CME 302: Numerical Linear Algebra

Samuel Wong
Department of Statistics
Stanford University

Abstract

The course starts with a review of linear algebra, covering norms, properties, subspaces, eigenvalues/eigenvectors, Schur decomposition, and SVD. We then move to floating point arithmetic and error analysis for an understanding of numerical stability. Afterwards, we learn about LU and QR decomposition, and apply QR decomposition to solve least squares problems for full rank and deficient matrices. Iterative methods to find eigenvalues and then introduced, with methods to find 1 or more eigenvalues (along with compute cost). Ways to optimize or improve these algorithms are discussed. Then, iterative methods to solve linear systems are introduced (both splitting methods as well as Krylov methods where we search for a solution in the Krylov subspace). Lastly, we have a section on a taste of direct methods and an example about Up-looking Cholesky.

Contents

1 Linear Algebra Review **3**
1.1 Norms . . . 3
1.2 Basic Properties . . . 3
1.2.1 Identities . . . 3
1.2.2 Triangular Matrices . . . 3
1.2.3 Projector Matrix . . . 4
1.2.4 Orthogonal Matrices (Unitary) . . . 4
1.2.5 Projections . . . 4
1.3 4 Fundamental Subspaces . . . 5
1.4 Eigenvalues/Eigenvectors . . . 5
1.4.1 Definitions . . . 5
1.4.2 Gershgorin Disk Theorem . . . 6
1.4.3 Unitarily Diagonalizable Matrices . . . 6
1.4.4 Jordan Form . . . 6
1.5 Schur Decomposition . . . 6
1.6 Singular Value Decomposition . . . 6

2 Error Analysis **8**
2.1 Floating Point Arithmetic . . . 8
2.2 Forward/Backward Error Analysis . . . 8

3 LU Factorization **9**
3.1 Basic LU . . . 9
3.2 Pivoting . . . 9
3.3 Cholesky Factorization . . . 9

4 QR Decomposition **11**
4.1 Methods for QR Decomposition . . . 11
4.2 Least Squares Problem . . . 11

5 Iterative Methods to Find Eigenvalues **13**
5.1 Power Iteration . . . 13
5.2 Inverse Iteration . . . 13
5.3 Orthogonal Iteration . . . 13
5.4 QR Iteration . . . 14

6 Iterative Methods for Solving Linear Systems **16**
6.1 Splitting Methods . . . 16
6.1.1 Jacobi . . . 16
6.1.2 Gauss-Seidel . . . 16
6.1.3 Successive Over-Relaxation (SOR) . . . 16
6.1.4 Chebyshev Semi-Iterative Method . . . 17
6.2 Krylov Methods . . . 17
6.2.1 Conjugate Gradient . . . 18
6.2.2 A-conjugacy of search directions . . . 18
6.2.3 Conjugate Gradient Calculation . . . 18
6.2.4 GMRES . . . 19
6.3 Preconditioning . . . 19
6.3.1 Preconditioning for CG . . . 19

7 Direct Methods **20**
7.1 Matrix Storage . . . 20
7.2 Solving Triangular Sparse Systems . . . 20
7.3 Up-Looking Cholesky . . . 20
7.4 Elimination Trees . . . 21
7.5 Algorithm to go from $A \to E_T$ . . . 21
7.6 Algorithm to go from $E_T \to G_{ch}$ . . . 22

1 Linear Algebra Review

1.1 Norms

Satisfies:

1. Only zero has norm zero: $\|x\| = 0$ iff x = 0.
2. $\|\alpha x\| = |\alpha| \, \|x\|$ for any scalar α and any vector x.
3. $\|x + y\| \leq \|x\| + \|y\|$ for any vectors x,y (Triangle Inequality).

Vector Norms:
$\|x\|_1 = \sum_{i=1}^{m} |x_i|$
$\|x\|_2 = \sqrt{\sum_{i=1}^{m} |x_i|^2}$
$\|x\|_\infty = \max_{i \in [1,m]} |x_i|$
$\|x\|_p = (\sum_{i=1}^{m} |x_i|^p)^{1/p}$

$x^T y = \|x\|_2 \, \|y\|_2 \, cos\theta$

Holder's Inequality: For p, q such that $1/p + 1/q = 1$, $|x^T y| = \|x\|_p \, \|y\|_q$

Matrix Norms: $\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p$
For p-norm, $\|Ax\|_p \leq \|A\|_p \, \|x\|_p$

Frobenius norm: $\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} = \sqrt{tr(AA^H)}$

Both p-norm and Frobenius norm, $\|AB\| \leq \|A\| \, \|B\|$ (submultiplicative norms)
Also a Triangle Inequality is: $\|x\| - \|y\| \leq \|x - y\|$

1.2 Basic Properties

1.2.1 Identities

$(AB)C = A(BC)$
$(AB)^{-1} = B^{-1}A^{-1}$
$(A^{-1})^T = (A^T)^{-1}$
$(AB^T) = B^T A^T$
$A^{-1}A = AA^{-1} = I$ must be invertible to do this, otherwise singular. Invertible iff columns of A are linearly independent.

Sherman-Morrison-Woodbury: $(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1}$

Trace of matrix is invariant under change of basis:
$tr(A) = tr(P^{-1}AP)$
$tr(A(BC)) = tr((BC)A)$

$det(\alpha A) = \alpha^n det(A)$
$det(AB) = det(A) det(B)$

Matrix is singular iff $det(A) = 0$
$det(A) = det(A^T)$
$det(A^{-1} = det(A)^{-1}$

TRACE and DET ONLY FOR SQUARE!

1.2.2 Triangular Matrices

Sum of two triangular matrices is triangular.
Product of two triangular matrices is triangular.
Inverse of a triangular matrix is triangular.

If A is triangular, the eigenvalues of A appear on the diagonal. Algebraic multiplicity is the number of times it appears on the diagonal.

1.2.3 Projector Matrix

Projector matrix: Square matrix and $P^2 = P$
$\|P\|_2 \geq 1$

Orthogonal projector: $P = P^T$ and $\|P\|_2 = 1$

1.2.4 Orthogonal Matrices (Unitary)

$Q^TQ = I = QQ^T = QQ^{-1}$
$\Rightarrow Q^T = Q^{-1}$

$U^HU = UU^H = I$ same as orthogonal but for complex

$\|Qx\|_2 = \|x\|_2$ orthogonal matrix preserves the 2-norm
$\|QA\|_2 = \|A\|_2$ orthogonal matrix preserves the 2-norm
$\|QA\|_F = \|A\|_F$ orthogonal matrix preserves the Frobenius-norm

$det(Q) = \pm 1$ either rotation or reflection

1.2.5 Projections

$$P = I - UU^H$$
$$Px = (I - UU^H)x$$
$$Px = x - UU^Hx$$
$$Px = x - UU^Hx$$
$$Px = x - \begin{bmatrix} | & & | \\ u_1 & \dots & u_r \\ | & & | \end{bmatrix} \begin{bmatrix} — & u_1^H & — \\ & \vdots & \\ — & u_2^H & — \end{bmatrix} \begin{bmatrix} | \\ x \\ | \end{bmatrix}$$

$$Px = x - \begin{bmatrix} | & & | \\ u_1 & \dots & u_r \\ | & & | \end{bmatrix} \begin{bmatrix} u_1^Hx \\ \vdots \\ u_2^Hx \end{bmatrix}$$

We know $x = x_1u_1 + x_2u_2 + ... + x_ru_r + ... + x_nu_n$.

$$Px = x - \begin{bmatrix} | & & | \\ u_1 & \dots & u_r \\ | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_r \end{bmatrix}$$

since all u_i's orthogonal

$$Px = x - x_1u_1 + ... + x_ru_r$$

subtract off component u_1 through u_r

$$Px = x_{r+1}u_{r+1} + ... + x_nu_n$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots \\ \vdots & \ddots & \\ a_{K1} & & a_{KK} \end{bmatrix}$$

1.3 4 Fundamental Subspaces

$A \in \mathbb{R}^{mxn}$

1. $Range(A) = R(A)$
   - in $\mathbb{R}^m$
   - all vectors of the form Ax
2. $Kernel(A) = N(A)$
   - in $\mathbb{R}^n$
   - all vectors of the form $Ay = 0$
3. $Rowspace(A) = R(A^T)$
   - in $\mathbb{R}^n$
4. $Leftnullspace(A) = N(A^T)$
   - in $\mathbb{R}^m$

$N(A^T) = R(A)^\perp$
$dim(N(A^T)) + dim(R(A)) = m$

$N(A) = R(A^T)^\perp$
$dim(N(A)) + dim(R(A^T)) = n$

dimension of $R(A)$ = rank = dimension of $R(A^T)$

1.4 Eigenvalues/Eigenvectors

1.4.1 Definitions

$Ax = \lambda x$
$det(A) = \prod_{i=1}^n \lambda_i$
$tr(A) = \sum_{i=1}^n \lambda_i$

Diagonalizable = if A has n linearly independent eigenvectors

$A = X\Lambda X^{-1}$ (eigendecomposition)
$A^k = X\Lambda^k X^{-1}$
$f(A) = Xf(\Lambda)X^{-1}$

Eigenspace = nullspace of $A - \lambda I$
Characteristic polynomial: $p_A(x) = det(A - xI)$
Algebraic multiplicity example: $P = (x-2)^3(x-1)^2$
$\lambda = 2$ has algebraic multiplicity 3
$\lambda = 1$ has algebraic multiplicity 2

Geometric multiplicity: dimension of eigenspace spanned by the eigenvalue
If algebraic multiplicity = geometric multiplicity for each λ, then A is diagonalizable, else defective if algebraic > geometric (geometric will not exceed algebraic)

Matrix	Eigenvalues
A^{-1}	$\frac{1}{\lambda}$
A^T	λ
$-A$	$-\lambda$
$A - \mu I$	$\lambda - \mu$
A^2	λ^2
cA	$c\lambda$

1.4.2 Gershgorin Disk Theorem

$A \in \mathbb{C}^{nxn}$. For $1 \leq i \leq n$, the Gershgorin disk has center a_{ii} and radius $\sqrt{\sum_{j \neq i} |a_{ij}|}$. All eigenvalues of $A \in \mathbb{C}^{nxn}$ are located in one of its disks.

1.4.3 Unitarily Diagonalizable Matrices

Unitarily diagonalizable means that $A = Q\Lambda Q^H$
A matrix is unitarily diagonalizable iff $A^H A = AA^H$. If this condition holds we say that A is normal.

1.4.4 Jordan Form

Jordan block:

$$J = \begin{bmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda & 1 \\ 0 & \dots & \dots & 0 & \lambda \end{bmatrix}$$

All square matrices can be written $A = XJX^{-1}$, $A \in \mathbb{C}^{nxn}$.

Each Jordan block corresponds to 1 eigenvalue, but each eigenvalue can have several Jordan blocks corresponding to it. The number of times a λ appears on the diagonal is the algebraic multiplicity. The number of Jordan blocks related to λ is the geometric multiplicity.

Ex: If 2 is an eigenvalue with the geom mul = 3, alg mul = 4

$$J = \begin{bmatrix} 2 & 1 & 0 & \dots & 0 \\ 0 & 2 & 0 & & \vdots \\ \vdots & 0 & 2 & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 2 \end{bmatrix}$$

1.5 Schur Decomposition

Any square matrix $A \in \mathbb{C}^{nxn}$ can be written as $A = UTU^H$, where U is unitary, T = upper triangular, and the eigenvalues of A on diagonal of T.

Real Schur Decomposition: $A = QTQ^T$, where Q is orthogonal, T is 2x2 block upper triangular where the 2x2 block has eigenvalues that are complex conjugates.

1.6 Singular Value Decomposition

$A = U\Sigma V^T$ is the singular value decomposition.

$\sigma_1 ... \sigma_n$ are the singular values and are ordered $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$.

Columns of U are the left singular vectors. Columns of V are right singular vectors. If A is real, then U,V are real orthogonal.

$$A = U\Sigma V^T$$

$$= \left(\begin{array}{cccccc} u_1 & & u_r & u_{r+1} & & u_m \\ \vert & \cdots & \vert & \vert & \cdots & \vert \\ \vert & & \vert & \vert & & \vert \\ \multicolumn{3}{c}{\underbrace{\qquad\qquad\qquad}_{R(A)}} & \multicolumn{3}{c}{\underbrace{\qquad\qquad\qquad}_{N(A^T)}} \end{array}\right) \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} \left(\begin{array}{c|l} \text{———} & v_1^T \\ \vdots & \\ \text{———} & v_r^T \\ \text{———} & v_{r+1}^T \\ \vdots & \\ \text{———} & v_n^T \end{array}\right) \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} R(A^T) \\ \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} N(A) \end{array}$$

Norm properties:
$\|A\|_2 = \sigma_1(A)$
$\left\|A^{-1}\right\|_2 = \frac{1}{\sigma_n(A)}$
$\|A\|_F = \sqrt{\sum_i^{min\{m,n\}} \sigma_i^2}$

Condition number $\kappa(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\sigma_1}{\sigma_n}$, which we want as small as possible for stability.

Relating σ_i's with λ_i's:
$A^T A v_i = \sigma_i^2 v_i$
$AA^T u_i = \sigma_i^2 u_i$

When A is symmetric, $\sigma_i = |\lambda_i|$
When A is orthogonal, $\sigma_1 = \sigma_2 = ... = \sigma_n = 1$

Eckhart-Young: Let B $= \sum_i^r \sigma_i u_i v_i^H$. Then $\|A - B\|_2 = \sigma_{r+1}$, $\|A - B\|_F^2 = \sum_{i=r+1}^n \sigma_i^2$. B minimizes $\|A - B\|_2$ and $\|A - B\|_F$ over matrices with rank at most r.

2 Error Analysis

2.1 Floating Point Arithmetic

Values stored in a computer are floating type, which may not be stored with some numerical inaccuracy. This is because we work with base 10 and the computer stores values in base 2 (Ex: try representing 0.3 in base 2).

Floating point formula:

$\pm(1 + \sum_{i=1}^{p-1} d_i 2^{-i})2^e$, where $e =$ exponent, $p =$ precision, $1 + \sum_{i=1}^{p-1} d_i 2^{-i}$ is the significand (mantissa).

Unit roundoff $u = \frac{1}{2} *$ distance between 1 and the next floating point number. For double precision (64 bits), $u \approx 10^{-16}$.

2.2 Forward/Backward Error Analysis

Forward Error Analysis: look for bounds between computed quantity $\widetilde{f}(A, b)$ and true f(A,b). However, it is difficult to compute the forward error. Forward error is: $\left\|\widetilde{f}(x) - f(x)\right\|$

Backward Error Analysis: What problem did our algorithm actually solve? Backward error is easier to compute. $\|\widetilde{x} - x\|$

Sensitivity = Forward Error / Backward Error = $\frac{\left\|\widetilde{f}(x)-f(x)\right\|}{\|\widetilde{x}-x\|} = \frac{\|f(\widetilde{x})-f(x)\|}{\|\widetilde{x}-x\|}$

The intuition is how much an output $f(x)$ changes when x is perturbed.

Forward Error: $\left\|\widetilde{f}(x) - f(x)\right\|$
Backward Error: $\|\widetilde{x} - x\|$ where $f(\widetilde{x}) = \widetilde{f}(x)$
Sensitivity: $\frac{\left\|\widetilde{f}(x)-f(x)\right\|}{\|\widetilde{x}-x\|}$
Relative forward error: $\frac{\left\|\widetilde{f}(x)-f(x)\right\|}{\|f(x)\|}$
Relative backward error: $\frac{\|\widetilde{x}-x\|}{\|x\|}$
Relative sensitivity: $\frac{\left\|\widetilde{f}(x)-f(x)\right\|\|x\|}{\|\widetilde{x}-x\|\|f(x)\|}$

3 LU Factorization

3.1 Basic LU

$A = LU$

Want to solve $Ax = b$
$\Rightarrow LUx = b$
Solve $Ly = b$ for y, which is $O(n^2)$
$\Rightarrow y = L^{-1}b$
Solve $Ux = y$ for x, which is $O(n^2)$
$\Rightarrow x = U^{-1}y$

To get LU (using outer products):

1. $a_1^T = u_1 T$
2. Assume $l_{11} = 1$
3. $l_1 = \frac{a_1}{u_{11}}$
4. $A \leftarrow A - l_1 u_1^T$. A should now have 0's in the first row and column.
5. Keep repeating (going right and down A).

To get LU (using Gauss-Jordan):

$$G_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \\ \frac{-a_{21}}{a_{11}} & 1 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ \frac{-a_{n1}}{a_{11}} & 0 & \dots & 1 \end{bmatrix}$$

$G_{n-1}...G_2G_1A = U$, where $G_{n_1}...G_2G_1 = L^{-1}$

LU exists iff for all $k \leq n-1$ we have $det(A[1:k, 1:k]) \neq 0$. If zero pivot, our algorithm fails.

3.2 Pivoting

We may be able to permute to prevent algorithm from failing due to zero pivot or small pivots (numerical instability).

Use $PA = LU$

Example: $L_2^{-1}P_2L_1^{-1}P_1A$ is ok, can be simplified to
$\Rightarrow L^{-1}PA = U$
$\Rightarrow PA = LU$, and since P is orthogonal, so $P^TP = I$
$\Rightarrow A = P^TLU$

Can also permute columns using Q, use $PAQ^T = LU$

Example: $L_2^{-1}P_2L_1^{-1}P_1AQ_1^TQ_2^T... = U$ can be simplified to
$\Rightarrow PAQ^T = LU$

Full pivoting may be expensive so can use rook pivoting as a cheaper proxy. Rook pivoting works by searching inside row or column until we find an entry that is maximal in both its row and its column.

3.3 Cholesky Factorization

Cholesky Factorization is basically LU, but for SPD Matrices. Compared to regular LU, Cholesky reduces the computation cost/storage by 2 and also guarantees that the algorithm without pivoting (very stable!).

Definition of SPD (symmetric positive definite):

1. A is symmetric: $A = A^T$
2. A is positive definite: $x^T Ax > 0$ for all nonzero vectors x, or equivalently if all eigenvalues are strictly positive.

Properties:

1. $A = Q\Lambda Q^T$
2. If A is SPD and B is nonsingular, $B^T AB$ is also SPD.

Cholesky Factorization steps:
We want $A = LDL^T = GG^T$. As a result, $\|G\|_2^2 = \|A\|_2$.

1. $a_{11} = l_{11}^2$, which assumes $a_{11} > 0$
2. $l_{il} = \frac{a_{il}}{\sqrt{a_{11}}}$
3. $A \leftarrow A - l_1 l_1^T$. A should now have 0's in the first row and column.
4. Keep repeating (going right and down A).

How can we guarantee that we won't run into zero pivots with Cholesky? The answer is because of the SPD Lemma.

SPD Lemma: A SPD implies $X^T AX$ is SPD for any X that is full column rank.

Proof: Let $B = X^T AX$.

1. B is symmetric since $B^T = (X^T AX)^T = X^T AX$
2. WTS $z^T Bz > 0$.
   $z^T X^T AXz = (Xz)^T AXz > 0$ since Xz is a vector. This only holds if $Xz \neq 0$, which implies X must be full rank since there is no solution in the null space.

4 QR Decomposition

4.1 Methods for QR Decomposition

QR Decomposition takes a matrix A and factors it into Q (orthogonal) and R (upper triangular). If A is square, then Q and R are square. If A is skinny (n x r), there are two choices: Q can be skinny (n x r) and R square (r x r, or Q is square (n x n) and R is skinny (n x r) with zeros in rows r + 1 to n.

To perform QR decomposition, there are 3 different methods:

1. Householder (Q will be square). We are solving $Q^T A = R$. To find this Q that transforms A into an upper triangular matrix, we map x to $\|x\| e_1$. We use reflections in this process.

   $P = I - \beta v v^T$ where $v = x - \|x\| e_1$ and $\beta = \frac{2}{v^T v}$.
   We keep iterating such that $Q_{n-1}^T ... Q_1^T A = R$.

   Householder is great on dense matrices, as one large Q can transform A into R. Complexity is $O(n^2)$.

2. Givens (Q will be square) We are solving $Q^T A = R$. To find this Q that transforms A into an upper triangular matrix, we take 2 x 2 blocks of the 2 rows that we want to work on. We iterate through many Given's matrices working on 2 specific rows at a time.

$$G^T = \begin{bmatrix} g_{11} & g_{12} & 0 & \dots & 0 \\ g_{21} & g_{22} & 0 & & \vdots \\ \vdots & 0 & 1 & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix}$$

   The 2 x 2 block of G^T is

$$G^T = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

   $c = \frac{u_1}{\|u\|_2}$ and $s = \frac{-u_2}{\|u\|_2}$

   Givens rotations are great on sparse matrices, as we can pinpoint exactly where (which entry) we need to apply the Givens to. Complexity is $O(n^2)$ but can be a smaller $O(n^2)$ for sparse matrices than Householder, but will be slower for dense matrices.

3. Gram-Schmidt (can produce tall, thin Q, not just square Q)
   $a_k = \sum_{i=1}^{k} r_{ik} q_i$
   $r_{ik} = q_i^T a_k$, where $i < k$
   $r_{kk} = \left\| a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right\|$
   $q_k = \frac{a_k - \sum_{i=1}^{k-1} r_{ik} q_i}{r_{kk}}$

4.2 Least Squares Problem

Want to find $argmin_x \|Ax - b\|_2$. Three different to do this: normal equations (full rank), QR method (full rank), and SVD (for rank deficient).

1. Method of normal equations

$e = Ax - b$ is orthogonal to the range of A when minimizing the 2-norm.
$A^T(Ax - b) = 0$
Cons: A^TA is the square of the condition number A, so this can get inaccurate if A is poorly conditioned.

2. QR Method for least squares

   We know $R(A) = R(Q)$ so the error is also orthogonal to the range of Q when minimizing the 2-norm.
   $Q^T(Ax - b) = 0$
   $Q^T(QRx - b) = 0$
   $Rx = Q^Tb$
   Pros: We're now only looking at the condition number of R, which is on the order of the condition number of A.

3. SVD for rank deficient A

   Want to not only minimize $\|Ax - b\|_2$, but since there are many solutions x^* (since there are vectors in the null space since A is rank deficient), we also want to minimize $\|x\|_2$ in order to order the "best" solution x_0, where $x_0 \perp N(A)$.
   $A = U\Sigma V^T$, but we ignore the columns of U, V that correspond to zeros in Σ, therefore
   $A = \widetilde{U}\widetilde{\Sigma}\widetilde{V^T}$
   Plug this into $A^T(Ax - b) = 0$.
   $(\widetilde{U}\widetilde{\Sigma}\widetilde{V^T})^T(\widetilde{U}\widetilde{\Sigma}\widetilde{V^T}x - b) = 0$
   $(\widetilde{V}\widetilde{\Sigma}\widetilde{U^T})(\widetilde{U}\widetilde{\Sigma}\widetilde{V^T}x - b) = 0$
   $\widetilde{V}\widetilde{\Sigma}^2\widetilde{V^T}x - \widetilde{V}\widetilde{\Sigma}\widetilde{U^T}b = 0$
   $\widetilde{\Sigma}\widetilde{V^T}x = \widetilde{U^T}b$

5 Iterative Methods to Find Eigenvalues

5.1 Power Iteration

Finds the largest eigenvector and eigenvalue. First, compute the largest eigenvector x_1. Then find eigenvalue λ_1. Assume $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$. Basic idea: multiply A over and over again to a random vector z, normalize, and eventually you'll get the largest eigenvector.

A is diagonalizable, so $A = X\Lambda X^{-1}$.
$\Rightarrow A^k = X\Lambda^k X^{-1}$. Let $X^{-1} = Y$.

$A^k = \sum_{i=1}^{n} \lambda_i^k x_i y_i^H$
$A^k \approx \lambda_1^k x_1 y_1^H$
$A^k z \approx \lambda_1^k x_1 y_1^H z$; $y_1^H z$ is a constant
$A^k z \approx (\lambda_1^k y_1^H z) x_1$
$\Rightarrow \widetilde{x_1} \approx \frac{A^k z}{\|A^k z\|_2}$

Take the $\widetilde{x_1}$ and solve for λ_1. $A\widetilde{x_1} = \lambda_1 \widetilde{x_1}$
$\widetilde{x_1}^H A \widetilde{x_1} = \lambda_1$
Convergence: $O((\frac{|\lambda_2|}{|\lambda_1|})^k)$

5.2 Inverse Iteration

Finds the eigenvector and eigenvalue that is closest to a value μ. Assumes we know a μ that is close to a λ_i. Find eigenvector x_i, then use that to find eigenvalue λ_i. Basic idea: multiply $(A - \mu I)^{-1}$ over and over again to a random vector z, normalize, and eventually you'll get the eigenvector closest to eigenvalue μ. A more improved version of Inverse Iteration is Rayleigh Quotient Iteration, where at each iteration, we update our μ to the newest value of λ for faster convergence.

$(A - \mu I)^{-1}$ has same eigenvectors as A.

Perform power iteration on the matrix $(A - \mu I)^{-1}$
The $\lambda((A - \mu I)^{-1}) = \frac{1}{\lambda_i - \mu}$
$\widetilde{x_i} \approx \frac{((A-\mu I)^{-1})^k z}{\|((A-\mu I)^{-1})^k z\|_2}$

Take the $\widetilde{x_i}$ and solve for λ_i. $A\widetilde{x_i} = \lambda_i \widetilde{x_i}$
$\widetilde{x_i}^H A \widetilde{x_i} = \lambda_i$
Convergence: $O((\frac{|\lambda_i - \mu|}{|\lambda_j - \mu|})^k)$

5.3 Orthogonal Iteration

Finds all the eigenvalues and eigenvectors. The process is essentially power iteration for x_1. Then q_2 goes to $(I - x_1 x_1^T) x_2$, etc. for all q_i's. $span\{q_1, ..., q_r\} = span\{x_1, ..., x_r\}$ You get the $\lambda_1, ..., \lambda_r$. Basic idea: multiply A over and over again to an orthogonal matrix (starting with the I (identity) in iteration 1), do QR decomposition on AQ, then take that Q and use that as the orthogonal matrix for the next iteration.

$A = X\Lambda X^{-1}$
QR decomposition of X where we have first r eigenvectors. $[x_1|...|x_r] = Q^x R^x$
$span\{x_1, x_2\} = span\{q_1^x, q_2^x\}$
Also, $span\{q_1, q_2\} = span\{x_1, x_2\}$
Since $span\{q_1, q_2\} \approx span\{q_1^x, q_2^x\}$ and $q_1 \approx q_1^x$, therefore since $q_2 \perp q_1$, then $q_2 \approx \pm q_2^x$. By induction, $q_i \approx \pm q_i^x$ for $i = 1, .., r$.

$Q \Rightarrow Q^x$
$Q^T A Q \approx (Q^x)^T A (Q^x)$
$A = X\Lambda X^{-1}$
$\Rightarrow A = Q^x R^x \Lambda (R^x)^{-1} (Q^x)^T$
$\Rightarrow (Q^x)^T A (Q^x) = R^x \Lambda (R^x)^{-1}$ where $R^x \Lambda (R^x)^{-1}$ is upper triangular.

$\Rightarrow Q^T A Q = R^x \Lambda (R^x)^{-1}$
$\Rightarrow Q^T A Q = T$, which is the same as the Schur decomposition. Therefore we can read the eigenvalues of A off of the diagonal

of T once we create it by performing $Q^T AQ = T$.
Convergence: $O((\frac{|\lambda_{r+1}|}{|\lambda_r|})^k)$

5.4 QR Iteration

With orthogonal iteration, we update Q_k^T but we have to create $T_k = (Q_k^T)AQ_k$ each iteration to find the eigenvalues. So, the question is can we go directly from T_k to T_{k+1}?

Basic idea: do QR decomposition, obtain the Q and the R, multiply RQ to get the next iteration.

$T_k = Q_k^T AQ_k$
$T_{k+1} = Q_{k+1}^T AQ_{k+1}$

From orthogonal iteration, we perform $Z = AQ_k$ and we also perform QR decomposition on Z such that $Q_{k+1}R_{k+1} = qr(Z)$. Therefore $\Rightarrow Q_{k+1}R_{k+1} = AQ_k$ and also $R_{k+1}Q_k^T = Q_{k+1}^T A$. Doing math substitution, we find that
$\Rightarrow T_k = Q_k^T Q_{k+1}R_{k+1} = U_{k+1}R_{k+1}$
$\Rightarrow T_{k+1} = R_{k+1}Q_k^T Q_{k+1} = R_{k+1}U_{k+1}$.

However, this is inefficient as each QR factorization is $O(n^3)$ and if we do n iterations, then this will be $O(n^4)$! So we do 2 improvements:

1. First make A upper Hessenberg ($O(n^3)$), then run QR iteration on H. ($O(n^3)$).
   - Use Householder reflections such that $P_1 v = \|v\| e_1$.

$$Q_1 = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & p_{11} & \dots & \dots & p_{1,n-1} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & p_{n-1,1} & \dots & \dots & p_{n-1,n-1} \end{bmatrix}$$

     You have to use the Householder starting at column and row 2, otherwise, when you right-apply it, you lose the sparcity again. We need to both left and right apply so that by similar matrix property, $Q^T AQ = H$, $\lambda(A) = \lambda(H)$ (can try the multiplication yourself to see).

     After we apply $n-1$ of these matrices, we obtain that $Q^T AQ = H$ where H is upper Hessenberg! Each Householder is $O(n^2)$ (don't need to use the entire matrix, only the vectors to create it), so n Householder's is $O(n^3)$.
   - Run QR iteration on H.

     $R_k = G_{n-1}^T ... G_1^T T_{k-1}$. As before, run many Given's rotations (left apply) on T_{k_1} to get R_k. After this, we need to (right apply the Given's rotations to keep the similar matrix property).
     $T_{k+1} = R_k G_1 ... G_{n-1}$, which is $O(n^2)$.

     Because the starting A (or T) is upper Hessenberg, we only need n of them (and since each Given's is $O(n)$), so the total is $O(n^2)$.
     Assuming we perform n iterations of QR iteration, then this is also $O(n^3)$.

     In order to optimize space, we can use bulge chasing. Previously, we first left applied all the Given's and then we right applied all of them. With bulge chasing, we first apply G_1^T and G_1 on both sides of T, then do G_2 etc. However, we have a bulge until the end. (Try the math by multiplying G_1^T and G_1 on both sides of an upper Hessenberg matrix). However, at the very end of the bulge chasing, we have an H, but is it the same H as first left applying then right applying?

     We know it is the same by the Implicit Q Theorem.

     Implicit Q Theorem: If H is upper Hessenberg, and U and W are orthogonal such that $U^T HU$ and $W^T HW$ are both upper Hessenberg and unreduced - if first column of U and W are equal, then U = W up to a sign.

Proof: Can prove this by showing $U^T HU = A_1$ and $W^T HW = A_2$ creates the same A_1 and A_2. Use a Gram-Schmidt type of process to go column by column of $A_1 U^T = U^T H$ and $A_2 W^T = W^T H$.

2. QR with Shift

Helps with matrices where $|\lambda_1| = |\lambda_2|$ but different signs, or same signs with different complex parts.

$T_{k-1} - \mu_k I = U_k R_k \Rightarrow U_k^H (T_{k-1} - \mu_k I) = R_k$
plug into $T_k = R_k U_k + \mu_k I$

Therefore, $T_k = U_k^H (T_{k-1} - \mu_k I) U_k + \mu_k I$
$T_k = U_k^H T_{k-1} U_k - \mu_k I + \mu_k I$
$T_k = U_k^H T_{k-1} U_k$
$\lambda(T_k) = \lambda(T_{k-1})$

- Try to find μ_k that is an eigenvalue of A. If $\mu_k = \lambda_i$, $T_{k-1} - \mu_k I$ is singular. Then QR decompostion of $T_{k-1} - \mu_k I = U_k R_k$ has last row of R_k is zero. True only because T_k is an unreduced upper Hessenberg so $T_{k-1} - \mu_k I$ is also unreduced upper Hessenberg. Since $n-1$ first columns of $T_{k-1} - \mu_k I$ are linearly independent, this implies the last row of R_k is zero.

  Now we can use something called deflation! What we do is we remove $T_k[n, n]$ and recurse on $T_k[1 : n-1, 1 : n-1]$. This will now reduce our compute for the following iterations.

  If H is upper Hessenberg,
$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}$$
  , then $\lambda(H) = \lambda(H_{11}) \cup \lambda(H_{22})$

- If not exact that $\mu_k = \lambda_i$, keep running shifted QR iteration with deflation. If when updating $T_k = R_k U_k + \mu_k I$, any sub-diagonal element of T_k is zero(or sufficiently close to 0), write
$$T_k \approx \begin{bmatrix} (T_k)_{11} & (T_k)_{12} \\ 0 & (T_k)_{22} \end{bmatrix}$$
  , and recurse on $(T_k)_{11}$ and $(T_k)_{22}$ simultaneously.

6 Iterative Methods for Solving Linear Systems

6.1 Splitting Methods

Use iterative methods over direct methods when:

- matrix A is very large and direct methods are computationally expensive
- A is sparse and Ax is easily computed.

Separate $A = M - N$, and therefore when solving $Ax = b$, we get
$x^{k+1} = M^{-1}Nx^k + M^{-1}b$.

The convergence of this depends on $M^{-1}N$. The error e at each step is:
$e^{k+1} = x^{k+1} - x$
$e^{k+1} = M^{-1}Nx^k + M^{-1}b - (M^{-1}Nx + M^{-1}b)$
$e^{k+1} = M^{-1}Nx^k - M^{-1}Nx$
$e^{k+1} = M^{-1}Ne^k$
$e^{k+1} = (M^{-1}N)^k e^0$

Theorem: Convergence happens only when the spectral radius of $M^{-1}N$, $\rho(M^{-1}N) < 1$, where the spectral radius of the matrix is the absolute value of the largest eigenvalue.

Proof: Let $Gv = \lambda v$, where $G = M^{-1}N$.
Pick $x^0 = x + v$ as a first guess.
$G^k e^0 = G^k(x^0 - x) = G^k v = \lambda^k v$, which means that if $|\lambda| < 1$ as $k \to \infty$, the error goes to 0.

6.1.1 Jacobi

Let $A = D - L - U$, where $M = D$, and $N = L + U$; (L, U have zeros on the diagonal).
$Dx^{k+1} = (L + U)x^k + b$
Converges for any guess x^0 when A is strictly diagonally dominant: $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$

As usual, let $G = M^{-1}N$. We can prove this using Gershgorin disc theorem which says that all eigenvalues lie in $D_i = \{z \in \mathbb{C} | |z - g_{ii}| \leq \sum_{j \neq i} |g_{ij}|\}$. We know that all $g_{ii} = 0$ since the diagonal of G is 0.
$\sum_{j \neq i} |g_{ij}| = \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} < 1$.

6.1.2 Gauss-Seidel

Let $A = D - L - U$, where $M = D - L$, and $N = U$; (L, U have zeros on the diagonal).
$(D - L)x^{k+1} = Ux^k + b$

Theorem: Householder-John Theorem If real matrices A and $A - B - B^T$ are SPD, and B is real, then $\rho(H)$ where $H = (A - B)^{-1}B$ satisfies $\rho(H) < 1$.

Use Householder-John to prove convergence of Gauss-Seidel for SPD A.
Choose $A = D - L - U$, $B = -U = -L^T \Rightarrow A - B - B^T = (D - L - U) - (-U) - (-L) = D$ is SPD.
$H = [(D - L - U) - (-U)]^{-1}(-U) = -(D - L)^{-1}U = -G$, so $\rho(G) < 1$ so convergence.

6.1.3 Successive Over-Relaxation (SOR)

Use Gauss-Seidel but try to accelerate convergence using ω.
$Dx^{k+1} = b + Lx^{k+1} + Ux^k$ (from Gauss-Seidel)
$x^{k+1} = x^k + [D^{-1}(b + Lx^{k+1} + Ux^k) - x^k]$
$x^{k+1} = x^k + \Delta x^k$

Choose $0 < \omega < 2$ so that
$x^{k+1} = x^k + \omega[D^{-1}(b + Lx^{k+1} + Ux^k) - x^k]$
$\Rightarrow (D - \omega L)x^{k+1} = \omega b + [(1 - \omega)D + \omega U]x^k$ (SOR)

Theorem: If $\omega \notin (0, 2)$, there exists a guess x^0 so that SOR will not converge. Conversely if A is SPD, SOR converges for all $\omega \in (0, 2)$

Proof: $det(M^{-1}N) = det(G) = \frac{det[(1-\omega)D+\omega U]}{det(D-\omega L)}$ from the SOR formula.

$det(G) = \frac{\prod_{i=1}^{n}(1-\omega)d_{ii}}{\prod_{i=1}^{n} d_{ii}}$, since L, U are triangular with 0's on diagonal. $det(G) = (1-\omega)^n$
$|1-\omega| \leq |\lambda_{max}(G)|$ since $det(G)$ is the product of eigenvalues.
$|1-\omega| < 1$ is the only time when convergence is possible.

Use Householder-John to prove converse.
Pick $A = \omega A$, $B = (\omega - 1)D - \omega U$
ωA is SPD and $A - B - B^T = \omega A - [(\omega-1)D - \omega U] - [(\omega-1)D - \omega U]^T$
$\Rightarrow A - B - B^T = (2-\omega)D$ using $A = D - L - U$ and $U^T = L$

$(A-B)^{-1}B = (\omega A - [(\omega-1)D - \omega U])^{-1}[(\omega-1)D - \omega U]$
$(A-B)^{-1}B = (D - \omega L)^{-1}[(\omega-1)D - \omega U]$
$(A-B)^{-1}B = -G$
$\rho(G) < 1$

6.1.4 Chebyshev Semi-Iterative Method

$x^{k+1} = x^k + \omega_k([M^{-1}b + Gx^k] - x^k)$
$e^k = (I - \omega_{k-1}M^{-1}A)e^{k-1} = (I - \omega_{k-1}M^{-1}A)...(I - \omega_0 M^{-1}A)e^0$
$q_k(x) = (1-\omega_{k-1}x)(1-\omega_{k-2}x)...(1-\omega_0 x)$
We want to minimize $\|q_k(M^{-1}A)e^0\|$ where $q_k(0) = 1$.

If M, A are SPD, then $M^{-1}A$ has positive eigenvalues because we can pick $M = GG^T \Rightarrow G^{-1}M = G^T$ and then $G^T M^{-1}A(G^T)^{-1} = G^{-1}MM^{-1}A(G^T)^{-1} = G^{-1}A(G^T)^{-1}$, so by similar matrix property, $M^{-1}A$ has positive eigenvalues.

$M^{-1}A$ diagonalizable so then $q_k(M^{-1}A) = Xq_k(\Lambda)X^{-1} =$

$$X \begin{bmatrix} q_k(\lambda_1) & 0 & \dots & \dots & 0 \\ 0 & q_k(\lambda_2) & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & q_k(\lambda_n) \end{bmatrix} X^{-1}$$

so $\|q_k(M^{-1}A)e^0\| \leq max_\lambda q_k(\lambda)\|e^0\|$, so we want to find a polynomial q_k small on the eigenvalues of $M^{-1}A$ to bound $\|e^k\|$.

Assume $M^{-1}A$ has positive and real eigenvalues contained in $[\alpha, \beta]$, then the error in Chebyshev semi-iterative method with step sizes $\omega_0, \omega_1, ..., \omega_{k-1}$ satisfies

$\|e^k\| \leq \frac{\|e^0\|}{T_k(\frac{\beta+\alpha}{\beta-\alpha})}$ where $T_k(x) = cos(kcos(x)^{-1})$

6.2 Krylov Methods

Basic idea: We want to find iterative techniques to solve $Ax = b$. We cleverly realize that we can look for solutions in increasing dimensions of the Krylov subspace in order to find a close solution.

Iterative technique to solve $Ax = b$.
Start with $Mx^k = b + Nx^{k-1}$ and choose $M = I$ assuming A is close to I. $\Rightarrow Ix^k = x^k = b + (I-A)x^{k-1}$. Therefore $x \in \mathcal{K}(A, b, k) = span\{b, Ab, ..., A^{k-1}b\}$.

Want to minimize $r^k = b - Ax^k$. Also $x - x^k$.
Let Q_k be orthogonal basis for $\mathcal{K}(A, b, k)$, then $x^k = Q_k y$ and $r^k = b - AQ_k y$.

Matrix Type	Method	Minimize r^k in what norm?	Orthogonality Condition
SPD	Conjugate Gradient	A^{-1}	$r^k \perp \mathcal{K}(A,b,k)$
Symmetric	MINRES	2	$r^k \perp A\mathcal{K}(A,b,k)$
General	GMRES	2	$r^k \perp A\mathcal{K}(A,b,k)$

6.2.1 Conjugate Gradient

Basic idea: We want to find an iterative way to solve $Ax = b$ for SPD matrices. We look to minimize the residual in the A^{-1} norm.

Minimize $\left\|r^k\right\|_{A^{-1}}^2$
$(b - Ax^k)^T A^{-1}(b - Ax^k)$
$(b - Ax^k)^T A^{-1}(b - Ax^k)$
$(b - Ax^k)^T A^{-1}AA^{-1}(b - Ax^k)$, and A is SPD so $A^{-1} = (A^{-1})^T$
$(A^{-1}b - A^{-1}Ax^k)^T A(A^{-1}b - A^{-1}Ax^k)$, and $A^{-1}b = x$
$(x - x^k)^T A(x - x^k)$
$\left\|x - x^k\right\|_A^2$, which we can rewrite as $(x - Q_k y)^T A(x - Q_k y)$, and when we minimize (by taking the gradient of that expression and setting that to 0), we get that $Q_k^T AQ_k y = Q_k^T b$. We can see that $Q_k^T AQ_k$ is the T_k from Lanzcos and $Q_k^T b = b\left\|e_1\right\|$ since we pick the first column of Q_k to be b.
Since $Q_k^T(AQ_k y - b) = 0$, therefore $r^k \perp Q_k$.

6.2.2 A-conjugacy of search directions

We really want to make sure of two things. The first is that we're always moving in directions orthogonal to directions we have searched before. This gives us the optimal path toward the minimum. The second is that we're always searching in a different dimension (that the Krylov subspace is growing), because if the subspace isn't growing, we have already found the optimal solution.

Theorem: $(\Delta x^k)^T A\Delta x^l = 0$, where $k \neq l$

Proof: $r^k - r^{k+1} = (b - Ax^k) - (b - Ax^{k+1}) = Ax^{k+1} - Ax^k = A\Delta x^k$.
We know $r^k \perp Q_k$ and $r^{k+1} \perp Q_{k+1}$ so $A\Delta x^k \perp Q_k$. ($\Delta x^k = x^{k+1} - x^k$).
$\Delta x^l = x^{l+1} - x^l \in span\{Q_{l+1}\}$.
Let $l \leq k+1$, then $A\Delta x^k \perp \Delta x^l$ so $(\Delta x^l)^T(A\Delta x^k = 0)$. Since A is SPD, $(\Delta x^l)^T(A^T)\Delta x^k = 0)$, so $\Delta x^k \perp A\Delta x^l$. Therefore, $(\Delta x^k)^T A\Delta x^l = 0$; where $k \neq l$

Theorem: $span\{r^0...r^k\} = span\{\Delta x^0...\Delta x^k\} = span\{b, Ab...A^k b\} = span\{q^1...q^{k+1}\} = span\{p^1...p^{k+1}\}$ if we have not converged.

Proof: 1) Show $span\{\Delta x^0...\Delta x^k\} = span\{b, Ab...A^k b\}$. We know $\Delta x^k \subseteq \mathcal{K}(A,b,k+1)$ since $\Delta x^k = x^{k+1} - x^k$. Suppose $\Delta x^k \in \mathcal{K}(A,b,k)$, then $x^{k+1} = x^k$ so $\Delta x^k = 0$ which means we have converged. This is a contradiction since we have not converged, so we know $\Delta x^k \in \mathcal{K}(A,b,k+1)$.
2) Show $span\{r^0...r^k\} = span\{b, Ab...A^k b\}$.
$r^k = b - Ax^k \in \mathcal{K}(A,b,k+1)$ since $x^k \in \mathcal{K}(A,b,k)$. But we know that $r^k \perp \mathcal{K}(A,b,k)$, so r^k must be $\in \mathcal{K}(A,b,k+1)$.

Theorem: $Ap^k \perp p^l$, where $l \neq k$.

Proof: Start with result from A-conjugacy for Δx, where we know $A\Delta x^k \perp \Delta x^l$, where $l \neq k$. We also have defined $\Delta x^k = \mu^{k+1}p^{k+1}$. Therefore, $Ap^{k+1} \perp p^{l+1}$ and $\Rightarrow Ap^k \perp p^l$, where $l \neq k$.

Theorem: $r^k \perp Ap^l$, where $l \leq k-1$.

Proof: We know $p^l \in \mathcal{K}(A,b,l)$, so $Ap^l \in \mathcal{K}(A,b,l+1)$. We also know $r^k \perp \mathcal{K}(A,b,k)$, so $r^k \perp Ap^l$, where $l \leq k-1$.

6.2.3 Conjugate Gradient Calculation

We know that $span\{r^0...r^k\} = span\{p^1...p^{k+1}\}$, so we can write $R = PU$, where R is the matrix of $\{r^0...r^k\}$, P is the matrix of $\{p^1...p^{k+1}\}$, and U is upper triangular and contains the scalars. The goal is to find x^{k+1} and r^{k+1}, when we are given r^k, x^k, p^k, and A.

Since we proved above that $r^k \perp Ap^l$, where $l \leq k-1$, when looking at $R = PU$, we can write $r^k = p^{k+1} - \tau_k p^k$ (all the other $p^{k-1}...p^1$ go to 0).
Multiply by $(p^k)^T A$ to get
$(p^k)^T Ar^k = (p^k)^T Ap^{k+1} - \tau_k (p^k)^T Ap^k$
$\Rightarrow (p^k)^T Ar^k = -\tau_k (p^k)^T Ap^k$ (other term is 0 since $p^k \perp p^{k+1}$)
$\Rightarrow \tau_k = -\frac{(p^k)^T Ar^k}{(p^k)^T Ap^k}$

Use τ_k to calculate p^{k+1}
$p^{k+1} = r^k + \tau_k p^k$ (from above)

$\Delta x^k = x^{k+1} - x^k = \mu_{k+1} p^{k+1}$
$\Rightarrow Ax^{k+1} - Ax^k = \mu_{k+1} Ap^{k+1}$
$\Rightarrow r^k - r^{k+1} = \mu_{k+1} Ap^{k+1}$
$\Rightarrow (p^{k+1})^T (r^k - r^{k+1}) = \mu_{k+1} (p^{k+1})^T Ap^{k+1}$
$\Rightarrow (p^{k+1})^T r^k = \mu_{k+1} (p^{k+1})^T Ap^{k+1}$
$\Rightarrow (r^k + \tau_k p^k)^T r^k = \mu_{k+1} (p^{k+1})^T Ap^{k+1}$
$\Rightarrow (r^k)^T r^k = \mu_{k+1} (p^{k+1})^T Ap^{k+1}$
$\Rightarrow \mu_{k+1} = \frac{(r^k)^T r^k}{(p^{k+1})^T Ap^{k+1}}$

Therefore $x^{k+1} = x^k + \mu_{k+1} p^{k+1}$
$r^{k+1} = r^k - \mu_{k+1} Ap^{k+1}$

Convergence of CG: $\left\| x - x^k \right\|_A = \left\| r^k \right\|_A^{-1} \leq 2 \left\| r^0 \right\|_A^{-1} (\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1})^k$

6.2.4 GMRES

Basic idea: We want to find an iterative way to solve $Ax = b$ for general non-singular matrices (does not have to be SPD). We look to minimize the residual in the 2-norm.

Minimize $\left\| r^k \right\|_2 = \left\| \beta_0 e_1 - T_k y^k \right\|_2$ where T_k is upper Hessenberg.

Convergence of GMRES: $\left\| r^k \right\|_2 \leq \kappa(X) min_{p(0)=1, p of degree k} max_{\lambda evalue of A} |p(\lambda)|$, where $\kappa(X)$ is the condition number of X in $A = X \Lambda X^{-1}$

6.3 Preconditioning

Condition number and distribution of eigenvalues of A are important when determining convergence of iterative methods.

If A is ill-conditioned, we can precondition A using preconditioners.

Left preconditioning: $(M_1 A)x = M_1 b$
Right preconditioning: $(AM_2)z = b$, and solve $x = M_2 z$
Symmetric preconditioning: $(M_1 AM_2)z = M_1 b$, and solve $x = M_2 z$

Tradeoff for good preconditioner between:
1) Cost of applying M_1 (and/or M_2) such as $A^{-1} = M_1$
2) Computational savings resulting from reduction in number of iterations.

6.3.1 Preconditioning for CG

We use CAC to maintain symmetry so
$CACz = Cb$, where $Cz = x$
A lot of times we only have $M = C^2, not C$.

Then the preconditioned problem is $MAx = Mb$ since MA is similar to CAC, and solving $MAx = Mb$ = equivalent to solving $(CAC)C^{-1}x = Cb$. Then $C^{-1}x \in \mathcal{K}_k((CAC), Cb)$ iff $x \in \mathcal{K}_k(MA, Mb)$.

7 Direct Methods

If we want to not use iterative methods, but to use direct methods, here is a taste of what we could do. An example is provided in this section regarding doing a Cholesky Factorization using a direct method.

7.1 Matrix Storage

1. Coordinate format: (i, j, a_{ij})
   Not great for operations such as multiplying Ax.

2. Compressed Sparse Row (CSR)

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

   nzval = [4.1, 2.9, ..., 1.0]; nonzero values
   colval = [1, 3, ..., 4]; column values
   rowptr = [1, 3, 6, 8 , 9]; where each row starts/ends

3. Compressed Sparse Column (CSC)
   nzval = [4.1, 1.2, ..., 1.0]; nonzero values
   rowval = [1, 2, ..., 4]; row values
   colptr = [1, 3, 5, 7, 9]; where each column starts/ends

   Turning a sparse matrix into a graph: there is a directed edge (i, j) from vertex i to j iff $a_{ji} \neq 0$.

7.2 Solving Triangular Sparse Systems

1. L sparse, b dense: $Lx = b$
   Loop through nonzero entries of L row by row and solve.

2. L sparse, b sparse: $Lx = b$
   Let x be the solution to $Lx = b$. For any i so that x_i is non-zero, either:

   - $b_i \neq 0$ OR
   - There exists some $j < i$ so that $l_{ij} \neq 0$ and $x_j \neq 0$.

   In a directed graph G, we say that a vertex i is reachable from a vertex j if there is a directed path from j to i in G. (That is, if there is a sequence of edges $(j, i_1), (i_1, i_2), ..., (i_{k-1}, i_k), (i_k, i)$ where all edges on a graph). The set of vertices i reachable from a vertex j is called the reach of j.

   To solve $Lx = b$ now:

   (a) First, run DFS to find the set X.
   (b) Then run a modified version of algorithm to solve $Lx = b$ when L is sparse, but only update X_j for $j \in X$.

7.3 Up-Looking Cholesky

We have a matrix A that is SPD. We want to do Cholesky factorization such that $A = LL^T$. Assume you have already solved kxk of L, we want to iterate to find the next row/column. Let L' be the block of L and L'^T be the block of L^T that has already been solved. We want to find x and w, which would be the next row/column of the Cholesky factorization.

$$\begin{bmatrix} L' & 0 \\ x^T & w \end{bmatrix} \begin{bmatrix} L'^T & x \\ 0 & w \end{bmatrix} = \begin{bmatrix} A' & b \\ b^T & a \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} L'L'^T & L'x \\ L'^T x^T & x^T x + w^2 \end{bmatrix} = \begin{bmatrix} A' & b \\ b^T & a \end{bmatrix}$$

$\Rightarrow L'x = b$
$\Rightarrow x^T x + w^2 = a \Rightarrow w = \sqrt{a - x^T x}$

Typical $Lx = b$ solve:

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ \vdots & l_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ l_{n1} & \dots & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

Therefore, $\sum_{j \le i} l_{ij} x_j = b_i$
$\Rightarrow x_i = \frac{1}{l_{ii}}(b_i - \sum_{j<i} l_{ij} x_j)$
Therefore, x_i may be $\neq 0$ when either 1) $b_i \neq 0$ OR 2) there exists j such that $l_{ij} \neq 0$ and $x_j \neq 0$.

With Up-Looking Cholesky we solve:

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ \vdots & l_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ l_{i-1,1} & \dots & \dots & l_{i-1,i-1} \end{bmatrix} \begin{bmatrix} l_{i1} \\ \vdots \\ \vdots \\ l_{i,i-1} \end{bmatrix} = \begin{bmatrix} a_{i1} \\ \vdots \\ \vdots \\ a_{i,i-1} \end{bmatrix}$$

Therefore, $a_{ji} = a_{ij} = \sum_{k \le j} l_{jk} l_{ik}$; where $k \le j \le i$
$\Rightarrow l_{ij} = \frac{1}{l_{ii}}(a_{ij} - \sum_{k<j} l_{jk} l_{ik})$
Therefore, l_{ij} may be $\neq 0$ when either 1) $a_{ij} \neq 0$ OR 2) there exists $k < j$ such that $l_{jk} \neq 0$ and $l_{ik} \neq 0$.

7.4 Elimination Trees

We want to look at A and see the sparcity pattern of L. The steps are to go from $A \to E_T \to G_{ch} \to L$. What we have computed above is the G_{ch}, which has a 1-1 correspondence with the entries in L. What we next want to show is that the reach under E_T is the same as the reach of G_{ch}. Then we can remove all elements below the first entry below the diagonal in each column and keep the same reach.

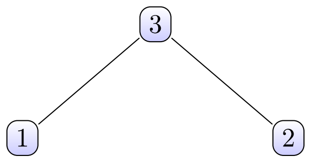
Theorem: E_T has the same reach as G_{ch}.
For any column number j, let i' be the smallest element such that $L_{i'j} \neq 0$. Let us show that removing the edge $j \to i$ with $i > i'$ such that $L_{ij} \neq 0$ does not change the reach.

Proof: We use the theorem that if $L_{ij} \neq 0$ and $L_{i'j} \neq 0$, then $L_{ii'} \neq 0$. If I remove the $j \to i$ from G_{ch}, let k be an element that was in the reach of j. Either we had a path $j \to k_1 \to ... \to k$ (i is not in the path), OR we had a path $j \to i \to k_1 \to ... \to k$. Now we still have the path $j \to i' \to i \to k_1 \to ... \to k$.

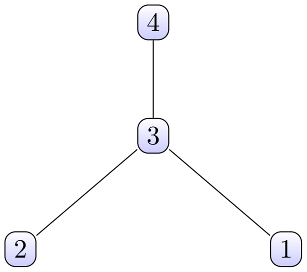
7.5 Algorithm to go from $A \to E_T$

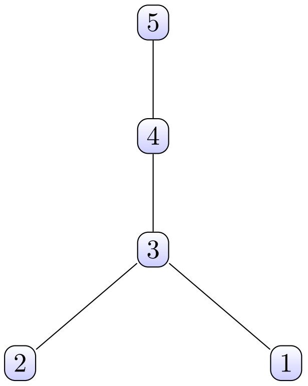
$$Example : A = \begin{bmatrix} X & 0 & 0 & 0 & 0 \\ 0 & X & 0 & 0 & 0 \\ X & X & X & 0 & 0 \\ 0 & X & 0 & X & 0 \\ X & 0 & 0 & 0 & X \end{bmatrix}$$

1. Row 1 has 1 entry
2. Row 2 has 1 entry



3. Row 3 has an element at $l_{3,2}$ and $l_{3,1}$

4. Row 4 has an element at $l_{4,2}$ so climb up the tree. 

5. Row 5 has element at $l_{5,1}$ so climb up the tree. 

7.6 Algorithm to go from $E_T \rightarrow G_{ch}$

Theorem 1: Using the tree directly above, if $l_{5,3} \neq 0$, then $l_{5,4} \neq 0$, but we don't know about $l_{5,1}$.
Theorem 2: If j is a leaf, i is the parent, $a_{ij} \neq 0$.

Example (using tree above):

1. $l_{3,1}, l_{3,2} \neq 0$ from Theorem 2.
2. $l_{4,2} \neq 0$ since $a_{4,2} \neq 0$.
3. $l_{4,3} \neq 0$ from Theorem 1 since $l_{4,2} \neq 0$.
4. $l_{5,1} \neq 0$ since $a_{5,1} \neq 0$.
5. $l_{5,3}, l_{5,4} \neq 0$ from Theorem 1 since $l_{5,1} \neq 0$

Definitions:

Leaf: nothing under
Parent: directly above
Ancestor: anywhere on top
Descendant: anywhere below
Root: the highest node on top