1 Linear algebra review

1.1 Vector products

inner product: $x^T y = \sum x_i * y_i$ • $x^T y = \|x\|_2 \|y\|_2 \cos\theta \iff |x^T y| \leq \|x\|_2 \|y\|_2$ • $x^T y = 0 \Leftrightarrow x \perp y$
The **outer product** results in a matrix, the outer sum of the two vectors

1.2 Norms

All norms: $\|x\|_x = 0 \Leftrightarrow x = 0$ • $\|\alpha x\|_x = |\alpha| \|x\|_x$ • $\|x + y\|_x \leq \|x\|_x + \|y\|_x$ • $\|x - y\|_x \geq \|x\|_x - \|y\|_x$
Vector norms: $\|x\|_1 = \sum_{i=1}^n |x_i|$ • $\|x\|_2 = \sqrt{\sum_{i=1}^n (x_i)^2}$ • $\|x\|_\infty = \max_{i \in i,\ldots,n} |x_i|$ • $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty$
Matrix norms:

- $\|A\|_\infty = \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{\|x\|_\infty = 1} \|Ax\|_\infty = \max_i \left\|a_i^T\right\|_1$
- $\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} = \sqrt{tr(AA^T)} = \sqrt{tr(A^T A)} = \sqrt{\sum_{k=1}^{min(m,n)} \sigma_k^2}$
- $\|AB\|_p \leq \|A\|_p \|B\|_p$ (not always true for Frobenius norms) • $\|Ay\|_p \leq \|A\|_p \|y\|_p$
- $\|Qx\|_x = \|x\|_x$ • $\|QA\|_x = \|A\|_x$ (for Q orthogonal)
- $\|A\|_2 \leq \sqrt{m} \|A\|_\infty$ • $\|A\|_\infty \leq \sqrt{n} \|A\|_2$

1.3 Matrix properties

Determinant: represents how the volume of a hypercube is transformed by the matrix.

- For square matrix: $det(\alpha A) = \alpha^n det(A)$ • $det(AB) = det(A)det(B)$ • A singular $\Leftrightarrow det(A) = 0$
- For all matrices: $det(A) = det(A^T)$ • $det(A^{-1}) = \frac{1}{det(A)}$

Trace: $tr(A) = \sum_{i=1}^n a_{ii}$ • $tr(A) = tr(A^T)$ • $tr(A + \alpha B) = tr(A) + \alpha tr(B)$ • $tr(ABCD) = tr(BCDA)$ • $tr(uv^T) = v^T u$
Inverse and transpose: $A^T (A^{-1})^T = (A^{-1}A)^T = I^T = I = I^T = (AA^{-1})^T = (A^{-1})^T A^T$
Sherman-Morrison-Woodbury formula: $(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1}$ for $A \in \mathbb{R}^{n \times n}, U, V \in \mathbb{R}^{n \times k}$
Proof: $LHS^{-1} * RHS$: $(A+UV^T)(A^{-1} - A^{-1}U(I+V^T A^{-1}U)^{-1}V^T A^{-1})$. The end result is $I \Longrightarrow RHS$ is an inverse of $(A+UV^T)$

Orthogonal matrices: $q_i^T q_j = 1$ for $i = j$, and $q_i^T q_j = 0$ for $i \neq j$. Equivalently, $Q^T Q = I$. For square matrices, $Q^T Q = QQ^T = I$

1.4 Projections, reflections, and rotations

Projections: A projection, v, of vector x onto vector y $v = \frac{y^T x}{y^T y} y$. Interpreted as portion of x in the direction of y ($y^T x$), times the direction of y, divided by the length of y twice ($y^T y$)
Projection matrices are square matrices, P, s.t., $P^2 = P$.

Reflection: $P^2 = I$ • $P = I - \beta vv^T$, with $\beta = \frac{2}{v^T v}$, and v the vector orthogonal to the line/plane of reflection • $Px = x \Leftrightarrow v^T x = 0$ ("fixed points" of P)

1.5 Symmetric Positive Definite (SPD) Matrices

For A**, SPD:** i) $A = A^T$, ii) $x^T Ax > 0 \; \forall x \neq 0$, iii) $a_{ii} > 0$, iv) $\lambda(A) \geq 0$, v) for B nonsingular, $B^T AB$ is also SPD.
Tricks for proofs: i) Multiply by e_i since $e_i \neq 0$, ii) Use matrix transposes$x^T A^T = (Ax)^T$ to rearrange formulas

$B^T AB$ **is SPD** if A SPD and B nonsingular: $x^T B^T ABx = (Bx)^T A(Bx) > 0, (\text{since } B \text{ nonsingular} \Rightarrow Bx \neq 0)$

1.6 Eigenvalues

Eigenvalues: $Ax = \lambda x \Longrightarrow (A - \lambda I)x = 0$ • $\lambda(A) = \lambda(A^T)$ • $\lambda_1 = \max_{x \neq 0} \frac{x^T Ax}{\|x\|_2^2}$ • $det(A) = \prod_{i=1}^n \lambda_i$ • $tr(A) = \sum_{i=1}^n \lambda_i$
The **algebraic multiplicity** of an eigenvalue, λ_i, is the number of times that λ_1 appears in $\lambda(A)$
The **geometric multiplicity** of an eigenvalue, λ_i, is the dimension of the space spanned by the eigenvectors of λ_i

Triangular matrices: $t_{ii} = \lambda_i, \forall i \in \{1, \ldots, n\}$ • T nonsingular $\Leftrightarrow$ all $t_{ii} \neq 0$

Gershgorin disc theorem: all $\lambda \in \lambda(A) \subset \mathbb{D}_i = \{z \in \mathbb{C} \mid |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|\}$
Proof: $Ax = \lambda x \longleftrightarrow (A - \lambda I)x = 0 \longleftrightarrow \sum_{j \neq i} a_{ij}x_j + (a_{ii} - \lambda)x_i = 0,\ \forall i \in \{1, \ldots, n\}$

$$|(a_{ii} - \lambda)| = |\sum_{j \neq i} \frac{a_{ij}x_j}{x_i}| \leq \sum_{j \neq i} |\frac{a_{ij}x_j}{x_i}| \text{ , by triangle inequality for } i \ s.t. |x_i| = \max_i |x_i| \Longrightarrow |(\lambda - a_{ii})| \quad \leq \sum_{j \neq i} |a_{ij}|, \text{ since } |\frac{x_j}{x_i}| \leq 1$$

2 Matrix Decompositions

Schur Decomposition: (any matrix) $A = QTQ^H$, Q unitary ($Q^HQ = I$), T upper triangular. Q orthogonal when $A \in \mathbb{R}^{n \times n}$

Eigenvalue Decomposition: (A diagonalizable) $A = X\Lambda X^{-1}$, Λ diagonal with $\lambda(A)$. X orthogonal for A real symmetric

2.1 Singular Value Decomposition

Definition: $A = U\Sigma V^H$. When $A \in \mathbb{R}^{m \times n}$, $A = U\Sigma V^T$ with $U, V, \Sigma \in \mathbb{R}$

Derivation: Below considers case when A full rank. Observe A^TA symmetric: $(A^TA)^T = A^TA$

A^TA symmetric $\Rightarrow \exists\ Q$ orthogonal and Λ diagonal matrix of λ_i s.t.,

$A^TA = Q\Lambda Q^T \Longleftrightarrow Q^TA^TAQ = Q^TQ\Lambda Q^TQ \longleftrightarrow (AQ)^T(AQ) = \Lambda$

note AQ is orthogonal, but scaled to the eigenvalue in that row: $\lambda_i = \|Aq_i\|_2^2$

$A = AQQ^T = (AQ)Q^T = AQD^{-1}DQ^T$, where D has $\sqrt{\lambda_i}$ on diagonal and D^{-1}has $\frac{1}{\sqrt{\lambda_i}}$ on diagonal

$A = U\Sigma V^T$, where $U = AQD^{-1}, \Sigma = D, V^T = Q^T$

SVD properties:

- σ_i always ≥ 0 • $\|A\|_2 = \sigma_1$ • $\left\|A^{-1}\right\|_2 = \frac{1}{\sigma_n}$ when A nonsingular • $\|A\|_F = \sqrt{\sum_i^{min\{n,m\}} \sigma_i^2}$
- $\lambda(A^TA) = \lambda(AA^T) = (\sigma(A))^2$ • **Condition number**, $\kappa(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\sigma_1}{\sigma_n}$
- When A symmetric, $\sigma_i = |\lambda_i|$ • When A orthogonal, $\sigma_1 = \cdots = \sigma_n = 1$
- V contains the eigenvectors of A^TA ($A^TAv_i = \sigma_i^2 v_i$) • U contains the eigenvectors of AA^T ($AA^Tu_i = \sigma_i^2 u_i$)

3 Error analysis

Floating point equation: $\pm(\sum_{i=1}^{t-1} d_i\beta^{-i})\beta^e$
Unit roundoff: $u = \frac{1}{2} \times \beta^{-(t-1)}$ (distance between the smallest digits stored in a floating-point number). For double precision floating point numbers (64 bits), $u \approx 10^{-16}$
Conditioning: relative sensitivity of a problem • Sensitivity: $\frac{\left\|\tilde{f}(x) - f(x)\right\|_p}{\|\tilde{x} - x\|_p}$ • Relative sensitivity: $\frac{\left\|\tilde{f}(x) - f(x)\right\|_p \|x\|_p}{\|\tilde{x} - x\|_p \|f(x)\|_p}$

4 LU Factorization

$Ax = b \Longrightarrow LUx = b \Longrightarrow Lz = b, Ux = z$

Description: $LU = l_1u_1^T + \cdots + l_nu_n^T$, and when l_1, u_1^T are from lower/upper respectively, $LU - l_1u_1^T$ yields a matrix with zeros in the first row and column
Basic altorithm: $u_1^T = a_1^T$ • $l_1 = a_1/a_{11}$ • $A' \leftarrow A - l_1u_1^T$. In practice (and somewhat confusingly), A' is now referred to as A
Observe each l_i, u_i^T constructed are the rows/columns of the lower and upper triangular matrices of L, U respectively.

4.1 Pivoting

4.1.1 When pivoting is needed

When pivot = 0: Basic algorithm relies on the pivots, $a_{kk} \neq 0$. This will occur if none of the $k \times k$ blocks of A, $A[1:k, 1:k]$, have a determinant of 0. **Proof by induction**:

Case k=1:

$A_1 = L_1U_1 \longleftrightarrow det(A_1) = det(L_1U_1) \longleftrightarrow det(A_1) = det(L_1)det(U_1)$, by property of determinants

$det(A_1) = det(U_1)$, since determinant of a triangular matrix is a product of the diagonals and the diagonal of L_1 are 1's

$det(A_1) = a_{11} = u_{11} \rightarrow$ so when determinant is not zero, we have a nonzero pivot

Case k=n: assumed to be true

When small values on diagonal: If entries of L large this algorithm can generate roundoff errors.

4.2 Cholesky factorization

$A = GG^T$, with G lower triangular. **Sketch of proof:**

- An SPD matrix, $A = \begin{bmatrix} a & C^T \\ C & B \end{bmatrix}$ where a is 1x1, C is n-1x1, and b is n-1xn-1
- First step of LU: $A = L_1U_1 \iff \begin{bmatrix} a & C^T \\ C & B \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ C/a & I \end{bmatrix} \begin{bmatrix} a & C^T \\ 0 & B - (1/a)CC^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ C/a & I \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & B - (1/a)CC^T \end{bmatrix} \begin{bmatrix} 1 & C^T/a \\ 0 & I \end{bmatrix}$
- Observing i) A symmetric $\Rightarrow B$ symmetric $\Rightarrow B - (1/a)CC^T$ symmetric ii) A is SPD $\Longrightarrow a = e_1^T A e_1 > 0$
- $\begin{bmatrix} 1 & 0 \\ C/a & I \end{bmatrix}$ nonsingular $\Longrightarrow \begin{bmatrix} a & 0 \\ 0 & B - (1/a)CC^T \end{bmatrix}$ SPD (A SPD $\Rightarrow B^TAB$ SPD)
- This shows this process preserves SPD. Complete proof with induction

Continuing with this factorization, we get $A = LDL^T$. Common to rewrite $A = LDL^T = GG^T$, where $G = LD^{\frac{1}{2}}$

4.3 Schur complement

For $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, after k steps of LU, $A' = \begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{bmatrix} \begin{bmatrix} I & A_{21}A_{11}^{-1} \\ 0 & I \end{bmatrix}$

Schur complement: The bottom-right block of A', A'_{22}, equal to $A_{22} - A_{21}A_{11}^{-1}A_{12}$ from the original matrix.

4.3.1 Schur complement derivation

At any step in the LU factorization, A can be written in the form $A' = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$

From this equality, we can create a system of equations

$$U_{12} = L_{11}^{-1}A_{12} \bullet L_{11}^{-1} = L_{21}A_{21}A_{11}^{-1} \bullet A_{22} - L_{21}U_{12} = L_{22}U_{22} \bullet A_{22} - A_{21}A_{11}^{-1}A_{12} = L_{22}U_{22}$$

Notice that the Schur complement equals the product of $L_{22}U_{22}$

We can then show A'_{22} in the LU factorization is equal to $A_{22} - L_{21}U_{12}$ (since at each step we're subtracting $l_iU_i^T$), and get

$$A'_{22} = A_{22} - L_{21}U_{12} = (L_{21}U_{12} + L_{22}U_{22}) - L_{21}U_{12} = L_{22}U_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$$

5 QR factorization

The QR factorization is unique: for full rank matrix, A:

$$A = QR \longleftrightarrow Q^TA = R \longleftrightarrow^T Q^TA = R^TR \longleftrightarrow (QR)^TA = R^TR \longleftrightarrow A^TA = R^TR$$

We now have a matrix, A^TA that can be written of the form R^TR, which is the structure of the Cholesky factorization. Suffice to show that A^TA is Symmetric and Positive Definite (SPD) to prove the uniqueness of R.

5.1 Householder reflection

Householder reflection algorithm: Construct Q^T for each column in A that projects it onto a corresponding column of an upper right triangular matrix, R. **The key** to the iterative part of the algorithm is to construct $Q_i^T, i > 1$ with an identity matrix in the upper-left $i-1 \times i-1$ quadrant, and a smaller Q_i^{*T} in the lower right $n-i \times n-i$ quadrant

Constructing the Householder reflection: maps $a \rightarrow \|a\|_2 e_1$ with $P = I - \beta vv^T$, where $v = a - \|a\|_2 e_1$, and $\beta = 2/v^Tv$

- Mechanics: multiplying Px is the same as taking the vector x and subtracting $\frac{2vv^T}{v^Tv}x$ from it, twice the projection of x onto v
- In our case we want to reflect a onto $\|a\|_2 e_1$. $a + \|a\|_2 e_1$ is the line of reflection, and $v = a - \|a\|_2 e_1$, perpendicular to this, is the vector that defines the line of reflection

5.2 Givens transformation

Useful for sparse matrices. A **Givens rotation** rotates $u = (u_1, u_2)^T$ to $\|u\|_2 e_1$. The matrix that does this, G^T, is defined by

$$G^T = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, c = \frac{u_1}{\|u\|_2}, s = -\frac{u_2}{\|u\|_2}$$

A matrix, P_i, can be constructed to only contain this targeted transformation. Sequentially, the P_i's can multiply A to arrive at R

5.3 Gram-Schmidt transformation

For $A \in \mathbb{R}^{m \times n}$ is tall and thin. Similar to LU, **Gram-Schmidt Transformation** starts with $A = QR = q_1 r_1^T + \dots q_m r_m^T$:

$r_{11} = \|a_1\|_2$, since $\|a_1\|_2 = \|q_1 r_{11}\|_2$ and q_i orthogonal • $q_1 = \frac{1}{r_{11}} a_1$, since $a_1 = q_1 r_{11}$ by construction of QR

$r_{1j} = q_1^T a_j$, (repeat for all j) since $a_j = q_1 r_{1j} + \dots + q_j r_{jj} \iff q_1^T a_j = q_1^T q_1 r_{1j} + \dots + q_1^T q_j r_{jj} \iff q_1^T a_j = r_{1j}$, since q_i orthonormal

$A' = A - q_1 r_1^T$

Repeat for A', the construction of r_{kk}, q_k, r_{kj}: • $a_k = \sum_{i=1}^{k} r_{ik} q_i = r_{kk} q_k + \sum_{i=1}^{k-1} r_{ik} q_i$ • 1. $r_{ik} = q_i^T a_k$ for each $r_{ik}, i < k$, since Q orthonormal and q_{k-1} known • 2. $z = r_{kk} q_k = q_k - \sum_{i=1}^{k-1} r_{ik} q_i$ • 3. $r_{kk} = \|z\|_2$, $q_k = \frac{z}{r_{kk}}$

5.4 QR factorization to solve least-squares problems

Least-squares problem: $argmin_x \|Ax - b\|_2$.

Method of normal equations: A full rank. The point, x which solves $argmin_x \|Ax - b\|_2$ is one where $b - Ax \perp Range(A)$. To solve for this:

$$\text{Want: } (b - Ax) \perp \{z | z = Ay\} \longleftrightarrow (b - Ax) \perp range(A) \longleftrightarrow (b - Ax) \perp a_i, \forall i \in A$$
$$a_1^T (b - Ax) = 0, \forall i \in A \longleftrightarrow A^T (b - Ax) = 0 \longleftrightarrow x = (A^T A)^{-1} A^T b$$

Use Cholesky for fast/accurate solve since $A^T A$ is SPD. Notice $\kappa(A^T A) = \kappa(A)^2$, so inaccurate method if A poorly conditioned

QR method for least squares: A full rank. Attempts to address issue of poor contiioning.

$$A^T (Ax - b) = 0 \longleftrightarrow R^T Q^T (Ax - b) = 0$$
$$Q^T (Ax - b) = 0, \text{ since we assume } A, R \text{ full rank (multiply both sides by } R^{-T})$$
$$Q^T QRx - Q^T b = 0 \longleftrightarrow Rx = Q^T b \longleftrightarrow x = R^{-1} Q^T b$$

SVD for rank-deficient A: A not full rank, we can get infinite solutions (a line of points that satisfy $argmin_x \|Ax - b\|_2$). To choose x, we add constraint $\min_x \|x\|_2$ to our original objective function. Use "thin" SVD so Σ has an inverse and calculate x as

$$(Ax - b) \perp range(A) \longleftrightarrow (Ax - b) \perp range(U), \text{ since } R(A) = R(U) \text{ for } A = U\Sigma V^T$$
$$U^T (Ax - b) = 0 \longleftrightarrow U^T (U\Sigma V^T x - b) = 0 \longleftrightarrow \Sigma V^T x = U^T b$$
$$x = V\Sigma^{-1} U^T b \text{ (the "thin" SVD here provides a nonsingular } \Sigma \in \mathbb{R}^{r \times r}, \text{ so we can take the inverse}$$

Observe for $\min_x \|x\|_2$ that the $x \perp N(A)$ is the shortest vector between $N(A)$ and the vector/plane of solutions to $argmin_x \|Ax - b\|_2$. This value must be in $Range(V)$ since $Range(V) = N(A)^\perp$

6 Iterative methods to find eigenvalues

6.1 Select eigenvalue theorems

Eigenvalues of similar matrices: For S nonsingular and $A = S^{-1} B S$, then i) $\lambda(A) = \lambda(B)$ and ii) x eigenvector of $A \Leftrightarrow S^{-1} x$ eigenvector of B.

i) $\lambda(A) = \lambda(B) : det(A - \lambda I) = det(S^{-1}) det(A - \lambda I) det(S) = det(S^{-1}(A - \lambda I)S) = det(B - \lambda I)$

ii) x e-vect of $A \Leftrightarrow S^{-1} x$ e-vect of $B : Ax = \lambda x \to S^{-1} Ax = \lambda S^{-1} x \to S^{-1} A S S^{-1} x = \lambda S^{-1} x \to B(S^{-1} x) = \lambda(S^{-1} x)$

Eigenvalues from invariant subspaces: $X \in \mathbb{R}^{n \times m}$ invariant subspace of $A \in \mathbb{R}^{n \times n} \Leftrightarrow \exists B \in \mathbb{R}^{n \times m}$ s.t. $AX = XB$

$$\Rightarrow: X \text{ invariant} \longrightarrow Ax_i \in X \longrightarrow Ax_i = \sum_{j=1}^{m} x_j b_{ji} \longrightarrow AX = XB$$

$$\Leftarrow: AX = XB \longrightarrow Ax_i = \sum_{j=1}^{m} x_j b_{ji} \longrightarrow Ax_i \in X \longrightarrow X \text{ invariant}$$

Furthermore, when $AX = XB$, the m eigenvalues of B are also eigenvalues of A: $By = \lambda y \longrightarrow XBy = \lambda Xy \longrightarrow AXy = \lambda Xy$

6.2 Power iteration

Assuming $A = X\Lambda X^{-1}$, diagonalizable, given $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n \in \lambda(A)$. Implemented through the following method:

1. q_0, vector chosen at random
2. $z_k = Aq_k = A^k q_0$, evaluating for convergence if $z_k \parallel q_k \rightarrow z_k^T x_k = \|z\|_2 \|x\|_2$
3. $q_{k+1} = \dfrac{z_k}{\|z_k\|_2} = \dfrac{A^k q_0}{\|A^k q_0\|_2} \approx (\dfrac{\lambda_2}{|\lambda_1|})^k x_1$

Since $A^k q_0 = Aq_k \approx \lambda_1 x_1$, where $\|x_1\|_2 = 1$ $(WLOG)$ and $q_k \parallel x_1$, we can solve for λ:

$$Aq_k \approx \lambda_1 x_1 \Longrightarrow Ax_1 \approx \lambda_1 x_1 \Rightarrow x_1^H A x_1 \approx \lambda_1$$

Convergence: $O((|\dfrac{\lambda_1}{\lambda_2}|)^k)$, since

$$A^k q_0 = \sum_i \alpha_i A^k x_i = \sum_1 \alpha_i \lambda_i^k x_i = \alpha_1 \lambda_1^k (x_i + \frac{\alpha_2}{\alpha_1}(\frac{\lambda_2}{\lambda_1})^k + \cdots + \frac{\alpha_n}{\alpha_1}(\frac{\lambda_n}{\lambda_1})^k) \Longleftrightarrow \left\|A^k q_0\right\|_2 = |\alpha_1 \lambda_1^k|(1 + O(\frac{\lambda_2}{\lambda_1})^k)$$

6.3 Inverse iteration

This process finds the eigenvector of A that to μ. Do this by iteratively multiplying matrix $(A - \mu I)^{-1}$ instead of A.
Observe $(A - \mu I)^{-1}$ has the same eigenvectors of A:

$$(A - \mu I)^{-1} x = \lambda x \longleftrightarrow x = (A - \mu I)x = \lambda Ax - \lambda \mu x \longleftrightarrow \lambda Ax = x + \lambda \mu x \longleftrightarrow Ax = \frac{(1 + \lambda\mu)}{\lambda} x$$

Performing the power iteration on $(A - \mu I)^{-1}$, the largest eigenvalue to emerge will be of the form $\frac{1}{\lambda_i - \mu}$, and we get

$$(A - \mu I)^{-1k} q_0 = (A - \mu I)^{-1} q_k \approx \lambda_i x_i, \text{ where } \|x_i\|_2 = 1 \ (WLOG) \text{ and } q_k \parallel x_i$$

Since x_i is also an eigenvalue of A, we can solve $x_i^H A x_i = \lambda_i$ for the λ_i closest in magnitude to μ.
Convergence: $O(|\frac{\lambda_i - \mu}{\lambda_j - \mu}|)^k)$, where λ_j is the next closest eigenvalue to μ

6.4 Orthogonal iteration

This process finds r eigenvalues of A in a single iterative process. Assume we use power iteration to compute q_1. To get q_2:

$$A^k = \lambda_1 x_1 y_1^T + \lambda_2 x_2 y_2^T + \cdots \Longrightarrow PA^k = \lambda_1 P x_1 y_1^T + \lambda_2 P x_2 y_2^T + \ldots, \text{ where } P = I - x_1 x_1^T$$

$$PA^k = 0 + \lambda_2 P x_2 y_2^T + \ldots, \text{ since } Px_1 = Ix_1 - x_1 x_1^T x_1 = x_1 - x_1 = 0 \Longrightarrow, \text{ and apply power iteration on } PA \text{to to reveal } \lambda_2$$

The general process is: • Start with λ_1, q_1 from power iteration • Build P_2, orthogonal projector onto $\{q_1\}^\perp$, use power iteration to reveal (λ_2, q_2) • Build P_r, orthogonal projector onto $\{q_1, \ldots, q_{r-1}\}^\perp$, use power iteration to reveal (λ_r, q_r)
Now consider the QR decomposition of X, observing its connection to the Schur Decomposition:

$$A = X\Lambda X^{-1} = QR\Lambda R^{-1} Q^H = QTQ^H, \text{ where upper triangular } T = R\Lambda R^{-1}$$

• The eigenvalues of A are on the diagonal of T • By construction, each column of Q is projecting the corresponding column of X onto a vector orthogonal to the preceding ones • The span of the columns of $Q, span\{q_1, \ldots, q_n\}$ will be equal to the span of the columns of $X, span\{x_1, \ldots, x_n\}$.
The process for the **orthogonal iteration** is:

1. $AQ_k \rightarrow Z$, where k is the iteration and $Q_0 = I$
2. $Z \rightarrow Q_{k+1} R_{k+1}$, the QR factorization of Z
3. Repeat $AQ_{k+1} \rightarrow Z$ and eventually $Q_k \rightarrow Q$

Note in each iteration we are calculating $Q_{k+1}^H A Q_k = R_{k+1}$

6.4.1 Reveal eigenvectors of A from T

Motivation: $A = X\Lambda X^{-1}$ can be hard to calculate.

$$A = X\Lambda X^{-1} = QR\Lambda R^{-1}Q^H = QTQ^H, \text{ where } T = R\Lambda R^{-1}$$
$$A = QY\Lambda Y^{-1}Q^H, \text{ where } T = Y\Lambda Y^{-1} \text{ is easier to compute}$$

Focusing on $T = Y\Lambda Y^{-1}$, choose some λ_i (we could get from power or QR iteration).

$$Tx = \lambda_i x \Longrightarrow (T - \lambda_i I)x = 0 \Longrightarrow (T - \lambda_i I)x = \begin{bmatrix} T_{11} - \lambda_i I & T_{12} & T_{13} \\ 0 & 0 & T_{23} \\ 0 & 0 & T_{33} - \lambda_i I \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{ where one diagonal element is 0}$$

And solve with back substitution:

$$X_3 = 0 : (T_{33} - \lambda_i I)X_3 = 0$$
$$X_2 \text{ is a free parameter} \in \mathbb{R} : 0X_2 + T_{33}X_3 = 0 \Longrightarrow 0X_2 = 0$$
$$X_1 = -(T_{11} - \lambda_i I)^{-1}T_{12}X_2 : (T_{11} - \lambda_i I)X_1 + T_{12}X_2 + T_{13}X_3 = 0$$

So, if T upper triangular with λ_i on diagonal of T, you can figure out all the columns of Y for $T = Y\Lambda Y^{-1}$. It follows the eigenvectors of A are Qy_i. Note, $(T_{11} - \lambda_i I)$ nonsingular as long as the algebraic multiplicity of λ_i is 1.

6.4.2 Rate of convergence in orthogonal (and QR) iteration

Property: the angle between two subspaces, U and V, is defined as $\left\|UU^T - VV^T\right\|_2$
In orthogonal interation, the span of those i columns of Q_k, $span\{q_1, \cdots, q_i\} \longrightarrow$ the span of those columns of X, $span\{x_1, \cdots, x_i\}$. Convergence is dictated by how quickly these spans converge. The rate of convergence is $O(|\frac{\lambda_{i+1}}{\lambda_i}|^k)$.

6.5 QR iteration

The QR iteration builds directly on the framework of the orthogonal iteration. In orthogonal iteration, we compute T_{k+1} with the eigenvalues of A appearing on the diagonal of T_{k+1}: $Q_{k+1}^H AQ_k = T_{k+1}$ with $AQ_k = Z = Q_{k+1}T_{k+1}$
In the QR iteration, we ask if we can go from T_k to T_{k+1} directly. Observe

$$A = Q_k T_k Q_k^H \Longrightarrow T_k = Q_k^H AQ_k \bullet AQ_k = Q_{k+1}R_{k+1} \Longrightarrow Q_{k+1}^H A = R_{k+1}Q_k^H$$
$$T_k = Q_k^H(Q_{k+1}R_{k+1}) \longrightarrow T_k = U_{k+1}R_{k+1} \text{ for } U_{k+1} = Q_k^H Q_{k+1} \bullet T_{k+1} = (R_{k+1}Q_k^H)Q_{k+1} \longrightarrow T_{k+1} = R_{k+1}U_{k+1} \text{ for } U_{k+1} = Q_k^H Q_{k+1}$$

So we have an algorithm for $T_k \to T_{k+1}$, this process is the **QR iteration**:
1. $T_k \longrightarrow U_{k+1}R_{k+1}$, the QR factorization of T_k • 2. $R_{k+1}U_{k+1} \longrightarrow T_{k+1}$ • 3. Repeat with T_{k+1}

Property: R_{k+1} is the same in both QR factorization of $A = Q_{k+1}R_{k+1}$ and $T_k = U_{k+1}R_{k+1}$

$$case\ 1 : A = AQ_0 = Q_1R_1, A = T_0 = U_1R_1^*, \text{ and } T_1 = Q_k^H AQ_1$$
$$U_1R_1^* = Q_0^T Q_1 R_1 = Q_1R_1 \Longrightarrow R_1^* = R_1 \text{ and } U_1 = Q_0^T Q_1$$
$$case\ k : \text{Assume } R_k^* = R_k, U_k = Q_{k-1}^T Q_k, \text{ and } T_k = Q_k^H AQ_k$$
$$case\ k+1 : AQ_k = Q_{k+1}R_{k+1}$$
$$T_k = U_{k+1}R_{k+1}^* = Q_k^H AQ_k = Q_k^H Q_{k+1}R_{k+1} \Longrightarrow R_{k+1}^* = R_{k+1} \text{ and } U_{k+1} = Q_k^H Q_{k+1}$$
$$T_{k+1} = R_{k+1}U_{k+1} = Q_{k+1}^H(Q_{k+1}R_{k+1})U_{k+1} = Q_{k+1}^H AQ_kQ_k^H Q_{k+1} \Longrightarrow T_{k+1} = Q_{k+1}^H AQ_{k+1}$$

6.6 QR iteration on upper Hessenberg

Each QR iteration step of a dense matrix is $O(n^3)$. If we run for $O(k)$ iterations, then this algorithm is $O(kn^3)$.
If we first convert A to upper Hessenberg with $O(n^3)$, and Givens rotations ($O(n^2)$), we can reduce algorithm to $O(n^3 + kn^2)$:

$$\text{Choose } Q_1^T = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1 \end{bmatrix} \text{ to perform a Householder rotation onto the first two entries of } a_1 \in A$$

$$\text{Observe } Q_1^T AQ_1 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1 \end{bmatrix} A \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P_1^T} \end{bmatrix} = \begin{bmatrix} x & x & \cdots \\ x & x & \cdots \\ 0 & x & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \text{ where } a_{11} \text{ is never changed, the rest of } a_1$$

$$\text{is only operated on by } \tilde{P}_1, \text{ and the rest of } a_1^T \text{ is only operated on by } \tilde{P_1^T}$$

$$\text{Continuing on, } Q_n^T \ldots Q_2^T Q_1^T AQ_1Q_2 \ldots Q_n = H = Q^H AQ \text{ where } Q_k^T = \begin{bmatrix} I_k & 0 \\ 0 & \tilde{P}_k \end{bmatrix}$$

H remains upper Hessenberg in QR iteration: This follows since in the first step of QR iteration, H_k is transformed to R_k with givens rotations, $U_k^H H_k = R_k$. And in the second step of QR iteration, H_{k+1} is created as $R_k U_k = H_{k+1} = U_k^H H_k U_k$. Since U_k is a series of givens rotations, these rotations can be constructed/ordered so that H_{k+1} preserves upper Hessenberg.

6.7 QR iteration with shift

When λ_{i+1} is close to λ_i this accelerates convergence. First observe for $\lambda_i \in \lambda(A) \to (\lambda_i - \mu) \in \lambda(A - \mu I)$. In this algorithm, at each step we shift T_k by μI. For μ close to λ_{i+1} close to λ_i, the resulting converence, $|[(\lambda_{i+1} - \mu)/(\lambda_i - \mu)]|^k$ will be faster. Shift does not require that $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$.

1. $\mu_k = T_k[n, n]$
2. $(T_k - \mu_k I) \longrightarrow U_{k+1} R_{k+1}$, QR factorization of the shifted T_k
3. $R_k U_k + \mu_k I \longrightarrow T_{k+1}$, and repeat!

Observe, this shift preserves the original QR iteration:

$$(T_k - \mu_I) = U_{k+1} R_{k+1} \Longrightarrow U_{k+1}^H T_k - \mu_k U_{k+1}^H = R_{k+1}$$
$$T_{k+1} = R_{k+1} U_{k+1} + \mu_k I \Longrightarrow T_{k+1} = (U_{k+1}^H T_k - \mu_k U_{k+1}^H) U_{k+1} + \mu_k I = U_{k+1}^H T_k U_{k+1} - \mu_k I + \mu_k I = U_{k+1}^H T_k U_{k+1}$$

6.8 QR iteration with deflation

Deflation allows us to break up the current QR iteration process into two smaller/easier problems.

- If any sub-diagonal element of an upper Hessenberg matrix, H, is 0, it can be written as $H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}$ with H_{11} and H_{22} upper Hessenberg and $\lambda(H) = \lambda(H_{11}) \cup \lambda(H_{22})$
- Therefore, if when updating $T_k = R_k U_k + \mu_k I$, any sub-diagonal element of $T_k = 0$, then T_k can be written in this form and the QR iteration can be performed on $(T_k)_{11}$ and $(T_k)_{22}$ separately (simpler problems)

Theorem: $\lambda(H) = \lambda(H_{11}) \cup \lambda(H_{22})$ for H block upper triangular. **Proof:**

$$\Longrightarrow Hx = \lambda x \longrightarrow \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} H_{11}x_1 + H_{12}x_2 \\ H_{22}x_2 \end{bmatrix} = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \end{bmatrix}$$
and either $x_2 = 0$ and $\lambda \in \lambda(H_{11})$ or not and $\lambda \in \lambda(H_{22})$
$$\Longleftarrow H_{11}p_1 = \lambda p_1 \longrightarrow \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \begin{bmatrix} p_1 \\ 0 \end{bmatrix} = \begin{bmatrix} H_{11}p_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \lambda p_1 \\ 0 \end{bmatrix}$$
$$\Longleftarrow H_{22}p_2 = \lambda p_2 \longrightarrow \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \begin{bmatrix} x \\ p_2 \end{bmatrix} = \begin{bmatrix} H_{11}x + H_{12}p_2 \\ H_{22}p_2 \end{bmatrix} = \begin{bmatrix} \lambda x_1 \\ 0 \end{bmatrix}$$
where $H_{11}x + H_{12}p_2 = \lambda x$ for $x = -(H_{11} - \lambda I)^{-1} H_{12} p_2$, making $\lambda \in \lambda(H)$

Theorem: If H is singular unreduced upper Hessenberg, then in QR factorization, $H = QR$, the last row of R is zero.
Explanation: When constructing QR iteration, each column of R can be linearly independent from the previous ones (since we're adding a dimension) except for the last one (since H and R must be singluar):

$$h_1 = h_{11}e_1 + h_{21}e_2 \qquad h_2 = h_{12}e_1 + h_{22}e_2 + h_{32}e_3 \qquad h_{n-1} = \sum_{i=1}^{n} h_{n-1,i} e_i$$

7 Finding eigenvalues of sparse matrices

Reducing complexity in sparse matrices through matrix-vector products, $Ax = \sum_j a_{ij} x_j$, where you can skip all a_{ij} when $a_{ij} = 0$.

7.1 Arnoldi process

Reveals the first k eigenvalues of a sparse matrix using a process similar to *Gram-Schmidt*
With Arnoldi process, we start with equation $Q^H A Q = H \Longrightarrow AQ = QH$ and use Q to make H where each subsequent column of AQ is made orthogonal to all preceeding columns.
The process follows: • 1. Begin with random $q_1 \in Q$, such that $\|q_1\|_2 = 1$. (Iterate through each of the first k columns of Q with • 2. $Aq_j = \sum_{k=1}^{j+1} h_{kj} q_k$, observing we can recover all h_{ij} for $i \leq j$ since $q_i^T A q_j = h_{ij}$ • 3. $Aq_j = \sum_{k=1}^{j} h_{kj} q_k + h_{j+1,j} q_{j+1}$ • 4. $r = Aq_j - \sum_{k=1}^{j} h_{kj} q_k = h_{j+1,j} q_{j+1}$, where only r is unknown • 5. $\|q_{j+1}\|_2 = 1 \Longrightarrow h_{j+1,j} = \|r\|_2$ and $q_{j+1} = \frac{r}{h_{j+1,j}}$

The output of this process is k columns of Q and the upper $k \times k$ block of upper Hessenberg matrix, H, which can be used in the QR iteration to reveal k eigenvalues close to $\lambda(A)$:

$$AQ = QH \implies AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T, \text{ where } Q_k = Q[:, 1:k], H_k = [1:k, 1:k]$$
$$AQ_k = Q_k X_k \Lambda_k X_k^{-1} + h_{k+1,k} q_{k+1} e_k^T, \text{ where } H_k = X_k \Lambda_k X_k^{-1} \text{ through QR iteration}$$
$$A(Q_k X_k) = (Q_k X_k)\Lambda_k + h_{k+1,k} q_{k+1} x_k^T, \text{ where } x_k^T \text{ is the } k^{th} \text{ column of } X$$

And we get an equation where i) $AQ_k \approx Q_k H_k$, ii) Λ_k contains k eigenvalues close to $\lambda_i \in \lambda(A)$, iii) $(Q_k X_k)$ serve as eigenvectors for those eigenvalues, and iv) $h_{k+1,k} q_{k+1} x_k^T$ represents something like an error term.

7.1.1 QR factorization of Krylov subspace contains Q_k from Arnoldi

Proof: We show for $K_k = Q_k R_k$, that R_k is upper triangular.

$$\text{Start with } Q^T K_k = R \text{ upper triangular for } K_k = \begin{bmatrix} | & | & & | \\ q_1 & Aq_1 & \dots & A^k q_1 \\ | & | & & | \end{bmatrix}$$
$$Q^T k_j = Q^T A^{j-1} q_1 = Q^T Q H^{j-1} Q^T q_1, \text{ since } A^k = Q^T H^k Q$$
$$= H^{j-1} Q^T q_1 = H^{j-1} e_1, \text{ since } Q \text{ orthogonal}$$
$$\implies r_j \in R = h_1 \in H^{j-1}, \text{ which has top } j \text{ rows nonzero}$$

This also means Q_k forms a basis for $K(A, q_1, k)$.

7.1.2 Arnoldi process generates a minimal polynomial

Polynomial properties

- If A is diagonalizable, i.e., $A = X\Lambda X^{-1}$, then polynomial $f(A) = X f(\Lambda) X^{-1}$
- **Characteristic polynomial** of A is $p_A(z) = det(zI - A) = \prod(z - \lambda_i)$ and $p_A(\lambda_i) = 0$ for $\lambda_i \in \lambda(A)$
- $f(A) = 0 \implies \lambda_i \in \lambda(A)$ are the roots of the polynomial (e.g., $p_A(A) = X p_A(\Lambda) X^{-1} = 0$)

Our hope with the Arnoldi process is that for $p_k(H_k) = 0$, revealed in Arnoldi, $p_k(A)$ is minimally small among degree $k-1$ polynomials. Instead of showing $\|p_K(A)\|_2$ is minimized (which is hard), we show $\|p_K(A) q_1\|_2$ is minimized

7.2 Lanczos process

The **Lanczos process** is a parallel process to the Arnoldi process, but for symmetric matrices. Reminder: A symmetric upper Hessenberg matrix, T is tri-diagonal. The process follows

1. $\alpha_k = q_k^T A q_k \implies \alpha_k q_k = A q_k$ • 2. $r_k = A q_k - \beta_{k-1} q_{k-1} - \alpha_k q_k \implies r_k = \beta_{k-1} q_{k-1}$, r_k becomes the orthogonal part of Aq_k
3. $\beta_k = \|r_k\|_2$ • 4. $q_{k+1} = \dfrac{r_k}{\beta_k}$

The orthogonalization in step 2 is reduced from $O(k)$ in Arnoldi to $O(1)$ in Lanczos because of the symmetry of A

8 Iterative splitting methods for solving linear systems

General idea: For $A = M - N$, where M nonsingular: $Ax = b \iff Mx - Nx = b \iff x = M^{-1}Nx + M^{-1}b$
Convergence depends on $M^{-1}N$: Define the error at step $k+1$ as $e_{k+1} = x_{k+1} - x$, then

$$e_{k+1} = x_{k+1} - x = M^{-1}Nx_k + M^{-1}b - M^{-1}Nx - M^{-1}b$$
$$= M^{-1}Nx_k - M^{-1}Nx = M^{-1}N(x_k - x) = M^{-1}Ne_k = (M^{-1}N)^k e_0$$

Convergence only occurs when the spectral radius of $M^{-1}N$, $\rho(M^{-1}N) < 1$. Where $\rho(A) = \max_{\lambda_i \in \lambda(A)} |\lambda_i|$. **Proof:**

$$\text{Let } Gv = \lambda v, \text{ where } G = M^{-1}N \text{ and pick } x_0 = x + v \text{ as a first guess solve}$$
$$\text{Then } e_k = G^k e_0 = G^k(x_0 - x) = G^k v = \lambda^k v \text{ and } e_k \to 0 \text{if} |\lambda| < 1$$

8.1 Jacobi

Definition: Let $A = D - L - U$, and choose $M = D$ and $N = L + U$. Then we have $Dx_{k+1} = (L + U)x_k + b$
Convergence: Converges for any x_0 when A is strictly diagonally dominant: $|a_i i| > \sum_{j \neq i} |a_i j|$
Proof:

$$g_{ii} = 0 \text{ for } G = M^{-1}N \implies \lambda \in D_i = \{z \mid |z| \leq \sum_{i \neq j} |g_{ij}|\}, \text{ by Gershgorin Disc Theorem}$$

$$\text{Suffice to show } \sum_{i \neq j} |g_{ij}| < 1 : \ \sum_{i \neq j} |g_{ij}| < 1 \implies \sum_{i \neq j} \frac{|a_{ij}|}{|a_{ii}|} < 1 \implies \sum_{i \neq j} |a_{ij}| < |a_{ii}| \implies A \text{ is strictly diagonoally dominant}$$

8.2 Gauss-Seidel

Definition: Let $A = D - L - U$, and choose $M = D - L$ and $N = U$. Then we have $(D - L)x_{k+1} = Ux_k + b$
Householder-John Theorem: if $A, B \in \mathbb{R}^{m \times n}$ and A and $[A - B - B^T]$ are SPD, then $\rho(H) < 1$ where $H = (A - B)^{-1}B$.
Proof:

$$Hx = \lambda x \implies (A - B)^{-1}Bx = \lambda x \implies Bx = \lambda(A - B)x \implies Bx = \frac{\lambda}{1 + \lambda} Ax$$

$$x^H Bx = \frac{\lambda}{1 + \lambda} x^H Ax \implies x^H B^T x = \frac{\tilde{\lambda}}{1 + \tilde{\lambda}} x^H A^T x, \text{ by taking the complex conjugate of both sides}$$

$$0 < x^H Ax - x^H Bx - x^H B^T x = \frac{1 - |\lambda|^2}{|1 + \lambda|^2} x^H Ax, \text{ since } A - B - B^T \text{ SPD} \implies 0 < x^H Ax \implies |\lambda| < 1, \text{ since } A \text{ SPD}$$

Convergence: If A SPD, then Gauss-Seidel converges for any x_0
Proof: Choose A SPD with $A = D - L - U$ and $B = -U = -L^T$ and use Householder-John theorem.

8.3 Successive Over-Relaxation (SOR)

Definition: Using the Gauss-Seidel method, we attempt to increase acceleration using $0 < \omega < 2$. The iterative method is $x_{k+1} = x_k + \omega \left[D^{-1}(b + Lx_{k+1} + Ux_k) - x_k\right]$

$$\text{In SOC: } x_{k+1} = x_k + \omega \Delta x_{k+1} = x_k + \omega(D^{-1}(b + Lx_{k+1} + Ux_k) - x_k)$$

SOR is itself a splitting method with $A = D - L - U$, $M = \frac{1}{\omega} D - L$ and $N = (\frac{1}{\omega} - 1)D + U$
Convergence: If A SPD, then SOR converges for any $\omega \in (0, 2)$
Proof: In the Householder-John Theroem, choose $A_{HJ} = \omega A$ and $B_{HJ} = (\omega - 1)D - \omega U$
Also, for $\omega \notin (0, 2)$ there exists x_0 s.t. SOR will not converge
Proof:

$$det(G) = det(M^{-1}N) = \frac{det((1 - \omega)D + \omega U)}{det(D - \omega L)} = \frac{\prod_i (1 - \omega) d_{ii}}{\prod_i d_{ii}} = (1 - \omega)^n, \text{ since } L, U \text{ triangular with 0 on diagonal}$$

$$det(G) = \prod_{\lambda_i \in \lambda(G)} \lambda_i = (1 - \omega)^n \implies |\lambda_{max}(G)| \geq |1 - \omega| \implies \text{Convergence only when } |1 - \omega| < 1$$

8.4 Chebyshev Semi-iterative Method

The most efficient splitting method, but requires us knowing the interval containing $\lambda(A)$. It also uses knob, ω, but we can choose to update this at each step
Definition: With $A = M - N$, $G = M^{-1}N$, and $\omega_k \in \mathbb{R}$, the iterative method is

$$x_{k+1} = x_k + \omega_k((M^{-1}b + Gx_k) - x_k) = x_k + \omega_k M^{-1}(b - Ax_k)$$

$$\text{With } e_k = (I - \omega_{k-1} M^{-1} A)e_{k-1} = \left(\prod_{i=0}^{k-1} (I - \omega_i M^{-1} A) \right) e_0$$

We can minimize error e_k with $\left\| q_k(M^{-1}A)e_0 \right\|_2$, where $q_k(x) = (1 - \omega_{k-1}x) \dots (1 - \omega_0 x)$.

$$\|e_0\|_2 = \left\| q_k(M^{-1}A)e_0 \right\|_2 \leq \max_\lambda q_k(\lambda) \|e_0\|_2$$

$$\text{Recalling } q_k(M^{-1}A) = X q_k(\Lambda) X^{-1} \text{ for } M^{-1}A = X \Lambda X^{-1} \text{ with each diagonal element in } q_k(\Lambda) = q_k(\lambda_i)$$

$$\|e_k\|_2 \leq \max_\lambda q_k(\lambda) \|e_0\|_2 \leq \frac{\|e_0\|_2}{T_k\left(\frac{\beta + \alpha}{\beta - \alpha}\right)}, \text{ where } |q_k(x)| \leq \frac{1}{T_k\left(\frac{\beta + \alpha}{\beta - \alpha}\right)}$$

9 Iterative Krylov methods for solving linear systems

Unrolling splitting methods, we see x_k is a linear combination of $\{b, Ab, \ldots, A^{k-1}b\}$, a Krylov Subspace of A. We can write

- $x_k = Q_k y$ for Q_k the orthonormal basis of $\mathcal{K}_k(A, b, k)$ • $r_k = b - Ax_k = b - AQ_k y$ • Minimizing $r_k \iff r_k \perp \mathcal{K}(A, b, k)$.

9.1 Conjugate Gradient

Krylov method for SPD matrices. Error, $x - Q_k y$ minimized in A norm. Residual, r_k, minimized in A^{-1} norm. O(n) per iteration.

Naive approach to CG

$$\text{Minimize } \|r_k\|^2_{A^{-1}} = (b - AQ_k y)^T A^{-1}(b - AQ_k y) = b^T x - 2y^T Q_k^T b + y^T Q_k^T A Q_k y$$
$$\frac{d}{dy}\left(b^T x - 2y^T Q_k^T b + y^T Q_k^T A Q_k y\right) = 0 \Longrightarrow Q_k^T A Q_k y = Q_k^T b, \text{ minimizes y}$$

And the iterative process becomes: • Construct Q_k from Lanczos process (O(n)) • Compute y from $Q_k^T A Q_k y = Q_k^T b = \|b\|_2 e_1$ (O(k), since H_k assembled in Lanczos) • Compute $x_k = Q_k y$ (O(kn), the expensive step we'll try to simplify)

Efficient approach to CG: We increase efficiency by working with search directions, $\Delta x_k = x_{k+1} - x_k$, instead of x_{k+1} directly. Search directions, $\Delta x_k, \Delta x_l$ are A-conjugate: $(\Delta x_k)^T A \Delta x_l = 0$ for $k \neq l$. **Proof:**

$$r_k - r_{k+1} = (b - Ax_k) - (b - Ax_{k+1}) = Ax_{k+1} - Ax_k = A\Delta x_k \bullet \text{ Since } r_k, r_{k+1} \perp Q_k \Longrightarrow A\Delta x_k \perp Q_k$$
$$\Delta x_l = x_{l+1} - x_l \in \mathcal{K}_{l+1} \text{ and } \Delta x_k = x_{k+1} - x_k \in \mathcal{K}_{k+1} \Longrightarrow \text{For } l < k, A\Delta x_k \perp \Delta x_l \Rightarrow \Delta x_l^T A \Delta x_k = 0$$
$$\Delta x_l^T A \Delta x_k = (\Delta x_l^T A^T)\Delta x_k \Longrightarrow A\Delta x_l \perp \Delta x_k, \text{ since } A \text{ SPD}$$
$$\therefore (\Delta x_k)^T A \Delta x_l = 0 \text{ for } k \neq l$$

We can work with search directions directly using $\Delta x_k = \mu_{k+1} p_{k+1} \quad p_{k+1} = r_k + \sum_{l=1}^k \tau_{lk} p_l$. Determined by

$$p'_{k+1} \in span\{r_0, \ldots, r_k\} = span\{p_1, \ldots, p_k, r_k\} \Longrightarrow p'_{k+1} = \alpha_k r_k + \sum_{l=1}^k \tau'_{lk} p'_l \text{ for some } \alpha_k, \tau'_{lk}$$

$$p_{k+1} = r_k + \sum_{l=1}^k \tau_{lk} p_l, \text{ setting } \alpha_k = \mu_{k+1}, \;\; p_{k+1} = \frac{1}{\mu_{k+1}} p'_{k+1}, \text{ and } \tau_{lk} = \frac{\mu_l}{\mu_k}\tau'_{lk}$$

Let $p'_k = \Delta x_{k-1}$. We rely on several properties to achieve the above

- **Property:** when $x_{k+1} = x_k \Rightarrow r_{k+1} = r_k \Rightarrow r_k = 0$ since $r_k \in \mathcal{K}_{k+1} \perp r_{k+1}$
- **Property:** $span\{p'_1, \ldots, p'_l\} = \mathcal{K}_l$
  - $x_k, x_{k-1} \in \mathcal{K}_k \rightarrow p'_k \in \mathcal{K}_k \iff x_k \neq x_{k-1} \rightarrow \Delta x_{k-1} \neq 0 \rightarrow p'_k \notin \mathcal{K}_{k-1}$
  - $p'_k \in \mathcal{K}_k$ and $p'_k \notin \mathcal{K}_l$ for $l < k \Longrightarrow span\{p'_1, \ldots, p'_l\} = \mathcal{K}_l$
- **Property:** $span\{r_0, \ldots, r_{l-1}\} = \mathcal{K}_l$
  - $b, Ax_{k-1} \in \mathcal{K}_k \Longrightarrow r_{k-1} = b - Ax_{k-1} \in \mathcal{K}_k \iff r_{k+1} \neq r_k \Longrightarrow r_{k-1} \notin \mathcal{K}_{k-1}$
  - $r_{k-1} \in \mathcal{K}_k$ and $r_{k-1} \notin \mathcal{K}_l$ for $l < k \Longrightarrow span\{r_0, \ldots, r_{l-1}\} = \mathcal{K}_l$
- **Property:** $span\{r_0, \ldots, r_{l-1}\} = \mathcal{K}_l = span\{p'_1, \ldots, p'_l\}$
- **Property:** $\mu_{k+1} = \frac{r_k^T r_k}{p_{k+1}^T A p_{k+1}}$

  $$r_k - r_{k+1} = A\Delta x_{k+1} = \mu_{k+1} A p_{k+1} \iff p_{k+1}^T r_k = \mu_{k+1} p_{k+1}^T A p_{k+1}, \text{ since } -p_{k+1}^T r_{k+1} = 0$$
  $$\mu_{k+1} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}} = \frac{r_k^T r_k}{p_{k+1}^T A p_{k+1}}, \text{ since } p_{k+1} = r_k + \tau_k p_k \iff r_k^T p_{k+1} = r_k^T r_k$$

- **Property:** $\tau_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$

  $$p_{k+1} = r_k + \sum_{l=1}^k \tau_{lk} p_l \iff p_k^T A p_{k+1} = p_k^T A r_k + \sum_{l=1}^k \tau_{lk} p_k^T A p_l \iff 0 = p_k^T A r_k + \tau_{kk} p_k^T A p_k$$
  $$\tau_k = \frac{-p_k^T A r_k}{p_k^T A p_k} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}, \text{ since } Ap_k = \frac{1}{\mu_k}(r_{k+1} - r_k) \iff r_k^T A p_k = \frac{r_k^T r_k}{\mu_k} \text{ with } \mu_k = \frac{r_{k-1}^T r_{k-1}}{p_k^T A p_k}$$

9.1.1 Key orthogonality results

- $r_k \perp Q_k$, by construction of r_k to minimize $\|r_k\|_{A^{-1}}$
- $r_k \perp r_l$ for $l \neq k$, since $span\{r_0, \ldots, r_{l-1}\} = \mathcal{K}_l$
- $p_k \perp Ap_l$ for $l \neq k$, since $r_k, r_{k-1} \perp \mathcal{K}_{k-1}$ and $Ap_k = \frac{1}{\mu_k} A\Delta x_k = \frac{1}{\mu_k} A(r_k - r_{k-1})$ then $Ap_k \perp \mathcal{K}_{k-1}$, but for $l < k$, $p_l = \frac{1}{\mu_l}(x_l - x_{l-1}) \in \mathcal{K}_{k-1}$ so $p_l \perp Ap_k$. We can get same result for $l > k$ using A SPD properties.
- $r_k \perp p_l$ for $l \leq k$, TODO
- $r_k \perp Ap_l$ for $l < k$ since $p_l^T A r_k = (Ap_l)^T r_k$ with $Ap_l = \frac{1}{\mu_l}(x_l - x_{l-1}) \in \mathcal{K}_{l+1}$ and $r_k \perp \mathcal{K}_{l+1}$ for $l + 1 < k + 1 \Leftrightarrow l < k$

Congugate Gradient algorithm

(i) Choose some x_0 (could be $x_0 = 0$) • (ii) $r_0 = b - Ax_0,\ p_0 = 0,\ k = 1$ • (iv) Return x_{k-1}

(iii) While $r_{k-1} \neq 0$

$$\tau_{k-1} = \frac{\|r_{k-1}\|_2^2}{\|r_{k-1}\|_2^2} \bullet p_k = r_{k-1} + \tau_{k-1} p_{k-1} \bullet \mu_k = \frac{\|r_{k-1}\|_2^2}{p_k^T A p_k} \bullet x_k = x_{k-1} + \mu_k p_k \bullet r_k = r_{k-1} - \mu_k A p_k \bullet k \leftarrow k + 1$$

9.2 GMRES

Krylov method for general matrices, Generalized Minimal Residual Method (GMRES). Error, $x - x_k = x - Q_k y$ minimized in $A^T A$ norm. Residual, r_k, minimized in 2-norm. O(kn) per iteration.

$$\|x - x_k\|_{A^T A}^2 = (x - x_k)^T A^T A (x - x_k) = (b - Ax_k)^T (b - Ax_k) = \|b - Ax_k\|_2^2 = \|r_k\|_2^2$$

We can use least squares methods to minimize y in $\|r_k\|_2^2$:

$$\begin{aligned} \|r_k\|_2^2 = \|b - Ax_k\|_2^2 &= \left\|Q_{k+1} Q_{k+1}^T b - Q_{k+1} Q_{k+1}^T A Q_k y\right\|_2 \\ &= \left\|Q_{k+1}(Q_{k+1}^T b - Q_{k+1}^T A Q_k y)\right\|_2 = \left\|Q_{k+1}^T b - Q_{k+1}^T A Q_k y\right\|_2 = \left\|\|b\|_2 e_1 - H_k y\right\|_2 \end{aligned}$$

Lastly, use givens rotations to make H_k upper triangular, and then find solution y_k.

9.3 Preconditioning

CG predonditioning: Choose M SPD and define $C^2 = M$. CG preconditioning follows symmetric preconditioning, solving $CACy = Cb$ and $Cy = x$, but with tricks that only require computing MA.

- MA is similar to CAC (suffice to show $\lambda(MA) = \lambda(CAC)$)

$$MAx = \lambda x \Leftrightarrow CCACz = \lambda Cx \Leftrightarrow CACz = \lambda z, \text{ for } x = Cz$$
$$CACy = \lambda y \Leftrightarrow MACy = \lambda Cy \Leftrightarrow MAx = \lambda x, \text{ for x = Cy}$$

- $MAx = Mb \Longleftrightarrow CCAx = CCb \Leftrightarrow CACC^{-1}x = Cb$
- $C^{-1}x \in \mathcal{K}(CAC, Cb, k) \Longleftrightarrow x \in \mathcal{K}(MA, Mb, k)$

$$C^{-1}x \in \mathcal{K}(CAC, Cb, k) \Longleftrightarrow C^{-1}x = \alpha_0 Cb + \sum_{i=1}^{k} \alpha_i (CAC)^i Cb = \alpha_0 Cb + \sum_{i=1}^{k} \alpha_i CA(MA)^{i-1} CCb$$
$$\Longleftrightarrow x = \alpha_0 Mb + \sum_{i=1}^{k} \alpha_i MA(MA)^{i-1} Mb = \sum_{i=1}^{k} \alpha_i (MA)^i Mb \Longleftrightarrow x \in \mathcal{K}(MA, Mb, k)$$

10 Direct methods

Matrix storage: $A = \begin{bmatrix} 4.1 & 0 & 2.9 & 0 \\ 1.2 & -0.3 & 0 & -0.1 \\ 0 & 7.2 & 9.2 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$

Coordinate format (COO): (i, j, a_{ij}), e.g. $COO(A) = (1,1,4.1), (1,3,2.9), \ldots, (4,4,1.0)$

Compressed Sparse Row (CSR): $\{nzval, colval, rowptr\}$, e.g.,

$$CSR(A) = \begin{cases} nzval & = [4.1, 2.9, \ldots, 1.0] \text{ (nonzero values)} \\ colval & = [1, 3, \ldots, 4] \text{ (column values)} \\ rowptr & = [1, 3, 6, 8, 9] \text{ (index of nzval that starts each row)} \end{cases}$$

Compressed Sparse Column (CSC): $\{nzval, rowval, colptr\}$

Conceptualizing the graph of A, G_A: Edge $j \to i$ exists if $a_{ij} \neq 0$

10.1 Solving triangular sparse systems

Dense b: For $Lx = b$ with sparse lower triangular matrix, L, and dense vector, b: $x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right)$
Sparse b: We can improve on this code when b sparse by first determining the nonzero pattern of x (nontrivial).

$$\text{Rules for the nonzero pattern in } x\text{: } x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right) \Longrightarrow \begin{cases} b_i \neq 0 & \Rightarrow x_i \neq 0 \\ \exists j < i \text{ s.t. } l_{ij} \neq 0 \text{ and } x_j \neq 0 & \Rightarrow x_i \neq 0 \end{cases}$$

Using **graph theory**, the statements above are equivalent to saying • X is the set of nodes reachable from B, the set of nodes for which $b_i \neq 0$, on G_L • The reach of node j is all i for which there is a path $j \rightsquigarrow i$ • Use recursive backtracking ("depth-first search") to determine set X.

10.2 Cholesky factorization

Up-looking Cholesky factorization: Starting with known L', an upper $k \times k$ block of L, we can determine the $(k+1)^{st}$ row/column of L

(i) For A SPD, initialize $L' = \sqrt{a_{11}}$
(ii) For $k = 2, \ldots, n$, write top $k \times k$ block as $\begin{bmatrix} L' & 0 \\ x^T & w \end{bmatrix} \begin{bmatrix} L'^T & x \\ 0 & w \end{bmatrix} = \begin{bmatrix} A' & b \\ b^T & a \end{bmatrix}$
Solve $L'x = b$ for x, compute $w = \sqrt{a - x^T x}$, and update/return $L' = \begin{bmatrix} L' & 0 \\ x^T & w \end{bmatrix}$

Notice $L'_{k-1} l_k^T = a_k^T$, so we can leverage the same nonzero pattern relationship as above:

$$l_{ij} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{jk} l_{ik} \right) \Longrightarrow \text{For } j < i \begin{cases} a_{ij} \neq 0 & \Rightarrow l_{ij} \neq 0 \\ \exists k < j \text{ s.t. } l_{jk} \neq 0 \text{ and } l_{ik} \neq 0 & \Rightarrow l_{ij} \neq 0 \end{cases}$$

10.2.1 Elimination trees

Elimination Tree, E_T, is a graph of the first nonzero off-diagonal element in each column of L.

The elimination tree has the same reach as G_L: For any j, let i' be the smallest row index s.t., $L_{i'j} \neq 0$. We show removing $j \to i$ from G_L, with $i > i'$ does not change the reach of G_L
Proof: Consider $k \in Reach(j)$ and how/if it changes after we remove edge $j \to i$: • If i wasn't in the path of $j \rightsquigarrow k$, then the reach is unchanged • If i was in path of $j \to i \rightsquigarrow k$, the reach is still unchanged because $l_{ij} \neq 0$ and $l_{i'j} \neq 0 \Longrightarrow l_{i'i} \neq 0$, so a path, $j \rightsquigarrow k$, can still be constructed: $j \to i' \to i \rightsquigarrow k$

10.2.2 Worked example of $A \longrightarrow E_T \longrightarrow G_L$:

$$A = \begin{bmatrix} X & - & X & X & X \\ - & X & - & - & X \\ X & - & X & - & - \\ X & - & - & X & - \\ X & X & - & - & X \end{bmatrix} \longrightarrow E_T = \begin{cases} i=1: & a_{11} \neq 0 \Rightarrow ① \text{ (only looking at or below the diagonal } \forall i) \\ i=2: & ① \mid a_{22} \neq 0 \Rightarrow ② \\ i=3: & a_{31} \neq 0 \Rightarrow ① \to ③ \mid ② \\ i=4: & a_{41} \neq 0 \Rightarrow ① \to ③ \to ④ \mid ② \\ i=5: & a_{51} \neq 0 \Rightarrow ① \to ③ \to ④ \to ⑤ \leftarrow ② \Leftarrow a_{52} \neq 0 \text{ (final } E_T) \end{cases}$$

$$E_T \longrightarrow G_L : \begin{cases} i=1 : l_{11} l_{21} = a_{12} \Longleftrightarrow [X][l_{21}] = [-] & a_{12} = 0 \Rightarrow l_{21} = 0 \\ i=2 : \begin{bmatrix} X & - \\ - & X \end{bmatrix} \begin{bmatrix} l_{31} \\ l_{32} \end{bmatrix} = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} X \\ - \end{bmatrix} & a_{13} \neq 0 \Rightarrow l_{31} \neq 0 \\ i=3 : \begin{bmatrix} X & - & - \\ - & X & - \\ X & - & X \end{bmatrix} \begin{bmatrix} l_{41} \\ l_{42} \\ l_{43} \end{bmatrix} = \begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix} = \begin{bmatrix} X \\ - \\ - \end{bmatrix} & a_{14} \neq 0 \Rightarrow l_{41} \neq 0 \Longrightarrow l_{31} \neq 0 \Rightarrow l_{43} \neq 0 \\ & \text{(since 3 in reach of 1} \in G_L) \\ i=4 : \ldots \end{cases}$$