• For one R.H.S., how many operations?

• For each of n rows:
  – Do n times:
    • For each of n+1 columns:
      – One add, one multiply

• Total = $n^3+n^2$ multiplies, same # of adds

• Asymptotic behavior: when n is large, dominated by $n^3$
Faster Algorithms

- Our goal is an algorithm that does this in $\frac{1}{3} n^3$ operations, and does not require all R.H.S. to be known at beginning
- Before we see that, let’s look at a few special cases that are even faster
Tridiagonal Systems

- Common special case:

\[
\begin{bmatrix}
  a_{11} & a_{12} & 0 & 0 & \cdots & b_1 \\
  a_{21} & a_{22} & a_{23} & 0 & \cdots & b_2 \\
  0 & a_{32} & a_{33} & a_{34} & \cdots & b_3 \\
  0 & 0 & a_{43} & a_{44} & \cdots & b_4 \\
  \vdots & \vdots & \vdots & \ddots & \vdots
\end{bmatrix}
\]

- Only main diagonal + 1 above and 1 below
Solving Tridiagonal Systems

- When solving using Gaussian elimination:
  - Constant # of multiplies/adds in each row
  - Each row only affects 2 others

\[
\begin{bmatrix}
  a_{11} & a_{12} & 0 & 0 & \cdots & | & b_1 \\
  a_{21} & a_{22} & a_{23} & 0 & \cdots & | & b_2 \\
  0 & a_{32} & a_{33} & a_{34} & \cdots & | & b_3 \\
  0 & 0 & a_{43} & a_{44} & \cdots & | & b_4 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & | & \vdots \\
\end{bmatrix}
\]
Running Time

• 2n loops, 4 multiply/adds per loop (assuming correct bookkeeping)
• This running time has a fundamentally different dependence on $n$: linear instead of cubic
  – Can say that tridiagonal algorithm is $O(n)$ while Gauss-Jordan is $O(n^3)$
• In general, a banded system of bandwidth $w$ requires $O(wn)$ storage and $O(w^2n)$ computations.
Big-O Notation

• Informally, $O(n^3)$ means that the dominant term for large $n$ is cubic.

• More precisely, there exist a $c$ and $n_0$ such that

\[
\text{running time} \leq c \cdot n^3
\]

if

\[n > n_0\]

• This type of asymptotic analysis is often used to characterize different algorithms.
Triangular Systems are nice!

- Another special case: lower-triangular

\[
\begin{bmatrix}
    a_{11} & 0 & 0 & 0 & \cdots & b_1 \\
    a_{21} & a_{22} & 0 & 0 & \cdots & b_2 \\
    a_{31} & a_{32} & a_{33} & 0 & \cdots & b_3 \\
    a_{41} & a_{42} & a_{43} & a_{44} & \cdots & b_4 \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots
\end{bmatrix}
\]
Triangular Systems

- Solve by forward substitution

\[
\begin{bmatrix}
a_{11} & 0 & 0 & \cdots & \mid & b_1 \\
a_{21} & a_{22} & 0 & 0 & \cdots & b_2 \\
a_{31} & a_{32} & a_{33} & 0 & \cdots & b_3 \\
a_{41} & a_{42} & a_{43} & a_{44} & \cdots & b_4 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}
\]

\[x_1 = \frac{b_1}{a_{11}}\]
Triangular Systems

- Solve by forward substitution

\[
\begin{bmatrix}
  a_{11} & 0 & 0 & 0 & \cdots & b_1 \\
  a_{21} & a_{22} & 0 & 0 & \cdots & b_2 \\
  a_{31} & a_{32} & a_{33} & 0 & \cdots & b_3 \\
  a_{41} & a_{42} & a_{43} & a_{44} & \cdots & b_4 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\end{bmatrix}
\]

\[
x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}
\]
Triangular Systems

• Solve by forward substitution

\[
\begin{bmatrix}
  a_{11} & 0 & 0 & 0 & \cdots \\
  a_{21} & a_{22} & 0 & 0 & \cdots \\
  a_{31} & a_{32} & a_{33} & 0 & \cdots \\
  a_{41} & a_{42} & a_{43} & a_{44} & \cdots \\
  \vdots & \vdots & \vdots & \vdots & \ddots \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \vdots \\
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3 \\
  b_4 \\
  \vdots \\
\end{bmatrix}
\]

\[
x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}
\]
Triangular Systems

- If $A$ is upper triangular, solve by backsubstitution

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
    0 & a_{22} & a_{23} & a_{24} & a_{25} \\
    0 & 0 & a_{33} & a_{34} & a_{35} \\
    0 & 0 & 0 & a_{44} & a_{45} \\
    0 & 0 & 0 & 0 & a_{55}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    x_4 \\
    x_5
\end{bmatrix}
\begin{bmatrix}
    b_1 \\
    b_2 \\
    b_3 \\
    b_4 \\
    b_5
\end{bmatrix}
\]

\[x_5 = \frac{b_5}{a_{55}}\]
Triangular Systems

- If $A$ is upper triangular, solve by backsubstitution

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
  0 & a_{22} & a_{23} & a_{24} & a_{25} \\
  0 & 0 & a_{33} & a_{34} & a_{35} \\
  0 & 0 & 0 & a_{44} & a_{45} \\
  0 & 0 & 0 & 0 & a_{55}
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3 \\
  b_4 \\
  b_5
\end{bmatrix}
\]

$x_4 = \frac{b_4 - a_{45}x_5}{a_{44}}$
Triangular Systems

- Both of these special cases can be solved in $O(n^2)$ time
- This motivates a factorization approach to solving arbitrary systems:
  - Find a way of writing $A$ as $LU$, where $L$ and $U$ are both triangular
  - $Ax=b \Rightarrow LUx=b \Rightarrow Ld=b \Rightarrow Ux=d$
  - Time for **factoring matrix** dominates computation
Solving $Ax = b$ with LU Decomposition of $A$
Symmetric Matrices: Cholesky Decomposition

- For symmetric matrices, choose $U = L^T$  
  \[(A = LL^T)\]
- Perform decomposition

  \[
  \begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{12} & a_{22} & a_{23} \\
    a_{13} & a_{23} & a_{33}
  \end{bmatrix} \Rightarrow
  \begin{bmatrix}
    l_{11} & 0 & 0 \\
    l_{21} & l_{22} & 0 \\
    l_{31} & l_{32} & l_{33}
  \end{bmatrix}
  \begin{bmatrix}
    l_{11} & l_{21} & l_{31} \\
    l_{12} & l_{22} & l_{32} \\
    l_{13} & l_{23} & l_{33}
  \end{bmatrix}
  \]

- $Ax = b \implies LL^T x = b \implies Ld = b \implies L^T x = d$
Cholesky Decomposition

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{12} & a_{22} & a_{23} \\
a_{13} & a_{23} & a_{33}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
l_{11} & 0 & 0 \\
l_{21} & l_{22} & 0 \\
l_{31} & l_{32} & l_{33}
\end{bmatrix}
\begin{bmatrix}
l_{11} & l_{21} & l_{31} \\
l_{11} & l_{22} & l_{32} \\
0 & 0 & l_{33}
\end{bmatrix}
\]

\[l_{11}^2 = a_{11} \quad \Rightarrow \quad l_{11} = \sqrt{a_{11}}\]

\[l_{11} l_{21} = a_{12} \quad \Rightarrow \quad l_{21} = \frac{a_{12}}{l_{11}}\]

\[l_{11} l_{31} = a_{13} \quad \Rightarrow \quad l_{31} = \frac{a_{13}}{l_{11}}\]

\[l_{21}^2 + l_{22}^2 = a_{22} \quad \Rightarrow \quad l_{22} = \sqrt{a_{22} - l_{21}^2}\]

\[l_{21} l_{31} + l_{22} l_{32} = a_{23} \quad \Rightarrow \quad l_{32} = \frac{a_{23} - l_{21} l_{31}}{l_{22}}\]
Cholesky Decomposition

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{12} & a_{22} & a_{23} \\
a_{13} & a_{23} & a_{33}
\end{bmatrix} \Rightarrow \begin{bmatrix}
l_{11} & 0 & 0 \\
l_{21} & l_{22} & 0 \\
l_{31} & l_{32} & l_{33}
\end{bmatrix} \begin{bmatrix}
l_{11} & l_{21} & l_{31} \\
l_{21} & l_{22} & l_{32} \\
l_{31} & l_{32} & l_{33}
\end{bmatrix}
\]

\[l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}\]

\[l_{ji} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{ii}}\]
Cholesky Decomposition

- This fails if it requires taking square root of a negative number
- Need another condition on A: positive definite

i.e., For any \( v \), \( v^T A v > 0 \)

(Equivalently, all positive eigenvalues)
Cholesky Decomposition

• Running time turns out to be $\frac{1}{6}n^3$ multiplications + $\frac{1}{6}n^3$ additions
  – Still cubic, but lower constant
  – Half as much computation & storage as LU

• Result: this is preferred method for solving symmetric positive definite systems
LU Decomposition

For more general matrices, factor $A$ into $LU$, where $L$ is lower triangular and $U$ is upper triangular

\[
\begin{align*}
Ax &= b \\
LUx &= b \\
Ly &= b \\
Ux &= y
\end{align*}
\]

Last 2 steps in $O(n^2)$ time, so total time dominated by decomposition
\[ A = LU \]

- More unknowns than equations!
- Let all \( l_{ii} = 1 \) (Doolittle’s method)
  - Or, could have chosen to let all \( u_{ii} = 1 \) (Crout’s method)
Doolittle Factorization for LU Decomposition

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}
\begin{bmatrix}
u_{11} \\
u_{12} \\
u_{13}
\end{bmatrix}
\]

\[
u_{11} = a_{11}
\]

\[
l_{21}u_{11} = a_{21} \quad \Rightarrow \quad l_{21} = \frac{a_{21}}{u_{11}}
\]

\[
l_{31}u_{11} = a_{31} \quad \Rightarrow \quad l_{31} = \frac{a_{31}}{u_{11}}
\]

\[
u_{12} = a_{12}
\]

\[
l_{21}u_{12} + u_{22} = a_{22} \quad \Rightarrow \quad u_{22} = a_{22} - l_{21}u_{12}
\]

\[
l_{31}u_{12} + l_{32}u_{22} = a_{32} \quad \Rightarrow \quad l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}}
\]
Doolittle Factorization

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}
\begin{bmatrix}
u_{11} & u_{12} & u_{13} \\
u_{22} & u_{23} & \\
u_{33} & \end{bmatrix}
\]

- For \( i = 1..n \)
  - For \( j = 1..i \)
  - For \( j = i+1..n \)

\[
u_{ji} = a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}\]

\[
l_{ji} = \frac{a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}}{u_{ii}}\]
Doolittle Factorization

• Interesting note: # of outputs = # of inputs, algorithm only refers to elements of A, not b

• Can do this in-place!
  – Algorithm replaces A with matrix of l and u values, 1s are implied
  – Resulting matrix must be interpreted in a special way: not a regular matrix
  – Can rewrite forward/backsubstitution routines to use this “packed” l-u matrix

\[
\begin{bmatrix}
 l_{21} & l_{31} \\
 u_{12} & u_{22} \\
 u_{13} & u_{23} & u_{33}
\end{bmatrix}
\]
LU Decomposition

• Running time is about $\frac{1}{3}n^3$ multiplies, same number of adds
  – Independent of RHS, each of which requires $O(n^2)$ back/forward substitution
  – This is the preferred general method for solving linear equations

• Pivoting very important
  – Partial pivoting is sufficient, and widely implemented
  – LU with pivoting can succeed even if matrix is singular (!)
    (but back/forward substitution fails…)
Matrix Inversion using LU

• LU depend only on A, not on b
• Re-use L & U for multiple values of b
  – i.e., repeat back-substitution
• How to compute $A^{-1}$?

$$AA^{-1} = \mathbf{I} \text{ (n×n identity matrix), e.g.}$$

→ Use LU decomposition with

$$b_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad b_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$