1 System of Linear Equations, $Ax = b$

1.1 p-Norm and Condition Number

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

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

$$\infty\text{-Norm}: \quad \|\vec{x}\|_\infty \;=\; \max |x_i|$$

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

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

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

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

- $\|AB\| \;\leq\; \|A\| \cdot \|B\|$
- $\|Ax\| \;\leq\; \|A\| \cdot \|x\|$

} For p-norms (not necessarily in general)

Func./Vec. Condition Number (Solve for y):

$$[\text{rel.}]\ \text{cond}_f(x) = \left|\frac{\Delta y/y}{\Delta x/x}\right| = \left|\frac{\text{Rel. Forward Err.}}{\text{Rel. Backward Err.}}\right|$$

$$= \boxed{\left|\frac{[f(\hat{x})-f(x)]/f(x)}{[\hat{x}-x]/x}\right|} \quad \text{(vector)}$$

$$= \boxed{\left|\frac{xf'(x)}{f(x)}\right|} \quad \text{(func.)}$$

$$[\text{abs.}]\ \text{cond}_f(\hat{x}) = \left|\frac{\Delta y}{\Delta x}\right| = \left|\frac{\text{Abs. Forward Err.}}{\text{Abs. Backward Err.}}\right|$$

$$\left(\begin{smallmatrix}\text{use when } f(x)=0\\ \text{or } x=0\end{smallmatrix}\right) = \boxed{|f'(\hat{x})|}$$

Matrix Condition Number:

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

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

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

  $\rightarrow \frac{\|z\|}{\|y\|} \;\leq\; \max \frac{\|z\|}{\|y\|} \overset{?}{=} \|A^{-1}\|$ (optimize)

Inverse Condition Number (Solve for x):

$$\boxed{\text{cond}_{f^{-1}}(x) \;=\; 1/\text{cond}_f(x)}$$

1.2 Error Bounds and Residuals

Error Bound: $\boxed{\frac{\|\hat{x}-x\|}{\|x\|} \lessapprox \text{cond}(A)\ \epsilon_{\text{mach}}}$ → 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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)

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

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

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

- 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. ($a_{jj} > \sum_{i, i \neq j} a_{ij}$)
- A can be singular

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

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

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

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

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

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

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

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

(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)

$$\boxed{\begin{aligned} &A = MD \qquad \begin{matrix}(M \text{ is elem. elim.}) \\ (D \text{ is diag.})\end{matrix} \\ &M = (\dots M_2 P_2 M_1 P_1)^{-1} \end{aligned}}$$

$$\begin{aligned} \{\dots\} b &= (\dots M_2 P_2 M_1 P_1) A x \\ M^{-1} b &= (P_1^T M_1^{-1} P_2^T M_2^{-1} \dots)^{-1} A x \\ &= M^{-1}(MD)x = y \end{aligned}$$

$$\boxed{M^{-1} b = y \ , \quad y = Dx \qquad \text{(division)}}$$

- 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}

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

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

$$= \left[\begin{array}{cc|ccc} 1 & 0 & a'_{11} & \cdots & \\ 0 & 1 & \vdots & & a'_{nn} \end{array}\right]$$

1.5 Symmetric Matrices

Positive Definite: $\boxed{x^T A x > 0} \Leftrightarrow \boxed{\lambda > 0}$ $(Ax = \lambda x)$

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

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

- 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 : $\boxed{PAP^T = LDL^T}$
- 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

$$\tilde{A}\tilde{x} = b = (A - uv^T)\tilde{x}$$

$$\rightarrow \tilde{x} = \tilde{A}^{-1}b$$

$$\boxed{\begin{aligned} \tilde{A}^{-1} &= (A - uv^T)^{-1} = A^{-1} + \tfrac{A^{-1}u}{1 - v^T(A^{-1}u)}\, v^T A^{-1} \\ \tilde{A}^{-1}b &= \tilde{x} = (A^{-1}b) + \tfrac{A^{-1}u}{1 - v^T(A^{-1}u)}\, v^T(A^{-1}b) \\ &\qquad\qquad x + \tfrac{y}{1 - v^T y}\, v^T x \end{aligned}}$$

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

$$\begin{aligned}
\text{Explicit Inversion :} &\quad \begin{matrix} LUA^{-1} = I \\ D^{-1}M^{-1}I = A^{-1} \end{matrix} \rightarrow \mathcal{O}(n^3) \quad , \quad A^{-1}b = x \rightarrow \mathcal{O}(n^2) \\
\text{Gaussian Elimination :} &\quad A = LU \rightarrow \mathcal{O}(n^3/3) \quad , \quad LUx = b \rightarrow \mathcal{O}(n^2) \\
\text{Gaussian-Jordan :} &\quad A = MD \rightarrow \mathcal{O}(n^3/2) \quad , \quad MDx = b \rightarrow \mathcal{O}(n) \\
\text{Symmetric :} &\quad \begin{matrix} A = LL^T \\ PAP^T = LDL^T \end{matrix} \rightarrow \mathcal{O}(n^3/6) \quad , \quad LL^T x = b \rightarrow \mathcal{O}(n^2) \\
\text{Banded :} &\quad A_\beta = LU \rightarrow \mathcal{O}(\beta^2 n) \quad , \quad LUx = b \rightarrow \mathcal{O}(\beta n) \\
\text{Sherman-Woodbury :} &\quad \tilde{A} = A - uv^T \rightarrow \mathcal{O}(n^2) \quad , \quad \tilde{x} = \tilde{A}b \rightarrow \mathcal{O}(n^2)
\end{aligned}$$

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 systmes, 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
- $\boxed{\begin{array}{c}\text{rank}(A) = n \\ \text{(full column rank)}\end{array}}$ $\Rightarrow$ $A(x_1 - x_2) = 0$ (unique x)
  $(x_1 - x_2) = 0 \rightarrow x_1 = x_2$

Example - Vandermonde Matrix, A:

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

Decompose b :

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

Projector of A, P

Projector : $P^2 = P \rightarrow PA = A$ (Idempotent) (Projector of A)

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

Minimize residual, r:

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

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

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

Symmetric : $(A^T A)^T = A^T A$

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

Nonsingular : $A^T Ax = 0$
$\rightarrow \|Ax\|^2 = 0 = Ax$
$\rightarrow (x = 0)$

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

Pseudoinverse, A^+

$$\boxed{\begin{aligned} x &= (A^T A)^{-1} A^T b \\ &\equiv A^+ b \end{aligned}} \rightarrow \boxed{\begin{aligned} A^+ &\equiv (A^T A)^{-1} A^T \\ A^+ A &= I \end{aligned}}$$

Ortho. Proj., P

$$\boxed{\begin{aligned} Ax &= A(A^T A)^{-1} A^T b \\ &= Pb \end{aligned}} \rightarrow \boxed{\begin{aligned} P &= A(A^T A)^{-1} A^T \\ &= AA^+ \end{aligned}}$$

System of Normal Equations Issues:

- Info can be lost forming A^TA, e.g, $A = \begin{pmatrix} 1 & 0 \\ \epsilon & 0 \\ 0 & \epsilon \end{pmatrix} \rightarrow A^TA = \begin{pmatrix} 1+\epsilon^2 & 1 \\ 1 & 1+\epsilon^2 \end{pmatrix} \approx \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ (singular)
- System of Normal Equations: $\boxed{\text{cond}(A^TA) = [\text{cond}(A)]^2}$

2.1 Error Bounds and Residuals

Error Bound: $\boxed{\dfrac{\|\Delta x\|}{\|x\|} \lessapprox \text{cond}(A)\ \epsilon_{\text{mach}}}$ $\rightarrow$ 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:

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

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

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

- $\|\Delta x\| \ \leq\ \|A^+\| \cdot \|\Delta b\|$

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

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

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

- $\cancel{A^TAx} + A^T\Delta Ax + (\Delta A)^TAx + \cancel{(\Delta A)^T\Delta Ax} = \cancel{A^Tb} + (\Delta A)^Tb$
  $+ A^TA\Delta x + \cancel{A^T\Delta A\Delta x} + \cancel{(\Delta A)^TA\Delta x} + \cancel{(\Delta A)^T\Delta A\Delta x}$
- $\|\Delta x\| = \|(A^TA)^{-1}(\Delta A)^Tr - A^+\Delta Ax\|$
  $\leq \|(A^TA)^{-1}\| \cdot \|\Delta A\| \cdot \|r\| + \|A^+\| \cdot \|\Delta A\| \cdot \|x\|$

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

2.2 Solving $A^TAx = A^Tb$ with an Augmented Matrix

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

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

2.3 QR Decomposition

Motivation: $Q^TA = \begin{pmatrix} R \\ 0 \end{pmatrix}$ (R is upp. triag.) $\rightarrow$ $Q^TAx + Q^Tr = Q^Tb$, $\begin{pmatrix} Rx \\ 0 \end{pmatrix} + \begin{pmatrix} r'_1 \\ r'_2 \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \end{pmatrix}$ $\rightarrow$ $\|r'\|^2 = \cancel{\|b'_1 - Rx\|^2} + \|b'_2\|^2$

$\downarrow$

$\boxed{Rx = b'_1}$, $r' = \begin{pmatrix} 0 \\ b'_2 \end{pmatrix}$ (solve with back-sub)

Orthogonal Matrix, Q

$\boxed{Q^TQ = QQ^T = I}$

QR Factorization

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

Reduced QR Factorization

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

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

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

- $\|Qv\|^2 = \langle v|Q^TQv\rangle = \|v\|^2$
  $\|Q^Tv\|^2 = \langle v|QQ^Tv\rangle = \|v\|^2$

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

$$H_1a_1 = \alpha_1e_1 \quad (\|a_1\| = |\alpha_1|)$$

$$H_i \dots H_1a_i = \sum_j^i c_je_j = H_n \dots H_1a_i \quad (\|a_i\|^2 = |\alpha_1|^2 = \sum_j^i c_j^2)$$

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

$$\langle H_i \dots H_1r|e_j\rangle = 0 \quad (1 \le j \le i)$$

$\rightarrow$ Q^TA 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

$\left\{Q_\parallel = Q_{m\times n} \;\middle|\; \text{span}(Q_\parallel) = \text{span}(A)\right\}$

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

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

$\rightarrow$ $Q_\parallel^TAx = Q_\parallel^TPb = \cancel{Q_\parallel^TQ_\parallel}Q_\parallel^Tb$

$= Q_\parallel^Tb$ (System of Orthogonal Equations?)

$$A = Q_\parallel R = \begin{pmatrix} | & | & | \\ \vec{q_1} & \dots & \vec{q_n} \\ | & | & | \end{pmatrix}\begin{pmatrix} r_{11} & \dots & r_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{nn} \end{pmatrix} = \begin{pmatrix} | & | & | \\ \vec{a_1} & \dots & \vec{a_n} \\ | & | & | \end{pmatrix}$$

- $\vec{a_j} = \sum_i^j r_{ij} \cdot \vec{q_i}$

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

2.3.1 Householder Transformation/Elementary Reflector, H

$$H\vec{a_1} = \alpha_1\vec{e_1} \quad \overset{\|a_1\| = |\alpha_1|}{\text{(rotation)}}$$

$$= \boxed{\vec{a_1} - 2\hat{v}(\hat{v}\cdot\vec{a_1})} \quad \underset{[v_\perp \text{ bisects } \theta(a_1, e_1)]}{}$$

$$\rightarrow \quad \boxed{H = I - \hat{v}\hat{v}^T = I - \frac{2vv^T}{v^Tv}}$$

- $H = H^T = H^{-1}$ (symmetric and orthogonal)

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

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

$$\alpha_1 = \pm\|a_1\| \quad \rightarrow \quad \boxed{\alpha_i = -\text{sign}(a_i)\|a_i\|}$$ (avoid "cancellation" in finite-calc. of v above)

$$H_j \ldots H_1 a_i = a_i^j \quad \rightarrow \quad \boxed{v_{j+1} = \begin{pmatrix} 0 \\ \vdots \\ (a_i^j)_i \\ \vdots \\ (a_i^j)_m \end{pmatrix} - \alpha_i e_i}$$

- Store v_i and R into A and an extra n-vector.
- Q and H can be computed if needed.
- When column i is completed, row i is too.

2.3.2 Givens Rotation, G

$$\boxed{G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}} \rightarrow \; Gx = G\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \pm\begin{pmatrix} \|a\| \\ 0 \end{pmatrix}$$

$$\rightarrow \quad \boxed{c = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}}$$

- creates 0's one at a time, rotating all column vectors CLOCKWISE
- Doesn't necessarily preserve 0's.
- Create zeros columns-wise, starting from the bottom of the vector and moving up.
- useful for spare matrices
- When column i is completed, row i is too.

Ex: $G_5(2,4) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -s & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

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

- $t = \frac{a_2}{a_1} < 1 \quad \rightarrow \quad c = \frac{1}{\sqrt{1+t^2}} \quad , \quad s = c \cdot t$
- $\tau = \frac{a_1}{a_2} < 1 \quad \rightarrow \quad s = \frac{1}{\sqrt{1+\tau^2}} \quad , \quad c = s \cdot \tau$

2.3.3 Gram-Schmidt Orthogonalization

Column-Oriented: Build the $\widehat{q_i}$ column vector first, then solve for $\widehat{q}_{i+1}$, and continue up to $\widehat{q_n}$.

Classical: Each sucessive term in the sum for q_i depends on the original vector, a_i.

- Extra storage is needed for a_i.
- 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.

$$Q_\parallel^T = \begin{pmatrix} \widehat{q_1} : q_1 = a_1 \\ \widehat{q_2} : q_2 = a_2 - \widehat{q_1}(\widehat{q_1} \cdot a_2) \\ \widehat{q_3} : q_3 = a_3 - \widehat{q_1}(\widehat{q_1} \cdot a_3) - \widehat{q_2}(\widehat{q_2} \cdot a_3) \\ \vdots \\ \widehat{q_n} : q_n = a_n - \sum_j^n \widehat{q_j}(\widehat{q_j} \cdot a_n) \end{pmatrix}$$

$$R = \begin{pmatrix} \|q_1\| & \widehat{q_1} \cdot a_2 & \widehat{q_1} \cdot a_3 & \dots & \widehat{q_1} \cdot a_n \\ 0 & \|q_2\| & \widehat{q_2} \cdot a_3 & \dots & \widehat{q_2} \cdot a_n \\ 0 & 0 & \|q_3\| & \dots & \widehat{q_3} \cdot a_n \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \|q_n\| \end{pmatrix}$$

Modified: Each sucessive term in the q_i sum, $q_i^{[k+1]}$, depends on its previous term, $q_i^{[k]}$.

$$\boxed{\begin{aligned} q_i^{[k+1]} &\equiv q_i^{[k]} - \widehat{q}_k \langle \widehat{q}_k | \widehat{q}_i{}^{[k]} \rangle \\ q_i^{[k]} &\equiv \widehat{q}_k \langle \widehat{q}_k | \widehat{q}_i{}^{[k]} \rangle + q_i^{[k+1]} \\ &\equiv \sum_{k'=k}^{i} \widehat{q}_{k'} \langle \widehat{q}_{k'} | \widehat{q}_i{}^{[k']} \rangle \end{aligned}} \quad \left\{ \boxed{\begin{aligned} q_i &\equiv q_i^{[i]} \\ q_i^{[1]} &\equiv a_i \\ &= \sum_{k=1}^{i} \widehat{q}_k \langle \widehat{q}_k | \widehat{q}_i{}^{[k]} \rangle \\ q_1 &= x_0 \end{aligned}} \right.$$

- No storage for a_i is needed.
- Less error in orthog. with finite-precision arithmatic.

$$Q_n^T = \begin{pmatrix} \widehat{q_1} \ : \ q_1 = a_1 \\ \widehat{q_2} \ : \ q_2 = q_2^{[2]} \ = \overbrace{a_2}^{q_2^{[1]}} - \widehat{q_1}(\widehat{q_1} \cdot \widehat{q_2}^{[1]}) \\ \widehat{q_3} \ : \ q_3 = q_3^{[3]} \ = \underbrace{a_3 - \widehat{q_1}(r_{13})}_{q_3^{[2]}} - \widehat{q_2}\langle \widehat{q_2} | \widehat{q_3}^{[2]} \rangle \\ \vdots \\ \widehat{q_n} \ : \ q_n = q_n^{[n]} \ = \ a_n - \sum_{m=1}^{n-1} \widehat{q}_m(r_{mn}) \end{pmatrix}$$

$$R_n = \begin{pmatrix} \|q_1\| & \langle \widehat{q_1} | \widehat{q_2}^{[1]} \rangle & \langle \widehat{q_1} | \widehat{q_3}^{[1]} \rangle & \dots & \langle \widehat{q_1} | \widehat{q_n}^{[1]} \rangle \\ 0 & \|q_2\| & \langle \widehat{q_2} | \widehat{q_3}^{[2]} \rangle & \dots & \langle \widehat{q_2} | \widehat{q_n}^{[2]} \rangle \\ 0 & 0 & \|q_3\| & \dots & \langle \widehat{q_3} | \widehat{q_n}^{[3]} \rangle \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \|q_n\| \end{pmatrix}$$

Row Oriented: For all $1 \le i \le n$, build each q_i from it's sum, term by term.

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

Augmented Matrix:

$$\big(A \mid b\big) = \big(Q_\parallel \mid q_{n+1}\big) \begin{pmatrix} R & b_1' \\ 0 & \rho \end{pmatrix}$$

$$\begin{pmatrix} | & | & | & | \\ a_1 & .. & a_n & b \\ | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | & | & | \\ \widehat{q_1} & \dots & \widehat{q_n} & q_{n+1} \\ | & | & | & | \end{pmatrix} \begin{pmatrix} r_{11} & \dots & r_{1n} & | \\ & \ddots & \vdots & b_1' \\ 0 & \ddots & r_{nn} & | \\ 0 & \dots & 0 & \rho \end{pmatrix}$$

- Use Gram-Schmidt QR on this, then solve $Rx = b_1'$
- This method is prefered 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

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

$$\boxed{\begin{array}{rcl} (Q^T AP)(P^T x) & = & Q^T b \\ \begin{pmatrix} R & S \\ 0 & 0' \end{pmatrix} \quad \begin{pmatrix} z \\ 0 \end{pmatrix} & = & \begin{pmatrix} b_1' \\ b_2' \end{pmatrix} \end{array}}$$

- $0'$ 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, $\boxed{Rz = b_1' \ , \quad x = P\begin{pmatrix} z \\ 0 \end{pmatrix}}$
- For the minimized-norm solution with the smallest $\|x\|$, S must be annihilated.
- For another method or if underdetermined ($m < n$), something like SVD Decomp. can be used.

2.4 Singular Value Decomposition (SVD)

$$\boldsymbol{A} = \boxed{\boldsymbol{U\Sigma V^T}} = \begin{pmatrix} | & & | & | \\ \boldsymbol{u_1} & \dots & \boldsymbol{u_k} & \boldsymbol{u_m} \\ | & & | & | \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma_1} & 0 & \dots & 0 \\ 0 & \boldsymbol{\sigma_2} & \ddots & \vdots \\ 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \ddots & \boldsymbol{\sigma_n} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} \text{——} \boldsymbol{v_1} \text{——} \\ \vdots \\ \text{——} \boldsymbol{v_k} \text{——} \\ \vdots \\ \text{——} \boldsymbol{v_n} \text{——} \end{pmatrix}$$

$$= \boxed{\boldsymbol{U_\parallel \Sigma_1 V^T}} = (\boldsymbol{U_\parallel}\ \boldsymbol{U_\perp}) \begin{pmatrix} \boldsymbol{\Sigma_1} \\ \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{V_{0\perp}^T} \\ \boldsymbol{V_{0\parallel}^T} \end{pmatrix} = \boxed{\sum_{\boldsymbol{i}}^{\boldsymbol{n}} \boldsymbol{\sigma_i \cdot u_i v_i^T}}$$

$$\boldsymbol{AA^T = U\mathcal{D}U^T}$$
$$\boldsymbol{A^TA = VDV^T}$$
$$\boldsymbol{\mathcal{D} = \Sigma\Sigma^T}$$
$$\boldsymbol{D = \Sigma^T\Sigma}$$
$$\boldsymbol{\sigma_i = \sqrt{d_i}}$$

- Underdetermined, $m < n$ is possible too.
- Analagous 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 $\sigma_1 \geq \sigma_2 \geq \dots$
- $\forall(k < i),\ \sigma_i = 0 \quad \Rightarrow \quad \text{rank}(A) = k < n$
- $U_\parallel = U_{m\times k}: \quad \text{span}(U_\parallel) = \text{span}(A)\ , \quad \text{span}(U_\perp) = \text{span}(A)^\perp$
- $V_{0\perp} = V_{n\times k}: \quad \text{span}(V_{0\parallel}) = \text{null}(A)\ , \quad \text{span}(V_{0\perp}) = \text{null}(A)^\perp$ $\qquad \text{null}(A) = \{x : Ax = 0\}$

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

- $Ax + r = b \ \rightarrow$
  $\boxed{x = A^+b = \left(V\Sigma^+U^T\right)b}$
- $\boxed{x_{\min} = \sum_{\sigma_i \neq 0} \frac{u_i \cdot b}{\sigma_i}\ v_i}$ 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.: $A \approx A_k = \sum_i^k \sigma_i \left(u_i v_i^T\right)$

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

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

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

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

- $\widehat{A}$ 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)

Jacobian : $J_f = (\vec{\nabla} \otimes \vec{f}\,)^T = (\vec{\nabla}\vec{f}^{\,T})^T$

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

Hessian : $H_f = \vec{\nabla} \otimes (\vec{\nabla} f) = \vec{\nabla}(\vec{\nabla} f)^T$

- $Ax = \lambda x\ ,\ \ \det(A) \neq 0\ \Rightarrow\ A^{-1}x = (1/\lambda)x$
- Shifting : $(A - \sigma I)x = (\lambda - \sigma)x$
- Simple: Normal [Algebraic] Mult. of 1
- Defective: Geo. Mult. $<$ Alg. Mult. (eig. vec. #)

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

3.1 Schur Form

Unitary (for all): $\boxed{T^\dagger AT = R}$ (T is unitary) (R is upp. triang.)

- $A\vec{x_i} = r_{ii}\vec{x_i}$
- $0 = (R - r_{ii}I)\ \vec{x_i}$

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

$$\Downarrow$$

$$\vec{x_i} = \begin{pmatrix} \vec{y} \\ -1 \\ 0 \end{pmatrix}\ :\ \underset{\text{(nonsingular)}}{(R_{11} - r_{ii}I)\ \vec{y} = \vec{u}}$$

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

Real (only for real matrix) : $\boxed{Q^TAQ = R}$ (Q is ortho.) (R is block upp. triang.)

- $R = \begin{pmatrix} R_{11} & \dots & R_{1p} \\ \mathcal{O} & \ddots & \vdots \\ \mathcal{O} & \mathcal{O} & R_{pp} \end{pmatrix}$
- $\lambda_i(A) = \lambda(R_{ii})$
- $R_{ii} = \begin{cases} R_{1\times1} & \text{(real eigenvalue of } A\text{)} \\ R_{2\times2} & \text{(complex eigenvalue pairs of } A\text{)} \end{cases}$
- All other entries are real
- Reducible: If $PAP^T = R$ (P is permu.)

Block Upper Triangular Transformation

- $X_\parallel = (\vec{x_1}\ \dots\ \vec{x_p})\ ,\ \ Ax_i \in \text{span}(x_i) = \mathcal{S}$
  $\Rightarrow\ AX_\parallel = X_\parallel B$
- $I_n = X^{-1}X = \begin{pmatrix} Y_\parallel \\ Y_\perp \end{pmatrix}(X_\parallel\ \ X_\perp) = \begin{pmatrix} Y_\parallel X_\parallel & Y_\parallel X_\perp \\ Y_\perp X_\parallel & Y_\perp X_\perp \end{pmatrix} = \begin{pmatrix} I_p & \mathcal{O} \\ \mathcal{O} & I_{n-p} \end{pmatrix}$
- $X^{-1}AX = \begin{pmatrix} Y_\parallel AX_\parallel & Y_\parallel AX_\perp \\ Y_\perp AX_\parallel & Y_\perp AX_\perp \end{pmatrix} = \begin{pmatrix} Y_\parallel X_\parallel B & Y_\parallel AX_\perp \\ Y_\perp X_\parallel B & Y_\perp AX_\perp \end{pmatrix} = \begin{pmatrix} B & Y_\parallel AX_\perp \\ \mathcal{O} & Y_\perp AX_\perp \end{pmatrix} = R$

3.2 Upper Hessenberg Transformation (see Householder)

$\rightarrow\ H_j^{(k)} = I - \hat{v}\hat{v}^T$

$\rightarrow\ v_{j+1} = \left[\, 0\ \ldots\ 0\ \ (a_i^j)_k\ \ldots\ \ (a_i^j)_m \,\right]^T - \alpha_i e_i$

- Extra 0s are added to v to start del. from a diff. row, k.

4 Eigenvalue Equation, $Ax = \lambda x$

4.1 Error Bound and Conditioning

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

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

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

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

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

- $Ax = \lambda x\ ,\ \ y^H A = \lambda y^H$
- $\underline{\lambda \text{ is simple}}\ \Rightarrow\ y^H x \neq 0\ \ (?)$
- $y^H \cancel{Ax} + \cancel{y^H A \Delta x} + y^H(\Delta A)x + \cancel{y^H(\Delta A)\Delta x}$
  $\approx y^H \cancel{\lambda x} + \cancel{y^H \lambda \Delta x} + y^H(\Delta\lambda)x + \cancel{y^H(\Delta\lambda)\Delta x}$

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

- $AA^\dagger = A^\dagger A\ \rightarrow\ \mathrm{cond}(A) = 1$
- 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

Power Iteration:

$$A^k x_0\ =\ A^k \sum_{i=1}^{n} c_i v_i\ =\ \sum_{i=1}^{n} c_i \left(A^k v_i\right)\ =\ \sum_{i=1}^{n} c_i \left(\lambda_i^k v_i\right)$$

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

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

- Eigenvalue converges to λ_i of largest modulus.
- v_i converges to lin. com. if mult. max λ_i.
- Normalize to $\mathrm{norm}_\infty = 1$ to prevent over/underflow.
- Fails if $\langle v_i | x_0 \rangle = 0$ (unlikely w/ round. error).
- Real A and x_0 won't ever converge a complex.
- Convergence Rate: $C = |\lambda_2/\lambda_1|$

Power Iteration w/ Shifts:

$$\boxed{\Big[A'x_{k-1} = (A-\sigma I)x_{k-1} = x_k\Big] \;\Rightarrow\; \Big[(\lambda_1'+\sigma) \to \tfrac{1}{\lambda_1'+\sigma}\Big]}$$

- Convergence Rate: $C = \left|\dfrac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < |\lambda_2/\lambda_1|$
- $x_k \to v_1 \;:\; \sigma = (\lambda_2 + \lambda_n)/2$
- $x_k \to v_n \;:\; \sigma = (\lambda_1 + \lambda_{n-1})/2$

Inverse [Power] Iteration:

$$\Big[\left(A^{-1}\right)^k x_0 = x_k\Big] \;\Rightarrow\; \boxed{\left(Ax_{k+1} = x_k\right) \to \left(\lambda_1 \to \tfrac{1}{\lambda_1}\right)}$$

- Use LU Decomp. or Cholesky to solve for x_{k+1}
- Eigenvalue converges to λ_i of smallest modulus.

Inverse Iteration w/ Shifts:

$$\boxed{\Big[A'x_{k+1} = (A-\sigma I)x_{k+1} = x_k\Big] \;\Rightarrow\; \Big[(\lambda_1'+\sigma) \to \tfrac{1}{\lambda_1'+\sigma}\Big]}$$

- Eigenvalues converges to λ_i closest to σ.
- Using a new shift each iteration requires refactoring each time.

Rayleigh Quotient (for shifts):

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

- Derived with Normal eq. / Least Lin. Regression
- Use λ as σ for shifts

Deflation (concept):

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

- Use A_{k+1} when λ_1 is found to good accuracy, then repeat with remaining λ_i.

Simultaneous/Subspace [Power] Iteration:

$$X_0 = X_{n\times p} \;\;,\;\; \operatorname{rank}(X_0) = p$$

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

$$\to \boxed{\lim_{k\to\infty} \operatorname{span}\big(X_k\big) = \operatorname{span}(A_p)}$$

- All are found at the same time.
- $\operatorname{span}(A_p)$ is the span of the first p eigenvectors of largest magnitude.
- Ill-conditioned since all columns of X_k converge to v_1 (though at different rates so they're still ortho.).
- Normalize to $\text{norm}_\infty = 1$ to prevent over/underflow.

Orthogonal [Subspace] Iteration:

$$X = X_{n\times p} \;\;,\;\; \operatorname{rank}(X) = p$$

$$\to \boxed{X_k = Q_kR_k \quad (Q = Q_{n\times p})}$$

$$\to \boxed{X_{k+1} = AQ_k} \;= Q_kB_{p\times p}$$

- $\operatorname{span}(Q) = \operatorname{span}(X)$ (see QR Decomp.).
- $B_{p\times p}$ is triag. if $|\lambda_i| > |\lambda_{i+1}|$; else it's block triag. (see Schur Form).
- Orthogonalization is expensive, and convergence may be slow.

4.2.1 QR Iteration

$$X_{k+1} \equiv AQ_k \quad \big(\text{use } X_0 = I \rightarrow X_1 = A\big)$$

$$Q_{k+1}R_{k+1} = Q_k\big(Q_k^H A Q_k\big)$$

$$\rightarrow \boxed{A_k \equiv Q_k^H X_{k+1} = Q_k^H A Q_k}$$

$$= \begin{cases} \big(Q_k^H Q_{k+1}\big)R_{k+1} \equiv \boxed{Q_{k+1}^{(A)} R_{k+1}} \\ Q_k^H (Q_k R_k Q_{k-1}^H) Q_k = R_k Q_k^{(A)} \end{cases}$$

$$\rightarrow \boxed{A_{k+1} = R_{k+1} Q_{k+1}^{(A)}}$$

- $A_k = B_{p\times p}$ (from above), so diag. entries are λ.
- If $(p = n)$ and $(X_0 = I)$, then $\boxed{Q_k = Q_1^{(A)} \dots Q_k^{(A)}}$ and $\boxed{R^{(k)} = R_k \dots R_1}$.
- Since $(A = Q_1 R_1)$, then by induction, $\boxed{A^k = Q_k R^{(k)}}$
- Still, orthogonalization is $\mathcal{O}(n^3)$ expensive and convergence may be slow.

Inverse QR Iteration

$$\Big(A_k^{-H} = Q_{k+1}^{(A)} R_{k+1}^{-H}\Big) \ , \ \Big(A_{k+1}^{-H} = R_{k+1}^{-H} Q_{k+1}^{(A)}\Big)$$

$$\rightarrow \big(A^{-H}\big)^k = Q_k^{(A)} R_k^{-H}$$

- R^H is low. triang.; $Q^{(A)}$ is built backwards from v_n to v_1.
- Columns of $Q^{(A)}$ for inv. QR iter. of A^H are the same as QR iter. of A.
- Means QR iter. of A is an implicit inv. iter., so shifts are recommended from v_n to v_1 (see below).

QR Iteration w/ Shifts & Deflation

Rayleigh Quotient: $\sigma^{(n)} = \frac{\langle q_n|A|q_n\rangle}{\langle q_n|q_n\rangle} = \langle q_n|A|q_n\rangle = (A_k)_{nn}$

$$\rightarrow \boxed{Q_{k+1}^{(A)} R_{k+1} = A_k - \sigma_k^{(n)} I}$$

$$\rightarrow \boxed{A_{k+1} = R_{k+1} Q_{k+1}^{(A)} + \sigma_k^{(n)} I}$$

$$\rightarrow \text{ Deflation: } \lim_{k\to\infty} A_k = \begin{pmatrix} A'_{k+1} & b^T \\ 0 & \sigma_k^{(n)} \end{pmatrix}$$

- Diag. entries of A_k are automatically Rayleigh Quotients.
- σ is the last diag. entry, which corresponds to v_n (see above).
- If $\sigma_k = \lambda_n \rightarrow$ the last row of $Q_{k+1}^{(A)} R_{k+1}$ and R_{k+1} is $0 \rightarrow$ the last row of $R_{k+1} Q_{k+1}^{(A)}$ is $0 \rightarrow$ Deflation (see right).
- Fails if σ is halfway between two λ_i and favors neither. Or cancellation occurs (rare). Also might require complex arithmatic. Other more robust shifts are available. Convergence is only a few iterations, but cost to factor is still $\mathcal{O}(n^3)$.

Hessenberg QR Iteration [w/ Shifts]

(see Hessenberg Transformation, H)

~~$H = H^{(1)} \Rightarrow H^H A H$~~ isn't Hessenberg at all

$H = H^{(2)} \Rightarrow H^H A H$ is Hessenberg at subdiag.

$$\rightarrow \boxed{A_1 = H^H A H = HAH}$$

$$\rightarrow \boxed{A_k \ [-\sigma I] = R_k Q_k^{(A)} \quad (k > 1)}$$

$$= R_k\Big(A_{k-1} \ [-\sigma I]\Big) R_k^{-1} \quad \text{is Hessenberg at subdiag.}$$

- Transformation to Hessenberg is done once and costs $\mathcal{O}(n^3)$.
- Givens Rotations to factorize a Hessenberg matrix at each iter. costs $\mathcal{O}(n^2)$.
- Less iter. needed since already near triagular.
- If $A = A^H$, then the transform is tri-diag that costs $\mathcal{O}(n)$ and A_k approaches diag.
- Still expensive if n is large.
- Excessive storage if large and sparse.
- No advantage if only a few λ_i are needed.

4.3 Krylov Subspace

Characteristic Polynomial: $\Big[Ax = \lambda x\Big] \;\Rightarrow\; \Big[p_n(\lambda) = 0\Big]$

$$\boxed{p_n(\lambda) \;=\; \left(\sum_{i=0}^{n-1} c_i \lambda^i\right) + \lambda^n \;=\; c_0 + c_1\lambda + \cdots + c_{n-1}\lambda^{n-1} + \lambda^n}$$

Companion Matrix: $\Big[p_n(\lambda) = 0\Big] \Rightarrow\; C_n x = \lambda x\;,\; C_n = \begin{pmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_{n-1} \end{pmatrix} = (\, e_2 \;\; e_3 \;\; \dots \;\; e_n \;\; -\vec{c}\,)$

Krylov Matrix: $\boxed{K_k = K_{n\times k} = \begin{pmatrix} x_0 & x_1 & \dots & x_{k-1} \end{pmatrix} = \begin{pmatrix} x_0 & Ax_0 & \dots & A^{k-1}x_0 \end{pmatrix}}$ ($x_0 \neq 0$ vector)

Krylov Subspace: $\boxed{\mathcal{K}_k = \text{span}(K_k)} \;\rightarrow\; \boxed{\mathcal{K}_n = \text{span}(K_n) = \text{span}(A)}$

$$\left.\begin{aligned} AK_n &= \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \\ &= K_n \begin{pmatrix} e_2 & e_3 & \dots & \vec{a} \end{pmatrix} \qquad (\vec{a} = K_n^{-1}x_n) \\ &= K_n C_n \end{aligned}\right\} \Rightarrow \boxed{K_n^{-1} A K_n = C_n}$$

- Assume $\exists K^{-1}$
- C_n is Hessenberg
- Sucessive columns of K_n converge to v_1, so problem is ill-conditioned

4.3.1 Arnoldi Iteration

$\boxed{K_n = Q_n R_n}$

Arnoldi iteration is simply modified, column-oriented Gram-Schmidt, the benefit being that Q_n is built by column-vector. R_n is also built by column when building Q_n.

$$\boxed{\begin{aligned} q_n^{[i+1]} &\equiv q_n^{[i]} - \widehat{q}_i \langle \widehat{q}_i | \widehat{q}_n^{\,[i]} \rangle \\ q_n^{[i]} &\equiv \widehat{q}_i \langle \widehat{q}_i | \widehat{q}_n^{\,[i]} \rangle + q_n^{[i+1]} \\ &\equiv \sum_{i'=i}^{n} \widehat{q}_{i'} \langle \widehat{q}_{i'} | \widehat{q}_n^{\,[i']} \rangle \end{aligned}} \left\{ \boxed{\begin{aligned} q_n &\equiv q_n^{[n]} \\ q_n^{[1]} &\equiv A\widehat{q}_{n-1} \\ &= \sum_{i=1}^{n} \widehat{q}_i \langle \widehat{q}_i | \widehat{q}_n^{\,[i]} \rangle \\ q_1 &= x_0 \end{aligned}} \right.$$

$$Q_n^T = \begin{pmatrix} \widehat{q_1} \;:\; q_1 = x_0 & & & \\ \widehat{q_2} \;:\; q_2 = q_2^{[2]} = \overbrace{A\widehat{q_1}}^{q_2^{[1]}} - \widehat{q_1}(\widehat{q_1}^{\,H}\widehat{q_2}^{\,[1]}) & & & \\ \widehat{q_3} \;:\; q_3 = q_3^{[3]} = \underbrace{A\widehat{q_2} - \widehat{q_1}(r_{13})}_{q_3^{[2]}} - \widehat{q_2}\langle \widehat{q_2} | \widehat{q_3}^{\,[2]} \rangle & & & \\ \vdots & & & \\ \widehat{q_k} \;:\; q_k = q_k^{[k]} = A\widehat{q}_{k-1} - \sum_{j=1}^{k-1} \widehat{q_j}(r_{jk}) & & & \end{pmatrix}$$

$$R_n = \begin{pmatrix} \|q_1\| & \langle \widehat{q_1} | \widehat{q_2}^{\,[1]} \rangle & \langle \widehat{q_1} | \widehat{q_3}^{\,[1]} \rangle & \dots & \langle \widehat{q_1} | \widehat{q_n}^{\,[1]} \rangle \\ 0 & \|q_2\| & \langle \widehat{q_2} | \widehat{q_3}^{\,[2]} \rangle & \dots & \langle \widehat{q_2} | \widehat{q_n}^{\,[2]} \rangle \\ 0 & 0 & \|q_3\| & \dots & \langle \widehat{q_3} | \widehat{q_n}^{\,[3]} \rangle \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \|q_n\| \end{pmatrix}$$

$\boxed{Q_n^{-1} A Q_n = R_n C_n R_n^{-1} \;\equiv\; H}$ $\left(\begin{smallmatrix}\text{Still} \\ \text{Hessenberg}\end{smallmatrix}\right)$

$\boxed{AQ_n = Q_n H}$

Instead of building R_n from $K_n = Q_n R_n$, build H from $AQ_n = Q_n H$ (also by column as Q_n is built).

$$A\widehat{q_j} = \sum_{i=1}^{n} \widehat{q_i} H_{ij} = \sum_{i=1}^{n} \widehat{q_i} \langle \widehat{q_i} | \widehat{q}_{j+1}^{\,[i]} \rangle$$

$$\langle \widehat{q_i} | A | \widehat{q_j} \rangle = H_{ij} = \langle \widehat{q_i} | \widehat{q}_{j+1}^{\,[i]} \rangle$$

$$H = \begin{pmatrix} \langle \widehat{q_1} | A | \widehat{q_1} \rangle & \langle \widehat{q_1} | A | \widehat{q_2} \rangle & \langle \widehat{q_1} | A | \widehat{q_3} \rangle & \dots & \langle \widehat{q_1} | \widehat{q_n}^{\,[1]} \rangle \\ \|q_2\| & \langle \widehat{q_2} | \widehat{q_3}^{\,[2]} \rangle & \langle \widehat{q_2} | \widehat{q_4}^{\,[2]} \rangle & \dots & \langle \widehat{q_2} | \widehat{q_n}^{\,[2]} \rangle \\ 0 & \|q_3\| & \langle \widehat{q_3} | \widehat{q_4}^{\,[3]} \rangle & \dots & \langle \widehat{q_3} | \widehat{q_n}^{\,[3]} \rangle \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \|q_n\| & \langle \widehat{q_n} | \widehat{q}_{n+1}^{\,[n]} \rangle \end{pmatrix}$$

4.3.2 Rayleigh-Ritz Procedure

$$\boxed{\begin{aligned} Q_n &= \begin{bmatrix} Q_k & U_k \end{bmatrix} \\ Q_k &= [\, \widehat{q_1} \;\; \widehat{q_2} \;\; \dots \;\; \widehat{q_k} \,] \\ U_k &= [\, \widehat{q}_{k+1} \;\; \dots \;\; \widehat{q_n} \,] \end{aligned}} \Rightarrow \quad \begin{aligned} Q_n^H A Q_n &= \begin{bmatrix} Q_k^H \\ U_k^H \end{bmatrix} A \begin{bmatrix} Q_k & U_k \end{bmatrix} \\ &= \begin{bmatrix} Q_k^H A Q_k & Q_k^H A U_k \\ U_k^H A Q_k & U_k^H A U_k \end{bmatrix} = H = \begin{bmatrix} H_k & ||| \\ \tilde{H}_k & ||| \end{bmatrix} \end{aligned}$$

- $\boxed{H_k = Q_k^H A Q_k}$ (Is hessenberg) ($\sim$ Rayleigh Quot.)
- $\boxed{\tilde{H}_k = \begin{bmatrix} \mathbf{0}^T & \|q_{k+1}\| \\ \mathcal{O} & \mathbf{0} \end{bmatrix}}$

- Q_k are the first k vectors of Q_n found through Arnoldi Iteration, called Arnoldi vectors.
- The eigenvalues of H_k are called Ritz values, and its eigenvectors premultiplied by Q_k are called Ritz vectors.
- At iteration $H_{k=n} = H$, then the Ritz values/vectors are equal to the eigenvalues/vectors of A.
- The Ritz values/vectors of H_k converge to A's eigenvalues/vectors as $k \to n$, and may be found from QR/Orthogonal iteration.
- Extreme eigenvalues are found quickest; shifting + inverting can be used to find interior eigenvalues.
- $\big(\|q_{k+1}\| = 0 \to \tilde{H}_k = 0\big) \to H$ is block triag., $\mathcal{K}_k$ is an invar. subspace, and the Ritz values/vectors ARE A's eigenvalues/vectors.
- For each kth iteration, multiplying Aq costs $\mathcal{O}(n^2)$, orthogonalizing costs $\mathcal{O}(kn)$, and computing the Ritz values/vectors costs $\mathcal{O}(k^3)$. Storage is also needed for Q_k and H_k. As this increases with k, it's better to do a few iterations, then restart the Arnoldi process with a better x_0 based on already computed info.

4.3.3 Lanczoz Iteration (when $A = A^H$)

$$\boxed{\begin{aligned} A = A^H \;\Rightarrow\; Q_n^H A Q_n &= H_n \quad \text{hessen. and symm. = tridiag.} \\ T_k &\equiv H_k \quad \text{(common notation in practice)} \end{aligned}}$$

$$Q_n^T = \begin{pmatrix} \widehat{q_1} & : & q_1 & = & x_0 & \\ \widehat{q_2} & : & q_2 & = & A\widehat{q_1} - \widehat{q_1}\langle \widehat{q_1}|A\widehat{q_1}\rangle - \overbrace{\widehat{q_0}\langle q_1|q_1\rangle}^{\equiv 0} & \\ \widehat{q_3} & : & q_3 & = & A\widehat{q_2} - \widehat{q_2}\langle \widehat{q_2}|A\widehat{q_2}\rangle - \widehat{q_1}\langle q_2|q_2\rangle QR & \\ \vdots & & & & & \\ \widehat{q}_{i+1} & : & q_{i+1} & = & A\widehat{q_i} - \widehat{q_i}\langle \widehat{q_i}|A\widehat{q_i}\rangle - \widehat{q}_{i-1}\langle q_i|q_i\rangle & \end{pmatrix}$$

Similar to Arnoldi Iteration in building Q_n and H by column and using the Rayleigh-Ritz procedure to estimate the eigenvalues/vectors, but much more simplified with an only three-term reoccurance for Q_n, and a symm. tridiag. H for storage and computation of the Ritz values/vectors.

$$H = T = \begin{pmatrix} \langle \widehat{q_1}|A|\widehat{q_1}\rangle & \langle \widehat{q_2}|\widehat{q_2}\rangle & 0 & 0 \\ \|q_2\| & \langle \widehat{q_2}|A|\widehat{q_2}\rangle & \ddots & 0 \\ 0 & \ddots & \ddots & \|q_n\| \\ 0 & 0 & \|q_n\| & \langle q_n|A|q_n\rangle \end{pmatrix}$$

- Only the diagonal and subdiagonal of T need to be stored.
- Extreme eigenvalues are found quickest; shifting + inverting can be used to find interior eigenvalues.
- $\big(\|q_{k+1}\| = 0 \to \tilde{T}_k = 0\big) \to T$ is block triag., $\mathcal{K}_k$ is an invar. subspace, and the Ritz values/vectors ARE A's eigenvalues/vectors.
- Finite-arithmatic means the vectors are more likely to lose orthogonality. Reorthogonalizing them might be expensive, so sometimes it is ignored if the Ritz values are close enough. Not reorthogonalizing might lead to repeated eigenvalues.

4.4 Jacobi Method (when $A = A^T$)

J is a Givens Rotation Matrix

$$\boxed{A_{k+1} = J^T A_k J = J^T \begin{pmatrix} a & b \\ b & d \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}}$$

$$\boxed{0 = 1 + t \cdot \tfrac{(a-d)}{b} - t^2} \quad \text{(Choose smaller } |t = \tfrac{s}{c}|\text{)}$$

- Creates 0s by rotating the column vectors and then the row vectors.
- The row vector rotation doesn't preserve 0s at all.
- $\lim_{k\to\infty} A_k = \text{Diag.}$
- Each matrix sweep costs the same as tridiag. + QR
- Easy to use with parallel-computing.