ECE 595, Section 10
Numerical Simulations
Lecture 5: Linear Algebra

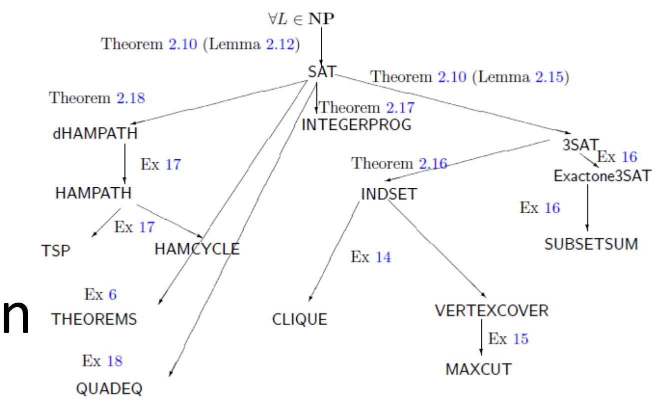
Prof. Peter Bermel

January 16, 2013

Outline

- Recap from Monday
- Overview: Computational Linear Algebra
- Gaussian Elimination
- LU Decomposition
- Singular Value Decomposition
- Sparse Matrices
- QR Decomposition

Recap from Monday



- $\mathrm{P} \subseteq \mathrm{NP} \subseteq \cup_{c \geq 1} \mathbf{DTIME}\left(2^{n^c}\right)$
- **NP**-complete problems:
  - Can be Karp-reduced (via Cook-Levin theorem) to hardest general problem in Conjunctive Normal Form: SAT
  - Only soluble in polynomial time if **P=NP**
- Unknown whether **P=NP**
  - If not, some problems will remain quite difficult: can use heuristics to compensate
  - If so, impressive applications may be possible

Overview: Computational Linear Algebra

- Broadly speaking: the solution of matrix problems, such as $A \cdot x = b$ or A^{-1}
- Unique solutions not always guaranteed
  - May have mismatch of equations & unknowns
  - Possible degeneracy (aka singularity)
  - Near degeneracies → large round-off errors
- On the other end of the spectrum, some special features can help
  - Sparse values
  - Banded diagonals

Gauss-Jordan Elimination: No Pivoting

- Based on transforming $A \cdot x = b$ to $x = A^{-1}b$
- Without pivoting:
  - Normalize element on diagonal to unity
  - Subtract later columns
- Potential problems:
  - Element on diagonal is zero
  - Element on diagonal is near zero

Gauss-Jordan Elimination: Pivoting

- Pivoting allows one more flexibility
  - Interchange rows (partial pivoting)
  - Interchange rows and columns (full pivoting)
- Goal: put biggest element on diagonal
- Caveat: some (artificial?) exceptions
- Solution: implicit pivoting

Gaussian Elimination

- Reduce original matrix to partially empty (e.g., on lower left) – to be called U

$$\begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

- Perform backsubstitution to find the solution:

$$x_i = \frac{1}{a'_{ii}} \left[b'_i - \sum_{j=i+1}^{N} a'_{ij} x_j \right]$$

LU Decomposition

- Rewrite input matrix A as a product of lower-triangular and upper-triangular matrices, i.e.,

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

- Then solve with $L \cdot y = b$ (forward substitution)

$$y_i = \frac{1}{\alpha_{ii}} \left[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j \right]$$

- Finally, solve via $U \cdot x = y$ (backsubstitution)

$$x_i = \frac{1}{\beta_{ii}} \left[y_i - \sum_{j=i+1}^{N} \beta_{ij} x_j \right]$$

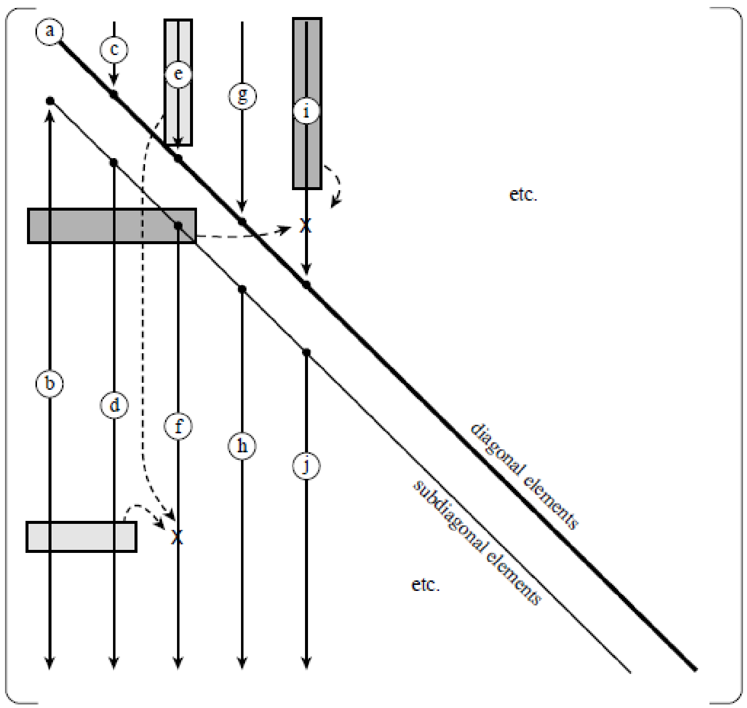
LU Decomposition

- To construct the LU matrices, use Crout's algorithm to compute the decomposition in place:

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj}$$

$$\alpha_{ij} = \frac{1}{\beta_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} \alpha_{ik}\beta_{kj}\right)$$

$$\begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ \alpha_{21} & \beta_{22} & \beta_{23} & \beta_{24} \\ \alpha_{31} & \alpha_{32} & \beta_{33} & \beta_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \beta_{44} \end{bmatrix}$$



LU Decomposition

- Execution time:
  - total number of elements computed: N^2
  - Total number of operations per element (average): N/3
- Inversion: backsubstitute after factorization
- Determinant of an LU factorization:

$$\det = \prod_{j=1}^{N} \beta_{jj}$$

Band Diagonal Matrices

- Band diagonal means only k diagonals are non-zero
- Common special case: tridiagonal:

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & & & \\ a_2 & b_2 & c_2 & \cdots & & & \\ & & & \cdots & & & \\ & & & \cdots & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & \cdots & 0 & a_N & b_N \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_{N-1} \\ r_N \end{bmatrix}$$

- Can perform LU decomposition in kN operations

Iterative Improvement

- If our exact solution is $A \cdot x = b$
- And we already have $A \cdot x' = b'$
- Then since $A \cdot x' - A \cdot x = b' - b$
- We can subtract $A^{-1}(b' - b)$ from x'
- Can repeat until reach limits of precision
- This can be formalized and used to devise certain useful guesses for our starting point b'

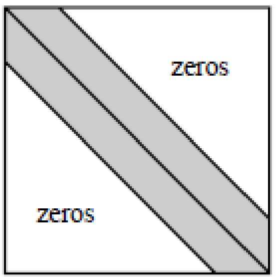
Singular Value Decomposition

- Based on theorem: any matrix $A = U \cdot W \cdot V$, where:

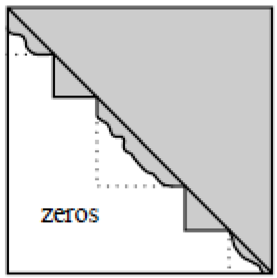
$$\begin{pmatrix} \\ & \mathbf{A} & \\ \\ \end{pmatrix} = \begin{pmatrix} \\ & \mathbf{U} & \\ \\ \end{pmatrix} \cdot \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \cdots & \\ & & \cdots & \\ & & & w_N \end{pmatrix} \cdot \begin{pmatrix} & \mathbf{V}^T & \end{pmatrix}$$

- U and V are both orthogonal: $U^T U = 1; V^T V = 1$
- Inversion is easy: $A^{-1} = V \cdot \frac{\mathbf{1}}{W} \cdot U^T$
- Condition number is set by max w_j/min w_j

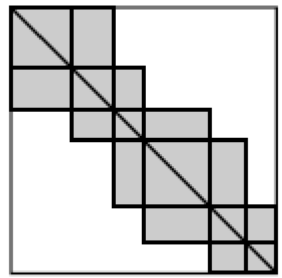
Sparse Linear Systems

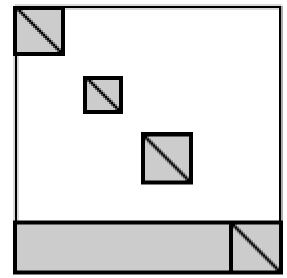


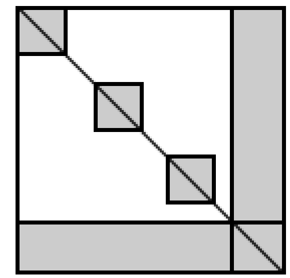
Band diagonal

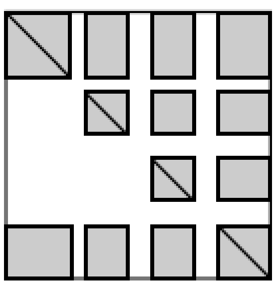


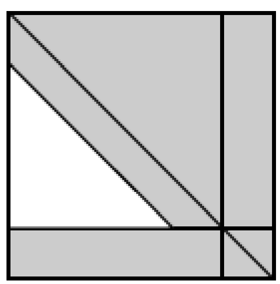
Block triangular

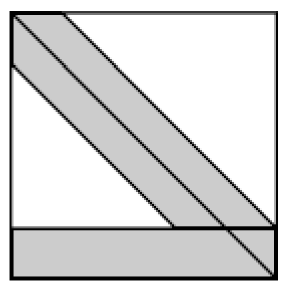

Block tridiagonal

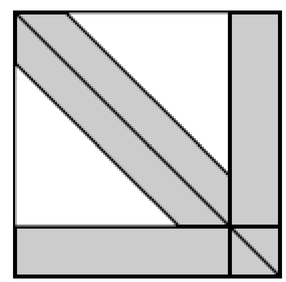

Single-bordered block diagonal

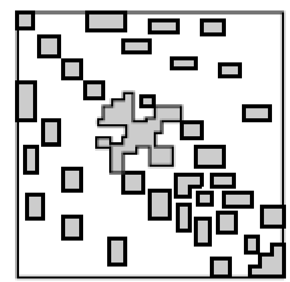

Double-bordered block diagonal


Single-bordered block triangular


Bordered band-triangular


Single-bordered band diagonal


Double-bordered band diagonal


Other

Vandermonde Matrices

- To solve the problem of moments, construct Vandermonde matrices which look like:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^{N-1} \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Cholesky Decomposition

- Can be thought of as taking the square root of a matrix A, such that $A = LL^T$
- Writing out explicitly yields following equations:

$$L_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{1/2}$$

$$L_{ji} = \frac{1}{L_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} L_{ik} L_{jk} \right)$$

QR Decomposition

- Write $A = Q \cdot R$, where $Q^T Q = 1$, and *R* is upper triangular
- Can then solve $R \cdot x = Q^T b$
- Matrix from a series of Householder transformation, s.t. $Q = \prod_i Q_i$
  – each $Q_i = 1 - 2w \cdot w^T$
  – Choose vector *w* to eliminate off-diagonal entries in one row + one column

Next Class

- Discussion of root finding and optimization
- Please read Chapter 10 of “Numerical Recipes” by W.H. Press *et al*.