# NUMPY VÀ PHÉP TÍNH MA TRẬN

**Nguyễn Mạnh Hùng**

ĐẠI HỌC GTVT

03 - 2023

# Nội dung

1. Ma trận
2. Các phép toán trên ma trận
3. Hệ phương trình tuyến tính
4. Thực hành

# Khái niệm

## Ma trận (matrix)

Ma trận kích thước $m \times n$ là một bảng chữ nhật gồm $m.n$ số được xếp thành $m$ hàng và $n$ cột:

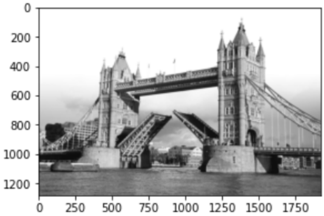
$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

trong đó $a_{ij}$ là phần tử của ma trận nằm ở hàng thứ $i$ và cột thứ $j$.

# Khái niệm

Ma trận có thể sử dụng để mô tả dữ liệu dạng bảng:

- *Ảnh xám*: là ma trận độ sáng của các điểm ảnh




```
[[255.        255.        ... 255.        255.       ]
 [255.        255.        ... 255.        255.       ]
 [255.        255.        ... 255.        255.       ]
 ...
 [105.333336 108.333336 ...  85.333336  87.333336]
 [104.666664 107.333336 ...  84.333336  86.333336]
 [101.666664 103.666664 ...  81.333336  81.333336]]
```

- *Lợi nhuận đầu tư*: ma trận lợi nhuận của danh mục đầu tư gồm $n$ tài sản trong khoảng thời gian $T$.

| Date | AAPL | GOOG | MMM | AMZN |
|---|---|---|---|---|
| March 1, 2016 | 0.00219 | 0.00006 | −0.00113 | 0.00202 |
| March 2, 2016 | 0.00744 | −0.00894 | −0.00019 | −0.00468 |
| March 3, 2016 | 0.01488 | −0.00215 | 0.00433 | −0.00407 |

# Khái niệm

- **Ma trận vuông** (square matrix): có số hàng = số cột = $n$, gọi là ma trận là vuông cấp $n$. Ví dụ ma trận vuông cấp 3:

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 3 & 1 & 5 \\ -2 & 4 & 4 \end{pmatrix}$$

- Giả sử $A = (a_{ij})_{n \times n}$ là một ma trận vuông cấp $n$. Dãy số:

$$(a_{11}, a_{22}, \ldots, a_{nn})$$

được gọi là **đường chéo** (diagonal) của ma trận. Tổng tất cả các phần tử thuộc đường chéo được gọi là "**vết**" của ma trận:

$$\mathrm{trace}(A) = a_{11} + a_{22} + \cdots + a_{nn}$$

# Khái niệm

- **Ma trận tam giác trên/dưới** (upper/lower triangle matrix): là ma trận vuông, có các phần tử nằm phía dưới/trên đường chéo bằng 0:

$$U = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}; \quad L = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \dots & \dots & \ddots & \dots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

- **Ma trận đường chéo** (diagonal matrix): là ma trận vuông, có các phần tử nằm ngoài đường chéo bằng 0:

$$L = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

# Khái niệm

- **Ma trận đơn vị** (identity matrix): là ma trận vuông, các phần tử nằm trên đường chéo bằng 1, các phần tử còn lại bằng 0. Ví dụ:

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Ma trận không** (zero matrix): có các phần tử đều bằng 0. Ví dụ:

$$\theta_{2\times 3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

# Khởi tạo ma trận

- Tạo ma trận bằng mảng ndarray:

```
[1]: import numpy as np
     a=np.array([[1,-2,1],[2,4,-3]])
     b=np.arange(3)
     print("a = ",a)
     print("b = ",b)

a =  [[ 1 -2  1]
 [ 2  4 -3]]
b =  [0 1 2]
```

- Ghép nối, xóa mảng:

```
In [2]: c=np.vstack([a,b])
        d=np.delete(c,1,axis=0)
        e=np.delete(c,1,axis=1)
        print("Ghép ma trận",c,sep="\n")
        print("Xóa hàng",d,sep="\n")
        print("Xóa cột",e,sep="\n")

Ghép ma trận
[[ 1 -2  1]
 [ 2  4 -3]
 [ 0  1  2]]
Xóa hàng
[[ 1 -2  1]
 [ 0  1  2]]
Xóa cột
[[ 1  1]
 [ 2 -3]
 [ 0  2]]
```

# Khởi tạo ma trận

Tạo ma trận bằng hàm tích hợp sẵn trong NumPy:

- **np.empty**(*shape*, *dtype*, *order*): trả về một ma trận mà các phần tử chưa được khởi tạo,

  | | |
  |---|---|
  | *shape* : | kích thước của ma trận, |
  | *dtype* : | (optional) kiểu dữ liệu phần tử, |
  | *order* : | (optional) 'C' hoặc 'F'. |

  Giá trị của các phần tử của ma trận là các số sẵn có trong các ô nhớ được cấp phát.
- **np.zeros**(*shape*): trả về ma trận không.
- **np.ones**(*shape*) : trả về ma trận có phần tử bằng 1.

# Khởi tạo ma trận

- **np.eye**(*n*, *M*, *k*, *dtype*): trả về một ma trận mà các phần tử thuộc đường chéo bằng 1, các phần tử còn lại bằng 0,

| | | |
|---:|---|---|
| *n* | : | số hàng của ma trận, |
| *M* | : | số cột của ma trận, giá trị mặc định bằng *n*, |
| *k* | : | chỉ số đường chéo, |
| *dtype* | : | kiểu dữ liệu phần tử. |

- **np.random.rand**(*shape*): trả về ma trận mà giá trị phần tử được sinh ngẫu nhiên trong khoảng [0,1).

# Khởi tạo ma trận

- **np.identity**($n$): trả về ma trận đơn vị cấp $n$.

```
In [17]: np.identity(3)

Out[17]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

- **np.diag**($v$): trả về một ma trận đường chéo, trong đó $v$ là một danh sách hoặc một kiểu tương đương.

```
In [18]: a=np.random.randint(-5,5,3)
         np.diag(a)

Out[18]: array([[ 3,  0,  0],
                [ 0, -3,  0],
                [ 0,  0,  3]])
```

# Khởi tạo ma trận

- **np.tril**$(A,k)$: tạo ra ma trận tam giác dưới.
- **np.triu**$(A,k)$: tạo ra ma trận tam giác trên.

  $A$ : mảng 2 chiều hoặc tương đương,
  $k$ : (optional) chỉ số đường chéo, mặc định $k = 0$.

```
In [19]: a=np.random.randint(-5,5,9).reshape(3,3)
         print(a)
         print(np.tril(a))
         print(np.triu(a,1))

[[ 3 -2  4]
 [-4 -2  4]
 [-2  3  3]]

[[ 3  0  0]      [[ 0 -2  4]
 [-4 -2  0]       [ 0  0  4]
 [-2  3  3]]      [ 0  0  0]]
```

# Các phép toán

## Hai ma trận bằng nhau

$$\begin{pmatrix} 9 & x \\ y & 5 \end{pmatrix} = \begin{pmatrix} 9 & 4 \\ 6 & 5 \end{pmatrix} \Leftrightarrow \begin{cases} x = 4 \\ y = 6 \end{cases}$$

## Chuyển vị ma trận (transpose)

$$\begin{pmatrix} 1 & 5 \\ 7 & 2 \\ -3 & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & 7 & -3 \\ 5 & 2 & 1 \end{pmatrix}$$

# Các phép toán

## Cộng hai ma trận (addition)

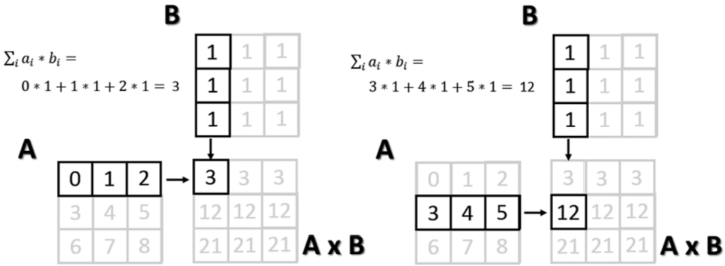
$$\begin{pmatrix} 2 & 1 \\ -3 & 5 \\ 4 & 3 \end{pmatrix} + \begin{pmatrix} 7 & -1 \\ 4 & 2 \\ -6 & 1 \end{pmatrix} = \begin{pmatrix} 2+7 & 1+(-1) \\ -3+4 & 5+2 \\ 4+(-6) & 3+1 \end{pmatrix} = \begin{pmatrix} 9 & 0 \\ 1 & 7 \\ -2 & 4 \end{pmatrix}$$

## Nhân một số với ma trận (scalar multiplication)

$$-2\begin{pmatrix} 1 & 6 \\ 2 & 5 \\ 9 & 3 \end{pmatrix} = \begin{pmatrix} (-2).1 & (-2).6 \\ (-2).2 & (-2).5 \\ (-2).9 & (-2).3 \end{pmatrix} = \begin{pmatrix} -2 & -12 \\ -4 & -10 \\ -18 & -6 \end{pmatrix}$$

# Các phép toán

## Nhân ma trận với ma trận (matrix multiplication)




## Lũy thừa của ma trận (power of a matrix)

$$A^k = \underbrace{A \cdots A}_{k \text{ lần}}$$

# Các phép toán

## Định thức (determinant)

$$\begin{vmatrix} 1 & 4 & -2 \\ 3 & 2 & 1 \\ -6 & 0 & 3 \end{vmatrix} = 1(-1)^{1+1}\begin{vmatrix} 2 & 1 \\ 0 & 3 \end{vmatrix} + 4(-1)^{1+2}\begin{vmatrix} 3 & 1 \\ -6 & 3 \end{vmatrix} + (-2)(-1)^{1+3}\begin{vmatrix} 3 & 2 \\ -6 & 0 \end{vmatrix}$$

$$= 1(6-0) - 4(9+6) - 2(0+12) = -78$$

## Hạng (rank)

$$A = \begin{bmatrix} 2 & 5 & -3 & -4 \\ 4 & 7 & -4 & -3 \\ 6 & 9 & -5 & 2 \\ 0 & -9 & 6 & 5 \end{bmatrix} \Rightarrow v_1 = \begin{bmatrix} 2 \\ 4 \\ 6 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 5 \\ 7 \\ 9 \\ -9 \end{bmatrix}, v_3 = \begin{bmatrix} -3 \\ -4 \\ -5 \\ 6 \end{bmatrix}, v_4 = \begin{bmatrix} -4 \\ -3 \\ 2 \\ 5 \end{bmatrix}$$

$$\Rightarrow \mathrm{rank}(A) = \dim(\mathrm{span}(v_1, v_2, v_3, v_4))$$

# Tính toán với NumPy

- Cộng hai ma trận:

```
In [6]: a=np.array([[-1,4,2],[3,1,-5]])
        b=np.arange(6).reshape(2,3)
        print(a," +",b," =",a+b,sep="\n")

[[-1  4  2]
 [ 3  1 -5]]
 +
[[0 1 2]
 [3 4 5]]
 =
[[-1  5  4]
 [ 6  5  0]]
```

- Nhân một số với ma trận:

```
In [7]: a=np.array([[-1,4,2],[3,1,-5]])
        -2*a

Out[7]: array([[ 2, -8, -4],
               [-6, -2, 10]])
```

# Tính toán với NumPy

- Nhân ma trận với ma trận: hàm **np.matmul**()

```
In [8]: a=np.array([[-1,4,2],[3,1,-5]])
        b=np.random.randint(-10,10,(3,2))
        print(b)
        print("ab=",np.matmul(a,b),sep="\n")
        print("ba=",np.matmul(b,a),sep="\n")

[[-4  4]
 [ 3  6]
 [-1  1]]
ab=
[[14 22]
 [-4 13]]
ba=
[[ 16 -12 -28]
 [ 15  18 -24]
 [  4  -3  -7]]
```

# Tính toán với NumPy

- Nhân ma trận với ma trận: hàm **np.dot**()

```
In [9]: a=np.array([[-1,4,2],[3,1,-5]])
        b=np.random.randint(-10,10,(3,2))
        print(b)
        print("ab=",np.dot(a,b),sep="\n")

[[-9  8]
 [-7 -3]
 [ 6 -3]]
ab=
[[ -7 -26]
 [-64  36]]
```

# Tính toán với NumPy

**Chú ý:** Phép toán $A * B$ sẽ được thực hiện bằng cách nhân các phần tử của ma trận $A$ với phần tử của ma trận $B$ ở vị trí tương ứng. Do đó tích $A * B$ không phải là phép nhân ma trận với ma trận như được định nghĩa trong đại số ma trận.

```
In [10]: a=np.array([[3,2],[-4,1]])
         b=np.arange(4).reshape(2,2)
         print(a,"*",b,"=",a*b,sep="\n")

[[ 3  2]
 [-4  1]]
*
[[0 1]
 [2 3]]
=
[[ 0  2]
 [-8  3]]
```

# Tính toán với NumPy

- Lũy thừa ma trận: hàm **np.linalg.matrix_power**()

```
In [11]: x=np.arange(9).reshape(3,3)
         print(np.matmul(x,x))
         np.linalg.matrix_power(x,2)

         [[ 15  18  21]
          [ 42  54  66]
          [ 69  90 111]]

Out[11]: array([[ 15,  18,  21],
                [ 42,  54,  66],
                [ 69,  90, 111]])
```

# Tính toán với NumPy

- Chuyển vị ma trận: hàm **np.transpose**()

```
In [12]: a=np.array([[-1,4,2],[3,1,-5]])
         np.transpose(a)

Out[12]: array([[-1,  3],
                [ 4,  1],
                [ 2, -5]])
```

```
In [13]: a.T

Out[13]: array([[-1,  3],
                [ 4,  1],
                [ 2, -5]])
```

- Tính "vết" của ma trận vuông: hàm **np.trace**()

```
In [14]: A=np.random.randint(-5,5,(3,3))
         print(A)
         print("trace(A) =",np.trace(A))

         [[ 1  4 -5]
          [ 4 -1 -3]
          [ 2 -5 -1]]
         trace(A) = -1
```

# Tính toán với NumPy

- Định thức của ma trận: hàm **np.linalg.det()**

```
In [15]: a=np.arange(9).reshape(3,3)
         print(a)
         print("det(a)=",np.linalg.det(a))

[[0 1 2]
 [3 4 5]
 [6 7 8]]
det(a)= 0.0
```

- Hạng của ma trận: hàm **np.linalg.matrix_rank()**

```
In [16]: a=np.array([[-3,1,1,1],[1,-3,1,1],[1,1,-3,1],[1,1,1,-3]])
         print(a)
         print("rank(a) =",np.linalg.matrix_rank(a))

[[-3  1  1  1]
 [ 1 -3  1  1]
 [ 1  1 -3  1]
 [ 1  1  1 -3]]
rank(a) = 3
```

# Hệ phương trình tuyến tính

## Biểu diễn ma trận

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Kí hiệu các ma trận

$$\begin{aligned} \mathrm{A} = [a_{ij}]_{m\times n} &: \quad \text{ma trận hệ số của ẩn} \\ \mathrm{x} = [x_j]_{n\times 1} &: \quad \text{ma trận ẩn} \\ \mathrm{b} = [b_i]_{m\times 1} &: \quad \text{ma trận hệ số tự do} \end{aligned}$$

# Giải hệ phương trình với NumPy

- **np.linalg.inv**$(A)$ : trả về ma trận nghịch đảo của ma trận vuông $A$

```
In [1]: import numpy as np
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        print("detA=",np.linalg.det(A))
        print("inv(A)=",np.linalg.inv(A))

        detA= -2.0
        inv(A)= [[-13.    7.    4.5]
         [-10.    5.    3. ]
         [ -2.    1.    0.5]]
```

- **np.linalg.solve**$(A, b)$ : trả về nghiệm của phương trình $Ax = b$

```
In [2]: import numpy as np
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        b=np.array([4,5,6])
        np.linalg.solve(A,b)

Out[2]: array([10.,  3., -0.])
```

# Hệ phương trình tuyến tính

## Nghiệm bình phương tối thiểu (least squares solution)

Trong một số trường hợp, hệ phương trình tuyến tính $Ax = b$ vô nghiệm. Ta muốn tìm một véc tơ x sao cho Ax "gần b nhất" có thể. Véc tơ x được gọi là nghiệm bình phương tối thiểu của hệ phương trình nếu

$$\|Ax - b\|^2 \to \min$$

Nghiệm bình phương tối thiểu thỏa mãn phương trình:

$$(A^T A)x = A^T b$$

Do đó, nếu $(A^T A)$ khả nghịch thì

$$x = (A^T A)^{-1} A^T b$$

# Hệ phương trình tuyến tính

- **np.linalg.lstsq**$(A,b)$ : trả về nghiệm bình phương tối thiểu của hệ phương trình $Ax = b$, hạng và các giá trị kì dị của ma trận $A$

```
In [5]: import numpy as np
        A=np.array([[1,-2,3,1],[2,-5,12,4],[0,2,-10,1]])
        b=np.array([4,5,6])
        np.linalg.lstsq(A,b,rcond=None)

Out[5]: (array([ 1.80053908, -2.46630728, -1.05121294,  0.42048518]),
         array([], dtype=float64),
         3,
         array([17.12877864,  3.88620317,  0.70877874]))
```

- Hàm **lstsq**() khi áp dụng để giải hệ phương trình có vô số nghiệm sẽ trả về kết quả là nghiệm bình phương tối thiểu có độ lớn nhỏ nhất.

# Thực hành 2

## Bài tập 2.1

Cho ma trận

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} 4 & -2 & 1 \\ 3 & 1 & -1 \\ -1 & 5 & 0 \end{bmatrix}$$

a) Chứng tỏ rằng $\mathrm{A}^3 = 0$. Tính $(\mathrm{I} - \mathrm{A})(\mathrm{I} + \mathrm{A} + \mathrm{A}^2)$.

b) Các ma trận $(A + B)^2$ và $A^2 + 2AB + B^2$ có bằng nhau không?

# Thực hành 2

## Bài tập 2.2

Cho hai ma trận $A$ và $B$ như sau:

$$A = \begin{pmatrix} 1 & 4 \\ -3 & 2 \\ -2 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} -4 & 1 & 9 \\ 2 & 0 & -3 \end{pmatrix}$$

a) Tạo ma trận khối: $C = \begin{bmatrix} I & -B \\ A & H \end{bmatrix}$, trong đó $I$ là ma trận đơn vị và $H$ là ma trận có các phần tử bằng 2.

b) Tính định thức của ma trận $C$ và so sánh với định thức của ma trận $(H + AB)$.

# Thực hành 2

## Bài tập 2.3

Giải hệ phương trình tuyến tính sau đây:

$$\begin{cases} x_1 - 2x_2 + 2x_3 - x_4 & = 3 \\ 2x_1 + 4x_2 - 5x_3 + x_4 & = -1 \\ -6x_1 - x_2 + 3x_3 + 4x_4 & = -3 \\ 4x_1 + 2x_2 + x_3 - 8x_4 & = 4 \end{cases}$$

# Thực hành 2

## Bài tập 2.4 (Nội suy)

Trong mặt phẳng toạ độ $(x, y)$ cho các điểm sau đây:

$$(-2, 3),\ (-1, -5),\ (2, -5),\ (3, -17)$$

Cho đa thức dưới dạng: $p = \alpha_0 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3$, trong đó

$$\begin{cases} p_1 = 1 + x \\ p_2 = x(x-2) \\ p_3 = x^3 - x^2 + 2x \end{cases}$$

Hãy tìm $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ sao cho đa thức $p$ đi qua tất cả các điểm dữ liệu được cho.