Chapter 11 Gaussian Elimination (I)

Speaker: Lung-Sheng Chien

Reference book: David Kincaid, Numerical Analysis

OutLine

- Basic operation of matrix
  - representation
  - three elementary matrices
- Example of Gaussian Elimination (GE)
- Formal description of GE
- MATLAB usage

Matrix notation in MATLAB

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix}$$

```
>> A = [6 -2 2 4 ; 12 -8 6 10; 3 -13 9 3 ; -6 4 1 -18]

A =

     6    -2     2     4
    12    -8     6    10
     3   -13     9     3
    -6     4     1   -18
```

$$A(1,:) = A(1,1:4) = \text{first row of } A = \begin{pmatrix} 6 & -2 & 2 & 4 \end{pmatrix}$$

```
>> A(1,:)

ans =

     6    -2     2     4

>> A(1,1:4)

ans =

     6    -2     2     4
```

```
>> A(:,2)

ans =

    -2
    -8
   -13
     4
```

$$A(:,2) = A(1:4,2) = \text{second column of } A = \begin{pmatrix} -2 \\ -8 \\ -13 \\ 4 \end{pmatrix}$$

Matrix-vector product

Inner-product based —— 横×直

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} 6\,x1 + (-2)\,x2 + 2\,x3 + 4\,x4 \\ 12\,x1 + (-8)\,x2 + 6\,x3 + 10\,x4 \\ 3\,x1 + (-13)\,x2 + 9\,x3 + 3\,x4 \\ (-6)\,x1 + 4\,x2 + 1\,x3 + (-18)\,x4 \end{pmatrix}$$

outer-product based —— 直×横

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = x1 \begin{pmatrix} 6 \\ 12 \\ 3 \\ -6 \end{pmatrix} + x2 \begin{pmatrix} -2 \\ -8 \\ -13 \\ 4 \end{pmatrix} + x3 \begin{pmatrix} 2 \\ 6 \\ 9 \\ 1 \end{pmatrix} + x4 \begin{pmatrix} 4 \\ 10 \\ 3 \\ -18 \end{pmatrix}$$

Matrix-vector product: MATLAB implementation

compute $b = Ax$

Inner-product based

```
for i = 1:4
    for j = 1:4
        b(i) = A(i,j)*x(j)
    end
end
```

outer-product based

```
for j = 1:4
    for i = 1:4
        b(i) = A(i,j)*x(j)
    end
end
```

matvec.m

```
function  b = matvec(A,x)
%
%  matrix-vector product:
%      inner-product based
%

[m,n] = size(A) ;
b = zeros(m,1) ;

for i = 1:m
   for j =1:n
      b(i) = A(i,j)*x(j) ;
   end
end

```

Question: which one is better

Column-major nature in MATLAB

physical index : 1D

6	1
12	2
3	3
-6	4
-2	5
-8	6
-13	7
4	8
2	9
6	10
9	11
1	12
4	13
10	14
3	15
-18	16

Logical index : 2D

6	-2	2	4
12	-8	6	10
3	-13	9	3
-6	4	1	-18

Question: how does column-major affect inner-product based matrix-vector product and outer-product based matrix-vector product?

Matrix representation: outer-product

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \sum_{i,j=1}^{3} a_{ij} e_i e_j^T = \begin{Bmatrix} a_{11} e_1 e_1^T + a_{12} e_1 e_2^T + a_{13} e_1 e_3^T + \\ a_{21} e_2 e_1^T + a_{22} e_2 e_2^T + a_{23} e_2 e_3^T + \\ a_{31} e_3 e_1^T + a_{32} e_3 e_2^T + a_{33} e_3 e_3^T \end{Bmatrix}$$

where $e_3 e_2^T = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ is outer-product representation

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \sum_{i,j=1}^{3} a_{ij} e_i \left(e_j^T x \right) = \sum_{i=1}^{3} e_i \left(\sum_{j=1}^{3} a_{ij} x_j \right) \xrightarrow{inner-product\ based} \begin{pmatrix} A(1,:)\,x \\ A(2,:)\,x \\ A(3,:)\,x \end{pmatrix}$$

Elementary matrix [1]

(1) The interchange of two rows in ***A:*** $A(s,:) \leftrightarrow A(t,:)$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

Define permutation matrix $P = (1,3,2) \equiv \begin{pmatrix} e_1^T \\ e_3^T \\ e_2^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

$$Px = \begin{pmatrix} e_1^T \\ e_3^T \\ e_2^T \end{pmatrix} x \xrightarrow{inner-product\ based} \begin{pmatrix} e_1^T x \\ e_3^T x \\ e_2^T x \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix}$$

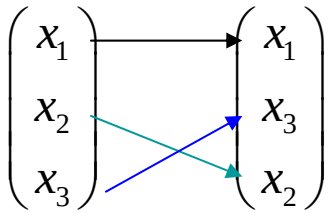
How to explain?

Question 1: why $P^{-1} = P^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

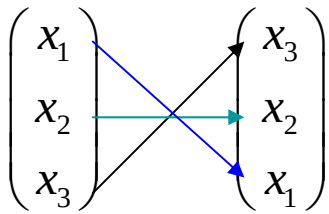
Question 2: how to easily obtain P^{-1}

Concatenation of permutation matrices

$$P_1 = (1,3,2) \equiv \begin{pmatrix} e_1^T \\ e_3^T \\ e_2^T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$ such that $$P_1 x = \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix}$$

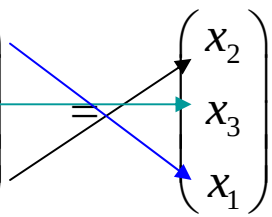


$$P_2 = (3,2,1) \equiv \begin{pmatrix} e_3^T \\ e_2^T \\ e_1^T \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$ such that $$P_2 x = \begin{pmatrix} x_3 \\ x_2 \\ x_1 \end{pmatrix}$$



Question: $P_2 P_1 = (3,2,1)(1,3,2) = ?$

$$P_2 (P_1 x) = P_2 \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_1 \end{pmatrix}$$



implies $P_2 P_1 = (2,3,1)$

Direct calculation $$P_2 P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Elementary matrix [2]

(2) Multiplying one row by a nonzero constant: $\lambda A(s,:) \to A(s,:)$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ \lambda a_{21} & \lambda a_{22} & \lambda a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Define scaling matrix $S_2(\lambda) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{pmatrix}$

How to explain?

$$S_2(\lambda)\, x \xrightarrow{inner-product\ based} \begin{pmatrix} x_1 \\ \lambda x_2 \\ x_3 \end{pmatrix}$$

$$S_2(\lambda)^{-1} = S_2(\lambda^{-1}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

since $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \xrightarrow{S_2(\lambda)} \begin{pmatrix} x_1 \\ \lambda x_2 \\ x_3 \end{pmatrix} \xrightarrow{S_2(1/\lambda)} \begin{pmatrix} x_1 \\ \frac{1}{\lambda}(\lambda x_2) \\ x_3 \end{pmatrix}$

Elementary matrix [3]

(3) Adding to one row a multiple of another: $A(s,:)+\lambda A(t,:)\rightarrow A(s,:)$

$$\begin{pmatrix}1&0&0\\0&1&0\\0&\lambda&1\end{pmatrix}\begin{pmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{pmatrix}=\begin{pmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\\lambda a_{21}+a_{31}&\lambda a_{22}+a_{32}&\lambda a_{23}+a_{33}\end{pmatrix}$$

Define GE (Gaussian Elimination) matrix $L_{32}(\lambda)\equiv\begin{pmatrix}1&0&0\\0&1&0\\0&\lambda&1\end{pmatrix}$

How to explain?

$$L_{32}(\lambda)x=\begin{pmatrix}1&0&0\\0&1&0\\0&\lambda&1\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\xrightarrow{inner-product\ based}\begin{pmatrix}x_1\\x_2\\\lambda x_2+x_3\end{pmatrix}$$

$$L_{32}(\lambda)=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}+\begin{pmatrix}0&0&0\\0&0&0\\0&\lambda&0\end{pmatrix}=I+\lambda e_3e_2^T$$

outer-product representation

$$L_{32}(\lambda)x = \left(I + \lambda e_3 e_2^T\right)x = x + \lambda e_3\left(e_2^T x\right) = x + \left(\lambda x_2\right)e_3 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \lambda x_2 \end{pmatrix}$$

$L_{32}(\lambda)x$: multiply x_2 by λ and add λx_2 to x_3

$L_{32}(\lambda)A$: multiply $A(2,:)$ by λ and add $\lambda A(2,:)$ to $A(3,:)$

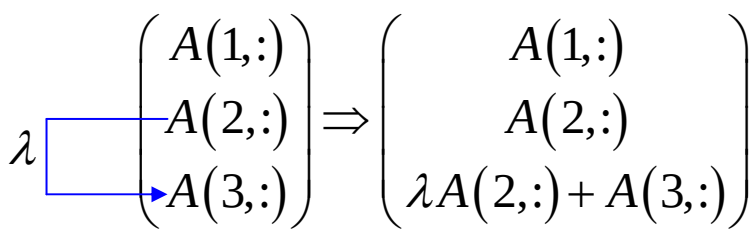
for $i > j$

$L_{ij}(\lambda)x$: multiply x_j by λ and add λx_j to x_i

$L_{ij}(\lambda)A$: multiply $A(j,:)$ by λ and add $\lambda A(j,:)$ to $A(i,:)$

Use MATLAB notation

$$L_{32}(\lambda)A = \begin{pmatrix} A(1,:) \\ A(2,:) \\ \lambda A(2,:) + A(3,:) \end{pmatrix}$$



Concatenation of GE matrices

Suppose

$$L_{21}(\lambda) = \begin{pmatrix} 1 & & \\ \lambda & 1 & \\ & & 1 \end{pmatrix} \quad \text{such that} \quad L_{21}(\lambda)A = \begin{pmatrix} A(1,:) \\ \lambda A(1,:) + A(2,:) \\ A(3,:) \end{pmatrix}$$

$$L_{31}(\mu) = \begin{pmatrix} 1 & & \\ & 1 & \\ \mu & & 1 \end{pmatrix} \quad \text{such that} \quad L_{31}(\mu)A = \begin{pmatrix} A(1,:) \\ A(2,:) \\ \mu A(1,:) + A(3,:) \end{pmatrix}$$

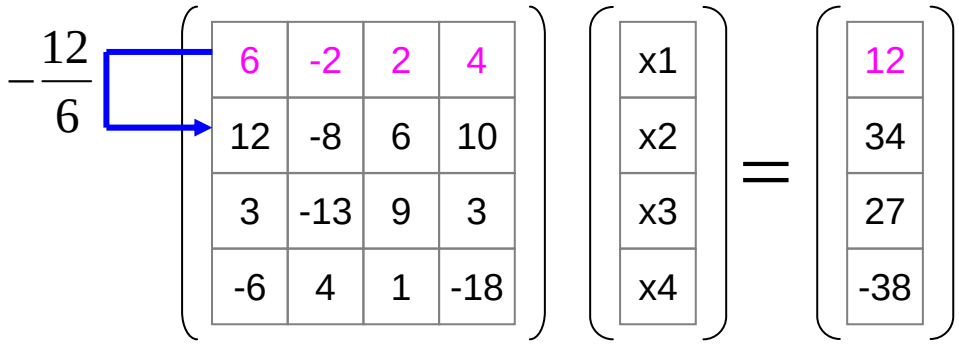
Question: $L_{31}(\mu)L_{21}(\lambda) = ?$

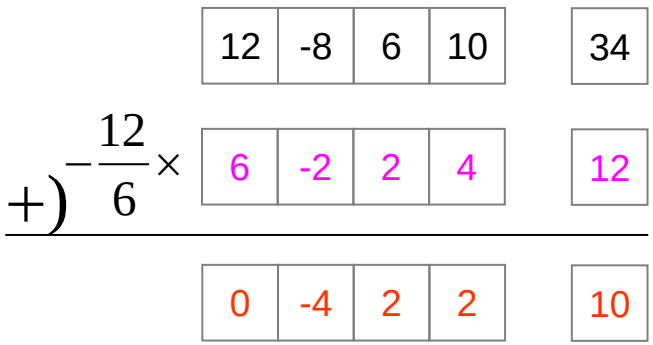
$$L_{31}(\mu)L_{21}(\lambda)A = L_{31}(\mu)\begin{pmatrix} A(1,:) \\ \lambda A(1,:) + A(2,:) \\ A(3,:) \end{pmatrix} = \begin{pmatrix} A(1,:) \\ \lambda A(1,:) + A(2,:) \\ \mu A(1,:) + A(3,:) \end{pmatrix}$$

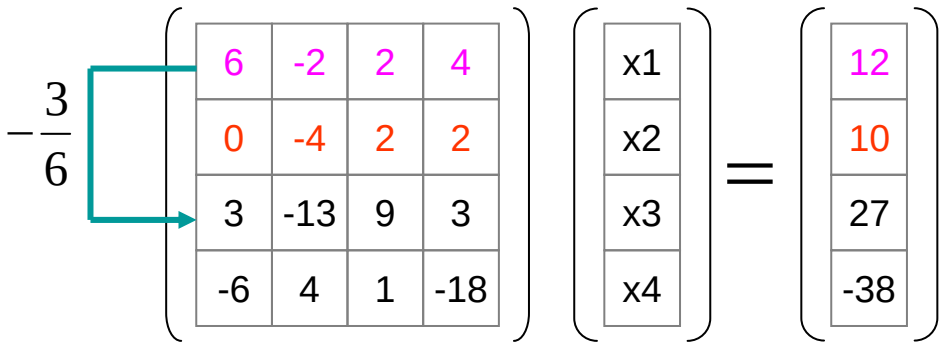
$$\longrightarrow \quad L_{31}(\mu)L_{21}(\lambda) = \begin{pmatrix} 1 & & \\ \lambda & 1 & \\ \mu & & 1 \end{pmatrix} = L_{21}(\lambda)L_{31}(\mu)$$

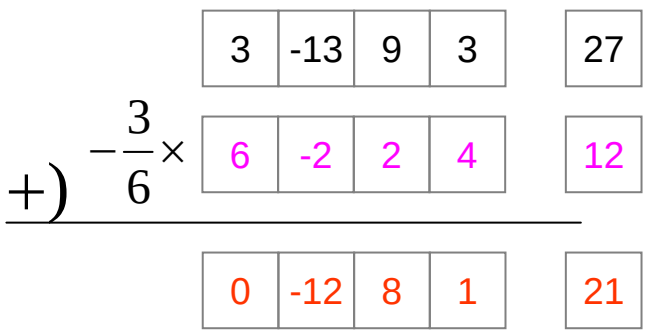
OutLine

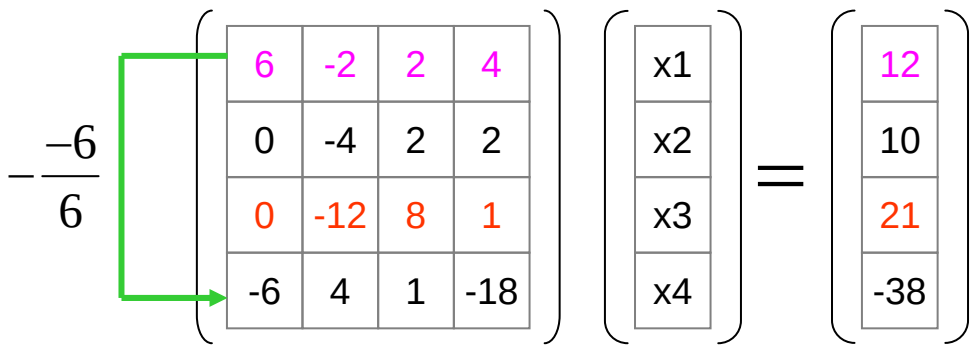
- Basic operation of matrix
- Example of Gaussian Elimination (GE)
  - forward elimination to upper triangle form
  - backward substitution
- Formal description of GE
- MATLAB usage

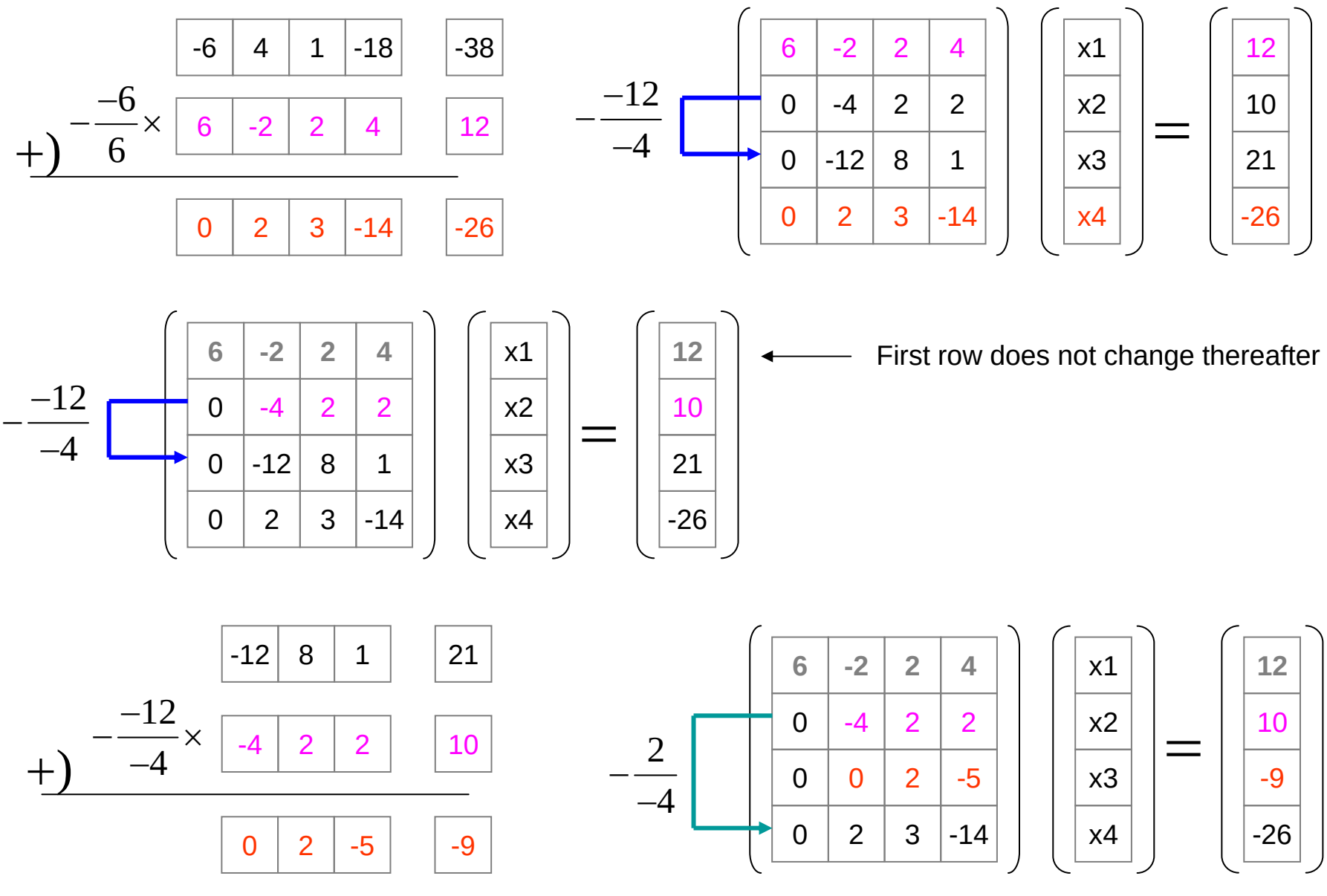

$-\frac{12}{6}$
6 -2 2 4
12 -8 6 10
3 -13 9 3
-6 4 1 -18
x1
x2
x3
x4
=
12
34
27
-38


12 -8 6 10 34
+) $-\frac{12}{6}\times$ 6 -2 2 4 12
0 -4 2 2 10


$-\frac{3}{6}$
6 -2 2 4
0 -4 2 2
3 -13 9 3
-6 4 1 -18
x1
x2
x3
x4
=
12
10
27
-38


3 -13 9 3 27
+) $-\frac{3}{6}\times$ 6 -2 2 4 12
0 -12 8 1 21


$-\frac{-6}{6}$
6 -2 2 4
0 -4 2 2
0 -12 8 1
-6 4 1 -18
x1
x2
x3
x4
=
12
10
21
-38


-6 4 1 -18 -38
+) $-\frac{-6}{6}\times$ 6 -2 2 4 12
0 2 3 -14 -26
$-\frac{-12}{-4}$
6 -2 2 4 | x1 | 12
0 -4 2 2 | x2 | 10
0 -12 8 1 | x3 | 21
0 2 3 -14 | x4 | -26
$-\frac{-12}{-4}$
6 -2 2 4 | x1 | 12
0 -4 2 2 | x2 | 10
0 -12 8 1 | x3 | 21
0 2 3 -14 | x4 | -26
First row does not change thereafter
-12 8 1 21
+) $-\frac{-12}{-4}\times$ -4 2 2 10
0 2 -5 -9
$-\frac{2}{-4}$
6 -2 2 4 | x1 | 12
0 -4 2 2 | x2 | 10
0 0 2 -5 | x3 | -9
0 2 3 -14 | x4 | -26

	2	3	-14	-26
$+)\ -\frac{2}{-4}\times$	-4	2	2	10
	0	4	-13	-21

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \\ -9 \\ -21 \end{pmatrix}$$

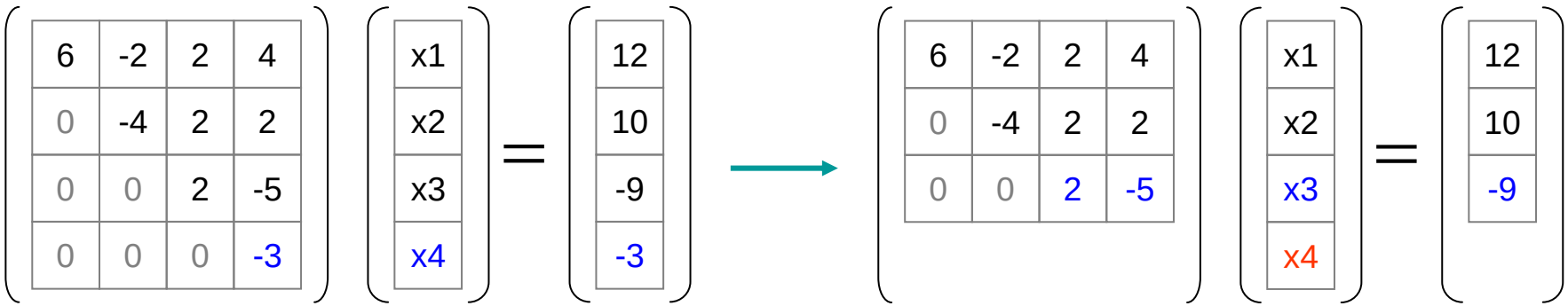
$-\frac{4}{2}$

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \\ -9 \\ -21 \end{pmatrix}$$

	4	-13	-21
$+)\ -\frac{4}{2}\times$	2	-5	-9
	0	-3	-3

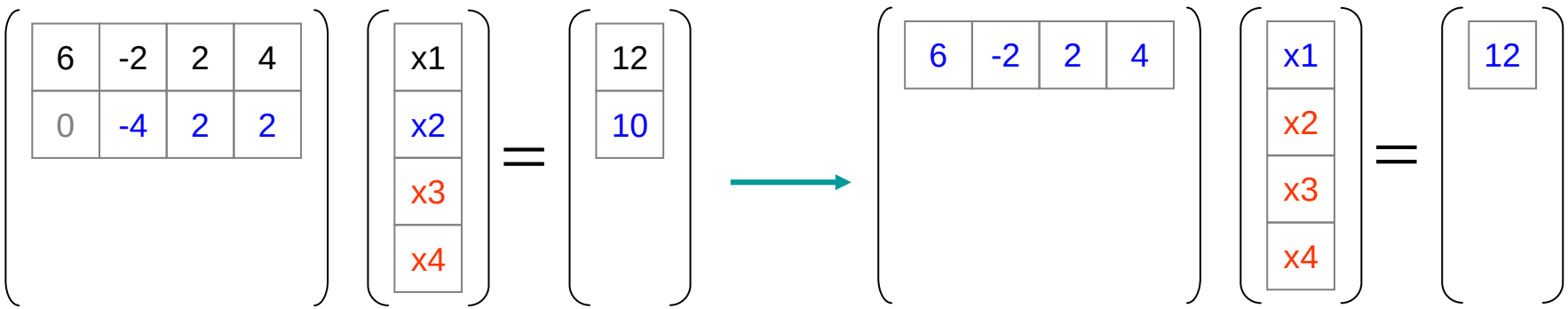
$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \\ -9 \\ -3 \end{pmatrix}$$

Backward substitution: inner-product-based



$-3x_4 = -3 \Rightarrow x_4 = 1$

$2x_3 - 5x_4 = -9 \Rightarrow x_3 = -2$



$-4x_2 + 2x_3 + 2x_4 = 10 \Rightarrow x_2 = -3$

$6x_1 - 2x_2 + 2x_3 + 4x_4 = 12 \Rightarrow x_1 = 1$

Backward substitution: outer-product-based

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \\ -9 \\ -3 \end{pmatrix}$$

$$-3x_4 = -3 \Rightarrow x_4 = 1$$

$$\begin{pmatrix} 6 & -2 & 2 \\ 0 & -4 & 2 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} 12 \\ 10 \\ -9 \end{pmatrix} - x4 \begin{pmatrix} 4 \\ 2 \\ -5 \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \\ -4 \end{pmatrix}$$

$$2x_3 = -4 \Rightarrow x_3 = -2$$

$$\begin{pmatrix} 6 & -2 \\ 0 & -4 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \end{pmatrix} - x3 \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 12 \end{pmatrix}$$

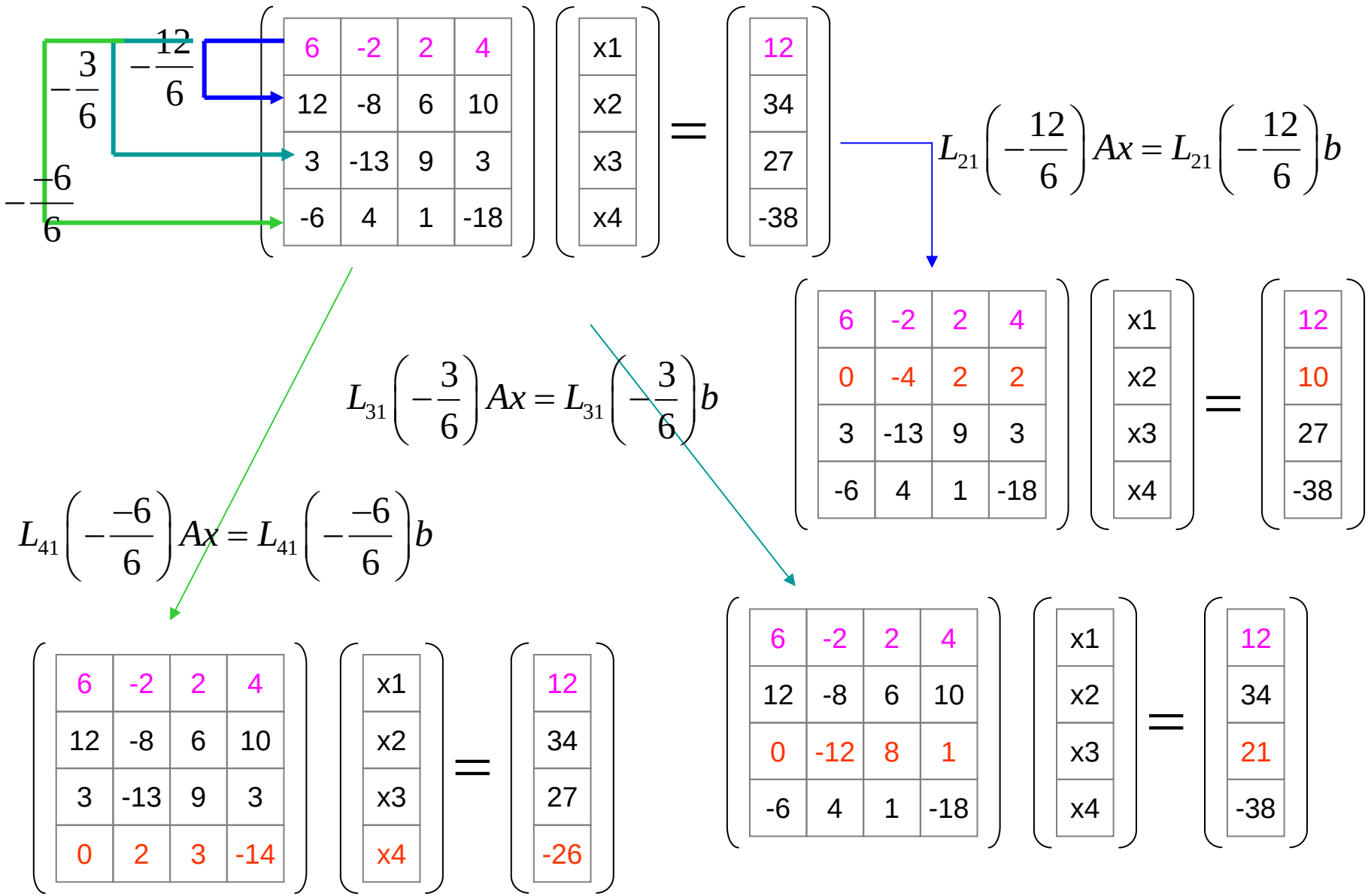
$$-4x_2 = 12 \Rightarrow x_2 = -3$$

$$\begin{pmatrix} 6 \end{pmatrix} \begin{pmatrix} x1 \end{pmatrix} = \begin{pmatrix} 12 \end{pmatrix} - x2 \begin{pmatrix} -2 \end{pmatrix} = \begin{pmatrix} 6 \end{pmatrix}$$

$$6x_1 = 6 \Rightarrow x_1 = 1$$

OutLine

- Basic operation of matrix
- Example of Gaussian Elimination (GE)
- Formal description of GE
  - component-wise and column-wise representation
  - recursive structure
- MATLAB usage


$-\frac{12}{6}$
$-\frac{3}{6}$
$-\frac{-6}{6}$
6 -2 2 4
12 -8 6 10
3 -13 9 3
-6 4 1 -18
x1 x2 x3 x4
= 12 34 27 -38
$L_{21}\left(-\frac{12}{6}\right)Ax = L_{21}\left(-\frac{12}{6}\right)b$
6 -2 2 4
0 -4 2 2
3 -13 9 3
-6 4 1 -18
x1 x2 x3 x4
= 12 10 27 -38
$L_{31}\left(-\frac{3}{6}\right)Ax = L_{31}\left(-\frac{3}{6}\right)b$
$L_{41}\left(-\frac{-6}{6}\right)Ax = L_{41}\left(-\frac{-6}{6}\right)b$
6 -2 2 4
12 -8 6 10
3 -13 9 3
0 2 3 -14
x1 x2 x3 x4
= 12 34 27 -26
6 -2 2 4
12 -8 6 10
0 -12 8 1
-6 4 1 -18
x1 x2 x3 x4
= 12 34 21 -38

Eliminate first column: component-wise

$$L_{41}(1)L_{31}(-0.5)L_{21}(-2)Ax = L_{41}(1)L_{31}(-0.5)L_{21}(-2)b$$

6	-2	2	4
12	-8	6	10
3	-13	9	3
-6	4	1	-18

x1
x2
x3
x4

=

12
34
27
-38

→

6	-2	2	4
0	-4	2	2
0	-12	8	1
0	2	3	-14

x1
x2
x3
x4

=

12
10
21
-26

where $L_{41}(1)L_{31}(-0.5)L_{21}(-2) = \begin{pmatrix} 1 & & & \\ -2 & 1 & & \\ -0.5 & & 1 & \\ 1 & & & 1 \end{pmatrix}$

Exercise: check $\begin{pmatrix} 1 & & & \\ -2 & 1 & & \\ -0.5 & & 1 & \\ 1 & & & 1 \end{pmatrix}\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{pmatrix}$ by MATLAB

Eliminate first column: column-wise

define $A = A^{(1)} = \left(\begin{array}{c|ccc} 6 & -2 & 2 & 4 \\ \hline 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{array}\right) \equiv \begin{pmatrix} a_{11} & B^T \\ C & \bar{A} \end{pmatrix}$ and $L^{(1)} \equiv \left(\begin{array}{c|cc} 1 & & \\ \hline -2 & 1 & \\ -0.5 & & 1 \\ 1 & & 1 \end{array}\right) = \begin{pmatrix} 1 & 0 \\ -C/a_{11} & I_{3\times 3} \end{pmatrix}$

then $L^{(1)}A = \begin{pmatrix} 1 & 0 \\ -\dfrac{C}{a_{11}} & I_{3\times 3} \end{pmatrix} \begin{pmatrix} a_{11} & B^T \\ C & \bar{A} \end{pmatrix} = \begin{pmatrix} a_{11} & B^T \\ 0 & \bar{A} - \dfrac{CB^T}{a_{11}} \end{pmatrix} = \left(\begin{array}{c|ccc} 6 & -2 & 2 & 4 \\ \hline 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{array}\right)$

Exercise: check $A^{(2)} \equiv \bar{A} - \dfrac{CB^T}{a_{11}} = \begin{pmatrix} -4 & 2 & 2 \\ -12 & 8 & 1 \\ 2 & 3 & -14 \end{pmatrix}$ by MATLAB

Notation: $A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{pmatrix} \xrightarrow{L^{(1)}} L^{(1)}A = \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ \hline 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & a_{42}^{(2)} & a_{43}^{(2)} & a_{44}^{(2)} \end{array}\right) = \begin{pmatrix} a_{11}^{(1)} & \times \\ 0 & A^{(2)} \end{pmatrix}$

Eliminate second column: component-wise

$$-\frac{-12}{-4}$$

6	-2	2	4
0	-4	2	2
0	-12	8	1
0	2	3	-14

x1
x2
x3
x4

$=$

12
10
21
-26

$$L_{32}\left(-\frac{-12}{-4}\right)\left(L^{(1)}A = L^{(1)}b\right)$$

$$-\frac{2}{-4}$$

6	-2	2	4
0	-4	2	2
0	0	2	-5
0	2	3	-14

x1
x2
x3
x4

$=$

12
10
-9
-26

6	-2	2	4
0	-4	2	2
0	0	2	-5
0	0	4	-13

x1
x2
x3
x4

$=$

12
10
-9
-21

$$L_{42}\left(-\frac{2}{-4}\right)L_{32}\left(-\frac{-12}{-4}\right)\left(L^{(1)}A = L^{(1)}b\right)$$

Eliminate second column: column-wise

define $L^{(2)} \equiv L_{42}(0.5)\,L_{32}(-3) = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & -3 & 1 & \\ & 0.5 & & 1 \end{pmatrix}$

Exercise: check $L^{(2)}\left(L^{(1)}A\right) = \left(\begin{array}{c|ccc} 1 & & & \\ \hline & 1 & & \\ & -3 & 1 & \\ & 0.5 & & 1 \end{array}\right)\left(\begin{array}{c|ccc} 6 & -2 & 2 & 4 \\ \hline 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{array}\right) = \left(\begin{array}{c|ccc} 6 & -2 & 2 & 4 \\ \hline 0 & -4 & 2 & 2 \\ \hline 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{array}\right)$

Exercise: check $L^{(2)}L^{(1)} = \begin{pmatrix} 1 & & & \\ -2 & 1 & & \\ -0.5 & -3 & 1 & \\ 1 & 0.5 & & 1 \end{pmatrix} \neq L^{(1)}L^{(2)}$, why? Give a simple explanation.

Notation: $L^{(1)}A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & a_{42}^{(2)} & a_{43}^{(2)} & a_{44}^{(2)} \end{pmatrix} \xrightarrow{L^{(2)}} L^{(2)}L^{(1)}A = \left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ \hline 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ \hline 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} \\ 0 & 0 & a_{43}^{(3)} & a_{44}^{(3)} \end{array}\right) = \begin{pmatrix} a_{11}^{(1)} & \times & \times \\ 0 & a_{22}^{(2)} & \times \\ 0 & 0 & A^{(3)} \end{pmatrix}$

Eliminate third column: component-wise

$-\frac{4}{2}$

6	-2	2	4
0	-4	2	2
0	0	2	-5
0	0	4	-13

x1
x2
x3
x4

$=$

12
10
-9
-21

$$L_{43}\left(-\frac{4}{2}\right)\left(L^{(2)}L^{(1)}A = L^{(2)}L^{(1)}b\right)$$

define $L^{(3)} \equiv L_{43}(-2) = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -2 & 1 \end{pmatrix}$

6	-2	2	4
0	-4	2	2
0	0	2	-5
0	0	0	-3

x1
x2
x3
x4

$=$

12
10
-9
-3

then $L^{(3)}L^{(2)}L^{(1)} = \begin{pmatrix} 1 & & & \\ -2 & 1 & & \\ -0.5 & -3 & 1 & \\ 1 & 0.5 & -2 & 1 \end{pmatrix} \neq L^{(2)}L^{(3)}L^{(1)} \neq L^{(2)}L^{(1)}L^{(3)}$

LU-decomposition

Notation:

$$L^{(2)}L^{(1)}A=\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} \\ 0 & 0 & a_{43}^{(3)} & a_{44}^{(3)} \end{pmatrix} \xrightarrow{L^{(3)}} L^{(3)}L^{(2)}L^{(1)}A=\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} \\ 0 & 0 & 0 & a_{44}^{(4)} \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} & \times & \times & \times \\ 0 & a_{22}^{(2)} & \times & \times \\ 0 & 0 & a_{33}^{(3)} & \times \\ 0 & 0 & 0 & A^{(4)} \end{pmatrix}$$

$$L^{(3)}L^{(2)}L^{(1)}A=U=\begin{pmatrix} 6 & -2 & 2 & 4 \\ & -4 & 2 & 2 \\ & & 2 & -5 \\ & & & -3 \end{pmatrix} \longrightarrow A=\left(L^{(3)}L^{(2)}L^{(1)}\right)^{-1}U\equiv LU$$ ***LU*-decomposition**

Exercise: check $L=\left(L^{(1)}\right)^{-1}\left(L^{(2)}\right)^{-1}\left(L^{(3)}\right)^{-1}=\begin{pmatrix} 1 & & & \\ 2 & 1 & & \\ 0.5 & 3 & 1 & \\ -1 & -0.5 & 2 & 1 \end{pmatrix}$

Question: Do you have any effective method to write down matrix L

Question: why don't we care about right hand side vector b

Problems about LU-decomposition

$$A = LU \longrightarrow \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ 2 & 1 & & \\ 0.5 & 3 & 1 & \\ -1 & -0.5 & 2 & 1 \end{pmatrix} \begin{pmatrix} 6 & -2 & 2 & 4 \\ & -4 & 2 & 2 \\ & & 2 & -5 \\ & & & -3 \end{pmatrix}$$

Question 1: what condition does ***LU***-decomposition fail?

Question 2: does any invertible matrix has ***LU***-decomposition?

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \longrightarrow A = LU\,?$$

Question 3: How to measure “goodness of ***LU***-decomposition“?

Question 4: what is order of performance of ***LU***-decomposition (operation count)?

Question 5: How to parallelize ***LU***-decomposition?

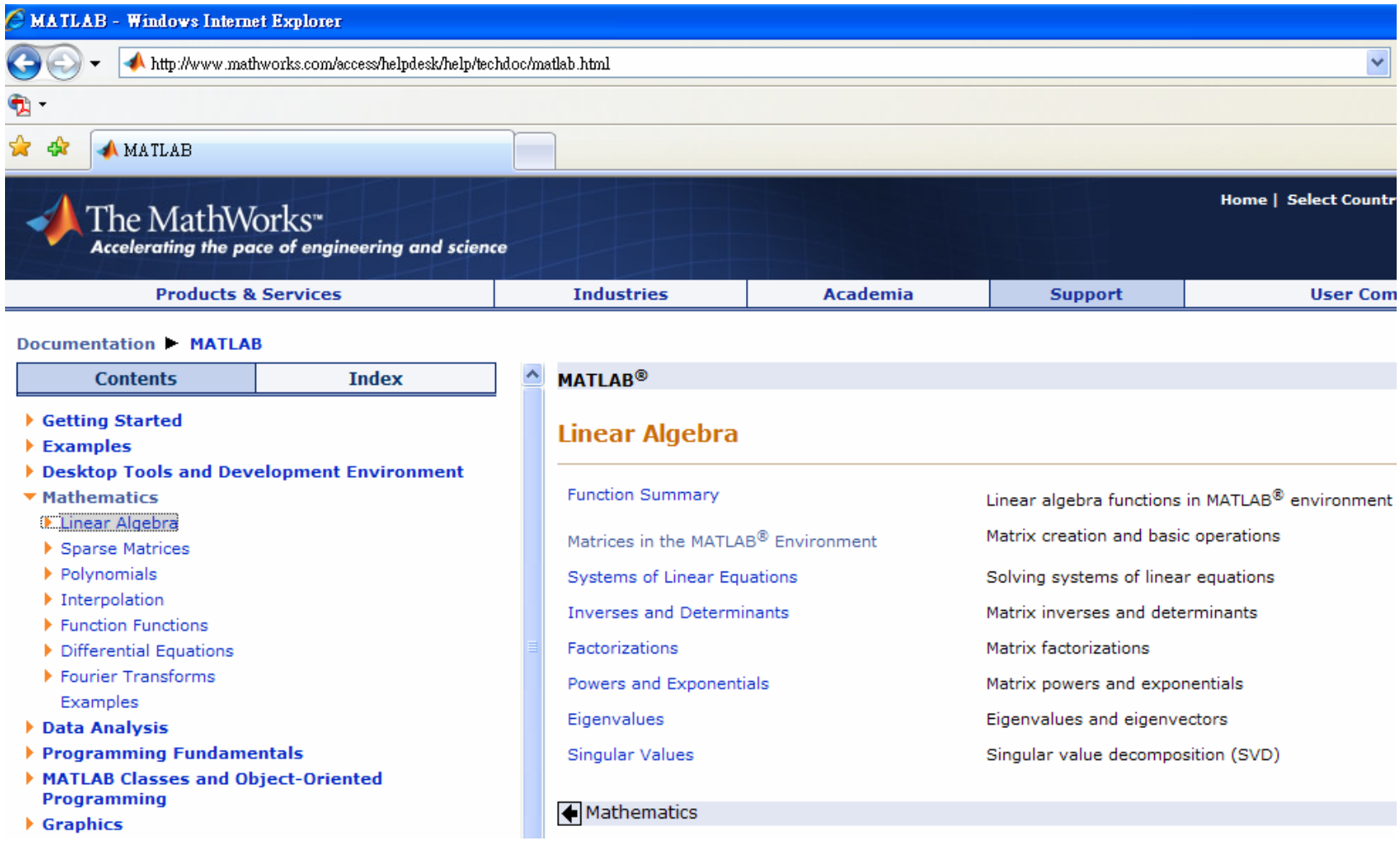
Question 6: How to implement ***LU***-decomposition with help of ***GPU***?

OutLine

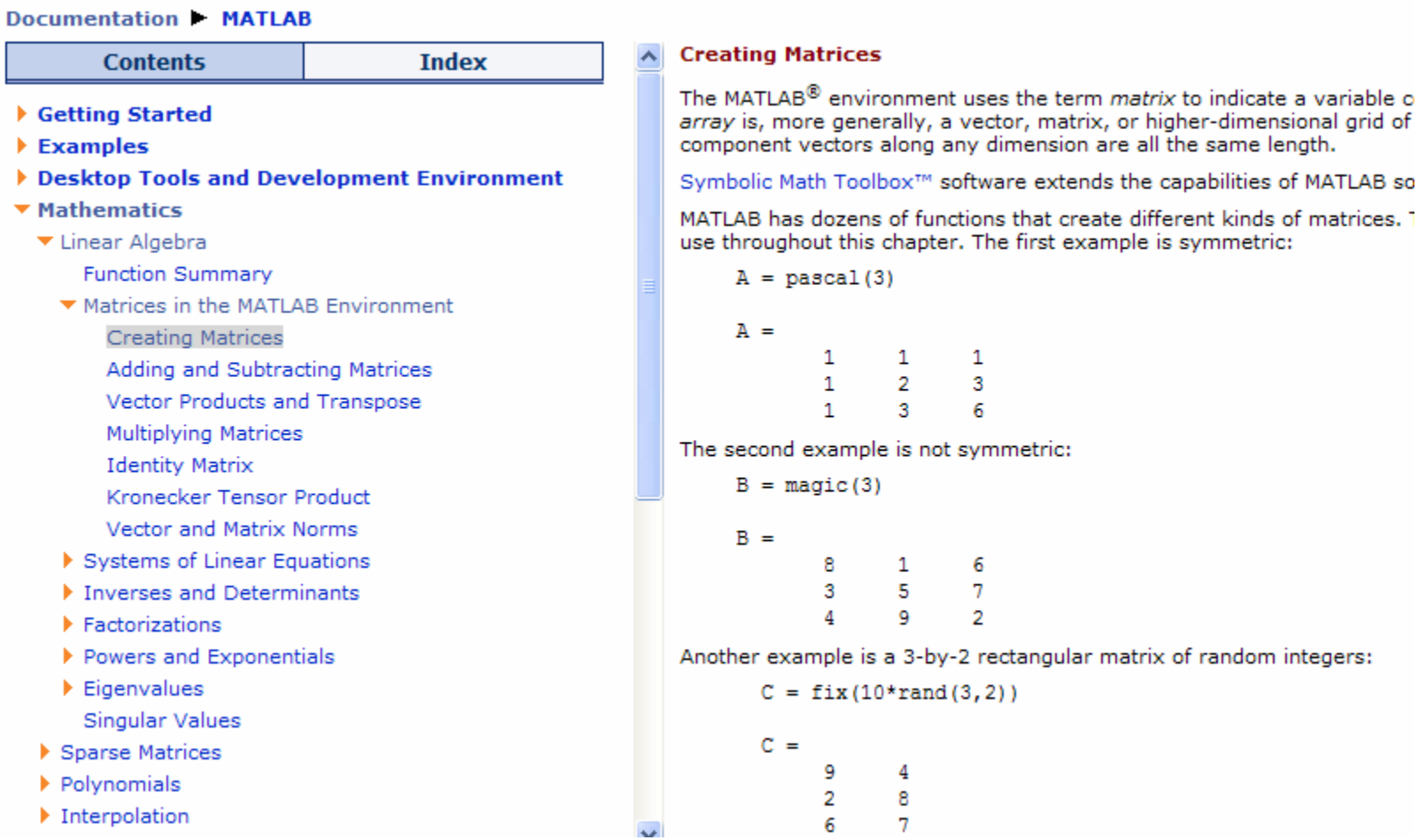
- Basic operation of matrix
- Example of Gaussian Elimination (GE)
- Formal description of GE
- MATLAB usage
  - website resource
  - “help” command
  - M-file

MATLAB website

http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html



MATLAB: create matrix



MATLAB: LU-factorization

Documentation ▶ MATLAB

Contents | Index

- Getting Started
- Examples
- Desktop Tools and Development Environment
- Mathematics
  - Linear Algebra
    - Function Summary
    - Matrices in the MATLAB Environment
      - Creating Matrices
      - Adding and Subtracting Matrices
      - Vector Products and Transpose
      - Multiplying Matrices
      - Identity Matrix
      - Kronecker Tensor Product
      - Vector and Matrix Norms
    - Systems of Linear Equations
    - Inverses and Determinants
    - Factorizations
      - Introduction
      - Cholesky Factorization
      - LU Factorization
      - QR Factorization
    - Powers and Exponentials
    - Eigenvalues
    - Singular Values

Back to Top

LU Factorization

LU factorization, or Gaussian elimination, expresses any square matrix A as the triangular matrix

$$A = LU$$

where L is a permutation of a lower triangular matrix with ones on its diagonal a

The permutations are necessary for both theoretical and computational reasons.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

cannot be expressed as the product of triangular matrices without interchanging

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix}$$

can be expressed as the product of triangular matrices, when ε is small the elem the permutations are not strictly necessary, they are desirable. Partial pivoting e that the elements of U are not much larger than those of A.

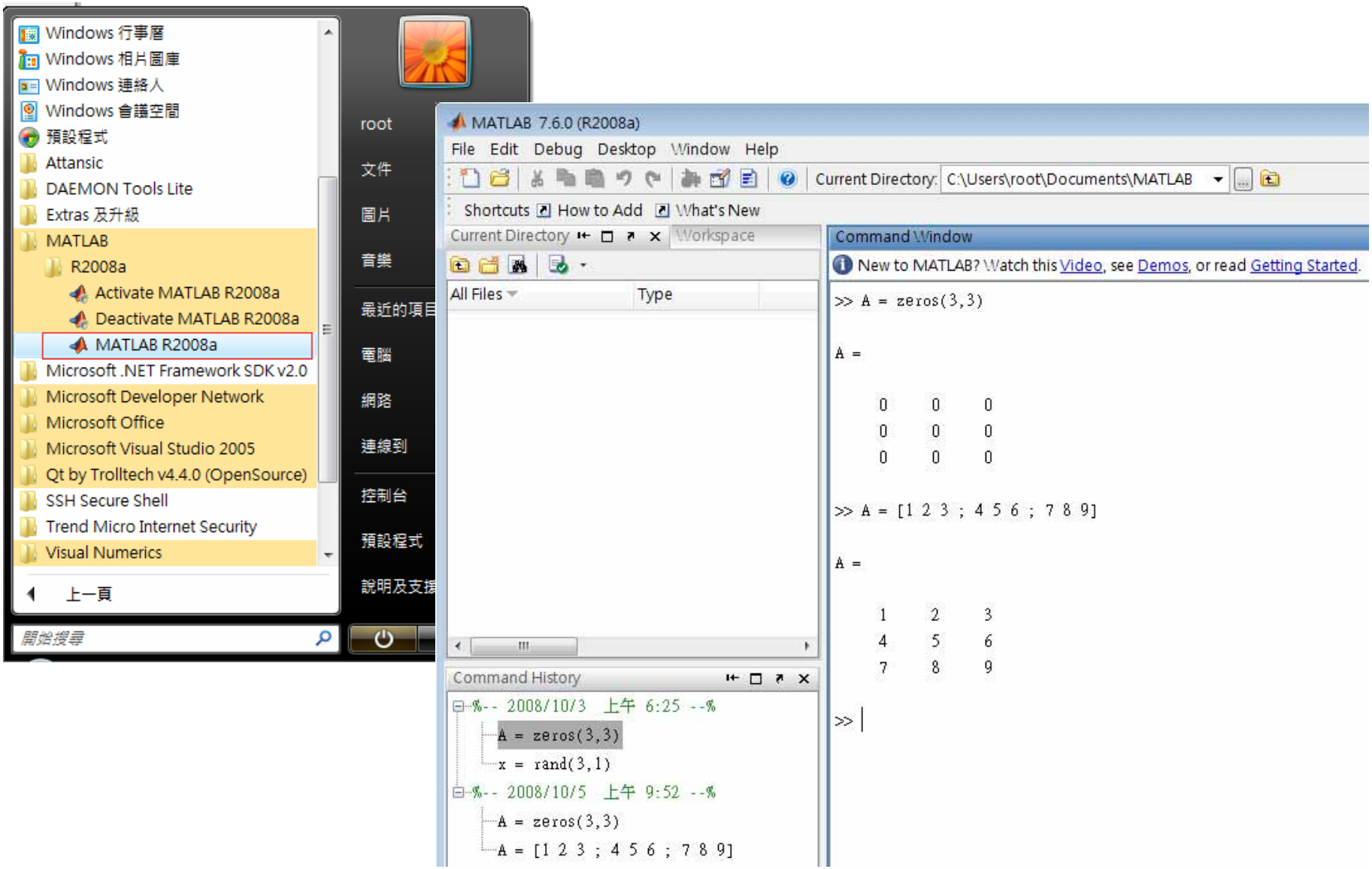
For example

```
[L,U] = lu(B)

L =
    1.0000         0         0
    0.3750    0.5441    1.0000
    0.5000    1.0000         0

U =
    8.0000    1.0000    6.0000
         0    8.5000   -1.0000
```

Start MATLAB 2008



Command “help” : documentation

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting St

```
>> help >
  Operators and special characters.

  Arithmetic operators.
    plus        - Plus                                +
    uplus       - Unary plus                          +
    minus       - Minus                               -
    uminus      - Unary minus                         -
    mtimes      - Matrix multiply                     *
    times       - Array multiply                      .*
    mpower      - Matrix power                        ^
    power       - Array power                         .^
    mldivide    - Backslash or left matrix divide     \
    mrdivide    - Slash or right matrix divide        /
    ldivide     - Left array divide                   .\
    rdivide     - Right array divide                  ./
    kron        - Kronecker tensor product            kron

  Relational operators.
    eq          - Equal                               ==
    ne          - Not equal                           ~=
    lt          - Less than                           <
    gt          - Greater than                        >
    le          - Less than or equal                  <=
    ge          - Greater than or equal               >=
```

!= in C

```
Logical operators.
  relop       - Short-circuit logical AND          &&
  relop       - Short-circuit logical OR           ||
  and         - Element-wise logical AND           &
  or          - Element-wise logical OR            |
  not         - Logical NOT                        ~
  xor         - Logical EXCLUSIVE OR
  any         - True if any element of vector is nonzero
  all         - True if all elements of vector are nonzero

Special characters.
  colon       - Colon                              :
  paren       - Parentheses and subscripting       ( )
  paren       - Brackets                           [ ]
  paren       - Braces and subscripting            { }
  punct       - Function handle creation           @
```

Loop in MATLAB

```
>> help while
 WHILE  Repeat statements an indefinite number of times.
    The general form of a WHILE statement is:

       WHILE expression
         statements
       END

    The statements are executed while the real part of the expression
    has all non-zero elements. The expression is usually the result of
    expr rop expr where rop is ==, <, >, <=, >=, or ~=.

    The BREAK statement can be used to terminate the loop prematurely.

    For example (assuming A already defined):

            E = 0*A; F = E + eye(size(E)); N = 1;
            while norm(E+F-E,1) > 0,
               E = E + F;
               F = A*F/N;
               N = N + 1;
            end

    See also for, if, switch, break, continue, end.

    Reference page in Help browser
       doc while
```

```
>> help for
 FOR    Repeat statements a specific number of times.
    The general form of a FOR statement is:

       FOR variable = expr, statement, ..., statement END

    The columns of the expression are stored one at a time in
    the variable and then the following statements, up to the
    END, are executed. The expression is often of the form X:Y,
    in which case its columns are simply scalars. Some examples
    (assume N has already been assigned a value).

         for R = 1:N
             for C = 1:N
                 A(R,C) = 1/(R+C-1);
             end
         end

    Step S with increments of -0.1
         for S = 1.0: -0.1: 0.0, do_some_task(S), end

    Set E to the unit N-vectors
         for E = eye(N), do_some_task(E), end

    Long loops are more memory efficient when the colon expression appears
    in the FOR statement since the index vector is never created.

    The BREAK statement can be used to terminate the loop prematurely.

    See also parfor, if, while, switch, break, continue, end, colon.
```

Decision-making in MATLAB

```
>> help if
 IF Conditionally execute statements.
    The general form of the IF statement is

       IF expression
         statements
       ELSEIF expression
         statements
       ELSE
         statements
       END

    The statements are executed if the real part of the expression
    has all non-zero elements. The ELSE and ELSEIF parts are optional.
    Zero or more ELSEIF parts can be used as well as nested IF's.
    The expression is usually of the form expr rop expr where
    rop is ==, <, >, <=, >=, or ~=.

    Example
       if I == J
         A(I,J) = 2;
       elseif abs(I-J) == 1
         A(I,J) = -1;
       else
         A(I,J) = 0;
       end

    See also relop, else, elseif, end, for, while, switch.
```

```
>> help switch

 SWITCH Switch among several cases based on expression.
    The general form of the SWITCH statement is:

        SWITCH switch_expr
          CASE case_expr,
            statement, ..., statement
          CASE {case_expr1, case_expr2, case_expr3,...}
            statement, ..., statement
         ...
          OTHERWISE,
            statement, ..., statement
        END

 To execute a certain block of code based on what the string, METHOD,
 is set to,

     method = 'Bilinear';

     switch lower(METHOD)
       case {'linear','bilinear'}
         disp('Method is linear')
       case 'cubic'
         disp('Method is cubic')
       case 'nearest'
         disp('Method is nearest')
       otherwise
         disp('Unknown method.')
     end

     Method is linear

 See also CASE, OTHERWISE, IF, WHILE, FOR, END.
```

IO in MATLAB

```
>> help fprintf
 FPRINTF Write formatted data to file.
    COUNT = FPRINTF(FID,FORMAT,A,...) formats the data in the real
    part of array A (and in any additional array arguments), under
    control of the specified FORMAT string, and writes it to the file
    associated with file identifier FID.  COUNT is the number of bytes
    successfully written. FID is an integer file identifier obtained
    from FOPEN. It can also be 1 for standard output (the screen) or 2
    for standard error. If FID is omitted, output goes to the screen.
                          .
                          .
                          .
    Create a text file called exp.txt containing a short table of the
    exponential function. (On Windows platforms, it is recommended that
    you use FOPEN with the mode set to 'wt' to create a text file for
    writing.)

        x = 0:.1:1; y = [x; exp(x)];
        fid = fopen('exp.txt','wt');
        fprintf(fid,'%6.2f  %12.8f\n',y);
        fclose(fid);

    Now examine the contents of exp.txt:

        type exp.txt
           0.00    1.00000000
           0.10    1.10517092
                ...
           1.00    2.71828183

    See the reference page in the online help for other exceptions,
    extensions, or platform-specific behavior.

    See also fopen, fscanf, sprintf, fwrite, disp, diary, save, input
```

```
>> help disp

 DISP Display array.
    DISP(X) displays the array, without printing the array name.  In
    all other ways it's the same as leaving the semicolon off an
    expression except that empty arrays don't display.

    If X is a string, the text is displayed.

    See also INT2STR, NUM2STR, SPRINTF, RATS, FORMAT.
```

```
>> help error

 ERROR  Display message and abort function.
    ERROR('MSG') displays the text MSG and causes an error exit
    from an M-file to the keyboard. If the string is empty, no
    action is taken.

    See also WARNING, LASTERR, ERRORDLG, DISP, DBSTOP.
```

LU-factorization in MATLAB

```
>> help lu
 LU     LU factorization.
    [L,U] = LU(A) stores an upper triangular matrix in U and a
    "psychologically lower triangular matrix" (i.e. a product of lower
    triangular and permutation matrices) in L, so that A = L*U. A can be
    rectangular.

    [L,U,P] = LU(A) returns unit lower triangular matrix L, upper
    triangular matrix U, and permutation matrix P so that P*A = L*U.

    [L,U,p] = LU(A,'vector') returns the permutation information as a
    vector instead of a matrix.  That is, p is a row vector such that
    A(p,:) = L*U.  Similarly, [L,U,P] = LU(A,'matrix') returns a
    permutation matrix P.  This is the default behavior.

    Y = LU(A) returns the output from LAPACK'S DGETRF or ZGETRF routine if
    A is full. If A is sparse, Y contains the strict lower triangle of L
    embedded in the same matrix as the upper triangle of U. In both full
    and sparse cases, the permutation information is lost.

    [L,U,P,Q] = LU(A) returns unit lower triangular matrix L, upper
    triangular matrix U, a permutation matrix P and a column reordering
    matrix Q so that P*A*Q = L*U for sparse non-empty A. This uses UMFPACK
    and is significantly more time and memory efficient than the other
    syntaxes, even when used with COLAMD.
```

$A = LU$

$PA = LU$ P : *permutation*

$PAQ = LU$ P, Q : *permutation*

M-file in MATLAB

matvec.m

Description of function matvec, write specificaiton of input parameter

Consistency check

```
 1 function  b = matvec(A,x)
 2 %
 3 %  matrix-vector product: inner-product based
 4 %
 5 %  Input - A is m x n matrix
 6 %          x is n x 1 vector
 7 %  output - b is m x 1 vector
 8 %
 9 if ( 2 ~= ndims(A) )
10   error('A is not a 2D matrix');
11 end
12 if ( 2 ~= ndims(x) )
13   error('x is not a 1D vector');
14 end
15
16 [m,n] = size(A) ;
17 [mx, nx] = size(x) ;
18
19 % verify dimension of A and x
20 if ( nx ~= 1 )
21    error('x is not a column vector');
22 end
23 if ( mx ~= n )
24    error('dimension of A and x do not match');
25 end
26
27 b = zeros(m,1) ;
28
29 for i = 1:m
30   for j =1:n
31      b(i) = A(i,j)*x(j) ;
32   end
33 end
34
```

$compute \;\; b = Ax$

Inner-product based

$for \quad i \;\; = 1:4$

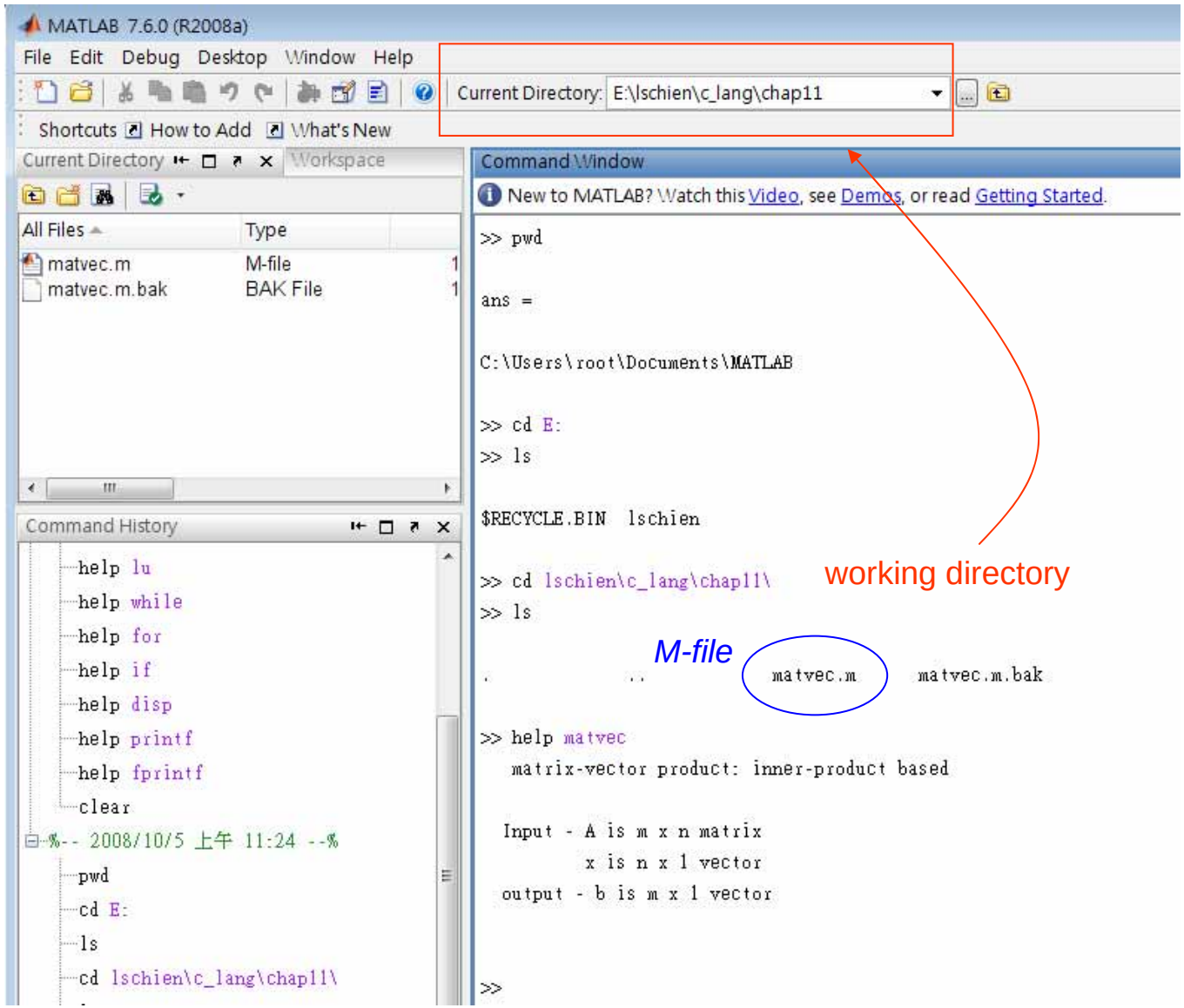
$\quad for \quad j \;\; = 1:4$

$\quad\quad b(i) = A(i,j) * x(j)$

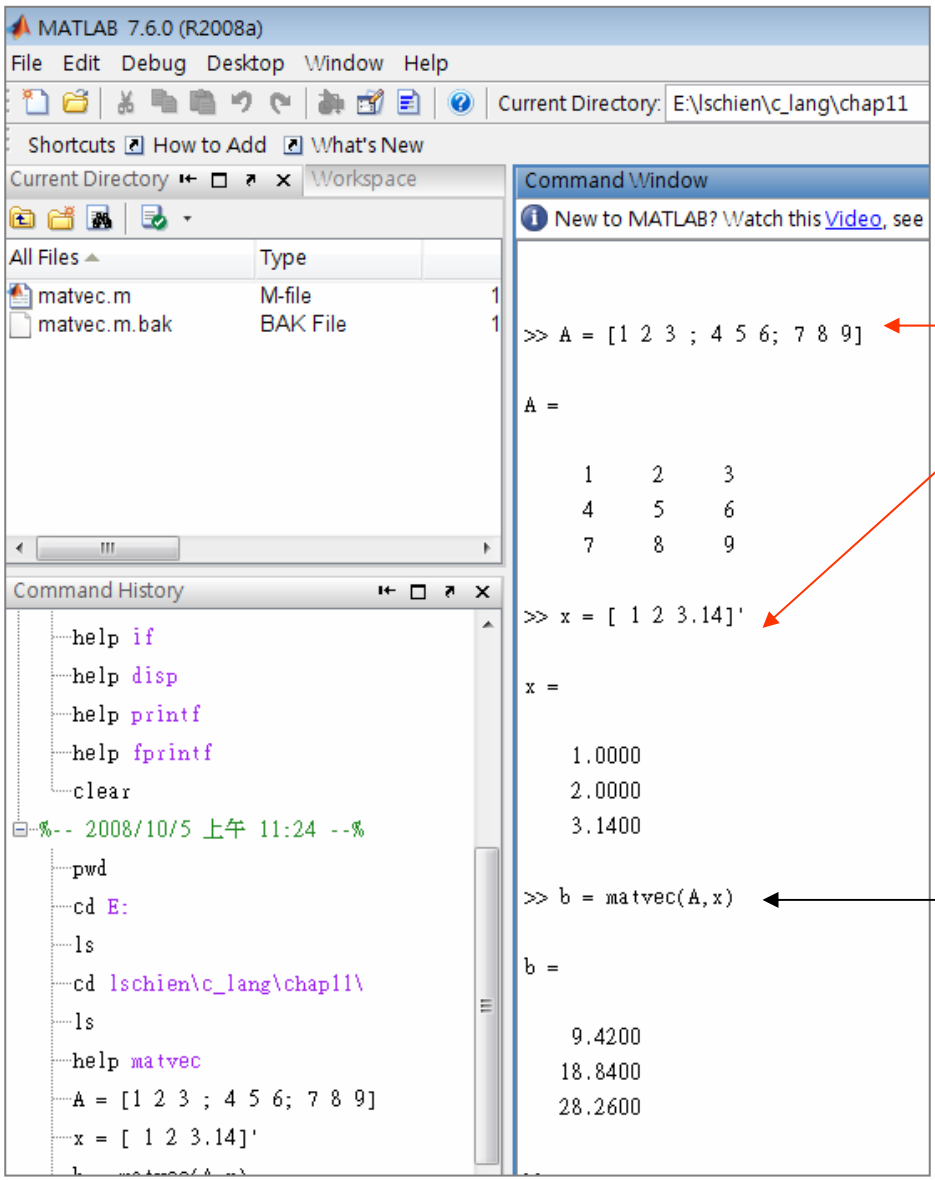
$\quad end$

end

Move to working directory



Execute M-file



Construct matrix

Execute M-file, like function call in C-language

Exercise: forward / backward substitution

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = LU = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ l_{31} & l_{32} & l_{33} & \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ & u_{22} & u_{23} & u_{24} \\ & & u_{33} & u_{34} \\ & & & u_{44} \end{pmatrix}$$

$$Ax=b \longrightarrow LUx=b \longrightarrow \begin{array}{l} y=L^{-1}b \\ x=U^{-1}y \end{array}$$

Forward substitution $y=L^{-1}b$ since $y_1 \to y_2 \to y_3 \to y_4$

backward substitution $x=U^{-1}y$ since $x_4 \to x_3 \to x_2 \to x_1$

Write MATLAB code to do forward substitution and backward substitution

Exercise: LU-decomposition

$$A=\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} \quad L^{(1)} \equiv \begin{pmatrix} 1 & & & \\ -2 & 1 & & \\ -0.5 & & 1 & \\ 1 & & & 1 \end{pmatrix} \quad L^{(2)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & -3 & 1 & \\ & 0.5 & & 1 \end{pmatrix} \quad L^{(3)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -2 & 1 \end{pmatrix}$$

$$L^{(3)}L^{(2)}L^{(1)}A = U = \begin{pmatrix} 6 & -2 & 2 & 4 \\ & -4 & 2 & 2 \\ & & 2 & -5 \\ & & & -3 \end{pmatrix} \longrightarrow A = \left(L^{(3)}L^{(2)}L^{(1)}\right)^{-1} U \equiv LU$$

LU-decomposition

You should find recursive structure of the decomposition