k^{-H}
\end{aligned}$$

QR Iteration with Shifts

$$\begin{aligned}
\text{Rayleigh Quotient Shift: } \sigma &= \frac{\langle q_p | A | q_p \rangle}{\langle q_p | q_p \rangle} = \langle q_p | A | q_p \rangle \\
\rightarrow Q_{k+1}^{(A)} R_{k+1} &= A_k - \sigma I \\
\rightarrow R_{k+1} Q_{k+1}^{(A)} &= A_{k+1} + \sigma I \\
\rightarrow \text{Deflation: } \lim_{k \rightarrow \infty} A_k &= \begin{pmatrix} A'_{k+1} & b^T \\ 0 & \lambda_p \end{pmatrix}
\end{aligned}$$

Hessenberg QR Iteration

$$\begin{aligned}
\text{Rayleigh Quotient Shift: } \sigma &= \frac{\langle q_p | A | q_p \rangle}{\langle q_p | q_p \rangle} = \langle q_p | A | q_p \rangle \\
\rightarrow Q_{k+1}^{(A)} R_{k+1} &= A_k - \sigma I \\
\rightarrow R_{k+1} Q_{k+1}^{(A)} &= A_{k+1} + \sigma I
\end{aligned}$$