# Solving Linear Systems

Sanzheng Qiao

Department of Computing and Software
McMaster University

January 2014

# Outline

# Introduction

Problem setting: Solve for $x$ in the system of linear equations:

$$Ax = b$$

$A$: $n$-by-$n$, nonsingular;
$b$: $n$-by-1.

Since $A$ is nonsingular, this system has a unique solution for any right-hand-side vector $b$.

# Introduction (cont.)

In this part, we focus on the methods for solving $Ax = b$ where $A$ is a dense matrix, so that matrix $A$ is stored in a two dimensional array.

When $A$ is very large and sparse, it is often stored in a special data structure to avoid storing many zero entries. For example, a tridiagonal matrix is stored in three vectors (diagonals). There are methods for solving very large and sparse linear systems. They will be briefly discussed later.

# Text book method 1: Cramer's rule

Cramer's rule is a standard text book method for solving linear systems. The $i$th entry $x_i$ of the solution $x$ is given by

$$x_i = \det(A_i) / \det(A),$$

where $A_i$ is the matrix formed by replacing the $i$th column of $A$ by $b$.

Cramer's rule is of theoretical importance. It gives the solution in explicit form.

# Text book method 1: Cramer's rule (cont.)

The Cramer's rule is impractical. It may be useful for very small systems, such as $n = 2$ or 3.

Prohibitively inefficient. We need to compute $n + 1$ determinants of order $n$, each of which is a sum of $n!$ products and each product requires $n - 1$ multiplications. Total of $(n + 1) \cdot n! \cdot (n - 1)$ floating-point multiplications or additions.

**Question**

How long does it take to solve a system of order 30 on a computer that can perform two billion floating-point addition or multiplication operations (flops) per second?

Answer: $10^{32}/(2 \times 10^9) \approx 5 \times 10^{22}$ seconds!

# Method 2: Compute $A^{-1}$, then $x = A^{-1}b$

Usually it is unnecessary and inefficient to compute $A^{-1}$, and the computed solution is inaccurate.

Example. Solve for $x$ in $7x = 21$ ($n = 1$)

In a floating-point system with $\beta = 10$ and $t = 4$.
$1/7 = 1.429 \times 10^{-1}$, then $21 \times 0.1429 = 3.001$ (one division and one multiplication).

Whereas $x = 21/7 = 3.000$ (one division).

# Example

Example. Assuming the floating-point system with $\beta = 10$ and $t = 4$, and the linear system:

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 6 \end{bmatrix}$$

Exact solution:

$$x_1 = 0, \quad x_2 = -1, \quad x_3 = 1$$

# Forward elimination

Stage 1. Forward elimination

Step 1. Eliminate $x_1$ in equations (2) and (3).

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 6 \end{bmatrix}$$

$$-(-3/10) \times (1) + (2) \to (2)$$
$$-(5/10) \times (1) + (3) \to (3)$$

pivot: $a_{11} = 10$

multipliers: $m_{21} = -(-3/10)$, $m_{31} = -(5/10)$.

# Forward elimination (cont.)

The updated system:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.1 \\ 2.5 \end{bmatrix}$$

# Matrix form

Use the multipliers to form an elementary matrix:

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0.3 & 1 & 0 \\ -0.5 & 0 & 1 \end{bmatrix},$$

then

$$M_1 A = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \qquad M_1 b = \begin{bmatrix} 7 \\ 6.1 \\ 2.5 \end{bmatrix}.$$

# Forward elimination (cont.)

Step 2. Eliminate $x_2$ in equation (3).

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.1 \\ 2.5 \end{bmatrix}$$

$$-(2.5/-0.1) \times (2) + (3) \rightarrow (3)$$

pivot: $a_{22} = -0.1$, multiplier: $m_{32} = 2.5/-0.1$.

# Forward elimination (cont.)

The updated system:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.1 \\ 155 \end{bmatrix}$$

The original general linear system is reduced to an upper triangular linear system.

# Matrix form

Use the multipler to form an elementary matrix

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2.5/0.1 & 1 \end{bmatrix}$$

then

$$M_2M_1A = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix} \quad M_2M_1b = \begin{bmatrix} 7 \\ 6.1 \\ 155 \end{bmatrix}$$

# Backward substitution

Stage 2. Backward substitution.
The upper triangular system:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.1 \\ 155 \end{bmatrix}.$$

Solve for the solution vector backwards:

$$\begin{aligned} x_3 &= 155/155 = 1.000 \\ x_2 &= (6.1 - 6x_3)/(-0.1) = -1.000 \\ x_1 &= (7 - (-7)x_2 - 0x_3)/10 = 0 \end{aligned}$$

# Properties of elementary matrix

It is simple to invert an elementary matrix:

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{bmatrix} \quad M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m_{32} & 1 \end{bmatrix}$$

It is simple to multiply elementary matrices:

$$M_1^{-1} M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & -m_{32} & 1 \end{bmatrix}$$

Notice the order.

# Putting things together

Triangularization of $A$:

$$M_2M_1A = \begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.1 & 6 \\ 0 & 0 & 155 \end{bmatrix} = U$$

$$M_2M_1b = \begin{bmatrix} 7 \\ 6.1 \\ 155 \end{bmatrix}$$

A decomposition:

$$A = M_1^{-1}M_2^{-1}U$$

## Putting things together (cont.)

The product

$$M_1^{-1}M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & -m_{32} & 1 \end{bmatrix} = L,$$

is a lower triangular matrix.

In general, $A = LU$.

The LU decomposition. (L: lower triangular; U: upper triangular)

# Algorithm. Gaussian elimination without pivoting

Given an $n$-by-$n$ matrix $A$, this algorithm computes lower triangular $L$ and upper triangular $U$ so that $A = LU$. On return, $A$ is overwritten by $L$ and $U$.

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                    - A(k+1:n,k)*A(k,k+1:n);
end
```

$L = \text{eye}(n, n) + \text{tril}(A, -1)$
$U = \text{triu}(A)$

# Example

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                    - A(k+1:n,k)*A(k,k+1:n);
end
```

Original $A$

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{bmatrix}$$

# Example

```
for k=1 to n-1
    A(k+1:n,k) = A(k+1:n,k)/A(k,k);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                     - A(k+1:n,k)*A(k,k+1:n);
end
```

$k = 1$, calculate multipliers

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & 2 & 6 \\ 0.5 & -1 & 5 \end{bmatrix}$$

# Example

```
for k=1 to n-1
    A(k+1:n,k) = A(k+1:n,k)/A(k,k);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                   - A(k+1:n,k)*A(k,k+1:n);
end
```

$k = 1$, update submatrix

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & -0.1 & 6 \\ 0.5 & 2.5 & 5 \end{bmatrix}$$

# Example

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                        - A(k+1:n,k)*A(k,k+1:n);
end
```

$k = 2$, calculate multiplier

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & -0.1 & 6 \\ 0.5 & -25 & 5 \end{bmatrix}$$

# Example

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                 - A(k+1:n,k)*A(k,k+1:n);
end
```

$k = 2$, update submatrix

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & -0.1 & 6 \\ 0.5 & -25 & 155 \end{bmatrix}$$

# Example

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                    - A(k+1:n,k)*A(k,k+1:n);
end
```

Final $A$

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & -0.1 & 6 \\ 0.5 & -25 & 155 \end{bmatrix}$$

$L = \text{eye}(n, n) + \text{tril}(A, -1)$
$U = \text{triu}(A)$

# Solving triangular systems

Solve $Ly = b$:

```
b(1) = b(1)/L(1,1);
for k=2 to n
   tmp = b(k) - L(k,1:k-1)*b(1:k-1);
   b(k) = tmp/L(k,k);
end
```

Initial

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 4 \\ 6 \end{bmatrix}$$

# Solving triangular systems

Solve $Ly = b$:

```
b(1) = b(1)/L(1,1);
for k=2 to n
   tmp = b(k) - L(k,1:k-1)*b(1:k-1);
   b(k) = tmp/L(k,k);
end
```

$k = 2$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 6.1 \\ 6 \end{bmatrix}$$

# Solving triangular systems

Solve $Ly = b$:

```
b(1) = b(1)/L(1,1);
for k=2 to n
   tmp = b(k) - L(k,1:k-1)*b(1:k-1);
   b(k) = tmp/L(k,k);
end
```

$k = 3$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & -25 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 6.1 \\ 155 \end{bmatrix}$$

Solve $Ux = b$: Similar (backward).

# Operation counts

LU decomposition:

```
for k=1 to n-1
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                  - A(k+1:n,k)*A(k,k+1:n);
end
```

$$\int_1^{n-1} (n-k) + 2(n-k)^2 dk \approx \frac{2}{3}n^3.$$

# Operation counts

Solving triangular systems:

```
b(1) = b(1)/L(1,1);
for k=2 to n
   tmp = b(k) - L(k,1:k-1)*b(1:k-1);
   b(k) = tmp/L(k,k);
end
```

$$2 \int_2^n 2(k-1)dk \approx 2n^2.$$

# Operation counts

**Question**

How long does it take to solve a system of order 30 on a computer that can perform two billion floating-point addition or multiplication operations (flops) per second?

Answer: $(0.7 \times 30^3 + 2 \times 30^2)/(2 \times 10^9) \approx 10^{-5}$ seconds!

# What is pivoting?

Change the (2,2)-entry of $A$ slightly from 2 to 2.099 and $b_2$ in $b$ accordingly so that the exact solution is unchanged.

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 3.901 \\ 6 \end{bmatrix}$$

Exact solution: $(0, -1, 1)^{\mathrm{T}}$.

# What is pivoting? (cont.)

Forward elimination ($\beta = 10$, $p = 4$)

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 2.5 \end{bmatrix}$$

pivot: 10, multipliers: 0.3, $-0.5$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 0 & 1.501 \times 10^4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 1.500 \times 10^4 \end{bmatrix}$$

pivot: $-0.001$; multiplier: 2500.

# What is pivoting? (cont.)

Backward substitution

$x_3 = 1.500 \times 10^4 / 1.501 \times 10^4 = 9.993 \times 10^{-1}$

$x_2 = (6.001 - 6x_3)/(-0.001) = 5.0$

What went wrong?

Step 1: multipliers $m_{21} = 0.3$, $m_{31} = -0.5$

$$\begin{bmatrix} 1.000 \times 10^1 & -7.000 & 0 \\ 0 & -1.000 \times 10^{-3} & 6.000 \\ 0 & +2.500 & 5.000 \end{bmatrix} \qquad \begin{bmatrix} 7.000 \\ 6.001 \\ 2.500 \end{bmatrix}$$

Exact, no rounding errors.

# What is pivoting? (cont.)

Step 2: multiplier $m_{31} = 2.500\mathrm{E}+3$, exact.

$$\begin{bmatrix} 1.000 \times 10^1 & -7.000 & 0 \\ 0 & -1.000 \times 10^{-3} & 6.000 \\ 0 & 0 & 1.501 \times 10^4 \end{bmatrix} \begin{bmatrix} 7.000 \\ 6.001 \\ 1.500 \times 10^4 \end{bmatrix}$$

The rounding error in $1.501 \times 10^4$ ($A(3,3)$, the exact result is $15,005$) or $1.500 \times 10^4$ ($b_3$, the exact result is $15,005$) equals half of its ulp ($(0.001 \times 10^4)/2 = 5$), which has the same size as the size of the solution.

# What is pivoting? (cont.)

Back solve:

| | computed | exact |
|---|---|---|
| $x_3$ | 1.001 | 1.0 |
| $x_2$ | $\frac{6.001-6\times 1.001}{-0.001} = -5.000$ | $\frac{6.001-6\times 1.0}{-0.001} = -1.0$ |

Catastrophic cancellation.

# What is pivoting? (cont.)

Error in the result can be as large as half of the ulp of the largest intermediate results.

Solution:

Avoid large intermediate results (entries in the lower-right submatrices).

Avoid small pivots (causing large multipliers).

How?

Interchange equations (rows).

# Pivoting

Forward elimination step 2:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & -0.001 & 6 \\ 0 & 2.5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 6.001 \\ 2.5 \end{bmatrix}$$

Matrix form:

$$A \leftarrow M_1 A, \quad M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0.3 & 1 & 0 \\ -0.5 & 0 & 1 \end{bmatrix}$$

# Pivoting (cont.)

Interchange equations (rows) (2) and (3) to avoid small pivot,

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & -0.001 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2.5 \\ 6.001 \end{bmatrix}$$

Matrix form:

$$A \leftarrow P_2 M_1 A, \quad P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

pivot: 2.5, multiplier: $4 \times 10^{-4}$

# Pivoting (cont.)

The updated system:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2.5 \\ 6.002 \end{bmatrix}$$

Matrix form:

$$A \leftarrow M_2 P_2 M_1 A, \quad M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 \times 10^{-4} & 1 \end{bmatrix}$$

# Pivoting (cont.)

The updated system:

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 2.5 \\ 6.002 \end{bmatrix}$$

Backward substitution:

$$\begin{aligned} x_3 &= 6.002/6.002 = 1.000 \\ x_2 &= (2.5 - 5x_3)/2.5 = -1.000 \\ x_1 &= (7 - (-7)x_2 - 0x_3)/10 = 0 \end{aligned}$$

# LU decomposition

$$M_2 P_2 M_1 A = U, \quad (M_1^{-1} P_2^{-1} M_2^{-1}) U = A$$

$$M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & 1 & 0 \\ 0.5 & 0 & 1 \end{bmatrix} \quad P_2 = P_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 \times 10^{-4} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix}$$

# LU decomposition (cont.)

But

$$M_1^{-1}P_2M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -0.3 & -4\times 10^{-4} & 1 \\ 0.5 & 1 & 0 \end{bmatrix}$$

is not lower triangular.

However,

$$(P_2M_1^{-1}P_2)M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & -4\times 10^{-4} & 1 \end{bmatrix}$$

is lower triangular and elementary! Call it $L$.

$(P_2M_1^{-1}P_2M_2^{-1})U = P_2A$ is an LU decomposition of $P_2A$, (row) permuted $A$.

# LU decomposition (cont.)

Consider

$$\hat{M}_1^{-1} := P_2 M_1^{-1} P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & 0 & 1 \end{bmatrix}$$

equivalent to interchanging $m_{21}$ and $m_{31}$.

In general, suppose that $M_2 P_2 M_1 P_1 A = U$, that is, $A = P_1 M_1^{-1} P_2 M_2^{-1} U$, then

$$P_2 P_1 A = ((P_2 M_1^{-1} P_2) M_2^{-1}) U$$

An LU decomposition of permuted $A$.

# LU decomposition (cont.)

In $n$-dimensional case, $M_{n-1}P_{n-1}\cdots M_2P_2M_1P_1A = U$. Then

$$
\begin{aligned}
A &= P_1M_1^{-1}P_2M_2^{-1}\cdots P_{n-2}M_{n-2}^{-1}P_{n-1}M_{n-1}^{-1}U \\
&= P_1M_1^{-1}P_2M_2^{-1}\cdots P_{n-3}M_{n-3}^{-1}P_{n-2}P_{n-1} \\
&\quad (P_{n-1}M_{n-2}^{-1}P_{n-1})M_{n-1}^{-1}U \\
&= P_1 ... P_{n-1}(P_{n-1} ... P_2M_1^{-1}P_2 ... P_{n-1})\cdots \\
&\quad (P_{n-1}M_{n-2}^{-1}P_{n-1})M_{n-1}^{-1}U
\end{aligned}
$$

LU decomposition $P_{n-1} ... P_1A = LU$ of permuted $A$.

In programming, $A$ is overwritten by $L$ and $U$ and $L$ is stored in the lower part of $A$. When we interchange rows of $A$, we also interchange corresponding (entire) rows of $L$.

# Pivoting (cont.)

Algorithm. Gaussian elimination with partial pivoting.

```
for k=1 to n-1
   find the pivot:  max(|A(k:n,k)|);
   record the pivoting row index in p(k);
   interchange rows A(k,:) and A(p(k),:);
   A(k+1:n,k) = A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n) = A(k+1:n,k+1:n)
                  - A(k+1:n,k)*A(k,k+1:n);
end
```

# Example

Original

$$\begin{bmatrix} -3 & 2.099 & 6 \\ 10 & -7 & 0 \\ 5 & -1 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

# Example

$k = 1$, pivot

$$\begin{bmatrix} -3 & 2.099 & 6 \\ \color{red}{10} & -7 & 0 \\ 5 & -1 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} \color{red}{2} \\ 2 \\ 1 \end{bmatrix}$$

# Example

$k = 1$, permute $P_1$

$$\begin{bmatrix} \color{red}{10} & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

# Example

$k = 1$, Multipliers $M_1$

$$\begin{bmatrix} \color{red}{10} & -7 & 0 \\ \color{teal}{-0.3} & 2.099 & 6 \\ \color{teal}{0.5} & -1 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

# Example

$k = 1$, update

$$\begin{bmatrix} 10 & -7 & 0 \\ -0.3 & -0.001 & 6 \\ 0.5 & 2.5 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}$$

# Example

$k = 2$, pivot

$$\begin{bmatrix} 10 & -7 & 0 \\ \color{teal}{-0.3} & -0.001 & 6 \\ \color{teal}{0.5} & \color{red}{2.5} & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ \color{red}{3} \\ -1 \end{bmatrix}$$

# Example

$k = 2$, permute $P_2$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0.5 & 2.5 & 5 \\ -0.3 & -0.001 & 6 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

# Example

$k = 2$, multiplier $M_2$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0.5 & 2.5 & 5 \\ -0.3 & -0.0004 & 6 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

# Example

$k = 2$, update

$$\begin{bmatrix} 10 & -7 & 0 \\ 0.5 & 2.5 & 5 \\ -0.3 & -0.0004 & 6.002 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

# Example

$$
\begin{aligned}
LU &= \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & -0.0004 & 1 \end{bmatrix} \begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \\
&= \begin{bmatrix} 10 & -7 & 0 \\ 5 & -1 & 5 \\ -3 & 2.099 & 6 \end{bmatrix},
\end{aligned}
$$

permuted $A$

$$
P_2 P_1 \begin{bmatrix} -3 & 2.099 & 6 \\ 10 & -7 & 0 \\ 5 & -1 & 5 \end{bmatrix} \qquad p = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}
$$

$\det(A) = p(n)U(1,1)...U(n,n) = 150.05$

# Basic Linear Algebra Subroutines (BLAS)

Operations in GE with pivoting:

- imax: find the index of $\max(|A(k:n,k)|)$
- swap: interchange $A(k,:)$ and $A(p(k),:)$
- scal: scalar-vector multiplication: $A(k,k)^{-1}A(k+1:n,k)$
- axpy: $\alpha x + y$, vector update: $A(k+1:n,j) - A(k,j)A(k+1:n,k)$, for $j = k+1:n$

# Problem setting

Given the decomposition $LU = PA$, solve $Ax = b$

or solve $PAx = Pb$

Solve two triangular systems:

stage 1, solve $Ly = Pb$ (forward solve)
stage 2, solve $Ux = y$ (back solve)

$$LUx = Pb$$

Consider $Ux = y$, given $U$ and $y$, the solution $x$ overwrites $y$.

# Two versions

Row version (C):

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
    $y(i) = y(i) - U(i, i+1:n)y(i+1:n);$
    $y(i) = y(i)/U(i,i);$
end for $i$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
  $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1);$
  $y(i) = y(i)/U(i,i);$
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 7 \\ 2.5 \\ 6.002 \end{bmatrix}$$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n)$;
for $i = n-1 : -1 : 1$;
  $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1)$;
  $y(i) = y(i)/U(i,i)$;
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 7 \\ 2.5 \\ 1.0 \end{bmatrix}$$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
  $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1);$
  $y(i) = y(i)/U(i,i);$
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 7 \\ -2.5 \\ 1.0 \end{bmatrix}$$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
    $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1);$
    $y(i) = y(i)/U(i,i);$
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 7 \\ -1.0 \\ 1.0 \end{bmatrix}$$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
    $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1);$
    $y(i) = y(i)/U(i,i);$
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

# Column version (FORTRAN)

$y(n) = y(n)/U(n,n);$
for $i = n-1 : -1 : 1;$
  $y(1:i) = y(1:i) - y(i+1)U(1:i, i+1);$
  $y(i) = y(i)/U(i,i);$
end for $i$

$$\begin{bmatrix} 10 & -7 & 0 \\ 0 & 2.5 & 5 \\ 0 & 0 & 6.002 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

# Introduction

Estimating the error in the computed solution.
Computed solution: $\hat{x}$
Exact solution: $x$
Relative forward error: $\|x - \hat{x}\|/\|\hat{x}\|$
But we usually don't know $x$.
Check the residual $r = b - A\hat{x}$?
A contrived example.
$\beta = 10$, $p = 3$

$$\begin{bmatrix} 1.15 & 1.00 \\ 1.41 & 1.22 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.15 \\ 2.63 \end{bmatrix}$$

# Introduction (cont.)

Gaussian elimination with partial pivoting
Interchange two rows; multiplier: $1.15/1.41 = 0.816$;

$$\begin{bmatrix} 1.41 & 1.22 \\ 0 & 0.004 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.63 \\ 0.00 \end{bmatrix}$$

$\hat{x}_2 = 0$, $\hat{x}_1 = 2.63/1.41 = 1.87$.
Residual: $r_1 = 0.01$, $r_2 = 0$.
Exact solution: $x_1 = x_2 = 1$.

Small residual does not imply small error in solution.

# Introduction (cont.)

Check the pivots?

If the pivots are small, $A$ is nearly singular; however, $A$ might be nearly singular but none of the pivots are small.

How do we estimate the error in the computed solution?
We know that the relative forward error:

$$\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \leq \text{cond} \cdot \text{backward error}.$$

# Estimating backward error

Suppose that the computed solution $\hat{x}$ is the exact solution of the perturbed system:

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

then the relative backward error is

$$\frac{\|\Delta A\|}{\|A\|}.$$

How do we get $\|\Delta A\|$?

# Estimating backward error

Since

$$\frac{\|b - A\hat{x}\|}{\|A\|\,\|\hat{x}\|} = \frac{\|\Delta A\hat{x}\|}{\|A\|\,\|\hat{x}\|} \leq \frac{\|\Delta A\|}{\|A\|},$$

in practice, the relative residual

$$\frac{\|b - A\hat{x}\|}{\|A\|\,\|\hat{x}\|}$$

can be used as an estimate for the backward error.

In other words, the relative residual is an indication of the stability of the algorithm.

# Condition of a matrix

The sensitivity of the solution $x$ to the perturbations on $A$ and $b$.
Measure of nearness of singularity.

Vector norms:

$\|x\| > 0$, if $x \neq 0$
$\|0\| = 0$
$\|cx\| = |c|\,\|x\|$, for all scalars $c$
$\|x + y\| \leq \|x\| + \|y\|$

Examples.

$\|x\|_2 = \left(\sum_i |x_i|^2\right)^{1/2}$
$\|x\|_1 = \sum_i |x_1|$
$\|x\|_\infty = \max_i(|x_i|)$

# Condition of a matrix (cont.)

Think of a matrix as a linear transformation between two vector spaces.
The range of possible change:

$$M = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \|A\|$$
$$m = \min_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

When $A$ is singular, $m = 0$.
Examples of matrix norms.

$\|A\|_1 = \max_j \left( \sum_i |a_{ij}| \right)$
$\|A\|_\infty = \max_i \left( \sum_j |a_{ij}| \right)$

# Condition of a matrix

Measurement: $\mathrm{cond}(A) = M/m$
If $A$ is singular, $m = 0$, $\mathrm{cond}(A) = \infty$.
If $A$ is nonsingular,

$$m^{-1} = \max_{x \neq 0} \frac{\|x\|}{\|Ax\|} = \max_{y \neq 0} \frac{\|A^{-1}y\|}{\|y\|} = \|A^{-1}\|$$

**Condition for inverting $A$**

$\mathrm{cond}(A) = \|A\| \, \|A^{-1}\|$

## Condition of a matrix (cont.)

Perturbing $b$:

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

Since $Ax = b$ and $A(\Delta x) = \Delta b$,

$$\|b\| \leq M\|x\| \quad \text{and} \quad \|\Delta b\| \geq m\|\Delta x\|$$

Thus

$$\frac{\|\Delta x\|}{\|x\|} \leq \mathrm{cond}(A)\frac{\|\Delta b\|}{\|b\|}$$

A relative error magnification factor.

How do we get $\|A^{-1}\|$?

# Estimating $\|A^{-1}\|_1$

Basic idea:

Determine a vector $e$ (all components 1 or $-1$) so that the solution for $A^{\mathrm{T}}Az = e$ is heuristically large.

Given $PA = LU$ ($A^{\mathrm{T}} = U^{\mathrm{T}}L^{\mathrm{T}}P$ and $A^{\mathrm{T}}A = U^{\mathrm{T}}L^{\mathrm{T}}LU$), determine $e$ so that the solution $w$ for $U^{\mathrm{T}}w = e$ is heuristically large;

solve for $y$ in $L^{\mathrm{T}}y = w$;
solve for $z$ in $Az = y$;
normalize $\|z\|_1/\|y\|_1 \approx \|A^{-1}\|_1$.

Cost:
If the LU decomposition $PA = LU$ ($O(n^3)$) is available,
it requires solving four triangular systems ($O(n^2)$).

# Example revisited

$$A = \begin{bmatrix} 1.15 & 1.00 \\ 1.41 & 1.22 \end{bmatrix}, \quad b = \begin{bmatrix} 2.15 \\ 2.63 \end{bmatrix}$$

Exact solution: $x = [1\ 1]^{\mathrm{T}}$

Computed solution: $\hat{x} = [1.87\ 0.00]^{\mathrm{T}}$, $\beta = 10$, $t = 3$.

# Example revisited

The computed solution is the exact solution of

$$\hat{A} = \begin{bmatrix} 215/187 & 1.00 \\ 263/187 & 1.22 \end{bmatrix}, \quad \text{and} \quad b$$

The perturbation

$$\Delta A = A - \hat{A} \approx \begin{bmatrix} 2.67 \times 10^{-4} & 0 \\ 3.58 \times 10^{-3} & 0 \end{bmatrix}$$

Relative change in $A$:

$$\frac{\|\Delta A\|}{\|A\|} = 1.5 \times 10^{-3} := \rho u$$

# Example revisited

Relative error in the computed solution:

$$\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \approx 0.7088$$

Almost 100% error, that is, zero digit accuracy.

The condition number

$$\mathrm{cond}(A) \approx 8.26 \times 10^2$$

Almost $u^{-1} = \beta^t$.

$$\frac{\|x - \hat{x}\|}{\|\hat{x}\|} \leq \rho \, \mathrm{cond}(A) \, u$$

The relative change in $A$ is magnified by $\mathrm{cond}(A)$.

# Remarks

- The solution computed by GE with partial pivoting can be viewed as the exact solution of a slightly perturbed coefficient matrix. The relative perturbation is usually $\rho\, u$, where $\rho$ is a constant of the same size as $\beta$. In other words, in practice, GE with partial pivoting is stable.
- The relative error in the computed solution (by GE with partial pivoting) is roughly of the size $\mathrm{cond}(A)u$.
- In practice, the entries in the coefficient matrix $A$ and the right-hand-side vector $b$ contain measurement errors. In the computed solution, The measurement error is roughly magnified by $\mathrm{cond}(A)$. For example, if the measurement accuracy is four decimal digits and the condition number is about $10^2$, then we expect the computed solution has two decimal digit accuracy.

## Symmetric positive definite systems

Symmetric: $A^{\mathrm{T}} = A$, or $a_{ij} = a_{ji}$
Positive definite: $x^{\mathrm{T}}Ax > 0$, for any $x \neq 0$
Properties:

- nonsingular
- all principal submatrices are symmetric and positive definite

The method of choice: Cholesky factorization

$$A = LL^{\mathrm{T}}$$

$L$: lower triangular

## Cholesky factorization

Idea: An equation

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$$a_{11} = l_{11} l_{11}$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$a_{21} = l_{21} l_{11}$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$$a_{22} = l_{21}l_{21} + l_{22}l_{22}$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$$a_{31} = l_{31} l_{11}$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$$a_{32} = l_{31}l_{21} + l_{32}l_{22}$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

$$a_{33} = l_{31}l_{31} + l_{32}l_{32} + l_{33}l_{33}$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

# Cholesky factorization

In general,

$$
\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix}
= \begin{bmatrix} l_{11} & & & 0 \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}
\begin{bmatrix} l_{11} & l_{21} & \cdots & l_{1n} \\ & l_{22} & \cdots & l_{2n} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix}.
$$

# Cholesky factorization (cont.)

Consider the first row of $L$: $l_{11} = \sqrt{a_{11}}$.
Now we consider the general case: the $i$th row.
Suppose we have computed the rows 1 through $i-1$ of $L$, then

$$l_{ij} = (a_{ji} - \sum_{k=1}^{j-1} l_{jk} l_{ik})/l_{jj} \quad j = 1 : i-1$$

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

# Algorithm. Cholesky factorization (scalar)

This algorithm computes a lower triangular $L$ from an $n$-by-$n$ symmetric and positive definite $A$ so that $A = LL^{\mathrm{T}}$.

```
for i=1 to n
   for j=1 to i     % compute L(i,1:i)
      s = A(j,i);
      for k=1 to j-1
         s = s - L(j,k)*L(i,k);
      end
      if j<i
         L(i,j) = s/L(j,j);
      else % j=i
         L(i,i) = sqrt(s);
      end
   end
end
```

# Example

$$A = \begin{bmatrix} 25 & 10 & 10 \\ 10 & 53 & 32 \\ 10 & 32 & 36 \end{bmatrix}$$

Initial

$$L = \begin{bmatrix} 25 & & \\ 10 & 53 & \\ 10 & 32 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 1, j = 1$

$$L = \begin{bmatrix} 5 & & \\ 10 & 53 & \\ 10 & 32 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 2, j = 1$

$$L = \begin{bmatrix} 5 & & \\ 2 & 53 & \\ 10 & 32 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 2, j = 2$

$$L = \begin{bmatrix} 5 & & \\ 2 & 7 & \\ 10 & 32 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 3, j = 1$

$$L = \begin{bmatrix} 5 & & \\ 2 & 7 & \\ 2 & 32 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 3, j = 2$

$$L = \begin{bmatrix} 5 & & \\ 2 & 7 & \\ 2 & 4 & 36 \end{bmatrix}$$

# Example (cont.)

```
for i=1 to n
    for j=1 to i
        L(i,j) = L(i,j) - L(j,1:j-1)'*L(i,1:j-1);
        if j<i
            L(i,j) = L(i,j)/L(j,j);
        else
            L(i,i) = sqrt(L(i,i));
        end
    end
end
```

$i = 3$, $j = 3$ (final)

$$L = \begin{bmatrix} 5 & & \\ 2 & 7 & \\ 2 & 4 & 4 \end{bmatrix}$$

# Cholesky factorization

- storage $n(n+1)/2$
- flops $n^3/3 + O(n^2)$
- stable
- the cheapest way to check if a symmetric matrix is positive definite

# Software packages

Direct methods for general linear systems

- **NETLIB** LAPACK: sgetrf, sgetrs, sgecon

Direct methods for symmetric and positive definite systems

- **NETLIB** LAPACK: spotrf, spotrs

Direct methods for symmetric and indefinite systems

- **NETLIB** LAPACK: ssytrf, ssytrs

Direct methods for sparse systems

- **NETLIB** SuperLU, SPARSE
- **MATLAB** colmmd, symmmd, symrcm

# Summary

- Gaussian elimination with pivoting (`decomp`, `solve`): Working on one matrix, matrix update. Improve instability by avoiding small pivots (controlling the sizes of intermediate results)
- Error estimates: Condition number of a matrix.
- Special systems: Triangle, symmetric and positive definite, Cholesky factorization.

# References

[1] George E. Forsyth and Michael A. Malcolm and Cleve B. Moler. Computer Methods for Mathematical Computations. Prentice-Hall, Inc., 1977.
Ch 3.

[2] Nicholas J. Higham. Accuracy and Stability of Numerical Algorithms. Second Edition. SIAM. Philadelphia, PA, 2002.
Ch 9, Ch 10, Ch 28 (A Gallery of Test Matrices).

[3] Gene H. Golub and Charles F. Van Loan. Matrix Computations, Third Edition. The Johns Hopkins University Press, Baltimore, MD, 1996.
Ch 3.